LNAI 5255

Jean-François Boulicaut
Michael R. Berthold
Tamás Horváth (Eds.)

Discovery Science

11th International Conference, DS 2008
Budapest, Hungary, October 2008
Proceedings

Springer

Lecture Notes in Artificial Intelligence 5255

Edited by R. Goebel, J. Siekmann, and W. Wahlster

Subseries of Lecture Notes in Computer Science

Jean-François Boulicaut
Michael R. Berthold Tamás Horváth (Eds.)

Discovery Science

11th International Conference, DS 2008
Budapest, Hungary, October 13-16, 2008
Proceedings

Springer

Series Editors

Randy Goebel, University of Alberta, Edmonton, Canada
Jörg Siekmann, University of Saarland, Saarbrücken, Germany
Wolfgang Wahlster, DFKI and University of Saarland, Saarbrücken, Germany

Volume Editors

Jean-François Boulicaut
University of Lyon, INSA Lyon, LIRIS CNRS UMR 5205
69621 Villeurbanne Cedex, France
E-mail: jean-francois.boulicaut@insa-lyon.fr

Michael R. Berthold
University of Konstanz, Department of Computer and Information Science
Box M 712, 78457 Konstanz, Germany
E-mail: michael.berthold@uni-Konstanz.de

Tamás Horváth
University of Bonn and Fraunhofer IAIS
Schloss Birlinghoven, 53754 Sankt Augustin, Germany
E-mail: tamas.horvath@iais.fraunhofer.de

Library of Congress Control Number: 2008936492

CR Subject Classification (1998): I.2, H.2.8, H.3, J.1, J.2

LNCS Sublibrary: SL 7 – Artificial Intelligence

ISSN 0302-9743
ISBN-10 3-540-88410-6 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-88410-1 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12537389 06/3180 5 4 3 2 1 0

Preface

It is our pleasure to present the proceedings of Discovery Science 2008, the 11th International Conference on Discovery Science held in Budapest, Hungary, October 13-16, 2008. It was co-located with ALT 2008, the 19th International Conference on Algorithmic Learning Theory, whose proceedings are available in the twin volume LNAI 5254. This combination of DS and ALT conferences has been successfully organized each year since 2002. It provides a forum for the researchers working on many different aspects of scientific discovery. Indeed, ALT/DS 2008 covered both the possibility to automate part of the scientific discovery and the necessary support to the human process of discovery in science. Interestingly, this co-location also provided the opportunity for an exciting joint program of tutorials and invited talks. The number of submitted papers was 58, i.e., slightly more than the previous year. The Program Committee members were involved in a rigorous selection process based on three reviews per paper. At the end, we selected 26 long papers thanks to the recommendations of the experts based on relevance, novelty, significance, technical quality, and clarity. Although some short papers were submitted, none of them was selected.

We wish to express our gratitude to:

– The authors of the submitted papers
– The Program Committee members and the additional referees for their contribution to the crucial selection process
– The prestigious invited speakers Imre Csiszár, Daniel A. Keim, László Lovász, Heikki Mannila, and Tom Mitchell
– Sašo Džeroski and João Gama, who accepted to give tutorials
– The members of the Discovery Science Steering Committee and especially its Chair Einoshin Suzuki
– Akira Ishino, Ayumi Shinohara, Eiji Takimoto, and Thomas Zeugmann for their support in preparing ALT/DS 2008
– The Local Organization Chair János Csirik (University of Szeged, Hungary) and Gusztáv Hencsey (SCOPE Meetings Ltd., Hungary) for local arrangements
– Richard van de Stadt (www.borbala.com) for his efficient support in the management of the whole submission and evaluation process
– Springer for co-operation in publishing the proceedings
– We gratefully acknowledge the financial support of Aegon Hungary, Fraunhofer IAIS (Sankt Augustin, Germany), INSA Lyon (France), the University of Konstanz (Germany), and the University of Szeged (Hungary).

July 2008

Jean-François Boulicaut
Michael R. Berthold
Tamás Horváth

Organization

Conference Chair

Tamás Horváth	University of Bonn and Fraunhofer IAIS, Germany

Program Committee

Jean-François Boulicaut (Co-chair)	INSA Lyon, France
Michael R. Berthold (Co-chair)	University of Konstanz, Germany
Daniel Berrar	University of Ulster, UK
Guillaume Beslon	IXXI Lyon, France
Jérémy Besson	MII Vilnius, Lithuania
Simon Colton	Imperial College London, UK
Vincent Corruble	Université Pierre et Marie Curie, France
Tapio Elomaa	Tampere University of Technology, Finland
Ingrid Fischer	University of Konstanz, Germany
Johannes Fürnkranz	Technical University Darmstadt, Germany
João Gama	University of Porto, Portugal
Ricard Gavalda	Universitat Politècnica de Catalunya, Spain
Lawrence Hall	University of South Florida, USA
Kouichi Hirata	Kyushu Institute of Technology, Japan
Tu Bao Ho	Advanced Institute of Science and Technology, Japan
Jaakko Hollmén	Helsinki University of Technology, Finland
Kristian Kersting	Massachusetts Institute of Technology, USA
Stefan Kramer	Technische Universität München, Germany
Nada Lavrač	Jozef Stefan Institute, Slovenia
Taneli Mielikäinen	Nokia Research Center, USA
Tetsuhiro Miyahara	Hiroshima City University, Japan
Dunja Mladenic	Jozef Stefan Institute, Slovenia
Shinichi Morishita	University of Tokyo, Japan
Bernhard Pfahringer	University of Waikato, New Zealand
Rudy Setiono	National University of Singapore, Singapore
Einoshin Suzuki	Kyushu University, Japan
Masayuki Takeda	Kyushu University, Japan
Ljupco Todorovski	University of Ljubljana, Slovenia
Luis Torgo	University of Porto, Portugal
Kuniaki Uehara	Kobe University, Japan

Takashi Washio	Osaka University, Japan
Gerhard Widmer	Johannes Kepler University Linz, Austria
Filip Železný	Czech Technical University in Prague, Czech Republic

Local Arrangements Chair

János Csirik	University of Szeged and RGAI Szeged, Hungary

Additional Referees

Hiroki Arimura
Hideo Bannai
Albert Bifet
Vineet Chaoji
John Charnley
Arjun Dasgupta
Jure Ferlez
Kohei Hatano
Corneliu Henegar
Tran Dang Hung
Shunsuke Inenaga
Aneta Ivanovska
Akihiko Izutani
Alipio Jorge
Saori Kawasaki
Dragi Kocev
Tetsuji Kuboyama
Marianne Mueller
Canh Hao Nguyen
Panče Panov
Nguyen Thanh Phuong
Ulrich Rückert
Raquel Sebastiao
Kazuhiro Seki
Kimiaki Shirahama
Larry Shoemaker
Pedro Torres
Ville Tuulos
Yoshiaki Yasumura

Table of Contents

On Iterative Algorithms with an Information Geometry Background

Imre Csiszár

Alfréd Rényi Institute of Mathematics, Hungarian Academy of Sciences, Budapest, Hungary
csiszar@renyi.hu

Abstract. Several extremum problems in Statistics and Artificial Intelligence, e.g., likelihood maximization, are often solved by iterative algorithms such as iterative scaling or the EM algorithm, admitting an intuitive "geometric" interpretatation as iterated projections in the sense of Kullback information divergence. Such iterative algorithms, including those using Bregman rather than Kullback divergences, will be surveyed. It will be hinted to that the celebrated belief propagation (or sum-product) algorithm may also admit a similar interpretation.

Visual Analytics: Combining Automated Discovery with Interactive Visualizations

Daniel A. Keim, Florian Mansmann, Daniela Oelke, and Hartmut Ziegler

University of Konstanz, Germany
`first.lastname@uni-konstanz.de`
`http://infovis.uni-konstanz.de`

Abstract. In numerous application areas fast growing data sets develop with ever higher complexity and dynamics. A central challenge is to filter the substantial information and to communicate it to humans in an appropriate way. Approaches, which work either on a purely analytical or on a purely visual level, do not sufficiently help due to the dynamics and complexity of the underlying processes or due to a situation with intelligent opponents. Only a combination of data analysis and visualization techniques make an effective access to the otherwise unmanageably complex data sets possible.

Visual analysis techniques extend the perceptual and cognitive abilities of humans with automatic data analysis techniques, and help to gain insights for optimizing and steering complicated processes. In the paper, we introduce the basic idea of Visual Analytics, explain how automated discovery and visual analysis methods can be combined, discuss the main challenges of Visual Analytics, and show that combining automatic and visual analysis is the only chance to capture the complex, changing characteristics of the data. To further explain the Visual Analytics process, we provide examples from the area of document analysis.

1 Introduction

The information overload is a well-known phenomenon of the information age, since our ability to collect and store data is increasing at a faster rate than our ability to analyze it. In numerous application areas fast growing data sets develop with ever higher complexity and dynamics. The analysis of these massive volumes of data is crucial in many application domains. For decision makers it is an essential task to rapidly extract relevant information from the immense volumes of data. Software tools help analysts to organize their information, generate overviews and explore the information in order to extract potentially useful information. Most of these data analysis systems still rely on visualization and interaction metaphors which have been developed more than a decade ago and it is questionable whether they are able to meet the demands of the ever-increasing masses of information. In fact, huge investments in time and money are often lost, because we lack the possibilities to make proper use of the available data. The basic idea of Visual Analytics is to visually represent the information, allowing the human to directly interact with the data to gain insight, draw conclusions, and ultimately make better decisions. The visual representation of the information reduces complex cognitive work needed to perform certain tasks. "People use visual analytics tools and techniques to synthesize information and derive insight from massive, dynamic, ambiguous, and often conflicting data ... to provide timely, defensible, and understandable assessments" [1].

The goal of Visual Analytics research is to turn the information overload into an opportunity. Decision-makers should be enabled to examine this massive information stream to take effective actions in real-time situations. For informed decisions, it is indispensable to include humans in the data analysis process and combine their flexibility, creativity, and background knowledge with the enormous storage capacity and the computational power of today's computers. The specific advantage of Visual Analytics is that decision makers may focus their full cognitive and perceptual attention on the decision, while allowing them to apply advanced computational methods to make the discovery process more effective.

The rest of this paper is structured as follows: Section 2 defines Visual Analytics, discusses related research areas, and presents a model of the Visual Analytics Process. In Section 3, we discuss the major technical challenges of the field. To foster a deeper understanding of Visual Analytics, Section 4 details examples of how visual and automatic methods can be used for an advanced interactive document analysis. Finally, Section 5 summarizes the key aspects of our paper.

2 Visual Analytics

In this section we will discuss Visual Analytics by defining it, by listing related research areas, and by presenting a model of the Visual Analytics Process.

2.1 Definition

According to [1], Visual Analytics is the science of analytical reasoning supported by interactive visual interfaces. Today, data is produced at an incredible rate and the ability to collect and store the data is increasing at a faster rate than the ability to analyze it. Over the last decades, a large number of automatic data analysis methods have been developed. However, the complex nature of many problems makes it indispensable to include human intelligence at an early stage in the data analysis process. Visual Analytics methods allow decision makers to combine their human flexibility, creativity, and background knowledge with the enormous storage and processing capacities of today's computers to gain insight into complex problems. Using advanced visual interfaces, humans may directly interact with the data analysis capabilities of today's computer, allowing them to make well-informed decisions in complex situations.

2.2 Related Research Areas

Visual Analytics can be seen as an integral approach combining visualization, human factors, and data analysis. Figure 1 illustrates the research areas related to Visual Analytics. Besides visualization and data analysis, especially human factors, including the areas of cognition and perception, play an important role in the communication between the human and the computer, as well as in the decision-making process. With respect to visualization, Visual Analytics relates to the areas of Information Visualization and Computer Graphics, and with respect to data analysis, it profits from methodologies developed in the fields of information retrieval, data management & knowledge representation as well as data mining.

2.3 The Visual Analytics Process

The Visual Analytics Process combines automatic and visual analysis methods with a tight coupling through human interaction in order to gain knowledge from data. Figure 2 shows an abstract overview of the different stages (represented through ovals) and their transitions (arrows) in the Visual Analytics Process.

In many application scenarios, heterogeneous data sources need to be integrated before visual or automatic analysis methods can be applied. Therefore, the first step is often to preprocess and transform the data to derive different representations for further exploration (as indicated by the *Transformation* arrow in Figure 2). Other typical preprocessing tasks include data cleaning, normalization, grouping, or integration of heterogeneous data sources.

After the transformation, the analyst may choose between applying visual or automatic analysis methods. If an automated analysis is used first, data mining methods are applied to generate models of the original data. Once a model is created the analyst has to evaluate and refine the models, which can best be done by interacting with the data. Visualizations allow the analysts to interact

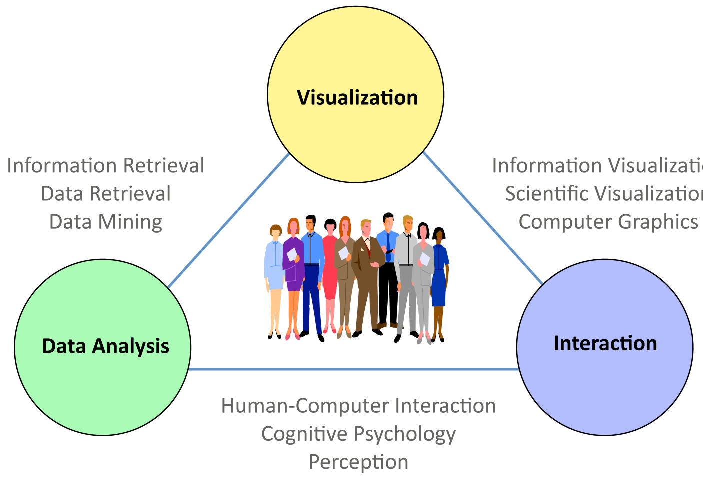


Fig. 1. Research Areas Related to Visual Analytics

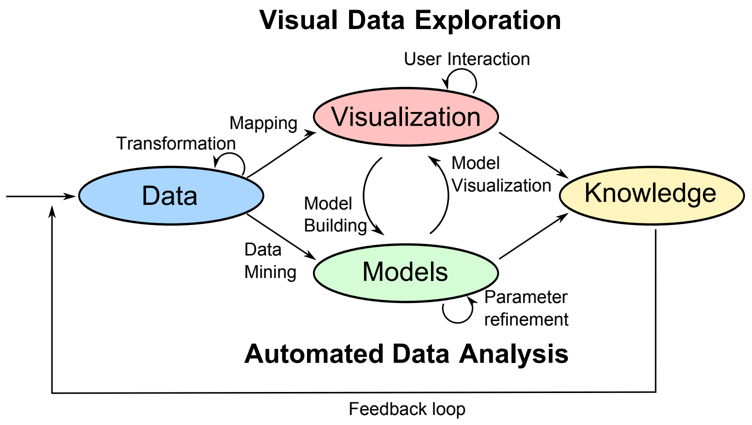


Fig. 2. The *Visual Analytics Process* is characterized through interaction between data, visualizations, models about the data, and the users in order to discover knowledge

with the automatic methods by modifying parameters or selecting other analysis algorithms. Model visualization can then be used to evaluate the findings of the generated models. Alternating between visual and automatic methods is characteristic for the Visual Analytics process and leads to a continuous refinement and verification of preliminary results. Misleading results in an intermediate step can thus be discovered at an early stage, leading to better results and a higher confidence. If a visual data exploration is performed first, the user has to confirm the generated hypotheses by an automated analysis. User interaction with the visualization is needed to reveal insightful information, for instance by zooming in on different data areas or by considering different visual views on the data. Findings in the visualizations can be used to steer model building in the automatic analysis. In summary, in the Visual Analytics Process knowledge can be gained from visualization, automatic analysis, as well as the preceding interactions between visualizations, models, and the human analysts.

The Visual Analytics Process aims at tightly coupling automated analysis methods and interactive visual representations. The classic way of visually exploring data as defined by the Information Seeking Mantra ("Overview first, Zoom/Filter, Details on demand") [2] therefore needs to be extended to the Visual Analytics Mantra [3]:

"Analyze First -
Show the Important -
Zoom, Filter, and Analyze Further -
Details on Demand"

With massive data sets at hand all three steps of the Information Seeking Mantra are difficult to implement. An overview visualization without losing interesting patterns is difficult to create, since the amount of pixels of the display does not keep pace with the increasing flood of data. In Visual Analytics, it is therefore not sufficient to just retrieve and display the data using a visual metaphor; it is rather necessary to analyze the data according to its value of interest, show the most relevant aspects of the data, and at the same time provide interaction models, which allow the user to get details of the data on demand.

3 Challenges of Visual Discovery

With information technology becoming a standard in most areas in the past years, more and more digital information is generated and collected. As the amount of data is continuously growing and the amount of pixels on the display remains rather constant, the huge amount of data to be visualized exceeds the limited amount of pixels of a display by several orders of magnitude. One key challenge of Visual Analytics is therefore *scalability* as it determines the ability to process large datasets in terms of computational overhead. In particular, since we are dealing with visualization techniques, the visual scalability of the techniques has to be considered, which is defined as the capability of visualization tools to effectively display large data sets in terms of either the number or the dimension of individual data elements [4]. While relying on increased hardware performance to cope with larger and larger problems, researchers need to design more effective Visual Analytics algorithms to bring this data onto the screen in an appropriate way.

Tremendous streams of time related or real time data are generated by dynamic processes, arising in business, networks, or telecommunications. Examples are sensor logs, web statistics, network traffic logs, or atmospheric and meteorological records. Analyzing these *data streams* is an important challenge, since the sheer amount of data does often not allow to record all data at full detail. This results in the need for effective compression and feature extraction to manage and access the data. Furthermore, real-time requirements put an additional burden upon the application developers. To enable quick identification of important information and timely reaction to critical process states or alarming incidents, analysis techniques and metaphors need to be developed, which render the user capable of analyzing real time data streams by presenting the results instantly in a meaningful and intuitive way.

To be capable of accessing information from a number of different sources, real-world applications require scalable methods for the *synthesis of heterogeneous types of data*. The heterogeneous data sources may include collections of vector data, strings, text documents, graphs, or multimedia objects. Integrating these data sources touches a number of fundamental problems in decision theory, information theory, statistics, and machine learning, evidently posing a challenge for Visual Analytics, too. The focus on scalable and robust methods for fusing complex heterogeneous data sources is thus key to a more effective analysis process. Computational biology is one such application domain where the human genome, for example, is accompanied by real-valued gene expression data, functional annotation of genes, genotyping information, a graph of interacting proteins, equations describing the dynamics of a system, localization of proteins in a cell, and natural language documents in the form of papers describing experiments or partial models.

Visual Analytics can also help to close the *Semantic Gap.* Since humans are the ultimate instance for defining semantics, Visual Analytics may significantly improve the way semantic definitions are obtained and refined. In particular, methods from semantics research may capture associations and complex relationships within the data sets to support decision-centered visualization. While ontology-driven techniques and systems have already started to enable new semantic applications in a wide span of areas, further research is necessary to increase our capabilities for creating and maintaining large domain ontologies and automatic extraction of semantic meta data, since the integration of different ontologies to link various datasets is hardly automated yet. Research challenges arise from the size of ontologies, content heterogeneity, and link analysis over ontology instances or meta data. New Visual Analytics methods to resolve semantic heterogeneity and discover complex relationships are thus needed.

Finally, *evaluation* as a systematic determination of merit, worth, and significance of a technique or system is essential to the success of Visual Analytics. Different aspects need to be considered when evaluating a system, such as functional testing, performance benchmarks, measurement of the effectiveness of the display, economic success, user studies, assessment of its impact on decision-making, etc. Note that not all of these aspects are orthogonal nor can they always be applied. Since Visual Analytics deals with unprecedented data sizes, many developed applications contain novel features to support a previously unsolvable analysis task. In such a case, the lack of a competing system turns a meaningful evaluation into a challenge in itself.

4 Example Application: Visual Document Analysis

Document Analysis is an area in which the need for visual analysis techniques is quite obvious. Large amounts of information are only available in textual form (e.g. books, newspapers, patents, service reports, etc.). But often these valuable resources are not used, because reading and analyzing the documents would take too much effort. Take for example a company's need to know the public opinion about one of its products and especially about rumors regarding that product. Knowing about such rumors is important to be able to quickly react to undesired developments and to effectively influence the public opinion in a favorable way. The Internet is a great place for understanding the public opinion since nowadays a significant percentage of the population participates in writing blogs, commenting on products at merchant sites, stating their opinions in forums, etc. And people read other people's comments to get information and form their opinion. With current search engines, however, it is not easy to find the relevant information related to the public opinion about a company's product, since search engines usually return millions of hits with only a small percentage being relevant to the task.

The example shows that it is impossible for a human to find and analyze all the relevant documents. On the other hand, an automatic semantic analysis of the documents is still infeasible today due to a) the impressive flexibility and complexity of natural language as well as b) the need to semantically interpret the content. The challenge that researchers try to tackle with Visual Analysis techniques is how to allow the human and computer to effectively work together to bridge the Semantic Gap.

Text can be analyzed on different abstraction levels:

- statistical level (e.g. frequencies of (specific) words, average sentence length, number of tokens or types, etc.)
- structural level (structural components of a document, such as header, footer, title, abstract, etc.)
- syntactical level (principles and rules for constructing sentences)
- semantic level (linguistic meaning)
- pragmatic level (meaning in context; consequence for actions)

The higher the abstraction level the more difficult it is for the computer to appropriately analyze a text. Counting words and characters as done at the statistical level is a simple task which can easily be performed by the computer. The identification of the structure of a document and the analysis of the syntax is already more challenging but can still be computationally approached (see e.g. the techniques presented in [5] [6]). Analyses on the semantic and pragmatic level are much more challenging. The idea of Visual Analytics is to let the human and the computer cooperate in solving the task. Humans contribute background knowledge, interpretation, and semantic analysis of the text whereas the computer supports the human analysts in the best possible way to enable them to deal with large data sets, e.g. by performing the time-consuming preprocessing and filtering steps.

4.1 Quasi-semantic Document Properties

The vision for automatic document analysis is to teach the computer to understand a document in a way similar to humans including its semantic and pragmatic aspects. Since this goal seems to be too ambitious at the current state of research, we start by teaching the computer to analyze a document with respect to one semantic aspect at a time. This task is relevant in many real application scenarios. Often large amounts of documents have to be analyzed with respect to a certain analysis question. Examples for such document analysis questions include:

- What is the public opinion regarding a product / a politician / a “hot” news topic, etc. that is expressed in news articles, blogs, discussion groups, etc. on the Internet?
- How trustworthy are the statements?

- How much emotion content is contained in the documents (e.g. hate in terrorist webpages)?

We call the document property that is central to the considered document analysis question a *quasi-semantic property*. We define quasi-semantic properties as higher-level document properties that capture one semantic aspect of a document (e.g. positive / negative statements with respect to a given product name). Most quasi-semantic properties cannot be measured directly. Nevertheless, combinations of low-level features (i.e. statistical, structural and syntactical features) can be used to approximate quasi-semantic properties of the documents, which help to bridge the semantic gap. The challenge is how to determine the best combination of low-level features to approximate such higher-level document properties.

4.2 Opinion Analysis

Figure 3 shows a set of documents that have been analyzed with respect to a quasi-semantic property that tries to assess the positive or negative opinion expressed in the documents. To automatically assess the polarity of a sentence we counted the number of opinion signal words. The signal words were given in form of two lists that contain adjectives, verbs and nouns (such as “bad”, “problem”, “wonderful”, etc.) that hint at a subjective statement and its polarity. The algorithm can easily be improved by taking context dependent signal words or negation into account (cf. [7] and [8]). For illustrative purposes, we use the title pages of the November 2007 issues of *The Telegraph* as text corpus. The figure shows that there are some articles that are completely negative (e.g. the article in the lower right corner) and others that are mostly positive (such as the articles about the Queen in the 1st column, 3rd row). Interestingly, there are also articles with quite mixed opinions or with a sudden change in polarity (for example, the first article in the last column, 1st row or the lower left article in the 4th column, 1st row). The example demonstrates that by combining automatic and visual methods it becomes possible to quickly analyze a document corpus with respect to a quasi-semantic property without reading it.

4.3 Authorship Attribution

Our second case study shows how Visual Analytics techniques can be used to analyze the discrimination power of low-level features that are commonly used in authorship attribution. Given some documents with known authorship the task of authorship attribution is to assign a document with unknown authorship to the author that has written it. Thus, in this case the quasi-semantic property that we would like to measure is the writing style of an author.

In previous work we focused on the development of techniques that support the analysis of low-level features and thus can be used to find (combinations of) low-level features that are able to approximate a desired quasi-semantic property. In fully automatic document analysis often just a single feature value is

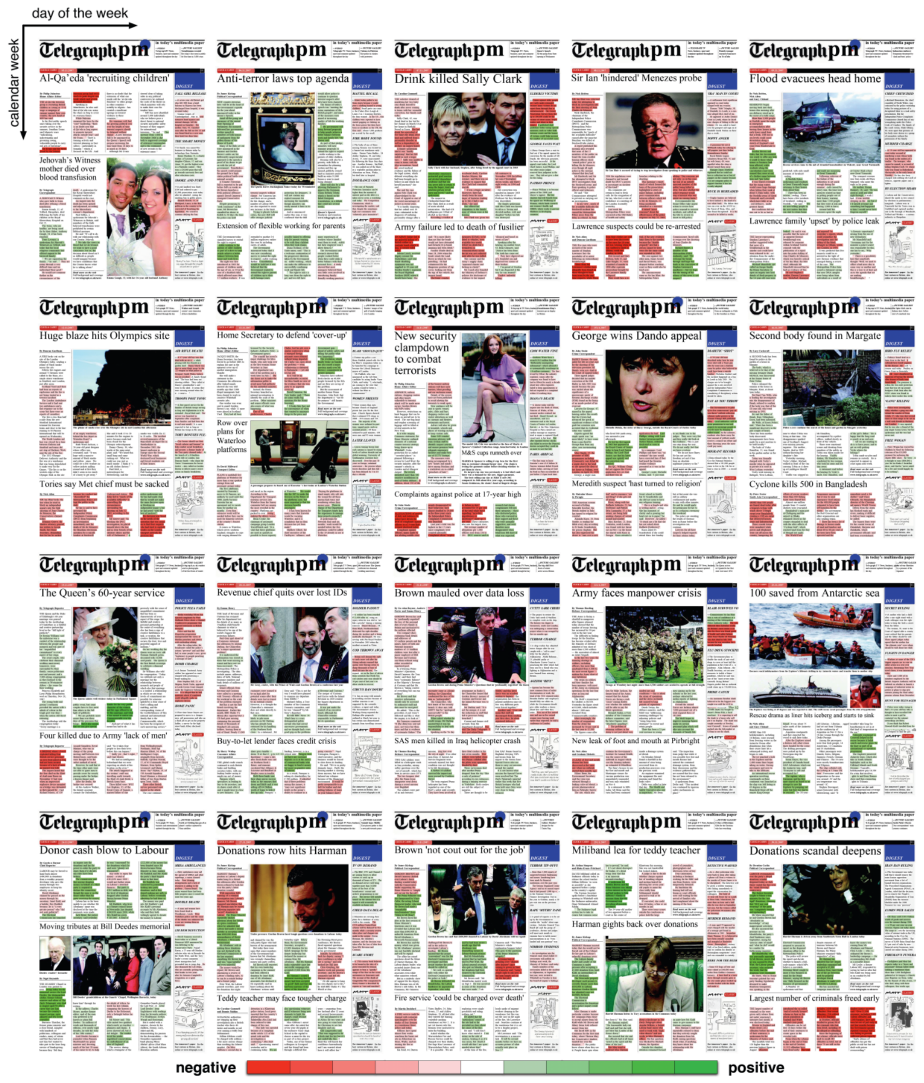
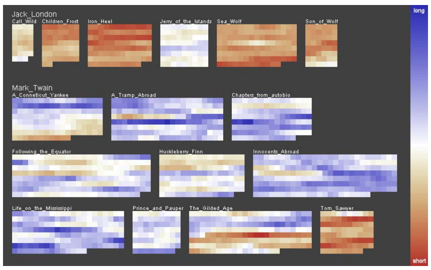
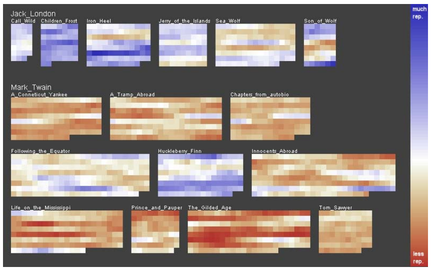

day of the week
calendar week
Al-Qa'eda 'recruiting children'
Anti-terror laws top agenda
Drink killed Sally Clark
Sir Ian 'hindered' Menezes probe
Flood evacuees head home
Huge blaze hits Olympics site
Home Secretary to defend 'cover-up'
New security clampdown to combat terrorists
George wins Dando appeal
Second body found in Margate
The Queen's 60-year service
Revenue chief quits over lost IDs
Brown mauled over data loss
Army faces manpower crisis
100 saved from Antarctic sea
Donor cash blow to Labour
Donations row hits Harman
Brown 'not cout out for the job'
Miliband lea for teddy teacher
Donations scandal deepens
negative
positive

Fig. 3. Title pages of *The Telegraph* in November 2007. The text has been analyzed with respect to a quasi-semantic property ‘that tries to assess the positive or negative opinion expressed in the documents. Sentences with positive statements are highlighted in green, the ones with negative statements in red, respectively. The degree of positiveness or negativeness is denoted by the intensity of the color. (courtesy of *The Telegraph*)

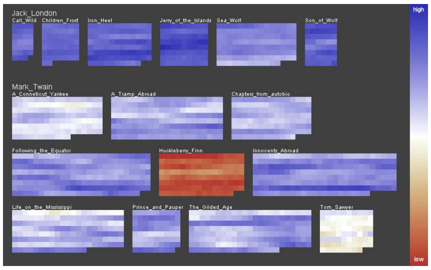
calculated per document. With the use of visualization techniques, it is possible to extract a sequence of feature values and present it to the user as a characteristic fingerprint for each document. By doing this it is possible to analyze the development of the values across the document in detail. Figure 4 shows our Literature Fingerprinting technique which was first presented in [9]. In Figure 4,



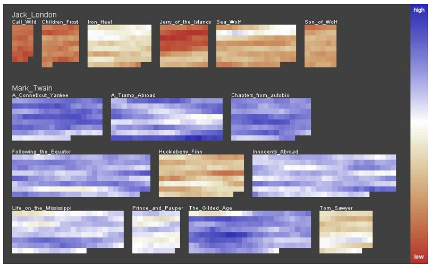
(a) Average sentence length



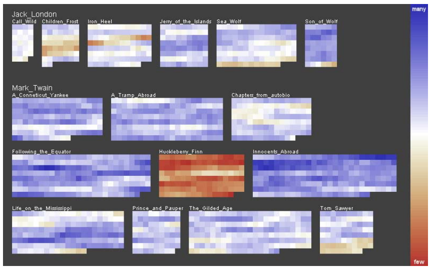
(b) Simpson's Index



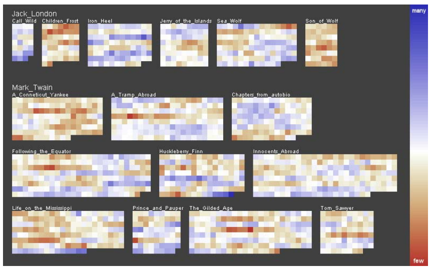
(c) Function words (First Dimension after PCA)



(d) Function words (Second Dimension after PCA)



(e) Hapax Legomena



(f) Hapax Dislegomena

Fig. 4. *Literature Fingerprinting Technique (see [9]).* Instead of calculating a single feature value per document, a sequence of feature values is extracted and presented to the user as a characteristic fingerprint for each document. In the example above, the technique is used to analyze the discrimination power of text features for authorship attribution. Each pixel represents the feature value for one text block and the grouped pixels belong to one book. The different feature values are mapped to color. If a feature is able to discriminate between the two authors, the books in the first row (that have been written by J. London) are visually different from the remaining books (written by M. Twain). Each subfigure shows the visualization of the values of one specific low-level feature that is commonly used for authorship attribution. It can easily be seen that not all features are able to discriminate between the two authors. Furthermore, it is interesting to observe that the book *Huckleberry Finn* (middle book in the middle column of the books of M. Twain) sticks out in a number of features as if it was not written by Mark Twain.

the technique is applied to several books of Jack London (first row of each subfigure) and Mark Twain (last three rows of each subfigure). Each pixel represents a text block of 10,000 words and the pixels are arranged from left to right and top to bottom as they appear in the sequence of the book. Neighboring blocks have an overlap of 9,000 words to obtain a continuous and split-point independent representation. Color is mapped to the feature value, ranging from blue for high feature values to red for low feature values. In this example the values of five different features have been calculated:

- the average sentence length
- the frequencies of specific function words; the resulting high-dimensional feature vectors are projected into low-dimensional space using a Principal Component Analysis and the first and second dimension are visualized in Figures 4(c) and 4(d).
- three vocabulary measures, namely Hapax Legomena Index, Hapax Dislegomena Index and Simpson's Index which are calculated as follows:
  Hapax Legomena Index (R):
  $$R = \frac{100 \log N}{1 - V_1/V}$$
  Hapax Dislegomena Index (D):
  $$D = \frac{V_2}{V}$$
  Simpson's Index (S):
  $$S = \frac{\sum_{r=1}^{\infty} r(r-1)V_r}{N(N-1)}$$
  where N = the number of tokens V = the number of types V_r = the number of lexical units that occur exactly r times

Please refer to [10] for an overview of the different features that are used for authorship attribution.

Each subfigure shows visualizations of all documents for one specific low-level feature. If the feature is able to discriminate between the two authors, the books in the first row (books by Jack London) have to be different from the ones in the last three rows (books by Mark Twain). It can easily be seen that there are some low-level features for which this is largely true, e.g. average sentence length in Figure 4(a) but also Simpson's Index in Figure 4(b). Others do not seem to have any discrimination power with respect to the two authors at all (e.g. Hapax Dislegomena which is depicted in Figure 4(f)). Interestingly, there is one book of Mark Twain that sticks out in many visualization, namely *The Adventures of Huckleberry Finn* (middle book in the middle row of the books by Mark Twain). The writing style of this book seems to be totally different from all the other books of Mark Twain.

This case study shows a small example of how Visual Analytics may help in better solving complex analysis tasks. The visual analysis enables the analyst to detect problems with the low-level feature used and adapt the similarity measures to make the authorship attribution more effective. While the example clearly shows the advantage of Visual Analytics it is only a first step toward a Visual Document Analysis system which tightly integrates automated document analysis and interactive document exploration capabilities.

5 Conclusions

Since data volumes are increasing at a faster pace than our ability to analyze them, there is a pressing need for automatic analysis methods. However, most automatic analysis methods require a well-defined problem and often return large and complex models. Visual Analytics turns the information overload problem into an opportunity by integrating interactive data exploration with advanced knowledge discovery algorithms.

In this paper, we motivate and define Visual Analytics, present a model of the Visual Analytics Process for a deeper understanding of how methods from visual data exploration and information mining can be combined to gain insights into large and complex datasets. The paper sketches the main challenges of Visual Analytics and describes why these challenges are difficult to solve. In particular, we give a demonstrative example of how Visual Analytics methods can help to gain insights in document analysis with an application to the authorship attribution problem.

Acknowledgement

We thank Jörn Schneidewind for for helpful comments on the manuscript.

References

1. Thomas, J., Cook, K.: Illuminating the Path: Research and Development Agenda for Visual Analytics. IEEE Press, Los Alamitos (2005)
2. Shneiderman, B.: The eyes have it: A task by data type taxonomy for information visualizations. In: IEEE Symposium on Visual Languages, pp. 336–343 (1996)
3. Keim, D.A., Mansmann, F., Schneidewind, J., Ziegler, H.: Challenges in visual data analysis. In: Information Visualization (IV 2006), London, United Kingdom, July 5-7. IEEE, Los Alamitos (2006)
4. Eick, S.G.: Visual scalability. Journal of Computational & Graphical Statistics (March 2002)
5. Klein, D., Manning, C.D.: Accurate unlexicalized parsing. In: ACL 2003: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics, pp. 423–430. Association for Computational Linguistics (2003), http://nlp.stanford.edu/software/lex-parser.shtml
6. Hadjar, K., Rigamonti, M., Lalanne, D., Ingold, R.: Xed: a new tool for extracting hidden structures from electronic documents. In: International Workshop on Document Image Analysis for Libraries, pp. 212–224 (2004)

7. Oelke, D., Bak, P., Keim, D., Last, M., Danon, G.: Visual evaluation of text features for document summarization and analysis. In: IEEE Symposium on Visual Analytics and Technology (VAST 2008) (to appear, 2008)
8. Ding, X., Liu, B., Yu, P.S.: A holistic lexicon-based approach to opinion mining. In: WSDM 2008: Proceedings of the international conference on Web search and web data mining, pp. 231–240. ACM, New York (2008)
9. Keim, D., Oelke, D.: Literature fingerprinting: A new method for visual literary analysis. In: IEEE Symposium on Visual Analytics and Technology (VAST 2007) (2007)
10. Holmes, D.I.: Authorship Attribution. Computers and the Humanities 28, 87–106 (1994)

Some Mathematics Behind Graph Property Testing

László Lovász

Department of Computer Science, Eötvös Loránd University, Budapest, Hungary
lovasz@cs.elte.hu

Abstract. Graph property testing is the third reincarnation of the same general question, after statistics and learning theory. In its simplest form, we have a huge graph (we don't even know its size), and we draw a sample of the node set of bounded size. What properties of the graph can be deduced from this sample?

The graph property testing model was first introduced by Goldreich, Goldwasser and Ron (but related questions were considered before). In the context of dense graphs, a very general result is due to Alon and Shapira, who proved that every hereditary graph property is testable.

Using the theory of graph limits, Lovász and Szegedy defined an analytic version of the (dense) graph property testing problem, which can be formulated as studying an unknown 2-variable symmetric function through sampling from its domain and studying the random graph obtained when using the function values as edge probabilities. This analytic version allows for simpler formulation of the problems, and leads to various characterizations of testable properties. These results can be applied to the original graph-theoretic property testing. In particular, they lead to a new combinatorial characterization of testable graph properties.

We survey these results, along with analogous results for graphs with bounded degree.

Finding Total and Partial Orders from Data for Seriation

Heikki Mannila

HIIT
Helsinki University of Technology and
University of Helsinki, Finland
heikki.mannila@tkk.fi

Abstract. Ordering and ranking items of different types (observations, web pages, etc.) are important tasks in various applications, such as query processing and scientific data mining. We consider different problems of inferring total or partial orders from data, with special emphasis on applications to the seriation problem in paleontology. Seriation can be viewed as the task of ordering rows of a 0-1 matrix so that certain conditions hold. We review different approaches to this task, including spectral ordering methods, techniques for finding partial orders, and probabilistic models using MCMC methods.

Joint work with Antti Ukkonen, Aris Gionis, Mikael Fortelius, Kai Puolamäki, and Jukka Jernvall.

1 Introduction

Ordering and ranking items of some type (observations, web pages, etc.) is one of the basic tasks in computing. In the traditional case of sorting items on the basis of a key field the result is a total order. More recent applications such as ranking of database query results [1,2,3,4,5,6,7] and web ranking [8,9,10,11,12] concentrate on finding a subset of relevant items and an ordering for those. Ordering tasks occur also in machine learning [13,14,15,16].

Here we look at cases where the ordering task is more hard to define. For simplicity we consider 0-1 matrices. Given a 0-1 matrix M with n rows and m columns, what is a good ordering of the rows? Without any background information there is no way of preferring one order of the rows to others. However, if there is some knowledge about the process that produces the data, we can state that one order is more likely than another.

Our motivating application comes from paleontology, where the 0-1 matrix has as rows fossil sites and as columns taxa (typically species or genera). Each site represents a collection of taxa that lived in the same area at approximately the same time. The seriation task is to order the rows of this matrix into an order that reflects the age of the fossil sites. The background information that can be used in this task is that the occurrences of a taxon should be more or less consecutive.

The study of paleontological seriation leads to interesting theoretical and practical problems. In this paper we review some of the work on finding good total or partial orders [17,18,19,20,21,22] that stems from the paleontological applications. While the description is mostly in terms of the application, most of the methods are quite general.

The work has been done with Antti Ukkonen, Aris Gionis, Mikael Fortelius, Kai Puolamäki and Jukka Jernvall.

2 Seriation Problem in Paleontology

A *fossil site* can be defined as a collection of fossil remains collected from some location, typically in a sedimentary deposit. A site represents a subset of the taxa (typically species or genera) that lived at a certain location at approximately the same time. Sites and their taxa are naturally represented by an occurrence matrix, i.e., a 0-1 matrix M, where the rows correspond to sites and the columns correspond to taxa. The entry $M(i, j)$ indicates that taxon j has been found at site i.

Seriation is the task of temporal ordering of fossil sites, and it is a fundamental problem in paleontology. Conventional paleontological seriation (biostratigraphy) is based on the use of stratigraphic superposition information, which typically is not available in large datasets compiled from a wide variety of sources. (See [18] and references therein.) Radioisotope and other methods can be used to obtain fairly accurate ages for sites, but typically these methods can only be used for a subset of the localities.

Starting from a 0-1 matrix M, what is the background information that can be used to select an ordering π for the rows of M? A very simple form of such information is that the occurrences (the 1s) of a taxon do not occur randomly in the column. Rather, first a taxon does not exist, then it becomes extant, and then becomes extinct (dies out). That is, in a good ordering of the rows the ones for each taxon should be consecutive. As an example, consider the two orderings of the rows of the same matrix:

$$\begin{array}{ccc} 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{array} \qquad\qquad \begin{array}{ccc} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \end{array}$$

On the left, the first and third taxon show a pattern $1 \cdots 0 \cdots 1$ of presences and absences: there is a 0 between two 1s. Such 0s are known as *Lazarus events*. The left matrix has four Lazarus events, but on the right there are none.

Given an ordering $\pi = (v_1, v_2, \ldots, v_n)$ of the rows of M, we refer by v_{ia} to the value in column a of row v_i. The *Lazarus count* $L(M, \pi)$ of M with respect to π is

$$L(M,\pi) = \big|\{\ (i,a) \mid v_{ia} = 0 \\ \wedge \exists j : v_{ja} = 1 \wedge j < i \\ \wedge \exists k : v_{ka} = 1 \wedge i < k\}\big|.$$

That is, $L(M,\pi)$ is the number of zeros in the data that are between the first and last ones in their column.

Finding if there is an ordering π such that $L(M,\pi) = 0$ is the problem of determining whether a 0-1 matrix has the *consecutive ones property*, i.e., whether there is an ordering such that all 1s are consecutive in all columns. This property can be tested in linear time [23,24].

However, in real data there is a lot of noise: false positives (false ones) and false negatives (false zeros). Especially false zeros (cases where a taxon is not observed when it was actually present) are abundant: their number can be about as high as the number of true ones. That is, $L(M,\pi) > 0$ for all total orders π. A number of methods (see, e.g., [25,26,20]) have been proposed for finding a good ordering.

If fossil data is available from different layers in the same geographic location, i.e., we can have stratigraphic information indicating that certain sites must be in a specific order. Furthermore, when geochronologic data about the ages of the fossils is available, it also generates an ordering for a subset of the sites.

3 General Formulation

The problem of finding a good ordering of rows of paleontological data matrices can be viewed as an instance of a more general problem. Namely, given a taxon a and three rows u, v, and w such that $M(u,a) = M(w,a) = 1$ but $M(v,a) = 0$, we know that the ordering in which $u < v < w$ or $u > v > w$ will generate a Lazarus event in the data. Thus all such triples can be viewed as constraints that the ordering should aim to avoid. Stratigraphic information and geochronologic ages of the sites, on the other hand, give constraints that should be satisfied: certain sites should be in a given order.

Given a set U of items to be ordered, a *constraint* α consists of a sequence of elements from U, i.e., $\alpha = (u_1, u_2, \ldots, u_k)$ for some $k > 1$, where the u_i's are distinct elements from U.

An ordering π of the elements of U *agrees* with α, if π orders the elements in C in the order given by π. Given a set C of constraints, denote by $\Gamma(\pi, C)$ the set of constraints from C that agree with π.

Given two sets of constraints P (positive constraints) and N (negative constraints), the *constrained ordering* problem is to find an ordering π of the elements of U that agrees with as many positive constraints and with as few negative constraints as possible. For example, we could search for the ordering π minimizing

$$|P \setminus \Gamma(\pi,P)| + |\Gamma(\pi,N)|,$$

or some other function of the sets $\Gamma(\pi,P)$ and $\Gamma(\pi,N)$.

In the paleontological case the positive constraints are typically hard constraints: we would like all of them to be satisfied. Thus, the goal in that application would be to obtain an ordering π such that $\Gamma(\pi, P) = P$ and $\Gamma(\pi, N)$ is as small as possible.

As another example, in the feedback arc set problem the input is a tournament on a set of edges, i.e., a directed graph $G = (V, E)$ where for each $u, v \in V$ either $(u, v) \in E$ or $(v, u) \in E$, and the task is to find an ordering $\pi = (u_1, \ldots, u_n)$ of V such that the number of pairs (i, j) with $i < j$ and $(u_j, u_i) \in E$ is minimized. Thus the feedback arc set problem is an instance of the constrained ordering problem with only positive constraints and with the goal of minimizing the size of $P \setminus \Gamma(\pi, P)$.

4 Spectral Approaches

Spectral methods are based on computing eigenvalues and eigenvectors of certain matrices, and then using these for various data analysis tasks; these approaches have turned out to be very efficient for many problems (see, e.g., [27,28]).

A simple way of using spectral methods for ordering items is as follows. Given a similarity matrix S between sites. Consider the Laplacian matrix H, with entries $H(i, j) = -S(i, j)$ for $i \neq j$ and $H(i, i) = \sum_{j;j \neq i} S(i, j)$. Then H is a symmetric matrix and its rows sum to 0; thus the smallest eigenvalue is 0. The eigenvector x of corresponding to second smallest eigenvalue of H is the vector minimizing

$$\sum_{i,j} S(i, j)(x_i - x_j)^2$$

subject to $\sum_i x_i = 0$ and $\sum_i x_i^2 = 1$. See [29,26] for more details. That is, the vector v is a way of embedding the entries into one dimension in a way that minimizes the stress $(x_i - x_j)^2$ weighted by the similarity matrix S. For seriation, a simple way of using a spectral approach is to use as the similarity matrix S the product MM^T or some scaled version of it.

According to Hill [30], the history of spectral methods for discrete data and ordering goes back to at least to the work of Hirschfeld and Fisher in the 1930s, under the name correspondence analysis. See also [31] for some early use of the method.

Such a spectral method works quite well for paleontological applications and other seriation tasks [20,31]. In [20] we studied the behavior of the simple spectral method for seriation of Neogene mammal fossil sites. In this database [32] the true geochronologic ages of some of the sites are known from radiometric measurements. For such sites the correlation between the true age and the coefficient of the eigenvector is 0.97 – 0.99, indicating a very high degree of agreement. The correlation between the eigenvector and the age estimated by using traditional methods (so-called MN classification) is 0.94 – 0.97. Note that the MN classification uses all available information, while the spectral method only uses the matrix of presences and absences. In terms of the number of Lazarus events, it seems quite difficult to improve in any significant way from the spectral results; for example, local search approaches did not yield any clear improvements compared to the spectral results.

A small drawback in the method is that if the similarity matrix has several disconnected components (e.g., if there are sites that contain only taxa not seen elsewhere), the eigenvector will assign most of the weight to the sites in one of the components and thus the method will not work. Therefore in practice one has to augment the method with additional steps that handle such degenerate cases. Also, the method cannot find the arrow of time: if x is a solution for the above task, also $-x$ is.

An interesting question is the following: why does the spectral method work well in the seriation task? After all, the function that the seriation method minimizes is not directly connected to the number of Lazarus events in the matrix.

Atkins et al. [26] show that the spectral approach does find the optimum in the case when there is an ordering with the consecutive ones property. What happens in the case when there is an ordering that almost has this property? Could one prove that the spectral method does indeed find the ordering?

Question 1. Why does the spectral method work well in the seriation task? Is there a way of improving its performance?

The spectral method, as described above, is not able to handle positive constraints, such as stratigraphic information. In practice this would be an important requirement.

Question 2. Is there a way of extending the spectral method so that also positive constraints can be taken into account?

Such an extension might already exist in the optimization literature. Given a constraint such as that sites i', j', and k' have to be in this order, what we would like to do is minimize

$$\sum_{i,j} S(i,j)(x_i - x_j)^2$$

subject to

$$\sum_i x_i = 0 \text{ and } \sum_i x_i^2 = 1 \text{ and } x_{i'} < x_{j'} < x_{k'}.$$

5 Finding Partial Orders

In the preceding section we considered the problem of finding a single total order that minimizes a certain function. However, it can be the case that there are several orderings that are equally good with respect to, say, the Lazarus count. The available data may be insufficient to yield a precise order on the rows. As an example, consider the matrix below on the left, having one Lazarus event. No permutation would yield a completely error-free ordering; thus this matrix does not have the consecutive ones property. But there are two other permutations of the three middle rows that result in one Lazarus event as well:

$$\begin{matrix} 1&0&0 \\ 1&1&0 \\ 1&\mathbf{0}&1 \\ 0&1&1 \\ 0&0&1 \end{matrix} \quad \text{and} \quad \begin{matrix} 1&0&0 \\ 1&1&0 \\ \mathbf{0}&1&1 \\ 1&0&1 \\ 0&0&1 \end{matrix} \quad \text{and} \quad \begin{matrix} 1&0&0 \\ 1&0&1 \\ 1&1&\mathbf{0} \\ 0&1&1 \\ 0&0&1 \end{matrix}$$

Every other permutation has a higher Lazarus count. Consider the indices (1 2 3 4 5) of the rows of the matrix on the left. Then the second one is given by (1 2 4 3 5) and the third one by (1 3 2 4 5). Forcing a total order on the rows of the above matrix by minimizing the Lazarus count is thus bound to contain some randomness, since all three orders are in fact equally good. For example, spectral ordering gives (1 2 3 4 5) as a solution. There is no reason to prefer this over the two others if the only criterion is number of Lazarus events.

Thus it makes sense to search for partial orders instead of total orders [33]. Consider a partial order $\preccurlyeq$ among the row indices a 0-1 matrix M. The number of Lazarus events in M with respect to $\preccurlyeq$ can be defined in exactly the same way as for the case of a total order: how many 0s there are in the data which are between two 1s in (this time) the partial order:

$$\begin{aligned} L(M, \preccurlyeq) = \big|\{\ (i,a) \mid v_{ia} = 0 \\ \wedge \exists j : v_{ja} = 1 \wedge j \preccurlyeq i \\ \wedge \exists k : v_{ka} = 1 \wedge i \preccurlyeq k\}\big|. \end{aligned}$$

Of course, a trivial way of minimizing this quantity is to take $\preccurlyeq$ to be the trivial partial order, where $v \preccurlyeq w$ holds only if $v = w$. Then there are no Lazarus events, but the resulting partial order is not informative.

Thus, a possible problem definition would be to search for a partial order $\preccurlyeq$ minimizing the quantity

$$L(M, \preccurlyeq) + \alpha(\preccurlyeq)$$

where $\alpha(\preccurlyeq)$ is some function of $\preccurlyeq$ that is small for total orders and becomes larger as $\preccurlyeq$ approaches the trivial partial order. (The approach used in [33] was not based on an explicit function α; the algorithm described there is a heuristic local search method.)

There are different possible choices for the function α. Perhaps the most natural one would be to use as $\alpha(\preccurlyeq)$ the logarithm of the number of linear extensions of $\preccurlyeq$, indicating how many bits in addition to $\preccurlyeq$ we need to transmit to give a total order. Computing the number of linear extensions is ♯P-complete, and hence using this definition is not without problems. Simpler alternatives would be, say, the size of $\preccurlyeq$, which for n rows varies between n for the trivial partial order and $n(n+1)/2$ for a total order.

Arbitrary partial orders are complex objects. Our experience in [33] was that the resulting partial orders were considered highly interesting by the paleontologists, but robustness of the results was sometimes a problem. Likewise, searching over arbitrary partial orders seems a hard task. Thus it probably makes sense to look for subclasses of partial orders with more tractable properties.

Question 3. What are good subclasses of partial orders for the seriation problem? The partial orders in the subclass should be easy to understand, sufficiently versatile to model the natural situations, and their algorithmic properties should be simple.

An interesting subclass is formed by bucket orders, considered in the next section.

6 Finding Bucket Orders

A simple class of partial orders is the class of series-parallel partial orders; the problem of finding them was considered in [34]. An even simpler class is the set of bucket orders, i.e., total orders with ties. That is, a bucket order $<_B$ on a set U of items is an ordered partition $U_1, U_2, \ldots, U_k$ of U; for $u, v \in U$ we have $u <_B v$ if and only if $u \in U_i$ and $v \in U_j$ with $i < j$ (see, e.g., [35,3]). A bucket order is a series-parallel partial order, and the number of linear extensions of a bucket order is easy to compute.

In [19] we studied the problem of finding bucket orders. As input we used a pair order matrix for the items to be ordered, i.e., a nonnegative matrix T with entry $T(u, v)$ indicating how likely row u is to precede row v. We assume that $T(u, v) + T(v, u) = 1$ for all items u and v.

Note that a bucket order $<_B$ also immediately defines a pair order matrix T_B by $T_B(u, v) = 0.5$ if u and v are in the same bucket, and otherwise $T_B(u, v) = 1$ or 0.

Given a pair order matrix T, the task is to find a bucket order that best describes it. This task is an instance of the general framework above, with the modification that the constraints have weights.

Based on the insightful pivot algorithm of Ailon et al. [36] for the feedback arc set problem, we gave a randomized *bucket pivot* algorithm that works as follows [19,21]. (See, e.g., [37,38] for additional work inspired by Ailon's idea.) Given the pair order matrix for items U, choose an element u at random. Let $\beta > 0$ be a constant such as 1/4. Divide U into three sets:

$$\begin{aligned} V &= \{v \in U \mid 0.5 - \beta \leq T(u, v) < 0.5 + \beta\} \\ U_1 &= \{v \in U \mid 0 \leq T(u, v) < 0.5 - \beta\} \\ U_2 &= \{v \in U \mid 0.5 + \beta \leq T(u, v) \leq 1\} \end{aligned}$$

Then V forms one of the buckets of the answer, and the others are obtained by calling the algorithm recursively on U_1 and U_2.

The bucket pivot algorithm achieves on expectation a constant approximation ratio for the NP-complete task of describing the given pair order matrix. Also in practice the performance of the algorithm is very good.

The bucket pivot algorithm requires as input the pair order matrix. This information is not immediately available in the seriation task, and it turns out to be relatively cumbersome to obtain the pair order matrix. This is a pity, as the simplicity of the bucket pivot algorithm would make it very useful.

Question 4. Can the pivot idea be used directly on paleontological data?

7 Probabilistic Models

The previous approaches have been combinatorial in nature. A natural alternative is to consider probabilistic models. In [18] we described a straightforward probabilistic model for the seriation task. The parameters that are used to describe the data are the origination and extinction time for each taxon, the ordering of the sites, and for the probabilities of errors (wrong zeros and wrong ones). When the ordering of the sites is given, the origination and extinction parameters describe the interval in which the taxon is assumed to be present. An occurrence of the taxon outside this interval is considered to be an error, as is any nonoccurrence inside this interval. Given the parameters, the likelihood of the data depends on the number of false and true ones and zeros. The task we consider is to find parameter vectors that yield high likelihood, i.e., have a small number of false ones and zeros.

The model can easily be augmented to incorporate stratigraphic or geochronologic information by requiring that the orderings satisfy the given hard constraints.

We used Markov chain Monte Carlo (MCMC) methods to find a posterior distribution for the parameters of the model. The design of suitable update operations for the parameters is challenging, as the search space is large; convergence of the MCMC method is also an issue. Applied to the NOW database data [32] the method shows how the ordering of the sites is in some cases were clearly determined, while there are also blocks of sites within which the order of the sites is more or less unspecified.

8 Concluding Remarks

We have described briefly some of the approaches that can be used to obtain total and partial orders from 0-1 data. The main motivating application is the seriation problem in paleontology. The methods depend on the application via the requirement of consecutive ones, i.e., that each taxon occurs and vanishes exactly once. Similar types of requirements are present also in other applications such as information propagation [39], where nodes receive a pieces of information, each from a single source. Document data provides also a challenging application area for the discovery of total and partial orders.

Acknowledgments

I would like to thank my co-authors in this work, Antti Ukkonen, Aris Gionis, Mikael Fortelius, Kai Puolamäki and Jukka Jernvall. I also thank Evimaria Terzi, Gemma Garriga, and Esa Junttila for useful discussions.

References

1. Agrawal, S., Chaudhuri, S., Das, G., Gionis, A.: Automated ranking of database query results. In: CIDR (2003)
2. Chaudhuri, S., Das, G., Hristidis, V., Weikum, G.: Probabilistic ranking of database query results. In: Proceedings of the 30th International Conference on Very Large Data Bases (VLDB) (2004)
3. Fagin, R., Kumar, R., Mahdian, M., Sivakumar, D., Vee, E.: Comparing and aggregating rankings with ties. In: Proceedings of the 23rd ACM Symposium on Principles of Database Systems (PODS) (2004)
4. Fagin, R., Kumar, R., Sivakumar, D.: Comparing top *k* lists. In: Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA) (2003)
5. Fagin, R., Kumar, R., Sivakumar, D.: Efficient similarity search and classification via rank aggregation. In: Proceedings of the ACM Conference on Management of Data (SIGMOD) (2003)
6. Ilyas, I.F., Shah, R., Aref, W.G., Vitter, J.S., Elmagarmid, A.K.: Rank-aware query optimization. In: Proceedings of the ACM Conference on Management of Data (SIGMOD) (2004)
7. Li, C., Chang, K., Ilyas, I., Song, S.: Query algebra and optimization for relational top-k queries. In: Proceedings of the ACM Conference on Management of Data (SIGMOD) (2005)
8. Borodin, A., Roberts, G.O., Rosenthal, J.S., Tsaparas, P.: Link analysis ranking: Algorithms, theory, and experiments. ACM Transactions on Internet Technology 5(1) (2005)
9. Brin, S., Page, L.: The anatomy of a large-scale hypertextual Web search engine. Computer Networks and ISDN Systems 30(1–7), 107–117 (1998)
10. Dwork, C., Kumar, R., Naor, M., Sivakumar, D.: Rank aggregation methods for the web. In: Proceedings of the 10th International World Wide Web Conference (WWW) (2001)
11. Haveliwala, T.: Topic-sensitive pagerank. In: Proceedings of the 11th International World Wide Web Conference (WWW) (2002)
12. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. Journal of the ACM 46(5) (1999)
13. Cohen, W.W., Schapire, R.E., Singer, Y.: Learning to order things. Journal of Artificial Intelligence Research 10, 243–270 (1999)
14. Crammer, K., Singer, Y.: Pranking with ranking. In: Conference on Neural Information Processing Systems (NIPS) (2001)
15. Fürnkranz, J., Hüllermeier, E.: Pairwise preference learning and ranking. In: Lavrač, N., Gamberger, D., Todorovski, L., Blockeel, H. (eds.) ECML 2003. LNCS (LNAI), vol. 2837. Springer, Heidelberg (2003)
16. Lebanon, G., Lafferty, J.D.: Cranking: Combining rankings using conditional probability models on permutations. In: ICML (2002)
17. Gionis, A., Kujala, T., Mannila, H.: Fragments of order. In: KDD 2003 (2003)
18. Puolamäki, K., Fortelius, M., Mannila, H.: Seriation in paleontological data using Markov Chain Monte Carlo methods. PLoS Computational Biology 2(2) (February 2006)
19. Gionis, A., Mannila, H., Puolamaki, K., Ukkonen, A.: Algorithms for discovering bucket orders from data. In: KDD (2006)
20. Fortelius, M., Gionis, A., Jernvall, J., Mannila, H.: Spectral ordering and biochronology of european fossil mammals. Paleobiology 32(2), 206–214 (2006)

21. Ukkonen, A.: Algorithms for Finding Orders and Analyzing Sets of Chains. PhD thesis, Helsinki University of Technology (2008)
22. Ukkonen, A., Mannila, H.: Finding outlying items in sets of partial rankings. In: Kok, J.N., Koronacki, J., López de Mántaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) PKDD 2007. LNCS (LNAI), vol. 4702. Springer, Heidelberg (2007)
23. Booth, K.S., Lueker, G.S.: Testing for the consecutive ones property, interval graphs, and graph planarity using P-Q tree algorithms. J. of Comp. and Syst. Sci. 13, 335–379 (1976)
24. Hsu, W.L.: A simple test for the consecutive ones property. Journal of Algorithms 43 (2002)
25. Brower, J., Kile, K.: Seriation of an original data matrix as applied to palaeoecology. Lethaia 21, 79–93 (1988)
26. Atkins, J.E., Boman, E.G., Hendrickson, B.: A spectral algorithm for seriation and the consecutive ones problem. SIAM Journal on Computing 28(1), 297–310 (1999)
27. Ng, A., Jordan, M., Weiss, Y.: On spectral clustering: Analysis and an algorithm. In: Advances in Neural Information Processing Systems (2001)
28. Azar, Y., Fiat, A., Karlin, A.R., McSherry, F., Saia, J.: Spectral analysis of data. In: ACM Symposium on Theory of Computing (2000)
29. Chung, F.R.K.: Spectral Graph Theory. CBMS Regional Conference Series in Mathematics (1997)
30. Hill, M.: Correspondence analysis: A neglected multivariate method. Applied Statistics 23, 340–354 (1974)
31. Kendall, D.G.: Abundance matrices and seriation in archaeology. Z. Wahscheinlichkeitstheorie verw. Geb. 17, 104–112 (1971)
32. Fortelius, M.: Neogene of the old world database of fossil mammals (NOW) (2006), http://www.helsinki.fi/science/now/
33. Ukkonen, A., Fortelius, M., Mannila, H.: Finding partial orders from unordered 0-1 data. In: Proceedings of the 11th ACM Conference on Knowledge Discovery and Data Mining (KDD) (2005)
34. Mannila, H., Meek, C.: Global partial orders from sequential data. In: KDD (2000)
35. Wilf, H.S.: Generatingfunctionology. Academic Press, London (1994), http://www.math.upenn.edu/~wilf/DownldGF.html
36. Ailon, N., Charikar, M., Newman, A.: Aggregating inconsistent information: ranking and clustering. In: Proceedings of the 37th ACM Symposium on Theory of Computing (STOC) (2005)
37. Coppersmith, D., Fleischer, L., Rudra, A.: Ordering by weighted number of wins gives a good ranking for weighted tournaments. In: Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 776–782 (2006)
38. van Zuylen, A., Hegde, R., Jain, K., Williamson, D.P.: Deterministic pivoting algorithms for constrained ranking and clustering problems. In: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, pp. 405–414 (2007)
39. Kempe, D., Kleinberg, J.M., Tardos, E.: Maximizing the spread of influence through a social network. In: KDD, pp. 137–146 (2003)

Computational Models of Neural Representations in the Human Brain

Tom M. Mitchell

Carnegie Mellon University, Pittsburgh, PA 15213, USA
tom.mitchell@cmu.edu
http://www.cs.cmu.edu/ tom

Extended Abstract

For many centuries scientists have wondered how the human brain represents thoughts in terms of the underlying biology of neural activity. Philosophers, linguists, cognitive scientists and others have proposed theories, for example suggesting that the brain organizes conceptual information in hierarchies of concepts, or that it instead represents different concepts in different local regions of the cortex.

Over the past decade rapid progress has been made on the study of human brain function, driven by the advent of modern brain imaging methods such as functional Magnetic Resonance Imaging (fMRI), which is able to produce three dimensional images of brain activity at a spatial resolution of approximately one millimeter. Using fMRI we have spent several years exploring the question of how the brain represents the meanings of invididual words in terms of patterns of neural activity observed with fMRI. The talk accompanying this abstract will present our results, and the use of machine learning methods to analyze this data and to develop predictive computational models. In particular, we ([1],[2]) have explored the following questions:

– *Can one observe differences in neural activity using fMRI, as people think about different items such as "hammer" versus "house"?* Many researchers have now demonstrated that fMRI does indeed reveal differences in neural activity due to considering different items. We present results [1] showing that it is possible to train a machine learning classifier to discover the different patterns of activity associated with different items, and to use this to classify which of several items a person is considering, based on their neural activity.
– *Are neural representations of concepts similar if the stimulus is a word, versus a line drawing of the object?* We tested this question by asking whether a machine learning classifier trained on fMRI data collected when a person reads words, could successfully distinguish which item they were thinking about when the stimuli were line drawings. The classifier performed nearly as accurately classifying fMRI activity generated by line drawing stimuli as by word stimuli, despite being trained on word stimuli. This result suggests that the neural activity captured by the classifier reflects the semantics of

the item, and not simply some surface perceptual features associated with the particular form of stimulus.

– *Are neural representations similar across different people?* We tested this question by asking whether a machine learning classifier trained on fMRI data collected from a group of people, could successfully distinguish which item a new person was thinking about, despite the fact that the classifier had never seen data from this new person. These experiments were performed for stimuli corresponding to concrete nouns (i.e., nouns such as "bicycle" and "tomato" which describe physical objects). We found the answer is yes, although accuracies vary by person. This result suggests that despite the fact that invididual people are clearly different, our brains use similar neural encodings of semantics of concrete nouns.
– *Can we discover underlying principles of neural representations sufficient to develop a computational model that predicts neural representations for arbitrary words?* We recently developed a computational model that predicts the neural representation for any concrete noun. While imperfect, this model performs well on the 100 words for which we have data to test it. The model is trained using a combination of fMRI data for dozens of words, plus data from a trillion word text corpus that reflects the way in which people typically use words in natural language. This model represents a new approach to computational studies of neural representations in the human brain.

Acknowledgments. The research summarized here is the product of a large team of collaborators, including Marcel Just, Andy Carlson, Kai-Min Chang, Rebecca Hutchinson, Vicente Maleve, Rob Mason, Mark Palatucci, Francisco Pereira, Indra Rustandi, Svetlana Shinkareva, Wei Wang and others. We are grateful for support from the W.M. Keck foundation and the National Science Foundation.

References

[1] Shinkareva, S., Mason, R., Malave, V., Wang, W., Mitchell, T., Just, M.: Using fMRI Brain Activation to Identify Cognitive States Associated with Perception of Tools and Dwellings. PLoS ONE 3(1) (2008); e1394. doi:10.1371/journal.pone.0001394
[2] Mitchell, T., Shinkareva, S., Carlson, A., Chang, K., Malave, V., Mason, R., Just, M.: Predicting Human Brain Activity Associated with the Meanings of Nouns. Science 320, 1191 (2008)

Unsupervised Classifier Selection Based on Two-Sample Test

Timo Aho, Tapio Elomaa, and Jussi Kujala

Department of Software Systems, Tampere University of Technology
P. O. Box 553 (Korkeakoulunkatu 1), FI-33101 Tampere, Finland
{timo.aho,tapio.elomaa,jussi.kujala}@tut.fi

Abstract. We propose a well-founded method of ranking a pool of m trained classifiers by their suitability for the current input of n instances. It can be used when dynamically selecting a single classifier as well as in weighting the base classifiers in an ensemble. No classifiers are executed during the process. Thus, the n instances, based on which we select the classifier, can as well be unlabeled. This is rare in previous work. The method works by comparing the training distributions of classifiers with the input distribution. Hence, the feasibility for unsupervised classification comes with a price of maintaining a small sample of the training data for each classifier in the pool.

In the general case our method takes time $O\big(m(t+n)^2\big)$ and space $O(mt+n)$, where t is the size of the stored sample from the training distribution for each classifier. However, for commonly used Gaussian and polynomial kernel functions we can execute the method more efficiently. In our experiments the proposed method was found to be accurate.

1 Introduction

The problem of *dynamic classifier selection* arises prominently in data stream classification [1], but it is also present in, e.g., tracking recurring drifting concepts [2,3], dynamical learning algorithm selection [4], and weighting or selecting the base classifiers in an ensemble [5,6]. All of these situations are dynamic in the sense that a classification algorithm has not been fixed beforehand in a separate training phase. Rather, the instances that are observed online affect our choice.

For example, in concept drifting the distribution underlying the data keeps changing over time. Often the states of the distribution reoccur after a while. Thus, it is useful to store and restore the data and classifiers of the past. Most of the practical restoring methods include choosing the classifier with the best fit for the current input. This can be the case with a single classifier [2] and with ensemble classifiers [6,7]. In the latter case the ranking information is used to weight the responses of classifiers depending on their suitability for the current input.

In either case, we have a set (pool) $\mathcal{H} = \{h_1, h_2, \ldots, h_m\}$ of varying kinds of classifiers $h\colon \mathcal{X} \to \mathcal{Y}$ available. The classifiers are trained on dissimilar instance distributions. Each classifier h_i maps any instance $\boldsymbol{x} \in \mathcal{X}$ to a class label $h_i(\boldsymbol{x}) \in$

$\mathcal{Y}$. Assume also that we can easily associate with each classifier a random sample drawn from its underlying training set. Moreover, we have a sequence of new instances $(\boldsymbol{x}) = \boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \ldots, \boldsymbol{x}_n$ available; they may be labeled (belong to $\mathcal{X} \times \mathcal{Y}$) or not (belong to $\mathcal{X}$). Keep in mind that this sequence may only be the first batch from a longer sequence.

How should we proceed when we want to obtain a good classifier for $(\boldsymbol{x})$? If the sequence is labeled the most straightforward approach, of course, is to employ the most suitable learning algorithm (see e.g. [8]), feed $(\boldsymbol{x})$ as training set, and take the resulting classifier as our choice. However, the set $(\boldsymbol{x})$ may be small when compared to the training sets on which the classifiers in $\mathcal{H}$ were trained. Also, the classifiers in the pool may, e.g., be more general than is possible to attain by training a new one.

Furthermore, if the classifiers are not of the same type, it is possible that different types of classifiers are chosen to best suit the underlying distributions. According to the "No Free Lunch" theorems [9] no single classifier type is superior in all situations. Experimental evidence also supports this [8]. Because of all these reasons, we concentrate on methods that choose the classifier from the pool $\mathcal{H}$.

The standard approach to choosing a classifier from $\mathcal{H}$ is to execute each of them on the input sequence $(\boldsymbol{x})$ and choose the most accurate one. This is the traditional solution to the dynamic classifier selection problem and has been used, e.g., by Watanabe [10]. Furthermore, the same method has generally been used for weighting the classifiers in an ensemble [6]. Assuming linear-time execution of classifiers, the time requirement of this procedure is $O(mn)$ for m classifiers and an example sequence of length n. In this case the sequence $(\boldsymbol{x})$ needs to be labeled. Thus, the method is infeasible for, e.g., uncategorized web pages. We are aiming at an efficient solution that would also let us use unsupervised learning.

Basically the method that we propose, MMDSel, compares sample distributions. This is done by maintaining a small sample of the training data for each classifier in the pool $\mathcal{H}$. However, it is not always necessary to store the samples explicitly. MMDSel gives the similarity of the samples drawn from the training data with the input sequence $(\boldsymbol{x})$. Thus, we can either rank the classifiers or select the one having a training distribution most similar to the input distribution. The main advantage of the method is that no classifier needs to be executed. Hence, it is useful in at least three different situations. First and most importantly, MMDSel can be used in unsupervised environments. Second, it may be more efficient when execution of classifiers is inefficient. There may also be other reasons — like privacy issues [11] — to prevent unnecessary classification. Third, our method does not need extensive preparation operation other than the sample storing (cf., e.g., meta-learning approaches [12]). Thus, the method is feasible for dynamic setting where the pool $\mathcal{H}$ may change online.

Most earlier approaches are usable only in the supervised setting [2,5]. However, Ali and Smith [8] aimed to find the most suitable classifier types for different kinds of distributions using fixed complexity measures. Some of the measures are statistical and depend only on the distribution of the data. Thus, the approach is also feasible for unsupervised learning. Nevertheless, Ali and Smith did not really study dynamic selection of trained classifiers. Rather, they aimed to find a rule-based classifier type selection grounded on prior knowledge of the problem. Moreover, Zhu, Wu, and Yang [1] gave a method in data stream model that includes partitioning the whole instance space into subsets according to the feature values. The approach uses class labels of instances only in finding the classification accuracy of base classifiers for these subsets. Hence, the method could partly be used under unsupervised conditions.

A field with some similarity with classifier selection is choosing with expert advice [13]. Here the task is to minimize the amount of unsatisfactory decisions for a sequence of tasks. Traditionally the experts and the distribution underlying the decision are quite static. Similarly, we also aim to find the expert that is the best in hindsight. However, in dynamic classifier selection framework the distribution alters all the time and, thus, we are interested in single decision, not in the decision sequence. We do not necessarily have any optimal classifier; any single classifier would usually have been a poor choice for the whole sequence. Rather the best classifier depends entirely on the current distribution. Also, the set of classifiers $\mathcal{H}$ may alter.

The remainder of this paper is organized as follows. In Section 2 we briefly examine the theoretical background of MMDSel. Then, in Section 3, we present a way to compute our method efficiently with some kernels. Section 4 reports on an empirical evaluation of the proposed approach. After that we give the concluding remarks of this work.

2 Classifier Selection Using Samples of Training Examples

The basic idea of MMDSel is to compare a kind of a fingerprint of the training distribution of a classifier with the one of the input distribution. One could, of course, use the classifier itself as a fingerprint. That is, we could train a new classifier for the input examples $(\boldsymbol{x})$ and search for the nearest one in the set $\mathcal{H}$. If the classifiers have an explicit weight vector that includes all the information about the classifier, we can simply search for the most similar weight from the pool. Support vector machines (SVMs) with a linear kernel are an example of such classifiers. With the trivial search the best classifier can be found in $O(m + f(n))$ time and $O(m + n)$ space. Here the function $f(n)$ is the average time consumption of classifier learning a set of n elements. Nearest neighbor methods [14,15,16] could be used to reduce the time requirement. This method, though, is only usable for certain classifiers and only in the supervised setting. Also, in our experiments the method appeared quite unreliable.

Instead, MMDSel rather stores a sample of the training set of each classifier. The sample is in fact a fingerprint of the training distribution. If the input sequence $(\boldsymbol{x})$ is drawn from the same (or a very similar) distribution, we should choose the classifier trained for it — most likely it is the most suitable one. To be exact, let the set $S = \{s_1, s_2, \ldots, s_m\}$ contain the training samples of size t for the m classifiers in $\mathcal{H}$. If we select the sample that is most similar with the sequence $(\boldsymbol{x})$ of length n, we should be able to find the classifier that best fits the current input without executing any of the classifiers.

The methods that compare distribution similarity via samples are called two-sample tests. For example, a two-sample test based on *maximum mean discrepancy* (MMD) proposed by Gretton et al. [17,18] and Smola et al. [19] uses $O\big((t+n)^2\big)$ time and $O(t+n)$ space. On the other hand, Borgwardt et al. [20] propose computing MMD in linear $O(t+n)$ time by randomized approximation. However, by our experiments, this leads to a significant reduction of accuracy.

MMD has previously been used for a somewhat similar application by Huang et al. [21]. They use MMD values to solve the sample selection bias problem where the test distribution differs from the training distribution.

There are several additional minor advantages in the two-sample approach adopted in MMDSel. Firstly, due to statistical insignificance of the labels, mislabeled examples have only a minor impact on the result. Also, in some cases this measure for the suitability of the classifier may be more appropriate than the error rate of classifiers used in the traditional solution [8].

2.1 Comparing Distributions with MMD

Let us now briefly introduce MMDSel and the measure MMD for executing it. The measure was originally designed for two-sample tests in which the problem is to find out whether two given samples come from different distributions. Formally we are given two samples X and Y drawn i.i.d. from distributions D_1 and D_2, respectively, and we want to know whether $D_1 \neq D_2$. In our application it would be useful to know also the "distance" or dissimilarity $d(D_1, D_2)$ between the distributions because the distribution may have changed only slightly.

In fact, MMD is a measure of dissimilarity — or more specifically, discrepancy — between the distributions. It is essentially the maximum difference between the mean of test function values on the distributions. Thus, informally, if the MMD between two samples is near zero the distributions are very similar. With the computed MMD value there are multiple methods to decide if we have enough evidence to reject the null hypothesis $D_1 = D_2$ or not [17]. Because we are only interested in rating the alternatives due to their similarity, the measure for the discrepancy itself is enough for our needs.

Formally MMD is defined by Gretton et al. [17] as follows.

Definition 1 *Let $\mathcal{F}$ be a class of test functions $f\colon \mathcal{X} \to \mathbb{R}$ and let D_1 and D_2 be distributions defined on the domain $\mathcal{X}$. Then the maximum mean discrepancy is*

$$\mathrm{MMD}[\mathcal{F}, D_1, D_2] := \sup_{f \in \mathcal{F}} \left(\boldsymbol{E}_{\boldsymbol{x} \sim D_1}[f(\boldsymbol{x})] - \boldsymbol{E}_{\boldsymbol{y} \sim D_2}[f(\boldsymbol{y})] \right) .$$

We select $\mathcal{F}$ to be a reproducing kernel Hilbert space (RKHS) with an associated kernel k. The kernel selection lets us control which properties of distributions are emphasized. Thus, we should choose the kernel according to the features in which we have an interest. Gretton et al. [17] also proved that, in compact domains $\mathcal{X}$, with so-called *universal kernels* [22] MMD attains zero value if and only if D_1 equals D_2. In practice this means that, given that the samples are large enough, two different distributions can be separated with a universal kernel family. Steinwart [22] proved that, e.g., Gaussian and Laplacian kernels are universal. Even if we are not interested in the two-sample test itself, these kernels could be useful in some situations.

Gretton et al. [17] devised an easier way to calculate MMD owing to the fact that in RKHS the function f can be evaluated by the inner product $f(x) = \langle k(x, \cdot), f \rangle$. Hence, let x and x' be independent random variables with distribution D_1 and, similarly, $y, y' \sim D_2$. The square of MMD behaves identically as a measure for discrepancy. For it a more useful form can be derived [17]:

$$\begin{aligned}\mathrm{MMD}^2[\mathcal{F}, D_1, D_2] = \mathbf{E}_{\boldsymbol{x},\boldsymbol{x}' \sim D_1}[k(\boldsymbol{x}, \boldsymbol{x}')] - 2\mathbf{E}_{\boldsymbol{x} \sim D_1, \boldsymbol{y} \sim D_2}[k(\boldsymbol{x}, \boldsymbol{y})] \\ + \mathbf{E}_{\boldsymbol{y},\boldsymbol{y}' \sim D_2}[k(\boldsymbol{y}, \boldsymbol{y}')] \ .\end{aligned} \tag{1}$$

In practice we are dealing with samples X and Y drawn i.i.d. from D_1 and D_2, respectively. Let $(\boldsymbol{z}_1, ..., \boldsymbol{z}_n)$ be i.i.d. random variables, where $\boldsymbol{z}_i = (\boldsymbol{x}_i, \boldsymbol{y}_i)$ and $\boldsymbol{x}_i \in X, \boldsymbol{y}_i \in Y$. Gretton et al. [17] proved that an unbiased empirical estimate for MMD squared is

$$\mathrm{MMD}^2[\mathcal{F}, X, Y] = \frac{1}{n(n-1)} \sum_{i \neq j}^{n} h(\boldsymbol{z}_i, \boldsymbol{z}_j) \ , \tag{2}$$

where $h(\boldsymbol{z}_i, \boldsymbol{z}_j) = k(\boldsymbol{x}_i, \boldsymbol{x}_j) + k(\boldsymbol{y}_i, \boldsymbol{y}_j) - k(\boldsymbol{x}_i, \boldsymbol{y}_j) - k(\boldsymbol{x}_j, \boldsymbol{y}_i)$. Using this we can easily compute the MMD value.

Gretton et al. [17] also gave a biased estimate that can be computed under some restrictions[1] for the kernel function as

$$\begin{aligned}\mathrm{MMD}^2[\mathcal{F}, X, Y] = \frac{1}{n^2} \sum_{i,j=1}^{n} k(\boldsymbol{x}_i, \boldsymbol{x}_j) - \frac{2}{nt} \sum_{i,j=1}^{n,t} k(\boldsymbol{x}_i, \boldsymbol{y}_j) \\ + \frac{1}{t^2} \sum_{i,j=1}^{t} k(\boldsymbol{y}_i, \boldsymbol{y}_j) \ ,\end{aligned} \tag{3}$$

where n and t are the sizes of samples X and Y. The estimate is biased, but there is an upper bound for the bias [17]. In our experiments we found this estimate to be slightly more accurate.

To execute MMDSel and rank the classifiers in a pool of size m we have to compute MMD for m pairs of samples. This results in the dissimilarity of each classifier with the current input distribution. In other words we have ranked the classifiers by their suitability for the current input. Thus we can easily either weight the classifiers according to the rank or choose the most suitable one.

[1] It is enough that for two i.i.d. random variables $\boldsymbol{x}, \boldsymbol{x}' \sim D$: $\mathbf{E}_{\boldsymbol{x},\boldsymbol{x}' \sim D}[k(\boldsymbol{x}, \boldsymbol{x}')] < \infty$ [17].

Because computing MMD for each sample pair takes $O\big((t+n)^2\big)$ time, MMDSEL can be executed in time $O\big(m(t+n)^2\big)$ and space $O(mt+n)$. However, below we show that for some kernels the evaluation can be done much more efficiently.

3 Computing MMD Efficiently

We now discuss some ways to improve the efficiency of MMDSEL. As mentioned the choice of a kernel affects the value of MMD. Thus, we should choose the kernel that is able to track the essential attributes of distributions — i.e., those that determine the similarity. For example, with a linear kernel MMD clearly indicates the difference between expected values.

3.1 Polynomial Kernels

For often-used polynomial kernels (1) can be simplified significantly. This leads to efficient computation of MMDSEL.

The main idea of the optimization is the following. For example, for a symmetric and linear (e.g., Euclidean inner product in $\mathbb{R}$) kernel (1) can be computed by

$$\begin{aligned}\mathrm{MMD}^2[\mathcal{F}, D_1, D_2] = k(\mathbf{E}_{\boldsymbol{x}\sim D_1}[\boldsymbol{x}], \mathbf{E}_{\boldsymbol{x}\sim D_1}[\boldsymbol{x}]) - 2\,k(\mathbf{E}_{\boldsymbol{x}\sim D_1}[\boldsymbol{x}], \mathbf{E}_{\boldsymbol{y}\sim D_2}[\boldsymbol{y}]) \\ + k(\mathbf{E}_{\boldsymbol{y}\sim D_2}[\boldsymbol{y}], \mathbf{E}_{\boldsymbol{y}\sim D_2}[\boldsymbol{y}]) \ .\end{aligned} \tag{4}$$

Hence, we do not need to store the sample for each classifier explicitly. It suffices to store the expected values. Moreover, it is enough to compute the expected value of input examples only once during the test. The time and space requirement for these preparations is clearly $O(mtd)$ for d dimensional data. On the other hand, during the actual selection process only a constant amount of inner products are calculated, yielding a time requirement of $O((m+n)d)$.

The optimization can be generalized to all the polynomial kernels $(\langle\cdot,\cdot\rangle + c)^p$ of a finite integer degree p, where $c \geq 0$ is a constant. Using the technique introduced by Raykar et al. [23] polynomial kernels can be calculated in

$$r_{pd} = \binom{p+d}{d}$$

time and space. Hence, MMDSEL can be executed in $O((m+n)r_{pd})$ time and space with $O(mtr_{pd})$ time and space preparation. With the natural assumption that $p \ll d$, the value r_{pd} can be upper bounded by $O(d^p)$.

Compared to the brute force solution in $O\big(md(t+n)^2\big)$ time and $O(d(mt+n))$ space, the optimized MMDSEL is useful for low values of p (e.g., quadratic or linear kernel) and for very large values of t or n.

3.2 Gaussian Kernels

Simple polynomial kernels may not be enough for our needs. Steinwart [22] defined kernel classes to be universal if they are dense on compact domains.

These kernels can approximate any other kernel if the sample sizes are increased enough. If we want to use a universal kernel, e.g., a Gaussian one, a similar optimization method can be used to approximate them.

Yang et al. [24] and Raykar et al. [23] present an improved fast Gaussian transformation (IFGT). Their approach is based on calculating only the first terms of the Taylor series representation of Gaussian kernel. In our case the basic idea of IFGT is to approximate the Gauss transform at a chosen point $\boldsymbol{y}_*$

$$\begin{aligned} G(\boldsymbol{x}_j) &= \sum_{i=1}^{t} \exp\left(-\|\boldsymbol{x}_j - \boldsymbol{y}_i\|^2/\sigma^2\right) \\ &= \sum_{i=1}^{t} \exp\left(-\|\boldsymbol{y}_i - \boldsymbol{y}_*\|^2/\sigma^2\right) \exp\left(-\|\boldsymbol{x}_j - \boldsymbol{y}_*\|^2/\sigma^2\right) \\ &\qquad \cdot \exp\left(2\langle \boldsymbol{x}_j - \boldsymbol{y}_*, \boldsymbol{y}_i - \boldsymbol{y}_*\rangle/\sigma^2\right) \ . \end{aligned} \tag{5}$$

However to gain accuracy they cluster the space and replace the vector $\boldsymbol{y}_*$ with the centers of these clusters.

With IFGT the Gaussian kernel can be approximated in

$$O\big(nk' r_{(p'-1)d} + nk\big)$$

time with

$$O\big(t \log k + t r_{(p'-1)d}\big)$$

time preparation. Here $k < t$ is the number of clusters in the example space and $k' < k$ the maximum number of neighbor clusters and p' the number of Taylor series terms to be computed. Both the p' and k' depend on the desired error bound $\epsilon > 0$ and k' also depends on Gaussian kernel bandwidth. The space usage is $O\big(k r_{(p'-1)d} + t + n\big)$. Recall that $r_{pd} = O(d^p)$.

The procedure is interesting in our application, because if we cluster the whole instance space at once, we can reduce the running time significantly. If the values corresponding to $\boldsymbol{y}_*$ in (5) are same for every classifier on the pool $\mathcal{H}$ we have to compute the terms including $\boldsymbol{y}_j$ in equation only once during the computation of MMDSel.

Hence, when computing (1) we can calculate the term $\mathbf{E}_{\boldsymbol{y},\boldsymbol{y}'\sim D_2}[k(\boldsymbol{y},\boldsymbol{y}')]$ completely for all the samples in S beforehand in the preparation phase. When $\mathbf{E}_{\boldsymbol{x}\sim D_1,\boldsymbol{y}\sim D_2}[k(\boldsymbol{x},\boldsymbol{y})]$ is expressed using (5) this is also the case for parts including $\boldsymbol{y}$. Thus the preparation phase takes

$$O\Big(mt\left(\log k + d^{p'}\right)\Big)$$

time.

Finally in the online phase we compute the term $\mathbf{E}_{\boldsymbol{x},\boldsymbol{x}'\sim D_1}[k(\boldsymbol{x},\boldsymbol{x}')]$ and the rest of the term $\mathbf{E}_{\boldsymbol{x}\sim D_1,\boldsymbol{y}\sim D_2}[k(\boldsymbol{x},\boldsymbol{y})]$. This takes time

$$O\Big((m + nk')\, d^{p'} + nk\Big) \ .$$

Table 1. Summary of asymptotic time and space requirement of MMDSel

Kernel	Online Time	Online Space	Preparation Time	Preparation Space
Linear	$O((m+n)d)$	$O((m+n)d)$	$O(mtd)$	$O(mtd)$
Integer polynomial	$O((m+n)d^p)$	$O((m+n)d^p)$	$O(mtd^p)$	$O(m(td+d^p))$
Gaussian (IFGT)	$O\Big((m+nk)\,d^{p'}\Big)$	$O\Big(mnd^{p'}\Big)$	$O\Big(mt\left(\log k+d^{p'}\right)\Big)$	$O\Big(mtd^{p'}\Big)$
General	$O\big(md(t+n)^2\big)$	$O(d(mt+n))$	—	—
General randomized	$O(md(t+n))$	$O(d(mt+n))$	—	—
Traditional method	mnd	nd	—	—

For Gaussian kernels there are also other optimization approaches. For example Lee, Gray and Moore [25] give an approach based on space partitioning trees. Also Herbster [26] presents a simple way to compute additive Gaussian kernels even more efficiently.

A summary of the asymptotic time and space requirement of MMDSel in different settings is presented in Table 1. The randomized general version of MMDSel is based on the randomized linear time computation of MMD proposed by Borgwardt et al. [20].

4 Empirical Evaluation

Let us now evaluate MMDSel experimentally. As a reference approach we use the traditional solution of executing each classifier on the input and selecting the most accurate one. The accuracy of a classifier is computed by counting how many times it predicts the right label given by the data set. The experiments are executed on domains of the UCI machine learning repository and MNIST datasets. MNIST consists of images of handwritten digits from 0 to 9. From the UCI repository we choose the classification datasets with the greatest number of different class labels.

For every dataset with l different labels we create all the $\binom{l}{2}$ different binary classification *tasks*. The labels for each task are changed to ± 1 and a SVM classifier is trained for each task. A sample of size 20% from each of the training sets for MMDSel is stored.

The test examples are divided similarly and for each test the best classifier is chosen. Hence, for each test set there is exactly one trained classifier with the same training distribution. We report how many times different methods select this one correct classifier. The test set size is 20% of the training set size.

In UCI datasets some labels are removed due to small number of examples. Also some textual values are changed to numerical ones. In addition all the datasets are randomly permuted. Due to efficiency the size of training set is restricted to 1 000 instances.

We have implemented the introduced methods in Matlab with both linear and Gaussian kernel. The kernel width for Gaussian kernels is chosen with the rule-of-thumb

$$\sigma = \sqrt{\text{median distance between points}/2}\ .$$

In the experiments MMDSel chooses the classifier with the smallest distribution discrepancy, regardless of the value. For MMD computation [17] we use the biased estimate of (3), because it is slightly more accurate than the unbiased one of (2). The reported MMDSel versions are tried for unlabeled data because removing labels does not affect the accuracy.

The accuracy of the methods in at least 1 000 separate tasks are reported in Table 2. Each task results in either a right or a wrong answer for a method.

In the overall performance MMDSel prevails. In nearly all of the data sets MMDSel with a Gaussian kernel seems to take the lead with the linear kernel version being slightly worse. However, with the traditional SVM classification the linear kernel version surpasses the Gaussian one.

There are, however, some interesting exceptions. The `vowel` data set includes features of English pronounced vowels. The traditional method is better on this. The `glass` data set contains the chemical composition of different glass types. The traditional method overtakes linear MMDSel version also on this set.

It is worth noting that the setting is quite unfavorable for the classification approach. The behavior of a classifier depends essentially on the hyperplane that is computed to separate training classes. Thus, in the testing phase we are searching for the classifier with a hyperplane separating the test classes. However, because of the test setting there could be multiple hyperplanes that give similar classification for the test set. Hence, many hypotheses would gain good classification accuracy. This could be partial explanation for the worse performance of the classification method. However, in reality the classification accuracies in our experiments seem to be quite diverse. Usually alone the chosen classifier has the best accuracy.

Gaussian kernel SVM performed surprisingly poorly in these evaluations. For example with the `ecoli` data set the result is only slightly better than a random guess. This is also emphasized with high dimensional data. A natural reason for this is the 'curse of dimensionality' that affects the Gaussian kernel [27]. This

Table 2. Accuracy of different methods after at least 1 000 test tasks

Data set			SVM Classification		MMDSel	
Name	Labels	Dimension	Gaussian	Linear	Gaussian	Linear
`mnist`	10	784	4.3	95.4	**100.0**	**100.0**
`abalone`	14	8	5.8	8.3	**18.2**	10.1
`ecoli`	5	7	11.3	56.3	**78.9**	75.0
`glass`	5	9	38.5	39.5	**46.2**	36.3
`letter`	26	16	77.5	84.2	**100.0**	97.6
`vowel-context`	10	12	38.4	**51.4**	21.6	18.8
`krkopt`	18	6	26.8	25.6	**32.9**	28.0
`led` with 10% noise	10	7	32.9	51.0	**93.1**	87.3
`mfeat`	10	649	9.8	64.6	**91.1**	74.1
`pendigits`	10	16	28.0	76.9	**100.0**	**100.0**
Mean	—	—	27.3	55.3	68.2	62.7

appears because of the small number of training examples compared to their dimension. Interestingly MMDSel does not seem to be affected by this. Maybe this is because MMDSel tracks down the underlying distribution instead of single points.

However, in additional experiments optimizing the Gaussian kernel width seemed to improve the SVM performance somewhat. Nevertheless, it did not reach the performance of linear SVM version on the high dimensional data sets. For example with `ecoli` data set the Gaussian SVM attained 44.0% accuracy with a smaller kernel width.

In summary, MMDSel would appear to offer a viable alternative to the traditional method. Only a small stored sample of the training distribution suffices to let us choose the correct classifier with high probability without even knowing the class labels of instances.

5 Conclusions

In this paper we introduced a method of ranking a pool of classifiers by their suitability for the current input. The MMDSel method is based on the similarities of classifier training distributions and the current input distribution. Thus, it is suitable for unsupervised learning. This advantage is, to the best of our knowledge, rare in previous work. Also, classification algorithm outputs are not used and thus the type of algorithms is entirely unlimited. Moreover, the pool may consist of different types of algorithms. We also showed that the test can be computed asymptotically efficiently with some optimization methods.

In our empirical evaluation the MMDSel with both linear and Gaussian kernels seems to be accurate enough to be a viable alternative for solving the given problem at least on some data.

There are some interesting open questions: Why the curse of dimensionality does not seem to affect MMDSel on Gaussian kernel as seen in our experiments? Also, surely different well-founded methods in addition to MMD for finding distribution similarities are possible.

Acknowledgments

This work was supported by Academy of Finland project "ALEA: Approximation and Learning Algorithms." Moreover, the work of T. Aho is financially supported by Tampere Graduate School in Information Science and Engineering (TISE) and the work of T. Elomaa by Academy of Finland project "Machine Learning and Online Data Structures".

References

1. Zhu, X., Wu, X., Yang, Y.: Effective classification of noisy data streams with attribute-oriented dynamic classifier selection. Knowledge and Information Systems 9(3), 339–363 (2006)

2. Klinkenberg, R.: Learning drifting concepts: Example selection vs. example weighting. Intelligent Data Analysis 8(3), 281–300 (2004)
3. Gama, J., Medas, P., Castillo, G., Rodrigues, P.P.: Learning with drift detection. In: Bazzan, A.L.C., Labidi, S. (eds.) SBIA 2004. LNCS (LNAI), vol. 3171, pp. 286–295. Springer, Heidelberg (2004)
4. Merz, C.J.: Dynamical selection of learning algorithms. In: Fisher, D., Lenz, H.J. (eds.) Learning from Data: Artificial Intelligence and Statistics. Lecture Notes in Statistics, vol. 112, pp. 281–290. Springer, Berlin (1996)
5. Ko, A.H., Sabourin, R., Britto Jr., A.S.: From dynamic classifier selection to dynamic ensemble selection. Pattern Recognition 41(5), 1735–1748 (2008)
6. Kolter, J.Z., Maloof, M.A.: Dynamic weighted majority: An ensemble method for drifting concepts. Journal of Machine Learning Research 8, 2755–2790 (2007)
7. Wang, H., Fan, W., Yu, P.S., Han, J.: Mining concept-drifting data streams using ensemble classifiers. In: Getoor, L., Senator, T.E., Domingos, P., Faloutsos, C. (eds.) Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 226–235. ACM Press, New York (2003)
8. Ali, S., Smith, K.A.: On learning algorithm selection for classification. Applied Soft Computing 6(2), 119–138 (2006)
9. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation 1(1), 67–82 (1997)
10. Watanabe, O.: Sequential sampling techniques for algorithmic learning theory. Theoretical Computer Science 348(1), 3–14 (2005)
11. Wu, X., Chu, C.H., Wang, Y., Liu, F., Yue, D.: Privacy preserving data mining research: Current status and key issues. In: Shi, Y., van Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds.) ICCS 2007. LNCS, vol. 4489, pp. 762–772. Springer, Heidelberg (2007)
12. Janssen, F., Fürnkranz, J.: On meta-learning rule learning heuristics. In: Proceedings of the Seventh IEEE International Conference on Data Mining, pp. 529–534. IEEE Computer Society, Los Alamitos (2007)
13. Cesa-Bianchi, N., Freund, Y., Haussler, D., Helmbold, D.P., Schapire, R.E., Warmuth, M.K.: How to use expert advice. Journal of the ACM 44(3), 427–485 (1997)
14. Andoni, A., Indyk, P.: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science, pp. 459–468. IEEE Computer Society, Los Alamitos (2006)
15. Charikar, M.S.: Similarity estimation techniques from rounding algorithms. In: Proceedings of the 34th Annual ACM Symposium on Theory of Computing, pp. 380–388. ACM Press, New York (2002)
16. Chan, T.M.: Closest-point problems simplified on the RAM. In: Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 472–473. SIAM, Philadelphia (2002)
17. Gretton, A., Borgwardt, K.M., Rasch, M., Schölkopf, B., Smola, A.J.: A kernel method for the two-sample-problem. In: Schölkopf, B., Platt, J.C., Hoffman, T. (eds.) Advances in Neural Information Processing Systems, vol. 19, pp. 513–520. MIT Press, Cambridge (2007)
18. Gretton, A., Borgwardt, K.M., Rasch, M., Schölkopf, B., Smola, A.J.: A kernel approach to comparing distributions. In: Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, pp. 1637–1641. AAAI Press, Menlo Park (2007)

19. Smola, A.J., Gretton, A., Song, L., Schölkopf, B.: A Hilbert space embedding for distributions. In: Hutter, M., Servedio, R.A., Takimoto, E. (eds.) ALT 2007. LNCS (LNAI), vol. 4754, pp. 13–31. Springer, Heidelberg (2007)
20. Borgwardt, K.M., Gretton, A., Rasch, M.J., Kriegel, H.P., Schölkopf, B., Smola, A.J.: Integrating structured biological data by Kernel Maximum Mean Discrepancy. Bioinformatics 22(14), 49–57 (2006)
21. Huang, J., Smola, A.J., Gretton, A., Borgwardt, K.M., Schölkopf, B.: Correcting sample selection bias by unlabeled data. In: Schölkopf, B., Platt, J.C., Hoffman, T. (eds.) Advances in Neural Information Processing Systems, vol. 19, pp. 601–608. MIT Press, Cambridge (2007)
22. Steinwart, I.: On the influence of the kernel on the consistency of support vector machines. Journal of Machine Learning Research 2, 67–93 (2001)
23. Raykar, V.C., Duraiswami, R.: The improved fast Gauss transform with applications to machine learning. In: Bottou, L., Chapelle, O., DeCoste, D., Weston, J. (eds.) Large-Scale Kernel Machines, pp. 175–201. MIT Press, Cambridge (2007)
24. Yang, C., Duraiswami, R., Davis, L.S.: Efficient kernel machines using the improved fast Gauss transform. In: Saul, L.K., Weiss, Y., Bottou, L. (eds.) Advances in Neural Information Processing Systems, vol. 17, pp. 1561–1568. MIT Press, Cambridge (2004)
25. Lee, D., Gray, A.G., Moore, A.W.: Dual-tree fast gauss transforms. In: Weiss, Y., Schölkopf, B., Platt, J. (eds.) Advances in Neural Information Processing Systems, vol. 18, pp. 747–754. MIT Press, Cambridge (2006)
26. Herbster, M.: Learning additive models online with fast evaluating kernels. In: Helmbold, D.P., Williamson, B. (eds.) COLT 2001 and EuroCOLT 2001. LNCS (LNAI), vol. 2111, pp. 444–460. Springer, Heidelberg (2001)
27. Bengio, Y., LeCun, Y.: Scaling learning algorithms towards AI. In: Bottou, L., Chapelle, O., DeCoste, D., Weston, J. (eds.) Large-Scale Kernel Machines, pp. 321–388. MIT Press, Cambridge (2007)

An Empirical Investigation of the Trade-Off between Consistency and Coverage in Rule Learning Heuristics

Frederik Janssen and Johannes Fürnkranz

TU Darmstadt, Knowledge Engineering Group
Hochschulstraße 10, D-64289 Darmstadt, Germany
{janssen,juffi}@ke.informatik.tu-darmstadt.de

Abstract. In this paper, we argue that search heuristics for inductive rule learning algorithms typically trade off consistency and coverage, and we investigate this trade-off by determining optimal parameter settings for five different parametrized heuristics. This empirical comparison yields several interesting results. Of considerable practical importance are the default values that we establish for these heuristics, and for which we show that they outperform commonly used instantiations of these heuristics. We also gain some theoretical insights. For example, we note that it is important to relate the rule coverage to the class distribution, but that the true positive rate should be weighted more heavily than the false positive rate. We also find that the optimal parameter settings of these heuristics effectively implement quite similar preference criteria.

1 Introduction

Evaluation metrics for rule learning typically, in one way or another, trade off consistency and coverage. On the one hand, rules should be as consistent as possible by only covering a small percentage of negative examples. On the other hand, rules with high coverage tend to be more reliable, even though they might be less precise on the training examples than alternative rules with lower coverage. An increase in coverage of a rule typically goes hand-in-hand with a decrease in consistency, and vice versa. In fact, the conventional top-down hill-climbing search for single rules follows exactly this principle: starting with the empty rule, conditions are greedily added, thereby decreasing coverage but increasing consistency.

In this work, we show that five well-known rule evaluation metrics (a cost trade-off, a relative cost trade-off, the *m*-estimate, the F-measure, and the Klösgen measures) provide parameters that allow to control this trade-off. After a brief discussion of these heuristics, we will report on an extensive experimental study with the goal of determining optimal values for each of their respective parameters, which will allow us to draw some interesting conclusions about heuristic rule learning.

2 Separate-and-Conquer Rule Learning

The goal of an inductive rule learning algorithm is to automatically learn rules that allow to map the examples of the training set to their respective classes. Algorithms differ in the way they learn individual rules, but most of them employ a *separate-and-conquer* or *covering* strategy for combining rules into a rule set [5], including RIPPER [3], arguably one of the most accurate rule learning algorithms today.

Separate-and-conquer rule learning can be divided into two main steps: First, a single rule is learned from the data (the *conquer* step). Then all examples which are covered by the learned rule are removed from the training set (the *separate* step), and the remaining examples are "conquered". The two steps are iterated until no more positive examples are left. In a simple version of the algorithm this ensures that every positive example is covered at least by one rule (*completeness*) and no negative example is included (*consistency*). More complex versions of the algorithm will allow certain degrees of incompleteness (leaving some examples uncovered) and inconsistencies (covering some negative examples).

For our experiments, we implemented a simple separate-and-conquer rule-learner with a top-down hill-climbing search for individual rules. Rules are greedily refined until no more negative examples are covered, and the best rule encountered in this refinement process (not necessarily the last rule) is returned. We did not employ explicit stopping criteria or pruning techniques for overfitting avoidance, because we wanted to gain a principal understanding of what constitutes a good rule evaluation metric.

3 Rule Learning Heuristics

As discussed above, individual rules should simultaneously optimize two criteria:

Coverage: the number of positive examples that are covered by the rule (p) should be maximized and

Consistency: the number of negative examples that are covered by the rule (n) should be minimized.

Thus, most heuristics depend on p and n, but combine these values in different ways. A few heuristics also include other parameters, such as the length of the rule, but we will not further consider those in this paper.[1] In the following, we will closely follow the terminology and notation introduced in [6]. As an evaluation framework coverage spaces [6], un-normalized ROC spaces, are used in the remainder of this paper. These allow to graphically interpret evaluation metrics by their isometrics.

3.1 Basic Heuristics

True Positive Rate (Recall) $h_{tpr} = h_{Recall} = \frac{p}{P}$

Computes the coverage on the positive examples only. It is – on its own – equivalent to simply using p (because P, the total number of positive examples, is constant for a given dataset). Due to its independence of covered negative examples, its isometrics are parallel horizontal lines.

[1] As longer rules typically cover fewer examples, we would argue that this is just another way of measuring coverage. Also, in [8] it was recently found that including rule length does not improve the performance on heuristics that have been derived by meta-learning.

False Positive Rate $h_{fpr} = \frac{n}{N}$
Computes the coverage on the negative examples only (N stands for the total number of negative examples). Its isometrics are parallel vertical lines.
Full Coverage $h_{Coverage} = \frac{p+n}{P+N}$
Computes the fraction of all covered examples. The maximum heuristic value is reached by the universal theory, which covers all examples (the point (N, P) of the coverage space). The isometrics are parallel lines with a slope of -1 (similar to those of the lower right graph in Figure 1).

3.2 Composite Heuristics

The heuristics shown in the previous section only optimize one of the two criteria. Two simple criteria, which try to optimize both criteria are
Precision $h_{Precision} = \frac{p}{p+n}$
Computes the fraction of correctly classified examples (p) among all covered examples (p+n). Its isometrics rotating around the origin.
Weighted Relative Accuracy (WRA) $h_{WRA} = h_{tpr} - h_{fpr}$
Computes the difference between the true positive rate and the false positive rate. The upper middle graph of Figure 1 shows the isometrics of WRA.

However, these two heuristics are known to have complementary disadvantages. Precision is known to overfit the data, i.e., to strongly prefer consistency over coverage. Conversely, the experimental evidence given in [11], which is consistent with our own experience, suggests that WRA has a tendency to overgeneralize, i.e., that it places too strong emphasis on coverage.

Thus, it is necessary to find the right trade-off between consistency and coverage. Many other heuristics implement fixed trade-offs between these criteria. In the next section, we will discuss five heuristics that allow to tune this trade-off with a parameter.

3.3 Parametrized Heuristics

In this section we show that the heuristics which we consider in this work all have a parameter that trades off consistency for coverage, but do so in different forms. The two cost measures directly trade off absolute or relative positive and negative coverage. Thereafter, we will see three measures that use $h_{Precision}$ for optimizing consistency, but use different measures (h_{Recall}, h_{WRA}, $h_{Coverage}$) for optimizing coverage.
Cost Measure $h_{cost} = c \cdot p - (1 - c) \cdot n$
Allows to directly trade off consistency and coverage with a parameter $c \in [0, 1]$. $c = 0$ only considers consistency, $c = 1$ only coverage. If $c = 1/2$, the resulting heuristic ($h_{Accuracy} = p - n$) is equivalent to **accuracy**, which computes the percentage of correctly classified examples among all training examples. The isometrics of this heuristics are parallel lines, with a slope of ${}^{(1-c)}/_{c}$.
Relative Cost Measure $h_{rcost} = c_r \cdot h_{tpr} - (1 - c_r) \cdot h_{fpr}$
Trades off the true positive rate and the false positive rate. This heuristic is quite similar to h_{cost}. In fact, for any particular data set, one can choose $c = \frac{N}{P+N} \cdot c_r$ to transform the cost measure into the relative cost measure. However, this normalization may (and will) make a difference if the same value is used across a wide variety of datasets with

different class distributions. Clearly, setting $c_r = 1/2$ is compatible (as defined in [6]) with WRA.

F-Measure $h_{\textit{F-Measure}} = \frac{(\beta^2+1)\cdot h_{\textit{Precision}}\cdot h_{\textit{Recall}}}{\beta^2\cdot h_{\textit{Precision}}+h_{\textit{Recall}}}$

The F-measure [10] has its origin in Information Retrieval and trades off the basic heuristics $h_{\textit{Precision}}$ and $h_{\textit{Recall}}$. Basically, the isometrics (for an illustration see [6]) are identical to those of precision, with the difference that the rotation point is not in the point $(0,0)$ but in a point $(-g,0)$, where g depends on the choice of β. If $\beta \to 0$, the origin moves towards $(0,0)$, and the isometrics correspond to those of $h_{\textit{Precision}}$. The more the parameter is increased the more the origin of the isometrics is shifted in the direction of the negative N-axis. The observable effect is that the lines in the isometrics becomes flatter and flatter. Conversely if $\beta \to \infty$ the resulting isometrics approach those of $h_{\textit{Recall}}$ which are horizontal parallel lines.

m-Estimate $h_{\textit{m-estimate}} = \frac{p+m\cdot\frac{P}{P+N}}{p+n+m}$

The idea of this parametrized heuristic [2] is to presume that a rule covers m training examples *a priori*, maintaining the distribution of the examples in the training set ($m \cdot {}^{P}/_{(P+N)}$ examples are positive). For $m = 2$ and assuming an equal example distribution ($P = N$), we get the **Laplace** heuristic $h_{\textit{Laplace}}$ as a special case.

If we inspect the isometrics in relation to the different parameter settings, we observe a similar behavior as discussed above for the F-measure, except that the origin of the turning point now does not move on the N-axis, but it is shifted in the direction of the negative diagonal of the coverage space (cf. [6] for an illustration). $m = 0$ corresponds to precision, and for $m \to \infty$ the isometrics become increasingly parallel to the diagonal of the coverage space, i.e., they approach the isometrics of $h_{\textit{WRA}}$. Thus, the m-estimate trades off $h_{\textit{Precision}}$ and $h_{\textit{WRA}}$.

Klösgen $h_{\textit{Klösgen}} = (h_{\textit{Coverage}})^{\omega} \cdot \left(h_{\textit{Precision}} - \frac{P}{P+N}\right)$

This family of measures was first proposed in [9] and trades off *Precision Gain* (the increase in precision compared to the default distribution ${}^{P}/_{(P+N)}$) and *Coverage*. The isometrics of *Precision Gain* on its own behave like the isometrics of precision, except that their labels differ (the diagonal now always corresponds to a value of 0).

Setting $\omega = 1$ results in WRA, and $\omega = 0$ yields *Precision Gain*. Thus, the Klösgen measure starts with the isometrics of $h_{\textit{Precision}}$ and first evolves into those of $h_{\textit{WRA}}$, just like the m-estimate. However, the transformation takes a different route, with non-linear isometrics. The first two graphs of Figure 1 show the result for the parameter settings $\omega = 0.5$ and $\omega = 1$ (WRA), which were suggested by Klösgen [9].

With a further increase of the parameter, the isometrics converge to $h_{\textit{Coverage}}$. The middle left graph shows the parameter setting $\omega = 2$, which was suggested in [13]. Contrary to the previous settings, the isometrics now avoid regions of low coverage, because low coverage is more severely penalized. A further increase of the parameter results in sharper and sharper bends of the isometrics. The influence of WRA (the part parallel to the diagonal) vanishes except for very narrow regions around the diagonal, and the isometrics gradually transform into those of coverage.

Another interesting variation of the Klösgen measure is to divide $h_{\textit{Coverage}}$ by $1 - h_{\textit{Coverage}}$ instead of raising it to the ω-th power. It has been shown before [9] that this is equivalent to **correlation** ($h_{\textit{Corr}} = \frac{p\cdot(N-n)-n\cdot(P-p)}{\sqrt{P\cdot N\cdot(p+n)\cdot(P-p+N-n)}}$).

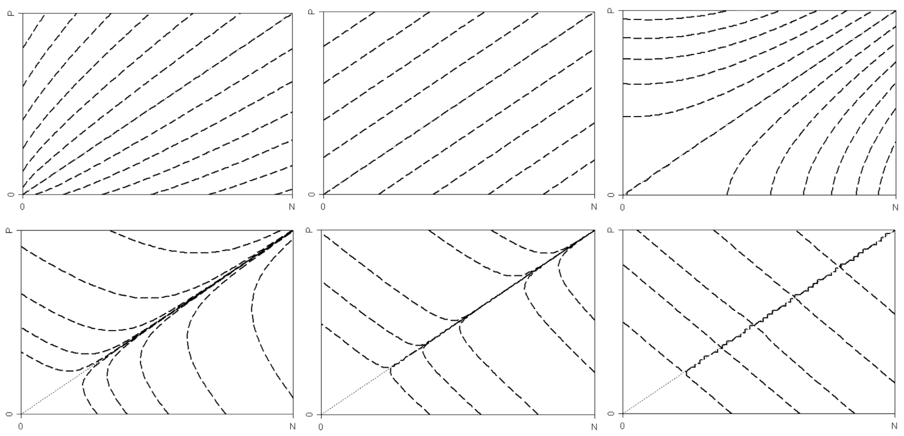


Fig. 1. Klösgen-Measure for $\omega = 0.5, 1, 2, 7, 30, 500$

4 Experimental Setup

The primary goal of our experimental work was to determine settings for the parametrized heuristics that are optimal in the sense that they will result in the best classification accuracy on a wide variety of datasets. Clearly, the optimal setting for individual datasets may vary.

We arbitrarily selected 27 *tuning datasets* from the UCI-Repository ([1]) for determining the optimal parameters. To check the validity of the found parameter settings, we selected 30 additional *validation datasets*. The names of all 57 datasets could be found in ([7]).

The performance on individual datasets was evaluated with a *10-fold stratified Cross Validation* implemented in *Weka* ([12]). As we have a large number of different individual results, a key issue is how to combine them into an overall value. We have experimented with several choices. Our primary method was the *macro-averaged accuracy* of one parametrization of a parametrized heuristic which is defined by the sum of all accuracies (the fraction of correctly classified examples among all examples) of the datasets normalized with the number of datasets. This method gives the same weight to all datasets. Alternatively, one could also give the same weight to each example, which results in *micro-averaged accuracy*. It is defined as the sum of all correctly classified examples divided by the total number of examples among all datasets. In effect, this method assigns a higher weight to datasets with many examples, whereas those with few examples get a smaller weight.

As there are large differences in the variances of the accuracies of the individual datasets, one could also focus only on the *ranking* of the heuristics and neglect the magnitude of the accuracies on different datasets. Small random variations in ranking performance will cancel out over multiple datasets, but if there is a consistent small advantage of one heuristic over the other this will be reflected in a substantial difference in the average rank (the sum of individual ranks normalized by the number of datasets).

Algorithm 1. SEARCHBESTPARAMETER$(a, b, i, h, dataSets)$

```
$acc_{former} = acc_{best}$                                  # global params
$params =$ CREATELIST$(a, b, i)$                             # initialize candidate params
$p_{best} =$ GETBESTPARAM$(h, params, dataSets)$
$acc_{best} =$ GETACCURACY$(p_{best})$
# stop if no substantial improvement ($t = 0.001$)
if $acc_{best} - acc_{former} < t$ then
    return $(p_{best})$
end if
# continue the search with a finer resolution
SEARCHBESTPARAMETER$(p_{best} - \frac{i}{2}, p_{best} + \frac{i}{2}, \frac{i}{10}, h, dataSets)$
```

Finally, we also measured the *size* of the learned theories by the average number of conditions.

5 The Search Strategy

This section describes our method for searching for the optimal parameter setting. Our expectation was that for all heuristics, a plot of accuracy over the parameter value will result in an inverse U-shape, i.e., there will be overfitting for small parameter values and over-generalization for large parameter values, with a region of optimality inbetween. Thus, we adopted a greedy search algorithm that continuously narrows down the region of interest. First, it tests a wide range of intuitively appealing parameter settings to get an idea of the general behavior of each of the five parametrized heuristics. The promising parameters were further narrowed down until we had a single point that represents a region of optimal performance.

Algorithm 1 shows the search procedure in detail. We start with a lower (a) and upper (b) bound of the region of interest, and sample the space between them with a certain interval width i. For measures with parameter space $[0, \infty)$ we used a logarithmic scale. For each sampled parameter value, we estimate its macro-averaged accuracy on all tuning datasets, and, based on the obtained results, narrow down the values a, b, and i.

Intuitively, the farther the lower border a and the upper border b of the interval are away from the best parameter p_{best}, and the denser the increment, the better are our chances to find the optimal parameter, but the higher are the computational demands. As a compromise, we used the following approach for adjusting the values of these parameters:

$$a \leftarrow p_{best} - \frac{i}{2},\ b \leftarrow p_{best} + \frac{i}{2} \text{ and } i \leftarrow \frac{i}{10}$$

This procedure is repeated until the accuracy does not increase significantly. As we compare macro-averaged accuracy values over several datasets, we adopted a simple approach that stops whenever the accuracy improvement falls below a threshold $t = 0.001$.

Obviously, the procedure is greedy and not guaranteed to find a global optimum. In particular, there is a risk to miss the best parameter due to the fact that the global best parameter may lie under or above the borders (if the best one so far is 1 for example, the interval that would be searched is $[0.5, 1.5]$; if the global optimum is 0.4, it would not be detected). Furthermore, we may miss a global optimum if it hides between two apparently lower values. If the curve is smooth, these assumptions are justified, but on real-world data we should not count on this. The second point can be addressed by keeping a list of candidate parameters that are all refined and from which the best one is selected. Hence it has to be defined how many candidates should be maintained. Therefore it is necessary to introduce a threshold that discriminates between a normal and a candidate parameter. It is not trivial to determine such a threshold. Due to this the number of candidate parameters is limited to 3 (all experiments confirmed that this is sufficient). The first problem could be addressed by re-searching the entire interval at a finer resolution, but, for the sake of efficiency, we chose the more efficient version.

However, also note that it is not really important to find an absolute global optimum. If we can identify a region that is likely to contain the best parameter for a wide variety of datasets, this would already be sufficient for our purposes. We interpret the found values as good representatives for optimal regions.

6 Results

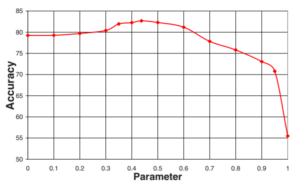
In this section we focus on the results of the search for optimal parameter values. We will illustrate the average accuracy of the different heuristics under various parameter settings, identify optimal parameters, compare their isometrics, and evaluate their general validity.

6.1 Optimal Parameters for the Five Heuristics

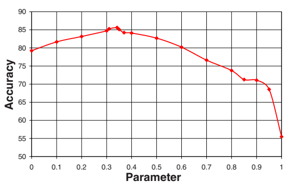
Our first goal was to obtain optimal parameter settings for the five heuristics. As discussed above, the found values are not meant to be interpreted as global optima, but as representatives for regions of optimal performance. Figure 2 shows the obtained performance curves.

Cost Measures. Figures 2 (a) and (b) show the results for the two cost measures. Compared to the other measures, these curves are comparably smooth, and optimal values could be identified quite easily. Optimizing only the consistency (i.e., minimizing the number of negative examples without paying attention to the number of covered positives) has a performance of close to $80\,\%$. Not surprisingly, this can be improved considerably for increasing values of the parameters c and c_r. The best performing values were found at $c = 0.437$ (for the cost metric) and $c_r = 0.342$ (for the relative cost metric). Further increasing these values will decrease performance because of over-generalization. If the parameter approaches 1, there is a steep descent because optimizing only the number of covered examples without regard to the covered negatives is, on its own, a very bad strategy.

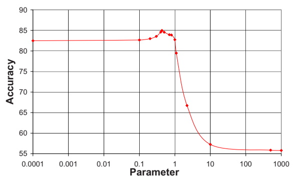
It is interesting to interpret the found values. Note, for example, that weighted relative accuracy, which has been previously advocated as rule learning heuristic ([11]), corresponds to a value of $c_r = 0.5$, equally weighting false positive rate and true positives



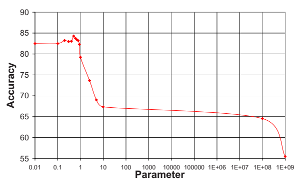
(a) cost measure



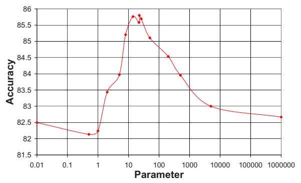
(b) relative cost measure



(c) Klösgen-measures



(d) F-measure



(e) m-estimate

Fig. 2. Macro-averaged Accuracy over parameter values for the five parametrized heuristics

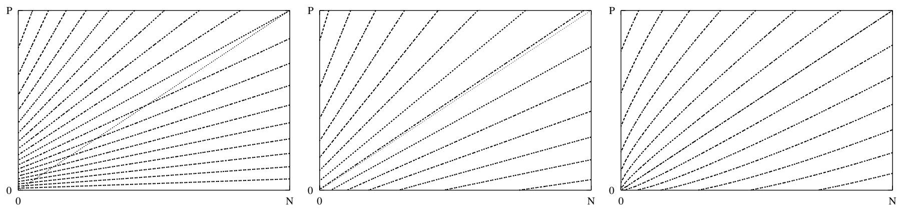
rate. Comparing this to the optimal region for this parameter, which is approximately between 0.3 and 0.35, it can be clearly seen that it pays off to give a higher weight to the true positive rate.[2]

This is confirmed by the results on the cost metric. The optimal value $c = 0.437$ corresponds to a ratio of positive to negative examples of $P/N = 1-c/c \approx 1.29$. In reality, however, for most example sets $P < N$ (for multi-class datasets we assume that P is the number of examples in the largest class). Thus, positive examples have to be given a higher weight than negative examples.

It is also interesting to compare the results of the absolute and relative cost measures: although, as we have stated above, the two are equivalent in the sense that for each individual dataset, one can be transformed into each other by picking an appropriate cost factor, the relative cost measure has a clearly better peak performance exceeding 85%. Thus, it seems to be quite important to incorporate the class distribution $P/(P+N)$ into the evaluation metric. This is also confirmed by the results of $h_{\textit{m-estimate}}$ and $h_{\textit{Klösgen}}$.

Klösgen measures. Figure 2 (c) shows the results for the Klösgen measures. In the region from 0.1 to 0.4 the accuracy increases continuously until it reaches a global optimum at 0.4323, which achieves an average accuracy of almost 85 %. After the second iteration of the SearchBestParameter algorithm, no better candidate parameters than 0.4 were found. The accuracy decreases again with parametrizations greater than 0.6. As illustrated in Figure 1, the interval $[0, 1]$ describes the trade-off between *Precision* ($\omega = 0$) and WRA ($\omega = 1$), whereas values of $\omega > 1$ trade off between WRA and *Coverage*. The bad performance in this region (presumably due to over-generalization) surprised us, because we originally expected that the behavior that is exhibited by the

[2] Interestingly, the optimal value of $c = 0.342$ corresponds almost exactly to the micro-averaged default accuracy of the largest class (for both tuning and validation datasets). We are still investigating whether this is coincidental or not.



(a) F-measure (b) m-Estimate (c) Klösgen

Fig. 3. Isometrics of the best parameter settings

Klösgen measure for $\omega = 2$, namely to avoid low coverage regions, is preferable over the version with $\omega = 0.5$, which has a slight preference for these regions (cf. Figure 1).

***F*-measure.** For the F-measure the same interval as with the Klösgen measures is of special interest (Figure 2 (d)). Already after the first iteration, the parameter 0.5 turned out to have the highest accuracy of $82.2904\,\%$. A better one could not be found during the following iterations. After the second pass two other candidate parameters, namely 0.493 with $84.1025\,\%$ and 0.509 with $84.2606\,\%$ were found. But both of them could not be refined to achieve a higher accuracy and were therefore ignored. The main difference between the Klösgen measures and the F-measure is that for the latter, the accuracy has a steep descent at a very high parametrization of $1 \cdot E^9$. At this point it overgeneralizes in the same way as the Klösgen measures or the cost measures.

***m*-estimate.** The behavior of the m-estimate differs from the other parametrized heuristics in several ways. In particular, it proved to be more difficult to search. For example, we can observe a small descent for low parameter settings (Figure 2 (e)). The main problem was that the first iteration exhibited no clear tendencies, so the region in which the best parameter should be could not be restricted.

As a consequence, we re-searched the interval $[0, 35]$ with a smaller increment of 1 because all parameters greater than 35 got accuracies under $85.3\,\%$ and we had to restrict the area of interest. After this second iteration there were 3 candidate parameters, from which 14 achieves the greatest accuracy. After a second run, 23.5 became optimal, which illustrates that it was necessary to maintain a list of candidate parameters. After a few more iterations, we found the optimal parameter at 22.466. The achieved accuracy of $85.87\,\%$ was the optimum among all heuristics.

6.2 Behavior of the Optimal Heuristics

In this section, we compare the parameters which have been found for the five heuristics (cf. also Table 1). In terms of macro-averaged accuracy, the m-estimate and the relative cost measure clearly outperformed the other parametrized heuristics, as well as a few standard heuristics, which we had also briefly mentioned in section 3.3). Interestingly, the relative cost measure performs much worse with respect to micro-averaged accuracy, indicating that it performs rather well on small datasets, but worse on larger

Table 1. Comparison of various results of the optimal parameter settings of the five heuristics (identified by their parameters), other commonly used rule learning heuristics, and JRip (Ripper) with and without pruning, sorted by their macro-averaged accuracy

(a) on the 27 tuning datasets

Heuristic	average accuracy		average	
	Macro	Micro	Rank	Size
$m = 22.466$	85.87	93.87 (1)	4.54 (1)	36.85 (4)
$c_r = 0.342$	85.61	92.50 (6)	5.54 (4)	26.11 (3)
$\omega = 0.4323$	84.82	93.62 (3)	5.28 (3)	48.26 (8)
JRip	84.45	93.80 (2)	5.12 (2)	16.93 (2)
$\beta = 0.5$	84.14	92.94 (5)	5.72 (5)	41.78 (6)
JRip-P	83.88	93.55 (4)	6.28 (6)	45.52 (7)
Correlation	83.68	92.39 (7)	7.17 (7)	37.48 (5)
WRA	82.87	90.43 (12)	7.80 (10)	14.22 (1)
$c = 0.437$	82.60	91.09 (11)	7.30 (8)	106.30 (12)
Precision	82.36	92.21 (9)	7.80 (10)	101.63 (11)
Laplace	82.28	92.26 (8)	7.31 (9)	91.81 (10)
Accuracy	82.24	91.31 (10)	8.11 (12)	85.93 (9)

(b) on the 30 validation datasets

Heuristic	average accuracy		average	
	Macro	Micro	Rank	Size
JRip	78.98	82.42 (1)	4.72 (1)	12.20 (2)
$c_r = 0.342$	78.87	81.80 (3)	5.28 (3)	25.30 (3)
$m = 22.466$	78.67	81.72 (4)	4.88 (2)	46.33 (4)
JRip-P	78.50	82.04 (2)	5.38 (4)	49.80 (6)
$\omega = 0.4323$	78.46	81.33 (6)	5.67 (6)	61.83 (8)
$\beta = 0.5$	78.12	81.52 (5)	5.43 (5)	51.57 (7)
Correlation	77.55	80.91 (7)	7.23 (8)	47.33 (5)
Laplace	76.87	79.76 (8)	7.08 (7)	117.00 (10)
Precision	76.22	79.53 (9)	7.83 (10)	128.37 (12)
$c = 0.437$	76.11	78.93 (11)	8.15 (11)	122.87 (11)
WRA	75.82	79.35 (10)	7.82 (9)	12.00 (1)
Accuracy	75.65	78.47 (12)	8.52 (12)	99.13 (9)

datasets. These two heuristics also outperform JRip (the Weka-implementation of Ripper ([3])) on the tuning datasets, but, as we will see further below, this performance gain does not quite carry over to new, independent datasets.

Figure 3 shows the isometrics of the best parameter settings of the m-estimate, the F-measure, and the Klösgen-measure.[3] Interestingly, we can see that—within the confinements of their different functionals—all measures try to implement a very similar heuristic. Minor differences are detectable in the low coverage region, where the F-measure is necessarily parallel to the N-axis and the isometrics of the Klösgen measures are slightly bended.

6.3 Validity of the Results

In order to make sure that our results are not only due to overfitting of the 27 tuning datasets, we also evaluated the found parameter values on 30 new validation datasets. The results are summarized in Table 1 for both the tuning datasets (left) and the test datasets (right). The numbers in brackets describe the rank of each heuristic according to the measure of the respective column.

Qualitatively, we can see that the relative performance of the heuristics in comparison to each other, and in comparison to the standard heuristics does not change much, with the exception of the considerably better performance of JRip, which indicates that some amount of overfitting has happened in the optimization phase. However, the performance of the best metrics is still comparable to the performance of JRip, although the latter achieves this performance with much smaller rule sizes.

Figure 4 displays a comparison of all classifiers done with the Nemenyi test suggested in ([4]). All tuned heuristics (except the cost measure) outperform the standard heuristics which is indicated by the large gap between them. The Klösgen measure is the only parametrized heuristic which is not significantly better than the Accuracy heuristic.

[3] Because of space limitations, we omit the corresponding figures for the cost metrics, but they are just parallel lines with slopes that are determined by their respective optimal parameter values (and, in the case of the relative cost measure, also by the class distribution).

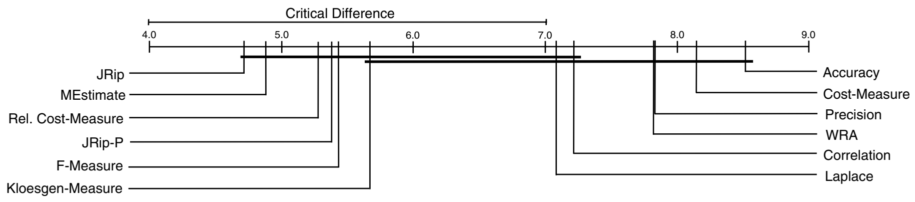


Fig. 4. Comparison of all classifiers against each other with the Nemenyi test. Groups of classifiers that are not significantly different (at $p = 0.05$) are connected.

7 Conclusions

The experimental study reported in this paper has provided several important insights into the behavior of greedy inductive rule learning algorithms. First, we have determined suitable default values for commonly used parametrized evaluation metrics such as the m-estimate. This is of considerable practical importance, as we showed that these new values outperformed conventional search heuristics and performed comparably to the RIPPER rule learning algorithm. Second, we found that heuristics which take the class distribution into account (e.g., by evaluate relative coverage instead of absolute coverage) outperform heuristics that ignore the class distribution (e.g., the F-measure which trades off recall and precision). Third, however, we found that for a good overall performance, it is necessary to weight the true positive rate more heavily than the false positive rate. This is most obvious in the optimal parameter value for the relative cost metric, but can also be observed in other well-performing heuristics, whose isometrics have a very steep slope in the important regions. Last but not least, we think that this has been the most exhaustive experimental comparison of different rule learning heuristics to date, yielding new insights into their comparative performance.

However, our results also have their limitations. For example, we have only evaluated overall performance over a wide variety of datasets. Obviously, we can expect a better performance if the parameter values are tuned to each individual dataset. We think that the good performance of RIPPER is due to the flexibility of post-pruning, which allows to adjust the level of generality of a rule to the characteristic of a particular dataset. We have deliberately ignored the possibility of pruning for this set of experiments, because our goal was to gain a principal understanding of what constitutes a good rule evaluation metric for separate-and-conquer learning. It is quite reasonable to expect that pruning strategies could further improve this performance. In particular, it can be expected that the performance of parameter values that result in slight overfitting can be considerably improved by pruning (whereas pruning can clearly not help in the case of over-generalization). We are currently investigating this issue.

Acknowledgements

This research was supported by the *German Science Foundation (DFG)* under grant FU 580/2-1.

References

[1] Asuncion, A., Newman, D.: UCI machine learning repository (2007), http://www.ics.uci.edu/~mlearn/{MLR}epository.html

[2] Cestnik, B.: Estimating probabilities: A crucial task in Machine Learning. In: Aiello, L. (ed.) Proceedings of the 9th European Conference on Artificial Intelligence (ECAI 1990), Stockholm, Sweden, pp. 147–150. Pitman (1990)

[3] Cohen, W.W.: Fast Effective Rule Induction. In: Prieditis, A., Russell, S. (eds.) Proceedings of the 12th International Conference on Machine Learning, Tahoe City, CA, July 9–12, 1995, pp. 115–123. Morgan Kaufmann, San Francisco (1995), http://citeseer.nj.nec.com/cohen95fast.html

[4] Demsar, J.: Statistical comparisons of classifiers over multiple datasets. Machine Learning Research (7), 1–30 (2006)

[5] Fürnkranz, J.: Separate-and-Conquer Rule Learning. Artificial Intelligence Review 13(1), 3–54 (1999), citeseer.ist.psu.edu/26490.html

[6] Fürnkranz, J., Flach, P.A.: ROC 'n' Rule Learning - Towards a Better Understanding of Covering Algorithms. Machine Learning 58(1), 39–77 (2005), http://www.cs.bris.ac.uk/Publications/Papers/2000264.pdf

[7] Janssen, F., Fürnkranz, J.: An empirical quest for optimal rule learning heuristics. Technical Report TUD-KE-2008-01, Knowledge Engineering Group, TU Darmstadt (2008), http://www.ke.informatik.tu-darmstadt.de/publications/reports/tud-ke-2008-01.pdf

[8] Janssen, F., Fürnkranz, J.: On meta-learning rule learning heuristics. In: Proceedings of the 7th IEEE Conference on Data Mining (ICDM 2007), Omaha, NE, pp. 529–534 (2007)

[9] Klösgen, W.: Problems for Knowledge Discovery in Databases and their Treatment in the Statistics Interpreter Explora. International Journal of Intelligent Systems 7, 649–673 (1992)

[10] Salton, G., McGill, M.J.: Introduction to Modern Information Retrieval. McGraw-Hill, Inc., New York (1986)

[11] Todorovski, L., Flach, P., Lavrac, N.: Predictive performance of weighted relative accuracy. In: Zighed, D.A., Komorowski, J., Żytkow, J.M. (eds.) PKDD 2000. LNCS (LNAI), vol. 1910, pp. 255–264. Springer, Heidelberg (2000), http://www.cs.bris.ac.uk/Publications/Papers/1000516.pdf

[12] Witten, I.H., Frank, E.: Data Mining — Practical Machine Learning Tools and Techniques, 2nd edn. Morgan Kaufmann Publishers, San Francisco (2005), http://www.cs.waikato.ac.nz/~ml/weka/

[13] Wrobel, S.: An Algorithm for Multi-relational discovery of Subgroups. In: Komorowski, J., Żytkow, J.M. (eds.) PKDD 1997. LNCS, vol. 1263, pp. 78–87. Springer, Heidelberg (1997)

Learning Model Trees from Data Streams

Elena Ikonomovska[1] and Joao Gama[2]

[1] FEIT – Ss. Cyril and Methodius University, Karpos II bb, 1000 Skopje, Macedonia
[2] LIAAD/INESC, FEP – University of Porto, Rua Campo Alegre 823, 4150 Porto, Portugal
elenai@feit.ukim.edu.mk, jgama@fep.up.pt

Abstract. In this paper we propose a fast and incremental algorithm for learning model trees from data streams (FIMT) for regression problems. The algorithm is incremental, works online, processes examples once at the speed they arrive, and maintains an any-time regression model. The leaves contain linear-models trained online from the examples that fall at that leaf, a process with low complexity. The use of linear models in the leaves increases its any-time global performance. FIMT is able to obtain competitive accuracy with batch learners even for medium size datasets, but with better training time in an order of magnitude. We study the properties of FIMT over several artificial and real datasets and evaluate its sensitivity on the order of examples and the noise level.

1 Introduction

Recently a new class of emerging applications has become widely recognized: applications in which data is generated at very high rates in the form of transient data streams. Examples of such applications include financial applications, network monitoring, security, telecommunication data management, web applications, manufacturing, sensor networks, and many others. The rapid generation of continuous streams of information has challenged the storage, computation and communication capabilities of computing systems. These vast amounts of data, arriving in high speeds need employment of data mining techniques for near-real time analysis and extraction of hidden knowledge. However, traditional data mining techniques cannot be directly applied to data streams. This is because most of them require that the data resides on disk or in memory, performing multiple or sequential scans over the data, with an assumption that the training set is finite and stationary.

An ideal knowledge discovery system operating in a typical streaming scenario, would have to incorporate the new information at the speed it arrives, incrementally update the decision model to the most recent data and supply the user with an any-time ready-to-use model. The system cannot have control over the order of examples and can process each example only once, as it must be discarded afterwards to satisfy the requirement for mining unbounded and rapid data streams. Such desired properties can be fulfilled by incremental learning algorithms, also known as online, successive or sequential methods. One very interesting research line in the field of incremental tree induction, suggests maintaining sufficient statistics required for computing the merit of a split at each decision node and performing the split decision only when there is enough statistical evidence in favor of a particular split test [1][2].

Regression trees similarly as decision trees have the desirable properties of being both accurate and easy to interpret. They are known for their simplicity and efficiency when dealing with domains with large number of variables and cases. Therefore they are considered as an important class of regression models. Regression trees are obtained using a fast divide and conquer greedy algorithm that recursively partitions the given training data into smaller subsets and recursively applies the same strategy to all of the subsets. The result is a tree shaped model with splitting rules in the internal nodes and predictions in the leaves. They are a natural generalization of decision trees for regression problems. Their successors, the model trees however are even more accurate because they use linear regression to fit models locally to particular areas of instance space instead of using a constant value. This makes them a more suitable choice in the regression domain.

In this paper we propose a fast and incremental algorithm for building a linear model tree (FIMT) from data streams. The linear models in the leaves of the tree are obtained by incrementally training a single layer neural network, a perceptron, using the incremental gradient descent method. The proposed algorithm can guarantee high asymptotic similarity of the incrementally learned tree to the one learned in a batch manner if provided with enough data. FIMT does not require storing of examples, since examples are seen only once and discarded afterwards. The only memory that is required is for storing the sufficient statistic for building the tree and the memory needed to store the model. It is an any-time algorithm in a sense that it can offer to the user a ready-to-use model tree only after a reasonable number of examples are seen. The accuracy of the model improves smoothly in time, as new information is incorporated in the tree incrementally updating the model to the most recent data. This incremental algorithm builds a competitive model tree faster than any batch algorithm for building model trees proposed thus far. We assume that the underlying distribution of the data is stationary.

The paper is organized as follows. In the next section we give an overview of the related work in the field. Then we describe the FIMT algorithm and present an empirical evaluation and sensitivity analysis. The paper concludes with the last section, outlining the main contributions of our work.

2 Related Work

2.1 Batch Learning of Model Trees

Standard algorithms for building models trees use the standard divide-and-conquer approach. This approach assumes that the training set is finite and stationary, and requires all the training data to be available before the learning process begins. The process starts with building a regression tree, and only after it assigns linear models in the tree leaves. The building phase consists of recursively splitting the instance space until the termination condition is met. The system typically chooses the split that maximizes some error reduction measure, with respect to the examples that fall at the current node. In Quinlan's algorithm M5 [3] the author has used as evaluation measure the standard deviation reduction measure. A different and more computationally expensive approach is proposed in [4] using a measure that corresponds to the distance from a linear regression plane. The termination criterion is met when the node is pure, i.e. the *y* values (values of predicted attribute) of all the instances that reach the node vary very slightly or when only a few instances remain. After the tree is grown in the pruning phase linear models are computed from the attributes referenced by tests in the pruned branch, using standard regression techniques.

2.2 Incremental Learning of Model Trees

Although regression trees as well as model trees are an interesting and efficient class of learners, little research has been done in the area of incremental regression or model tree induction. To our best knowledge there is only one paper addressing the problem in hand, i.e. [5]. The authors follow the method proposed by Siciliano and Mola [6] applying it in an incremental way. The approach is based on the nice idea of using statistical tests that can guarantee a stable splitting decision, supported by enough statistical evidence. The statistical test that is being used is based on the residual sums of squares which can be computed incrementally as new examples are seen. As the authors state, the key advantage of using such statistical test in an incremental implementation is that, if there is not enough evidence to discount the null hypothesis (of not splitting) with the desired degree of confidence, no split will be made until further evidence is accumulated.

The rest of the related work is in the field of incremental decision tree learning. Since model trees derive from the basic divide-and-conquer decision tree methodology, examining the relevant literature in incremental decision tree induction can give insights into how an incremental method for model tree induction can be developed.

2.3 Incremental Learning of Decision Trees

The problem of incremental decision tree induction fortunately has received the proper attention from the data mining community. There is a large literature on incremental decision tree learning but our focus of interest is on one particular research line initiated by Musick, Catlett and Russell [7]. The authors in their work have noted that only a small sample from the distribution might be enough in order to determine the best splitting test in a decision node. The answer to the question how many examples would be needed to confidently state that one attribute is better than the rest, lies in using statistical tests that would provide enough evidence to justify the decision. Examples of such algorithms are the Gratch's Sequential ID3 [1], VFDT [2] and the VFDTc algorithm [8]. The Sequential ID3 algorithm is a sequential decision tree induction method which guarantees similarity of the learned tree to the batch tree based on a statistical procedure called the sequential probability ratio test, but its guarantee is much looser than the one of the Hoeffding tree algorithm (VFDT). On the other hand, the VFDT system is considered as state-of-the-art and represents one of the best known algorithms for classifying streams of examples. The system gives a strong guarantee that the produced tree will be asymptotically arbitrarily close to the batch tree, given that enough examples are seen [2]. This is achieved by using a statistical bound, known as the Hoeffding bound. This method has proven very successful in deciding when to expand the tree and on which attribute. However, if two attributes have continuously similar values for the evaluation function, the Hoeffding bound would need potentially large number of examples to determine the best one with high confidence. To avoid this situation, the authors have proposed a tie breaking mechanism based on a user defined parameter τ, which determines the level of error that the user is willing to accept. When the situation is recognized as a tie, the leaf is transformed into a node, and the chosen splitting attribute is the current best one. Since the most computationally expensive operation is re-computing the evaluation measure G(·) and it has proven un-efficient to perform the computation after every new example, the authors proposed a user defined parameter n_{min} to specify the minimum number of examples that must be accumulated in a node before G is recomputed. Further more, the system has memory management feature which will deactivate the least promising leaves when system RAM is nearly all consumed. Reactivation is possible when the resources are available again. Some memory can be freed also by deactivating poor attributes early in the process of split evaluation. Our algorithm was implemented using the VFML library of Domingos [9], and therefore it came natural to adopt many of the proposed features of VFDT. In the next section we describe the theoretical foundations and the properties of the FIMT algorithm.

3 Fast and Incremental Model Tree Learner

A crucial observation of several authors is that, in order to determine the attribute to test at a given node of a decision tree, instead of examining all the examples it may be sufficient to examine only a subset of the training examples that pass trough that node. This is the basis of our incremental approach. After examining a specified number of examples, a split can be performed and the following examples can be used for the children nodes created with the split. Exactly how many examples are necessary at each node so we can be sure that our decision is stable, can be determined using statistical bounds. We propose the use of the Chernoff bound as a statistical support of the splitting decision. To calculate the measure of merit that determines the split, the algorithm must maintain sufficient statistics for every possible split point. The measure that we use is the standard deviation reduction (SDR) which is also used in the batch algorithm M5. The formula for the SDR measure is given below:

$$SDR = sd(T) - \sum_i \frac{|T_i|}{|T|} sd(T_i) \text{, and } sd(T) = \sqrt{\frac{1}{N}\left(\sum_{i=1}^{N} y_i^2 - \frac{1}{N}\left(\sum_{i=1}^{N} y_i\right)^2\right)} \quad (1)$$

where N is the number of examples which have passed through the corresponding leaf node of the model tree. The SDR measure has low computational complexity and the necessary statistics can be incrementally updated with every example from the stream. It becomes clear that we need to maintain the sums of y values (values of the predicted attribute) and squared y values, as well as the number of examples that have passed through that leaf node. The incremental algorithm doesn't assume a termination condition. Therefore, the process of computing linear models in the leaves must be performed simultaneously, while building the tree. We propose training neural networks without hidden layers - perceptrons in the leaves of the regression tree. The perceptrons can be trained in an incremental manner with low time complexity, which is linear with the dimensionality of the problem. The resulting algorithm incrementally maintains a linear model tree, updating the tree with every new example from the data stream. It doesn't need to store any examples and provides an up-to-date model at any instant. The processing time per example is dependent only on the dimensionality of the problem and the domain of the attributes and not on the number of previously seen examples. Therefore, the algorithm can mine very big data sets very efficiently. In the next subsections we describe the main properties of the algorithm.

3.1 Numerical Attributes

The ability to efficiently learn from numerical data is a very attractive property of a learning algorithm, especially if the problems that are being tackled are most naturally expressed by continuous numeric values. The common approach in the batch setting is to perform a preprocessing phase, typically consisted of discretizing the range of numerical attributes. Common data discretization techniques require an initial pass of the data prior to learning and this is not viable in the streaming setting. We address this problem in our system proposing a time efficient method for handling numerical attributes, which is based on a similar one proposed in [8].

The incremental algorithm builds the tree following a top-down approach. Every node corresponds to a time window of training examples with a different size. After performing a split, the succeeding examples are passed down to the leaves of the tree, where we update the necessary statistics for determining the next split. Common techniques for choosing the best splitting threshold must perform sorting of attribute values, to check all split points between the distinct values. When mining data streams, sorting is an expensive operation and has to be done efficiently and incrementally. The proposed method maintains a binary tree structure (Btree) located in the leaf, which tracks all the possible splitting points in the observed range of a numerical attribute. The Btree structure enables sorting values on the fly, as well as maintaining sufficient statistic for computing the measure of merit. It is an acyclic tree shaped graph consisted of nodes and branches. Except the leaf nodes, all the nodes in the structure have two branches leading to two other nodes. Each node of the structure holds a test of the form $value \leq key$. The keys are unique values from the domain of the corresponding numerical attribute. For maintaining sufficient statistics, in each node of the Btree structure are stored two arrays consisted of three elements. The first array maintains counter of the number of instances that have passed through this node with values less than or equal to the key in the node, as well as sum of y values and sum of squared y values. The second array maintains these statistics for the instances that passed through it but with values greater than the key in the node. The Btree is incrementally updated with every new example that has reached that leaf of the model tree. If the example has new unseen value in the Btree, this value is inserted by traversing the tree, till the node with the smallest key bigger than the y value of the example. The tree is traversed starting from the root node following the branches that correspond to the results of the tests in the nodes. While traversing the tree all the counters of the nodes on the way are updated. For each continuous attribute the system maintains a separate Btree structure. When an example reaches a leaf in the model tree, all the binary tree structures maintained in this leaf are being updated. Insertion of a new value in the structure has time complexity on average $O(log\ n)$ and

O(n) in the worst case, where n represents the number of unique values seen thus far in the leaf. When evaluating an attribute the whole tree must be traversed from left to right, to compute the SDR measure for each possible split point from a sorted sequence of the observed attribute values. This is easy to compute because all the nodes in the structure have the information about how many examples have passed that node with attribute values less or equal and greater than the key, as well as the sums of their y values. The way of traversing the Btree is by visiting first the left child node, second the parent node and last the right child node. The structure must be traversed only once per each attribute and the best splitting point can be computed on the fly. This operation has a time complexity of O(n), where n is the number of nodes or distinct values of the attribute.

3.2 Splitting Criteria

The splitting criteria is one of the most important aspects in the incremental induction of a model tree. We use the Chernoff bound to determine the point when there is enough statistical evidence accumulated in favor of a particular splitting attribute. The Chernoff bound is independent from the distribution. It uses the sum of independent random variables X and gives a relative or absolute approximation of the deviation of X from its expectation μ. The original form of the bound although stronger, is not handy to compute. Instead, we use a corollary given in [10]. Considering the equations $\varepsilon = \beta\overline{\mu}$ and $\overline{\mu} = \mu/n$, we obtain the following bound:

$$\Pr[\overline{X} - \overline{\mu} > \varepsilon] \leq \begin{cases} e^{-\frac{n\varepsilon^2}{2\overline{\mu}+\varepsilon}}, & \frac{\varepsilon}{\overline{\mu}} \geq 1 \\ e^{-\frac{n\varepsilon^2}{3\overline{\mu}}}, & 0 < \frac{\varepsilon}{\overline{\mu}} < 1 \end{cases} \tag{2}$$

where n is the number of random variables. The Chernoff bound states that, with probability 1 - δ, the true mean of a random variable is at least $\overline{\mu} - \varepsilon$, where

$$\varepsilon = \sqrt{\frac{3\overline{\mu}}{n}\ln\left(\frac{2}{\delta}\right)} \tag{3}$$

Let G(·) be the heuristic evaluation measure, used to determine the best attribute to split (this measure can be any adequate measure of merit). Assuming G is to be maximized, and let x_a be the attribute with the highest observed $\overline{G}(x_a)$ after seeing n examples, x_b the attribute with the second highest $\overline{G}(x_b)$ and $\Delta\overline{G} = \overline{G}(x_a) - \overline{G}(x_b) \geq 0$ be the difference between their observed values. Given a user specified parameter δ, the Chernoff bound can be used to guarantee with probability 1 - δ that if $\Delta\overline{G} > \varepsilon$ then attribute x_a is the correct choice. If after seeing n examples the observed difference between the best and the second-best attribute is greater than the error ε, the true mean difference will be $\Delta\overline{G}_{\text{true}} \geq \Delta\overline{G} - \varepsilon$ and from there $\Delta\overline{G}_{\text{true}} \geq 0$. Therefore, the Chernoff bound guarantees with probability 1 - δ, that the attribute x_a has a higher observed value for the measure of goodness. The observed difference can be seen as an average over the examples seen at that leaf and is used as the expected average $\overline{\mu}$ in (5). After determining the best attribute to split, the leaf is transformed into a decision node and the process is repeated for the new leaf nodes. The succeeding examples are passed down to the corresponding leaves and used to determine the best attributes to split at that level. Obviously, each splitting decision corresponds to a window frame of a certain number of examples that passed through that node while it was a leaf. If the distribution is stationary, with the guarantee of the Chernoff bound the tree built will be very similar to a batch one.

3.3 Linear Models in the Tree Leaves

In parallel with growing the tree is the process of building the linear models in the leaves of the model tree. This process is online and incremental. The base idea is to learn perceptrons in the leaves of the model tree using an online method for updating the weights. The trained perceptrons will represent the linear models with parameters equal to the values of the weights that correspond to the links with each of the attributes. We propose the incremental gradient descent method which enables us to update the weights of the perceptron links with every new example seen at the leaf. This method, based on the delta rule can converge even in the case when the examples that fall into the leaf are not linearly separable. The weights are initially set for the root node to random real valued numbers in the range [-1, 1]. When a new example comes we compute the output with the current weights and update each weight with the product of the difference between the output (o) and the real value (y), the normalized value of the corresponding attribute (x_i) and the learning rate η. This is best represented with the following formula:

$$w_i = w_i + \eta(o - y)x_i \ . \tag{4}$$

The learning phase of the perceptrons is in parallel with the process of growing a node, and ends when the node splits. The obtained linear model in the splitting node is passed down to its children, avoiding the need to train the perceptrons in the children nodes form a scratch. The learning of parent's perceptron continues in leaf nodes, independently and according to the examples that fall in the leaves. This can be seen as fine tuning of the linear model in the corresponding instance space. The learning rate can be kept constant or it can decrease with the number of examples seen during the process of learning. The perceptrons can handle only numerical attributes. The values of attributes should be normalized before the process of learning, so each of them will have the same influence during the process of training. This can be done incrementally by maintaining the necessary statistics. Categorical variables should be transformed into a set of artificial numerical variables before the process of learning.

4 Evaluation

In this section we present the results from the performed evaluation and sensitivity analysis of the FIMT learner. The evaluation was designed to cover several aspects of FIMT's general performances. We have performed comparison of FIMT with the equivalent incremental regression tree algorithm (FIRT – as prediction we use the mean of the examples that fall to the corresponding leaf), its batch version (BRT – forces splitting as long as the nodes are not "pure") and the Friedman's algorithm CART [12] over the accuracy, the model size, the memory allocation and the training time. As tools we have used bias-variance decomposition of the error and learning curves. We have also analyzed their sensitivity with respect to noise and the order of examples.

The experiments were performed using six well known artificial datasets: We have used the Fried dataset used by Friedman in [11]. It contains 10 continuous predictor attributes with independent values uniformly distributed in the interval [0, 1]. The Cart dataset proposed by Breiman et al. in [12] with 10 attributes all of them discrete. The 3DSin dataset used by A. Dobra and J. Gehrke in [13] and the 2DTest dataset used by S. Schaal and C. Atkeson [14], both containing two continuous attributes. Losc and Lexp datasets proposed by A. Karalic in [4] which contain five predictor attributes, one discrete and four continuous. FIMT is designed for mining huge, possibly unbounded datasets at high speeds. A system like FIMT is not very useful for small datasets. The lack of real world datasets big enough to test FIMT's performance has restricted us to only three datasets from the UCI Machine Learning Repository [15]. In our evaluation we have used the California Housing dataset, Census 8L, and Census 16H datasets. For all the experiments, we used initial parameter values of $\delta = 1\times10^{-6}$, $\tau = 0.001$ and $n_{min} = 300$. Algorithms were run on a AMD/2GHz. All of the results are averaged over 10 runs.

Figure 1 shows the accuracy of the learners averaged over 10 runs on the Fried dataset, which is the most complex one. Due to the memory and time constraints, the maximum number of examples for the batch learners was limited to 100k examples. The learning curves show that BRT has a clear advantage for the first 100k examples. CART is outperformed by all of the other learners. FIMT obtains the same accuracy with 340k examples and continues to improve its performance. The advantage of the batch learner can be explained due to the fact that the incremental algorithm uses each example only once, while the batch algorithm reuses the same example at multiple levels of the tree while making splitting decisions. The curves also show the apparent advantage in accuracy of the model tree over the regression tree, whose prediction strategy is the average of *y* values of the examples that have passed through that leaf. For mining 100k examples of the Fried dataset, BRT dynamically allocates around 170MB of memory, CART requires ~75MB, while FIRT/FIMT only ~20MB. On all of the artificial datasets FIMT consistently outperformed FIRT. More details are presented in Table 1. ASE stands for average squared error, STD stands for standard deviation and RMSE is abbreviation of relative mean squared error.

Figure 1 also shows the number of nodes of the trees induced by the learners, averaged over 10 runs on the Fried dataset. FIRT and FIMT generate exactly the same tree model except for the linear models in the leaves. Notice that the incremental learners when compared to their batch equivalent generate trees with substantially smaller number of nodes, which are able to achieve better accuracy given enough big datasets. The time required to learn from 100k examples of the Fried dataset for the batch learner (BRT) is around 310 seconds, for CART around 350 seconds, while for the incremental learners around 17 seconds.

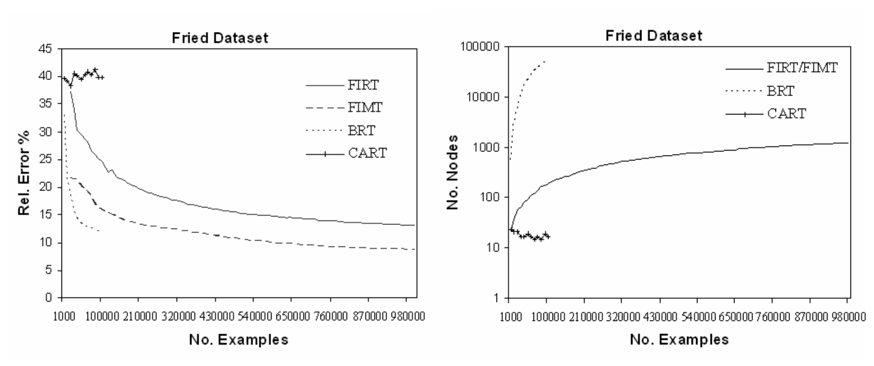


Fig. 1. Relative error and model size of FIMT, FIRT, BRT and CART as function of the number of examples on the Fried dataset

Table 1. Averaged results over ten runs for FIRT and FIMT on the artificial data sets for training sets of 1000k and test sets of 300k examples. The best results are bolded.

	FIRT					FIMT				
	ASE	STD	RMSE	Bias	Var.	ASE	STD	RMSE	Bias	Var.
3DSin	0.02817	0.00009	0.0115	0.0126	0.0165	0.02247	0.00008	**0.0091**	0.01116	0.0113
Cart	1.00076	0.00183	0.0519	0.99996	0.0006	1.00296	0.00259	**0.0518**	0.99938	0.0035
Fried	3.34584	0.00669	0.1353	2.19333	1.152	2.28359	0.00756	**0.0919**	1.81086	0.4726
2DTest	1.0071	0.00261	0.8835	1.00282	0.004	1.00658	0.00260	**0.8831**	1.00296	0.0035
Lexp	0.17841	0.00094	0.0295	0.08388	0.0945	0.00741	0.00006	**0.0012**	0.00384	0.0036
Losc	0.17776	0.00061	0.0284	0.16021	0.0175	0.14848	0.00057	**0.0238**	0.13906	0.0093

Table 2. Comparison of learners when given the same number of examples (100k/30k)

	BRT			FIRT			FIMT			CART		
	RMSE	Nodes	Mem.	RMSE	Nodes	Mem.	RMSE	Nodes	Mem.	RMSE	Nodes	Mem.
Fried	**0.117**	55243.8	165.38	0.194	404.6	21.47	0.138	377	20.09	0.399	17	75
Lexp	0.006	54682.4	108.61	0.045	429.4	10.28	**0.002**	434.6	10.05	0.146	26.8	45
Losc	0.0247	54662.8	108.6	0.028	471.8	10.54	**0.0244**	472.2	10.55	0.057	5	45
Cart	0.063	22780.2	83.1	**0.052**	107	0.2	0.062	107.4	0.21	0.120	23	55
2DTest	1.384	51215.4	73.53	0.89	87.8	4.43	**0.889**	87.6	4.3	0.932	5	25
3DSin	**0.0002**	53990.2	72.87	0.011	437	4.89	0.01	424.4	4.77	0.121	34.6	25

Further, we have designed the evaluation considering three important aspect: the learning capacity, ability of learning with constrained computational resources and the ability of fast learning. To evaluate the learning capacity we have compared all the algorithms over the same sizes of training and testing datasets. We use training datasets of size 100k and testing datasets of size 30k. The results of the comparison are presented in Table 2. From the results we can see that the incremental learners, especially FIMT is competitive with its batch equivalent, and for most of the problems obtains better accuracy. CART is outperformed for all of the problems.

Table 3. Comparison FIMT and CART on the artifical datasets when given the same computational resources in terms of RAM. The best results are bolded.

	FIMT						CART		
Dataset	ASE	STD	RMSE	Nodes	Dynamic Mem.	Ex.	RMSE	Nodes	RAM
3DSin	0.0415	0.0002	**0.0168**	253.4	18.99	400k	0.12092	34.6	50
Cart	1.00296	0.0026	**0.0518**	107	0.392	~/100k	0.12022	23	80
Fried	3.4388	0.0187	**0.1385**	496.8	39.593	400k	0.39879	17	100
2DTest	1.00598	0.0032	**0.8833**	109.2	20.382	600k	0.93212	5	50
Lexp	0.0219	0.0004	**0.0036**	236.2	29.505	300k	0.14545	26.8	70
Losc	0.1652	0.001	**0.0264**	139.2	31.042	400k	0.05683	5	70

The second aspect of our evaluation concerns the ability of learning with constrained computational resources. For the CART algorithm we use datasets of sizes 100k/30k (train/test). We have measured the memory required for CART and constrained the incremental algorithm to the same amount of RAM. The number of examples that FIMT is able to mine for the same computational resources, as well as the average error, the standard deviation, the relative error, the model sizes and the dynamically allocated memory required for the necessary statistics and the tree structure (in MB) are given in Table 3. We can see that FIMT has significantly better results than CART. This can be due to the fact that FIMT uses its advantage to learn from the quantities of examples that are beyond the processing ability of the conventional batch learners like CART. For the Cart dataset the number of examples that FIMT can mine is very big (~30000k) but the accuracy remains the same as for 100k examples.

In order to study the capabilities of fast learning, we compared the learners with constrained time of learning. The time limit corresponds to the time required for the batch learner to mine 100k examples. The incremental algorithms are not just competitive with the batch equivalent, but for most of the problems they obtain substantially lower error. FIMT and FIRT, are able to mine from 4 to 8 times more examples that the batch ones in the same time, and therefore easily achieve better accuracy.

We performed a study of the sensitivity of learners on the noise in the data. The test set was fixed on 300k examples with 10% of noise. The training set was fixed on 1000k examples and the level of noise was systematically increased from 0 to 70%. We measure the accuracy and the model size. From the results we conclude that the incremental learners are more robust to the noise, responding with smaller increase of the relative error. The size of the tree induced by the incremental learners continuously decreases with the increase of the noise in the training data. This was expected, because the noise increases the uncertainty in the concept. The incremental learners delay their decisions until more evidence has been accumulated.

Incremental algorithms are known to be very dependent on the order of the examples. We have analyzed the effect of different order of examples over the performance of incremental learners on several artificial datasets. In our experiments the test and training datasets are set to 300k and 1000k correspondingly. We shuffled the training set ten times and measured the error, the learning time, the model size as well as the argument chosen at the root node of the learned tree. The results show that the variance of the relative error is of the order from 10^{-6} to 10^{-10}. The learning time is not affected as well as the model size. The attribute at the root remained the same in all experiments, except for the 3DSin dataset where the root attribute was *Att*1 or *Att*2 which have same information value with respect to the predicted attribute.

Very useful analytical tool is the bias-variance decomposition of the error [16]. The bias component of the error is an indication of the intrinsic capability of the method to model the phenomenon under study and is independent of the training set. The variance is independent of the true value of the predicted variable and measures the variability in the predictions given different training sets. Experiments were performed for all the learners with training sets of fixed size 100k and test sets of fixed size 60k. Different set of experiments was also performed only for the incremental learners with training sets of size 1000k and test sets of size 300k. The experimental methodology is the following: We generate ten independent training sets and log the predictions of the corresponding models over the same test set. Those predictions are then used to compute the bias and the variance using the derived formula for squared loss in [17]. Results show that the incremental learners exhibit significantly lower variance than the batch learner, while the bias component of the error improves more in time, given bigger datasets. This suggests that the splitting decisions are stable and lead to a more reliable model which is not dependent on the selection of the training examples.

We have done some comparison of the learners on three real datasets from the UCI repository. We use the Census8L, Census16H and California housing datasets. The training sets contain two thirds of the examples. The rest was left for the test set. The relative errors on the Census8L dataset are: 18% for BRT, 46% for FIRT and 72% for FIMT. For Census16H dataset are: 62%, 61% and 103%, and for California housing are: 30%, 38% and 56% for BRT, FIRT and FIMT accordingly. The results show that FIRT is competitive with BRT considering the small number of examples, but FIMT is not useful for small dataset. This is due to the fact that training of the perceptrons requires more examples to be seen at the leaf. Therefore, for small datasets the average value is better prediction strategy than using the linear models.

5 Conclusion

In this paper we propose FIMT, a fast and incremental algorithm for learning regression trees from data streams. The algorithm is able to learn in a very small time per example and the only memory it requires is for storing sufficient statistics at tree leaves. The splitting criteria uses Chernoff bound to guarantee stable decisions. Tree leaves are equipped with linear-models trained online. Empirical studies show the advantage, in terms of accuracy, of using linear models in the tree leaves. Moreover, this feature improves accuracy at any-time. The incremental algorithm is competitive with its batch equivalent even for medium sized datasets and exhibits lower values for the variance component of the error. The sensitivity analysis shows that the incremental learners are more robust to noise. The quality of the produced models is not dependent on the order of examples and suggests stable decisions. In this work we assumed stationary distributions. Improving the algorithm towards non-stationary distributions and dynamic environments is left for future work.

Acknowledgments. Thanks to the financial support of projects KDubiq and ALESII under contract POSC/EIA/55340/2004.

References

1. Gratch, J.: Sequential Inductive Learning. In: 13th National Conference on Artificial Intelligence, pp. 779–786. AAAI Press, Menlo Park (1996)
2. Domingos, P., Hulten, G.: Mining High Speed Data Streams. In: 6th International Conference on Knowledge Discovery and Data Mining, pp. 71–80. ACM Press, New York (2000)
3. Quinlan, J.R.: Learning with Continuous Classes. In: 5th Australian Joint Conference on Artificial Intelligence, pp. 34–348. Adams & Sterling (1992)
4. Karalic, A.: Employing Linear Regression in Regression Tree Leaves. In: 10th European Conference on Artificial Intelligence, pp. 440–441. John Wiley & Sons, Chichester (1992)
5. Potts, D., Sammut, C.: Incremental Learning of Linear Model Trees. J. Machine Learning 61, 5–48 (2005)
6. Siciliano, R., Mola, F.: Modeling for Recursive Partitioning and Variable Selection. In: Computational Statistics, pp. 172–177. R. Dutter & W. Grossmann (1994)
7. Musick, R., Catlett, J., Russell, S.: Decision Theoretic Sub-sampling for Induction on Large Databases. In: 10th International Conference on Machine Learning, pp. 212–219. Morgan Kaufmann, San Francisco (1993)
8. Gama, J., Rocha, R., Medas, P.: Accurate Decision Trees for Mining High-Speed Data Streams. In: The 9th International Conference on Knowledge Discovery and Data Mining, pp. 52–528. KDD Press (2003)
9. Hulten, G., Domingos, P.: VFML – A toolkit for mining high-speed time-changing data streams (2003), `http://www.cs.washington.edu/dm/vfml/`
10. Angluin, D., Valiant, L.G.: Fast Probabilistic Algorithms for Hamiltonian Circuits and Matchings. J. Computer and System Sciences 19, 155–193 (1979)
11. Friedman, J.H.: Multivariate Adaptive Regression Splines. J. The Annals of Statistics 19, 1–141 (1991)
12. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. Chapman & Hall/CRC, Belmont (1984)
13. Dobra, A., Gehrke, J.: SECRET: A Scalable Linear Regression Tree Algorithm. In: 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 481–487. ACM Press, New York (2001)
14. Schaal, S., Atkeson, C.: Constructive Incremental Learning From only Local Information. J. Neural Computation 10, 2047–2084 (1998)
15. Blake, C., Keogh, E., Merz, C.: UCI Repository of Machine Learning Databases (1999)
16. Breiman, L.: Arcing Classifiers. J. The Annals of Statistics. 26(3), 801–849 (1998)
17. Geman, S., Bienenstock, E., Doursat, R.: Neural Networks and the Bias/Variance Dilemma. J. Neural Computation 4, 1–58 (1992)

Empirical Asymmetric Selective Transfer in Multi-objective Decision Trees

Beau Piccart, Jan Struyf, and Hendrik Blockeel

Department of Computer Science, Katholieke Universiteit Leuven,
Celestijnenlaan 200A, 3001 Leuven, Belgium
{Beau.Piccart,Jan.Struyf,Hendrik.Blockeel}@cs.kuleuven.be
http://www.cs.kuleuven.be

Abstract. We consider learning tasks where multiple target variables need to be predicted. Two approaches have been used in this setting: (a) build a separate single-target model for each target variable, and (b) build a multi-target model that predicts all targets simultaneously; the latter may exploit potential dependencies among the targets. For a given target, either (a) or (b) can yield the most accurate model. This shows that exploiting information available in other targets may be beneficial as well as detrimental to accuracy. This raises the question whether it is possible to find, for a given target (we call this the main target), the best subset of the other targets (the support targets) that, when combined with the main target in a multi-target model, results in the most accurate model for the main target. We propose Empirical Asymmetric Selective Transfer (EAST), a generally applicable algorithm that approximates such a subset. Applied to decision trees, EAST outperforms single-target decision trees, multi-target decision trees, and multi-target decision trees with target clustering.

1 Introduction

There has been increasing interest recently in simultaneous prediction of multiple variables, also known as multi-target prediction or multi-objective prediction. In typical classification or regression problems, there is a single target variable that needs to be predicted as accurately as possible. In multi-target prediction, on the other hand, the input is associated with a vector of target variables, and all of them need to be predicted as accurately as possible.

Multi-target prediction is encountered for instance in ecological modelling where the domain expert is interested in (simultaneously) predicting the frequencies of different organisms in river water [1] or agricultural soil [2]. It can also be applied in multi-label classification tasks [3], where a set of labels is to be predicted for each instance instead of a single label; in prediction tasks with a structured output space [4], such as hierarchical multi-label classification [5], where the output is structured as a taxonomy of labels (e.g., newsgroups); and in multi-task or transfer learning, where knowledge gained from learning one task is reused to better learn related tasks [6].

It has been shown that multi-target models can be more accurate than predicting each target individually with a separate single-target model [6]. This is a consequence of the fact that when the targets are related (e.g., if they represent frequently co-occurring organisms in the ecological modelling applications mentioned above), they can carry information about each other; the single-target approach is unable to exploit that information, while multi-target models naturally exploit it. This effect is known as inductive transfer: the information a target carries about the other targets is transferred to those other targets. Note the connection with collective classification [7]: the latter exploits dependencies among targets of different instances, while multi-target models exploit dependencies among the multiple targets of the same instance.

Multi-target models do not, however, always lead to more accurate prediction. As we will show, for a given target variable, the variable's single-target model may be more accurate than the multi-target model. That is, inductive transfer from other variables can be beneficial, but it may also be detrimental to accuracy. Let us focus on one particular target and call this the main target. The subset of targets that, when combined with the main target in a multi-target model, results in the most accurate model for the main target, may be non-trivial, i.e., different from the empty set and from the set of all targets. We call this set the support set for the main target. This paper investigates how we can best approximate this set. Note that the two natural extremes of this approach are the single-target model (the support set is empty) and the full multi-target model (the support set includes all targets).

Based on the above observation, we propose Empirical Asymmetric Selective Transfer (EAST), a greedy algorithm that approximates the support set for a given main target. EAST has the following advantages over other approaches that try to exploit transfer selectively [8,9,10,11]: (a) EAST does not assume transfer to be symmetric (in fact, we show that transfer can be asymmetric), (b) EAST estimates transfer empirically and does not rely on heuristic approximations (we show that heuristics may poorly approximate transfer), (c) EAST does not make explicit assumptions about the distribution of the different target variables, and (d) EAST is a general method in the sense that it can be combined with any multi-target learner (other methods are often tied to a particular type of models, such as neural networks).

EAST is the main contribution of this paper. A second contribution is that we show that exploiting transfer selectively is useful in the context of decision trees; previous work focused on other model types such as neural networks [9] and k-nearest neighbor [8]. Decision trees have the well-known advantage over these methods that they are easy to interpret.

The rest of this paper is organized as follows. Sec. 2 introduces single- and multi-target prediction formally and defines our problem setting. Sec. 3 discusses known methods for multi-target prediction and inductive transfer. Sec. 4 describes multi-target decision trees, and shows that their so-called transfer matrix is asymmetric. This motivates EAST, which we introduce in

Sec. 5. Sec. 6 presents experiments with EAST, and Sec. 7 states the main conclusions.

2 Single/Multi-target Prediction and Problem Setting

Assume we have a dataset S containing couples $(\mathbf{x}, \mathbf{y})$ with $\mathbf{x} \in X$ the input vector and $\mathbf{y} \in Y = Y_1 \times \cdots \times Y_n$ the target vector. Denote with $y_i \in Y_i$ the i'th component of $\mathbf{y}$.

A single-target learner learns from a dataset $S = \{(\mathbf{x}, y_i)\}$, with $y_i \in Y_i$ a scalar variable, a function $f_i : X \rightarrow Y_i$ such that $\sum_{(\mathbf{x},y_i)\in S} L_i(f_i(\mathbf{x}), y_i)$ is minimized, with L_i some loss function over Y_i.

A multi-target learner learns from a dataset $S = \{(\mathbf{x}, \mathbf{y})\}$, with $\mathbf{y} \in Y$ an n-dimensional vector, a function $F : X \rightarrow Y$ such that $\sum_{(\mathbf{x},\mathbf{y})\in S} L(F(\mathbf{x}), \mathbf{y})$ is minimized, with L a loss function over Y. Assume that L is monotonically increasing in each of the L_i (i.e., whenever one L_i increases while the other $L_{(\cdot)}$ remain constant, L increases too). For example, $L(\mathbf{y}, \mathbf{y}') = \sum_i L_i(y_i, y'_i)$.

It has been shown that by using multi-target learners, better predictive performance for the targets, on average, can be obtained [6]. That is, for any $(\mathbf{x}, \mathbf{y})$ drawn randomly from the population, on average, $L(F(\mathbf{x}), \mathbf{y}) < L([f_1(\mathbf{x}), \ldots, f_n(\mathbf{x})], \mathbf{y})$.

Under the monotonicity assumption mentioned above, obtaining better predictive performance on average implies that there must be individual targets for which the predictive performance, as measured on this single target, must improve. That is, there must be at least one i for which $L_i(F_i(\mathbf{x}), y_i) < L_i(f_i(\mathbf{x}), y_i)$, with $F_i(\mathbf{x})$ the i'th component of $F(\mathbf{x})$. This observation leads to the question whether single-target models could be improved by following the multi-target approach. That is: even when there is only one single target that we want to predict, we may be able to build a better model for predicting that target if we can exploit the information present in other, related, variables.

Thus, the problem setting becomes as follows. We are given a training set $S = \{(\mathbf{x}, \mathbf{y})\}$, and are interested in predicting y_n from $\mathbf{x}$. The variables $y_i, i \neq n$ need not be predicted, and will not be available at prediction time, but we can use them during the learning phase. We call this setting the single-target setting with support targets: one single target y_n (the main target) needs to be predicted, but a number of additional support targets $y_i, i \neq n$ are available at induction time, and can be used to improve the model for y_n.

Note the following important point: while the support targets are used during learning, they are not assumed to be available during the prediction phase. If they were, then an alternative to the method proposed here would be to learn a model that also uses the support targets as inputs (e.g., in the case of decision tree learning, to learn a tree that is allowed to test these attributes). It is quite likely that that would lead to better prediction. The model that we will try to learn here, is one that will make predictions without having that information; we use the support targets to learn a model that maps $\mathbf{x}$ onto a target attribute more accurately, even though the model itself has no access to the support targets.

3 Related Work

We first discuss known algorithms for multi-target prediction. After that, we turn to methods that selectively exploit transfer by partitioning the target variables. Finally, we discuss methods that explicitly assume transfer to be asymmetric.

Probably the most influential work on multi-target prediction is that by Caruana [6]. He proposes a neural network based approach where the different target variables are outputs of one single network. Considering the network as containing n different predictive models (one for each output), one could say that these models share the connection weights between the input layer and the hidden layer, but the weights between the hidden and output layers are particular to each individual output. In this way, the network finds a balance between modeling the shared properties of the different targets and modeling their particularities.

Besides neural networks, many other predictive models have been extended to multi-target prediction as well. This includes nearest neighbor methods [6,8], kernel methods [12], Bayesian approaches [13], logistic regression [14], Gaussian processes [15], and decision trees [16].

Thrun & O'Sullivan [8] explicitly consider the fact that among the target variables, some may be related while others may be unrelated. Hence, better predictive performance may be obtained if multi-target models are built that only include those variables that are indeed related. To this aim, they first cluster the target variables, and then learn a separate multi-target model for each cluster. The variables are clustered based on an empirical measure of relatedness. More recently, also Bakker & Heskes [13], Xue et al. [14], and Evgeniou et al. [12] have proposed methods that are based on clustering targets.

As we will show with an example, transfer may be asymmetric and methods that cluster targets may therefore be suboptimal. The following two approaches assume transfer to be asymmetric and also consider the single-target prediction with support targets setting. Nevertheless, they don't measure transfer empirically. They are also not directly applicable to decision trees.

Silver & Mercer [9] build on the work of Caruana, but use a different learning speed (a parameter of the back-propagation algorithm) for each target in the neural network. They set the learning speed of the support targets based on their relatedness to the main target. In later work, they compare a number of relatedness measures, such as correlation and mutual information, to control the learning speeds [17].

Kaski & Peltonen [11] propose a probabilistic model for each support target that is a mixture of the main target's model and a target specific model. They set the parameters of these mixture models to minimize the conditional log likelihood summed over all targets. The idea is that the support target specific models "explain away" the irrelevant data, and that the relevant data available for the support targets helps improve the model for the main target. The approach is validated with logistic regression models as base models.

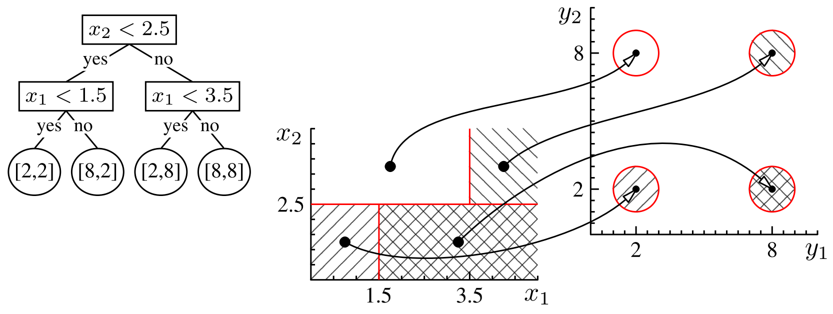


Fig. 1. A multi-target regression tree together with its mapping from the input to the target space. Multi-target regression trees work well if the training data maps hyper-rectangles in the input space to compact clusters in the target space.

4 Single-Target Prediction with Multi-target Trees

In this section, we briefly describe multi-target decision trees; we will use these as multi-target models in our experiments. Then, we study inductive transfer empirically for decision trees by constructing a so-called transfer matrix.

4.1 Multi-target Decision Trees

Most descriptions of decision tree learning assume a scalar target, which may be nominal (classification) or numerical (regression). Blockeel et al. [16] argued that decision tree learning can easily be extended towards the case of multi-target prediction, by extending the notion of class entropy or variance towards the multi-dimensional case. They define the variance of a set as the mean squared distance between any element of the set and the centroid of the set. Depending on the definition of distance, which could be Euclidean distance in a multi-dimensional target space, a decision tree will be built that gives accurate predictions for multiple target variables (Fig. 1 shows an example).

Similar to other multi-target models, it has been shown that multi-target trees can be more accurate than single-target trees. This holds for both multi-label classification trees [5] and for multi-objective regression trees [18].

4.2 The Transfer Matrix

To gain insight in the effect of applying multi-target models to single-target prediction, we construct a transfer matrix [8] $C = (c_{i,j})$, where $c_{i,j}$ is the expected gain in predictive performance for target y_i that the two-target model with targets y_i and y_j yields over the single-target model for y_i. In other words, $c_{i,j}$ indicates if inductive transfer from y_j to y_i is useful. We call $c_{i,j}$ the transfer from y_j to y_i.

Table 1 shows the transfer matrix for one of the datasets that we will use in the experimental evaluation. The multi- and single-target models on which the

Table 1. (a) Transfer matrix for the Soil Quality 1, Collembola groups dataset (a subset of S_2, see experimental setup). Cases where transfer is asymmetric are in italic. (b) Correlation matrix for this dataset.

(a)

$j \setminus i$	1	2	3	4	5
1	0	0.01	0.04	0.13	0
2	0	0	0.06	0.13	0
3	0.07	0.01	0	*0.13*	0
4	0.03	0	*-0.02*	0	*-0.01*
5	0.1	0.03	0.31	*0.18*	0

(b)

$j \setminus i$	1	2	3	4	5
1	1	0.03	0.12	0.05	0.07
2	0.03	1	0.08	0.27	0.45
3	0.12	0.08	1	0.31	0.14
4	0.05	0.27	0.31	1	0.19
5	0.07	0.45	0.14	0.19	1

matrix is based are regression trees. The matrix elements are the relative differences in 10-fold cross-validated Pearson correlation (a performance measure that is often used in regression problems) between the multi- and single-target model for y_i, averaged over 5 runs with different random folds; each row corresponds to a different support target y_j in the two-target model.

From the transfer matrix, we see that transfer is an asymmetric quantity: it is possible that y_i can be predicted more accurately if y_j is included as support target ($c_{i,j} > 0$), whereas the prediction for y_j actually deteriorates when y_i is included ($c_{j,i} < 0$).

While empirically measuring transfer as defined above is the most direct way of deciding which support target to use for predicting a given target, two important approximations to this approach have been used in previous work: (a) converting transfer into a symmetric quantity [8], and (b) using other measures, such as correlation, to somehow approximate transfer [17]. The advantage of such approximations is that they reduce the computational cost of the approach. For example, by combining (a) and (b), one could use the pairwise linear correlation between the targets to cluster the targets and then build a multi-target model for each cluster in the partition.

The disadvantage of (a) is that, because transfer is really asymmetric, replacing it by a symmetric approximation will result in suboptimal models for certain targets. Consider again Table 1. Clustering y_3 and y_4 together will result in a suboptimal model for y_3, while putting y_3 and y_4 in different multi-target models may result in a suboptimal model for y_4.

The disadvantage of (b) is that correlation is often not a good approximation for transfer. Table 1 shows the targets' correlation matrix. y_2 and y_5 have the highest correlation. Nevertheless, the transfer from y_2 to y_5 is zero and that from y_5 to y_2 is small. One reason is that transfer does not only depend on the values of the target variables. It also depends on other factors, such as the mapping that the data represents between input and output. Measures that only depend on the targets may therefore poorly approximate transfer. Fig. 1 illustrates this: the correlation between the targets is zero, yet both targets can be predicted accurately with the same tree structure. Therefore, we expect that data for both targets will be beneficial to finding this structure, and that $c_{1,2}$ and $c_{2,1}$ are positive.

These two disadvantages are alleviated by our approach, which we discuss next.

Algorithm 1. Empirical Asymmetric Selective Transfer (EAST).

```
1: input: dataset $S$, main target $t = y_n$, candidate support targets $T_s = \{y_i \mid i \neq n\}$.
2: $T^{(0)} := \{t\}$
3: $L^{(0)} :=$ cross-validate($T^{(0)}$, $L_n$, $S$)
4: for $i := 1$ to $n - 1$
5:     $L^{(i)} := \infty$
6:     for each $t_s \in (T_s - T^{(i-1)})$
7:         $L :=$ cross-validate($T^{(i-1)} \cup \{t_s\}$, $L_n$, $S$)
8:         if $L < L^{(i)}$
9:             $L^{(i)} := L$
10:            $T^{(i)} = T^{(i-1)} \cup \{t_s\}$
11: $i^* := \mathrm{argmin}_{i \in \{0,\ldots,(n-1)\}} L^{(i)}$
12: return induce($T^{(i^*)}$, $S$)
```

5 Empirical Asymmetric Selective Transfer

Since addition of extra support targets may increase the predictive accuracy for the main target, but is not guaranteed to do so, and moreover some target variables may help while others are detrimental, we can consider the following procedure: add extra target variables one by one, always selecting that target variable that results in the best model.

How can we select the "best" target to add to our current support set? As explained before, any measure that is symmetric or takes only relations among the target values into account will be suboptimal. Therefore, we directly measure the increase in predictive performance that a candidate support target yields using (internal) cross-validation. This takes into account all possible effects of including a certain support target.

Our "Empirical Asymmetric Selective Transfer (EAST)" procedure is described in Algorithm 1. It essentially implements the approach outlined above. The internal loop finds the next best candidate support target that can be added to the multi-target model. To do so, it calls the procedure cross-validate(T, L_n, S), which computes the 10-fold cross-validated loss L_n with regard to the main target y_n of a multi-target model with targets T constructed from S. The outer loop repeats this process until the support set is equal to the set of all targets. In the end, the algorithm returns the best support set found in this way.

The computational cost of EAST compares as follows to the execution time required for building a single-target decision tree (T_{ST}). Iteration i of EAST's outer loop costs $9 \cdot T_{\mathrm{ST}} \cdot (i+1) \cdot (n-i)$, because it tries $(n-i)$ candidate support targets and cross-validates for each candidate a $(i+1)$-target tree. Building one single $(i+1)$-target tree costs $\approx T_{\mathrm{ST}} \cdot (i+1)$; cross-validating it costs 9 times more (10 folds, each with a training set of $0.9 \cdot |S|$). $(n-1)$ iterations of EAST therefore cost $9 \cdot T_{\mathrm{ST}} \sum_{i=1}^{(n-1)} (i+1)(n-i)$, i.e., EAST is a factor $O(n^3)$ slower than building a single-target decision tree.

In our experiments, EAST's runtime proved to be acceptable because T_{ST} was relatively small. For example, for a dataset with 9 targets, EAST took on average

25 minutes; a factor 290 slower than T_{ST}. For large datasets, an alternative is to replace the cross-validation in EAST's internal loop by a single train/test split. This modification would make the algorithm about a factor 10 faster.

6 Experimental Evaluation

The aim of our experiments is to test to which extent EAST, for a given main target, succeeds in finding a good set of support targets. We do this by comparing EAST to two common baseline models: a single-target model for the main target (ST) and a multi-target model that includes all targets (MT). We also compare to the TC algorithm by Thrun & O'Sullivan [8], which we briefly describe next.

6.1 The TC Algorithm

We compare EAST to the task (target) clustering algorithm (TC) by Thrun & O'Sullivan [8], because clustering of targets is a straightforward and frequently used approach to exploit transfer selectively; TC is also quite general and can easily be used with multi-target decision trees as base models.

TC exhaustively searches for the clustering $C_1, \ldots, C_K$ of targets that maximizes $\frac{1}{n}\sum_{k=1}^{K}\sum_{y_i \in C_k}\frac{1}{|C_k|}\sum_{y_j \in C_k} c_{i,j}$, with $c_{i,j}$ the transfer from y_j to y_i. Next, it builds a multi-target model for each cluster C_k. As in EAST, transfer is estimated empirically using cross-validation (but TC only estimates the pairwise transfer between two targets, while EAST compares candidate support sets of arbitrary size).

The number of possible partitions grows exponentially with the number of targets. As a result, TC quickly becomes computationally infeasible, even for moderate numbers of targets. For example, computing all partitions for 9 targets took 4.5 minutes; for the 39 targets in dataset S_3 (Table 2) this would take about 2 years.

An alternative and faster approach would be to use an approximate method to compute the clustering, such as hierarchical agglomerative clustering. We chose not to pursue this because this was also not done by Thrun & O'Sullivan (they consider a relatively small number of tasks) [8]. An exact method is also a stronger baseline to compare our approach to.

6.2 Experimental Procedure

The datasets that we use are listed, together with their properties, in Table 2. Many datasets are of ecological nature. We omit the description of each dataset; the interested reader can find details in Ženko [19]. Each dataset represents a multi-target regression or classification task, and the number of targets varies from 2 to 39.

EAST has been implemented in the decision tree induction system Clus, which is available as open source software from http://www.cs.kuleuven.be/~dtai/clus. Clus also implements single- and multi-target decision trees, so all results

Table 2. Dataset properties. Datasets S_1 to S_6 are regression tasks, the remaining ones are classification tasks. N is the number of examples, $|\mathbf{x}|$ the number of input variables, and $|\mathbf{y}|$ is the number of target variables.

(a)

	Dataset	N	$\|\mathbf{x}\|$	$\|\mathbf{y}\|$
S_1	Sigmea Real	817	4	2
	Soil Quality 1	1944	139	
S_2	Acari/Coll. groups	"	"	9
S_3	Coll. species	"	"	39
S_4	Soil Quality 2	393	48	3
	Water quality	1060		
S_5	Plants/Animals	"	16	14
S_6	Chemical	"	836	16

(b)

	Dataset	N	$\|\mathbf{x}\|$	$\|\mathbf{y}\|$
S_7	Mediana	7953	78	5
S_8	Bridges	104	7	5
S_9	Monks	432	6	3
S_{10}	Thyroid	9172	29	7

that we present next are obtained with the same system and parameter settings. All parameters are set to their default values. To avoid overfitting we prune the trees using CART validation set based pruning, i.e., we use 70% training data for tree building and 30% for pruning (as suggested by Torgo [20]). We normalize each numerical target to zero mean and unit variance.

We compare for each dataset and target variable, the predictive performance of a traditional single-target tree (ST), a tree constructed by EAST with all other targets as candidate support targets, a multi-target tree including all targets (MT), and a multi-target tree from the clustering created by TC for datasets where this is computationally feasible. We estimate predictive performance as the 10-fold cross-validated misclassification error (for classification tasks) or Pearson correlation (for regression tasks). Correlation is often used as evaluation measure in ecological modelling. To compare EAST to ST, MT, and TC we count the number of targets on which it performs better (wins) and the number on which it performs worse (losses) and apply the sign test. Besides strict wins and losses, we also report significant wins and losses; to this end we use the corrected paired t-test [21] with significance level 0.05.

6.3 Results and Discussion

Table 3 gives an overview of the results. EAST outperforms the three other algorithms. The sign test applied to the wins/losses counts shows that EAST is significantly better than ST ($p = 0.0003$) and TC ($p = 0.029$). At the 5% level, it is just not significantly better than MT ($p = 0.057$). This result is mainly due to the losses on dataset S_3. If we would disregard this dataset, then EAST is clearly better than MT ($p = 0.0003$).

We conjecture that the losses of EAST compared to MT on S_3 are the result of a form of overfitting due to the many targets in this dataset and high variance of the performance estimates. EAST loses to MT when its internal cross-validation incorrectly estimates the performance of a subset that does not include all other targets higher than that of the subset corresponding to MT. The chance that

Table 3. Pairwise comparison of methods. For each pair A/B, the number of targets are given that represent (significant) wins and losses for A when compared to B. Entries marked *n/a* were computationally infeasible (see Section 6.1).

	Dataset	EAST/ST				EAST/MT				EAST/TC			
		#win		#loss		#win		#loss		#win		#loss	
S_1	Sigmea Real	1	(0)	0	(0)	0	(0)	0	(0)	0	(0)	1	(0)
S_2	SQ1 - Acari/Coll. groups	7	(0)	2	(0)	3	(1)	6	(0)	8	(0)	1	(0)
S_3	SQ2 - Coll. species	25	(4)	8	(0)	15	(0)	24	(1)	*n/a*			
S_4	Soil Quality 2	2	(0)	1	(0)	2	(0)	1	(0)	1	(0)	2	(0)
S_5	WQ - Plants/Animals	8	(1)	6	(0)	9	(1)	5	(0)	*n/a*			
S_6	WQ - Chemical	11	(3)	5	(0)	14	(3)	2	(0)	*n/a*			
S_7	Mediana	3	(2)	2	(0)	4	(0)	1	(0)	4	(0)	1	(0)
S_8	Bridges	2	(2)	3	(0)	5	(0)	0	(0)	4	(0)	1	(0)
S_9	Monks	1	(0)	0	(0)	2	(1)	0	(0)	0	(0)	1	(0)
S_{10}	Thyroid	5	(2)	2	(0)	6	(1)	1	(1)	5	(0)	2	(0)
	Total	65	(14)	29	(0)	60	(7)	40	(2)	22	(0)	9	(0)

Table 4. Cross-validated Pearson correlation for EAST, ST, and MT for each target t of dataset S_6 *WQ - Chemical*. •, ○ denote a statistically significant improvement or degradation of EAST when compared to ST or MT. ■,□ denote a statistically significant improvement or degradation of MT when compared to ST.

t	EAST	ST	MT	t	EAST	ST	MT
1	0.50±0.07	0.51±0.07	0.11±0.17•□	9	0.26±0.15	0.22±0.19	0.06±0.16□
2	0.34±0.09	0.36±0.09	0.22±0.12•□	10	0.44±0.22	0.11±0.21•	0.34±0.20■
3	0.42±0.10	0.38±0.14	0.31±0.16•	11	0.35±0.18	0.11±0.20•	0.32±0.12■
4	0.41±0.10	0.39±0.06	0.38±0.16	12	0.49±0.12	0.51±0.07	0.42±0.18
5	0.32±0.13	0.32±0.15	0.37±0.16	13	0.26±0.14	0.20±0.16	0.20±0.11
6	0.37±0.16	0.16±0.15•	0.24±0.15	14	0.39±0.23	0.46±0.22	0.43±0.16
7	0.38±0.14	0.42±0.09	0.26±0.17 □	15	0.48±0.21	0.38±0.24	0.44±0.16
8	0.27±0.13	0.19±0.20	0.26±0.16	16	0.56±0.22	0.51±0.16	0.52±0.16

this happens depends on the number of targets (more targets implies more candidate subsets, which in turn implies a higher chance that at least one of these is incorrectly estimated as better than MT) and on the variance of the performance estimates obtained in EAST's internal cross-validation. While this kind of overfitting is unavoidable (it can happen in general when performing model selection), we expect it to remain small and expect few significant losses. This is confirmed by the results in Table 3.

Table 4 shows detailed results for dataset S_6. First consider the comparison ST versus MT. ST works best on 8 targets and MT works best on 7, so neither is a clear winner. This illustrates again the advantage of selective transfer. For some targets, using all other targets as support targets may be best, while for others using no support targets may be best. EAST successfully discovers this and may find a non-trivial subset that performs even better. This is confirmed by the results: EAST outperforms the best of ST and MT on 10 targets.

7 Conclusions and Future Work

This paper addresses the single-target with support targets prediction task, where the goal is to build a model for the main target (or one model for each target in case of multiple targets), and where a number of candidate support targets are available (only) at model induction time, which may carry information about the main target. The paper's chief contribution is Empirical Asymmetric Selective Transfer (EAST), an algorithm that approximates the subset of support targets that, when predicted together with the main target in a multi-target model, maximally improves predictive performance of the main target.

Experiments show that EAST, on top of a multi-target decision tree learner, outperforms single-target decision trees, multi-target decision trees, and multi-target decision trees with target clustering.

We would like to address a few questions in future work. First, we will analyze in more depth to which degree the subset selected by EAST approximates the optimal support set. This analysis will include experiments to gain more insight in the overfitting behavior of EAST. Second, it is not clear if different multi-target learners can exploit the additional information available in other targets equally well. Therefore, we will test EAST in combination with different base-learners. This will give us more insight in the behavior of selective inductive transfer with those base-learners.

Acknowledgments. B. Piccart is supported by project G.0255.08 "Efficient microprocessor design using machine learning" funded by the Research Foundation - Flanders (FWO-Vlaanderen). J. Struyf and H. Blockeel are post-doctoral fellows of the Research Foundation - Flanders (FWO-Vlaanderen). The authors are grateful to S. Džeroski and B. Ženko for providing the datasets. This research utilizes the high performance computational resources provided by the K.U. Leuven (http://ludit.kuleuven.be/hpc).

References

1. Blockeel, H., Džeroski, S., Grbović, J.: Simultaneous prediction of multiple chemical parameters of river water quality with TILDE. In: 3rd European Conf. on Principles of Data Mining and Knowledge Discovery, pp. 32–40 (1999)
2. Demšar, D., Džeroski, S., Larsen, T., Struyf, J., Axelsen, J., Bruus Pedersen, M., Henning Krogh, P.: Using multi-objective classification to model communities of soil microarthropods. Ecol. Model. 191(1), 131–143 (2006)
3. Clare, A., King, R.: Knowledge discovery in multi-label phenotype data. In: 5th European Conf. on Principles of Data Mining and Knowledge Discovery, pp. 42–53 (2001)
4. Tsochantaridis, I., Joachims, T., Hofmann, T., Altun, Y.: Large margin methods for structured and interdependent output variables. J. Mach. Learn. Res. 6, 1453–1484 (2005)
5. Blockeel, H., Schietgat, L., Struyf, J., Džeroski, S., Clare, A.: Decision trees for hierarchical multilabel classification: A case study in functional genomics. In: 10th European Conf. on Principles and Practice of Knowledge Discovery in Databases, pp. 18–29 (2006)

6. Caruana, R.: Multitask learning. Mach. Learn. 28, 41–75 (1997)
7. Jensen, D., Neville, J., Gallagher, B.: Why collective inference improves relational classification. In: 10th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining, pp. 593–598 (2004)
8. Thrun, S., O'Sullivan, J.: Discovering structure in multiple learning tasks: The TC algorithm. In: 13th Int'l Conf. on Machine Learning, pp. 489–497 (1996)
9. Silver, D., Mercer, R.: The parallel transfer of task knowledge using dynamic learning rates based on a measure of relatedness. Connect. Sci. 8(2), 277–294 (1996)
10. Rosenstein, M.T., Marx, Z., Kaelbling, L.P., Dietterich, T.G., Whistler, B.C.: To transfer or not to transfer. In: NIPS 2005 Workshop on Transfer Learning, 4 pages (2005)
11. Kaski, S., Peltonen, J.: Learning from relevant tasks only. In: 18th European Conf. on Machine Learning, pp. 608–615 (2007)
12. Evgeniou, T., Micchelli, C.A., Pontil, M.: Learning multiple tasks with kernel methods. J. Mach. Learn. Res. 6, 615–637 (2005)
13. Bakker, B., Heskes, T.: Task clustering and gating for Bayesian multitask learning. J. Mach. Learn. Res. 4, 83–99 (2003)
14. Xue, Y., Liao, X., Carin, L., Krishnapuram, B.: Multi-task learning for classification with Dirichlet process priors. J. Mach. Learn. Res. 8, 35–63 (2007)
15. Yu, K., Tresp, V., Schwaighofer, A.: Learning Gaussian processes from multiple tasks. In: 22nd Int'l Conf. on Machine Learning, pp. 1012–1019 (2005)
16. Blockeel, H., De Raedt, L., Ramon, J.: Top-down induction of clustering trees. In: 15th Int'l Conf. on Machine Learning, pp. 55–63 (1998)
17. Silver, D.L., Mercer, R.E.: Selective functional transfer: Inductive bias from related tasks. In: IASTED Int'l Conf. on Artificial Intelligence and Soft Computing, pp. 182–189 (2001)
18. Struyf, J., Džeroski, S.: Constraint based induction of multi-objective regression trees. In: Knowledge Discovery in Inductive Databases, 4th Int'l Workshop, Revised, Selected and Invited Papers, pp. 222–233 (2006)
19. Ženko, B.: Learning predictive clustering rules. PhD thesis, University of Ljubljana, Slovenia (2007)
20. Torgo, L.: Error estimators for pruning regression trees. In: 10th European Conf. on Machine Learning, pp. 125–130 (1998)
21. Nadeau, C., Bengio, Y.: Inference for the generalization error. Mach. Learn. 52, 239–281 (2003)

Ensemble-Trees: Leveraging Ensemble Power Inside Decision Trees

Albrecht Zimmermann

Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, 3001 Leuven, Belgium
Albrecht.Zimmermann@cs.kuleuven.be

Abstract. Decision trees are among the most effective and interpretable classification algorithms while ensembles techniques have been proven to alleviate problems regarding over-fitting and variance. On the other hand, decision trees show a tendency to lack stability given small changes in the data, whereas interpreting an ensemble of trees is challenging to comprehend. We propose the technique of *Ensemble-Trees* which uses ensembles of rules *within* the test nodes to reduce over-fitting and variance effects. Validating the technique experimentally, we find that improvements in performance compared to ensembles of pruned trees exist, but also that the technique does less to reduce structural instability than could be expected.

1 Introduction

Decision trees have been a staple of classification learning in the machine learning field for a long time [1,2,3]. They are efficiently inducible, relatively straightforward to interpret, generalize well, and show good predictive performance. All of these benefits, however, are diminished by the fact that small changes in the data can lead to very different trees, over-fitting effects, and varying accuracies when employed to classify unseen examples.

To deal with problems in classification learning such as over-fitting, ensemble methods have been proposed. While the details vary, the main mechanism remains the same: a so-called "weak" learner, such as the afore-mentioned decision trees, is trained on a particular sample distribution of the available training data. The subset is then altered, possibly according to the performance of the learned classifier, and another classifier trained. In the end, an ensemble of weak classifiers is combined to classify unseen examples. The use of ensembles results in more stability w.r.t. predictive accuracy since over-fitting effects of several classifiers are balanced against each other. This advantage is however traded off both against difficulties with interpreting the complete classifier, since it can easily number tens of weak classifiers, and against increased training times.

To improve on the weaknesses of both decision trees and ensembles, we propose to essentially invert the ensemble construction process: Instead of inducing complete classical decision trees and combining those into a classifier, our solution – *Ensemble-Trees* – induces small ensembles of rules as tests in each inner node. The expected benefits would be relatively small and interpretable trees compared to ensembles of trees, and better structural stability w.r.t. changes in the data and higher predictive accuracy compared to classical decision trees.

The paper is structured as follows: in the following section, we will develop our approach, going over the main details of decision tree induction, k-best rule mining, and combination of rule predictions in the final tree. In Section 3, the connection to existing works will be made, followed by an experimental evaluation of our hypotheses regarding the behavior of *Ensemble-Trees* in Section 4. Finally, we discuss the results of this evaluation, and conclude, in Section 5.

2 From Decision Trees to *Ensemble-Trees*

Decision trees in different forms [1,2,3] have been one of the biggest success stories of classification learning, with Quinlan's C4.5 one of the most effective symbolic classification algorithms to date. Even when adapted to allow for regression or clustering, the main algorithm for inducing a decision tree stays more or less the same: For a given subset of the data, a "best" test according to an interestingness measure is selected, the data split according to this test, and the process repeated on the two subsets thus created. The interestingness measures used (e.g. information gain, intra-class variance) typically reward tests that make the subsets more homogeneous w.r.t. a target variable (e.g. the class label, or a numerical value). In general, a test can have any number of outcomes but for reasons that will be discussed later we limit ourselves to binary tests. The pseudo-code of a decision tree induction algorithm is shown as Algorithm 1.

Note that the best test as well as the prediction given in a leaf is not specifically instantiated in this pseudo- code and depends on the data mining task at hand.

Algorithm 1. The General Decision Tree Algorithm

```
Input: a data set $\mathcal{D}$, interestingness measure $\phi$, minimum leaf size $m$
Output: a decision tree $T$
$T$ =make_node($\mathcal{D}, \phi, m$)
return $T$

make_node($\mathcal{D}, \phi, m$)
Find best test $t$ on $\mathcal{D}$ according to $\phi$
$\mathcal{D}_{yes} = \{d \in \mathcal{D} \mid t$ matches $d\}$
$\mathcal{D}_{no} = \{d \in \mathcal{D} \mid t$ does not match $d\}$
if $|\mathcal{D}_{yes}| \geq m \wedge |\mathcal{D}_{no}| \geq m$ then
    $n_{yes}$ =make_node($\mathcal{D}_{yes}, \phi, m$)
    $n_{no}$ =make_node($\mathcal{D}_{no}, \phi, m$)
    return node($t, n_{yes}, n_{no}$)
else
    return leaf(prediction($\mathcal{D}$))
end if
```

In the popular C4.5 approach, the leaf prediction is the majority class in $\mathcal{D}$, and a binary test takes the form of the attribute-value pair that leads to the largest increase in homogeneity, measured by information gain. This means, however, that small changes in the data can have a strong effect on the composition of the tree. Consider ϕ to be information gain, and a data set with the following characteristics:

index	A_1	A_2	...	class
1	v_2	v_1	...	+
2	v_1	v_2	...	−
3	v_2	v_1	...	+
4	v_1	v_1	...	+
5	v_1	v_2	...	−
6	v_1	v_2	...	−
7	v_1	v_1	...	−
8	v_2	v_1	...	+
...	...	...	...	...

Obviously, a change in either the value of A_1 in instance 4, or the value of A_2 in instance 7 would improve the strength of these attributes as a test, respectively. Similarly, removal of one of those two instances, e.g. as part of a cross-validation, changes which attribute is chosen. Since the choice of test affects how the data is split, this effect ripples down through the rest of the tree. Assuming that A_1 and A_2 perform equally well on the rest of the data, the decision between them comes down to an arbitrary tie-breaker, giving the tree potentially rather different composition – and performance.

While attempts have been made to alleviate this problem, using linear combinations of attributes-value pairs in nodes [4], *option trees* [5], which in case of tests with very similar performance keep *all* of them and sort the data down all paths, or *random forests* [6], which randomly sample a subset of attributes as candidates for tests, these techniques strongly decrease comprehensibility of the tree, trading it off against better performance.

2.1 Ensembles of Rules and Efficient *k*-Best Induction

Ensembles of conjunctive rules, used together with conflict-resolution techniques such as (weighted) voting [7], or double induction [8] have been shown to perform better than decision lists, balancing the bias/over-fitting of single rules.

If this stabilizing effect of using a (small) ensemble of rules is brought to bear for tests inside a decision tree node, we expect that splits effected by those ensembles are less susceptible to changes in the data than splits effected by individual tests. This in turn should lead to trees having rather similar composition and structure, and ideally also similar rule ensembles in the test nodes. This also explains the name of our technique, *Ensemble-Trees*, since the tree-structure serves to tie the smaller ensembles together.

Additionally, over-fitting effects of individual tests can be expected to be reduced, potentially removing the need for post-pruning. Using conjunctive rules as tests always leads to binary splits, since a conjunction can be either *true* or *false*, thus the formulation of a test as binary in the general algorithm.

Using a branch-and-bound search based on the pruning techniques for convex interestingness measures (such as information gain, and intra-class variance) pioneered by Morishita and Sese [9], it is possible to efficiently induce the k best conjunctive rules w.r.t. the measure used. The branch-and-bound technique is based on calculating an upper bound for each rule encountered during enumeration. Only rules whose upper bound exceeds the kth-best score at the time of potential specialization *are* actually specialized. For further details, we refer the reader to Morishita and Sese's work.

Thus, the generic "Find best test" can be instantiated by "Find the k best rules". Since conjunctive rules induced in this way are quantified w.r.t. a *target value* such as the class label, the split becomes more complex than the straight-forward "match" used in the general algorithm. We will discuss the majority voting technique used in our approach in the next section.

2.2 Majority Splitting

Splitting the data according to a single attribute-value pair is straight-forward in that the pair either matches an instance, or does not match it, and the class prediction is implicit in how the distribution of class labels changes in the subsets produced. We adopt the convention that the left branch of a test node is the "yes" branch, i.e. instances that are matched by a test are sorted to the left, unmatched instances to the right.

Once several tests are used to split the data, the situation becomes a bit more complex however. Consider the following three rules for a binary prediction problem:

1. $A_1 = v_1 \wedge A_2 = v_1 \Rightarrow +$
2. $A_3 = v_1 \wedge A_4 = v_1 \Rightarrow -$
3. $A_1 = v_1 \wedge A_5 = v_2 \Rightarrow -$

and a subset of instances:

index	A_1	A_2	A_3	A_4	A_5	class
1	v_1	v_1	v_2	v_1	v_1	$+$
2	v_1	v_1	v_1	v_1	v_1	$+$
3	v_1	v_2	v_1	v_1	v_2	$-$
4	v_1	v_1	v_1	v_2	v_2	$-$
5	v_2	v_1	v_2	v_1	v_1	$+$

Considering the first instance, we see that rule 1 matches, therefore advocating that the instance be sorted to the *left*. Rules 2 and 3 do not match, thus advocating to sort the instance to the *right*. Since they form the majority, it would be sorted to the right. However, their prediction is *opposite* to the prediction of the first rule. Therefore, the match/non-match of rules 1, and rules 2 and 3 have a different semantic interpretation. In the interest of making class distribution more homogeneous to facilitate classification, it is intuitive to interpret the non-match of rules 2 and 3 as in effect advocating that instance 1 be sorted to the *left*. All three rules are then in agreement and instance 1 is sorted left.

The convention that matched instances are sorted to the left therefore is transformed into the convention that instance that are matched by rules *that predict the first class value* (or not matched by rules predicting the second class label) are sorted to the left, while in the opposite case being sorted to the right. There is of course the alternative of having three branches – one for instances for whom the first class is predicted, one for those with class two, and one for unmatched instances, which could be evaluated in an extension of our technique.

Looking at instance 2, we observe that it is matched by both rules 1 and 2, garnering one vote for being sorted into either direction. Rule 3 acts as a tie-breaker in this case, leading to instance 2 being sorted left. This example illustrates that the k that governs the amount of rules used in a test node should be odd so that ties do not have to be broken arbitrarily.

2.3 Inducing Ensemble Trees

Having covered the basic ingredients for inducing *Ensemble-Trees* in the preceding sections, Algorithm 2 shows the pseudo-code of the complete algorithm.

Note, that the Ensemble Tree algorithm includes another pre-pruning/ stopping criterion: If it is not possible to find the user-specified amount of rules, not enough different splitting tests improving homogeneity can be induced, and the node is turned into a leaf. This can occur since typically k-best interestingness mining rejects rules which have the same score as one of their generalizations, since this translates into conjuncts that do not add any information. Similarly, and additionally, to the minimum size of a leaf m, this criterion pre-prunes test nodes that are not supported by enough data to be expected to generalize well.

3 Related Approaches

As mentioned above, *Ensemble-Trees* can be considered as extensions of classical decision trees such as the trees induced by the C4.5 algorithm, in that the main difference lies in the replacement of the attribute-value pairs in test nodes by small ensembles of conjunctive rules. Our approach is not the first to attempt and modify the tests in nodes however.

3.1 Decision Trees with Complex Tests

To our knowledge the first that recognized the problem of similarly performing tests, and proposed a solution in the form of *option trees* were Kohavi and Kunz [5]. Their solution consists of not choosing one out of several tests arbitrarily

Algorithm 2. The Ensemble Tree Algorithm

```
Input: data set $\mathcal{D}$, interestingness measure $\phi$, minimum leaf size $m$, number
of rules $k$
Output: an Ensemble Tree $T$
$T$ =make_node($\mathcal{D}, \phi, m, k$)
return $T$

make_node($\mathcal{D}, \phi, m, k$)
Attempt to find $k$ best conjunctive rules $\mathcal{R} := \{r_i\}$ on $\mathcal{D}$ according to $\phi$
if $|\mathcal{R}| < k$ then
  return majority class in $\mathcal{D}$
end if
$\mathcal{D}_{left} = \{d \in \mathcal{D} \mid$ at least $\lceil k/2 \rceil$ rules vote $d$ is sorted left$\}$
$\mathcal{D}_{right} = \{d \in \mathcal{D} \mid$ at least $\lceil k/2 \rceil$ rules vote $d$ is sorted right$\}$
if $|\mathcal{D}_{left}| \geq m \wedge |\mathcal{D}_{right}| \geq m$ then
  $n_{left}$ =make_node($\mathcal{D}_{left}, \phi, m, k$)
  $n_{right}$ =make_node($\mathcal{D}_{no}, \phi, m, k$)
  return node($\mathcal{R}, n_{left}, n_{right}$)
else
  return leaf(majority_class($\mathcal{D}$))
end if
```

but instead creating "option nodes" having children nodes including all of these tests, and splitting the subset at each node. For classification, unseen examples are propagated downwards through the tree, and in case of an option node, the label predicted by the majority of child nodes chosen. While their approach improved significantly on the accuracy of C4.5 and an ensemble of bagged trees, the option trees themselves turned out to easily consist of one thousand nodes or more, comparable to the accumulated size of 50 bagged trees.

In [4], the authors suggest an algorithm consisting of a mixture of uni-variate (classical) decision tree tests, linear and non-linear combinations of attributes, using neural networks. The resulting trees are reported to be smaller than univariate trees and to improve on the accuracy in several cases. Obviously however, such combinations of attributes will be harder to interpret for end users.

In [10] and [11], data mining techniques were employed to mine complex fragments in graph- or tree-structured class-labeled data, which are then used as tests in a decision tree classifier. The motivation in these works is similar to ours in that the need for complex tests is acknowledged. Both techniques limit themselves to a single test in each node, however, not attempting to correct over-fitting effects during the induction of the trees.

3.2 Ensembles of Trees

Ensemble methods have been used with decision trees before, with the trees acting as the weak classifiers in the ensemble scheme. One of the best-known ensemble techniques, BAGGING [12], was proposed specifically with decision trees as weak classifiers. In bagging, repeated boot-strap sampling of a data set is performed, decision trees learned on each of these samples, and their predictions combined using a majority vote. While Bagging helps in reducing over-fitting effects, the fact that instead of one tree several are created can be an impediment to interpreting the resulting classifiers.

The Bagging idea is taken one step further in the construction of Random Forests [6]. Not only are instances resampled but for each test the set of attributes that are evaluated as potential tests is chosen randomly. These two effects, in addition to using the trees unpruned, essentially ensures that nothing beyond attribute-interactions can be understood when interpreting the final classifier – trading off accuracy against comprehensibility even further.

Boosting, as embodied for instance by the well-known AdaBoost system [13] on the other hand, iteratively induces weak classifiers on data sets whose distribution is altered according to the performance of preceding classifiers. Mostly, this takes the form of resampling misclassified instances, or re-weighting them. While boosting technique can be proven to approximate the actual underlying classification function to an arbitrary degree, the resulting ensemble is again rather difficult to interpret, especially given the changes in the underlying distribution that are effected during the learning process.

3.3 ART – Changing Distributions

Finally, the ART approach [14] is related to our work in the sense that it uses a "classical" mechanism for changing underlying class distributions – *sequential covering* – in a novel way, inducing several rules on each subset. The stated intentions are the same as ours but the resulting classifier is rather different. We believe that the main insight (one that the authors did not make explicit) is that existing techniques for creating subsets of the data to mine patterns or rules on, can be used in far more general ways than have been explored so far.

4 Experimental Evaluation

An experimental evaluation is used to answer

Q1. How well *Ensemble-Trees* are suited to the classification task, especially in the absence of post-pruning.

Q2. Whether the use of ensembles of rules as tests in the inner nodes leads to more stable trees in the presence of changes in the underlying data, and whether the resulting trees are smaller (and better interpretable) than ensembles of trees.

We use several UCI data sets to compare *Ensemble-Trees* to the WEKA [15] implementations of C4.5, Bagging, and AdaBoost, each with C4.5 as the weak classifier.

Since we limit ourselves to nominal attribute values in this work, numerical attributes were discretized, using ten equal-width bins. With the branch-and-bound technique used for inducing the rule ensembles limited to binary classes, we used only data sets with two labels. However, given that every classification problem can be translated into a number of *one-against-one*, or *one-against-all* problems, we do not see this as a significant drawback of our approach.

We performed experiments with the minimum leaf size parameter m set to $2, 3, 4, 5, 10$ and in case of large data sets, 10% of the training data. As the interestingness measure ϕ, information gain was used. BAGGING and ADABOOST were set to 10 iterations of inducing weak classifiers, and we built *Ensemble-Trees* with $k = 3$, and $k = 5$, respectively. ADABOOST was used both in the resampling and the re-weighting mode. Decision tree and ensemble method results are reported using pruned trees while *Ensemble-Trees* are always unpruned.

4.1 Predictive Accuracy

Predictive accuracy was evaluated using a class-validated 10-fold cross-validation, with the folds being the same for each method. Table 1 reports mean and standard deviation for a minimum leaf size $m = 5$, except for the Trains data set, where $m = 2$. While details vary, the main trends can be observed for all minimum leaf sizes evaluated in the experiments.

Table 1. Predictive accuracies for decision trees/*Ensemble-Trees*, minimum leaf size 5

Data set	C4.5	$\text{ET}_{k=3}$	$\text{ET}_{k=5}$	BAGGING	ADABOOST_{RS}	ADABOOST_{RW}
Breast-Cancer	73.42 ± 5.44●	78.69 ± 4.34	80.14 ± 6.16	73.77 ± 6.98	67.09 ± 10.10●	66.77 ± 6.81●
Breast-Wisconsin	94.56 ± 2.93	95.28 ± 1.35	95.14 ± 1.80	95.42 ± 2.76	96.28 ± 1.81	95.99 ± 1.14
Credit-A	84.20 ± 2.93	85.51 ± 2.56	85.51 ± 2.56	85.22 ± 2.35	82.75 ± 3.45	82.03 ± 4.44●
Credit-G	71.90 ± 3.96●	80.33 ± 2.00	79.10 ± 5.09	74.40 ± 4.06●	72.60 ± 3.24●	70.30 ± 4.00●
Heart-Statlog	82.96 ± 8.04	81.85 ± 5.08	79.63 ± 6.82	80.74 ± 8.52	80.37 ± 7.82	78.52 ± 7.57
Hepatitis	84.50 ± 6.22	89.58 ± 5.61	90.92 ± 5.54	83.25 ± 5.35●	83.17 ± 7.07●	82.71 ± 8.27●
Ionosphere	88.60 ± 5.88	91.44 ± 3.82	88.92 ± 5.80	91.15 ± 4.37	90.87 ± 5.69	91.15 ± 6.25
Molec. Biol. Prom.	78.09 ± 14.57	83.73 ± 9.28	83.64 ± 11.36	88.00 ± 13.00	85.73 ± 10.96	88.64 ± 5.94
Mushroom	100 ± 0○	99.95 ± 0.06	99.95 ± 0.06	96.31 ± 5.94	100 ± 0○	100 ± 0○
Tic-Tac-Toe	91.76 ± 3.81○	76.20 ± 1.47	72.86 ± 4.59	96.87 ± 1.55○	98.02 ± 1.59○	96.97 ± 2.62○
Trains	60.00 ± 51.64	50.00 ± 52.70	50.00 ± 52.70	40.00 ± 51.64	80.00 ± 42.16	50.00 ± 52.70
Voting-Record	95.85 ± 2.83●	98.39 ± 1.11	98.62 ± 1.19	95.62 ± 2.29●	94.94 ± 3.04●	96.08 ± 3.09●

The table shows that *Ensemble-Trees* perform mostly well w.r.t. classification. In several cases, *Ensemble-Trees* are significantly better than ensemble methods (● denotes a significant loss of a technique at the 5%-level using paired t-test), while being outperformed only on Mushroom (barely), and Tic-Tac-Toe (○ denotes a significant win at the 5%-level).

Inspection of the standard deviations, however, gives a first indication that *Ensemble-Trees* do not turn out to be more stable performance-wise when the data composition changes. In fact, *Ensemble-Trees* using only three rules per test-node ensemble show *less* standard deviation, i.e. less variance, w.r.t. accuracy than *Ensemble-Trees* with larger ensembles.

4.2 Size and Stability of Induced Trees

The second question we evaluated was concerned with stability of induced trees w.r.t. changes in the data, and the comprehensibility of *Ensemble-Trees* and ensembles of trees, respectively. We used the 10-fold cross-validation mechanism to simulate changes in the underlying data, and report on size characteristics

Table 2. Number of nodes per tree for C4.5/*Ensemble-Trees*, accumulated number of nodes for all trees of the respective ensembles

Data set	C4.5	ET$_{k=3}$	ET$_{k=5}$	BAGGING	ADABOOST$_{RS}$	ADABOOST$_{RW}$
Breast-Cancer	13.6 ± 6.4	7.6 ± 3.5	9.4 ± 5.1	425.4 ± 19.4	485.2 ± 18.2	509.4 ± 14.7
Breast-Wisconsin	14.8 ± 1.5	13.4 ± 2.8	9.0 ± 2.1	154.2 ± 12.2	334.4 ± 12.2	331.8 ± 15.4
Credit-A	19.2 ± 4.2	16.4 ± 6.2	10.6 ± 6.1	226.6 ± 8.5	698.6 ± 17.0	725.2 ± 30.4
Credit-G	86.4 ± 9.2	22.8 ± 6.1	18.8 ± 8.9	896.0 ± 23.7	1212.2 ± 25.7	1239.2 ± 27.9
Heart-Statlog	14.4 ± 2.3	11.2 ± 2.2	10.8 ± 3.9	193.2 ± 14.5	302.6 ± 14.3	310.0 ± 14.5
Hepatitis	6.2 ± 3.0	10.6 ± 4.1	9.2 ± 3.0	86.6 ± 10.4	153.6 ± 7.9	165.4 ± 7.9
Ionosphere	17.2 ± 3.7	12.2 ± 6.0	10.4 ± 4.0	165.6 ± 7.6	243.4 ± 8.9	253.0 ± 9.2
Molec. Biol. Prom.	11.6 ± 1.0	5.4 ± 0.8	6.4 ± 1.3	98.2 ± 8.3	83.4 ± 3.9	85.8 ± 6.0
Mushroom	17	17	21	153.8 ± 5.6	17	169.4 ± 15.0
Tic-Tac-Toe	53.4 ± 2.8	3	2.8 ± 0.6	573.4 ± 9.2	700.8 ± 31.4	736.6 ± 23.7
Trains	3	3	3	30.2 ± 0.6	23.5 ± 7.5	28.4 ± 0.8
Voting-Record	8.2 ± 1.0	13.0 ± 3.4	13.2 ± 4.0	75.6 ± 6.7	184.2 ± 18.0	181.6 ± 15.7

of the trees in Tables 2 and 3. For C4.5 decision trees and *Ensemble-Trees*, we report the mean and standard deviation of sizes (number of nodes) and maximal depths, i. e. length of the longest branch, of the different trees. Since ensemble methods induce a different tree in each iteration, averaging over iterations *and* the folds becomes a difficult endeavor, and we report on the accumulated number of nodes (of all trees per fold).

Inspection of Table 2 shows that *Ensemble-Trees* always have a lot less nodes than ensembles of trees. Even if the accumulated sizes are normalized by dividing by the number of trees, this difference is still pronounced in most cases. Given that the ensemble methods do not improve markedly on the accuracy of *Ensemble-Trees* in most cases, quite a few more bags/iterations would be needed for better performance (except for the Mushroom data set), in turn leading to even less comprehensible final classifiers.

Ensemble-Trees also typically have somewhat fewer nodes than classical C4.5 decision trees, due to the more expressive tests in the nodes. The cases where this trend does not hold (the Hepatitis, and Voting-Record data sets) or is exaggerated (the Breast-Cancer, and Credit-G data sets) are also the ones where *Ensemble-Trees* perform best compared to the other approaches. However, in the case of the Tic-Tac-Toe data set, the small number of nodes is actually a symptom of an underlying characteristic, with another being the atrocious performance of *Ensemble-Trees*, compared to the other methods.

To explain the mechanism at work here, Figure 1 shows binary class data points in two-dimensional space, and the decision surfaces of three rules r_1, r_2, r_3. As can be seen, each of these rules predicts the positive class, advocating that the instances they cover be sorted to the left. Additionally, however, all of them advocate the sorting of the instances covered by the other two rules to the *right*. The result of the majority vote on this is that all instances are sorted to the right, the left subset is empty and thus its size less than m, leading to the formation of a leaf. The rather aggressive pre-pruning effected by this stopping criterion becomes problematic on data sets with small, non-overlapping sub-regions in the

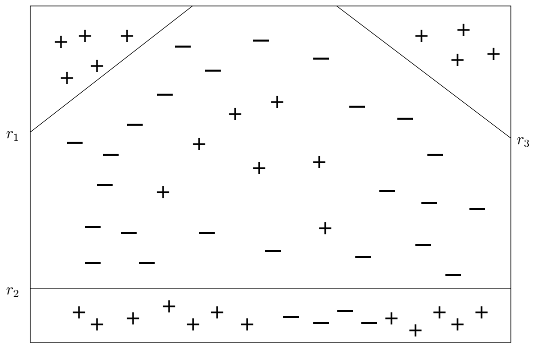


Fig. 1. Binary class data and decision surfaces of three discriminatory rules

Table 3. Averaged maximal depths for C4.5 trees/*Ensemble-Trees*

Data set	C4.5	$\text{ET}_{k=3}$	$\text{ET}_{k=5}$
Breast-Cancer	4.1 ± 1.5	2.90 ± 1.2	3.5 ± 2.2
Breast-Wisconsin	4.4 ± 0.5	4.2 ± 0.9	3.9 ± 0.9
Credit-A	6.7 ± 1.3	5.8 ± 2.7	4.0 ± 2.7
Credit-G	23.9 ± 2.3	7.9 ± 2.1	18.8 ± 8.9
Heart-Statlog	3.2 ± 0.8	3.8 ± 0.8	3.6 ± 1.0
Hepatitis	1.1 ± 1.5	4.7 ± 2.0	4.0 ± 1.3
Ionosphere	6.9 ± 1.6	5.6 ± 3.0	4.6 ± 1.8
Molec. Biol. Prom.	2.8 ± 1.0	2.2 ± 0.4	2.7 ± 0.7
Mushroom	4	5	5
Tic-Tac-Toe	6.0	1	0.9 ± 0.3
Trains	1	1	1
Voting-Record	2.6 ± 0.5	4.2 ± 0.8	4.4 ± 1.1

data. Tic-Tac-Toe is such a data set, as indicated by the rather large number of nodes (and therefore leaves) in – pruned – C4.5 decision trees.

The expected stabilization of trees w.r.t. changes in the data, however, cannot be observed. Neither in the number of nodes, nor in the maximal depths of trees do *Ensemble-Trees* markedly decrease the standard deviation over folds, compared to C4.5 pruned trees. Quite contrary, while the trees are shallower, and mostly smaller, *Ensemble-Trees* often show greater variance in both characteristics than classical decision trees do. Since the size measurements were extracted after post-pruning, we finally compare unpruned C4.5 trees to *Ensemble-Trees* in an attempt to understand how much of a stabilization effect post-pruning provides to the decision trees. The number of nodes is shown in Table 4, with the characteristics of *Ensemble-Trees* duplicated from Tables 2. Due to the page limit, we do not show the effects of pruning on the depth of trees here.

While Table 4 suggests that the reduction in variance for the size of a decision tree is a result of the post-pruning operation, and similar effects exist regarding

Table 4. Number of nodes per tree for unpruned C4.5/*Ensemble-Trees*

Data set	C4.5	$\mathrm{ET}_{k=3}$	$\mathrm{ET}_{k=5}$
Breast-Cancer	41.0 ± 8.7	7.6 ± 3.5	9.4 ± 5.1
Breast-Wisconsin	19.4 ± 4.9	13.4 ± 2.8	9.0 ± 2.1
Credit-A	35.8 ± 11.2	16.4 ± 6.2	10.6 ± 6.1
Credit-G	147.0 ± 10.1	22.8 ± 6.1	18.8 ± 8.9
Heart-Statlog	26.8 ± 3.3	11.2 ± 2.2	10.8 ± 3.9
Hepatitis	24.8 ± 2.6	10.6 ± 4.1	9.2 ± 3.0
Ionosphere	17.2 ± 3.7	12.2 ± 6.0	10.4 ± 4.0
Molec. Biol. Prom.	11.6 ± 1.0	5.4 ± 0.8	6.4 ± 1.3
Mushroom	21	17	21
Tic-Tac-Toe	68.2 ± 7.3	3	2.8 ± 0.6
Trains	3	3	3
Voting-Record	8.8 ± 1.1	13.0 ± 3.4	13.2 ± 4.0

maximal depth of the tree, this still means that *Ensemble-Trees* are structurally not more stable w.r.t. changes in the data than classical decision trees. While the use of ensembles of conjunctive rules as tests could bring more stability to the final tree, this promise remains unfulfilled. A potential improvement could lie in a better solution for the combination strategy, an issue we will investigate in future work.

A final subject that needs to be addressed is the interpretability of found solutions. Classical decision trees and ensemble methods are the borders of an interval that ranges from

- Easy (decision trees): paths are interpreted as conjunctions, tests as disjunctions, via
- Challenging (Bagging): individual trees are supposed to describe broadly the same phenomena, with the final classifier being an average of all, to
- Hard (Boosting): individual trees model local phenomena, with the final classifier being a weighted combination,

with *Ensemble-Trees* residing somewhere on the easier side of Bagging. As stated above, divide-and-conquer (the decision tree mechanism) is a way of changing class distributions. While decision trees are usually claimed to be easily interpretable, especially when written down in rule-form, this is deceiving in that each test is meaningful on a different distribution than the ones preceding it. So removing the tree structure is a blessing in disguise since a relevant part of the full model is not seen by the user anymore. Once the tree structure is kept, however, with conjunctive rules in the nodes themselves having a clear semantic, an *Ensemble-Tree* is rather straight-forward to interpret, especially when small ensembles are used.

5 Conclusion

In this work we develop the classification technique of *Ensemble-Trees*, an attempt to leverage the power of ensemble techniques in dealing with over-fitting and variance effects, while retaining as much of the easy interpretability of decision trees as possible.

Our experimental evaluation showed that over-fitting effects seem to be prevented more efficiently, leading to better classification accuracy in several cases. The structural stabilization of individual trees that seems possible if more complex tests are used that balance each other's bias, does not materialize however.

While the advantages of *Ensemble-Trees* are clear – no need for post-pruning, shallower trees with fewer nodes, and far easier interpretability than existing ensemble methods using decision trees as weak classifiers, they come with a caveat. There exist data sets where the aggressive pre-pruning that results from the use of ensembles of rules is detrimental to the performance of the classifier. Potential remedies, such as changing the combination strategy, or dynamically adjusting the k parameter, will be explored in future work.

References

1. Breiman, L., Friedman, J., Stone, C.J., Olshen, R.: Classification and Regression Tree. Chapman & Hall, New York (1984)
2. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Francisco (1993)
3. Blockeel, H., De Raedt, L.: Top-down induction of first-order logical decision trees. Artif. Intell. 101, 285–297 (1998)
4. Murthy, S.K.: On Growing Better Decision Trees from Data. PhD thesis, John Hopkins University, Baltimore, Maryland, USA (1997)
5. Kohavi, R., Kunz, C.: Option decision trees with majority votes. In: Fisher, D.H. (ed.) ICML, pp. 161–169. Morgan Kaufmann, San Francisco (1997)
6. Breiman, L.: Random forests. Machine Learning 45, 5–32 (2001)
7. Zaki, M.J., Aggarwal, C.C.: XRules: an effective structural classifier for XML data. In: Getoor, L., Senator, T.E., Domingos, P., Faloutsos, C. (eds.) KDD, Washington, DC, USA, pp. 316–325. ACM, New York (2003)
8. Lindgren, T., Boström, H.: Resolving rule conflicts with double induction. In: Berthold, M.R., Lenz, H.-J., Bradley, E., Kruse, R., Borgelt, C. (eds.) IDA 2003. LNCS, vol. 2810, pp. 60–67. Springer, Heidelberg (2003)
9. Morishita, S., Sese, J.: Traversing itemset lattices with statistical metric pruning. In: PODS, Dallas, Texas, USA, pp. 226–236. ACM, New York (2000)
10. Geamsakul, W., Matsuda, T., Yoshida, T., Motoda, H., Washio, T.: Performance evaluation of decision tree graph-based induction. In: Grieser, G., Tanaka, Y., Yamamoto, A. (eds.) DS 2003. LNCS (LNAI), vol. 2843, pp. 128–140. Springer, Heidelberg (2003)
11. Bringmann, B., Zimmermann, A.: Tree^2 - decision trees for tree structured data. In: Jorge, A.M., Torgo, L., Brazdil, P.B., Camacho, R., Gama, J. (eds.) PKDD 2005. LNCS (LNAI), vol. 3721, pp. 46–58. Springer, Heidelberg (2005)
12. Breiman, L.: Bagging predictors. Machine Learning 24, 123–140 (1996)
13. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. J. Comput. Syst. Sci. 55, 119–139 (1997)
14. Galiano, F.B., Cubero, J.C., Sánchez, D., Serrano, J.M.: Art: A hybrid classification model. Machine Learning 54, 67–92 (2004)
15. Frank, E., Witten, I.H.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann, San Francisco (1999)

A Comparison between Neural Network Methods for Learning Aggregate Functions

Werner Uwents[1] and Hendrik Blockeel[1,2]

[1] Department of Computer Science, Katholieke Universiteit Leuven
[2] Leiden Institute of Advanced Computer Science, Leiden University

Abstract. In various application domains, data can be represented as bags of vectors instead of single vectors. Learning aggregate functions from such bags is a challenging problem. In this paper, a number of simple neural network approaches and a combined approach based on cascade-correlation are examined in order to handle this kind of data. Adapted feedforward networks, recurrent networks and networks with special aggregation units integrated in the network can all be used to construct networks that are capable of learning aggregate function. A combination of these three approaches is possible by using cascade-correlation, creating a method that automatically chooses the best of these options. Results on artificial and multi-instance data sets are reported, allowing a comparison between the different approaches.

1 Introduction

Instead of using the classical attribute-value representation, it is more natural in some application domains to represent data as bags of vectors. This means that each data instance is described by a bag of vectors, but has only one target vector. Different instances can have different numbers of vectors in their bags, making traditional machine learning methods that have a fixed number of inputs impractical to use for this type of data. Moreover, the ability to process bags can be seen as the crucial element that distinguishes attribute-value methods from relational approaches, as explained in [1]. If we look at the kind of features that are constructed by a relational learner, an essential property of these features is that they map *sets* or *bags* of objects to a single scalar value. Functions that perform this kind of mapping are called *aggregate functions*.

Typically, systems that are capable of learning this kind of aggregate features use some simple and predefined aggregate functions like max or count. Most often, a propositionalization approach is followed. In this kind of approach, each data element is *summarized* into a vector of fixed length before the actual learning process. The components of this vector are then used as features. The aggregate features can become more complex when nesting selection conditions is allowed. As has been pointed out by Blockeel and Bruynooghe [1], the features constructed by relational learning systems are generally of the form $\mathcal{F}(\sigma_C(S))$ with S a set of objects, C a condition defined over single objects, σ the selection operator from relational algebra, and $\mathcal{F}$ an aggregate function, which maps a set to a scalar. But to keep the propositionalization approach feasible, a limited set of $\mathcal{F}$ functions needs to be used, and the number of different C considered for σ_C must remain limited. For instance, Krogel and Wrobel [2] allow a single attribute test in C, but no conjunctions.

Propositionalization is not the only possibility however. Some systems allow to learn aggregate functions in a more flexible and direct way. For instance, in ILP, if we have a clause *happy_father(X) :- child(Y,X)*, the "feature" constructed is essentially of the form $\exists\, y : child(y, x)$, which tests if the set of all y's related to x through the *child* relation is empty or not. In the above clause, S is the set of children of x, C is true, and $\mathcal{F}$ is the "there exists" operator ($\exists$). For the clause *happy_father(X) :- child(Y,X), age(Y,A),* $A < 12$, S and $\mathcal{F}$ are the same, while C is a condition on the age of the children now. But a direct approach has disadvantages as well. The structure of the search space of features of the form $\mathcal{F}(\sigma_C(S))$ becomes much more complex and difficult to search when $\mathcal{F}$ can be something else than $\exists$ [3]. Blockeel and Bruynooghe [1] pointed out that for this reason these systems typically construct structurally complex conditions but always use the same, trivial aggregate function, namely $\exists$. The importance of using more complex aggregate functions has however been recognized by many people [2,4].

Therefore it is useful to study how direct approaches could include aggregate functions other than $\exists$. More recently, methods for learning more advanced features of the form $\mathcal{F}(\sigma_C(S))$ have been proposed. Vens, Van Assche et al. [5,6] proposed a random forest approach that avoids the problems of searching a complex-structured search space, while Vens [3] studied the monotonicity properties of features of the form $\mathcal{F}(\sigma_C(S))$ and showed how efficient refinement of such features is possible for the most commonly occurring aggregate functions.

All these are symbolic approaches to learning aggregate functions. In parallel, Uwents and Blockeel studied to what extent subsymbolic representations of aggregate features can be learned using neural network approaches. The advantage of this approach is that the complex search is avoided and that the function $\mathcal{F}$ and condition $\sigma_C(S)$ in the $\mathcal{F}(\sigma_C(S))$ feature are learned simultaneously and automatically. Recurrent neural networks were first proposed for this task, leading to the concept of relational neural networks [7]. While a regular network maps one input vector to an output vector, recurrent networks can map a sequence of input vectors to a single output vector. This property was exploited to handle sets of vectors, the elements of which were given as input to the network in random order. Recurrent networks are however not the only option for learning aggregate functions, a number of other network structures could be used as well. Recently, also a cascade-correlation approach has been proposed [8].

All the previous work focused on extending relational learning with the ability to learn aggregation functions, and evaluating how this ability affects overall predictive accuracy. In this work, however, we focus on the particular problem of learning aggregate functions of the form $\mathcal{F}(\sigma_C(S))$, with S a given set of tuples. From the point of view of relational learning, this is a simplified setting: there is only one 1-n relation, relating each single target to a single bag of tuples.

All these tuples are of the same type, and they do not participate in further relations. While this is a limitation from the point of view of relational learning, the exclusion of such further relational information allows us to get a better view on the inherent capacity of certain neural network structures to learn particular aggregate functions, which is the main aim of this research. Another point of difference with previous work is that we study a broader range of aggregate functions than has previously been done.

We will compare different methods for subsymbolic learning of aggregate functions from bags of vectors with an associated target value. In Section 2, we discuss three basic and simple methods that use adapted forms of standard neural network structures, as well as a combined approach using cascade-correlation. In Section 3, we present experimental results comparing the different approaches to each other. We conclude in Section 4.

2 Neural Network Methods

2.1 Simple Methods

There are some straightforward and rather simple ways to use standard neural networks structures in a slightly modified form to process bag data and learn aggregate functions. Three of these simple approaches are discussed here.

Adapted Feedforward Networks. Probably the simplest way to extend normal feedforward neural networks to bags, is to use a feedforward network with $m_{max} \times n$ inputs. In this formula, m_{max} is the maximum number of vectors a bag can contain, while n is the number of real numbers per vector. When a bag of size m is fed into the network, the first $m \times n$ inputs are filled with the values of the m vectors in the bag. The order in which this happens, is not important. The remaining $(m_{max} - m) \times n$ inputs are filled up with padding values. For these padding values, a fixed, given value is used.

One of the disadvantages is that m_{max} can be quite large. This results in a network with a lot of connections and corresponding weights to train, which is bad because it increases the complexity of the model and the danger of overfitting. Another disadvantage could be that the number of hidden neurons is fixed. It seems more logical that the number of hidden neurons should be somehow proportional to m, the cardinality of the input bag, because the complexity of the function is probably also proportional to the size of the bag. These disadvantages can be countered by duplicating the hidden neurons for each input vector in the bag and by sharing the weights. Because the function that has to be learned by the network should be invariant to permutations of the input vectors, a lot of weights can be shared across the network. More precisely, the hidden units are copied for each input vector and the weights are shared between these copies. The weight sharing also applies to the weights for the different input vectors within a copy. Finally, also the weights for the connections between these copies and the output units are shared. This network structure will be referred to as *sym* because of the symmetry introduced by the weight sharing.

Networks with Aggregation Units. It is also possible to add some predefined aggregate functions to the network. They are placed after a neuron. Activation values for this neuron are computed for all input vectors in the bag and then the aggregate function is used to combine all these activation values into a single value. The aggregate functions could be placed after the hidden or the output units. Typical functions would be sum or max. The only condition for these aggregate functions is that they have to be derivable to be able to compute the gradient. This kind of approach has been applied to multi-instance problems before, where a softmax function was used [9]. The softmax function is defined as

$$\frac{1}{M}\log\left(\sum_{i=1}^{n} e^{M\cdot x_i}\right) \tag{1}$$

where M is a constant and the larger the value of M, the closer this function will approximate the real max function. Instead of choosing a value for M that is large enough to give a good approximation, one could take the limit for M going to infinity, which gives the real max function and a formula for its derivative. The sum, softmax and max function will be considered here as possible aggregate functions. Note that the use of a specific kind of aggregate function in the network, does not necessarily limit the network to learning only this kind of aggregate function from the data. By learning the right weights, other aggregate functions could still be approximated. The different network types will be referred to as *sum*, *hsum*, *smx*, *hsmx*, *max* and *hmax*. The prefix *h* indicates that the function is placed after each hidden unit, otherwise they are placed after the output units.

Recurrent networks. Another possibility is to use recurrent networks. They are typically used to process time series or other sequences of data, but they could be used to process bags as well, as done in [7]. The m vectors of the input bag are fed into the network one after another. For each vector, the activation levels of the hidden neurons are updated. After processing all input vectors, the last activation levels are used to compute the activation level of the output neuron. This kind of approach could be problematic if applied to large bags because recurrent networks are hard to train on large sequences. In the rest of the paper, two types of recurrent networks will be considered, locally and fully recurrent networks. In the locally recurrent networks, denoted as *lrc*, each hidden unit has a recurrent connection only with itself. In fully recurrent networks, named *frc*, each hidden unit also has recurrent connections with all other hidden units.

2.2 Combined Method

The previously discussed network structures have two important drawbacks. First of all, one of the network types has to be selected. After that, also the number of units in the hidden layer has to be chosen. Both choices are not trivial. A cascade-correlation approach solves this problem and makes a combination of the simple methods possible. The idea behind the original cascade-correlation

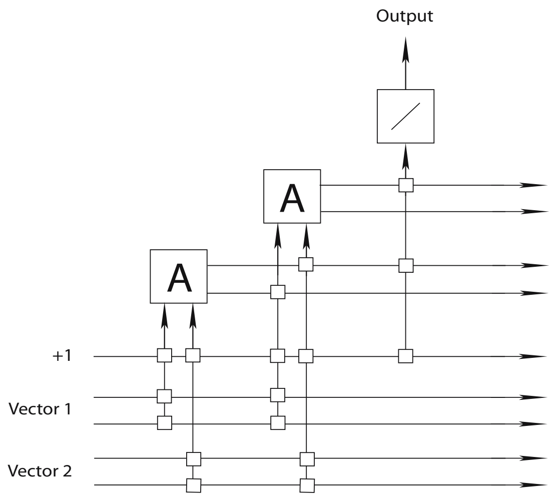


Fig. 1. Schema for an aggregate cascade-correlation network with 2 input vectors

algorithm [10] is to learn not only the weights, but also the structure of the network at the same time. This is done in a constructive way, meaning that only one neuron at a time is trained and then added to the network. One starts with a network without any hidden unit, and then hidden neurons are added, one by one, until some stopping criterion is satisfied. Once a hidden neuron has been added to the network, its weights remain fixed throughout the rest of the procedure. This also means that, besides the actual input vector, the output values of these existing hidden units can be used as extra inputs for any new hidden neuron. At the output, a linear function can be used.

The training of the network is done in two alternating phases. Before a new hidden neuron is added, its weights are trained while keeping the weights of all other hidden units fixed. This training is not done by minimizing the squared error between target and output, but by maximizing the correlation with the residual error. The residual error is defined as the difference between the actual target value and the output of the existing network, before adding the new neuron. The assumption is that its correlation with the residual error will make a new neuron useful in reducing the residual error and improving the prediction of the actual target. The maximization is done by computing the gradient and performing some form of gradient ascent. Instead of training only one candidate neuron at a time, a pool of neurons, initialized with random weights, can be trained. At the end, the best one is selected. This increases the chance that a good candidate will be found. Once the best candidate is selected and added to the network, the output weights for the updated network can be trained. If a linear function is used at the outputs, the output weights can be obtained by simple linear regression.

The concept of cascade-correlation networks can be extended to networks for learning aggregate functions, as proposed in [8]. The crucial difference is that instead of the simple hidden neurons, units that can process bags are used. These units come from the simple networks presented above. The *sym*, *hsum*, *hmax*, *hsmx* and *lrc* units can all be used as aggregation units in the hidden layer of the cascade-correlation network. For the rest, the network and the training of it works in the same way as for the feedforward cascade-correlation networks. A schema of an aggregate cascade-correlation network for 2 input vectors is shown in figure 1.

With all parts of the aggregate cascade-correlation network explained, it only remains to discuss the training of the network in more detail. Each time a new unit should be added to the hidden layer, a pool of units is created of all possible types. Weights are initialized randomly. After that, all units in the pool are trained for a number of iterations, similar to backpropagation. This training is basically a gradient ascent, maximizing the correlation with the outputs. The computation of the gradient depends of course on the type of unit. The gradient ascent itself is actually done by using resilient propagation, as described in [11]. This method has the advantage that the step size is determined automatically and convergence is faster than for a fixed step size. The basic idea is to increase the step size when the sign of the gradient remains the same, and decrease the step size when the sign changes.

When all units in the pool have been trained, the best one is chosen. In this case, the best unit is the one with the highest correlation. When the unit with the highest correlation has been chosen, it is installed in the network and the output weights have to be learned again. Because linear activation functions are used for the output units, the output weights can be determined with least squares linear regression.

3 Experiments

In this section, a number of experimental results will be discussed. First, a series of experiments is carried out on artificially created data sets. After that, the methods are evaluated on a multi-instance data set.

3.1 Simple Aggregates

A simple experiment to examine the capacity of the aggregate cascade-correlation network, is to create artificial data with predefined aggregate functions and train the networks on it. The data consists of bags with a variable number of elements. Each element of the bag is a vector with five components. Only the first or the first and second component are relevant for the target value, depending on the aggregate function under consideration. The values of these components are randomly generated, but in such a way that the target values are uniformly distributed over the possible target values. All the other components are filled with uniformly distributed random numbers from the interval $[-1, 1]$. It is very likely that the number of vectors in the bags influences the difficulty of the learning task, so different sizes are tested. The data sets denoted as small contain 5 to 10 vectors per bag, the medium data sets 20 to 30 and the large ones 50 to 100. Each data set contains 3000 bags. A range of different aggregate functions are used to construct the data sets:

1. **count:** the target is the number of vectors in the bag.
2. **sum:** the target is the sum of all values of the first component of the bag vectors.
3. **max:** the target is the maximum value of the first component of the bag vectors.
4. **avg:** the target is the average value of the first component of the bag vectors.
5. **stddev:** the target is the standard deviation of the values of the first component of the bag vectors.
6. **cmpcount:** the target is the number of bag vectors for which the value of the first component is smaller than the value of the second component.
7. **corr:** the target is the correlation between the first two components of the bag vectors.
8. **even:** the target is one if the number of positive values for the first component is even, and zero if it is odd.
9. **distr:** the target is one if the values of the first component come from a gaussian distribution, and zero if they are from a uniform distribution.
10. **select:** the target is one if at least one of the values of the first component lies in a given interval, and zero otherwise.
11. **conj:** the target is one if there is at least one vector in the bag for which the the first and the second component lie in a certain interval.
12. **disj:** the target is one if there is at least one vector in the bag for which the first or the second component fall in a certain interval.

The first 7 data sets have a numerical target, the others a nominal target. In case of a nominal target, the number of positive and negative examples are equal. Experiments are done using five-fold cross-validation. For each fold, 1800 examples are used as training set, 600 as validation set and another 600 as test set. For the simple approaches, the number of hidden units is 3 and resilient propagation [11] was used to train the networks for 1000 iterations. For the cascade-correlation method, the maximum number of hidden units is limited to 10. The number of candidate units trained in every step is 20. Each unit is trained for 500 iterations, which should be enough to have converged to optimal weights. For the data sets with nominal target, the accuracy is reported and for the sets with numerical targets the mean squared error is used. For the simple methods, only the best results are given and it is also indicated which types of network achieved this performance or a performance very close to it. The results are summarized in tables 1 and 2.

From the results, it is clear that most functions can be learned quite well. Only the *even* function seems impossible to learn. This is not very surprising, given that this is a kind of parity problem, which is known to be difficult. For the *distr* function, the number of vectors must be large to be able to learn it well. This makes sense because it is easier to say whether a bag of values come from a normal or uniform distribution if the bag is larger than when it is rather small. Table 1 also indicates which types of networks are the best to learn the considered concepts. There is no type that is best for all of them. Sometimes it is clear why certain types are the best, for instance for the sum and max data sets, but for other data sets it is not that obvious. This makes it hard to make the right choice beforehand. Furthermore, the results for the cascade-correlation network are always very close to the results for the best simple network or even beat this result. This shows that using the combined cascade-correlation approach is indeed a good idea.

3.2 Trains

The trains data sets are also artificially created data sets containing a number of trains. Every train consists of a number of cars, carrying some load. Some of the trains are eastbound, the others are westbound. The direction of the trains is what has to be learned and this target concept is based on the properties of the cars of a train and their loads. The cars of the train constitute a bag for each train. A data generator [12] for this train problem was used to create 12 data sets with different properties. Sets 1 to 4 consist of short trains, having 2 to 6 cars. Data sets 5 to 8 are similar to sets 1 to 4, except that they contain longer trains. Each of these trains consists of 20 to 29 cars. The used concepts are the same as for sets 1 to 4, except that the numbers in the aggregations are adapted to the longer length of the trains. Data sets 9 to 12 contain noisy data. This means that a number of samples have been mislabeled. The twelve datasets are defined as follows:

Table 1. Results for the simple aggregate data sets using the simple network structures. Accuracies or mean squared errors are given together with the standard deviations for five-fold cross-validation. The results are those for the best type of network. The types of network that produced this result or very similar results, are indicated in the last column.

		small	*medium*	*large*	*type*
MSE	count	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	hsum, sym
	sum	0.00 (0.00)	0.00 (0.00)	0.02 (0.01)	hsum, sym
	max	0.00 (0.00)	0.00 (0.00)	0.01 (0.01)	(h)max, (h)smx, lrc, frc
	avg	0.03 (0.03)	0.01 (0.01)	0.01 (0.01)	max, smx, lrc, frc, sym
	stddev	0.02 (0.01)	0.01 (0.01)	0.01 (0.01)	hmax, hsmx
	cmpcount	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	sym, hsum
	corr	0.05 (0.04)	0.04 (0.02)	0.01 (0.01)	sym, hsum, hmax, hsmx
accuracy	even	52.6 (0.0)	51.7 (0.6)	51.4 (0.5)	all types equal
	distr	63.3 (0.6)	74.6 (0.9)	80.3 (0.9)	hmax, hsmx
	select	99.6 (0.3)	99.9 (0.1)	99.8 (0.2)	smx
	conj	99.9 (0.0)	99.9 (0.0)	94.5 (0.4)	max, smx
	disj	89.3 (1.2)	76.7 (0.9)	74.6 (1.3)	hsmx

Table 2. Results for the simple aggregate data sets using cascade-correlation. Accuracies or mean squared errors are given together with the standard deviations for five-fold cross-validation.

		small	*medium*	*large*
MSE	count	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
	sum	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
	max	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
	avg	0.06 (0.06)	0.01 (0.01)	0.01 (0.01)
	stddev	0.02 (0.02)	0.01 (0.01)	0.00 (0.00)
	cmpcount	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
	corr	0.04 (0.04)	0.02 (0.02)	0.01 (0.01)
accuracy	even	52.33 (0.00)	50.10 (0.73)	51.90 (0.72)
	distr	66.18 (0.56)	76.68 (0.79)	84.10 (1.57)
	select	99.15 (0.41)	99.63 (0.15)	98.87 (0.48)
	conj	100.00 (0.00)	99.92 (0.11)	99.93 (0.20)
	disj	99.00 (0.43)	97.75 (0.55)	96.30 (1.81)

1. Trains having at least one circle load are eastbound, the others are westbound.
2. Trains having at least one circle or rectangle load and at least one car with peaked roof or 3 wheels are eastbound, the others are westbound.
3. Trains having more than 8 wheels in total are eastbound, the others are westbound.
4. Trains having more than 3 wheels in total and at least 2 rectangle loads and maximum 5 cars are eastbound, the others are westbound.
5. Same concept as for set 1.
6. Same concept as for set 2.
7. Trains having more than 53 wheels in total are eastbound, the others are westbound.
8. Trains having more than 45 wheels in total and at least 10 rectangle loads and maximum 27 cars are eastbound, the others are westbound.
9. Same concept as for set 1, but with 5% noise.
10. Same concept as for set 1, but with 15% noise.
11. Same concept as for set 3, but with 5% noise.
12. Same concept as for set 3, but with 15% noise.

Each data set contains 3000 instances. Experiments are done using five-fold cross-validation which means that for each fold 1800 instances are used for training, 600 for validation and 600 for testing. For the simple approaches, the number of hidden units is 5 and the number of training iterations is 1000. For the cascade-correlation method, the maximum number of hidden units is limited to 10. The number of candidate units trained in every step is 20. Each unit is trained for 500 iterations. The results are given in table 3. It is clear that most concepts can be learned well. Most of the data sets without noise have an accuracy very close to 100%. Only for set 8, which has the most difficult concept, is it impossible to get close to perfect accuracy. For the data sets with noise, the best accuracies are all close to 100% minus the percentage of noise. It is also clear that cascade-correlation has a performance that is superior to any of the simple methods, just as for the simple aggregate data sets.

Table 3. Accuracies using five-fold cross-validation for the trains data sets

	casc-corr	*sym*	*lrc*	*frc*	*sum*	*hsum*	*max*	*hmax*	*smx*	*hsmx*
1	99.9	99.5	96.9	100.0	83.8	99.7	100.0	100.0	99.7	100.0
2	100.0	99.3	97.7	96.8	77.3	97.9	94.5	99.5	94.6	99.6
3	100.0	99.9	96.0	98.9	55.0	100.0	65.3	83.6	69.1	84.7
4	98.7	98.7	81.3	94.7	78.6	88.0	86.5	77.0	86.1	81.0
5	99.7	88.4	81.1	92.5	56.8	99.2	94.9	87.2	98.0	86.4
6	99.4	86.9	68.8	73.9	54.7	83.1	72.7	69.4	70.0	78.5
7	99.4	97.3	51.4	51.3	52.3	98.5	56.5	62.4	57.8	63.0
8	84.5	77.5	65.2	71.0	55.3	79.2	65.2	67.2	67.5	66.3
9	95.7	93.7	93.2	93.6	80.5	95.6	90.8	95.6	90.1	95.4
10	85.7	83.1	82.9	81.7	68.9	85.0	81.9	85.2	81.0	85.1
11	95.7	95.0	81.0	93.0	56.1	94.3	68.4	80.2	69.1	80.8
12	85.4	84.1	71.8	82.0	53.8	80.5	62.1	72.3	64.4	72.1

3.3 Musk

Musk is a well-known multi-instance data set [13]. Each data instance stands for a molecule, represented by a bag of all its possible conformations. A conformation is described by 166 numerical features. The molecules have to be classified as musk or non-musk. The data set consists of two parts. The first part contains 92 molecules, the second part 102. In each bag, there are between 2 and 40 conformations for the first part, and between 1 and 1044 for the second part.

Experiments were carried out using 10-fold cross-validation. The simple methods have 5 hidden units and are trained for 500 iterations. For the cascade-correlation networks, a pool of 20 neurons and 500 training iterations are used in every step. The data sets are a bit too small to use part of them as validation set. Therefore, the value of the correlation is used as stopping criterion. In the beginning, the correlation of a newly trained unit will be very high. This correlation decreases during the following training steps. Training will be stopped when the correlation falls below 0.75.

The results for the musk data sets can be found in table 4. The best neural networks perform well compared with the other methods. They do not beat all of the multi-instance methods, but that is not unexpected given that these methods were specifically designed for multi-instance problems while these networks are more general. The cascade-correlation networks are all very small, in most cases with just one hidden unit. If one looks at the type of unit selected, then this is almost always a *max* or *smx* unit. This is the most logical choice in case of a multi-instance problem. The networks with this type of units also have the best performance for the simple networks.

Table 4. Accuracies and 95% confidence intervals for the musk data sets using 10-fold crossvalidation. Results for other methods are obtained from [13].

	musk 1	*musk 2*
iterated discrim APR	92.4 [87.0-97.8]	89.2 [83.2-95.2]
GFS elim-kde APR	91.3 [85.5-97.1]	80.4 [72.7-88.1]
GFS elim-count APR	90.2 [84.2-96.3]	75.5 [67.1-83.8]
GFS all-positive APR	83.7 [76.2-91.2]	66.7 [57.5-75.8]
all-positive APR	80.4 [72.3-88.5]	72.6 [63.9-81.2]
backpropagation	75.0 [66.2-83.8]	67.7 [58.6-76.7]
C4.5 (pruned)	68.5 [40.9-61.3]	58.8 [49.3-68.4]
casc-corr	85.3 [83.1-87.5]	75.5 [72.6-78.4]
sym	75.8 [74.1-77.5]	71.1 [68.8-73.4]
lrc	77.2 [75.6-78.8]	75.2 [72.7-77.7]
frc	74.9 [73.4-76.4]	73.3 [70.8-75.8]
sum	56.6 [54.2-59.0]	49.0 [46.3-51.7]
hsum	78.2 [76.6-79.8]	65.5 [63.9-67.1]
max	73.1 [71.5-75.7]	65.6 [63.4-67.8]
hmax	79.0 [77.9-80.1]	77.4 [75.7-79.1]
smx	72.2 [71.0-73.4]	62.8 [60.8-64.8]
hsmx	77.3 [75.7-78.9]	78.1 [76.7-79.5]

4 Conclusion

In this paper, some subsymbolic approaches to learn aggregate function from bag data were discussed. These methods are based on neural networks. First, three simple methods were presented: adapted feedforward networks, recurrent networks and networks with special aggregation units integrated in the network. Another possibility is to combine these three approaches by using cascade-correlation, creating a method that automatically chooses the best of these options.

Results on artificial and multi-instance data sets were reported to assess the capacity of the different network structures. These results clearly show that it is possible to learn aggregate functions or at least approximate them with neural networks. They also indicate that the combined cascade-correlation approach is superior to any of the simple methods.

References

1. Blockeel, H., Bruynooghe, M.: Aggregation versus selection bias, and relational neural networks. In: Getoor, L., Jensen, D. (eds.) IJCAI 2003 Workshop on Learning Statistical Models from Relational Data, SRL 2003, Acapulco, Mexico (2003)
2. Krogel, M.A., Wrobel, S.: Transformation-based learning using multirelational aggregation. In: Rouveirol, C., Sebag, M. (eds.) ILP 2001. LNCS (LNAI), vol. 2157, pp. 142–155. Springer, Heidelberg (2001)

3. Vens, C., Ramon, J., Blockeel, H.: Refining aggregate conditions in relational learning. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) PKDD 2006. LNCS (LNAI), vol. 4213, pp. 383–394. Springer, Heidelberg (2006)
4. Knobbe, A., Siebes, A., Marseille, B.: Involving aggregate functions in multi-relational search. In: Principles of Data Mining and Knowledge Discovery, Proceedings of the 6th European Conference, August 2002, pp. 287–298. Springer, Heidelberg (2002)
5. Vens, C., Van Assche, A., Blockeel, H., Dzeroski, S.: First order random forests with complex aggregates. In: Camacho, R., King, R., Srinivasan, A. (eds.) ILP 2004. LNCS (LNAI), vol. 3194, pp. 323–340. Springer, Heidelberg (2004)
6. Van Assche, A., Vens, C., Blockeel, H., Dzeroski, S.: First order random forests: Learning relational classifiers with complex aggregates. Machine Learning 64(1-3), 149–182 (2006)
7. Uwents, W., Blockeel, H.: Classifying relational data with neural networks. In: Kramer, S., Pfahringer, B. (eds.) ILP 2005. LNCS (LNAI), vol. 3625, pp. 384–396. Springer, Heidelberg (2005)
8. Uwents, W., Blockeel, H.: Learning relational neural networks using a cascade-correlation approach. In: Železný, F., Lavrač, N. (eds.) ILP 2008. LNCS (LNAI), vol. 5194, pp. 315–329. Springer, Heidelberg (2008)
9. Ramon, J., De Raedt, L.: Multi instance neural networks. In: Raedt, L.D., Kramer, S. (eds.) Proceedings of the ICML 2000 workshop on attribute-value and relational learning, pp. 53–60 (2000)
10. Fahlman, S.E., Lebiere, C.: The cascade-correlation learning architecture. In: Touretzky, D.S. (ed.) Advances in Neural Information Processing Systems, Denver, vol. 2, pp. 524–532. Morgan Kaufmann, San Mateo (1989)
11. Riedmiller, M., Braun, H.: A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In: Proc. of the IEEE Intl. Conf. on Neural Networks, San Francisco, CA, pp. 586–591 (1993)
12. Michie, D., Muggleton, S., Page, D., Srinivasan, A.: To the international computing community: A new east-west challenge. Technical report, Oxford University Computing Laboratory, Oxford, UK (1994)
13. Dietterich, T.G., Lathrop, R.H., Lozano-Perez, T.: Solving the multiple instance problem with axis-parallel rectangles. Artificial Intelligence 89(1-2), 31–71 (1997)

Smoothed Prediction of the Onset of Tree Stem Radius Increase Based on Temperature Patterns

Mikko Korpela[1], Harri Mäkinen[2], Mika Sulkava[1],
Pekka Nöjd[2], and Jaakko Hollmén[1]

[1] Helsinki Institute for Information Technology,
Helsinki University of Technology, Department of Information and Computer Science, P.O. Box 5400, FI-02015 TKK, Espoo, Finland
http://ics.tkk.fi

[2] Finnish Forest Research Institute (Metla), Vantaa Research Unit,
P.O. Box 18, FI-01301 Vantaa, Finland
http://www.metla.fi

Abstract. Possible changes of the growing season of trees would have significant consequences on forest production. Predicting the onset of tree growth on the basis of climate records can be used for estimating the magnitude of such changes. Conventional methods for estimating the onset of tree growth use cumulative temperature sums. These estimates, however, are quite coarse, and raise questions about making better use of the weather information available. We approach the problem of predicting the onset of tree growth with a predictor based on a combination of a k-nearest neighbor regressor and a linear regressor. The inputs are weighted sums of daily temperatures, where the weights are determined by a subset of Bernstein polynomials chosen with a variable selection methodology. The predictions are smoothed for consecutive days to give more accurate results. We compare our proposed solution to the more conventional approach. The proposed solution is found to be better.

1 Introduction

The amount and properties of wood produced are related to the timing and rate of wood formation during the growing season. Wood formation depends on genetical signaling, availability of resources for growth, temperature, tree water and nutrient status, and the stage of ontogenic development (e.g. [1,2,3]).

Despite the basic nature of the wood formation process, our present knowledge concerning the timing and rate of the phases of tree growth is surprisingly fragmentary. An important factor behind the gaps of knowledge is the difficulty in measuring xylem formation at short intervals. Point and band dendrometers have been used traditionally to monitor cambial dynamics (e.g. [4,5]). Changes in stem dimensions are not solely a result of wood formation; they are often caused by other processes, especially changes in stem hydration [6,7,8,9]. Since dendrometers measure stem radius or circumference changes rather than wood formation, it is difficult to distinguish between true wood formation and hydrological swelling and shrinking.

In our previous study, we evaluated the use of cumulative sum (CUSUM) charts for automatically determining the onset and cessation dates of radial increase based on dendrometer data [10]. The CUSUM chart is a tool for automatically detecting small changes in the mean of a signal solely based on the data. In order to produce reliable results, one has to choose suitable parameter values for the chart and onset and cessation levels for stem radius. Once configured properly, the method produced results similar to those determined by an expert. This study is a sequel to [10].

Our aim in this paper was to provide new insights into statistical methods that could be used for predicting the onset of radial increase based on weather data. During dormancy release and growth initiation in the boreal zone, the role of temperature is more important than the role of photoperiod or water availability [11,12,13]. Therefore, we compared different statistical methods to evaluate whether the expressed signal between temperature and the onset of radial increase could be strengthened by selection of appropriate method.

In this paper, we are dealing with two stochastic processes: one concerning the environmental factors, specifically temperature, and the other concerning the radial growth of trees. Our interest lies in modeling the relationship between these processes. We control the complexity of our research problem by reducing the growth process to a single number. Thus, our goal is to predict the onset date. Similarly, the complexity in the temperature data is controlled by computing features from a reasonably long sequence of past data. A subset of the features is selected based on their performance in the prediction task.

The rest of the paper is organized as follows: Section 2 introduces the methods used, from traditional and new ways of summarizing temperatures, through variable selection, to the prediction machinery. Section 3 describes our results. Section 4 concludes the paper with a summary, some comments, and future directions.

2 Material and Methods

2.1 Traditional Temperature Sums

One of the oldest ecological concepts is the “degree-day” or “physiological time” unit which mainly stems from the relationship between development rate of many organisms and temperature (e.g. [14]). Phenological phases of plants have been found to be related to accumulation of air temperatures above some threshold below which the development stops.

Most attempts to improve the concept have dealt with either different calculation procedures used to obtain degree day estimates [15,16,17], or including additional climatic parameters, such as soil water availability (e.g. [18]). We will limit ourselves to the modification of the method [19] commonly used in Finland, which computes degree-day sum as the cumulative sum of the differences between the daily mean temperature and the threshold of $+5\,^{\circ}\mathrm{C}$ (negative values are set to zero). At least five consecutive days meeting or exceeding the threshold are needed in order for the degree-days to accumulate from a given day.[1]

2.2 Bernstein Polynomials as Basis Functions on the Past

Instead of reducing the sequence of previous daily temperatures into just one value, we can compute several temperature features. Thus by making more effective use of temperature data, we hope to improve our ability to predict the onset. For this purpose, we use Bernstein polynomials [20]:

$$B_i^d(x) = \binom{d}{i} x^i (1-x)^{d-i} \ , \quad i = 0, \ldots, d \, . \tag{1}$$

Bernstein polynomials have a special property: for any $d \in \{0, 1, 2, \ldots\}$, the polynomials of degree d sum to unity:

$$\sum_{i=0}^{d} B_i^d(x) = 1 \quad \forall\, x \in \mathbb{R} \, . \tag{2}$$

When limited to the range $x \in [0, 1]$, the Bernstein polynomials are non-negative. The polynomials are symmetric, $B_i^d(x) = B_{d-i}^d(1-x)$, and have roots at both 0 and 1, except that $B_0^d(0) = B_d^d(1) = 1$. They also have a unique maximum value in the range, at $x = i/d$ for $B_i^d(x)$. Some examples of the polynomials in the limited range are shown in Fig. 1.

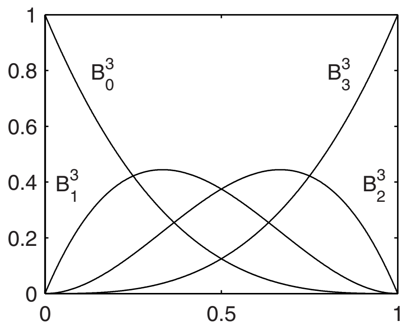


Fig. 1. The Bernstein polynomials of degree 3

The polynomials have an application related to Bézier curves [20]. We use them as weight functions in temperature features. Instead of summing temperatures above +5 °C as in the traditional temperature sum, we use the daily temperature values even if they are lower than the threshold. Thus, when we

[1] When computing the temperature sum every day in an online setting, this means a delay of up to four days, i.e. the system is not causal.

consider daily mean temperatures T_i and a history window of N previous days, the Bernstein-weighted temperature features of degree d on day i are

$$s_m(i) = \sum_{j=1}^{N} B_m^d\left(\frac{j-1}{N-1}\right) T_{i-j} \ , \quad m = 0, \ldots, d \ . \tag{3}$$

Since the Bernstein polynomials sum to one, each day in the temperature sequence gets equal weight, when we consider the combination of all temperature features (3). It is clear that each temperature sum s_m in (3) has a quite strong correlation with some s_n, at least for m and n that are close to each other. In the next section, we discuss selecting a subset of the temperature features. As a result of the selection, some days – or positions in the sequence of temperatures of N previous days – will have more weight than others.

2.3 Selecting Basis Functions

We prefer relatively sparse or parsimonious models, because they are typically easier to interpret and apply than more complex ones [21]. For selecting a suitable subset of temperature features defined in (3), we employed Best First Search (BFS) [22]. Simply testing all feature sets is infeasible due to the size of the search space.

BFS is a wrapper method, i.e. it uses the machine learning algorithm as part of the feature set search. In other words, the goodness of a feature set is defined as its performance on the machine learning task at hand. The basic idea of the algorithm is to advance in the state space, where a state represents a feature set. New states are created by adding or removing one variable (simple operator) or combining several additions or removals (compound operator) (Fig. 2). The next state chosen is the one which has the lowest cost among the candidate states. The search may start from any state. Typical choices are the empty set or the full set of variables. The search proceeds until no major improvements to the cost have been observed for a while. The exact behavior of the algorithm is controlled with two user-definable parameters: one for setting the threshold for considering a new state an improvement, the other for adjusting the stopping criterion.

2.4 Smoothed Prediction with a Combination of a Parametric and a Non-parametric Model

The k-nearest neighbor method (k-NN) is a traditional method used in classification [23], but it can also be used for regression in an analogous way. We employed a combination of a k-NN regression model and a linear regression model to predict the number of days until onset of radial increase. The aggregate prediction is simply a weighted mean of the two parts. Temperature features (3) were used as inputs and the number of days until the onset of radial increase as the target variable. Figure 3 shows an example of the output of the predictor machine, when predictions are made as a sequence, on consecutive days.

The k-NN part of the predictor finds k days of the training set (reference database) that are closest to the query day with respect to the temperature features. The prediction of the k-NN part is effectively the mean of the target variable on the neighboring days. When $\boldsymbol{s}$ is the vector of selected temperature features (3) on the query day, and $\boldsymbol{s}_i$ contains the corresponding values on reference day i, the k closest days are found by computing the Euclidean distances (4) (or equivalently squared distances) over all the reference data, sorting

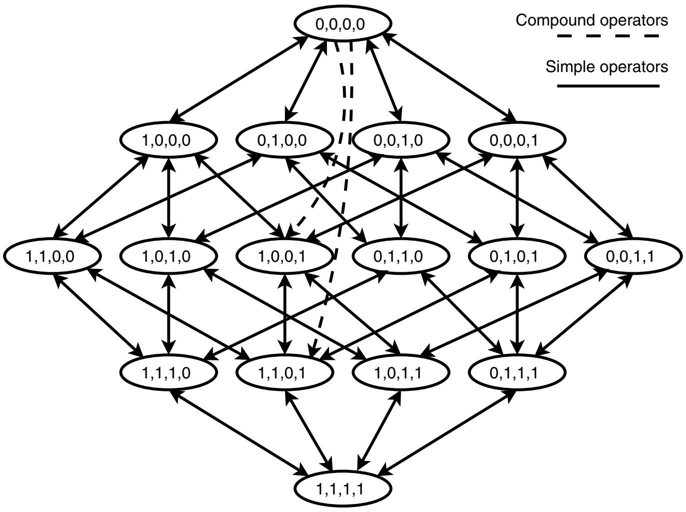


Fig. 2. Structure of state space in BFS, when selecting from four possible variables and creating new nodes using both simple and compound operators. Each state is a set of input variables (1 = "variable present", 0 = "variable absent"). Arrow from node a to node b means that b is a child of a.

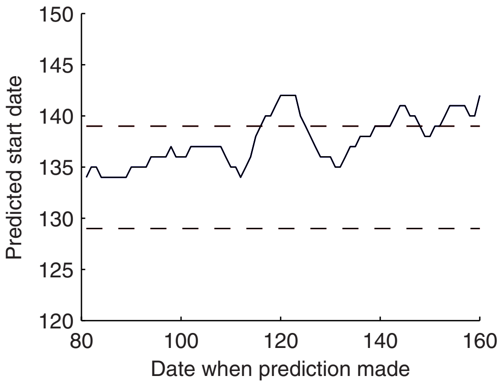


Fig. 3. Example output of the prediction machine. The measured onset region, determined by the minimum and maximum onset dates of the individual trees, is marked by the dashed lines.

the distances, and choosing the indices i that correspond to the k smallest distances. Due to the non-parametric nature of the k-NN part, the training data set becomes a part of the model, and is always needed when new predictions are made.

$$d_i = \sqrt{(\boldsymbol{s} - \boldsymbol{s}_i)^T(\boldsymbol{s} - \boldsymbol{s}_i)} \tag{4}$$

The other part of the composite predictor is a simple linear regression on selected temperature features (3), with time until onset of radial increase as the target variable. When $\boldsymbol{s}$ and $\boldsymbol{\beta}$ are the chosen temperature features (accompanied by a constant term) and regression coefficients, respectively (both column vectors), the regression model is as in (5). The regression result y_r is rounded to the closest integer, which may be considered superfluous, because the final output of the combined model is also rounded.

$$y_r = \boldsymbol{s}^T\boldsymbol{\beta} \tag{5}$$

In an ideal situation, the predictor would always point to the same onset date, independent of the day on which the prediction is made. When the predictions are made as a sequence, the predicted onset date will, however, fluctuate. We did a simple smoothing of the predictor's output by taking a weighted mean of the new prediction and the previous one. Then the effect of old, unsmoothed predictions, undergoes exponential decay as the prediction sequence is extended. The smoothing operates with unrounded values, but the final output is rounded to a precision of one day. Since the predictor actually gives the time until onset, the number of days separating the previous and current predictions is subtracted from the previous result before it is used as part of the new prediction. The smoothing process (without integer rounding) can be written as

$$\begin{aligned}\tilde{y}_i &= \alpha y_i + (1-\alpha)\tilde{y}_{i-1} \\ &= \alpha[y_i + (1-\alpha)y_{i-1} + \ldots + (1-\alpha)^{i-2}y_2] + (1-\alpha)^{i-1}y_1\ ,\end{aligned} \tag{6}$$

where $\alpha \in (0,1]$ adjusts the level of smoothing, $\tilde{y}_i$ is the i:th smoothed prediction in the prediction sequence, and y_i is the corresponding "novelty", the new part in the prediction. Note that the first prediction in the sequence is a special case, where no smoothing is applied.

2.5 Dendrometer Data

Sample trees were selected at two sites located 300 m from each other in Tuusula, southern Finland. In the first stand, the Norway spruce trees were growing in a pure spruce stand on fertile mineral soil classified as *Oxalis-Myrtillus* forest type [24]. Mean stem diameter of the sample trees at breast height was 27 cm and relative crown length was 68 %. The sample trees were monitored during the growing seasons of 2001–2005. In the second stand, four Norway spruce and four Scots pine trees were monitored during the growing seasons of 2002–2003, and another four spruce and four pine trees during the growing seasons of 2004–2005. They were growing in a mixed spruce-pine stand on a relatively fertile mineral soil classified as *Myrtillus* forest type [24]. The total number of observations (year $\times$ tree combinations) was, thus, 57. The stands and sample trees have been described in detail in [10].

Stainless-steel band-dendrometers were installed on each tree at a height of about 2 m. Changes in tree girth were measured at a resolution of 0.1 mm, corresponding to diameter change of about 0.03 mm. The output of the dendrometers was stored as 1-h averages. From these measurements, the daily values of stem circumference were calculated as the mean of hourly values and the circumference changes were converted to radial changes assuming a circular stem cross-section. The dendrometer has been described in detail in [5].

For each year and tree, the onset date of radial increase was determined visually and verified by the CUSUM method [10]. Depending on the year, we have the onset date of either 5 (year 2001) or 13 (2002–2005) trees. Because we wanted to evaluate the methods described above, i.e. detailed biological interpretation of the increment onset is beyond the scope of this study, we averaged the data from both sites and tree species. As mentioned above, swelling and shrinkage of the tree, caused by changes in water level, add noise to the data.

Temperature data at 3-hour intervals were obtained from a meteorological station of the Finnish Meteorological Institute located about 5 km from the study stands. From the temperature measurements, the average daily values were calculated as arithmetic means.

2.6 Test Setting

We run feature selection with BFS, selecting from a total of 40 or 20 temperature features weighted with Bernstein polynomials, corresponding to all polynomials of degree 39 or 19, respectively. The search is started from the empty set of features. Years 2001–2004 are used for the feature selection in a 4-fold cross-validation setting, where the onset date on one year is predicted based on data from the remaining three years. Year 2005 is reserved as a (very limited) test set for later use. From each year, days 81–200 are included, counting from the start of the year. The temperature features are computed from 80 previous days, i.e. the temperatures on days $\{x-80,\ldots,x-1\}$ are reduced to 40 or 20 numbers to be associated with day x.

The feature selection also incorporates the selection of k for k-NN. In our setting, the cost function of BFS tests the prediction machine with values of k ranging from 5 to 50 in steps of 5. The cost is the MSE of all predictions, where the predictions are made for each date in the range of 101–160 (April 11–June 9 in the case of "no leap year"). For comparison, the measured annual average onset dates range from 133 to 144. The average was taken over all the years; there are $4 \cdot (160 - 101 + 1) = 240$ error values to be averaged. In the feature selection phase, the prediction sequence is not smoothed: $\alpha = 1$ in (6). For later tests, $\alpha = 0.4$ was used, and the predictor was started at day 81, which allowed a stabilizing period of 20 days before the measurement of prediction error. The value of α was chosen by non-rigorous testing towards the goal of minimizing error. The results are not sensitive to the specific choice of this parameter.

3 Results

The traditional temperature sum alone does not provide an accurate prediction about the onset date (Fig. 4). When excluding the smallest values of temperature sum, the average onset date over different years could quite well be explained by a straight line. The variance between years, however, is large.

The performance of the Bernstein-weighted temperature features is presented in Table 1. The bottom part of the table lists comparative results, with no feature selection or using just the traditional temperature sum. In these cases, the k range tried was 10 to 110 in steps of 5. RMSE 1 means the square root of the error measure used in the feature selection tests: the mean of squared errors in the predictions sequences. For RMSE 2, each annual prediction sequence is reduced to a single error measure, where the onset date given by the predictor is set to be the first date in the sequence, where the predicted onset date and the date of prediction intersect, i.e. according to the predictor, growth has already started. The error is derived from the difference of this intersection date and the date derived from the girth band data. When only predicting the onset on one year, RMSE 2 would reduce to the absolute value of this error, but in the table, we also show the sign of the error. “Valid.” is the error in the validation tests used for selecting the features and k. “Test” and “Pseudo” are errors done with the selected features and k, using data from all the year (2001–2005). In the former case, only the year 2005 is predicted with the other years serving as reference data. In the latter case, all the years are predicted one by one, with the remaining years as reference data.

With our settings for BFS, the number of states evaluated was about 8000 out of 2^{20} or 25000–35000 out of 2^{40}. The search was set to stop, when the 500 latest state expansions have not resulted in a cost smaller than 1.001 times the best known cost.

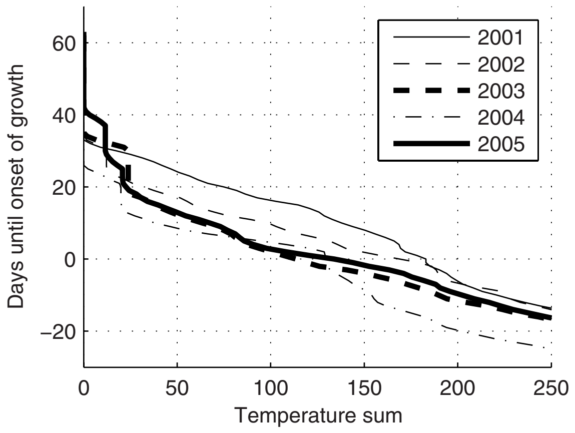


Fig. 4. The number of days until the onset of radial increase plotted against temperature sum in different years

Table 1. Prediction error. The k-NN component was always used. Also linear regression was used in some tests (see first column). The bottom part of the table shows comparative results. The column marked with * holds the signed absolute error instead of RMSE.

	#Features			RMSE 1 (sequence)			RMSE 2 (i.sect)		
Model	Chosen	Total	k	Valid.	Test	Pseudo	Valid.	Test*	Pseudo
lin + k-NN	16	40	5	5.2	5.0	4.9	6.4	5	5.8
k-NN	16	40	35	5.3	5.2	5.0	5.5	−2	4.5
lin + k-NN	8	20	5	5.4	6.4	5.3	5.1	6	5.5
k-NN	9	20	25	5.4	3.5	4.8	6.6	5	6.0
lin + k-NN	40	40	90	5.7	5.9	5.2	4.9	2	4.4
lin + k-NN	20	20	105	5.8	6.0	5.2	5.2	2	4.4
k-NN on temperature sum			50	8.5	2.5	7.0	8.7	1	6.9

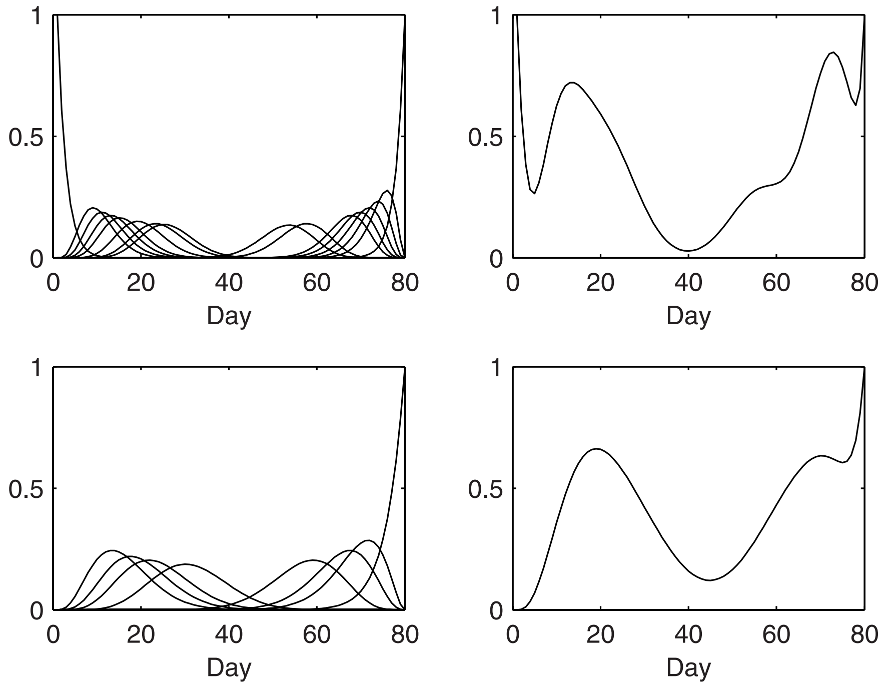


Fig. 5. Bernstein weighting functions chosen in rows 1 and 3 of Table 1. In the figures, the most distant past is at the right end of the x-axis. Top row: 16/40 features. Bottom row: 8/20 features. Left column: individual weighting functions. Right column: sum of weights on the left.

We look at the "RMSE 1" results first, because that is the optimization criterion in the validation tests. The tests, which average the error over several years, show that the Bernstein polynomial temperature features produce prediction accuracy superior to the traditional temperature sum. Adding one year of data always improved the average performance. The results from the prediction of a single year's growth (column "Test") show that although close to each other on the average, the different predictor settings may give quite different results for any individual year. Note that the "truth" used as the basis of the error measure is quite uncertain: when considering the onset dates suggested by each girth band on a given year, the standard deviation ranges from about 3.7 to 9.0, averaging at 6.8 (days). The error measure "RMSE 2" shows generally similar results in the sense that the traditional temperature sum is outperformed by the new features. Also here the variation in a single year's results is clearly seen.

The optimal k decreases noticeably, when linear regression is also used in the predictor machine. It seems that the addition of the simple regression model, quite logically, regularizes the output of the k-NN model. A similar effect is achieved with an increased k. When feature selection is omitted (the first two rows in the bottom part of Table 1), the optimal k is very large despite the presence of linear regression.

Figure 5 shows the features chosen by BFS in two of our tests. The results look almost the same. In both cases, some highly correlated features are included, but features concentrated in the middle part of the history window are curiously missing.

4 Summary and Conclusions

In this paper, we presented a method for predicting the onset of stem radius increase. The method is based on temperature features weighted with Bernstein polynomials, and a combination of two simple regression methods operating on those features. A subset of the features was selected with BFS. The motivation for the development of the features was to extract more information from the temperature time series than what is possible with the traditional temperature sum, thus improving prediction performance.

In our cross-validation tests, the proposed temperature features outperformed the traditional temperature sum. However, prediction accuracy still shows a large year-to-year variation. There are at least two plausible reasons for the variation: the nature of the dendrometer data (swelling, shrinking), and factors left out of the model (everything but temperature).

After these initial steps with the prediction framework, we can think of applications for it. For example, we could try to find a possible trend in the past onset dates. However, the potential differences between the current and past environment would have to be considered: have the environmental factors not included in the model changed so much over the years, that a model developed using present data is not valid for the past? Another application possibility is the modeling of phenomena other than the onset of stem radius increase.

Acknowledgments. This work has been funded by the Academy of Finland, in the research project "Analysis of dependencies in environmental time series data"

(AD/ED), grant number 116853. We are greatly indebted to Tapio Huttunen for skilful technical assistance in maintenance of the girth bands.

References

1. Fritts, H.C.: Tree Rings and Climate. Academic Press, London (1976)
2. Savidge, R.A.: Xylogenesis, genetic and environmental regulation. IAWA Journal 17, 269–310 (1996)
3. Vaganov, E.A., Hughes, M.K., Shashkin, A.V.: Growth Dynamics of Conifer Tree Rings. Ecological Studies, vol. 183. Springer, Heidelberg (2006)
4. Yoda, K., Suzuki, M., Suzuki, H.: Development and evaluation of a new type of opto-electronic dendrometer. IAWA Journal 21(4), 425–434 (2000)
5. Pesonen, E., Mielikäinen, K., Mäkinen, H.: A new girth band for measuring stem diameter changes. Forestry 77(5), 431–439 (2004)
6. Kuroda, K., Kiyono, Y.: Seasonal rhythms of xylem growth measured by the wounding method and with a band-dendrometer: an instance of Chamaecyparis obtusa. IAWA Journal 18, 291–299 (1997)
7. Deslauriers, A., Morin, H., Urbinati, C., Carrer, M.: Daily weather response of balsam fir (*Abies balsamea* (L.) Mill.) stem radius increment from dendrometer analysis in the boreal forests of Québec (Canada). Trees 17(6), 477–484 (2003)
8. Mäkinen, H., Nöjd, P., Saranpää, P.: Seasonal changes in stem radius and production of new tracheids in norway spruce. Tree Physiology 23(14), 959–968 (2003)
9. Mäkinen, H., Seo, J.W., Nöjd, P., Schmitt, U., Jalkanen, R.: Seasonal dynamics of wood formation: a comparison between pinning, microcoring and dendrometer measurements. European Journal of Forest Research (in print, 2008)
10. Sulkava, M., Mäkinen, H., Nöjd, P., Hollmén, J.: Detecting onset and cessation of tree stem radius increase based on dendrometer data. Environmetrics (submitted, 2008)
11. Partanen, J., Leinonen, I., Repo, T.: Effect of accumulated duration of the light period on bud burst in Norway spruce (Picea abies) of varying ages. Silva Fennica 35(1), 111–117 (2001)
12. Deslauriers, A., Morin, H., Begin, Y.: Cellular phenology of annual ring formation of Abies balsamea in the Quebec boreal forest (Canada). Canadian Journal of Forest Research 33(2), 190–200 (2003)
13. Seo, J.W., Eckstein, D., Jalkanen, R., Rickebusch, S., Schmitt, U.: Estimating the onset of cambial activity in Scots pine in northern Finland by means of the heat-sum approach. Tree Physiology 28, 105–112 (2008)
14. Wang, J.Y.: A critique of the heat unit approach to plant response studies. Ecology 41(4), 785–790 (1960)
15. Baskerville, G.L., Emin, P.: Rapid estimation of heat accumulation from maximum and minimum temperatures. Ecology 50(3), 514–517 (1969)
16. Allen, J.C.: A modified sine wave method for calculating degree-days. Environmental Entomology 5, 388–396 (1976)
17. Zalom, F.G., Goodell, P.B., Wilson, L.T., Barnett, W.W., Bentley, W.J.: Degree-days: The calculation and use of heat units in pest management. Leaflet 21373, University of California, DANR (1983)
18. Idso, S.B., Jackson, R.D., Reginato, R.J.: Extending the "degree day" concept of plant phenological development to include water stress effects. Ecology 59(3), 431–433 (1978)

19. Sarvas, R.: Investigations on the annual cycle of development of forest trees. Active period. Communicationes Instituti Forestalis Fenniae 76(3), 1–110 (1972)
20. Prautzsch, H., Boehm, W., Paluszny, M.: Bézier and B-Spline Techniques. Springer, Heidelberg (2002)
21. Sulkava, M., Tikka, J., Hollmén, J.: Sparse regression for analyzing the development of foliar nutrient concentrations in coniferous trees. Ecological Modelling 191(1), 118–130 (2006)
22. Kohavi, R., John, G.H.: Wrappers for feature subset selection. Artificial Intelligence 97(1–2), 273–324 (1997)
23. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd edn. John Wiley & Sons, Chichester (2001)
24. Cajander, A.K.: Forest types and their significance. Acta Forestalia Fennica 56(5), 1–71 (1949)

Feature Selection in Taxonomies with Applications to Paleontology

Gemma C. Garriga, Antti Ukkonen, and Heikki Mannila

HIIT
Helsinki University of Technology and University of Helsinki, Finland

Abstract. Taxonomies for a set of features occur in many real-world domains. An example is provided by paleontology, where the task is to determine the age of a fossil site on the basis of the taxa that have been found in it. As the fossil record is very noisy and there are lots of gaps in it, the challenge is to consider taxa at a suitable level of aggregation: species, genus, family, etc. For example, some species can be very suitable as features for the age prediction task, while for other parts of the taxonomy it would be better to use genus level or even higher levels of the hierarchy. A default choice is to select a fixed level (typically species or genus); this misses the potential gain of choosing the proper level for sets of species separately. Motivated by this application we study the problem of selecting an antichain from a taxonomy that covers all leaves and helps to predict better a specified target variable. Our experiments on paleontological data show that choosing antichains leads to better predictions than fixing specific levels of the taxonomy beforehand.

1 Introduction

Prediction and classification are popular tasks in data analysis. The input examples to these tasks are typically described in a vector space with the dimensions of a set of features. The goal of the learning algorithm is to construct a model that will predict a target variable or a class associated to the given input examples. The set of features describing the examples is crucial for the performance and accuracy of the final learned model. Often, combinations of the input features can provide a much better way to explain the structure of the data. The problem of feature selection and feature construction is nowadays an important challenge in data mining and machine learning [2,4,10].

We study feature selection problems on taxonomies. Taxonomies occur in many real-world domains and add a richer representation to the plain set of features. Consider a paleontological application, with a set of fossil species observed across different sites. Commonly, species are categorized into several taxonomic levels exhibiting the structure of a tree. A simple snapshot of a part of the primate taxonomy is shown in Figure 1. We have, for instance, that *Pliopithecidae* and *Hominidae.* are primates; and the genus *Homo* belongs to the family *Hominidae.* A family or genus is considered to occur at a site if at least one of the

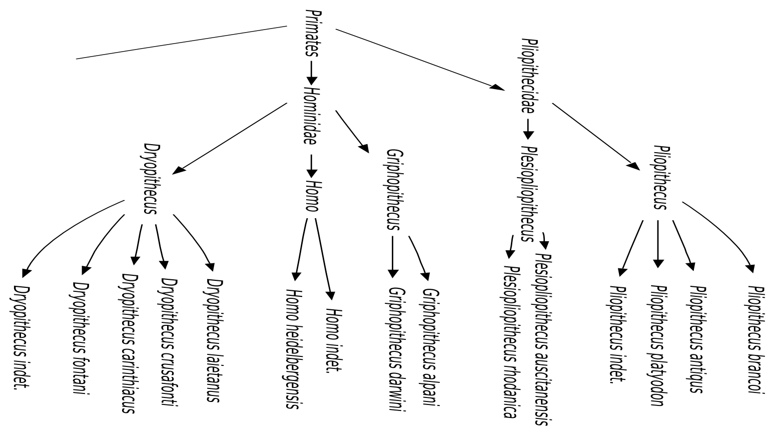


Fig. 1. Taxonomy of the primates species for a paleontological application

species below it in the taxonomy occurs there. In general, an occurrence of a internal node occurs if and only if at least one of the leaves below it occurs.

A fundamental task in this paleontological application is to predict the age of the site from the observed taxa in the data [7,8,9,12]. Only few sites have accurate age determinations from radioisotopic methods, and stratigraphic data (information about the layers in which the fossils are found) is also often missing. Thus the data about the taxa found at the site is all we have for determining the age of the site. The prediction task can be based on using different levels of the taxonomic hierarchy. Selecting aggregates of the features at the proper level of the taxonomy tree is critical. For example, combining the observed species at the genus level of the taxonomy can provide a much better predicting accuracy than using directly the leaf (species) level.

A common solution in practice is to choose a fixed level of the taxonomy to represent the new aggregated features; this implies combining all the species at the same height of the taxonomy tree. Although this solution is natural, it misses the potential gain of aggregating sets of species at different levels of the taxonomy separately. A toy illustration is given in Figure 2: in (a) we have small binary dataset with features from a to e and a given taxonomy on top of them; the variable we wish to predict is the real-valued named Age. In the paleontological example features are species and the age would represent how old each site is. Choosing nodes y and z from the taxonomy with the aggregator of logical "OR" we can better uncover the hidden structure in the data, shown in (b). We have that a, b, c have been aggregated at height 2 and d, e, at height 1 of the taxonomy level. For this example, having the different levels of aggregation is much better than selecting always the level of species (leaf nodes of the taxonomy) or any fixed level.

Motivated by the paleontological application [7,8,9,12], the general computational problem we consider is the following: select the best subset of the nodes in the taxonomy which are not comparable (i.e., form an antichain) and still cover all leaves, in order to improve the prediction accuracy of a target variable.

Note that this definition is also useful for other applications, for instance market basket data analysis, where we may wish to predict the age of a customer from the products in the shopping basket. Also for the market basket data application we have taxonomies available at the level of the items: e.g., wines and beers both belong to the category of alcoholic beverages.

We address the complexity of the problem, present several algorithms inspired by traditional feature selection approaches, yet exploiting the taxonomy tree structure, and show how to sample uniformly at random non-comparable nodes from the taxonomy tree. Our experiments on real paleontological data show that antichains are natural for this application: they often represent a better refined antichain than the one obtained by fixing the level of the taxonomy beforehand, and typically, the predictions are then more accurate.

2 Problem Definition

Let F be a set of features. For the paleontological application, F would correspond to the set of species observed in the data. A *taxonomy* $\mathcal{T}$ on the feature set F is a rooted tree whose leaf nodes are exactly the elements of F. For any node $x \in \mathcal{T}$ we define $\mathcal{T}(x)$ to be the set of leaf nodes whose ancestor x is. For the root node r of T we have that $\mathcal{T}(r) = F$. A taxonomy T defines a partial order between nodes $x, y \in \mathcal{T}$, as follows. We say that x precedes y, denoted $x \preceq y$, whenever $\mathcal{T}(y) \subseteq \mathcal{T}(x)$, i.e., x is an ancestor of y. If $x \not\preceq y$ and $y \not\preceq x$ we say the nodes are not comparable. Additionally, we denote with children(x) the children of a node $x \in \mathcal{T}$, and with parent(x) its parent.

An example of a taxonomy $\mathcal{T}$ is shown on top of Figure 2(a). The leaf nodes are the set of species $F = \{a, b, c, d, e\}$. The internal nodes of the taxonomy x, y, z and r represent possible categorizations of the data attributes: for instance $\mathcal{T}(y) = \{a, b, c\}$ could correspond to the grouping of carnivores; $\mathcal{T}(z) = \{d, e\}$ could correspond to herbivores and $\mathcal{T}(r) = \{a, b, c, d, e\}$ to all animals. We have that $y \preceq x$; y and z are not comparable; also, children$(r) = \{y, z\}$.

An *antichain* $X = \{x_1, \ldots, x_k\}$ of a taxonomy $\mathcal{T}$ is a subset of nodes from $\mathcal{T}$ that are not comparable. A *covering antichain* X is an antichain that covers all leaves, i.e. $\cup_{x \in X} \mathcal{T}(x) = F$. In the toy example in Figure 2, for instance $\{x, z\}$ is an antichain; an example of a covering antichain would be the set of nodes $\{x, c, z\}$.

Additionally, consider an $n \times m$ data matrix D where each row vector is defined along m dimensions of the feature set F. In the paleontological application the rows of the matrix correspond to sites and columns to species. We denote by D^f the $n \times 1$ column vector of D of a feature $f \in F$. In our application, D^f is a column vector with information of the absence/presence of species f in each site of the data. A useful alternative description of the matrix D is to see it as the the collection of column vectors D^f from $f \in F$, that is: $D = D^{f_1} : \ldots : D^{f_m}$ for all $f_i \in F$, where ":" denotes juxtaposition of column vectors.

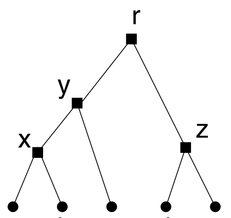
Given a node $x \in \mathcal{T}$, denote by $D(x)$ the aggregation of the columns in D covered by x. Formally, $D(x) = \alpha_x(D^{f_1}, .., D^{f_k})$ with $f_i \in T(x)$, where α_x

is a function computed over the input columns selected by x and returning a $n \times 1$ column. For example, α_x could be "OR" or "AND", over the covered columns; in the paleontological application it is typically "OR". In general, for a set of nodes $X = \{x_1, \ldots, x_k\}$ from $\mathcal{T}$ we let $D(X) = D(x_1) : \ldots : D(x_k)$ be the concatenation of projections from each node in X. Consider the example in Figure 2 and assume that the aggregation function α_x associated to every node $x \in \mathcal{T}$ is a logical "OR". The result of $D(\{y, z\})$ is shown in Figure 2(b).

Projecting the data on a set of nodes of the taxonomy often returns much better aggregates than the original values. To evaluate the quality of the different data projections given by a set of nodes we will consider a target variable $v \notin F$ whose value we wish to predict, i.e., we are interested in predicting the values of the column vector D^v. In Figure 2 the variable v corresponds to the age of the site. The goal of a learning algorithm $A(D, v)$ (e.g., a predictor or classifier depending on the domain of v) is to construct a model from D, in order to predict v for new unseen data points. We denote with $\text{err}[A(D, v)]$ the error returned by the inferred model. The error can be calculated, e.g., as the square of the differences between the predictions of the model $A(D, v)$ and the real value of v in a separate test set.

Following the example from Figure 2, the projection $D(\{y, z\})$ would allow a learning algorithm to clearly distinguish between two different segments of age: those that are younger and those that are older than 40 million years.

We study the following problem on feature selection on taxonomies.



a	b	c	d	e	Age
1	0	0	0	0	29
0	1	0	0	0	31
0	0	1	0	0	27
1	0	1	0	0	32
0	1	1	0	0	29
0	0	0	1	0	50
0	0	0	0	1	62
0	0	0	1	1	55

(a)

D(y)	D(z)	Age
1	0	29
1	0	31
1	0	27
1	0	32
1	0	29
0	1	50
0	1	62
0	1	55

(b)

Fig. 2. An example of a taxonomy tree on top of a set of features $F = \{a, b, c, d, e\}$ (a), and after projecting the data on nodes y and z of the taxonomy by means of an "OR" aggregator over the columns covered by these nodes (b).

Algorithm 1. `Greedy` algorithm

```
1: Input: Data $D$; a taxonomy $\mathcal{T}$; a learning algorithm $A$.
2: Output: Antichain $X$ from $\mathcal{T}$ and $\text{err}[A(D(X), v)]$
3: $X = \emptyset$
4: $N =$ all nodes from $\mathcal{T}$
5: repeat
6:     $x^* = \arg\min_{x \in N}\ \text{err}[A(D(X \cup \{x\}), v)]$ {Take the best next node from $\mathcal{T}$}
7:     $X = X \cup \{x^*\} \cup \{n \in N | n$ is sibling leaf of $x^*\}$
8:     $N = N \backslash \{n \in N | n \preceq x^*$ or $x^* \preceq n\}$
9: until $N$ is empty
10: Return $X$ and $\text{err}[A(D(X), v)]$
```

*Problem 1 (*TAXONOMY ANTICHAIN SELECTION). Given a dataset D defined over attributes $\mathcal{F} \cup \{v\}$, and let $\mathcal{T}$ be a taxonomy tree on the set F. Select a covering antichain X of T that minimizes $\text{err}[A(D(X), v)]$ for the variable v.

We refer to the TAXONOMY ANTICHAIN SELECTION problem as TAS. The idea of finding an antichain that covers all leaf nodes is natural in several applications: antichains represent sets of nonredundant features that, potentially, will explain the structure of the data much better than having an unmanageable number of leaf features F. This is especially the case in paleontology, where we would like aggregations of the species at different levels.

Proposition 1. *The* TAS *problem is NP-complete.*

This proposition is proven via a reduction from the SATISFIABILITY problem for certain choices of the aggregation function. Details are omitted due to space constraints.

3 Algorithms

This section describes four algorithms for the TAS problem. As a baseline for our proposals we also show how to sample antichains uniformly at random. Without loss of generality we assume that the taxonomy $\mathcal{T}$ is rooted, i.e., there is a node x such that $\mathcal{T}(x) = F$.

3.1 The Greedy Algorithm

The scheme of `Greedy` approach is shown in Algorithm 1. The idea is simple and inspired by a forward selection technique: at every step `Greedy` takes the node $x \in \mathcal{T}$ with the least error increase (line 6). To enforce the antichain constraint, all nodes in the path from/to the selected node x have to be removed as they cannot occur together with x in the same solution set. An incremental solution is constructed until all leaf nodes have been covered. Notice that when selecting

Algorithm 2. `Bottom-up Selection` algorithm

```
 1: Input: Data $D$; a taxonomy $\mathcal{T}$; a learning algorithm $A$.
 2: Output: Antichain $X$ from $\mathcal{T}$ and $\text{err}[A(D(X), v)]$
 3: $X = F$ {Initialize with leaves from $\mathcal{T}$}
 4: $e = \text{err}[A(D(X), v)]$
 5: $B^+ = \{y \mid y \in \mathcal{T},\ \text{children}(y) \subseteq X\}$ {Nodes from $\mathcal{T}$ upper bordering $X$}
 6: repeat
 7:   for all $n \in B^+$ do
 8:     $X'_n = X \setminus \text{children}(n) \cup \{n\}$ {Swap children of $n$ with $n$ in the antichain}
 9:     $e'_n = \text{err}[A(D(X'_n), v)]$
10:   end for
11:   $n^* = \arg\min_{n \in B^+} e'_n$ {Take the best from the above swaps}
12:   if $e'_{n^*} \leq e$ then
13:     $X = X'_{n^*}$
14:     $e = e'_{n^*}$
15:     $B^+ = B^+ \setminus n^* \cup \{y \mid y = \text{parent}(n^*),\ \text{children}(y) \subseteq X\}$ {Update $B^+$}
16:   end if
17: until Error $e$ cannot be further reduced
18: Return $X$ and $\text{err}[A(D(X), v)]$
```

a leaf node, this will always enforce the addition of all its sibling leaves into the solution set (line 7); that selecting a leaf node will always enforce the addition of all its siblings into the solution set, this is inherent to the requirement of finding an antichain that covers all leaf nodes.

3.2 Top-Down and Bottom-Up Selection Algorithms

The following solutions are inspired by traditional backward elimination algorithms. In our problem though, the steps for feature selection have to take into account the topology of the taxonomy tree and ensure the antichain constraint of the selected set of nodes. The scheme of the `Bottom-up Selection` algorithm is given in Algorithm 2. `Bottom-up Selection` starts by taking all the leaf nodes F as an initial antichain X (line 3). At all times, the positive border B^+ maintains the nodes from the taxonomy tree located right above the current antichain X (lines 5 and 15). The algorithm iterates to improve the solution in a bottom-up fashion: first, it evaluates all swaps between a node in the border B^+ and its children (currently belonging to the solution set) by calculating and storing the error loss (lines 7–10); then, the antichain X is updated with the best of those swaps (lines 12–16). The algorithm stops when the error cannot be further reduced with any of the swaps. A complementary approach is the `Top-down Selection` algorithm. The starting point X is initialized to be the root node of the taxonomy, and the negative border B^- maintains at all times nodes right below the current antichain solution. Similarly as before, the algorithm would try to find a better antichain by taking the best swap in a top-down fashion from the taxonomy tree.

Algorithm 3. `Taxonomy Min-Cut` algorithm

```
1: Input: Data $D$; a taxonomy $\mathcal{T}$; a learning algorithm $A$.
2: Output: Antichain $X$ from $\mathcal{T}$ and err$[A(D(X), v)]$
3: Let $\mathcal{T} = (V_\mathcal{T}, E_\mathcal{T})$ be the vertices and edges of the taxonomy $\mathcal{T}$
4: Let $G = (V, E)$ be an undirected weighted graph as follows,
5:     $V = V_\mathcal{T} \cup \{s, t\}$ {Nodes from $\mathcal{T}$, plus a source $s$ and a target $t$}
6:     $E = E_\mathcal{T} \cup \{(s, r) \mid r$ is root of $\mathcal{T}\} \cup \{(f, t) \mid f \in F\}$
7: Set the weight $w(e)$ of edges $e \in E$
8: for edge $e = (x, y) \in G$ do
9:   if $y$ is the target node $t \in G$ then
10:     $w(e) = +\infty$
11:   else
12:     $w(e) = \text{score}(y) \times h(y)$
13:   end if
14: end for
15: Find the minimum cut edges $C$ of $G$ {E.g. with Max-flow algorithms}
16: $X = \{y \mid (x, y) \in C\}$ {$X$ is the set of nodes selected below the min-cut}
17: Return $X$ and err$[A(D(X), v)]$
```

3.3 Minimum Cut Based Algorithm

The algorithm `Taxonomy Min-Cut` is based on finding the minimum cut of a graph G derived from $\mathcal{T}$. The MINIMUM CUT problem on a weighted undirected graph asks for a partition of the set of vertices into two parts such as the cut weight (sum of the weights on the edges connecting the two parts) is minimum. This problem can be solved using the max-flow min-cut theorem [1,6].

We map the antichain selection problem into a MINIMUM CUT problem by constructing an undirected graph G which is a simple augmented version of $\mathcal{T}$. A scheme is shown in Algorithm 3. First, we extend taxonomy $\mathcal{T}$ with two extra nodes: source node s and target node t (line 5); second, we add edges between the root of $\mathcal{T}$ and the source s, and between leaves of $\mathcal{T}$ and the target node t (line 6). For notational convenience we consider the undirected edges $(x, y) \in G$ to be implicitly directed towards the target node t, that is, whenever $x \preceq y$ we will always write the edge as (x, y). For example, we always have $(s, r) \in G$, being r the root of $\mathcal{T}$; also $(f, t) \in G$ for all leaves $f \in F$ of $\mathcal{T}$.

An s, t-cut is then a set of edges in G whose removal would partition G into two components, one containing s and the other containing t. The following property follows.

Proposition 2. *Let $\mathcal{T}$ be a taxonomy. Then every s, t-cut from G not containing edges from $t \in G$ corresponds to a covering antichain and vice versa.*

Briefly, for a set of s, t-cut edges C of G, we have a covering antichain $X = \{y \mid (x, y) \in C\}$. That is, nodes belonging to the antichain are just below the s, t-cut. Similarly, each antichain X in $\mathcal{T}$ identifies an s, t-cut C separating source and target in G: we only need to select the edges $C = \{(x,\text{parent}(x)) \mid x \in X\}$.

Algorithm 4. `Antichain Sampler` algorithm

```
1: **Input:** A taxonomy $\mathcal{T}$ rooted at $r$
2: **Output:** Random antichain $X$ from $\mathcal{T}$
3: Flip a biased coin that comes up heads with probability $1/\beta(r)$
4: if heads then
5:     $X = \{r\}$
6: else
7:     for nodes $y \in \text{children}(r)$ do
8:         Let $Z_y$ = Antichain Sampler($y$) {Recursive call with a tree rooted at $y$}
9:     end for
10:    $X = \bigcup_{y\in \text{children}(r)} Z_y$
11: end if
12: Return $X$
```

To use the min-cut idea, it only remains to set the weights of the edges in G appropriately (lines 8–14, Algorithm 3). We define the weight of an edge $(x, y) \in G$ to be the product of two factors: (1) the score of y, namely score(y), and (2) the height factor of y, namely $h(y)$. The score (1) of a node will depend on the data D. Because a min-cut algorithm looks for a minimum weight cut in G, the small values of score of a node indicate good quality of the node. For example, as the score for a node $x \in \mathcal{T}$ we can use the inverse of the correlation coefficient between x and the target variable v, that is $1/\rho(x, v)$, assuming $+\infty$ when $\rho(x, v) = 0$; or also, we can take directly err$[A(D(x), v)]$ as the score.

The height factor (2) of a node has to be inversely proportional to the distance from that node to the root of $\mathcal{T}$. The reason is that, by construction, the s, t-cuts close to the root contain less edges. E.g., the single edge $(s, r) \in G$ forms a very small s, t-cut on its own and might be selected even with bad score. Typically, we use as $h(x)$ either $1/\text{height}(x)$, or directly $h(x) = \mathcal{T}(x)$ (that is, the number of leaves covered by x). This last choice of $h(x)$ maintains the natural property of making all the s, t-cuts equally costly if nodes in the taxonomy are all equally good in their score function. Finally, the weight of edges involved with the target node t are set to $+\infty$ (line 10, Algorithm 3) accordingly with Proposition 2.

In practice, the `Taxonomy Min-cut` only requires to evaluate the model once per node; this is done when setting the scores of the edges (line 7, Algorithm 3), so the number of calls to the classifier is linear. On the other hand, the algorithms such as `Greedy` or `Bottom-up` have to call the classifier at each evaluation step.

3.4 Sampling a Random Antichain

As a baseline to compare the previous approaches we use random antichains. Sampling them uniformly turns out to be an interesting problem in its own right. Formally, given the taxonomy tree $\mathcal{T}$ rooted at $r \in \mathcal{T}$, let $\beta(r)$ be the total number of covering antichains of $\mathcal{T}$. Immediately, the recursion follows: $\beta(r) = \prod_{x\in \text{children}(r)} \beta(x) + 1$. For every $x \in \mathcal{T}$ we have that $\beta(x)$ corresponds to the number of antichains that can be sampled from the subtree of $\mathcal{T}$ rooted at x. For the leaf nodes $f \in F$ we have always that $\beta(f) = 1$. The scheme of the `Antichain Sampler` algorithm is shown in Algorithm 4. The proof of the following proposition comes naturally by induction on the recursion. Details are omitted in this paper due to space constraints.

Proposition 3. *The* `Antichain Sampler` *algorithm samples antichains from* $\mathcal{T}$ *uniformly at random.*

4 Experiments

The paleontological data contains information of fossil mammals in Europe and Asia [8]. Columns correspond to species and rows correspond to sites. There are originally around 2000 sites and 900 species. Because the raw data contains rare species, as well as sites with only few species, we omit some of these from consideration by creating variations of the original dataset. In the variations named PALEO_X_Y, we first remove species that occur less than X times in the raw data; after this, we remove sites that have less than Y species after the elimination of the rare species. For the experiments we use the datasets PALEO_10_10, PALEO_5_5 and PALEO_2_2. Note that by removing species and sites we are making the data more sparse, hence PALEO_2_2 is the sparsest data matrix of the three variations. The sizes of the taxonomies for these three datasets ranges from 700 and 2500 nodes. Also, in addition to the entire taxonomy, we will consider different subtrees separately, each one of which corresponds to an order, such as carnivores or rodents. The task is always to estimate the age of fossil discovery sites with linear regression.

We implemented the proposed algorithms as components of the Weka machine learning software.[1] For the calculation of the error function $\text{err}[A(D(x), v)]$ and evaluation of the models, we divide the dataset in two folds, training and test. The error used in the model construction phase is based on error in the test data. After that, we compare the solutions of the different algorithms with three criteria: (1) size in number of nodes of the returned antichain; (2) number of calls to the linear regressor needed to learn the final model (this provides an idea of its running time); (3) the correlation coefficient between predictions and actual ages in the test data; and (4) the root mean squared error of the model (RMSE, again in the test data).

The baseline for algorithms is always the random antichain. We complement each one of the experiments with a histogram of the correlation coefficients of the real and predicted ages of 100 random antichains when necessary. We also compare the algorithms with models that are based on selecting a certain level of the taxonomy. These levels are (from general to specific) FAMILY, GENUS and SPECIES. The SPECIES level consists of the leaf nodes, while the GENUS (FAMILY) level is located one (two) step(s) above the leaf level.

Table 1 reports the results when using the complete taxonomy tree available over the features. In PALEO_10_10 fixing the antichain at the level of the leaves (SPECIES) is the best obtained solution. In this case, algorithm `Bottom-up` is the

[1] `http://www.cs.waikato.ac.nz/ml/weka/`

Table 1. Results for the complete taxonomies with PALEO_10_10 and PALEO_5_5. *Columns:* "size" is the number of nodes of the discovered antichain; "calls" is the number of calls to the linear regressor in the model construction phase; "corr" corresponds to the the correlation coefficient of the linear regression when using the corresponding antichain (in the test data); "RMSE" reports the root mean squared error of the linear regression when using the corresponding antichain (in the test data). *Rows:* FAMILY, GENUS and SPECIES select a fix level of the taxonomy as the antichain. `Random` reports the mean values over 100 antichains.

	PALEO_10_10				PALEO_5_5			
	size	calls	corr	RMSE	size	calls	corr	RMSE
FAMILY	49	-	0.81	3.40	63	-	0.76	3.68
GENUS	237	-	0.20	16.4	373	-	0.60	6.53
SPECIES	428	-	0.88	2.62	867	-	0.65	6.04
`Greedy`	246	46534	0.67	4.89	464	140488	0.33	12.89
`Top-down`	145	1817	0.82	3.09	197	2619	0.79	3.91
`Bottom-up`	375	3737	0.86	2.65	794	10409	0.75	4.91
`Tax Min-cut`	201	726	0.51	6.62	306	1329	0.77	4.55
`Random`	325	-	0.83	3.04	615	-	0.1	150

closest to this species level. As we turn data into a sparser format with PALEO_5_5, we observe a clear gain given by `Tax Min-cut`, which still it does not beat `Top-down` but seems to gain learning power as the data goes sparser. Indeed, we observed that `Tax Min-cut` typically refines the best of the three fixed antichains (i.e., it upgrades the best of the three), typically this is the antichain at the GENUS level, and so, it can adapt better to sparser formats. Random antichains perform in average surprisingly well on PALEO_10_10; as the data turns sparser, random antichains are not able to predict as good as our proposed algorithms.

Finally we also ran the same experiments with PALEO_5_5 and PALEO_2_2 using different subtrees of the taxonomy. These correspond to taxonomies of two orders of mammals; the *Carnivora* and *Rodentia*. Comparison with random antichains can be seen in Figure 3. Here the vertical line indicates the correlation obtained by the `Tax Min-cut` algorithm, which we expect to perform well on

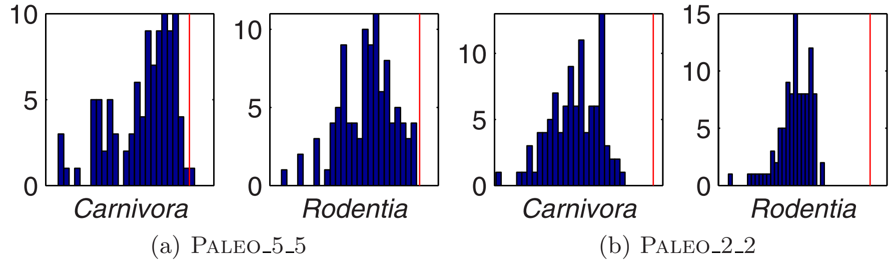


Fig. 3. Histograms of the correlation between predicted and real age from 100 random antichains in the *Carnivora* and *Rodentia* subtrees of the taxonomy using PALEO_5_5 and PALEO_2_2. The vertical line indicates the performance of the solution given by the `Tax Min-cut` algorithm in each case.

Table 2. Results for the *Carnivora* and *Rodentia* subtrees of the taxonomy with PALEO_5_5 and PALEO_2_2. `Random` reports the mean values over 100 antichains.

	PALEO_5_5								PALEO_2_2							
	Carnivora				*Rodentia*				*Carnivora*				*Rodentia*			
	size	calls	corr	RMSE	size	calls	corr	RMSE	size	calls	corr	RMSE	size	calls	corr	RMSE
FAMILY	12	-	0.47	4.90	10	-	0.52	4.74	17	-	0.41	5.45	8	-	0.44	5.37
GENUS	61	-	0.48	4.86	121	-	0.79	3.45	140	-	0.36	5.87	75	-	0.63	4.74
SPECIES	112	-	0.19	6.74	344	-	0.39	12.5	347	-	0.20	8.21	215	-	0.12	23.9
`Greedy`	67	3228	0.45	5.42	163	14017	0.73	4.40	139	9501	0.31	6.11	294	36640	0.60	5.38
`Top-down`	25	99	0.46	5.34	151	1176	0.74	4.20	75	258	0.40	5.58	190	2338	0.69	4.48
`Bottom-up`	48	578	0.45	5.38	232	2050	0.65	5.45	221	2073	0.35	5.89	500	4222	0.27	11.9
`Tax Min-cut`	33	188	0.47	5.30	106	476	0.78	3.77	96	460	0.42	5.55	234	861	0.67	4.67
`Random`	84	-	0.41	5.16	231	-	0.67	4.74	222	-	0.24	7.36	178	-	0.52	5.34

sparse inputs. With PALEO_5_5 we observe that `Random` can give results that are as good (or maybe even better) than those obtained with `Tax Min-cut`. However, in case of PALEO_2_2 the difference is more pronounced and `Tax Min-cut` gives consistently better results than simply selecting a random antichain. This is also the case with `Bottom-up`. The results reported in Table 2 show the same conclusion as above: `Tax Min-cut` and `Bottom-up` are able to generalize much better than other algorithms and select typically a better refinement of the GENUS level. From the computational perspective we should highlight that `Tax Min-cut` tends to run faster than the other algorithms, which typically call the linear regressor more times than the number of nodes in the taxonomy.

5 Related Work

The problem of selecting relevant features is a recurring theme in machine learning [2,4,10,11]. Typical strategies rely on a heuristic search over the combinatorial space of all features, e.g.: backward elimination, forward selection, the filter approach, and the wrapper approach. The algorithms presented in this paper are inspired by some of these techniques: `Greedy` can be seen as a taxonomic forward selection algorithm; `Top-down` and `Bottom-up` have the flavour of backward elimination approaches that take into account the space of the taxonomy; finally, the `Taxonomy Min-Cut` solution can be seen as a filter approach.

There is also relevant work in machine learning on the problem of learning classifiers from attribute valued taxonomies [5,15]. The work in [5] focuses on Bayesian Networks. The approaches in [15] focus on naïve Bayes classifiers and decision trees. Features are typically selected in a top-down fashion through the hypothesis space given by the taxonomy; also, the problem is studied specifically for the two mentioned learning algorithms. Our proposals complement those approaches by presenting feature selection in taxonomies as a general problem, independent of the learning algorithm one wishes to use. Other different scenarios for taxonomies occur when learning classifiers with class labels exhibiting a predefined class hierarchy, e.g. [3]. Finally, taxonomies have been the target of data mining as well: e.g. in association rules [13] or clustering [14].

6 Conclusions

We have considered the feature selection problem in taxonomies. Our motivating application is the problem of determining the age of fossil sites on the basis of the taxa found in the sites. We formulated the problem of finding the best selection of features from a hierarchy, showed that it is NP-complete, and gave four algorithms for the task. Of the algorithms, `Greedy`, `Bottom-up` and `Top-down` are inspired by previous feature selection approaches where taxonomies where not yet considered; `Tax Min-cut` uses the well-known min-cut max-flow theorem to provide with a quick and rather good choice of an antichain.

The empirical results show that from the proposed methods, especially `Tax Min-cut` works well; it performs at least as good as the best antichains that are based on a fixed level of the taxonomy. Future work involves applying the method to some interesting subsets of the paleontological sites, investigation of other applications, and further study of the properties of the algorithms.

References

1. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network flows: theory, algorithms, and applications. Prentice-Hall, Inc., Englewood Cliffs (1993)
2. Blum, A.L., Langley, P.: Selection of relevant features and examples in machine learning. Artif. Intell. 97(1-2), 245–271 (1997)
3. Cai, L., Hofmann, T.: Exploiting known taxonomies in learning overlapping concepts. In: IJCAI 2007, pp. 714–719 (2007)
4. Charikar, M., Guruswami, V., Kumar, R., Rajagopalan, S., Sahai, A.: Combinatorial feature selection problems. In: FOCS 2000, page 631 (2000)
5. desJardins, M., Getoor, L., Koller, D.: Using feature hierarchies in Bayesian network learning. In: Choueiry, B.Y., Walsh, T. (eds.) SARA 2000. LNCS (LNAI), vol. 1864, pp. 260–270. Springer, Heidelberg (2000)
6. Ford, L.R., Fulkerson, D.R.: Maximal flow through a network. Canadian Journal of Mathematics 8, 399–404 (1956)
7. Fortelius, M., Gionis, A., Jernvall, J., Mannila, H.: Spectral ordering and biochronology of european fossil mammals. Paleobiology 32, 206–214 (2006)
8. Fortelius, M.: Neogene of the old world database of fossil mammals (NOW) (2008), http://www.helsinki.fi/science/now/
9. Jernvall, J., Fortelius, M.: Common mammals drive the evolutionary increase of hypsodonty in the neogene. Nature 417, 538–540 (2002)
10. Kohavi, R., John, G.H.: Wrappers for feature subset selection. Artif. Intell. 97(1-2), 273–324 (1997)
11. Lavrač, N., Gamberger, D.: Relevancy in constraint-based subgroup discovery. In: Constraint-Based Mining and Inductive Databases, pp. 243–266 (2004)
12. Liow, L.H., Fortelius, M., Bingham, E., Lintulaakso, K., Mannila, H., Flynn, L., Stenseth, N.C.: Stenseth higher origination and extinction rates in larger mammals. PNAS 105, 6097–6102 (2008)
13. Srikant, R., Agrawal, R.: Mining generalized association rules. Future Gener. Comput. Syst. 13(2-3), 161–180 (1997)
14. Yun, C., Chuang, K., Chen, M.: Using category-based adherence to cluster market-basket data. In: ICDM 2002, p. 546 (2002)
15. Zhang, J., Kang, D.-K., Silvescu, A., Honavar, V.: Learning accurate and concise naïve bayes classifiers from attribute value taxonomies and data. Knowl. Inf. Syst. 9(2), 157–179 (2006)

Deduction Schemes for Association Rules

José L. Balcázar

Departament de Llenguatges i Sistemes Informàtics
Laboratori d'Algorísmica Relacional, Complexitat i Aprenentatge
Universitat Politècnica de Catalunya, Barcelona
balqui@lsi.upc.edu

Abstract. Several notions of redundancy exist for Association Rules. Often, these notions take the form "any dataset in which this first rule holds must obey also that second rule, therefore the second is redundant"; if we see datasets as interpretations (or models) in the logical sense, this is a form of logical entailment. In many logics, entailment has a syntactic counterpart in the form of a deduction calculus. We provide such a deduction calculus for existing notions of redundancy; then, we consider a very general notion of entailment, where a confidence threshold is fixed and several rules can act as simultaneous premises, and identify exactly the cases where a partial rule follows from two partial rules; we also give a deduction calculus for this setting.

Keywords: Association rules, redundancy, deductive calculus.

1 Motivation and Related Work

Data mining involves a wide spectrum of techniques; among them, Association Rule Mining is a prominent conceptual tool and, possibly, a cornerstone notion of the field, if there is one. Indeed, association rules are both among the most widely studied topic in data mining research and among the most widely employed data mining techniques in actual applications. Practitioners have reported impressive success stories (failures being less prone to receive publicity); researchers have provided a wealth of algorithms to compute diverse variants of association rules on datasets of diverse characteristics, and there are many extensions into similar notions for complex data. The volume of literature about the topic is daunting. A recent survey is [5] but additional materials appear in `http://wwwai.wu-wien.ac.at/~hahsler/research/association_rules/`, for instance, at the time of writing.

Implications, that is, association rules that hold in 100% of the cases, had been studied before in the research area of closure spaces (see, for instance, [9] and [19]). Implications can be seen also as conjunctions of definite Horn clauses, and the closure under intersection property that characterizes closures spaces corresponds to the fact, well-known in logic and knowledge representation, that Horn theories are exactly those closed under bitwise intersection of propositional models (see e.g. [11]). Thus, as a form of knowledge gathered from a dataset, implications have several advantages: explicit or implicit correspondence with Horn logic, therefore a tight parallel with functional dependencies and a clear, hardly disputable notion of redundancy that can be defined equivalently both in semantic terms and through a syntactic calculus. Specifically, in semantic terms, an implication $X \to Y$ is entailed from a set of implications $\mathcal{R}$ if every dataset in which all the implications of $\mathcal{R}$ hold must also satisfy $X \to Y$; and, syntactically, it is known that this happens if and only if $X \to Y$ is derivable from $\mathcal{R}$ via the Armstrong axiom schemes [17], namely, Reflexivity ($X \to X$), Augmentation (if $X \to Y$ and $X' \to Y'$ then $XX' \to YY'$, where juxtaposition denotes union) and Transitivity (if $X \to Y$ and $Y \to Z$ then $X \to Z$).

However, absolute implication analysis may become too limited for many application tasks. Already in [15] we find a proposal to consider partial rules, defined in relation to their so-called-there "precision", that is, the notion of intensity of implication now widely called "confidence": for a given rule $X \to Y$, the ratio of how often X and Y are seen together to how often X is seen.

Such search for implications or for partial rules was not used on really large datasets until the introduction of the notion of support bound, that is, an absolute threshold on how often the itemsets under analysis appear in the dataset. The idea of restricting the exploration for association rules to frequent itemsets, with respect to a support threshold, gave rise to the most widely discussed and applied algorithm, Apriori [2], and to an intense research activity.

A well-known difficulty in applied association rule mining lies in that, on large datasets, and for sensible settings of the confidence and support thresholds, huge amounts of association rules are often obtained, much beyond what any user of the data mining process may be expected to look at. Therefore, one research topic that has been worthy of attention is the identification of patterns that indicate redundancy of rules, and ways to avoid that redundancy [1], [6], [12], [13], [15], [16], [18]; see also section 6 of [5] and the references therein. All these definitions of redundancy are given either set-theoretically, in the sense of inclusions among sets of attributes, or in a more general way, by resorting to a confidence inequality: a rule is redundant with respect to another if it has at least the same confidence of the latter *for every dataset*.

By analogy to the case of implications, it is natural to raise the question of whether a deductive calculus for these different notions of redundancy among partial rules can be designed. (We will keep this terminology throughout the paper: implications are association rules of confidence 1, whereas partial rules are those having a confidence below 1.) The Armstrong axiom schemes that play such a role for implications are, in fact, no longer adequate: Reflexivity does hold for partial association rules, but Augmentation does not hold at all, whereas Transitivity takes a different form that affects the confidence of the rules: if the rule $A \to B$ (or $A \to AB$, which is equivalent) and the rule $B \to C$ both hold with confidence γ, we still know nothing about the confidence of $A \to C$; even the fact that both $A \to AB$ and $AB \to C$ hold with confidence γ only gives us a confidence of γ^2 for $A \to C$. Regarding Augmentation, enlarging the antecedent of a rule of confidence γ may give a rule with much smaller confidence, even zero:

think of a case where most of the times X appears it comes with Y, but it only comes with Z when Y is not present; then the confidence of $X \to Y$ may be high whereas the confidence of $XZ \to Y$ may be null. Similarly, a rule with several items in the consequent is *not* equivalent to the conjunction of the Horn-style rules with the same antecedent and each item of the consequent separately: if we look only into rules with singletons as consequents (as in the "basic association rules" of [14] or in the useful `apriori` implementation of Borgelt available on the web [4]) we are almost certain to lose information. Indeed, if the confidence of $X \to YZ$ is high, it means that Y and Z appear together in most of the transactions having X; but, with respect to the converse, the fact that both Y and Z appear in fractions at least γ of the transactions having X does not inform us that they show up *together* at a similar ratio of these transactions: only a ratio of $2\gamma - 1 < \gamma$ is guaranteed to hold.

We have provided in a recent contribution [3] an analysis of several particular notions of redundancy in the literature; here, we progress further along that study. First, we provide a deductive calculus that characterizes these notions of redundancy. In a way similar to the Armstrong axiom schemes, we give deduction schemes such that exactly the rules that are redundant can be deduced through these schemes. Then, we depart from most proposed notions of redundancy by considering the possibility that a rule is redundant with respect to a set of rules, instead of a single one; for that case we have our main contribution here. A first consideration is that we no longer have a single value of the confidence to compare; therefore, we take a position like the one in most cases of applications of association rule mining in practice, namely: fix a confidence threshold, and consider only rules whose confidence is above it; alternatively, an equivalent view would be that the confidence of all our conclusions should be at least the same as the minimum of the confidences of the premises. For instance, with items A, B, C, and D, assume that the confidence of the rules $A \to BC$ and $A \to BD$ is above γ in a dataset $\mathcal{D}$. What can be said, then, about the confidence of the rule $ACD \to B$ in $\mathcal{D}$? For instance, could one construct a dataset where the rules $A \to BC$ and $A \to BD$ hold with 65% confidence and, simultaneously, rule $ACD \to B$ falls below the same confidence threshold? The answer is counterintuitive: it is, in fact, impossible, and we will provide a full answer, characterizing exactly inference from two partial rules, as main result of this paper, and a corresponding deduction scheme extending our calculus. In fact, the existing notions of redundancy correspond exactly to entailment among association rules just for a specific confidence interval, and our results suggest the possibility of a pattern, where further values of the threshold would correspond, successively, to the ability of using three partial premises, four, and so on. However, to attain such a result, further efforts are still necessary.

2 Preliminaries

A dataset $\mathcal{D}$ is given; it consists of transactions, each of which is an itemset labeled by a unique transaction identifier. The identifiers allow for many transactions sharing the same itemset. Upper-case, often subscripted letters from the end of the alphabet, like X_1 or Y_0, denote itemsets. Juxtaposition denotes union of itemsets, as in XY; and $Z \subset X$ denotes proper subsets. For a transaction t, we denote $t \models X$ the fact that X is a subset of the itemset corresponding to t.

From the given dataset we obtain a notion of support of an itemset: $s_{\mathcal{D}}(X)$ is the cardinality of the set of transactions that include it, $\{t \in \mathcal{D} \mid t \models X\}$; sometimes, abusing language, we also refer to that set of transactions itself as support. Whenever $\mathcal{D}$ is clear, we drop the subindex: $s(X)$.

We immediately obtain by standard means (see, for instance, [9] or [18]) a notion of closed itemsets, namely, those that cannot be enlarged while maintaining the same support. The function that maps each itemset to the smallest closed set that contains it is known to be monotonic, extensive, and idempotent, that is, a closure operator. This notion will be reviewed in more detail later on.

Association rules are pairs of itemsets, denoted as $X \to Y$ for itemsets X and Y. Intuitively, they express that Y occurs particularly often among the transactions in which X occurs. More precisely, the confidence $c_{\mathcal{D}}(X \to Y)$ of an association rule $X \to Y$ in a dataset $\mathcal{D}$ is $\frac{s(XY)}{s(X)}$, that is, the ratio by which transactions having X have also Y; or, again, the observed empirical approximation to a conditional probability of Y given X. As with support, often we drop the subindex $\mathcal{D}$. This view suggests a form of correlation that, in many applications, is interpreted implicitly as a form of causality (which, however, is not guaranteed in any formal way; see the interesting discussion in [8]).

We resort to the convention that, if $s(X) = 0$ (which implies $s(XY) = 0$) we redefine the undefined confidence as 1, since the intuitive expression "all transactions having X do have also Y" becomes vacuously true. Also, it is immediate to check that $c_{\mathcal{D}}(X \to Y) = c_{\mathcal{D}}(X \to XY) = c_{\mathcal{D}}(X \to X'Y)$ for any subset $X' \subseteq X$. When two rules have the same left hand side, and the same union of left and right hand sides, we say that they are *equivalent by reflexivity*. Clearly their supports and confidences will always coincide.

We discuss briefly now the following natural notion of redundancy:

Definition 1. *$X_0 \to Y_0$ is plainly redundant with respect to $X_1 \to Y_1$ if the confidence of $X_0 \to Y_0$ is larger than or equal to the confidence of $X_1 \to Y_1$, whatever the dataset.*

That is: in that case, if a data mining process with confidence threshold γ provides both rules as output, $X_0 \to Y_0$ and $X_1 \to Y_1$, then rule $X_0 \to Y_0$ is uniformative, and can be ignored, because the fact that its confidence is at least γ is already guaranteed without computing it. Of course, such a redundancy will happen only if the various itemsets involved, such as X_0 and X_1, have some correlation. Several cases of redundancy have been identified already in the literature, and compared in [3]. We briefly review some related results.

Definition 2. *1.* [1] *If $Z_0 \neq \emptyset$, rule $X_0Z_0 \to Y_0$ is simply redundant with respect to $X_0 \to Y_0Z_0$.*

2. [1] *If $X_1 \subseteq X_0$ and $X_0Y_0 \subset X_1Y_1$, rule $X_0 \to Y_0$ is strictly redundant with respect to $X_1 \to Y_1$.*

3. [12] *Rule $X_1 \to Y_1$ covers rule $X_0 \to Y_0$ if $X_1 \subseteq X_0$ and $X_0Y_0 \subseteq X_1Y_1$.*

The original definition of cover in [12] is different but the same reference proves the equivalence with the formulation we are using here.

In these three cases, it is not difficult to see that the confidence and support of $X_0 \to Y_0$ is at least that of $X_1 \to Y_1$ ([1], [12]). Note that, in principle, there could possibly be many other ways of a rule being redundant with respect to another beyond covering, simple, and strict redundancies. Simple redundancy relates rules obtained from the same (frequent) set $X_0Y_0Z_0$. Strict redundancy focuses, instead, on rules extracted from two different (frequent) itemsets, say X_0Y_0 where X_0 will be considered as antecedent, and X_1Y_1, where X_1 will be antecedent, and under the conditions that $X_1 \subseteq X_0$ and $X_0Y_0 \subset X_1Y_1$ (the case $X_0Y_0 = X_1Y_1$ is already covered by simple redundancy). Our previous work has contributed the following result:

Theorem 1. [3] *Consider any two rules $X_0 \to Y_0$ and $X_1 \to Y_1$ where $Y_0 \not\subseteq X_0$. The following are equivalent:*

1. *$X_1 \to Y_1$ covers $X_0 \to Y_0$;*
2. *rule $X_0 \to Y_0$ is either simply redundant or strictly redundant with respect to $X_1 \to Y_1$, or they are equivalent by reflexivity;*
3. *both the confidence* **and** *the support of $X_0 \to Y_0$ are larger than or equal to those of $X_1 \to Y_1$, whatever the dataset;*
4. *rule $X_0 \to Y_0$ is plainly redundant with respect to $X_1 \to Y_1$.*

That is, the additional consideration of the support bound in formulation 3 (due to [1] as well) does not make the notion more restrictive, whereas the notion of covering catches all possible nontrivial situations of plain redundancy. The equivalence of the first two statements is immediate. Note that rules with $Y_0 \subseteq X_0$ have confidence 1: they state just reflexivity, and are uninformative.

A major application of the notion of redundancy is the construction of "bases": sets of rules that make redundant all the remaining rules mined. Our previous paper [3] has shown that the existing techniques for constructing bases with respect to these equivalent notions of redundancy do attain the minimum possible size of a basis. However, further reduction of the bases is still desirable, and the only way to obtain it is through some stronger notion of redundancy.

One such stronger notion existing in the literature, which indeed allows for smaller bases (most of the times) relies on handling separately implications from partial rules. Indeed, implications can be summarized better, because they allow for Transitivity and Augmentation to apply in order to find redundancies; moreover, they can be combined in a certain form of transitivity with a partial rule of confidence, say, γ to give rules of confidence at least γ. The best way to handle them is through a closure operator ([7], [9], [16], [18]). Specifically, given a dataset $\mathcal{D}$, the closure operator associated to $\mathcal{D}$ maps each itemset X to the largest itemset $\overline{X} \supseteq X$ that has the same support as X in $\mathcal{D}$; it can be defined in several alternative ways. A set is closed if it coincides with its closure. When

$\overline{X} = Y$ we also say that X is a generator of Y. Our definition gives directly that always $s(X) = s(\overline{X})$. We will make liberal use of this fact, which is easy to check also with other definitions of the closure operator, as stated in [16], [18], and others. Implications are intimately related to this closure operator: $c(X \to Y) = 1$ if and only if $Y \subseteq \overline{X}$. Several quite good algorithms exist to find the closed sets and their supports. In the literature, there are proposals of basis constructions out of closed sets with respect to a notion of redundancy based on closures, a natural generalization of equivalence by reflexivity that works as follows ([18], see also section 4 in [16]): given a dataset and a closure operator corresponding to implications that have confidence 1 in the dataset, two partial rules $X_0 \to Y_0$ and $X_1 \to Y_1$ such that $\overline{X_0} = \overline{X_1}$ and $\overline{X_0Y_0} = \overline{X_1Y_1}$ turn out to be equivalent in terms of support and confidence; the reason is that $s(X_0) = s(\overline{X_0}) = s(\overline{X_1}) = s(X_1)$, and $s(X_0Y_0) = s(\overline{X_0Y_0}) = s(\overline{X_1Y_1}) = s(X_1Y_1)$, by the property stated above that always $s(X) = s(\overline{X})$.

Let $\mathcal{B}$ be the set of implications, of confidence 1, in the dataset $\mathcal{D}$; alternatively, $\mathcal{B}$ can be the basis already known for implications in a dataset [7]. From here on, we require $0 < \gamma < 1$, leaving the rules of confidence 1 to be handled from $\mathcal{B}$. Our previous work has contributed the following property, along the same lines as Theorem 1:

Theorem 2. [3] *Let $\mathcal{B}$ be a set of implications. Let $X_2 \to Y_2$ be a rule not implied by $\mathcal{B}$, that is, where $Y_2 \not\subseteq \overline{X_2}$. Then, the following are equivalent:*

1. *$X_1 \subseteq \overline{X_2}$ and $X_2Y_2 \subseteq \overline{X_1Y_1}$*
2. *Every dataset $\mathcal{D}$ in which all the rules in $\mathcal{B}$ hold with confidence 1 gives $c_{\mathcal{D}}(X_2 \to Y_2) \geq c_{\mathcal{D}}(X_1 \to Y_1)$.*

In either case we say that rule $X_2 \to Y_2$ has closure-based redundancy relative to $\mathcal{B}$ with respect to rule $X_1 \to Y_1$.

3 Deduction Schemes for Redundancy

Redundancy is, in principle, more restrictive than entailment: so far, in the literature, redundancy has been taken mostly to signify a relationship between two association rules, as just described. We start our discussion of deduction systems along the same lines.

We give now a calculus consisting of three inference schemes: right-hand Reduction (rR), where the consequent is diminished; right-hand Augmentation (rA), where the consequent is enlarged; and left-hand Augmentation (ℓA), where the antecedent is enlarged. As customary in logic calculi, our rendering of each rule means that, if the facts above the line are already derived, we can immediately derive the fact below the line.

$$(rR) \quad \frac{X \to Y, \qquad Z \subseteq Y}{X \to Z}$$

$$(rA) \quad \frac{X \to Y}{X \to XY}$$

$$(\ell A) \quad \frac{X \to YZ}{XY \to Z}$$

We also allow always to state trivial rules: $\overline{X \to \emptyset}$, which, combined with (rA) and (rR), allows us to infer $X \to Y$ whenever $Y \subseteq X$.

Scheme (ℓA) is exactly simple redundancy from Definition 2. The Reduction Scheme (rR) allows us to "lose" information and find inequivalent rules (whose confidence may be larger).

It is not difficult to see that the calculus is sound, that is, for every dataset, the confidence of the rule below the line in any one of the three deduction schemes is, at least, the same as the confidence of the rule above the line. For the scheme (ℓA), this claim is Theorem 4.1 in [1]: correctness of simple redundancy. Likewise, soundness for (rR) it is the next result in [1], Theorem 4.2. The soundness of (rA) follows from equivalence by reflexivity. Also, trivial rules with empty right hand side are always sound.

Of course, this extends to chained applications of these schemes. In fact, if we start with a rule $X_1 \to Y_1$, and keep applying these three inference schemes to obtain new association rules, the rules we obtain are all plainly redundant with respect to $X_1 \to Y_1$.

The interesting property of these schemes, and our first contribution, is that the converse also holds; that is: whenever two rules are related by plain redundancy, it is always possible to prove it using just those inference schemes.

Theorem 3. *Rule $X_0 \to Y_0$ is plainly redundant with respect to rule $X_1 \to Y_1$ if and only if $X_0 \to Y_0$ can be derived from $X_1 \to Y_1$ by repeated application of the inference schemes (rR), (rA), and (ℓA).*

Proof. That all rules derived are plainly redundant has just been argued above. For the converse, assume that rule $X_0 \to Y_0$ is plainly redundant with respect to rule $X_1 \to Y_1$. By Theorem 1, we know that this implies that $X_1 \to Y_1$ covers $X_0 \to Y_0$, that is, by Definition 2, $X_1 \subseteq X_0$ and $X_0Y_0 \subseteq X_1Y_1$. Now, to infer $X_0 \to Y_0$ from $X_1 \to Y_1$, we chain up applications of our schemes as follows:

$$X_1 \to Y_1 \vdash_{(rA)} X_1 \to X_1Y_1 \vdash_{(rR)} X_1 \to X_0Y_0 \vdash_{(\ell A)} X_0 \to Y_0$$

where the second step makes use of the inclusion $X_0Y_0 \subseteq X_1Y_1$, and the last step makes use of the inclusion $X_1 \subseteq X_0$. Here, the standard derivation symbol $\vdash$ denotes derivability by application of the rule indicated as a subscript. □

3.1 Calculus for Closure-Based Redundancy

The calculus just given is not appropriate to handle closure-based redundancy, because it does not contemplate any form of Transitivity. The stronger calculus we provide now is sound and complete with respect to closure-based redundancy. We will use two different symbols for rules: we will keep $X_0 \to Y_0$ to denote association rules, of which we will lower-bound the confidence, and we will use the notation $X_0 \Rightarrow Y_0$ to denote implications. Our calculus for closure-based redundancy consists of four inference schemes, each of which reaches a partial rule from premises including a partial rule. Two of the schemes correspond to variants of Augmentation, one for enlarging the antecedent, the other for enlarging the consequent. The other two correspond to composition with an implication, one in the antecedent and one in the consequent: a form of controlled transitivity. Their names (rA), (ℓA), (rI), and (ℓI) indicate whether they operate at the right or left hand side and whether their effect is Augmentation or composition with an Implication.

$$(rA) \quad \frac{X \to Y, \qquad X \Rightarrow Z}{X \to YZ}$$

$$(rI) \quad \frac{X \to Y, \qquad Y \Rightarrow Z}{X \to Z}$$

$$(\ell A) \quad \frac{X \to YZ}{XY \to Z}$$

$$(\ell I) \quad \frac{X \to Y, \qquad Z \subseteq X, \qquad Z \Rightarrow X}{Z \to Y}$$

Again we allow as well to state directly rules with empty right hand side: $\frac{}{X \to \emptyset}$. Note that this opens the door to using (rA) with an empty Y, and this allows us to transform an implication into the corresponding partial rule. Also, (ℓA) could be stated equivalently with $XY \to YZ$ below the line, by (rA).

The connection with the previous, simpler calculus should be easy to understand: first, observe that the (ℓA) rules are identical. Now, if implications are not considered separately, the only cases where we know that $X_1 \Rightarrow Y_1$ are those where $Y_1 \subseteq X_1$; we see that (rI) corresponds, in that case, to (rR), whereas the (rA) schemes only differ on cases of equivalence by reflexivity. Finally, (ℓI) becomes fully trivial since $X \subseteq Z$ makes $X = Z$, and the partial rules above and below the line would coincide.

In the remaining of this section, we denote as $\mathcal{B} \cup \{X \to Y\} \vdash X' \to Y'$ the fact that, in the presence of the implications in the set $\mathcal{B}$, rule $X' \to Y'$ can be derived from rule $X \to Y$ using zero or more applications of the four deduction schemes.

We can characterize the deductive power of this calculus as follows: it is sound and complete with respect to the notion of closure-based redundancy; that is, all the rules it can prove are redundant, and all the redundant rules can be proved:

Theorem 4. *Let $\mathcal{B}$ consist of implications. Then, $\mathcal{B} \cup \{X_1 \to Y_1\} \vdash X_2 \to Y_2$ if and only if rule $X_2 \to Y_2$ has closure-based redundancy relative to $\mathcal{B}$ with respect to rule $X_1 \to Y_1$.*

Proof (Sketch). Proofs given (within a slightly different framework) in [18] provide directly the soundness of (rI) and the soundness of a combination of (rA) with (ℓI) that can be extended quite easily to obtain soundness of our four schemes. To prove completeness, we must see that all redundant rules can be derived. We assume the closure-based redundancy of $X_2 \to Y_2$ and resort to Theorem 2: we know that the inclusions $X_1 \subseteq \overline{X_2}$ and $X_2Y_2 \subseteq \overline{X_1Y_1}$ must hold. From the second inclusion, and the properties of the closure operator, we have that $\overline{X_2}Y_2 \subseteq \overline{X_1Y_1}$.

Now we can write a derivation in our calculus, taking into account these inclusions, as follows:
$X_1 \to Y_1 \vdash_{(rA)} X_1 \to X_1Y_1 \vdash_{(rI)} X_1 \to \overline{X_2}Y_2 \vdash_{(\ell A)} \overline{X_2} \to Y_2 \vdash_{(\ell I)} X_2 \to Y_2$

Thus, indeed the redundant rule is derivable, which proves completeness. □

4 Closure-Based Entailment

We move on towards the main contribution of this paper. The following question naturally arises: all these notions of redundancy only relate one partial rule to another partial rule, possibly in presence of implications. Is it indeed possible that a partial rule is entailed jointly by two partial rules, but not by a single one of them? Along this section, we are after a calculus for all partial rules whose confidence is above some fixed but arbitrary threshold γ. We will fully answer this question by, first, characterizing precisely the case where a partial rule follows from exactly two partial rules, the simplest case where our previous calculus becomes incomplete; and, second, proving that a sound and complete calculus can be constructed by adding one extra rule that allows us to conclude a consequent partial rule from two antecedent partial rules. We present the whole setting on top of closure-based redundancy, but an analogous development can be made on top of plain redundancy. We consider the following definition:

Definition 3. *Given a set $\mathcal{B}$ of implications, and a set $\mathcal{R}$ of partial rules, rule $X_0 \to Y_0$ is γ-redundant with respect to them, $\mathcal{B} \cup \mathcal{R} \models_\gamma X_0 \to Y_0$, if every dataset in which the rules of $\mathcal{B}$ have confidence 1 and the confidence of all the rules in $\mathcal{R}$ is at least γ must satisfy as well $X_0 \to Y_0$ with confidence at least γ.*

As an interesting example that does not need the presence of implications, consider the following fact, mentioned in the Introduction (the analogous statement for $\gamma < 1/2$ does not hold, as discussed below):

Proposition 1. *Let $\gamma \geq 1/2$. Assume that items A, B, C, D are present in $\mathcal{U}$ and that the confidence of the rules $A \to BC$ and $A \to BD$ is above γ in dataset $\mathcal{D}$. Then, the confidence of the rule $ACD \to B$ in $\mathcal{D}$ is also above γ.*

We omit the proof, since it is just the simplest particular case of our main result in the paper, which clarifies exactly these situations, and reads as follows:

Theorem 5. *Let $\mathcal{B}$ be a set of implications, and let $1/2 \leq \gamma < 1$. Then, $\mathcal{B} \cup \{X_1 \to Y_1, X_2 \to Y_2\} \models_\gamma X_0 \to Y_0$ if and only if either:*

1. $Y_0 \subseteq \overline{X_0}$, *or*
2. $\mathcal{B} \cup \{X_1 \to Y_1\} \models_\gamma X_0 \to Y_0$, *or*
3. $\mathcal{B} \cup \{X_2 \to Y_2\} \models_\gamma X_0 \to Y_0$, *or*
4. *all the following conditions simultaneously hold:*
   - (i) $X_1 \subseteq \overline{X_0}$
   - (ii) $X_2 \subseteq \overline{X_0}$
   - (iii) $X_1 \subseteq \overline{X_2 Y_2}$
   - (iv) $X_2 \subseteq \overline{X_1 Y_1}$
   - (v) $X_0 \subseteq \overline{X_1 Y_1 X_2 Y_2}$
   - (vi) $Y_0 \subseteq \overline{X_0 Y_1}$
   - (vii) $Y_0 \subseteq \overline{X_0 Y_2}$

Proof (Sketch). Let us discuss first the leftwards implication. In case (1) the consequent rule holds trivially. Clearly cases (2) and (3) also give the entailment, though in a somehow "improper" way. For case (4), we must argue that, if all the seven conditions hold, then the entailment happens. Thus, fix any dataset $\mathcal{D}$ where the confidences of the antecedent rules are at least γ: these assumptions can be written, respectively, $s(X_1Y_1) \geq \gamma s(X_1)$ and $s(X_2Y_2) \geq \gamma s(X_2)$, or equivalently for the corresponding closures.

We have to show that the confidence of $X_0 \to Y_0$ in $\mathcal{D}$ is also at least γ. Consider the following four sets of transactions from $\mathcal{D}$:

$A = \{t \in \mathcal{D} \mid t \models X_0Y_0\}$
$B = \{t \in \mathcal{D} \mid t \models X_0, t \not\models X_0Y_0\}$
$C = \{t \in \mathcal{D} \mid t \models X_1Y_1, t \not\models X_0\}$
$D = \{t \in \mathcal{D} \mid t \models X_2Y_2, t \not\models X_0\}$

and let a, b, c, and d be the respective cardinalities. Using condition (v), it can be seen that all four sets are mutually disjoint. Now we bound the supports of the involved itemsets as follows: clearly, by definition of A, $s(X_0Y_0) = a$. All tuples that satisfy X_0 are accounted for either as satisfying Y_0 as well, in A, or in B in case they don't; disjointness then guarantees that $s(X_0) = a + b$.

We see also that $s(X_1) \geq a + b + c + d$, because X_1 is satisfied by the tuples in C, by definition; by the tuples in A or B, by condition (i); and by the tuples in D, by condition (iii); again disjointness allows us to sum all four cardinalities. Similarly, using instead (ii) and (iv), we obtain $s(X_2) \geq a + b + c + d$.

If we split all the tuples that satisfy X_1Y_1 into two sets, those that additionally satisfy X_0, and those that don't, using conditions (i) and (vi) it can be argued that $s(X_1Y_1) \leq a + c$ and, symmetrically, resorting to (ii) and (vii), $s(X_2Y_2) \leq a + d$.

Thus we can write the following inequations:

$$a + c \geq s(X_1Y_1) \geq \gamma s(X_1) \geq \gamma(a + b + c + d)$$
$$a + d \geq s(X_2Y_2) \geq \gamma s(X_2) \geq \gamma(a + b + c + d)$$

Adding them up, using $\gamma \geq \frac{1}{2}$, and simplifying, we get $a \geq \gamma(a + b)$, so that $c(X_0 \to Y_0) = s(X_0Y_0)/s(X_0) = a/(a + b) \geq \gamma$ as was to be shown.

For the converse, we first point out that the bound $\gamma \geq \frac{1}{2}$ is not necessary for this part. The proof goes on by arguing the contrapositive, assuming that we are in neither of the four cases, and showing that the entailment does not happen, that is, it is possible to construct a counterexample dataset for which all the implications in $\mathcal{B}$ hold, and the two premise partial rules have confidence at least γ, whereas the rule in the conclusion has confidence strictly below γ. This requires us to construct a number of counterexamples through a somewhat long case analysis, omitted here for lack of space. □

4.1 Extending the Calculus

We work now towards a rule form. Let us say that an entailment is proper if the consequent follows from the given set of antecedents but does not follow from any proper subset thereof. We propose the following additional rule:

$$\frac{X_1 \rightarrow Y_1, \quad X_2 \rightarrow Y_2, \quad X_1 Y_1 \Rightarrow X_2, \quad X_2 Y_2 \Rightarrow X_1, \quad X_1 Y_1 X_2 Y_2 \Rightarrow Z}{X_1 X_2 Z \rightarrow \overline{X_1 Y_1 Z} \cap \overline{X_2 Y_2 Z}}$$

and state the following properties:

Theorem 6. *Given a threshold γ and a set $\mathcal{B}$ of implications,*

1. *this deduction scheme is sound, and*
2. *together with the deduction schemes in Subsection 3.1, it gives a calculus complete with respect to all entailments with two partial rules in the antecedent for $\gamma \geq 1/2$.*

Proof (sketch). This follows easily from Theorem 5, in that it implements the conditions of case (4); soundness is seen by directly checking that the conditions (i) to (vii) in case 4 of Theorem 5 hold. Completeness is argued by considering any rule $X_0 \rightarrow Y_0$ entailed by $X_1 \rightarrow Y_1$ and $X_2 \rightarrow Y_2$ jointly with respect to confidence threshold γ; if the entailment is improper, apply Theorem 4, otherwise just apply this new rule with $Z = \overline{X_0}$ to get $\overline{X_0} \rightarrow \overline{X_0 Y_1} \cap \overline{X_0 Y_2}$ and apply (ℓI) and (rI) to obtain $X_0 \rightarrow Y_0$. □

5 Conclusions

Our study here belongs to a larger program of research on the fundamentals of the Logic of Association Rules. We have described sound and complete variants of a deductive calculus for redundancy and entailment notions defined in terms of models, that is, datasets that assign a confidence value to each partial rule. The notions of redundancy correspond to already existing proposals, which discuss redundancy of a partial rule only with respect to another single partial rule; in our Theorem 5 we have moved beyond into the use of two partial rules. We believe this last step has been undertaken for the first time here since the early attempts of [15].

On the basis of our results as described here, it turns out that the following holds: for $0 < \gamma < 1/2$, there is γ-entailment if and only if either of cases 1, 2, or 3 in Theorem 5 is true; existing notions of redundancy are, therefore, fully appropriate for confidence below 1/2, but insufficient beyond. For larger confidence thresholds, our preliminary analyses are very suggestive of a general pattern, which we expect to be able to develop and apply to arbitrary values of γ: further values of the threshold correspond, successively, to the ability of using more and more partial premises. Namely, up to two partial rules can be used as premise, but not three, if $\gamma < 2/3$; up to three, but not four, if $\gamma < 3/4$; and so on. However, the combinatorics to fully characterize the case of two premises are already difficult enough for the current state of the art, and progressing along this line requires to build intuition to much further a degree. There remains to study also a comparison with other "semantic" redundancy schemes based on the actual values of the supports, such as those in [10] or, along a different track, [6].

Finally, we wish to discuss also the basis constructed in [3]: there we proved that the size of that basis is minimum with respect to closure-based redundancy.

It is worth to point out that our initial proofs of the results reported both here and there were much more involved, and it was thanks to the calculus described here that the minimum-size bases described in [3] were found, although further work has allowed us to simplify the analysis for expository purposes. It is also interesting to observe that, given the minimal basis described there, one can scan it to check the existence of pairs of rules that generate a third rule in the basis according to Theorem 5: then, removing these third rules gives a smaller basis with respect to general entailment. We expect to be able to establish a more general similar mechanism depending of the value of the threshold γ to reach absolutely minimum-size bases with respect to general entailment.

References

1. Aggarwal, C.C., Yu, P.S.: A new approach to online generation of association rules. IEEE Transactions on Knowledge and Data Engineering 13, 527–540 (2001)
2. Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., Verkamo, A.I.: Fast discovery of association rules. In: Fayyad, U., et al. (eds.) Advances in Knowledge Discovery and Data Mining, pp. 307–328. AAAI Press, Menlo Park
3. Balcázar, J.L.: Minimum-Size bases of Association Rules. In: PKDD, Antwerp (to appear, 2008)
4. Borgelt, C.: Efficient Implementations of Apriori and Eclat Workshop on Frequent Itemset Mining Implementations (2003)
5. Ceglar, A., Roddick, J.F.: Association mining. ACM Computing Surveys 38 (2006)
6. Cristofor, L., Simovici, D.: Generating an informative cover for association rules. In: ICDM 2002, pp. 597–613 (2002)
7. Guigues, J.-L., Duquenne, V.: Famille minimale d'implications informatives résultant d'un tableau de données binaires. Math. et Sciences Humaines 24, 5–18 (1986)
8. Freitas, A.: Understanding the crucial differences between classification and discovery of association rules. SIGKDD Explorations 2, 65–69 (2000)
9. Ganter, B., Wille, R.: Formal Concept Analysis. Springer, Heidelberg (1999)
10. Goethals, B., Muhonen, J., Toivonen, H.: Mining non-derivable association rules. In: SDM 2005 (2005)
11. Khardon, R., Roth, D.: Reasoning with models. Artif. Intell. 87, 187–213 (1996)
12. Kryszkiewicz, M.: Representative Association Rules. In: Wu, X., Kotagiri, R., Korb, K.B. (eds.) PAKDD 1998. LNCS, vol. 1394, pp. 198–209. Springer, Heidelberg (1998)
13. Kryszkiewicz, M.: Representative Association Rules and Minimum Condition Maximum Consequence Association Rules. In: Żytkow, J.M. (ed.) PKDD 1998. LNCS, vol. 1510, pp. 361–369. Springer, Heidelberg (1998)
14. Li, G., Hamilton, H.: Basic association rules. In: SDM 2004 (2004)
15. Luxenburger, M.: Implications partielles dans un contexte. Mathématiques et Sciences Humaines 29, 35–55 (1991)
16. Pasquier, N., Taouil, R., Bastide, Y., Stumme, G., Lakhal, L.: Generating a condensed representation for association rules. J. Intell. Inform. Sys. 24, 29–60 (2005)
17. Ullman, J., Widom, J.: A First Course in Database Systems (1997)
18. Zaki, M.: Mining non-redundant association rules. Data Mining and Knowledge Discovery 9, 223–248 (2004)
19. Zaki, M., Ogihara, M.: Theoretical foundations of association rules. In: DMKD Workshop on research issues in DMKD (1998)

Constructing Iceberg Lattices from Frequent Closures Using Generators

Laszlo Szathmary[1], Petko Valtchev[1], Amedeo Napoli[2], and Robert Godin[1]

[1] Dépt. d'Informatique UQAM, C.P. 8888,
Succ. Centre-Ville, Montréal H3C 3P8, Canada
Szathmary.L@gmail.com, valtchev.petko@uqam.ca, godin.robert@uqam.ca
[2] LORIA UMR 7503, B.P. 239, 54506 Vandœuvre-lès-Nancy Cedex, France
napoli@loria.fr

Abstract. Frequent closures (FCIs) and generators (FGs) as well as the precedence relation on FCIs are key components in the definition of a variety of association rule bases. Although their joint computation has been studied in concept analysis, no scalable algorithm exists for the task at present. We propose here to reverse a method from the latter field using a fundamental property of hypergraph theory. The goal is to extract the precedence relation from a more common mining output, i.e. closures and generators. The resulting order computation algorithm proves to be highly efficient, benefiting from peculiarities of generator families in typical mining datasets. Due to its genericity, the new algorithm fits an arbitrary FCI/FG-miner.

1 Introduction

The discovery of frequent patterns and meaningful association rules is a key data mining task [1] whereby a major challenge is the handling of the huge number of potentially useful patterns and rules. As a possible remedy, various subfamilies have been designed that losslessly represent the entire family of valid associations (see [2] for a survey). Some of the most popular bases involve subfamilies of frequent itemsets (FIs), e.g. closures (FCIs) or generators (FGs), which themselves losslessly represent the entire FI family. Part of these bases further require the precedence relation among closures as well (e.g. the *informative basis*).

The aforementioned three structural components, i.e. FCIs, FGs, and precedence, have been targeted in various configurations and from diverging viewpoints. A range of data mining methods compute generators and closures (e.g. *Titanic* [3] and *A-Close* [4]), and at least one targets closures and their order (e.g. *Charm-L* [5]). In concept analysis, in turn, the focus has been on computing both closures and order in the concept lattice [6], whereas a few methods also output the generators (e.g. [7,8]). Dedicated methods for precedence computation exist as well, yet their reliance on transaction-wise operations hurts their scalability (e.g. [9]).

Here we tackle the efficient computation of all three components. More precisely, we concentrate on the task of computing precedence links from the families of FCIs and FGs, i.e. from the output of some miners from the literature (see above). In our analysis of the problem, we reverse an argument of Pfaltz for computing generators from closures and precedence [7], using a fundamental result from hypergraph theory. The restated problem amounts to the computation of the minimal transversal graph of a given hypergraph, a popular problem that nevertheless withholds some secrets [10]. Based on an adaptation of a classical algorithm for the task, we provide our algorithm called *Snow* for efficient iceberg lattice construction. Our preliminary experiments show that the strategy is very suitable as the order computation cost is only a fraction of the one for discovering all FCIs and FGs.

The contribution of the paper is therefore threefold. First, we put forward an important interplay between precedence and generators with respect to closures. Second, we show that in practice it can be efficiently exploited to yield either the generators given the FCIs and the precedence relation, or the precedence relation given the FCIs and the generators. Third, the proposed concrete algorithm, *Snow*, provides the capabilities of iceberg lattice construction to any FCI/FG-miner.

The paper is organized as follows. Section 2 provides the basic concepts of frequent itemset mining, concept analysis, and hypergraph theory. In Section 3, we introduce the *Snow* algorithm. Finally, conclusions and future work are discussed in Section 4.

2 Background on Frequent Itemsets, Iceberg Lattices, and Hypergraphs

Here we recall the basic notions of frequent itemset mining, formal concept analysis, and hypergraph theory on which our approach is based.

2.1 Frequent Itemsets and Their Distinguished Subfamilies

Consider the following 5×5 sample dataset: $\mathcal{D} = \{(1,\ ACDE), (2,\ ABCDE), (3,\ AB), (4,\ D), (5,\ B)\}$. Throughout the paper, we will refer to this example as **"dataset $\mathcal{D}$"**.

We consider a set of *objects* or *transactions* $\mathcal{O} = \{o_1, o_2, \ldots, o_m\}$, a set of *attributes* or *items* $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$, and a relation $\mathcal{R} \subseteq \mathcal{O} \times \mathcal{A}$. A set of items is called an *itemset*. Each transaction has a unique identifier (*tid*), and a set of transactions is called a *tidset*.[1] For an itemset X, we denote its corresponding tidset, often called its *image*, as $t(X)$. For instance, in dataset $\mathcal{D}$, the image of AB is 23, i.e. $t(AB) = 23$. Conversely, $i(Y)$ is the itemset corresponding to a tidset Y. The *length* of an itemset is its cardinality, whereas an itemset of length k is called a k-itemset. The *support* of an itemset X, denoted by $supp(X)$, is the

[1] For convenience, we write an itemset $\{A, B, E\}$ as ABE, and a tidset $\{2,3\}$ as 23.

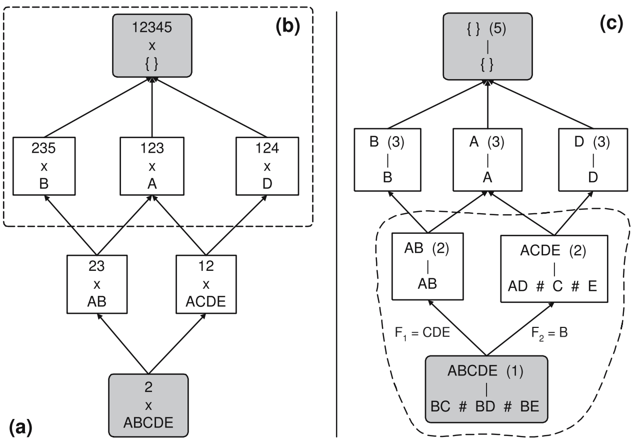


Fig. 1. Concept lattices of dataset $\mathcal{D}$. **(a)** The entire concept lattice. **(b)** An iceberg part of (a) by $min\_supp = 3$ (indicated by a dashed rectangle). **(c)** Although generators are not a formal part of the lattice, they are drawn within the respective nodes.

size of its image, i.e. $supp(X) = |t(X)|$. An itemset X is called *frequent*, if its support is not less than a given *minimum support* (denoted by min_supp), i.e. $supp(X) \geq min\_supp$. The image function induces an equivalence relation on $\wp(\mathcal{A})$: $X \cong Z$ iff $t(X) = t(Z)$ [11]. Moreover, an equivalence class has a unique maximum w.r.t. set inclusion and possibly several minima, called *closed* itemset (a.k.a. concept intents in concept analysis [12]) and *generator* itemsets (a.k.a. key-sets in database theory or free-sets), respectively. The support-oriented definitions exploiting the monotony of support upon $\subseteq$ in $\wp(\mathcal{A})$ are as follows:

Definition 1 (closed itemset; generator). *An itemset X is* closed[2] *(*generator*) if it has no proper superset (subset) with the same support (respectively).*

The *closure* operator assigns to an itemset X the maximum of its equivalence class. For instance, in dataset $\mathcal{D}$, the sets AB and AD are generators, and their closures are AB and $ACDE$, respectively (i.e. the equivalence class of AB is a singleton).

The families of frequent closed itemsets (FCIs) and frequent generators (FGs) are well-known reduced representations [13] for the set of all frequent itemsets (FIs). Furthermore, they underlie some non-redundant bases of valid association rules such as the *generic basis* [2]. Yet for other bases, the inclusion order between FCIs is essential, and, in some cases, the precedence order between those. The precedence relation $\prec$, henceforth referred to as merely *precedence*, is defined the following way: $X \prec Z$ iff *(i)* $X \subset Z$, and *(ii)* there exists no Y such that $X \subset Y \subset Z$. Here, X is called the (immediate) *predecessor* of Z.

[2] In the rest of the paper, closed itemsets are abbreviated as **"CIs"**.

The FCI family of a dataset together with $\prec$ compose the *iceberg lattice*, which is a complete meet-semi-lattice (bottomless if $\emptyset$ is the universal itemset). The iceberg corresponds to the frequent part of the CI lattice, also known in concept analysis [12] as the intent lattice of a context. It is dually isomorphic to the concept lattice of the same context (in data mining terms, the lattice of all pairs (tidset, itemset) where both are closed and mutually corresponding). We shall use here the latter structure as visualization basis hence the effective order on our drawings is $\supseteq$ rather than $\subseteq$. In Figure 1, (a) and (b) depict the concept lattice of dataset $\mathcal{D}$ and its iceberg part, respectively.

As we tackle here the computation of an FG-decorated iceberg, i.e. an iceberg lattice where generators are explicitly associated to their closure (see also Figure 1 (c)), existing approaches for related tasks are of interest. In the data mining field, FCIs together with associated FGs have been targeted by a growing set of levelwise FCI-miners such as *Titanic* [3], *A-Close* [14], etc. In contrast, the only case of mining FCIs with precedence that we are aware of is *Charm-L* [5]. The research in concept analysis algorithms has put the emphasis on the set of concepts, or CIs, and less so on precedence (see [6] for a good coverage of the topic). Few methods also compute the generators [7,8], whereas focused procedures retrieve precedence from the concept set, as in [9]. However, all but few concept analysis methods perform at least part of the computation transaction-wise, which makes them impractical for large datasets.

Yet a close examination of the most relevant approaches, i.e. those computing generator-decorated lattices such as in [7], reveals an interesting property that we shall exploit in our own method. In fact, as the latter paper indicates, from the set of all closures and their precedence, one may easily compute the generators for each closure.

The key notion here is the *blocker* of a family of sets (equivalent to a hypergraph transversal as we show below). Thus, given a ground set X and a family of subsets $\mathcal{X} \subseteq \wp(X)$, a blocker of $\mathcal{X}$ is a set $Z \subseteq X$ which intersects every member thereof to a non-empty result ($\forall T \in \mathcal{X}, Z \cap T \neq \emptyset$). A *minimal blocker* is the one which admits no other blocker as a proper subset. Blockers are brought into the closure lattice using the associated *faces*, i.e. the differences between two adjacent closures within the lattice. Formally, given two CIs X_1 and X_2 of a dataset such that $X_1 \prec X_2$, their associated face is $F = X_2 \setminus X_1$.

Example. Consider the closure lattice in Figure 1 (c). In a node, it depicts the corresponding CI, its support and the list of its generators. Let us consider the bottom concept with the closure $ABCDE$. It has two predecessors, thus its faces are: $F_1 = ABCDE \setminus AB = CDE$ and $F_2 = ABCDE \setminus ACDE = B$.

A basic property of the generators of a CI X states that they are the minimal blockers of the family of faces associated to X [7]:

Theorem 1. *Assume a CI X and let $\mathcal{F} = \{F_1, F_2, \ldots, F_k\}$ be its family of associated faces. Then a set $Z \subseteq X$ is a minimal generator of X iff X is a minimal blocker of $\mathcal{F}$.*

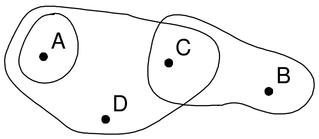


Fig. 2. A hypergraph $\mathcal{H}$, where $V = \{A, B, C, D\}$ and $\mathcal{E} = \{A,\ BC,\ ACD\}$

Example. The minimal blockers of the family $\{CDE, B\}$ are $\{BC, BD, BE\}$, which is exactly the set of minimal generators of $ABCDE$ (see Figure 1 (c)).

From Theorem 1, Pfaltz devised a method for computing the generators of a closed set from its predecessors in the lattice [15]. Although the precedence links are assumed as input there, their computation, as indicated above, requires expensive manipulations of tidsets. Thus, even though the algorithm itself employs scalable operations, its input is impossible to provide at low cost.

However, this apparent deadlock can be resolved by transposing the problem setting into the more general framework of hypergraphs and transversals.

2.2 Hypergraphs and Their Transversal Graphs

A hypergraph [16] is a generalization of a graph, where edges can connect arbitrary number of vertices.

Definition 2 (hypergraph). *A* hypergraph *is a pair $(V,\mathcal{E})$ of a finite set $V = \{v_1, v_2, \ldots, v_n\}$ and a family $\mathcal{E}$ of subsets of V. The elements of V are called vertices, the elements of $\mathcal{E}$ edges. A hypergraph is* simple *if none of its edges is contained in any other of its edges, i.e. $\forall \mathcal{E}_i, \mathcal{E}_j \in \mathcal{E} : \mathcal{E}_i \subseteq \mathcal{E}_j \Rightarrow i = j$.*

Example. The hypergraph $\mathcal{H}$ in Figure 2 is not simple because the edge A is contained in the edge ACD.

A transversal of a hypergraph $\mathcal{H}$ is a subset of its vertices intersecting each edge of $\mathcal{H}$. A *minimal* transversal does not contain any other transversal as proper subset.

Definition 3 (transversal). *Let $\mathcal{H} = (V, \mathcal{E})$ be a hypergraph. A set $T \subseteq V$ is called a* transversal *of $\mathcal{H}$ if it meets all edges of $\mathcal{H}$, i.e. $\forall E \in \mathcal{E} : T \cap E \neq \emptyset$. A transversal T is called* minimal *if no proper subset T' of T is a transversal.*

Clearly, the notion of (*minimal*) *blocker* in the work of Pfaltz [7] is equivalent to the notion of (minimal) transversal.

Example. The hypergraph $\mathcal{H}$ in Figure 2 has two minimal transversals: AB and AC. The sets ABC and ACD are transversals but they are not minimal.

Definition 4 (transversal hypergraph). *The family of all minimal transversals of $\mathcal{H}$ constitutes a hypergraph on V called the* transversal hypergraph *of $\mathcal{H}$, which is denoted by $Tr(\mathcal{H})$.*

Example. Considering the hypergraph $\mathcal{H}$ in Figure 2, $Tr(\mathcal{H}) = \{AB, AC\}$.

An obvious property of $Tr(\mathcal{H})$ is that it is necessarily simple. Next, a duality exists between a simple hypergraph and its transversal hypergraph [16]:

Proposition 1. *Let $\mathcal{G}$ and $\mathcal{H}$ be two simple hypergraphs. Then $\mathcal{G} = Tr(\mathcal{H})$ if and only if $\mathcal{H} = Tr(\mathcal{G})$.*

Consequently, computing twice the transversal hypergraph of a given simple hypergraph yields the initial hypergraph.

Corollary 1 (duality). *Let $\mathcal{H}$ be a simple hypergraph. Then $Tr(Tr(\mathcal{H})) = \mathcal{H}$.*

Example. Consider the following *simple* hypergraph: $\mathcal{G} = \{A, BC\}$. Then, $\mathcal{G}' = Tr(\mathcal{G}) = Tr(\{A, BC\}) = \{AB, AC\}$, and $Tr(\mathcal{G}') = Tr(\{AB, AC\}) = \{A, BC\}$.

From a computational point of view, the extraction of $Tr(\mathcal{G})$ from $\mathcal{G}$ is a tough problem as the former can be exponentially larger than the latter. In fact, the exact complexity class of the problem is still not known [10]. Yet many algorithms for the task exist and perform well in practice since the worst case rarely occurs. For instance, an incremental algorithm due to Berge [16] computes the $Tr(\mathcal{G})$ as the final member of a sequence of hypergraphs, each representing the transversal hypergraph of a subgraph of $\mathcal{G}$. It is noteworthy that the algorithm in [7] follows a similar pattern in computing minimal blockers of a set family.

3 The Snow Algorithm

Snow computes precedence links on FCIs from associated generators by exploiting the duality with faces.

3.1 Underlying Structural Results

As indicated in the previous section, a minimal blocker of a family of sets is an identical notion to a minimal transversal of a hypergraph. This trivially follows from the fact that each hypergraph $(V,\mathcal{E})$ is nothing else than a family of sets drawn from $\wp(V)$. Now following Theorem 1, we conclude that given a CI X, the associated generators compose the transversal hypergraph of its family of faces $\mathcal{F}$ seen as the hypergraph $(X,\mathcal{F})$.

Next, further to the basic property of a transversal hypergraph, we conclude that $(X,\mathcal{F})$ is necessarily simple. In order to apply Proposition 1, we must also show that the family of generators associated to a CI, say $\mathcal{G}$, forms a simple hypergraph. Yet this holds trivially due to the definition of generators. We can therefore advance that both families represent two mutually corresponding hypergraphs.

Table 1. Input of *Snow* on dataset $\mathcal{D}$ by $min\_supp = 1$

FCI (supp)	FGs
AB (2)	AB
$ABCDE$ (1)	BE; BD; BC
A (3)	A

FCI (supp)	FGs
B (3)	B
$ACDE$ (2)	E; C; AD
D (3)	D

Property 1. Let X be a closure and let $\mathcal{G}$ and $\mathcal{F}$ be the family of its generators and the family of its faces, respectively. Then, for the underlying hypergraphs it holds that $Tr(X, \mathcal{G}) = (X, \mathcal{F})$ and $Tr(X, \mathcal{F}) = (X, \mathcal{G})$.

Example. Let us again consider the bottom concept in Figure 1 (c) with $ABCDE$ as its CI. It has three generators: BC, BD, and BE. The transversal hypergraph of the generator family is $Tr(\{BC, BD, BE\}) = \{CDE, B\}$. That is, it corresponds exactly to the family of faces as computed above.

3.2 The Algorithm

The *Snow* algorithm exploits Property 1 by computing faces from generators. Thus, its input is made of FCIs and their associated FGs. Several algorithms can be used to produce this input, e.g. *A-Close* [4], *Titanic* [3], *Zart* [17], *Eclat-Z* [18], etc. Table 1 depicts a sample input of *Snow*.

On such data, *Snow* first computes the faces of a CI as the minimal transversals of its generator hypergraph. Next, each difference of the CI X with a face yields a predecessor of X in the closure lattice.

Example 1. Consider again $ABCDE$ with its generator family $\{BC, BD, BE\}$. First, we compute its transversal hypergraph: $Tr(\{BC, BD, BE\}) = \{CDE, B\}$. The two faces $F_1 = CDE$ and $F_2 = B$ indicate that there are two predecessors for $ABCDE$, say Z_1 and Z_2, where $Z_1 = ABCDE \setminus CDE = AB$, and $Z_2 = ABCDE \setminus B = ACDE$. Application of this procedure for all CIs yields the entire precedence relation for the CI lattice. □

The pseudo code of *Snow* is given in Algorithm 3.2. As input, *Snow* receives a set of CIs and their associated generators. The `identifyOrCreateTopCI` procedure looks for a CI whose support is 100%. If it does not find one, then it creates it by taking an empty set as the CI with 100% support and a void family of generators (see Figure 1 (c) for an example). The `getMinTransversals` function computes the transversal hypergraph of a given hypergraph. More precisely, given the family of generators of a CI X, the function returns the family of faces of X. It is noteworthy that any algorithm for transversal computation in a hypergraph would be appropriate here. In our current implementation, we use an optimized version of Berge's algorithm henceforth referred to as *BergeOpt* that we do not present here due to space limitations. The `getPredecessorCIs` function calculates the differences between a CI X and the family of faces of X. The

```
Algorithm 1 (Snow):

Description: build iceberg lattice from FCIs and FGs
Input:       a set of CIs and their associated generators

 1) identifyOrCreateTopCI($setOfFCIsAndFGs$);
 2) // find the predecessor(s) for each concept:
 3) for all $c$ in $setOfFCIsAndFGs$ {
 4)       $setOfFaces \leftarrow$ getMinTransversals($c$.generators);
 5)       $predecessorCIs \leftarrow$ getPredecessorCIs($c$.closure, $setOfFaces$);
 6)       loop over the CIs in $predecessorCIs$ ($p$) {
 7)             connect($c$, $p$);
 8)       }
 9) }
```

function returns the set of all CIs that are predecessors of X. The `connect` procedure links the current CI to its predecessors.

For a running example, see Example 1.

3.3 Experimental Results

The *Snow* algorithm was implemented in Java in the Coron data mining platform [19].[3] The experiments were carried out on a bi-processor Intel Quad Core Xeon 2.33 GHz machine with 4 GB RAM running under Ubuntu GNU/Linux. All times reported are real, wall clock times.

For the experiments, we used several real and synthetic dataset benchmarks. Database characteristics are shown in Table 2 (top). The chess and connect datasets are derived from their respective game steps. The Mushrooms database describes mushrooms characteristics. These three datasets can be found in the UC Irvine Machine Learning Database Repository. The pumsb, C20D10K, and C73D10K datasets contain census data from the PUMS sample file. The synthetic datasets T20I6D100K and T25I10D10K, using the IBM Almaden generator, are constructed according to the properties of market basket data. Typically, real datasets are very dense, while synthetic data are usually sparse.

Table 2 (bottom left and right) provides a summary of the experimental results. The first column specifies the various minimum support values for each of the datasets (low for the sparse dataset, higher for dense ones). The second and third columns comprise the number of FCIs and the execution time of *Snow* (given in seconds). The CPU time does not include the cost of computing FCIs and FGs since they are assumed as given.

As can be seen, *Snow* is able to discover the order very efficiently in both sparse and dense datasets. To explain the reason for that, recall that the only

[3] `http://coron.loria.fr`

Table 2. Top: database characteristics. **Bottom:** response times of *Snow*.

database name	# records	# non-empty attributes	# attributes (in average)	largest attribute
T20I6D100K	100,000	893	20	1,000
T25I10D10K	10,000	929	25	1,000
chess	3,196	75	37	75
connect	67,557	129	43	129
pumsb	49,046	2,113	74	7,116
MUSHROOMS	8,416	119	23	128
C20D10K	10,000	192	20	385
C73D10K	10,000	1,592	73	2,177

min_supp	# concepts (including top)	*Snow* (finding order)
T20I6D100K		
0.75%	4,711	0.11
0.50%	26,209	0.36
0.25%	149,218	3.24
T25I10D10K		
0.40%	83,063	1.07
0.30%	122,582	2.73
0.20%	184,301	4.48
chess		
65%	49,241	0.85
60%	98,393	1.77
55%	192,864	3.95
connect		
65%	49,707	0.54
60%	68,350	0.78
55%	94,917	1.82

min_supp	# concepts (including top)	*Snow* (finding order)
pumsb		
80%	33,296	1.95
78%	53,418	4.10
76%	82,539	7.08
MUSHROOMS		
20%	1,169	0.05
10%	4,850	0.17
5%	12,789	0.47
C20D10K		
0.50%	132,952	3.04
0.40%	151,394	4.37
0.30%	177,195	4.29
C73D10K		
65%	47,491	1.51
60%	108,428	3.97
55%	222,253	10.13

computationally intensive step in *Snow* is the transversal hypergraph construction. Thus, the total cost heavily depends on the efficiency of that step. Furthermore, to find out why the underlying algorithm *BergeOpt* performs so well, we investigated the size of its input data. Figure 3 shows the distribution of hypergraph sizes in the datasets T20I6D100K, MUSHROOMS, chess, and C20D10K.[4] Note that we obtained similar hypergraph-size distributions in the other four datasets too. Figure 3 indicates that most hypergraphs only have 1 edge, which is a trivial case, whereas large hypergraphs are relatively rare. As a consequence, *BergeOpt* and thus *Snow* perform very efficiently.

We interpret the above results as an indication that the good performance of *Snow* is independent of the density of the dataset. In other terms, provided that the input hypergraphs do not contain too many edges, i.e. there are only few FGs per FCIs, the computation is very fast. A natural question arises with this

[4] For instance, the dataset T20I6D100K by *min_supp* = 0.25% contains 149,019 1-edged hypergraphs, 171 2-edged hypergraphs, 25 3-edged hypergraphs, 0 4-edged hypergraphs, 1 5-edged hypergraph, and 1 6-edged hypergraph.

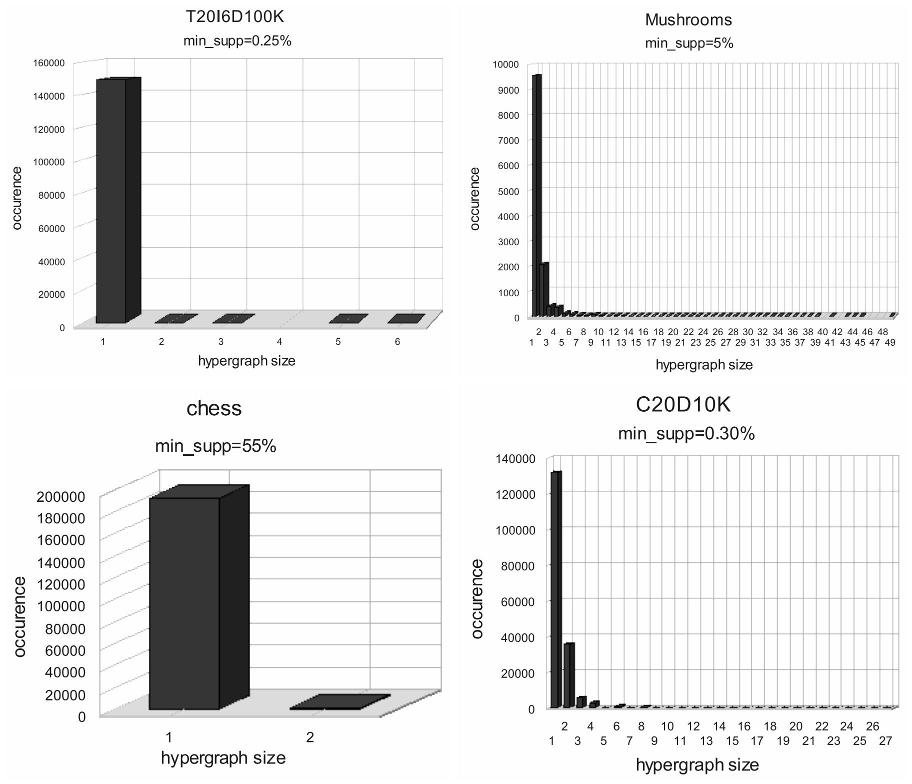


Fig. 3. Distribution of hypergraph sizes

observation: does the modest number of FGs in each class hold for all realistic datasets in the literature? If not, could one profile those datasets which meet this condition?

4 Conclusion

The computation of precedence of FCIs is a challenging task due to the potentially huge number of these. Indeed, as most of the existing algorithms rely on dimensions that may grow rapidly, they remain impractical for large datasets.

We presented here an approach for elegantly solving the problem starting from a rather common mining output, i.e. FCIs and their FGs, which is typically provided by levelwise FCI-miners. To that end, we reverse a computation principle initially introduced by Pfaltz in closure systems by translating it beforehand within the minimal transversal framework of hypergraph theory. Although the cost of the transversal hypergraph problem is potentially non-polynomial, the contributed algorithm, *Snow*, proved to be highly efficient in practice largely due to the low number of FGs associated to an FCI.

Based on this observation, we claim that *Snow* can enhance in a generic yet efficient manner any FCI/FG-miner, thus transforming the latter into an iceberg lattice constructor. Beside the possibility to compute valuable association rule bases from its output, the resulting method could also compete on the full-fledged concept lattice construction field.

On the methodological side, our study underlines the duality between generators and order w.r.t. closures: either can be used in combination with FCIs to yield the other one. It rises the natural question of whether FCIs alone could be used to efficiently retrieve both precedence and FGs. Conversely, it would be interesting to examine whether FCIs can in turn be easily computed from order and generators, yet this is more of a discrete mathematics challenge rather than data mining concern.

References

1. Agrawal, R., Srikant, R.: Fast Algorithms for Mining Association Rules in Large Databases. In: Proc. of the 20th Intl. Conf. on Very Large Data Bases (VLDB 1994), pp. 487–499. Morgan Kaufmann Publishers Inc., San Francisco (1994)
2. Kryszkiewicz, M.: Concise Representations of Association Rules. In: Proc. of the ESF Exploratory Workshop on Pattern Detection and Discovery, pp. 92–109 (2002)
3. Stumme, G., Taouil, R., Bastide, Y., Pasquier, N., Lakhal, L.: Computing Iceberg Concept Lattices with TITANIC. Data and Knowledge Engineering 42(2), 189–222 (2002)
4. Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Discovering Frequent Closed Itemsets for Association Rules. In: Beeri, C., Bruneman, P. (eds.) ICDT 1999. LNCS, vol. 1540, pp. 398–416. Springer, Heidelberg (1998)
5. Zaki, M.J., Hsiao, C.J.: Efficient Algorithms for Mining Closed Itemsets and Their Lattice Structure. IEEE Transactions on Knowledge and Data Engineering 17(4), 462–478 (2005)
6. Carpineto, C., Romano, G.: Concept Data Analysis: Theory and Applications. John Wiley & Sons, Ltd., Chichester (2004)
7. Pfaltz, J.L.: Incremental Transformation of Lattices: A Key to Effective Knowledge Discovery. In: Corradini, A., Ehrig, H., Kreowski, H.-J., Rozenberg, G. (eds.) ICGT 2002. LNCS, vol. 2505, pp. 351–362. Springer, Heidelberg (2002)
8. Nehme, K., Valtchev, P., Rouane, M.H., Godin, R.: On Computing the Minimal Generator Family for Concept Lattices and Icebergs. In: Ganter, B., Godin, R. (eds.) ICFCA 2005. LNCS (LNAI), vol. 3403, pp. 192–207. Springer, Heidelberg (2005)
9. Nourine, L., Raynaud, O.: A fast algorithm for building lattices. Inf. Process. Lett. 71(5–6), 199–204 (1999)
10. Eiter, T., Gottlob, G.: Identifying the Minimal Transversals of a Hypergraph and Related Problems. SIAM Journal on Computing 24(6), 1278–1304 (1995)
11. Bastide, Y., Taouil, R., Pasquier, N., Stumme, G., Lakhal, L.: Mining frequent patterns with counting inference. SIGKDD Explor. Newsl. 2(2), 66–75 (2000)
12. Ganter, B., Wille, R.: Formal concept analysis: mathematical foundations, p. 284. Springer, Heidelberg (1999)

13. Calders, T., Rigotti, C., Boulicaut, J.F.: A Survey on Condensed Representations for Frequent Sets. In: Boulicaut, J.-F., De Raedt, L., Mannila, H. (eds.) Constraint-Based Mining and Inductive Databases. LNCS (LNAI), vol. 3848, pp. 64–80. Springer, Heidelberg (2006)
14. Pasquier, N.: Mining association rules using formal concept analysis. In: Proc. of the 8th Intl. Conf. on Conceptual Structures (ICCS 2000), August 2000, pp. 259–264. Shaker-Verlag (2000)
15. Pfaltz, J.L., Taylor, C.M.: Scientific Knowledge Discovery through Iterative Transformation of Concept Lattices. In: Proc. of the Workshop on Discrete Applied Mathematics in conjunction with the 2nd SIAM Intl. Conf. on Data Mining, Arlington, VA, USA, pp. 65–74 (2002)
16. Berge, C.: Hypergraphs: Combinatorics of Finite Sets. North Holland, Amsterdam (1989)
17. Szathmary, L., Napoli, A., Kuznetsov, S.O.: ZART: A Multifunctional Itemset Mining Algorithm. In: Proc. of the 5th Intl. Conf. on Concept Lattices and Their Applications (CLA 2007), Montpellier, France, October 2007, pp. 26–37 (2007)
18. Szathmary, L., Valtchev, P., Napoli, A., Godin, R.: An Efficient Hybrid Algorithm for Mining Frequent Closures and Generators. In: Proc. of the 6th Intl. Conf. on Concept Lattices and Their Applications (CLA 2008), Olomouc, Czech Republic (accepted, 2008)
19. Szathmary, L.: Symbolic Data Mining Methods with the Coron Platform. PhD Thesis in Computer Science, Univ. Henri Poincaré – Nancy 1, France (November 2006)

Learning from Each Other

Christopher Dartnell[1], Éric Martin[2], and Jean Sallantin[1]

[1] LIRMM, UM2, 161, rue Ada, 34392 Montpellier Cedex 5, France
christopher.dartnell@gmail.com,js@lirmm.fr
[2] School of Computer Science and Eng., UNSW Sydney NSW 2052, Australia
emartin@cse.unsw.edu.au

Abstract. Since its inception, the field of machine learning has seen the advent of several learning paradigms, designed to frame the issues central to the learning activity, provide effective learning methods, and investigate the power and limitations inherent to the process of successful learning. In this article, we propose a formalization that underlies the key concepts of many such paradigms and discuss their relevance to scientific discovery, with the aim of assessing what scientists can expect from machines designed to assist them in their quest for the discovery of valid laws. We illustrate the formalization on several variations of a card game, and highlight the differences that paradigms impose on learners, as well as the assumptions they make on the nature of the learning process. We then use the formalization to describe a multi-agent interaction protocol, that has been inspired by these paradigms and that has been validated recently on some groups of agents. Finally, we propose extensions to this protocol.

Keywords: Logic, Machine learning, Interaction for Knowledge Discovery.

Introduction

Since machine learning emerged as a new field almost 50 years ago, several learning paradigms have been proposed with two main objectives: frame the very nature of the learning process, both qualitatively and quantitatively, and provide effective algorithms that can be implemented for specific learning tasks. Identification in the limit [1], query learning [2], and PAC-learning [3] are some of the paradigms that have had strong impact on the machine learning community. All those paradigms propose a different way of modeling reality, suggest a different form of interaction between a learner and its environment, and put forward different criteria of success for the learning process. We propose in Section 1 a logical formalization suitable to subsume these paradigms and discuss the relevance of these paradigms in the context of scientific discovery, and more precisely, discuss how the paradigms can provide a blueprint to assist practicing researchers with learning machines. We illustrate the formalization on several variations of a card game, and highlight how different paradigms affect the rules of the game and rely on different assumptions about how learners and the environment should be modeled. In particular, we distinguish between passive learning (Section 2) and active learning (Section 3), and emphasize their limits in the context of scientific discovery. We then use the formalization to reformulate in Section 4 a multi-agent interaction protocol inspired both by these paradigms and by the epistemology of Karl Popper [4]. This protocol has also been implemented under the form of card game and has been recently validated on some groups of agents in [5]. Finally, we propose extensions to this protocol.

1 Formal Basis

All problems to be considered will be based on an underlying abstract representation of some form of reality, which is as simple and intuitive as possible but rich enough to illustrate the various aspects of scientific discovery that we want to discuss in this paper. We take every possible form of reality to be an infinite sequence of cards taken from the classical deck of 52 cards, where every card can occur infinitely many times. Our representation of a sequence of cards is based on a logical language that has only two function symbols, namely a constant $\overline{0}$ and a unary function symbol s. The term obtained from $\overline{0}$ by n successive applications of s is denoted $\overline{n}$, and intuitively refers to the n^{th} card in the sequence.

To describe a card, we use some of the observational predicates, namely, the members of both sets of unary predicate symbols below:

$$Prd_{suit}(\mathcal{V}) = \{hearts, spades, clubs, diamonds\}, \text{ and}$$

$$Prd_{rank}(\mathcal{V}) = \{ace, two, ..., ten, jack, queen, king\}$$

This vocabulary is rich enough to describe any possible sequence of cards. For instance, the sequence that starts with the queen of hearts, followed by the king of spades, followed by the third of diamonds, ..., is represented by the theory

$$queen(\overline{0}), hearts(\overline{0}), king(\overline{1}), spades(\overline{1}), third(\overline{2}), diamonds(\overline{2}), ...$$

More generally, every possible sequence can be identified with a theory of the kind defined next:

Definition 1 (Possible Games). *We call* possible game *any theory $\mathcal{T}$ such that:*

- *for all $n \in \mathbb{N}$, $\mathcal{T}$ contains one and only one member of $Prd_{rank}(\mathcal{V})$,*
- *for all $n \in \mathbb{N}$, $\mathcal{T}$ contains one and only one member of $Prd_{suit}(\mathcal{V})$,*
- *$\mathcal{T}$ contains no other formula.*

Many possible games have no finite equivalent. Of course some sequences enjoy a finite representation. For instance, the sequence consisting of nothing but the queen of hearts is described by the formula

$$\forall x[hearts(x) \wedge queen(x)]. \tag{1}$$

When the problem is to discover the underlying possible game, games with a finite representation are clearly of special interest. However this does not mean that we should restrict ourselves to the class of possible games whose description is finite. Indeed, we also want to look at classification problems, where the aim is to determine whether the underlying possible game has some property *i.e.*, belongs to a particular class. In that case, what is of special interest is that the class be finitely describable. The difference between finitely and non-finitely describable classes can be illustrated in terms of complexity measure. For instance, [6], [7] proposed a formal framework to measure the level of desorder in a system. Intuitively, a random sequence of bytes can only be enumerated whereas an organised one can be compressed. The size of the smaller program able to generate this sequence then represents an acceptable approximation of its complexity. Finitey describable classes of possible games are those whose description can be compressed thanks to their organisation.

Both identification and classification will be discussed at length in the remainder of the paper. For the moment, let us focus on the finite description property. We shall start by providing a better intuition for the possible games that are finitely describable. For instance, the sequence which starts with ten queens of hearts, and then consists of nothing else but the king of spades, is captured by the following finite description of that possible game:

$$\{queen(\overline{n}), hearts(\overline{n}) \mid n < 10\} \cup \{king(\overline{10}), spades(\overline{10})\} \cup \\ \{\forall x[(king(x) \rightarrow king(s(x))) \wedge (spades(x) \rightarrow spades(s(x)))]\} \quad (2)$$

It is not hard to see that the class of finitely describable possible games is precisely the class of card sequences that, at some point, become cyclic. For instance, the class of possible games that describes sequences that start with a finite subsequence of queens of hearts, of arbitrary length, followed by nothing but the king of spades, is finally described by the following finite theory

$$\{\forall x[(king(x) \wedge spades(x)) \vee (queen(x) \wedge hearts(x))], \\ \exists x\, king(x), \forall x[king(x) \rightarrow king(s(x))]\} \quad (3)$$

When it comes to classification and describing properties, this minimal language captures classes of possible games that at some point, have a cyclic property. As an example, the class of possible games describing sequences where a queen occurs in any subsequence of three successive cards is equivalent to the formula

$$\forall x[queen(x) \vee queen(s(x)) \vee queen(s(s(x)))].$$

On the other hand, with the current set of predicate symbols, we cannot define the class of sequences where the queen of hearts alternates with the king of spades with an arbitrary sequence of cards between an occurrence of a queen of hearts and the next occurrence of the king of spades, and the other way around. We need to extend our minimal vocabulary to represent this kind of sequencial or temporal organisation. For instance, this class becomes finitely describable if we enrich the vocabulary with a new binary predicate symbol, $<$, interpreted as the strict order between natural numbers:

$$\begin{aligned}\{&\exists x[queen(x) \wedge hearts(x)],\\ &\quad\forall x[(queen(x) \wedge hearts(x)) \rightarrow \exists y(x < y \wedge king(y) \wedge spades(y))],\\ &\quad\quad\forall x[(king(x) \wedge spades(x)) \rightarrow \exists y(x < y \wedge queen(y) \wedge hearts(y))]\}\end{aligned} \tag{4}$$

We now consider vocabularies that extend the original vocabulary (consisting of $\overline{0}$, s, and the observational predicate symbols) with any number (possibly none) of predicate symbols whose interpretation over the set of possible games is fixed. We call these extra predicate symbols *theoretical predicates*. Such predicates enrich the expressive power as the $<$ symbol just discussed. Other theoretical predicates could just provide notational convenience, for instance with $red(x)$ as an abbreviation for $hearts(x) \vee diamonds(x)$.

We assume that every theoretical predicate is decidable as defined in Definition 2.

Definition 2. *Let $\mathcal{V}$ be a vocabulary consisting of all observational predicate symbols plus some number of theoretical predicate symbols. We say that $\mathcal{V}$ is* decidable *just in case for all $k \in \mathbb{N}$, $n_1, \ldots, n_k \in \mathbb{N}$, k-ary theoretical predicates q and possible games $\mathcal{G}$, the truth of $q(\overline{n_1}, \ldots, \overline{n_k})$ in $\mathcal{G}$ is determined by the truth in $\mathcal{G}$ of all formulas of the form $p(\overline{n_j})$ where $0 \leq j \leq k$ and p is an observational predicate.*

Equivalently, given a theoretical k-ary predicate symbol q, the atomic formula $q(x_1, \ldots, x_n)$ is an abbreviation for a boolean combination φ of formulas of the form $p(x)$ where p is an observational predicate symbol, and x is one of $x_1, \ldots,$ x_n; φ amounts of a definition of q. Note that φ could contain a subformula that itself defines another theoretical predicate which would simplify the definition of q, which instead of being defined from observational predicates only, would be defined of possibly both observational and theoretical predicates.

Intuitively, theoretical predicates define relationships between cards that only depend on what these cards are and not on other cards that might or might not occur in the sequence; theoretical predicates have a definitional character.

Even though we accept and promote the use of a rich vocabulary, we want to guarantee that any possible game or class of possible games respect the natural ordering of the sequence of cards. This means that if we want to claim that the tenth card is a queen of hearts, that should only be based on the tenth card or on any card in the sequence that occurs before. For this purpose, we consider special kind of logic programs that we call *iterative logic programs*. We first define a general notion of logic program (Definition 3) and then specialize it to iterative logic programs (Definition 4).

Definition 3. *A* logic program *is a finite set of rules* (*over a decidable vocabulary*) *whose heads are atomic formulas, whose bodies are positive formulas,* i.e.*, are built from atomic formulas using conjunction, disjunction, existential quantification and universal quantification, and such that all variables that occur free in the body of a rule also occur in the head of that rule.*

All free variables in any rule of a logic program are implicitly universally quantified. It might seem that being able to only generate atoms is too restrictive. The use of theoretical predicates allows us to circumvent this apparent restriction. Suppose for instance that we want to predict that the n^{th} card is either *hearts* or *diamonds* depending on the previous cards. We cannot use a rule whose head is $hearts(x) \vee diamonds(x)$, but we can work with a vocabulary that contains a theoretical predicate *red* with its intended meaning that can then be used in the heads of the logic program.

Definition 4. *A logic program $\mathcal{P}$ is said to be* iterative *iff the following holds. Consider a rule $r \in \mathcal{P}$ with head $\wp(x_1, \ldots, x_k)$ and body ψ (where no variable except possibly x_1, ..., x_k occur free in ψ). Let $n_1, \ldots, n_k \in \mathbb{N}$ be given and let $\mathcal{G}$ be a possible game. Then the truth of $\psi[\overline{n_1}/x_1, \ldots, \overline{n_k}/x_k]$ in $\mathcal{G}$ is determined by the truth of all formulas of the form $\wp(\overline{m})$ where $m < \max(n_1, \ldots, n_k)$ and $\wp$ is an observational predicate symbol.*

For instance, an iterative logic program could contain the following rule, which expresses that a card is a spades provided that it is a king and is preceded by at least one hearts:

$$(\exists y[y < x \wedge hearts(y)] \wedge king(x)) \rightarrow spades(x)$$

Given a logic program $\mathcal{P}$, we denote by $[\mathcal{P}]$ the $\subseteq$-minimal set of all closed atoms that can be generated from $\mathcal{P}$ following the following process. Consider a rule $r \in \mathcal{P}$ with head $\wp(x_1, \ldots, x_k)$ and body ψ. Let $n_1, \ldots, n_k \in \mathbb{N}$ be given, Assume that every possible game that is a model of $[P]$ is also a model of $\psi[\overline{n_1}/x_1, \ldots, \overline{n_k}/x_k]$. Then $\wp(\overline{n_1}, \ldots, \overline{n_k})$ belongs to $[P]$.

A key property of iterative logic programs is given by the following property:

Property 1. Let $\mathcal{P}$ be an iterative logic program over a (decidable) vocabulary $\mathcal{V}$, then $[\mathcal{P}]$ is a recursive set.

This property follows from:

- the fact that atomic formulas built from a theoretical predicate symbol can be replaced by formulas built from observational predicate symbols only;
- the fact that for every rule $\psi \rightarrow \wp(x_1, \ldots, x_k)$ of an iterative logic program and every $n_1, \ldots, n_k \in \mathbb{N}$, $\psi[\overline{n_1}/x_1, \ldots, \overline{n_k}/x_k]$ can be mechanically transformed into a quantification free sentence in which all terms belong to $\{\overline{m} \mid m \leq \max(n_1, \ldots, n_k)\}$.

Since $[\mathcal{P}]$ consists of nothing but atomic sentences, $[\mathcal{P}]$ is always consistent. But $[\mathcal{P}]$ can be unsatisfiable in the sense that it has no intended interpretation, for instance because it contains, for some $n \in \mathbb{N}$, two distinct formulas of the form $\wp_1(\overline{n})$ and $\wp_2(\overline{n})$ where $\wp_1$ and $\wp_2$ both belong to Prd_{rank} or both belong to Prd_{suit}. Or for another example, if $[\mathcal{P}]$ contains both $red(\overline{10})$ and $spades(\overline{10})$, then $\mathcal{P}$ is also unsatisfiable with respect to our class of intended interpretations.

An iterative logic program is said to be *equivocal* iff it has a model in the class of possible games $\mathcal{W}$, and it is said to be *univocal* iff it has exactly one model in $\mathcal{W}$. The next property follows from the argument described after Property 1.

Property 2. Both univocity and equivocity are co-semi-decidable properties of a logic program.

To end this section, we describe a class of univocal logic programs. Let $<_{suit}$ and $<_{rank}$ be two arbitrary order relations on $Prd_{suit}(\mathcal{V})$ and $Prd_{rank}(\mathcal{V})$, respectively. For instance, let $<_{suit}$ and $<_{rank}$ be defined as follow:

– $hearts <_{suit} diamonds <_{suit} clubs <_{suit} spades.$
– $ace <_{rank} two <_{rank} \cdots <_{rank} ten <_{rank} jack <_{rank} queen <_{rank} king.$

Any iterative program of the following type guarantees the univocity property:

$$\begin{aligned} \varphi_1(x) &\rightarrow hearts(x) \\ \neg\varphi_1(x) \wedge \varphi_2(x) &\rightarrow diamonds(x) \\ \neg\varphi_1(x) \wedge \neg\varphi_2(x) \wedge \varphi_3(x) &\rightarrow clubs(x) \\ \neg\varphi_1(x) \wedge \neg\varphi_2(x) \wedge \neg\varphi_3(x) \wedge \varphi_4(x) &\rightarrow spades(x) \end{aligned}$$

and similarly for predicates in $Prd_{rank}(\mathcal{V})$.

2 Passive Learning

The notion of univocal program is what we need to illustrate the problem of identification in the limit developed in [1].

A so called game master chooses a possible game $\mathcal{G}$ which can be described by a univocal program. At each step n, the game master reveals the card $\overline{n}$, so that the learner $\mathscr{L}$ discovers each card one by one. The aim of the learner is to find, among a infinite set of univocal programs $\mathcal{H} = \{\mathcal{P}_0, \mathcal{P}_1, \dots\}$ a hypothesis $\mathcal{P}_{\mathcal{G}}$ that is equivalent to $\mathcal{G}$. So after discovering a new card $\overline{n}$, $\mathscr{L}$ proposes, if it is consistent, a program $\mathcal{P}_{\mathcal{H}} \in \mathcal{H}$ that extends the sequence $\overline{0}$, $\overline{1}$, ..., $\overline{n}$, *i.e.* $[P_{\mathcal{H}}] \supseteq \{\wp_1(\overline{0}) \wedge \wp_2(\overline{0}) \wedge ... \wedge \wp_1(\overline{n}) \wedge \wp_2(\overline{n})\}, \wp_1 \in Prd_{suit}(\mathcal{V}), \wp_2 \in Prd_{rank}(\mathcal{V})$ until it eventually converges toward a correct hypothesis. Note that this kind of learning belongs to the paradigm of function learning where presenting positive data is equivalent to presenting both positive and negative data since the latter can be retrieved from the former.

Proposition 1. *The class of univocal programs is identifiable in the limit.*

This is consequence from Property 2 which allows to apply the usual enumeration technique.

Every incorrect hypothesis is refutable in the limit, so each step of the game might unvalidate $P_{\mathcal{H}}$. On the other hand, at no step in the game can $\mathscr{L}$ have a proof that its current hypothesis is correct. An acceptable strategy for this game would then be to arbitrarily order the set of hypotheses and select, every time when a new hypothesis is needed, the next one which is consistent with the current knowledge concerning $\mathcal{G}$.

We now provide an illustration of this game. Let $\mathcal{G}$ be the possible game that describes the sequence alternating the four of spades with the queen of hearts, starting with the former, until the 120^{th} card, after which the queen of spades is repeated an infinitely often. Let $\mathcal{H}$ contain the following programs:

– $\mathcal{P}_0$: every card in the sequence is the four of spades,
– $\mathcal{P}_1$: every card in the sequence is the queen of hearts,
– $\mathcal{P}_2$: the sequence alternates between the four of spades and the queen of hearts, starting with the former,
– $\mathcal{P}_3$: the sequence starts with 60 repetitions of the subsequence "four of spades, queen of hearts", then repeats the queen of spades infinitely often …

After the first card is revealed, $\mathscr{L}$ can propose any of the programs in $\mathcal{H}$ except P_1, for example P_0 . After the second card is revealed, P_0 no longer stands as a valid hypothesis and $\mathscr{L}$ has to *change his mind*, then proposing either P_2 or P_3, for example P_2. The identification process is said to be successful if $\mathscr{L}$ changes hypotheses finitely often.

3 Active Learning

We illustrated passive learning with a game in which a learner has to exactly identify a univocal program describing an infinite sequence which is revealed to it one card after another. We shall now illustrate active learning with a classification game in which the learner has to exactly identify an equivocal program by querying an oracle to test its hypotheses.

Let P_{target} be an equivocal logic program describing a set $\mathcal{W}_{target} \subseteq \mathcal{W}$ of possible games sharing certain properties, $\mathcal{W}$ being the set of all possible games, and let $\mathcal{H}$ be a possibly infinite set of equivocal programs.

At each step, $\mathscr{L}$ is allowed to make a query to an oracle among the types of queries introduced and studied in [2,8]:

– *Membership*: the input is a possible game $X \in \mathcal{W}$, and the answer is true if $X \in \mathcal{W}_{target}$, or false if X is a counterexample.
– *Equivalence*: the input is a set $\mathcal{W}_{\mathcal{H}} \in \mathcal{W}$ of possible games, and the answer is true if $\mathcal{W}_{\mathcal{H}} \equiv \mathcal{W}_{target}$, or a counterexample $X \in \mathcal{W}_H \Delta \mathcal{W}_{target}$.
– *Subset*: the input is a set $\mathcal{W}_{\mathcal{H}} \in \mathcal{W}$ of possible games, and the answer is true if $\mathcal{W}_{\mathcal{H}} \subseteq \mathcal{W}_{target}$ or a counterexample $X \in \mathcal{W}_{\mathcal{H}} - \mathcal{W}_{target}$.
– *Superset*: the input is a set $\mathcal{W}_{\mathcal{H}} \in \mathcal{W}$ of possible games, and the answer is true if $\mathcal{W}_{\mathcal{H}} \supseteq \mathcal{W}_{target}$, or a counterexample $X \in \mathcal{W}_{target} - \mathcal{W}_{\mathcal{H}}$.

The classification is said to be successful if after a finite number of queries and data acquisition, $\mathscr{L}$ converges toward a program $\mathcal{P}_{\mathcal{H}} \in \mathcal{H}$ such that $\mathcal{P}_{\mathcal{H}} \equiv \mathcal{P}_{target}$.

The classification paradigm models the interaction between the learner and its environment in ways that differ quite substantially from the paradigm of identification in the limit, most notably through the use of queries that allow the learner to play an active role during the learning process. Indeed, the learner does not wait for some useful information to be brought to his knowledge . He can now select the most informative piece of data determined by his current hypothesis and an exploration behavior. However, according to our general definition of possible games, all these queries are co-semi-decidable. Supposing that $\mathscr{L}$ has access to an oracle able to answer equivalence, subset, and superset queries presupposed that this oracle has computational capabilities greater than a Turing machine. Hence membership queries are not really relevant, since we cannot decide whether an equivocal program is actually univocal, namely, describes a unique possible game. This limitation reflects the type of difficulties encountered in the context of scientific discovery in which any object has a potentially infinite representation. Scientists lead finite experiments to gather information, and generalize observations to formulate theories whose proofs, according to [4], are out of reach. In [9], we illustrated with the game *Eleusis+Nobel* a possible restriction of membership queries to finite possible games. We also bypassed the problem of answering equivalence queries by distributing it on the community of players, in a social game of publication and refutation of conjectures. Encouraging results concerning the epistemological and didactic aspects of this game were presented in [5]. We now formally reformulate this game in section 4, introduce the use of subset and superset queries, and discuss the relevance of the distributed resolution of queries.

4 Interactive Learning and Scientific Discovery

Let us illustrate why Gold's paradigm is not suited to the context of scientific discovery. This paradigm describes a scientist who observe his environment and try to understand it, but who is unable to move. His perception of his environment is then limited to a unique point of view, and is constrained by the occurence of the different phenomena. On the contrary, active learning could intuitively describe the scientific method consisting in selecting the most informant piece of data, and the proof of the theory built on top of this information: a scientist design experimentations, which are limited in space and time, to reproduce and study a given phenomenon. The theory built on top of the gathered data is then published and submitted to a pair validation. We now propose to include this broad view of Science in a game that could illustrate how we *learn from each other*.

4.1 Reformulation of *Eleusis+Nobel* and Discussion

Players are allowed to make *experiments* by asking the game master if a particular subsequence X is consistent with $\mathcal{P}_{target}$ or not. A positive result does not guarantee that X is not an initial segment of a possible game outside $\mathcal{W}_{target}$, but a negative result guarantees that no possible game can consistently be created starting from X and therefore invalidates all hypotheses consistent with X. Instead of supposing the existence of an oracle such as the one defined for active learning, we allow players to *publish* conjectures *i.e.*, to ask the community to refute them. We define three types of conjectures according to the definitions of these three types of queries, all of which are refutable by exhibiting a counterexample. Publishing a conjecture does not guarantee any answer. In fact, a positive answer will usually not be known to be correct, but a counterexample will eventually be found if one exists. As with identification in the limit, the learner will never have any guarantee that a hypothesis is correct, but if a hypothesis is incorrect then a counterexample will occur at some point in the game.

The main difference with identification in the limit is then the use of experiments, which allow learners to explore the set of finite sequences at their will, on the way using exploration strategies, heuristics, and background knowledge. To ensure that uncorrect conjectures will be refuted, we create a social dynamic as usually done in game theory since [10]. Note that several target programs are made available so that learners can *organize* their experimentation time and switch from one problem to another. We then define a gain function such that each player has to optimize his or her personal gain either by publishing new conjectures, or by trying to refute existing ones. For instance, according to the gain function used in [9], players win $P = 1$ point for publishing a conjecture, and $R = 2$ points for finding a counterexample to such a query, loosing the same amount of points when one of their own conjectures is refuted. Note that only equivalence conjectures were implemented for our experimentations. Despite its simplicity, this gain function allows one to observe three different behaviors when humans play: altruists publish often, regardless of refutation risks, opportunists never publish and only try to refute others conjectures when they are published, and careful players seem to define a reasonable experimentation length before deciding whether a conjecture (theirs or another's) is true. Intuitively, the latter is the richest behavior in terms of strategy and risk management, but the former behavior ensures a constant flow between published and refuted theories. This minimal interaction is then sufficient to create a process of classification in the limit. This game can be adapted to a more popperian conception of scientific discovery by introducing the use of subset and superset queries as described in the following section.

4.2 Extensions

According to Popper [4], theories should be evaluated following their capacity to resist refutation. Several theories could then cohabit at the same time, even if they are contradictory, until some of them get refuted. How do we cater for a large range of theories in our framework? A natural generalization of the notion of equivocal programs is obtained by taking boolean combinations of equivocal programs. We can illustrate the elimination of inconsistent hypotheses as follows. The learner's initial hypothesis can be formulated as the disjunction $\bigvee_{\mathcal{P}\in\mathcal{H}}\mathcal{P}$ of all programs in $\mathcal{H}$, since no data is available to refute any of them when the game begins. After each experiment, the learner can then remove every inconsistent program from the disjunction and eventually converge towards a correct one or towards one that deserves publication. By allowing the use of subset and superset conjectures, we can also define publications as boolean combinations of possibly contradictory conjectures, whereas restricting publications to equivalence conjectures implied refuting the existing one before publishing any alternative. The gain function should then be adapted to reflect that the longer a theory remains unrefuted, the more credit it gets. For instance, instead of earning P points only when they publish a conjecture, learners could earn $f(P)$ points for each effective experiment that does not refute the conjecture. Refuting a conjecture would then be rewarded according to the credit it awarded before it was refuted.

Providing results concerning this extension will be the object of future work, and the formalization we presented was necessary to clearly define the new features that need to be implemented. As we noticed, this multi-agent declination of learning paradigms was inspired by epistemological considerations. This interaction protocol is suited to human interaction as our experimentations pointed out, and to the human-machine interaction which was shown to be necessary in order to assist scientific discovery with learning machines as introduced by [11] and as discussed in depth in [12]. Indeed, a crucial point in scientific discovery is the definition of new concepts , as pointed out in [13], [14] or [15] . This is formalized in the form of the *theoretical predicates* introduced in Definition 2. The vocabulary used to formulate conjectures can then be extended with new concepts provided that there are rules which permit to derive them from already defined concepts. The current interface to create publications, which already provides every concept used in the target rules, can then be adapted to let players define their own concepts, and use them to easily formulate their conjectures. Definition 2 ensures that both assistants and other players will understand the concepts and will be able to test and eventually refute any incorrect conjecture.

5 What about *PAC*-Learning?

In the case of passive learning, a distribution law can determine the probability of observing each piece of data. [3] introduces a notion of approximation to reflect the fact that learning might not be exact, either because of noise in the data, or simply because exact identification is not required. A success criteria for a *Probably Approximatively Correct* learning is then defined, that relies on these notions of distribution and approximation, and on a reliability criterion. Essentially, *PAC*-learning involves arbitrarily bounded notions, whereas the models of reality we defined in section 1 are infinite. For this reason, our formalization does not seem adapted to the PAC paradigm. However, it is worth describing what restrictions we need to impose to our formalization in order to illustrate *PAC*-learning, since it is widely used in the community.

Consider the canonical probability distribution $\mathcal{D}$ on $\mathcal{W}$, where for every closed atom φ_1 built on a predicate symbol in $Prd_{suit}(\mathcal{V})$, and every closed atom φ_2 built on a predicate symbol in $Prd_{suit}(\mathcal{V})$, $p(\varphi_1) = \frac{1}{4}$ and $p(\varphi_2) = \frac{1}{13}$. For every card $n \in \mathbb{N}$, $p(\varphi_1(\overline{n}) \wedge \varphi_2(\overline{n})) = \frac{1}{52}$, which provides the basis for a probability distribution on $\mathcal{W}$, similar to the usual probability measure on the Cantor space, except that we are dealing with a 52-branching tree rather than a 2-branching tree. This then allows one to compute, for any pair $(\mathcal{P}_1, \mathcal{P}_2)$ of equivocal logic programs, the probability that a possible game be described by $\mathcal{P}_1$ but not by $\mathcal{P}_2$, or that it be described by $\mathcal{P}_2$ but not by $\mathcal{P}_1$. In other words, this distribution allows one to determine a similarity degree between two equivocal logic programs.

In Valiant's formalization, finite examples are points in an n-dimensional space. Concepts are finite sets of such points, and examples are classified as belonging or not to a target concept. Restricting our formalization to bounded notions, a natural class of concepts is any set of initial segments of possible games of arbitrary length n. We can then consider a finite set X of card sequences of length n, randomly chosen with respect to $\mathcal{D}$, as our example set. If we define the size $\mathcal{C}$ of $\mathcal{P}_{target}$ as a function of the number of formulas in $\mathcal{P}_{target}$ and of the maximum size of these formulas, we now have all the ingredients needed to work in a PAC-like paradigm. First, we randomly choose a natural number n, then for each $i < n$, we randomly choose $\wp_1(\bar{i})$ and $\wp_2(\bar{i})$ according to $\mathcal{D}$, with $\wp_1 \in Prd_{suit}$ and $\wp_2 \in Prd_{rank}$, therefore yielding a set S of formulas describing these initial segments. The aim of the learner is to output a possibly equivocal logic program $\mathcal{P}_{\mathcal{H}}$ which almost every $x \in X$ is a model of, *i.e.*, which is a good enough approximation of W_{target}, still with respect to $\mathcal{D}$, as defined next. Set

$$\text{error}(h) = \sum_{x \in \mathcal{W}_{\mathcal{H}} \varDelta \mathcal{W}_{target}} D(x),$$

where $W_{\mathcal{H}} \varDelta \mathcal{W}_{target}$ represents the symmetric difference between the learner's classification and the target classification. Let ε be the approximation criterion, and δ the reliability criterion. The classification is considered successful if the learner proposes in time t and with probability $\mu \geq 1 - \delta$ a program $P_{\mathcal{H}}$ having an error rate $error(\mathcal{P}_{\mathcal{H}}) < \varepsilon$, and such that t is polynomial in the number of examples, the size of the target program, ε, and δ: $t = p(n, \mathcal{C}, \frac{1}{\varepsilon}, \frac{1}{\delta})$.

Conclusion

We studied the extent to which common machine learning paradigms are relevant to the scientific method which requires the revision of description models from the time they appear to be inconsistent or incomplete with respect to the set of observations. To this aim, we proposed a logical formalization and used it to point out the crucial differences between learning paradigms.

By using the notion of *univocal program*, we pointed out that identification in the limit describes an infinite learning process comparable to human learning. However, this paradigm provides the learner with a *passive* role which does not not allow him to use exploration strategies and heuristics.

With the introduction of an oracle and queries, [2] greatly modifies the assumptions concerning the learner who becomes an *actor* of his own learning. However, using the notion of *equivocal programs*, we emphasized the limitations of this paradigm as the computational capabilities required from the oracle are difficult to simulate in the context of scientific discovery.

We then reformulated in the proposed formalism a restriction of membership queries to finite experimentations as well as a multi-agent distribution of equivalence queries, inspired by game theory. In a popperian conception of scientific method, this interactive learning protocol places experimental proof out of reach of learners, and describes the refutation in the limit of uncorrect conjectures and a social accreditation of those that resist refutation. A preliminary prototype of this learning protocol has been implemented and has provided interesting results [5]. The formulation of the learning protocol as it has been described in this paper suggests several extensions that reflect on the formation of scientific theories. They will be the object of future work.

References

1. Gold, E.M.: Language identification in the limit. Information and Control 10(5), 447–474 (1967)
2. Angluin, D.: Queries and concept learning. Machine Learning 2, 319–342 (1988)
3. Valiant, L.: A theory of the learnable. Commun. ACM 27, 1134–1142 (1984)
4. Popper, K.R.: Conjectures and Refutations: The Growth of Scientific Knowledge. Harper and Row (1963)
5. Hagège, H., Dartnell, C., Sallantin, J.: Positivism against constructivism: A network game to learn epistemology. In: Discovery Science (2007)
6. Kolmogorov, A.N.: Grundbegriffe der Wahrscheinlichkeitrechnung (Fondements de la thorie des probabilits). Springer, Heidelberg (1930)
7. Chaumon, L., Mazliak, L., Yor, M.: A.n. kolmogorov, quelques aspects de l'oeuvre probabiliste (2003)
8. Angluin, D.: Queries revisited. Theoretical Computer Science 313, 175–194 (2004)
9. Dartnell, C., Sallantin, J.: Assisting scientific discovery with an adaptive problem solver. In: Discovery Science, pp. 99–112 (2005)
10. Neumann, J.V., Morgenstern, O.: Theory of Games and Economic Behavior. Princeton University Press, Princeton (1944)
11. Sallantin, J.: La découverte scientifique assistée par des agents rationnels. Revue des sciences et technologie de l'information, 15–30 (2003)
12. Nobrega, G.M.D., Cerri, S.: A contradiction driven approach to theory information: Conceptual issues pragmatics in human learning, potentialities. Journal of the Brazilian Computer Society 9, 37–55 (2003)
13. Muggleton, S., Buntine, W.: Machine invention of firstorder predicates by inverting resolution. In: Fifth International Conference on Machine Learning, pp. 339–352 (1988)
14. Muggleton, S.: Predicate invention and utilisation. Theoretical Artificial Intelligence, 127–130 (1994)
15. Banerji, R.B.: Learning theoretical terms. In: Muggleton, S. (ed.) Inductive Logic Programming, pp. 93–112 (1997)

Comparative Evaluation of Two Systems for the Visual Navigation of Encyclopedia Knowledge Spaces

Wolfgang Kienreich and Peter Kraker

Know-Center
Competence Center for Knowledge-Based Applications and Systems,
Inffeldgasse 21a, 8010 Graz, Austria
{wkien,pkraker}@know-center.at

Abstract. Modern digital encyclopedias contain hundreds of thousands of textual articles and multimedia elements. Alternative discovery techniques facilitating the visual exploration of encyclopedia knowledge spaces have recently received much attention. We present the results of a comparative usability evaluation of a two-dimensional and a three-dimensional visualization system integrated with the Brockhaus encyclopedia. Both systems enable visual navigation of article context. Results indicate that both systems perform comparably and that users prefer the threedimensional visualization system.

Keywords: Knowledge Visualization, Digital Encyclopedias, Visual Navigation, User Interfaces, Human-Computer Interaction, Formal Experiments.

1 Introduction

Modern digital encyclopedias contain hundreds of thousands of textual articles and multimedia elements which constitute a knowledge space encompassing virtually all areas of general interest. Traditional retrieval and discovery techniques in this domain have included keyword search for articles and cross-reference based navigation between articles. Alternative discovery techniques facilitating the visual exploration of encyclopedia knowledge spaces have recently received much attention. Such techniques have for example leveraged advances in knowledge visualization [1], geospatial information systems [2] and semantic technologies.

The German-language Brockhaus encyclopedia [3] provides two visualization systems enabling the visual navigation of article context. The two-dimensional "Knowledge Web" visualization presents topically related articles using abstract graphical elements. The three-dimensional "Knowledge Space" visualization presents topically related articles using figurative graphical elements as visual metaphors. The manufacturer expects the two systems to support navigation between articles and to encourage an exploratory approach to the encyclopedia. We present the results of a comparative usability evaluation of the two systems. The objective of the evaluation was to determine which of the two systems better meets the stated expectation.

2 Environment

This section introduces the application domain and outlines the scientific fields relevant for the work described in this paper. First, the domain of digital encyclopedias is presented. Then, an introduction to Knowledge Visualization and Usability Evaluation is given. These preliminaries serve to put the following discussion of systems, experiments and results into context.

2.1 Digital Encyclopedia

An encyclopedia is commonly defined as a compendium of knowledge consisting of articles about topics of general interest. Most general encyclopedias are primarily structured using an alphabetical index based on the article titles ("lemmas"). Modern digital encyclopedias supplement textual article content with images, audio and video clips and other multimedia elements. They provide search facilities based on keyword queries and navigation facilities based on cross-article references.

The "Brockhaus Enzyklopaedie" [3] is an example of a modern multimedia encyclopedia. This German language encyclopedia contains 240,000 articles, 30,000 multimedia elements and 350,000 keywords. It is the predominant encyclopedia in the German language domain (the famous "Duden" lexicon is published by the same company) and ranks among the world's largest encyclopedias. The Brockhaus encyclopedia features two novel systems for the visual navigation between articles based on an automatically generated article context (compare section 3).

2.2 Knowledge Visualization

In the context of computer science, the term visualization usually denotes the use of visual representations in support of quantitative or qualitative analysis of information. Modern visualization systems rely on real-time computer graphics to display interactive representations of complex information spaces.

Information Visualization is commonly defined as the use of computer-supported, interactive, visual representations of abstract data to amplify cognition. Work in the field leverages the fact that the human visual sense is capable of rapid perception, transformation and classification of information. Information Visualization is characterized by a clear focus on computer-supported media, abstract information, and individual information seeking in very large information spaces. [4]

Knowledge Visualization strives to complement and expand the focus of Information Visualization by concentrating on the process of knowledge transfer. It proposes to consider the intended function of a system, the knowledge type to be transferred, the characteristics of the recipients and the visualization types available when designing a visualization system [5]. Knowledge Visualization emphasizes the importance of balancing the use of visual models, metaphors and abstractions in order to optimize the transfer and perception of knowledge [6].

2.3 Usability Testing

Usability testing is commonly defined as the empirical testing of interface design with representative user groups. Various forms of usability tests have been proposed [7]:

Thinking aloud tests encourage users to verbalize thoughts while completing tasks. Usage studies observe the behavior of users in a natural environment. Formal experiments provide quantitative data by measuring performance under controlled conditions. All forms of usability testing share certain common properties [8]: The primary goal of any usability test is to improve usability. The composition of the test user group and the nature of the assigned tasks reflect the expected real-world situation. The test user's actions and expressions are observed, recorded and analyzed after the test to discover problems and suggest solutions.

There has been much debate about the number of users required to produce valid results in a usability test. Some studies suggest that five users should suffice [9]. Others have challenged this view. Several usability tests have yielded relevant results with user groups containing between five and ten individuals (e.g. [10]).

Comparative usability testing applies near-identical test conditions to two or more systems and analyzes test results to determine benefits and shortcomings of each tested system. The work presented in this paper is based on a formal experiment comparing two visualization systems which facilitate the navigation of encyclopedia articles. The experiment has been conducted with a group of six test users.

3 System

This section introduces relevant aspects of the Brockhaus Encyclopedia system. First, the context visualization framework integrated in the Brockhaus encyclopedia is presented. Then, the visualizations subjected to evaluation are described. Further information on the Brockhaus Encyclopedia is available from the manufacturer [3].

3.1 Context Visualization

The Brockhaus Encyclopedia provides a relevance-ranked list of related articles for each article. This list is commonly referred to as the article context. The context is precomputed based on a vector-space model [11] of article content. This model facilitates similarity search using terms identified as relevant by tf/idf weighting [12]. The results of the similarity search are filtered by a set of rules defined by domain experts and limited to at most 25 articles.

The encyclopedia window provides a text-based query component and an article display. It also contains links to a two-dimensional context visualization denoted "Knowledge Web" and to a three-dimensional context visualization denoted "Knowledge Space". Both visualizations operate in separate windows and are synchronized with the encyclopedia window. They share several features and interaction patterns. For example, clicking on an article symbol navigates to the article and an option to keep the visualization window on top of the encyclopedia window is present in both visualizations.

3.2 Knowledge Web Visualization

The two-dimensional „Knowledge Web" Visualization places the context origin in the center of the display and arranges context articles around it in a web-like manner.

Relevant articles are placed closer to the center. Article labels never overlap each other. Articles are represented by filled circles. The context origin is symbolized by a large white-shaded circle. Context articles are symbolized by smaller circles. Yellow fill color represents an article having an arbitrary topic. White fill color represents an article having the same topic as the context origin. A smaller, red circle within the circle symbolizing an article denotes the presence of multimedia content. Biographical articles are denoted by a label "P" inside the circle. Context articles are connected to the context origin by a radial line. Moving the mouse over a circle displays an article abstract. Clicking on a circle navigates to the appropriate article.

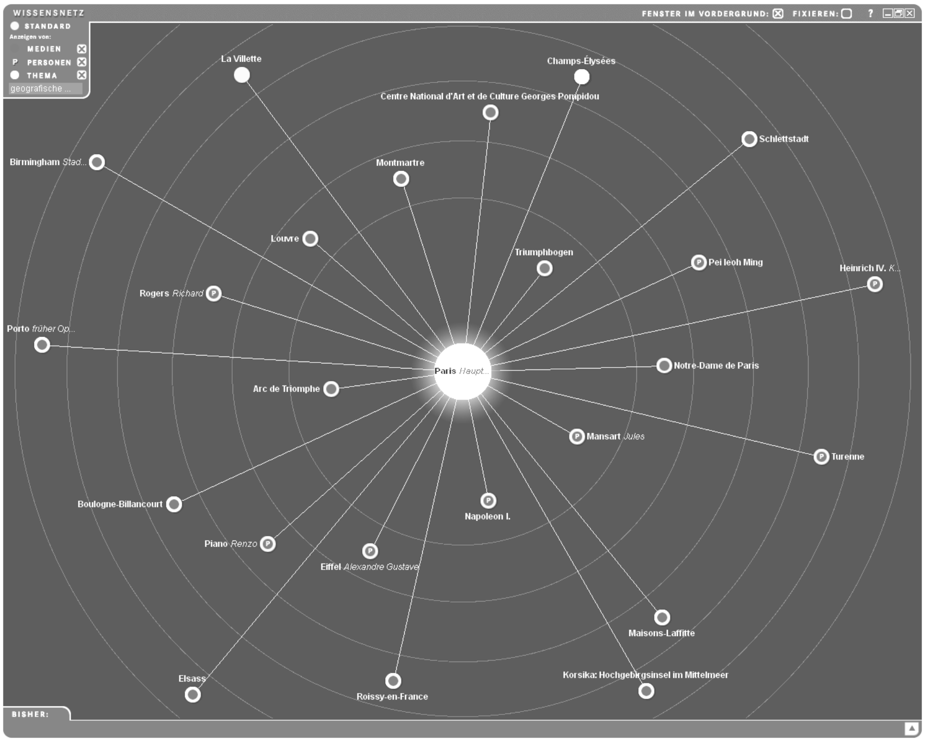


Fig. 1. "Knowledge Web" visualization: Context articles *(small icons)* arranged around the context origin *(large central disc)*

This visualization is an example for the use of visual formalisms [11]. Diagrammatic graphical elements (e.g. circles, lines) have been employed to generate an abstract visual representation. The designer has determined the semantics of the graphical elements based on requirements implied by the structure of article context. Visual formalisms can be devised for arbitrary complex knowledge spaces. However, a significant learning effort is required in order to interpret a visual formalism correctly.

3.3 Knowledge Space Visualization

The three-dimensional „Knowledge Space“ Visualization places the context origin in the center of a disc divided into topical segments and arranges context articles around it. Relevant articles are placed close to the center and each article is placed within the segment corresponding to its topic. Articles are represented by shapes according to type: Cylinders represent factual articles, spheres represent geographic articles, cones represent biographic articles and diamonds represent articles featuring premium content. Article labels are displayed above the shapes and never overlap each other. Dragging the mouse horizontally spins the disc around its central axis. Dragging the mouse vertically adjusts zoom factor and vertical view angle. Clicking on an object navigates to the appropriate article. Further details are provided in [1].

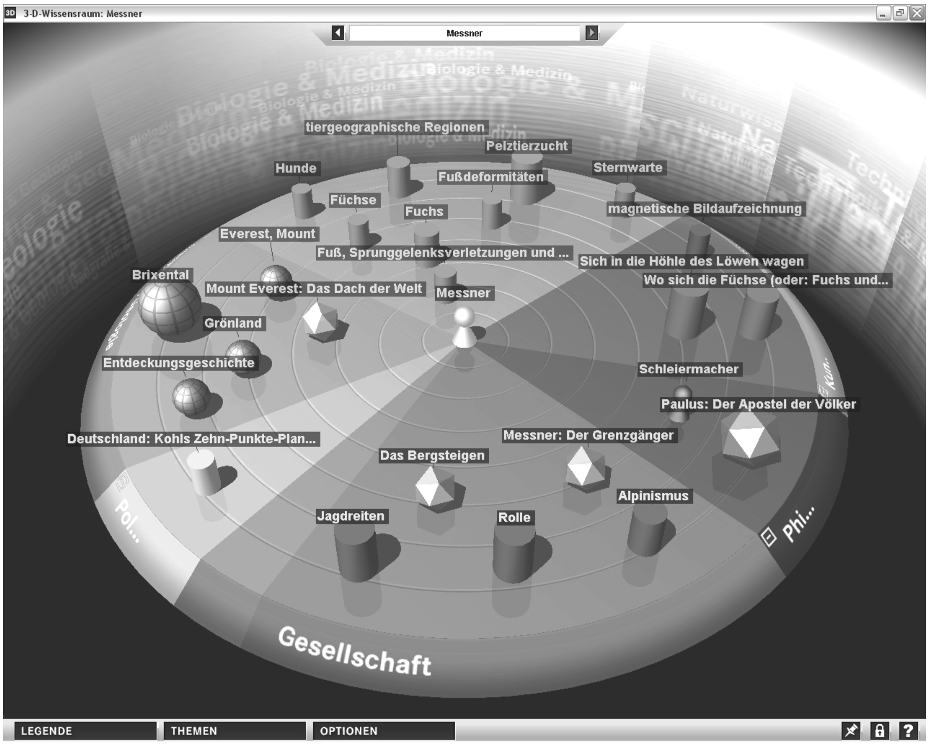


Fig. 2. "Knowledge Space" visualization: Context articles *(shapes on top of disc)* arranged around the context origin *(central small cone shape)* in segments *(disc slices)* denoting topics

This visualization is an example for the use of visual metaphors. The employed graphical elements have been derived from real-world equivalents (e.g. board game, turntable) which implicitly determine semantics. The designer has combined and adapted the graphical elements to match the structure of the article context. Visual metaphors have to be chosen carefully to convey relevant information and avoid misinterpretation [14]. However, analogies to the real-world equivalent enable intuitive interpretation and comprehension of a well-designed visual metaphor.

4 Evaluation

This section reports on the comparative evaluation of the visualization systems previously described. First, the type of experiment executed is introduced. Then, the evaluation design is outlined, the execution of the evaluation is described and the evaluation results are presented. Finally, an interpretation of the evaluation results is attempted in the context of related findings.

4.1 Formal Experiments

A formal experiment is a specific type of usability evaluation which is executed in a controlled environment and provides objective, quantitative data by measuring performance. Common performance indicators include the time required to finish a task, the number of tasks finished within a given period or the number of errors that occurred [15]. Because formal experiments do not explain the cause for the occurrence of observed phenomena, they are often combined with questionnaires, interviews and other means of gathering qualitative feedback.

Formal experiments can be adapted to compare two or more alternative interface designs. A comparative experiment can be structured according to one of the following designs: Between-groups experiments assign identical tasks to all users, and each user gets to work with one of the interfaces being tested. Within-groups experiments assign equivalent sets of tasks to users and users are then randomly assigned to different pools. Each pool is confronted with all of the interfaces in different order and with different sets of tasks. While the between-groups design avoids undesired learning effects, it does not account for variations in user skill levels. Such variations are controlled by the within-groups design.

4.2 Experiment Design

A formal experiment was defined to compare the performance of the two visualization systems presented. The objective of the experiment was to determine which of the two systems is better suited to support analysis and navigation of the article context.

A between-groups design was chosen for the experiment to eliminate the effects of varying user skills. Two equivalent sets of tasks (denoted T1 and T2) were designed. There were four categories of tasks (denoted A1 to A4) with increasing level of difficulty: Counting related articles of a biographical nature, finding articles belonging to a topical category, navigating to a specified item across one level of hierarchy and navigating to an item specified by contextual hints across two levels of hierarchy. For each category, two equivalent tasks were defined.

Six test users (denoted U1 to U6) were recruited. The selection of users was based on characteristics of real-world knowledge workers likely to utilize advanced encyclopedia features (compare table 1). The test user group included Psychologists, Designers, Engineers and Researchers with a medium to high level of education.

Table 1. Some characteristics of test users

	U1	U2	U3	U4	U5	U6
Age	32	44	27	28	30	32
Sex	f	f	f	f	m	m
Sight Aid	yes	no	yes	no	no	yes
Computer Experience (years)	12	8	13	10	12	18
Computer Usage (h/week)	35	30-40	50	50-60	60	60-70
Encyclopedia Usage (h/week)	2	1	8	1-2	1	2
Visualization Usage (h/week)	0	0	30	3	0	2

Test users were randomly assigned to one out of two test groups. The first group (denoted G1) worked on a first set of tasks in condition C1 (denoting that the two-dimensional knowledge web was used) and continued with a second set of tasks in condition 3D (using the three-dimensional knowledge space). The second group (denoted G2) worked on a first set of tasks in condition C2 and continued with a second set of tasks in condition C1. To account for differences between the two task sets, the sets were alternated between conditions from test to test (compare table 2).

Table 2. Permutation of users, conditions and tasks

Group	User	Condition	Task Set
G1	U1	C1-C2	T1-T2
G1	U2	C1-C2	T2-T1
G1	U3	C1-C2	T1-T2
G2	U4	C2-C1	T2-T1
G2	U5	C2-C1	T1-T2
G2	U6	C2-C1	T2-T1

To supplement the quantitative information gathered in the formal experiment, a questionnaire was designed. The questionnaire contained ten questions (denoted Q1 to Q10) which were to be answered using a seven-point polarity profile (compare table 3). Test users were asked to fill in the questionnaire for both conditions (C1, C2) immediately after participating in the experiment.

Table 3. Questionnaire structure

Index	Question	Scale (+3 to -3)
Q1	Ease of navigating to a specified article	easy - hard
Q2	Ease of locating articles belonging to a topic	easy - hard
Q3	Readability of texts	high - low
Q4	Quality of available help	good - bad
Q5	Relevance of information in article context	high – low
Q6	Design quality of visualization	high – low
Q7	Animation quality of visualization	high – low
Q8	Interaction speed of visualization	high - low
Q9	Overall impression of system	good - bad
Q10	Would you actually use the system	likely - unlikely

4.3 Experiment Execution

The experiment was executed in an office room shielded from external influences. The experimenter conducted each session following a script which outlined welcome procedures, administrative issues, execution of training and test and filling of the questionnaire. The validity of this script was verified by a test run some days before the experiment was executed.

The system features available in each condition (C1, C2) were presented by the experimenter. For both conditions, the location of the current article and of the related articles was pointed out. In condition C1 (the two-dimensional Knowledge Web), the graphical encoding of the related articles was explained. In condition C2 (the three-dimensional Knowledge Space), the segmentation of articles by topic and the meaning of article symbols was explained. In both conditions, the various means of interaction were presented. Users were then given time to familiarize themselves with the system. After users signaled their readiness to proceed, the system was reset, recording was started and the first task was assigned.

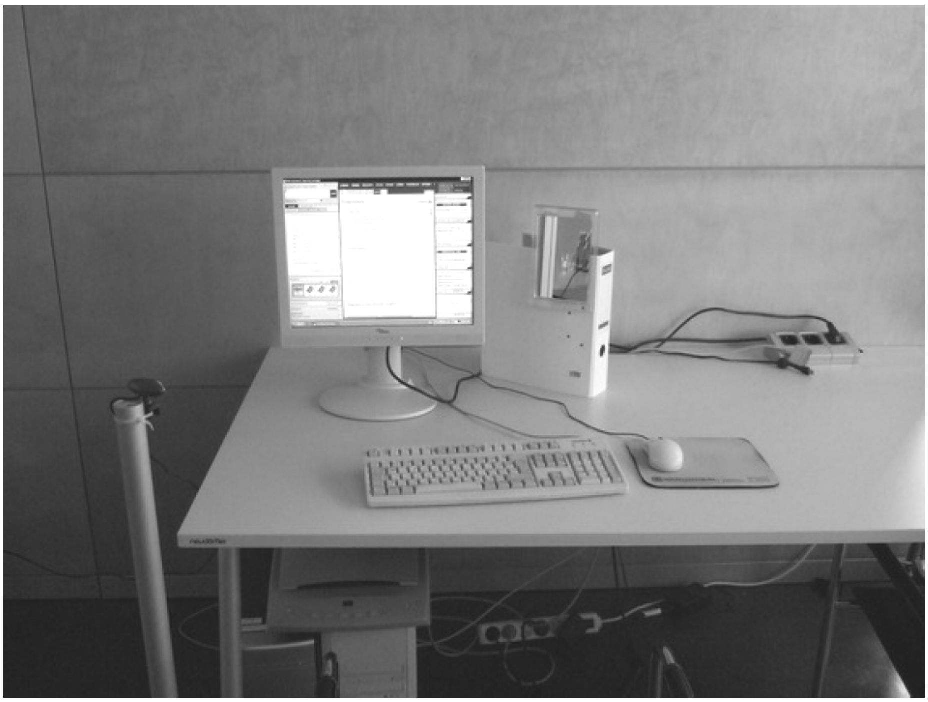


Fig. 3. Physical setup of the experiment: Camera and microphone (*left*), TFT display (*center*), mirror for capturing facial expressions (*right of display*), input devices (*in front of display*)

The experiment was conducted using a personal computer (Pentium 4, 2.0 GHz, 2GB RAM, Windows XP SP2), a TFT display and a standard keyboard and mouse. Display resolution was set to 1024 by 768 pixels. A custom build of the Brockhaus digital encyclopedia in version 1.01 was used.

4.4 Experiment Results

The results of the experiment were analyzed by processing the recorded video sequences and noting the time required by users for task completion. In several cases, users gave up on a task after some time. In other cases, users worked more than 300 seconds on a task and the experimenter suggested moving on to the next task. Such incidents were noted as timeout events and not included in the analysis.

Table 4. Time to task completion (in seconds, T denotes timeout)

Cond.	Task	U1	U2	U3	U4	U5	U6	Avg.
C1	A1	37	84	47	34	24	87	52,2
C1	A2	46	185	142	106	28	165	112,0
C1	A3	84	181	T	139	116	122	128,4
C1	A4	269	T	120	56	T	141	146,5
C2	A1	21	30	34	37	61	54	39,5
C2	A2	62	107	18	46	30	225	81,5
C2	A3	278	295	90	127	T	T	197,5
C2	A4	T	119	177	236	147	T	169,7

In both conditions (C1,C2), users required significantly more time to complete complex navigation tasks (A3,A4) as compared to simple recognition tasks (A1,A2). All of the timeout events occurred in complex navigation tasks (A3,A4). It must be noted that a paired t-test revealed no statistically significant differences between the two conditions for any of the tasks. However, this statement is not meaningful for tasks A3 and A4 due to the number of timeouts encountered (compare table 4).

Table 5. Questionnaire results (higher values denote positive response)

Index	Question	C1(mean)	C2(mean)
Q1	Ease of navigating to a specified article	+1.33	+2.17
Q2	Ease of locating articles belonging to a topic	-0.67	+2.50
Q3	Readability of texts	+2.67	+1.93
Q4	Quality of available help	-	-
Q5	Relevance of information in article context	+1.67	+1.93
Q6	Design quality of visualization	+0.83	+2.50
Q7	Animation quality of visualization	+0.83	+2.67
Q8	Interaction speed of visualization	+2.67	+2.67
Q9	Overall impression of system	+1.67	+2.67
Q10	Would you actually use the system	+1.67	+2.17

The analysis of the questionnaires filled after taking the test revealed significant differences between conditions. Users ranked the three-dimensional „Knowledge Space“ Visualization (C2) higher than the two-dimensional „Knowledge Web“ Visualization (C1) in all questions except Q3 (compare table 5). It must be noted that no ranking was acquired for question Q4 because the help facilities had been disabled for both conditions (C1,C2). This modification to the experimental setup was made after the design test (compare section 4.3) to avoid biasing results by time spent in studying help content.

Table 6. Problems reported by more than one user

Index	Problem	Condition
P1	Arrangement of history entries	C1
P2	Color-coding of symbols representing articles	C1
P3	Choice of symbols representing articles	C1
P4	Missing topical segmentation (as compared to C2)	C1
P5	Missing article abstract preview (as compared to C1)	C2
P6	Choice of symbol and caption for "lock" and "keep on top"	C1,C2
P7	System behavior on switching from and to Encyclopedia	C1,C2

The comments made by users in writing (on the questionnaire form) and verbally (recorded during the test session) served to refine the observations stated before. The visual appearance of the two-dimensional Knowledge Web (condition C1) and the common user interface components integrating both systems with the encyclopedia were the major source of comments (compare table 6).

4.5 Discussion of Results

The performance data gathered in the formal experiment indicates that users required more time for tasks involving navigation between article contexts (A3,A4) than for tasks involving analysis of a single article context (A1,A2). A high number of timeout events occurred in tasks involving navigation (A3,A4). This indicates a weak point in the evaluation design (e.g. tasks too complex, not enough training).

The statistical analysis of the performance data was inconclusive. No significant differences between the conditions (C1,C2) could be detected. This fact can be attributed in part to the high number of timeout events. The individual performance data shows a 32% advantage of C2 over C1 for task A1 and a 37% advantage of C2 over C1 for task A2 (no timeout events occurred in tasks A1 and A2).

The user feedback collected through the questionnaires indicates that users preferred C2 over C1 in most aspects. The topical segmentation available in C2 was particularly well liked (Q2, +53% acceptance). Users preferred design (Q6, +28% acceptance) and interactivity (Q7, +31% acceptance) of C2 over C1. The overall impression reported by users favored C2 (Q9, +17% acceptance). The presentation of text was the only aspect where C1 was ranked higher than C2 (Q3, +12% acceptance).

Comparative evaluations of two-dimensional and three-dimensional visualizations usually find that two-dimensional interfaces perform superior [16]. However, divergent evaluation results have been reported for systems providing overview and browsing functionality [17]. Findings in Knowledge Visualization indicate that for small sets of highly structured knowledge items, systems based on visual metaphors outperform systems based on visual abstractions [18]. These observations may explain why the three-dimensional "Knowledge Space" visualization (C2), which is based on the use of visual metaphors, performed comparably to (and was liked better than) the two-dimensional "Knowledge Web" visualization (C1), which is based on the use of visual abstractions, despite the higher cognitive load imposed by a three-dimensional visualization.

5 Future Work

The comparative performance data gathered in the formal experiment has proven inconclusive in the statistical analysis. This shortcoming can be attributed to the high number of timeout events and to the low number of test subjects. A second iteration of the usability evaluation will be designed to account for these factors. The number of users participating in the evaluation will be increased to 12. The evaluation design will be modified to provide an extended training phase. Navigational tasks (A3, A4) will be simplified and explanations for the tasks will be improved. Various smaller design problems (e.g. the presence of question Q4) will also be addressed. It is expected that the second iteration will provide conclusive statistical results on the comparative performance of the two conditions (C1,C2).

The presented evaluation did not measure the performance of the visualization systems in comparison with the traditional means of discovery provided by digital encyclopedias (e.g. keyword search, navigation by cross-reference). A new usability evaluation will be designed to include appropriate conditions and tasks.

The results obtained by the presented evaluation will be applied to revise the three-dimensional "Knowledge Space" visualization. The rendering of text elements will receive particular attention. The integration of the visualization systems with the encyclopedia will be redesigned to account for the issues reported by users.

6 Conclusions

The German-language Brockhaus encyclopedia provides a two-dimensional "Knowledge Web" visualization and a three-dimensional "Knowledge Space" visualization to facilitate the visual navigation of article context. We have presented the results of an evaluation which compared these two systems. The statistical analysis of performance data gathered in a formal experiment did not reveal significant differences in the performance of the two systems despite the higher cognitive load imposed by a three-dimensional visualization. The evaluation of user feedback gathered during the evaluation indicates that users prefer the three-dimensional „Knowledge Space“ visualization. We conclude that a three-dimensional visualization system based on visual metaphors can perform at least as well as a two-dimensional visualization system based on visual abstractions for certain application domains and tasks.

Acknowledgements

The Know-Center is funded within the Austrian COMET Program - Competence Centers for Excellent Technologies - under the auspices of the Austrian Ministry of Transport, Innovation and Technology, the Austrian Ministry of Economics and Labor and by the State of Styria. COMET is managed by the Austrian Research Promotion Agency FFG.

References

1. Kienreich, W., Granitzer, M.: Visualising Knowledge Webs for Encyclopedia Articles. In: 9th International Conference on Information Visualization. IEEE Computer Society Press, Washington (2005)
2. Kienreich, W., Granitzer, M., Lux, M.: Geospatial Anchoring of Encyclopedia Articles. In: Proceedings 10th International Conference on Information Visualization, London (2006)
3. Brockhaus Enzyklopädie Digital, Brockhaus Multimedial, Bibliografisches Institut & F.A. Brockhaus AG, Germany, `http://www.brockhaus.de`
4. Card, S.K., Mackinlay, J.D., Shneiderman, B.: Information Visualization: Using Vision to Think. Morgan-Kaufmann, San Francisco (1999)
5. Burkhard, R.: Towards a Framework and a Model for Knowledge Visualization: Synergies Between Information and Knowledge Visualization. In: Tergan, S.O. (ed.) Knowledge and Information Visualization: Searching for Synergies, pp. 238–255. Springer, Berlin (2005)
6. Kienreich, W.: Information Visualization: an Oblique View. MIA Journal on Informative Modelling 0(1), 7–17 (2006)
7. Bailey, R.W.: Human Performance Engineering. Prentice Hall, Upper Saddle River (1996)
8. Dumas, J.S., Redish, J.C.: A Practical Guide to Usability Testing. Intellect, Exeter (1999)
9. Virzi, R.A.: Refining the Test Phase of Usability Evaluation: How Many Subjects is Enough? Human Factors 34(4), 457–468 (1992)
10. Granitzer, M., Kienreich, W., Sabol, V., Andrews, K., Klieber, W.: Evaluating a System for Interactive Exploration of Large, Hierarchically Structured Document Repositories. In: 10th IEEE Symposium on Information Visualization, pp. 127–134. IEEE Computer Society Press, Washington (2004)
11. Salton, G., Wong, A., Yang, C.S.: A Vector Space Model for Automatic Indexing. Communications of the ACM 18(11), 613–620 (1975)
12. Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Information Processing & Management 24(5), 513–523 (1988)
13. Nardi, B.A., Zarmer, C.L.: Beyond Models and Metaphors: Visual Formalisms in User Interfacedesign. In: Twenty-Fourth Annual Hawaii International Conference on System Sciences, pp. 47–493. IEEE Computer Society Press, Washington (1991)
14. Eibl, M., Mandl, T., Stempfhuber, M.: Metaphors vs. Visual Formalisms in Visual Information Seeking. In: Panhellenic Conference on Human-Computer Interaction (PC-HCI 2001), Typorama (2001)
15. Helander, M.G., Landauer, T.K., Prabhu, P.V. (eds.): Handbook of Human-Computer Interaction. Elsevier, Amsterdam (1997)
16. Sebrechts, M.M., Vasilakis, J., Miller, M.S., Cugini, J.V., Laskowski, S.J.: Visualization of Search Results: A Comparative Evaluation of Text, 2D, and 3D Interfaces. In: 22nd Annual Int'l ACM-SIGIR Conf. Research and Development in Information Retrieval. ACM Press, NewYork (1999)
17. Perez, C., de Antonio, A.: 3D Visualization of Text Collections: an Experimental Study to Assess the Usefulness of 3D. In: Eighth International Conference on Information Visualisation, pp. 317–323. IEEE Computer Society Press, Washington (2004)
18. Burkhard, R., Meier, M.: Tube Map Visualization: Evaluation of a Novel Knowledge Visualization Application for the Transfer of Knowledge in Long-Term Projects. Journal of Universal Computer Science 11, 473–494 (2005)

A Framework for Knowledge Discovery in a Society of Agents

Gauvain Bourgne[1] and Vincent Corruble[2]

[1] LAMSADE, Université Paris-Dauphine
Paris 75775 Cedex 16 (France)
`bourgne@lamsade.dauphine.fr`
[2] LIP6, Univ. Pierre and Marie Curie
104, Avenue du Prsident Kennedy - 75016 - Paris (France)
`Vincent.Corruble@lip6.fr`

Abstract. This paper proposes initial steps towards a generic framework for modeling the scientific process. It is generic according to two main axes. First, it can be instantiated to cover various types of inferences usually considered relevant in science, second, and more important here, it allows for the modeling of the social dimension of scientific activity. After motivating this drive for genericity by looking at some results from philosophy of science, the paper presents the bases of the framework and its central reliance on the notion of consistency, both at the individual and group levels. It then instantiates the social dimension of the framework by proposing an actor-critic model of scientific interaction. The ideas proposed are illustrated with examples of hypothesis formation in medicine.

1 Introduction

Research in Machine Discovery has provided a rich sample of models that account well for the various forms of reasoning and inferences at work in the scientific discovery processes. Whether they focus on a problem solving perspective, where the definition of search spaces and heuristics is key, or on a logical perspective, where inferences such as induction and abduction are often central, one rather common assumption is that what is being modeled is the thought process of one scientist tackling a research problem. Indeed this is in keeping with the traditional line of research in Artificial Intelligence where one is interested in understanding or simulating intelligence seen as an individual process.

However, this view can be seen as somewhat outdated, or at least partial, when one looks at the growing literature, coming essentially from the Social Studies community, studying science as a fundamentally social process. In this paper, we describe a framework that is generic enough to model not only inferences of individual scientists but also interactions between them. Moreover, the proposed framework can also be instantiated to focus on various forms of reasoning such as induction and abduction. The outline of the paper goes as follows. First we give some background to motivate a framework that is generic in terms of inference, but more importantly, that can be used to model the social dimension of scientific activity. Then we introduce the basis of such a framework, followed by an example of instantiation into an actor-critic model. Throughout the paper, we illustrate the approach with a running example simulating a medical discussion between multiple experts tackling a problematic case.

2 Background

Views on Scientific Discovery have considerably evolved both in the history of philosophy of science and in the (much shorter) history of Artificial Intelligence. In philosophy of science, the classical view inherited from Greek philosophy was to look for rules of inferences that would lead to knowledge whose validity would be guaranteed. A major evolution, lead by individuals such as Comte, Hershel or Whewell in the 19th century and named consequentialism in [14], whose outcome was the Hypothetico-Deductive (HD) view, considered that the way a hypothesis was generated and the way it was evaluated could be completely disconnected. Indeed, a theory making good predictions (the D of HD) is considered acceptable, whatever the way it was generated. This lead then to the study of two problems, discovery per se, i.e. how new ideas, hypotheses, theories are generated, and justification, i.e. how they can be validated from a scientific standpoint. These two problems can also be considered as two stages in the discovery process, even though this is somewhat oversimplifying as the activity of a scientist usually alternates *discovery steps* and *justification steps*. The thinking over stages was refined further [5,18] with the idea of a discovery stage, where there is no evaluation, the *pursuit stage*, where hypotheses mature through a *refinement* process using some elements of *evaluation* but not necessarily in a strict sense, and justification which is the proper scientific *evaluation*. Philosophers of Science have during a long time period focused on this last stage because it was the only one deemed open to serious scientific investigation [17]. The advent of Artificial Intelligence, and more specifically, the ability to carry out complex and cognitively plausible computational simulations revived the interest in the discovery stage [19,4]. In this paper we focus indeed on the discovery and pursuit stages, which correspond to the process of hypothesis formation and refinement.

Another important evolution of modern philosophy of science concerns the focus of the study. While it makes perfect sense to study the thought process of an individual scientist, various researchers (e.g. [13] or [15]), have shown that discovery is a fundamentally social process. This idea has also impacted Computational Philosophy of Science as witnessed in publications such that [20] and [6]. The idea that, orthogonally to a multi-stage view of science, science can be structured vertically, i.e. ideas and hypotheses might come to the individual but then mature within groups of colleagues and are evaluated by communities of scientists. Furthermore, it is problematic to define precisely how these two axes (horizontal, that is, to simplify, maturation or time, and vertical, group organization from individuals to communities) combine, since one cannot assume that the community at large contributes only to evaluation. Indeed, through the organized confrontation of ideas between groups or individuals, it also participates in the pursuit and refinement of hypotheses.

It is therefore somewhat surprising to realize that the Machine Discovery has so far largely neglected this important dimension of scientific activity. The scope of processes modeled in computer programs has indeed grown richer in the last two decades since the pioneering Bacon systems [12]. As described in [9], there has been an evolution from models of science as problem solving in a hypothesis space, from models with search in two spaces (experiment space, and hypothesis space) such as in KEKADA [11] and a vision toward 4-spaces models (Experiment space, Hypothesis Space, Paradigm Space, Representation Space)[9]. But this evolution does not move away from the modeling of an individual scientist, or to be more precise, it does not mention explicitly what entity is being modeled, it is in a way de-incarnated. A way to overcome this limitation is to opt for an agent perspective, and further, in order to model the social aspect of knowledge construction, to opt for a multi-agent perspective, a scientific community being viewed as a *Society of Agents*. This opens the way towards a framework which is generic enough to study, model and simulate the inferences of individual scientists and to study, model and simulate the interactions between scientists within groups or communities so as to have a credible model of the formation of scientific ideas and hypotheses. This is the main aim of this paper to present some basic elements of such a framework, which is deeply rooted on a multi-agent perspective.

To illustrate the motivation on a concrete example, we give below an example of a dialog taken from a popular medical TV show that we will use in thereafter to address the issue of social hypothesis formation. Here we have a senior MD interacting with 3 other MDs with the goal of reaching a credible hypothesis on the cause of a patient's illness. Each interaction leads to the sharing of a relevant piece of knowledge which can then be contradicted, or possibly further refined. We wish to be able to simulate convincingly this sort of interaction.

Dr House:	This guy doesn't even get sick like a regular person. Instead of a list of symptoms and no cause, we have a list of possible causes for one symptom.
Dr Chase:	Is the symptom death?
Dr House:	Respiratory distress.[...]
Dr Forman:	It's the doping. Injecting extra red cells boosts your endurance level, but it also thickens your blood. Thick blood equals clots equals respiratory distress.
Dr House:	Not with a clean spiral chest CT.
Dr Cameron:	The guy's sleeping in a Hyperbaric Chamber. Over-oxygenation can cause cell damage,and if the cells in the lungs are damaged...
Dr House:	That'd cause pulmonary edema, which he doesn't have.
Dr Chase:	The supplements he's been taking contain yohimbe which can cause nerve paralysis.
Dr House:	Tox-screen was normal. All the tests were normal. There's no clot, no edema, and yet he still can't breathe. So there's something in here that we can't see.
Dr Forman:	Air! [...] This guy's been injecting himself how many times a day? All it'd take is one slip of the needle to cause an air embolus.
Dr House:	So, air is keeping him from breathing air. Well, let's go with that for the irony. Get a VQ scan and check his veins for bubbles.

3 Distributed Hypothesis Formation Framework

This section defines a general framework concerned with distributed formation of hypotheses [1]. This formalisation intends to encompass a variety of different hypothesis formation problems, and therefore explicit representation of knowledge is not specified. However, it is supposed that there exists some different kinds of knowledge in the system that are represented in some way. To refer to these and to the relations between them, we will use *abstract concepts*, that will have to be instantiated when dealing with specific problems. We shall see that these astract concepts are sufficient to develop some useful properties to describe a system, and, in the next section, that some general mechanisms can be proposed for distributed formation of hypothesis on the basis of these notions. We will first give our characterisation of the different kinds of knowledge and their possible representation, before introducing the notion of consistency relation, that will be detailed at different levels. Then, we will see how we can abstract away from the specifics of the reasoning used for different kinds of discovery process.

3.1 Knowledge Classification and Distribution

The first concern is to characterize the knowledge of the agents. We first give some introductory illustration using the informal example previously described.

Example 1. Let us try to formalize Forman's proposition, that is "It's the doping. Injecting extra red cells boosts your endurance level but it also thickens your blood. Thick blood equals clots equals respiratory distress.". We could get something like: (1) `Injection(RedCells)`, (2) `Injection(RedCells)` $\rightarrow$ `ThickBlood`, (3) `ThickBlood` $\wedge$ `ClotFormation(X)` $\rightarrow$ `Clot(X)`, (4) `ClotFormation(Chest)`, (5) `Clot(Chest)` $\rightarrow$ `RespiratoryDistress`.
We have different kinds of elements:

- Factual information about the patient, or *observations*: (1). In ideal case, these should be certain.
- General medical *theory*: (2),(3),(5). When making a diagnosis, it would usually also be considered certain and common to all member of the team.
- Assumed information, or *hypotheses*, that could account for observations : (4). This one is *revisable*, since new observations could prove it wrong.

Then we have the answer of Dr House : "Not with a clean spiral chest CT.", that can be described by: (6) `Clot(X)` $\rightarrow \neg$ `CT(X,Clean)`, (7) `CT(Chest,Clean)`. Formula (6) is again part of medical theory, whereas `Scanner(Chest,Normal)` is a *counter example*, that is an observation that contradicts the hypothesis `ClotFormation(Chest)` proposed by Dr Forman.

Intuitively, there seems to be some different kinds of knowledge according to their certain or revisable character, and factuality (general theory or patient data). In a distributed setting, one should also consider whether some knowledge is specific to some of the agents or common to all of them. In most cases, we will assume that agents are *initially homogeneous*, but will become differenciated through their history (perceiving and acquiring individually different knowledge). We shall thus represent knowledge of agent a_i with the following concepts:

Background theory. $\mathcal{T}^C$ represents *common prior knowledge common* to all agents. It is seen in our framework as *certain* knowledge.

Individual theory. $\mathcal{T}_i^I$ represents *individual knowledge*, specific to an agent. It is *certain* knowledge that differentiates the agents. We will further distinguish two kinds of knowledge in individual theories:

- **Observation set.** O_i represents *acquired knowledge* of each agent. It is *certain factual* knowledge that the agents have usually acquired through perception or communication. It contains all the *factual* information in the individual theory. Thus, it concerns specific data (*observations*) rather than general rules.
- **Rule set.** R_i contains all other non factual part of the individual theory. It represents *individual prior knowledge* differentiating the theoretical background of individual agents. That set can only grow through communication with other agents. Note that if the agents are *initially homogeneous*, this set is empty, and $\mathcal{T}_i^I = O_i$.

Hypothesis. h_i represents some *revisable* knowledge to which the agent is committed, though it knows it is not certain. It is usually built from its other knowledge, or interaction with other agents.

We will now illustrate these notions with an example instanciation of this framework to a simple abductive medical diagnosis.

Example 2. All agents knows the possible diseases and their symptoms, and they must build an hypothesis about a patient, each of them initially knowing only part of the symptoms. More formally, we consider:

– a set of possible disease $\mathcal{D} = \{$angina, bronchitis, flu, hayfever$\}$.
– a set of possible symptome $\mathcal{S} = \{$cough, mucus, fatigue, fever$\}$.
– a causality relation E linking a disease to the symptoms it can cause. We have $E = \{$(angina,fatigue), (angina,fever),(bronchitis, cough), (bronchitis, mucus), (flu, cough), (flu, fatigue), (flu, fever), (hayfever,mucus)$\}$. We define the function $effect(D, E)$ such that given a set of diseases D and a relation E, $effect(D, E) = \{s | \exists d \in \mathcal{D}, (d, s) \in E\}$, that gives us all the symptoms caused by a set of diseases according to some causality relation.

In this framework, we can thus have a *distributed diagnosis problem*, given by a set O_i of observations on the patient for each agent. For instance, we could have 3 agents, a_1, a_2, a_3 with $O_1 = \{$fatigue$\}$, $O_2 = \{$fever,$\neg$cough$\}$ and $O_3 = \{\,\}$.

Our different concepts would then map in the following way:

– *Background theory* $\mathcal{T}^C$ is the tuple $(\mathcal{D}, \mathcal{S}, E)$, that we shall call the *medical theory.* It is certain, and common to all agent.
– *observations* are either the presence or absence of a symptom.
– Then, agents being *initially homogeneous*, *rules sets* R_i are empty, and *individual theories* are just observation sets: $\mathcal{T}^I = O_i$.
– Finally, an *hypothesis* is a *diagnosis*, that is, a set of diseases.

If we did not consider agents to be initially homogeneous (that is, agents do not all have the same initial knowledge of medical theory), we would have a case of *distributed diagnosis with distributed theory.* Then we make the following adjustments:

- *Background theory* $\mathcal{T}^C = (\mathcal{D}, \mathcal{S}, E^C)$, where E^C is the symptoms of diseases that are known be everyone.
- Each agent knows part of the medical theory, represented by its *rules set* $R_i = E_i^I \subseteq E \setminus E^C$.
- *Individual theory* $\mathcal{T}_i^I = E_i^I \cup O_i$.

3.2 Consistency

As hypotheses are revisable, they might be contradictory with some new observations. The validity of an hypothesis is something that should be ensured and maintained. It is the basis of the evaluation process. However, the condition of this validity can vary according to the kinds of hypotheses and observations. We thus need some abstract notion, the consistency relation, to represent this.

Consistency of an Hypothesis with Certain Knowledge. To represent the adequation of what an agent believes with what it knows, we will use an abstract relation linking an hypothesis with certain knowledge : the *consistency relation.* As the background theory is common to all agents, we will incorporate it in the consistency relation. Thus, the consistency will link a hypothesis with an individual theory[1]. We shall denote this property by $Cons(h, \mathcal{T}^I)$, meaning that hypothesis h is *consistent* with individual theory $\mathcal{T}^I$. Such a relation can take different forms according to the reasoning involved. It can represent the notions of *anomaly* discussed in [6] or of *surprise* used in KEKADA[11]. Though we illustrate our presentation with a problem of medical diagnosis, several different instantiations of this framework can be found in [1], such as logical abduction [2] or inductive concept learning [3].

One important property that consistency can verify is the one of *compositionality.*

Definition 1 (Compositionality of a consistency relation). *A consistency relation is* compositional *iff the following equivalence is verified:*

$$Cons(h, \mathcal{T}^I) \text{ and } Cons(h, \mathcal{T}^{I'}) \Leftrightarrow Cons(h, \mathcal{T}^I \cup \mathcal{T}^{I'})$$

The $\Rightarrow$ direction of this equivalence is often called *additivity* [8]. It basically means that it is possible to consider independently each pieces of knowledge from the individual theory. The $\Leftarrow$ direction is best understood when we read the contrapositive: it says that when h is not consistent with some individual theory $\mathcal{T}^I$, it cannot become consistent again when that theory grows monotically. In

[1] Note that if agents are *initially homogeneous*, $\mathcal{T}^I$ can be replaced by an observation set O and consistency defined as a relation between h and O.

other words, an hypothesis assessed as inconsistent on the basis of an individual theory cannot become consistent when that theory grows. We refer to this latter property, following [8], as *incrementality*. It is often necessary to require that agents are initially homogeneous to ensure this property for the consistency relation.

Example 3. We now describe an instantiation of the consistency relation for distributed medical diagnosis problem. In this problem, an hypothesis (diagnosis) is valid if the assumed diseases causes the symptoms that are observed and no more.

More formally: $\forall h, \forall \mathcal{T}^I = E^I \cup O$, $Cons(h, \mathcal{T}^I)$ iff

- $\forall s \in \mathcal{S}, \neg s \notin O \vee s \notin effect(h, E^C \cup E^I)$ (coherence)
- $\forall s \in O \cap \mathcal{S}, s \in effect(h, E^C \cup E^I)$ (completeness)

This consistency relation is compositional iff agents are initially homogeneous ($\forall \mathcal{T}^I, E^I = \{\,\}$).

Thus, using the background theory defined earlier, common to all agents ($\forall i, E_i^I = \{\,\}$), and $h_1 = \{$angina,hayfever$\}$:

- h_1 is consistent with $\{\neg$ cough, fever, mucus$\}$ (coherent and complete).
- h_1 is inconsistent with $\{$fatigue, mucus, $\neg$ fever$\}$ (incoherence since fever should be an effect of angina).
- h_1 is inconsistent with $\{$cough, fever, mucus$\}$ (incompleteness since cough is not an effect of angina nor hayfever).

Group Consistency. Now we extend this notion of consistency to agents and groups of agents by introducing the notion of group-consistency:

Definition 2 (Group Consistency). *An agent a_i is group consistent wrt. the group of agents G ($GCons(a_i, G)$) iff $Cons(h_i, \cup_{i \in G} \mathcal{T}_i^I)$)*

A stronger notion of consistency requires any agent within the group to be consistent with the entire group.

Definition 3 (Mutual Consistency). *A group of agents is mutually consistent ($MCons(G)$) iff $\forall a_i \in G$, it is the case that $GCons(a_i, G)$.*

Now for the purpose of our work, we shall mainly be interested in some interesting particular cases which depend on the cardinality of group G:

Internal consistency. this is the limit case when G is limited to a single agent. In this case, Group and Mutual consistency collapse into a single notion that we shall call *internal consistency* ($ICons(a_i)$).

Peer consistency. when the group of agents we consider contains only two agents. In this case we can distinguish both the *peer consistency* of an agent wrt a fellow agent, and the *mutual peer consistency* of a group of two agents. This is especially important in our context, since our communication protocols only deal locally with bilateral communications.

MAS-consistency. we conclude with the limit case involving all agents within the society. In this case, we will refer to the *MAS-consistency* of an agent wrt to the society; and to the *mutual MAS-consistency* of a society of agents.

Besides these three cases, group-consistency is also a useful tool to describe the state of consistency of various subgroup of a system. If agents represents scientists, interesting sub-groups could be workgroups, laboratories, or communities of specialists of a given domain, depending on what the whole system represents (whole scientific community or smaller part of it).

3.3 Reasoning

The consistency notion captures the adequacy of an hypothesis, but does not make assumption on how to build such a hypothesis. There exists a number of different algorithms to produce hypotheses with a single agent, that can be used to model agents' internal reasoning. To abstract away from the specificity of different algorithms we will introduces the notion of *hypothesis formation function* and *internal revision mechanism* corresponding respectively to the generation process and the refinement or pursuit process.

Evaluation of an Hypothesis. An individual must be able to evaluate an hypothesis with respect to its observations. Especially, it must be able to determine if a given hypothesis is consistent with its observation memory. When it is not the case, an informed answer explaining the origin of the inconsistency would be useful for revising the hypothesis. If the consistency relation is compositionnal, such an answer can just be a *counter-example*, that is, a piece of information k that is not consistent with the hypothesis.

We will define an agent's*internal evaluation mechanism* ξ as a function that takes an hypothesis h and the individual theory $\mathcal{T}_i^I$ of the agent, and return a set of counter-examples C. If $Cons(h, \mathcal{T}_i^I)$ then C is empty, otherwise, C contains one or more counter examples for this hypothesis taken from $\mathcal{T}_i^I$ (that is C contains one or more pieces of information from $\mathcal{T}_i^I$ inconsistent with h).

Generation and Pursuit of an Hypothesis for an Individual. A *hypothesis formation mechanism* $\mathcal{E}_h$ is a process that takes an agent internal states and returns an hypothesis that is consistent with these internal states. Usually, we will restrict the internal states that can have impact on the result and consider only the individual theory $\mathcal{T}^I$. Thus, usually, $\mathcal{E}_h(a_i)$ can be written as $\mathcal{E}_h(\mathcal{T}_i^I)$.

When an agent already has some hypothesis h consistent with its individual theory $\mathcal{T}^I$ and receive some new piece of information k that contradicts its hypothesis, it can try to *refine* this hypothesis. This process is called the *pursuit* of an hypothesis [5,18]. An *internal pursuit mechanism* μ is thus a process triggered by an agent with hypothesis $h \in \langle$ and individual theory $\mathcal{T}^I$ that receives a new piece of information k to ensure that its internal consistency is maintained. This process adds k to the individual theory, and revise the hypothesis if necessary (we will denote the resulting hypothesis by $\mu(h, \mathcal{T}^I, k)$).

Collective Pursuit. For groups of agents, we will focus on *pursuit mechanisms*, assuming that the generation of a first hypothesis is rather an individual process. However this stance mostly affects the naming of our representation rather than the underlying formalisation. A *pursuit mechanism* $\mathcal{M}$ is a process by which an agent a_i receving a new observation o communicates with a group G of agents to refine its hypothesis, and possibly those of the other agents. We denote this application of $\mathcal{M}$ by a_i with G upon reception of o by $\mathcal{M}(a_i, G, o)$. Depending on the cardinality of the group G we will distinguish three levels:

Internal pursuit mechanism. The limit case when $n = 1$ has already been described previously with internal pursuit mechanism μ.

Local pursuit mechanism. A local pursuit mechanism $\mathcal{M}^2$ is the process by which an agent a_i that has received a piece of information k communicate with another agent a_j to refine its hypothesis, and possibly those of a_j. It will usually be based on some internal pursuit mechanism μ by which agents can take into account new informations. As pursuit also involves some evaluation, internal evaluation mechanisms ξ might also be used.

Global pursuit mechanism. A global pursuit mechanism $\mathcal{M}^n$ involves all agents in the system. It is a process by which an agent a_i receving some new piece of information k triggers a series of communications (usually local pursuit mechanisms) involving all agent in the system, and revising the internal states of a_i (its hypothesis) and possibly those of the other agents.

Note that though these pursuit mechanisms are defined here as being trigerred upon reception of an new piece of information, an agent can decide to trigger them to reach consistency at any moment (in which case $k = \{\}$). The aim of a pursuit mechanism is indeed to guarantee some property on the internal states of the agents.

Definition 4 (Guarantee of a property). *A pursuit mechanism* $\mathcal{M}$ guarantees *a property* $P(a_i, G)$ *iff it is the case that any execution of* $\mathcal{M}$ *by* a_i *with* G *will result in a situation where* $P(a_i, G)$ *holds.*

4 Learner Critic Mechanisms for Distributed Hypothesis Refinement

If we consider our example conversation betwen a senior MD and its assistant, we have some recurring structure. An assistant *proposes* a diagnosis to the senior MD that either refutes it with a *counter-example* or *accepts* it. Once an hypothesis has been accepted, further tests are planned to check it. This proposal and rebuttal structure, in which an individual proposes hypotheses and another evaluates them can be found in different domains. We refer to it as a learner and critic approach. Such approach have been used in a variety of learning problem and methods such as case-based reasoning [16] or concept learning [22]. It is also suggested in [20] as a possible division of work for knowledge discovery, learners being *constructive dogmatists* and critics being *skeptical critics* (following Kuipers' terminology[10]).

This section present some pursuit mechanisms based on our framework, that can thus benefit from its generality to be applied in different hypothesis formation contexts. that can be used to reach or maintain different levels of consistency in a system of agents, based on this learner and critic approach. An agent taking a role of *learner* builds locally an hypothesis that is proposed to other agents, acting as *critics*. These critics then criticize the hypothesis and answer by giving *counter-examples*, that is, pieces of information inconsistent with the proposed hypothesis, or by *accepting* the hypothesis. Thus *learners* have a role of *hypothesis generation* or *formation*, whereas *critics* have mainly an evaluation role.

4.1 A Local Pursuit Mechanism

Our basic local pursuit mechanism $\mathcal{M}_U^2$ is an asymmetric process called *Unilateral Hypothesis Exchange* (UHE). The agent applying the mechanism is the *learner*, actively building and refining an hypothesis. The second agent is a *critic*, using its knowledge to acknowledge or invalidate hypotheses.

It works as follows. The *learner* agent a_i first updates its hypothesis h_i to h'_i using an internal pursuit mechanism μ guaranteeing $ICons(a_i)$. Then it *proposes* it to the partner agent a_j, called *critic*, and a_j applies its evaluation mechanism ξ on this hypothesis. If $\xi(h) = \{\}$, it replies with $accept_{direct}$ and adopts h'_i as its new working hypothesis, otherwise it sends *counter-example(*$\xi(h)$*)*. Upon reception of one or more counter-examples, a_i applies again μ to revise h'_i and propose the resulting hypothesis as before, except that an acceptance will now result in a $accept_{indirect}$ message. If consistency is compositional, $\mathcal{M}_U^2$ *guarantees mutual peer consistency* for any pair of agents.

Example 4. We consider two agents in the distributed diagnosis problem. Their observation sets are: $O_1 = \{\text{cough}\}$ and $O_2 = \{\text{fever}, \neg \text{ mucus}\}$.
Their hypotheses are $h_1 = \{\text{bronchitis}\}$ and $h_2 = \{\text{angina}\}$. Now if these two agents were to communicate using the Uniteral Hypothesis Exchange Protocol, we would get the following dialogue.

a_1 sends to a_2	*propose*({bronchitis})
a_2 sends to a_1	*counter-example*(¬ mucus)
a_1 sends to a_2	*propose*({flu})
a_2 sends to a_1	*accept*

4.2 Global Pursuit Mechanisms

In a fully connected society of agents, where an agent can always communicate with another, it is possible to plan a series of local revisions in order to ensure MAS-consistency. In such case, we will present some complete global revision mechanism that articulates those local revisions in different ways, according to the parameters of the system or the requirements. If the structure is more complex, but still static and connected, then some agents must acts as relay to ensure peer-consistency of two agents that cannot communicate. We will not detail mechanisms for structured networks here, but a propagation mechanism to deal with this case can be found in [2]. We now present a complete global pursuit mechanism for fully connected societies of agent : *Clock-wise hypothesis.*

The general idea is to make repeated uses of a local mechanism guaranteeing mutual peer consistency to eventually get a MAS-consistent hypothesis adopted by all agents. The hypothesis must be validated by all agents in turn without being changed. Any change in the hypothesis forces us to check it again from the beginning. The *learner* agent a_1 first uses the local pursuit mechanism $\mathcal{M}_U^2$ to reach mutual peer-consistency with a_2. Then it does the same with agent a_3. If $\mathcal{M}_U^2$ ends with an $accept_{direct}$, then a_1 proceeds to exchange its hypothesis with the next agent (a_4), else ($accept_{indirect}$) it goes back to a_2. This iterates, each $accept_{indirect}$ restarting the process with a new hypothesis submitted to a_2. When a_n sends a $accept_{direct}$, it means that the hypothesis has been accepted and adopted in turn by all agents. In such a case, the mechanism ends, and this common hypothesis is consistent with every individual theory.

Therefore, if the consistency relation is compositional, $\mathcal{M}_C^n$ *guarantees mutual MAS-consistency* in any fully connected society of non-individualistic agents.

An example of variant for non-individualistic agent in fully connected societies is the *broadcasted hypothesis exchange* $\mathcal{M}_B^n$, in which, instead of sequentially proposing its hypothesis in turn to every critic agent, the learner agent broadcast its hypothesis to all critics. It then waits for individual answers (accept or counter-example) from all of them, and applies its internal pursuit process to braodcast a new hypothesis if any counter-example is received. Otherwise, it means the hypothesis is SMA-consistent, and the mechanism ends. Thus, $\mathcal{M}_B^n$ also guarantees mutual MAS-consistency for fully connected societies of agents if the consistency relation is compositionnal.

5 Conclusion

Through the presentation of the bases for a generic framework in this paper, we have shown that a multi-agent approach is useful to model the richness of scientific activity, including the social dimension of knowledge construction – which until now has been largely neglected by the machine discovery community. Our first motivation, as in [21], is therefore to go about reconciling Machine Discovery with relatively recent developments in social philosophy of science.

However we believe that the impact of this framework is potentially wider. The two typical motivations in AI are, on one hand, to obtain systems which are closer to human intelligence to the point of cognitive plausibility, and on the other hand to obtain systems which obtain higher levels of performance. So a relevant question to address concerns the performance of the distributed framework presented here. More precisely, does the distribution over a society of agents affect the performance compared to a mono-agent system that could access at negligible cost all information (knowledge, observation) available at a given time? A first answer is that though intuitively most would believe this impact to be negative, there are also some arguments to think it could be positive in some settings. In [3] the application of this framework with an order-dependent incremental bottom-up learning process to inductive concept learning yield interesting results. Multiplication of the evaluation and revision process caused by distribution improved the final accuracy. Besides, the whole area of ensemble learning in supervised learning [7], known to have significantly improved single learner performance, can be seen as a form of distribution that can easily be modeled in our multi-agent framework.

Furthermore, we know that a centralized organization of science is not credible for human scientists, because of the cognitive limitations specific to humans (in terms of memory, processing or communication). But one could argue that these limitations do not apply to computer systems. However the actual trend is to reproduce in computer-supported science (in particular e-science working over the internet) the distributed nature of human science. Databases and knowledge repositories are distributed internationally and computing capacity as well, especially within the grid metaphor. Therefore we believe that this multi-agent approach is even more relevant at a time of science virtualization.

References

1. Bourgne, G.: Propagation et Affinement d'Hypot'eses sous Contraintes Communicationnelles. PhD thesis, Université Dauphine - LAMSADE (2008)
2. Bourgne, G., Seghrouchni, A.E.F., Maudet, N.: Towards refinement of abductive or inductive hypothesis through propagation. In: Pre-Proc. of AIAI 2007, October 2007, pp. 20–40 (2007)
3. Bourgne, G., El Fallah Seghrouchni, A., Soldano, H.: Smile: Sound multi-agent incremental learning. In: AAMAS, pp. 164–171. ACM Press, New York (2007)
4. Corruble, V., Ganascia, J.-G.: Induction and the discovery of the causes of scurvy: A computational reconstruction. Artificial Intelligence 91(2), 205–223 (1997)
5. Curd, M.V.: The logic of discovery: an analysis of three approaches, Reidel, pp. 173–183 (1980)
6. Darden, L.: Anomaly-Driven Theory Redesign: Computational Philosophy of Science Experiments, pp. 62–78. Blackwell Publishers, Malden (1998)
7. Dietterich, T.G.: Ensemble Learning, pp. 405–408. MIT Press, Cambridge (2002)
8. Flach, P.A.: Abduction and induction: syllogistic and inferential perspectives. Technical report, University of Bristol, Bristol, UK (1996)
9. Klahr, D., Simon, H.A.: Studies of scientific discovery: Complementary approaches and convergent findings. Psychological Bulletin 125(5), 524–543 (1999)
10. Kuipers, T.A.F.: Structures in Science – Heuristic Patterns Based on Cognitive Structures. Kluwer Academic Publishers, Dordrecht (2001)
11. Kulkarni, D., Simon, H.A.: Experimentation in Machine Discovery. Morgan Kaufmann, San Francisco (1990)
12. Langley, P.: Bacon.1: A general discovery system. In: Proc. of Canadian AI 1978, Toronto, Ontario, pp. 173–180 (1978)
13. Latour, B.: Laboratory Life: The Social Construction of Scientific Facts (SAGE Library of Social Research), June 1979. Sage Publications, Inc., Thousand Oaks (1979)
14. Laudan, L.: Why Was the Logic of Discovery Abandoned?, pp. 173–183 (1980)
15. Longino, H.E.: Science as Social Knowledge: Values and Objectivity in Scientific Inquiry. Princeton University Press, Princeton (1990)

16. Ontañon, S., Plaza, E.: Recycling data for multi-agent learning. In: ICML 2005, pp. 633–640. ACM Press, New York (2005)
17. Popper, K.: The Logic of Science Discovery, Routledge, London (1959)
18. Schaffner, K.F.: Discovery and Explanation in Biology and Medicine. The university of Chicago Press, Chicago (1993)
19. Simon, H.A.: Does scientific discovery have a logic? Philosophy of Science 40(4), 471–480 (1973)
20. Thagard, P.: Computational philosophy of science. MIT Press, Cambridge (1988)
21. Wajnberg, C.D., Corruble, V., Ganascia, J.G., Moulines, C.U.: A structuralist approach towards computational scientific discovery. Knowledge Discovery 3245, 412–419 (2004)
22. Wang, J., Gasser, L.: Mutual online concept learning for multiple agents. In: AAMAS, pp. 362–369. ACM Press, New York (2002)

Active Learning for High Throughput Screening

Kurt De Grave, Jan Ramon, and Luc De Raedt

Katholieke Universiteit Leuven, Celestijnenlaan 200A, 3001 Leuven, Belgium
{Kurt.DeGrave,Jan.Ramon,Luc.DeRaedt}@cs.kuleuven.be
http://www.cs.kuleuven.be/cwis/research/dtai/

Abstract. An important task in many scientific and engineering disciplines is to set up experiments with the goal of finding the best instances (substances, compositions, designs) as evaluated on an unknown target function using limited resources. We study this problem using machine learning principles, and introduce the novel task of *active k-optimization*. The problem consists of approximating the k best instances with regard to an unknown function and the learner is active, that is, it can present a limited number of instances to an oracle for obtaining the target value. We also develop an algorithm based on Gaussian processes for tackling active k-optimization, and evaluate it on a challenging set of tasks related to structure-activity relationship prediction.

Keywords: Active Learning, Chemical compounds, Optimization, QSAR.

1 Introduction

The philosophy of science has since a long time studied scientific discovery processes, and recently, the artificial intelligence community has taken up the challenge as to study how scientific discovery processes can be automated [1]. One aspect that is quite central in scientific discovery, as well as in many engineering problems, is that of determining the next experiment to be carried out.

In this paper, we apply machine learning principles to select the next experiment in High Throughput Screening (HTS), an important step in the drug discovery process, in which many chemical compounds are screened against a biological assay. The goal of this step is to find a few lead compounds within the entire compound library that exhibit a very high activity in the assay. This is also the setting of the new robot that is currently under development in the robot scientist project [1] at the University of Aberystwyth. The task is akin to many other scientific and engineering disciplines, where the challenge is to identify or design those instances that have optimal performance according to some criterion that needs to be optimized. For instance, in membrane design, it is important to find those parameters of the process that yield the best performance [2]; in coherent laser control, the goal is to find the laser pulse that maximally catalyzes a chemical reaction [3]. In this type of application, the target criterion is unknown to the scientist or engineer and only partial information can be obtained by testing specific instances for their performance. Such tests correspond to experiments and can be quite expensive.

In HTS, it is not sufficient to find just a single optimal example. The optimal compound might ultimately not be usable as a starting point for the next step in the drug discovery process for various reasons unrelated to its performance in the assay. Therefore, a number of alternatives need to be found as well. Ideally, each of these near-optimal alternatives would have a different modus operandi. The challenge then is to identify the k best performing instances using as few experiments as possible. We will refer to this task as *active k-optimization*.

This task is closely related to global function optimization. It is also related to active learning in a regression setting [4], where the goal is to find a good approximation of the unknown target function by querying for the value of as few instances as possible. Whereas this approach allows one to identify the best scoring instances, it is also bound to waste resources in the low scoring regions of the function. Thus, in contrast to active regression, an extra ingredient is added to the problem that is reminiscent of reinforcement learning. The learner will have to find the right balance between exploring the space of possible instances and exploiting those regions of the search space that are expected to yield high scores according to the current approximation of the function. Finally, active k-optimization differs also from active concept-learning that has already been applied in applications such as structure-activity relationship prediction [5] in that a regression task has to be performed.

This paper is organized as follows: in Section 2 we formalize the problem, and in Section 3 we propose a Gaussian process model for tackling it. In Section 4 we investigate a number of different strategies for balancing exploration and exploitation. We evaluate our approach experimentally in Section 5. Finally, We discuss related work and possible extensions in Section 6.

2 Problem Statement

Our work is especially motivated by the structure-activity relationship domain, where high-throughput approaches assume the availability of a large, diverse but fixed library of compounds. Hence, the pool-based active learning setting is most appropriate. In this setting, the learner incurs a cost only when asking for the measurement of the target value of a particular instance, which must be selected from a known, finite pool. In principle, the learner may be able to exploit the distribution of the examples in the pool without cost. To some extent, this setting is therefore also a semi-supervised learning setting.

The problem sketched in section 1 can be more formally specified as follows:

Given:

– a pool $\mathcal{P}$ of instances,
– an unknown function f that maps instances $x \in \mathcal{P}$ on their target values $f(x)$,
– an oracle that can be queried for the target value of any example $x \in \mathcal{P}$,
– the maximal number N_{max} of queries that the oracle is willing to answer,
– the number k of best scoring examples searched for.

FIND:

– the top k instances in $\mathcal{P}$, that is, the k instances in $\mathcal{P}$ that have the highest values for f.

One can see that the above combinatorial optimization problem is a close relative to the problem of global function optimization. Algorithms developed in the discipline of global function optimization only consider $k = 1$ and are optimized for continuous domains. Still, largely the same concepts and techniques can be used.

From a machine learning perspective, the key challenge is to determine the policy for determining the next query to be asked, based on the already known examples. This policy will have to keep the right balance between exploring the whole pool of examples and exploiting those regions in the pool that look most promising.

3 Gaussian Process Model

We will use a Gaussian process model [6] for learning, also known as Kriging. In this section we briefly review the necessary theory. More detailed explanations can be found in several textbooks on the subject [7,8].

We first introduce some notation. We assume that there is a feature map $\phi : \mathcal{P} \rightarrow F$ mapping examples to a feature space F. We denote with $X_N = [x_1 x_2 \dots x_N]^\top$ the vector of the N first examples, with $T_N = [t_1 t_2 \dots t_N]^\top$ the vector of their target values, and with $\Phi_N = [\phi(x_1)\phi(x_2)\dots\phi(x_N)]^\top$ the matrix where each row is the image of an example (abusing notation in case F has infinite dimension). For our objective criterion, $\|T_N\|_{\text{best}-k}$ is the average of the k largest elements of the vector T_N, where we assume all target values to be positive. The notation $\|\cdots\|_{\text{best}-k}$ is warranted since the function satisfies all properties of a vector norm under this assumption.

We assume that there is a linear approximate model for the target value $t(x)$ of instances

$$m(x) = w^\top \phi(x) \tag{1}$$

(with $w \in F$ a weight vector) such that the values of the modeling error $t(x) - m(x)$ for examples randomly drawn from the pool $\mathcal{P}$ are independently Gaussian distributed with zero mean and variance σ^2. We use the following notation to denote that a random variable has a Gaussian distribution:

$$t(x) - m(x) \sim \mathcal{N}(0, \sigma^2). \tag{2}$$

In matrix notation Equation (2) becomes $P(T_N|\Phi_N, w) \sim \mathcal{N}(\Phi_N w, \sigma^2 I)$. Our prior belief for the vector w is Gaussian with zero mean and covariance matrix Σ_{pw}. One can compute the posterior $P(w|T_N, \Phi_N)$ using

$$P(w|T_N, \Phi_N) \propto P(w)P(T_N|\Phi_N, w). \tag{3}$$

A straightforward derivation gives

$$w|T_N, X_N \sim \mathcal{N}(\bar{w}_N, \Sigma_{w,N}) \tag{4}$$

where the mean $\bar{w}_N$ and variance $\Sigma_{w,N}$ is

$$\begin{aligned} \bar{w}_N &= \sigma^{-2}\Sigma_{w,N}\Phi_N^\top T_N \\ \Sigma_{w,N} &= (\sigma^{-2}\Phi_N^\top \Phi_N + \Sigma_{pw}^{-1})^{-1}. \end{aligned}$$

For a new example x_* we can then estimate the target value by

$$t_*|X_N, T_N, x_* \sim \mathcal{N}\big(\bar{w}_N^\top \phi(x_*), \phi(x_*)^\top \Sigma_{w,N}\phi(x_*)\big) \tag{5}$$

One can show that this formula is equivalent to the following distribution which does not refer to feature space explicitly:

$$t_*|X_N, T_N, x_* \sim \mathcal{N}\big(\bar{t}_*, var(t_*)\big) \tag{6}$$

where

$$\bar{t}_* = k(x_*, X_N)(k(X_N, X_N) + \sigma^2 I_N)^{-1} T_N \tag{7}$$

$$var(t_*) = k(x_*, x_*) - k(x_*, X_N)(k(X_N, X_N) + \sigma^2 I_N)^{-1} k(X_N, x_*) \tag{8}$$

and where k is a kernel defined by

$$k(x, y) = \phi(x)^\top \Sigma_{pw}\phi(y) \tag{9}$$

Here, we use the abbreviations

$$\begin{aligned} k(x_*, X_N) &= \big[k(x_*, x_1)k(x_*, x_2)\dots k(x_*, x_N)\big]^\top \\ k(X_N, x_*) &= k(x_*, X_N)^\top \end{aligned}$$

for vectors of kernel values, and

$$k(X_N, X_N) = [k(x_1, X_N)k(x_2, X_N)\dots k(x_N, X_N)]$$

for a matrix of kernel values. $k(X_N, X_N)$ is called the Gram matrix.

4 Selection Strategies

Different example selection strategies exist. In geostatistics, they are called infill sampling criteria [9].

In active learning, in line with the customary goal of inducing a model with maximal accuracy on future examples, most approaches involve a strategy aiming at greedily improving the quality of the model in regions of the example space where its quality is lowest. One can select new examples for which the predictions of the model are least certain or most ambiguous. Depending on the learning algorithm, this translates to near decision boundary selection, ensemble entropy reduction, version space shrinking, and others. In our model, it translates to *maximum variance* on the predicted value or $\arg\max(var(t_*))$.

Since our goal is not model accuracy but finding good instances, a more appropriate strategy is to select the example that the current model predicts to have the best target value, or $\arg\max(\bar{t}_*)$. We will refer to this as the *maximum predicted* strategy. For continuous domains, it is not guaranteed to find the global, or even a local minimum [10].

A less vulnerable strategy is Cox and John's lower confidence bound criterion [11], which we will refer to as the *optimistic* strategy. The idea is to not sample the example in the database where the expected reward $\bar{t}_*$ is maximal, but the example where $\bar{t}_* + b \cdot var(t_*)$ is maximal. The parameter b is the level of optimism. It determines the balance between exploitation and exploration. It is obvious that the maximum predicted and maximum variance strategies are special cases of the optimistic strategy, with $b = 0$ and $b = \infty$ respectively. In a continuous domain, this strategy is not guaranteed to find the global optimum because its sampling is not dense [10].

Another strategy is to select the example x_{N+1} that has the highest probability of improving the current solution [12]. One can estimate this probability as follows. Let the current step be N, the value of the set of k best examples be $\|T_N\|_{\text{best}-k}$ and the k-th best example be $x_{\#(k,N)}$ with target value $t_{\#(k,N)}$. When we query example x_{N+1}, either t_{N+1} is smaller than or equal to $t_{\#(k,N)}$, or t_{N+1} is greater. In the first case, our set of k best examples does not change, and $\|T_{N+1}\|_{\text{best}-k} = \|T_N\|_{\text{best}-k}$. In the latter case, x_{N+1} will replace the k-th best example in the set and the solution will improve. Therefore, this strategy selects the example x_{N+1} that maximizes $P(t_{N+1} > t_{\#(k,N)})$. We can evaluate this probability computing the cumulative Gaussian

$$P(t_{N+1} > t_{\#(k,N)}) = \int_{t=t_{\#(k,N)}}^{\infty} \mathcal{N}(\bar{t_*}, var(t_*))dt \ , \tag{10}$$

where $\bar{t}_{N+1}$ and $\text{var}(t_{N+1})$ can be obtained from Equations (7, 8). In agreement with [13], we call this the *most probable improvement* (MPI) strategy.

Yet another variant is the strategy used in the Efficient Global Optimization (EGO) algorithm [14]. EGO selects the example it expects to improve most upon the current best, i.e the one with highest

$$\mathbb{E}[\max(0, t - t_{\#(k,N)})] = \int_{t=t_{\#(k,N)}}^{\infty} (t - t_{\#(k,N)})\mathcal{N}(\bar{t_*}, var(t_*))dt \ . \tag{11}$$

This criterion is called *maximum expected improvement* (MEI).

In real-world applications it is not only important to find a solution quickly, but also to know when the optimal (or an adequate) solution has been found. The trade-off one has to make here is between budget and quality.

In a large number of situations, one will have a fixed budget and the goal will be to have an optimal solution when the budget is exhausted. Sometimes however, one can save significantly on the budget when a slightly suboptimal solution is acceptable or when the risk of having a suboptimal solution is small.

One approach is to bound the probability that any of the non-queried examples is better than the k-th best example so far. From Equation (10) we can compute for a particular example x that has not been queried the probability that its target value t will be larger than $t_{\#(k,N)}$. We can then write

$$P(\exists x \in \mathcal{P} \setminus X_N : f(x) > t_{\#(k,N)}) \leq \sum_{x \in \mathcal{P} \setminus X_N} P(f(x) > t_{\#(k,N)}) \qquad (12)$$

which is a tight upper bound if the individual $P(f(x) > t_{\#(k,N)})$ are small (as is the case when we consider to stop querying) and independent.

5 Experimental Evaluation

As sketched in the introduction, we shall experimentally evaluate our collection of methods in the area of high throughput screening in the context of drug lead discovery. In particular, we shall evaluate the algorithms on the US National Cancer Institute (NCI) 60 anticancer drug screen (NCI60) dataset [15]. This repository contains measurements of the inhibitory power of tens of thousands of chemical compounds against 59 different[1] cancer cell lines. NCI reports the log-concentration required for 50% cancer cell growth inhibition (GI_{50}) as well as cytostatic and cytotoxic effect measures, but we only used the log GI_{50} data. Real world drug discovery screening operations would normally include non-toxicity in the measure to optimize for[2].

To perform a measurement, each compound is diluted repeatedly, yielding a geometric series of concentrations. The actual GI_{50} can turn out to be outside the range of concentrations chosen a-priori. In that case, one only knows an upper or lower bound for the value, and a new measurement for that compound must be performed to collapse the interval to a point value. We ignored such out of bounds measurements.

To improve interpretability of the experiments, an equally sized pool of 2,000 compounds was randomly selected from each assay. This also saved computational resources, though it would have been possible to use the entire dataset, for all algorithms are only of complexity $O(N_{max} \cdot \#\mathcal{P})$ as long as both the number of features and the budget are constant.

We used a linear kernel. The chemical structure of each compound was represented as 1024 FP2 fingerprints, calculated using Open Babel 2.1.0. The algorithms were bootstrapped with GI_{50} measurements of ten random compounds. Since the result depends on this random boot sample, each experiment was repeated 20 times and the results were averaged.

In each assay, NCI measured some compounds repeatedly. For these compounds, the dataset lists the standard deviation among the measurements, as

[1] One of the originally 60 cell lines was evicted because it was essentially a replicate of another [16].

[2] E.g. the specificity index, usually defined as the log-ratio of the toxic concentration to the effective concentration.

well as the average. In order to estimate the measurement error for each assay, we used the unweighted average standard deviation over all repeated measurements in the assay. This value was used as the standard deviation σ in the Gaussian term of our model in Equation (2).

To evaluate our algorithms in practice, we recorded $\|T_N\|_{\text{best}-k}$ as a function of the fraction of compounds tested. For every setting (selection strategy, value of k), these functions were then averaged over the 59 datasets considered. Figure 1 plots these curves for $k \in \{1, 10, 25, 100\}$ for all described strategies and random selection. For the optimistic strategy, we tested optimism levels of 0.5, 1, and 2.

Table 1 lists for several budgets N_{max} which strategy is best (attains the highest $\|T_{N_{max}}\|_{\text{best}-k}$). The budget is shown as a percentage of the pool size. For each different strategy, the table then also gives the Wilcoxon signed-rank test p-value for the null hypothesis that the difference between the top-k values of this strategy and those of the best strategy is on average 0.

We are now well equipped to answer four important questions about our algorithm:

Q1. Do active k-optimization strategies isolate valuable instances quicker than random selection?
Q2. What is the relative performance of the different selection strategies listed in Section 4?
Q3. Do strategies that take k into account perform better than strategies that do not?
Q4. Can the stopping criterion (Eq. 12) be used to decide when a near-optimal solution has been found?

5.1 Expedience

From the results presented in Table 1 and Figure 1, one can see that random example selection clearly performs worse than all other selection methods in all settings, except for the maximum variance strategy which does still worse for large budgets, especially for $k = 100$. We can conclude that the answer to question **Q**1 is positive, because actively choosing examples with one of the presented strategies substantially speeds up the finding of examples with high target values.

It is remarkable that the starting points of the random strategy are lower for higher k. This is due to the fact that the distribution of target values is skewed: compounds with very small target values are sparser than compounds with very large target values. In this way, $\|T_N\|_{\text{best}-k}$ decreases only slowly while $\|T_N\|_{\text{worst}-k}$ (the average of the k smallest elements of T_N) increases quickly for larger k. This causes the value of a random sample to be lower when scaled to a $[0, 1]$ interval. That the non-random strategies start higher than the random strategy for $k = 25$ and $k = 100$ is due to the fact that there are only 10 bootstrapping examples, and the non-random strategies actively select 15 (for $k = 25$) or 90 (for $k = 100$) examples before they can be evaluated a first time.

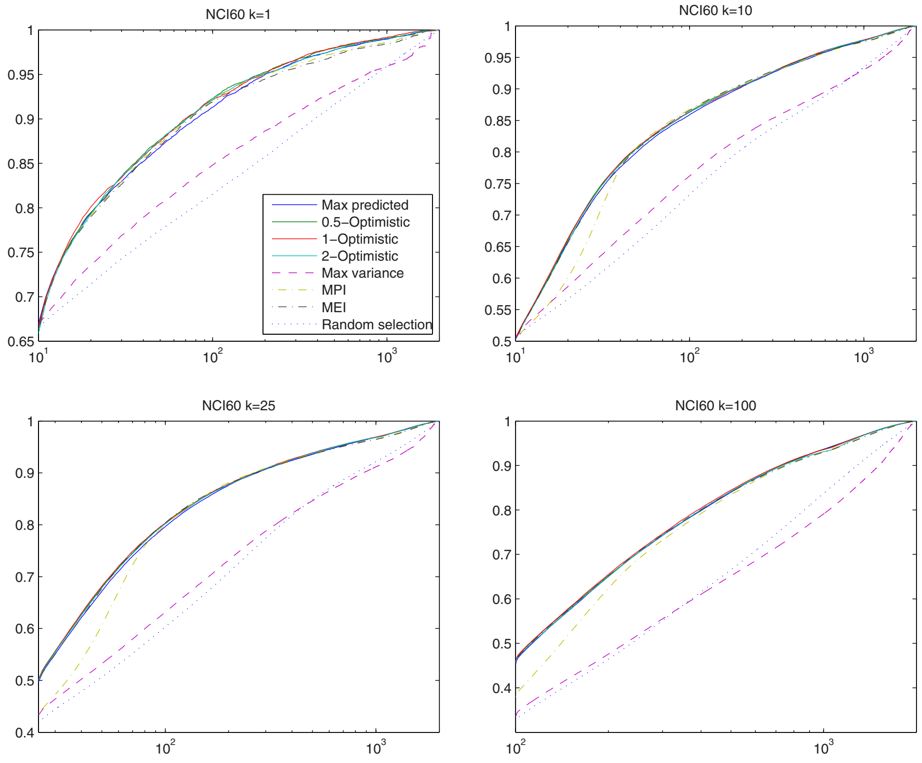


Fig. 1. The value of $\|T_N\|_{\text{best}-k}$ in each step, for all proposed active learning strategies and random selection, averaged over 20 runs for each of the 59 datasets. A log scale is used on the horizontal axis to reveal the performance for small as well as large budgets. The vertical axis is scaled to place the aggregate target value of the overall k best compounds at one and the worst k compounds at zero.

5.2 Relative Performance

Unsurprisingly, one can see that querying the maximally uncertain example is (in contrast to settings where one tries to optimize accuracy) not a good k-optimization strategy.

Overall, on the NCI60 datasets, the optimistic strategy with an optimism level of 1 was most robust. In all situations considered, it performed either best or not significantly worse than the best strategy. The difference with 2 and 0.5 optimism is more pronounced for higher values of k. Note that we exploited the information in the NCI datasets about the accuracy of the measurements. For other datasets that do not allow to estimate the accuracy of the input data, it may be harder to come up with a good value for $var(t_*)$. One can use a maximum likelihood estimate, at the cost of some robustness [9].

Greedily querying the example for which the highest target value is predicted, performs slightly worse than the optimistic strategy. The MPI strategy performs

Table 1. p-values for $k \in \{1, 10, 25, 100\}$. ϵ indicates that $p < 10^{-8}$.

Budget	10%	15%	20%	25%	10%	15%	20%	25%
k	1				10			
Max predicted	0.305	0.304	0.039	0.088	0.106	0.497	0.040	0.021
Optimistic ($b = 0.5$)	**Best**	**Best**	0.282	0.392	0.274	0.456	0.251	0.111
Optimistic ($b = 1$)	0.837	0.776	**Best**	**Best**	0.141	0.390	**Best**	**Best**
Optimistic ($b = 2$)	0.898	0.472	0.094	0.229	0.179	0.298	0.179	0.298
Max variance	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ
MPI	0.946	0.538	0.108	0.174	0.455	0.230	0.189	0.052
MEI	0.037	0.057	0.005	0.047	**Best**	**Best**	0.934	0.809
Random	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ
Budget	10%	15%	20%	25%	10%	15%	20%	25%
k	25				100			
Max predicted	0.074	0.437	0.015	0.015	0.046	0.063	0.003	0.010
Optimistic (b= 0.5)	0.202	0.319	0.177	0.192	0.197	0.118	0.007	0.022
Optimistic ($b = 1$)	0.280	0.634	**Best**	**Best**	**Best**	**Best**	**Best**	**Best**
Optimistic ($b = 2$)	0.083	0.673	0.264	0.478	0.042	0.170	0.016	0.068
Max variance	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ
MPI	**Best**	**Best**	0.385	0.141	10^{-5}	0.005	0.003	0.001
MEI	0.487	0.184	0.158	0.083	0.254	0.492	0.019	0.001
Random	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ

worse than the optimistic strategies in the very beginning, except for $k = 1$. It performs (and allegedly behaves) similarly to the maximum variance strategy when it hasn't seen many more examples than the 10 random bootstraps. From about 5% for $k = 10$ and 10% for $k = 25$, its performance is competitive and sometimes best, but it again becomes suboptimal for high budgets. The MEI strategy performs extremely well for $k = 10$, but is outperformed in some other settings. This concludes our answer to question **Q**2.

5.3 Utility of Advance Knowledge of k

In Table 2 and Figure 2 we see that the active learning strategies that explicitly take k into account, perform far better than their global optimization ($k = 1$) peers, except for the warmup of MPI. However, from question **Q**2 we learned that the most robust strategy on our datasets, 1-optimism, performs as well. Since optimism does not rely on prior knowledge of k, the answer to question **Q**3 is negative.

5.4 Stopping Criterion

To evaluate the stopping criterion, we used Equation (12) to estimate $P(\exists x \in \mathcal{P} \setminus X_N : f(x) > t_{\#(k,N)})$, the probability that there exists an unseen example in the pool $\mathcal{P}$ which is better than the k-th best seen so far. We did so during one experiment with the MPI strategy for every data set, and recorded these

Table 2. p-values for $k = 25$. ϵ indicates that $p < 10^{-8}$.

Budget	10%	15%	20%	25%
MPI $k_{target} = 1$	10^{-7}	ϵ	ϵ	ϵ
MEI $k_{target} = 1$	ϵ	ϵ	ϵ	ϵ
MPI $k_{target} = 25$	**Best**	**Best**	**Best**	**Best**
MEI $k_{target} = 25$	0.487	0.184	0.394	0.640

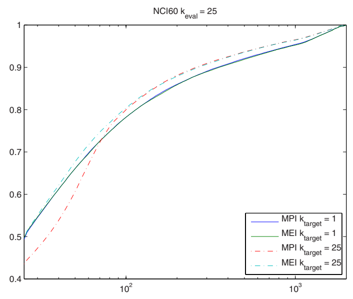


Fig. 2. The value of $\|T_N\|_{\text{best}-25}$ in each step, for the MPI and MEI strategies, optimizing for either k=1 or k=25

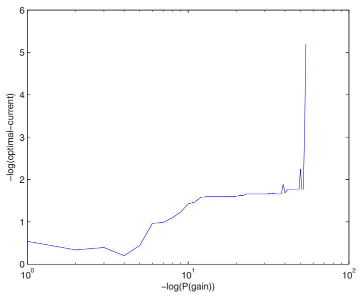


Fig. 3. Stopping criterion: negative logarithm of the difference between the optimal solution and current solution plotted against the negative logarithm of predicted probability of suboptimality according to Equation (12)

probabilities together with the differences between the solution at that point and the optimal solution. In this way we can evaluate how much value one would lose on average if one would stop the screening when the probability of finding anything better would drop below a certain threshold.

In Figure 3, the negative logarithm of the differences between solution so far and optimal solution, i.e. $-\log(\|f(\mathcal{P})\|_{\text{best}-k} - \|T_N\|_{\text{best}-k})$, is plotted against the negative logarithm of the estimated probability that there is still a better solution, i.e. $-\log(P(\exists x \in \mathcal{P} \setminus X_N : f(x) > t_{\#(k,N)}))$. The standard deviations on the points in this curve are all below 0.2.

From Figure 3 one can see that there is a good relation between the estimated probability that the best solution has not yet been found and the optimality of the current solution. In particular, when the stopping criterion predicts a very small probability of finding a better solution, one can be confident that querying more examples will not be very useful. This answers question **Q**4 positively.

6 Related Work and Possible Extensions

To summarize, we introduced the active k-optimization problem in a machine learning context, we developed an approach based on Gaussian processes to tackling it, and we applied it to a challenging structure-activity relationship prediction task, demonstrating good performance.

Our work is related to several articles that combine kernel methods and Gaussian processes both in the machine learning and the global optimization communities. In machine learning, one aims at improving prediction accuracy, and common strategies select the most uncertain examples, or select the examples that maximize information gain. In global optimization, Gaussian processes are a popular surrogate to save on expensive function evaluations [9,13].

The setting we introduced is also important for applications in HTS, where so far active learning has only been applied for classification or regression purposes but not for optimization. E.g. [5] shows that the maximum-predicted strategy works well for discriminating rare active compounds from inactives using an SVM. Furthermore, the NCI database has been used as benchmark for several machine learning approaches [18,19,20]. As the results show, classification of compounds can be learned to a certain extent, but accurate prediction (classifying borderline cases) is still harder than finding extreme values as in our setting.

Two interesting further questions for research are 1) whether one could make further gains by devising a strategy that also takes into account a budget that is fixed from the start, and 2) whether one can select several examples to be queried together in a single batch before getting the target values for all of them. This is often needed in HTS. To address the first question, one could e.g. focus the first fraction of the budget more on exploration and the last part only on exploitation. The second question requires one to spread selections over the space in order to avoid obtaining too many correlated values [17]. A few authors have touched upon this problem in the context of surrogate-based optimization, but the batch size was algorithm driven as opposed to application constraint driven, e.g. [10]. This raises a more general problem: given some collected X_N, T_N training data and a pool $\mathcal{P}$ of examples that one could query next, select n new examples to query. In such a situation, it may not be optimal to select examples that individually optimize some criterion. In the ideal case, one would like to optimize the joint contribution of the entire batch. E.g. if $k = 1$, the probability that querying examples $x_{N+1} \ldots x_{N+n}$ would improve the solution would be

$$P\big(\max\{t_{N+1} \ldots t_{N+n}\} > t_{\#(k,N)} \mid X_N, T_N\big)$$

which evaluates to the integral of a Gaussian over a union of half-spaces. For large n, it is nontrivial to select the n examples for which this value is maximized. However, one can efficiently select $x_{N+1} \ldots x_{N+n}$ in order, such that in every step the example x_{N+i} is selected that maximizes

$$P\big(\max\{t_{N+1} \ldots t_{N+i}\} > t_{\#(k,N)} \mid X_{N+i-1}, T_{N+i-1}\big)\ .$$

Acknowledgements. Kurt De Grave is supported by GOA/08/008 "Probabilistic Logic Learning". Jan Ramon is a post-doctoral fellow of the Fund for Scientific Research (FWO) of Flanders. High performance computational resources were provided by http://ludit.kuleuven.be/hpc.

References

1. King, R., et al.: Functional genomic hypothesis generation and experimentation by a robot scientist. Nature 427, 247–252 (2004)
2. Vandezande, P., et al.: High throughput screening for rapid development of membranes and membrane processes. J. Membrane Science 250(1-2), 305–310 (2005)
3. Form, N., et al.: Parameterisation of an acousto-optic programmable dispersive filter for closed-loop learning experiments. J. Modern Optics 55(1), 1–13 (2007)
4. Cohn, D., Ghahramani, Z., Jordan, M.I.: Active Learning with Statistical Models. J. Artificial Intelligence Research 4, 129–145 (1996)
5. Warmuth, M.K., et al.: Active learning with support vector machines in the drug discovery process. J. Chem. Inf. Comput. Sci. 43(2), 667–673 (2003)
6. Gibbs, M.: Bayesian Gaussian Processes for Regression and Classification. PhD thesis, University of Cambridge (1997)
7. Rasmussen, C., Williams, C.: Gaussian Processes for Machine Learning. MIT Press, Cambridge (2006)
8. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, Heidelberg (2006)
9. Sasena, M.J.: Flexibility and Efficiency Enhancements for Constrained Global Design Optimization with Kriging Approximations. PhD thesis, University of Michigan (2002)
10. Jones, D.R.: A taxonomy of global optimization methods based on response surfaces. J. Global Optimization 21, 345–383 (2001)
11. Cox, D.D., John, S.: SDO: a statistical method for global optimization. In: Multidisciplinary Design Optimization, Hampton, VA, pp. 315–329. SIAM, Philadelphia (1995)
12. Kushner, H.J.: A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. J. Basic Engineering, 97–106 (March 1964)
13. Lizotte, D., et al.: Automatic gait optimization with gaussian process regression. In: Proc. 20th Int. Joint Conference on Artificial Intelligence, pp. 944–949 (2007)
14. Jones, D.R., Schonlau, M.: Efficient global optimization of expensive black-box functions. J. Global Optimization 13(4), 455–492 (1998)
15. Shoemaker, R.: The NCI60 human tumour cell line anticancer drug screen. Nat. Rev. Cancer 6, 813–823 (2006)
16. Nishizuka, S., et al.: Proteomic profiling of the NCI-60 cancer cell lines using new high-density reverse-phase lysate microarrays. PNAS 100, 14229–14234 (2003)
17. Guestrin, C., Krause, A., Singh, A.P.: Near-optimal sensor placement in gaussian processes. In: ICML 2005, pp. 265–272 (2005)
18. Swamidass, S.J., et al.: Kernels for small molecules and the prediction of mutagenicity, toxicity and anti-cancer activity. Bioinformatics 21(suppl 1), 359–368 (2005)
19. Ceroni, A., Costa, F., Frasconi, P.: Classification of small molecules by two- and three-dimensional decomposition kernels. Bioinformatics 23(16), 2038–2045 (2007)
20. Menchetti, S., Costa, F., Frasconi, P.: Weighted decomposition kernels. In: ICML 2005, pp. 585–592 (2005)

An Efficiently Computable Graph-Based Metric for the Classification of Small Molecules

Leander Schietgat, Jan Ramon, Maurice Bruynooghe, and Hendrik Blockeel

Department of Computer Science, Katholieke Universiteit Leuven
Celestijnenlaan 200A, 3001 Leuven, Belgium
{firstname.lastname}@cs.kuleuven.be

Abstract. In machine learning, there has been an increased interest in metrics on structured data. The application we focus on is drug discovery. Although graphs have become very popular for the representation of molecules, a lot of operations on graphs are NP-complete. Representing the molecules as outerplanar graphs, a subclass within general graphs, and using the block-and-bridge preserving subgraph isomorphism, we define a metric and we present an algorithm for computing it in polynomial time. We evaluate this metric and more generally also the block-and-bridge preserving matching operator on a large dataset of molecules, obtaining favorable results.

1 Introduction

Metrics are important components of several machine learning methods. Recently, there has been an increased interest in metrics that express a similarity between structured objects. This is relevant for multiple application domains including drug discovery [1], image recognition and computer vision [2].

The application on which we focus is the classification of small molecules. An important step towards the discovery of new drugs is the identification of chemical compounds that play an active role in the regulation of biological processes or disease states. Because of the costs involved, pharmaceutical companies are interested in *virtual screening* techniques. Given a molecular database and a molecule with a desired function, these techniques select automatically a limited number of candidates expressing this function. This seriously decreases the amount of molecules that should be screened in the lab.

Since it is widely known that molecules with a similar structure tend to have the same function [3], structural similarity search among small molecules is the standard tool used for virtual screening and in silico drug development. The task then comes down to finding an appropriate similarity measure between molecules. Such a structural similarity measure should ideally fulfil two requirements: (1) it should be efficiently computable, which is important when analyzing large molecular databases, and (2) the notion of similarity should discriminate between molecules w.r.t. the activity at interest. Finding such similarity measures is one of the current challenges in chemoinformatics [1,4].

The use of graphs as a representation of molecules has become very popular [5]. Graphs are excellent representations for binary relational data: a vertex can represent an entity (e.g., an atom), while an edge indicates a relationship between two entities (e.g., a bond). However, similarity measures between graphs that aim at using all available structural information often involve the matching of subgraphs or other combinatorial operations trying to align graphs optimally. The subgraph isomorphism problem is NP-complete for general graphs [6]. For this reason, typical approaches have resorted to a transformation of molecules into vectors. In such transformations either information is lost or the size of the resulting representation can grow exponentially [4].

Nevertheless, previous theoretical work on graphs has shown that in many cases, when some constraints on the structure of the graphs hold, efficient matching algorithms exist [7,8]. Sequences, trees and outerplanar graphs (the latter are graphs which can be embedded in the plane in such a way that all of their vertices lie on the boundary of the outer face) are examples of subclasses of general graphs. Investigation on the NCI[1] database, a collection of datasets containing over 250,000 chemical compounds, revealed that only 8.8% of these graphs are trees, while 94.5% of them are outerplanar [8]. Metrics and algorithms able to deal with outerplanar graphs will thus be much more practical than those handling trees. Recently, a new "block-and-bridge preserving" matching operator (BBP) for outerplanar graphs was introduced [8]. It is based on the idea that it only makes sense to match "similar" parts of the graphs. Using this operator, frequent patterns can be mined efficiently.

The contributions of this paper are twofold. First, we present a new metric based on the ideas of the BBP matching operator. This metric has three important properties: (1) it is computable in polynomial time, (2) as it reflects the size of the maximum common connected subgraph, it has an intuitive meaning making results of methods such as instance-based learning interpretable, and (3) we show that it provides the right level of abstraction in order to discriminate between molecules. Second, we empirically investigate the practical usefulness of the BBP matching operator in general.

We start with a discussion of previous work (Section 2). In Section 3 we present the proposed metric used for the empirical study in Section 4. In Section 5 we draw conclusions and present future work.

2 Related Work

The earliest work in developing similarity measures for molecules can be found around the study of quantitative structure-activity relationships (QSAR), which uses physicochemical properties [9] or linear segments [10] to describe the molecule in a so-called fingerprint. Once these features are stored in a vector, a number of similarity measures (e.g., the Tanimoto coefficient [10]) and machine learning methods can be applied.

[1] National Cancer Institute (`http://cactus.nci.nih.gov/`).

There are two main difficulties with these approaches: (1) domain knowledge is required to select the proper descriptors, and (2) information about the molecular structure is lost. As there is a common agreement that this structure, i.e. the arrangement of the atoms and their bonds in structural space, is important when developing a similarity measure, there has been an increased interest in relational learning algorithms. One framework in which these approaches have been developed is that of inductive logic programming (ILP). These algorithms turn out to work very well on this task [11].

Actually, the ILP representation is much more general than what is needed to model chemical compounds and to discover substructures. Therefore, a number of researchers aim at achieving better performance by building systems that use special-purpose data structures for representing graphs and provide well-chosen primitives for manipulating them (e.g., [12]). In this way they transform the problem of binding chemical substructures into that of finding subgraphs in a graph. Still, finding subgraphs requires subgraph isomorphism matchings. For this reason, a two-step approach was proposed [4]: first, the complete database is searched for frequent subgraphs and then, a molecule is represented as a bit-vector that encodes the occurrences of these subgraphs in the molecule. Since the graph miner automatically selects the most interesting patterns, this approach solves the first difficulty mentioned above, while the second is only partly solved.

Following the increased emphasis on structure, graph kernels have been developed as well [13,1]. Such kernels depend mostly on a decomposition strategy, which avoids the computational complexity. Ceroni et al. [1] have introduced the weighted decomposition kernel (WDK), which obtains state-of-the-art results when used in molecule classification. It is based on a decomposition of the molecule in a selector (a single vertex) and a context (a fixed-radius subgraph surrounding the selector). By selecting an appropriate kernel for these structures, the computation of the WDK kernel remains feasible. In [1], it is also shown that better predictive performance can be obtained by combining 2D and 3D information. Here we only concentrate on the structural (2D) information.

Raymond et al. [5] give an elaborate overview of existing similarity measures that are specifically graph-based. Most of these algorithms avoid the computational complexity by computing approximate values. Raymond et al. [14] also propose an exact multi-step algorithm which defines a similarity based on computing the maximum common subgraph. This algorithm is theoretically still NP-complete, but makes use of advanced heuristics to reduce the number of matchings required.

Our method is based on this idea of graph similarity in function of the maximum common subgraph, and thus shares the intuitiveness of [14], although the latter requires an advanced graph-theoretical problem transformation and is difficult to implement. Another difference is that, by using the BBP subgraph isomorphism of [8], we can obtain a polynomial algorithm which only takes into account a subset of matchings and in this way imposes a bias on the features that will be used for computing the similarity. By using the graph structure directly, the two difficulties mentioned above are avoided.

3 A Metric Based on the Maximum Common Subgraph

Before presenting the algorithm in Section 3.3, we will give some relevant definitions (Section 3.1), and a formal problem description (Section 3.2).

3.1 Preliminaries

This section gives the relevant definitions necessary to understand the algorithm. An overview of graph theory can for example be found in [15].

In this paper, we consider undirected labeled graphs. If G is a graph, we denote with $V(G)$ the set of vertices of G and with $E(G)$ the set of edges of G. Σ is a finite set of labels and $\lambda : V \cup E \to \Sigma$ is a function assigning a label to each element of V ∪ E. The **size** $|\cdot| : \mathcal{G} \to \mathbb{R}$ of a graph is a function mapping a graph to a real number of the form $|G| = \sum_{x \in V(G) \cup E(G)} w_{\lambda_G(x)}$, where each possible label of $l \in \Sigma$ has been assigned a weight w_l.

A sequence $x_0, x_1, \ldots, x_n$ of vertices is a **path** from x_0 to x_n iff $\{x_i, x_{i+1}\} \in E(G)$, for all $i \in [0, n-1]$. A **cycle** $x_0, \ldots, x_n$ is a path such that $x_0 = x_n$. A path without repeated vertices is a simple path; a cycle without repeated vertices apart from the start and end vertex is a simple cycle. A graph G is **connected** if there is a path between any pair of its vertices; it is **biconnected** if for any two vertices u and v of G, there is a simple cycle containing u and v.

A graph is **planar** if it has a planar embedding, i.e. it can be drawn in the plane in such a way that no two edges intersect except at a vertex in common. The regions formed by the edges in a planar embedding are called faces. There is one unbounded face, which is called the outer face. A biconnected component or **block** of a graph G is a maximal subgraph of G that is biconnected. A **bridge** is an edge that does not belong to a block. An **outerplanar** graph is a planar graph which can be embedded in the plane in such a way that all of its vertices lie on the boundary of the outer face. An outerplanar graph consists entirely of blocks and bridges. We will denote the set of all outerplanar graphs with $\mathcal{G}_{op}$.

Two graphs G and H are **isomorphic** if there exists a bijection $\varphi : V(G) \to V(H)$ such that for every $u, v \in V(G)$ the following holds: (i) $\{u, v\} \in E(G)$ iff $\{\varphi(u), \varphi(v)\} \in E(H)$, (ii) $\lambda_G(u) = \lambda_H(\varphi(u))$, and (iii) if $\{u, v\} \in E(G)$ then $\lambda_G(\{u, v\}) = \lambda_H(\{\varphi(u), \varphi(v)\})$. Isomorphism is an equivalence relation on $\mathcal{G}$. We will denote the set of all equivalence classes under isomorphism with $\mathcal{G}^{\equiv}$.

A graph G is **subgraph isomorphic** to H, denoted $G \preceq H$, iff G is isomorphic to a subgraph of H. The subgraph isomorphism problem, which decides whether G is subgraph isomorphic to H is known to be NP-complete [6]; this also holds for outerplanar graphs [16].

A **block and bridge preserving (BBP) subgraph isomorphism** from G to H is a subgraph isomorphism from G to H mapping (i) the set of bridges of G to the set of bridges of H and (ii) different blocks of G to different blocks of H. We denote that a graph G is BBP subgraph isomorphic to H by $G \sqsubseteq H$. Contrary to the subgraph isomorphism problem, the BBP subgraph isomorphism problem is computable in polynomial time for outerplanar graphs [8]. For trees, which are special outerplanar graphs (i.e., block-free), the BBP subgraph isomorphism is equivalent to the subtree isomorphism.

A **common connected subgraph** I of two graphs G and H is a connected graph such that $I \preceq G$ and $I \preceq H$; it is a **maximum common connected subgraph** (MCCS) when in addition there exists no other common subgraph J, such that $I \preceq J$ and J is not isomorphic to I. Finding the MCCS between two arbitrary graphs is NP-hard [6]. However, the MCCS problem for two trees can be solved in polynomial time [6].

A **matching** f between sets A and B is a relation such that for all (a_1, b_1), $(a_2, b_2) \in f$: $a_1 = a_2$ iff $b_1 = b_2$, so each element of A is associated with at most one element of B and vice versa. A weighted maximal matching problem, also known as the assignment problem, is an optimization problem where two sets A and B are given together with a weight function $w : A \times B \to \mathbb{R}$ and the task is to find a matching m between A and B such that $\sum_{(a,b)\in m} w(a, b)$ is maximal.

A **metric** is a function $d : \Omega \times \Omega \to \mathbb{R}$ for which for any $x, y, z \in \Omega$, (i) $d(x, y) = 0 \Leftrightarrow x = y$, (ii) $d(x, y) = d(y, x)$, and (iii) $d(x, z) \leq d(x, y) + d(y, z)$.

3.2 Formal Problem Description

The goal of this paper is to develop a metric on $\mathcal{G}^{\equiv}_{op}$. Bunke and Shearer [17] proposed a distance function on graphs based on the maximum common connected subgraph: $d_{bs}(G, H) = 1 - \frac{|MCCS(G,H)|}{\max(|G|,|H|)}$, with $|G|$ equal to the number of vertices in G. They proved that d_{bs} is a metric. We can easily extend this proof to the more general case where $|G|$ is determined by the function *size*. Some variants with similar properties and performances are reviewed in [18].

As computing the MCCS can be used to decide subgraph isomorphism (G is a subgraph of H iff $|E(MCCS(G, H))| = |E(G)|$), the argument of [16] can be used to show that computing the MCCS is NP-hard even in the case of (general) outerplanar graphs. We therefore consider in this paper a maximum common connected subgraph under BBP subgraph isomorphism and show that its size can be computed efficiently. We say that I is an MCCS under BBP subgraph isomorphism of two outerplanar graphs G and H if I is a maximum connected graph for which $I \sqsubseteq G$ and $I \sqsubseteq H$, i.e. blocks are only mapped to blocks, bridges are only mapped to bridges. From an application point of view, the BBP subgraph isomorphism can be motivated from the fact that in molecules cyclic structures and linear fragments usually behave differently, and hence treating them separately might well be a good thing.

3.3 Algorithm

Our algorithm to compute the size of the MCCS of two outerplanar graphs G and H is based on a dynamic programming strategy. First, we will generate subgraphs (which we call *children*) of G and H, which will be ordered according to their size. Then, we will compute the size of the MCCS for each pair of generated children, building on the already computed solutions for pairs of smaller children. In this way, we obtain a solution for the size of the MCCS of the original graphs.

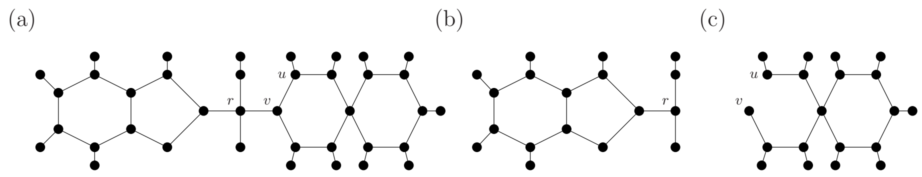


Fig. 1. (a) An outerplanar graph G^r. (b) Its NBSS G^r_v. (c) Its BSS $G|_{\curvearrowright[u,v[}$

Enumeration of Children. Given an outerplanar graph G, we define two kinds of subgraphs of G: the non-block-splitting-subgraphs and the block-splitting-subgraphs. Intuitively, the former are subgraphs in which a block is either entirely included in the subgraph or not; the latter are subgraphs which are created by splitting a block. We call these the **children** of G. It is convenient to use G^r_r as a notation for G where vertex r, the root, is distinguished from other vertices.

With $\{r, n\}$ a bridge in an outerplanar graph G, we denote with G^r_n the connected graph containing r that is obtained by removing the bridge (and hence the part reachable via the bridge). Similarly, with B a block containing vertex r we denote with G^r_B the connected graph containing r that is obtained by removing all vertices and edges from the block B except r. Each of these graphs is a **non-block-splitting-subgraph** (NBSS) of G.

A **block-splitting-subgraph** (BSS) is obtained by removing a part of a block between two vertices. Given the vertices, the orientation (clockwise or counterclockwise) determines which part is removed. More formally, with u and v two vertices in the same block of G, $G|_{o[u,v[}$ denotes the connected graph including u (and v) that is obtained by removing the vertices between v and u on the Hamiltonian cycle over all vertices of the block that follow the orientation o and by removing all edges connected to v except those belonging to the block[2]. Fig. 1 shows an outerplanar graph G, and an example of an NBSS and a BSS.

We now show how children of NBSSs and BSSs can be enumerated in a systematic way. First, consider an NBSS G^r_x (with x a vertex or a block as defined above). (i) If $\{r, n\}$ is a bridge of G^r_x, G^r_n is an NBSS child of G^r_x (the subgraph with root n that remains after removing the bridge). (ii) If B is a block in G^r_x that contains r as one of its vertices, G^r_B is an NBSS child of G^r_x. (iii) If B is a block in G^r_x that contains r as one of its vertices, has $\{r, x_0, \ldots, x_n, r\}$ as its Hamiltonian cycle (with orientation o) and has $\{r, x_i\}$ (with $i < n$) as one of its edges, $G|_{o[x_i,r[}$ is a BSS child of G^r_x. Note that there can be several BSS children. Each of them can participate in constructing the MCCS of G^r_x and some other graph. However, for a particular block B, the different BSS children are competing: only one of them can contribute to a particular MCCS.

[2] Of course an edge is also removed when one of its vertices is removed. We use the notation "$[u, v[$" to stress that the edges of u and all vertices between u and v are kept while those of v are removed. Note that $G|_{o[u,u[}$ equals G.

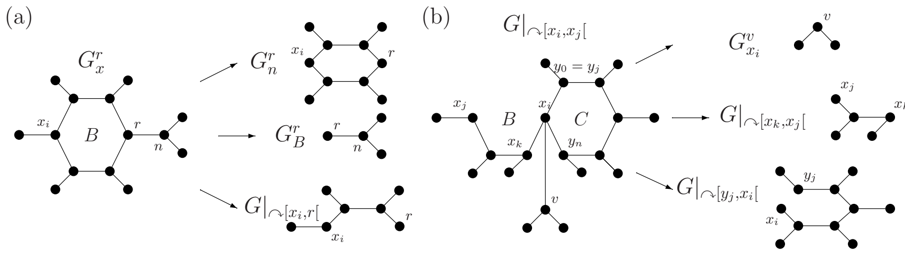

(a)
G_x^r
x_i
B
r
n
G_n^r
G_B^r
$G|_{\curvearrowright[x_i,r[}$
(b)
$G|_{\curvearrowright[x_i,x_j[}$
$y_0 = y_j$
x_j
x_k
C
y_n
v
$G_{x_i}^v$
$G|_{\curvearrowright[x_k,x_j[}$
$G|_{\curvearrowright[y_j,x_i[}$
y_j

Fig. 2. (a) Children of an NBSS G_x^r. (b) Children of a BSS $G|_{\curvearrowright[x_i,x_j[}$

Second, consider a BSS $G|_{o[x_i,x_j[}$ (splitting a block B with Hamiltonian cycle $x_0,\dots,x_n$ and $0 \le i < j < n$). (i) If $\{x_i, v\}$ is a bridge, then $G_{x_i}^v$ is an NBSS child of $G|_{o[x_i,x_j[}$. (ii) If $\{x_i, x_k\}$ is an edge with $i < k < j$, then $G|_{o[x_k,x_j[}$ is a BSS child of $G|_{o[x_i,x_j[}$. (iii) If C is a block different from B that has x_i as one of its vertices and if $x_i, y_o, \dots, y_n, x_i$ is the Hamiltonian cycle of C (with orientation o) and if $\{x_i, y_j\}$ with $j < n$ is an edge, then $G|_{o[y_j,x_i[}$ is a BSS child of $G|_{o[x_i,x_j[}$. Also here the different BSS children associated with the same block are competing for contributing to an MCCS (in an MCCS their can only be one type (ii) child and for each block C only one type (iii) child). In Fig. 2 we give some examples of NBSS and BSS children (note that only BSSs of the orientation $\curvearrowright$ are shown).

We will denote with $\mathcal{N}(G)$ the set of all NBSS children of an outerplanar graph G and with $\mathcal{B}(G)$ the set of all BSS children of G. $\mathcal{C}(G) = \mathcal{N}(G) \cup \mathcal{B}(G)$ denotes the set of all children of G. Next, we will follow a dynamic programming approach to compute the size of the MCCS for each pair of children. As we want to make sure we process all $(c_g, c_h) \in \mathcal{C}(G) \times \mathcal{C}(H)$ in increasing size, we will first order them lexicographically according to the function *size*. The base algorithm described below returns the maximal size of the MCCS containing the root vertices of both graphs; it is equal to the maximal value of the size of the MCCSs of all pairs analyzed so far.

Computing the Size of the MCCS of Two Graphs. When computing the size of the MCCS of two outerplanar graphs G and H, it suffices to compute the size of the MCCS of G_r^r with r a randomly chosen vertex from G with each graph in the set $\{H_s^s \mid s \text{ is a vertex in } H\}$ and taking the maximum (the number of pairs to consider is equal to the number of vertices in the smallest graph). When creating pairs of BSS children from a pair of NBSS graphs, it suffices to choose one orientation for the first and to consider both orientations for the second.

We will now describe how to match two children of the same type (either an NBSS of G and an NBSS of H or a BSS of G and a BSS of H). The key idea is to consider appropriate combinations of descendants of these children, and to extend their MCCSs (for which only the roots r of G_x^r and (u, v) of $G|_{o[u,v[}$ are important) into an MCCS of these children themselves. The dynamic programming approach implies that we have access to the size of the MCCS of all possible pairs of children.

More formally, an MCCS of two NBSSs G_i^r and H_j^s is a maximal connected graph S_t^t for which there are two BBP subgraph isomorphism mappings $\varphi_G : S_t^t \rightarrow G_i^r$ and $\varphi_H : S_t^t \rightarrow H_j^s$ such that $\varphi_G(t) = r$ and $\varphi_H(t) = s$. An MCCS of two BSSs $G|_{o_g[u_g,v_g[}$ and $H|_{o_h[u_h,v_h[}$ is a maximal connected graph $S|_{o_s[u_s,v_s[}$ for which there are two BBP subgraph isomorphism mappings $\varphi_G : S|_{o_s[u_s,v_s[} \rightarrow G|_{o_g[u_g,v_g[}$ and $\varphi_H : S|_{o_s[u_s,v_s[} \rightarrow H|_{o_h[u_h,v_h[}$ such that $\varphi_G(u_s) = u_g$, $\varphi_g(v_s) = v_g$, $\varphi_h(u_s) = u_h$ and $\varphi_h(v_s) = v_h$.

In order to find the size of an MCCS of two NBSSs G_i^r and H_j^s (whose roots have the same label), we make a weighted maximal matching between the set of children of G_i^r and the set of children of H_j^s (matching BSSs with BSSs and NBSSs with NBSSs, and checking that the connecting edges have identical labels) using Munkres' algorithm [19]. In order to find the size of an MCCS of two BSSs $G|_{o^G[x_i,r[}$ and $H|_{o^H[y_j,s[}$ (splitting a block B_G and a block B_H respectively), where r and s have the same label, the MCCS size is the result of the best matching between the MCCS size of the children $G|_{o^G[x_i,x_k[}$ and $H|_{o^H[y_j,y_l[}$ (in which the attached NBSSs are also matched). If $\{r, x_i\}$ and $\{s, y_j\}$ are the only remaining edges of both blocks and they have identical labels, $w_{\lambda(\{x_i,r\})}$ is added to the MCCS size.

One can show that this algorithm works correctly by verifying for every newly computed size of the MCCS of two given relevant subgraphs that this value is a correct increment of the size of the MCCS of their earlier computed children. One can also prove the following theorem:

Theorem 1. *There is an algorithm computing the size of the maximum common subgraph under BBP subgraph isomorphism of two given outerplanar graphs in time* $O(|V(G)|^{5/2}.|V(H)|^{5/2})$.

PROOF SKETCH: The result follows from counting the number of relevant graphs to consider and using known bounds on the running times of the algorithms used in the dynamic programming step (e.g., a maximal matching can happen in cubic time [19]). Due to space restrictions, we omit a full proof of the correctness and the time complexity of the algorithm.

4 Experiments

In this section, we want to answer the following questions:

1. How does the predictive performance of the BBP-based metric compare to state-of-the-art methods? (Q1)
2. Is it possible to boost performance of other classification methods when using the BBP subgraph isomorphism as matching operator? (Q2)

In order to answer Q1, we will compare the BBP-based metric to a metric based on 2D fingerprints [10] and a metric based on the WDK kernel [1], which we will evaluate in instance-based learning (IBL). As for Q2, since the predictive performance of the BBP matching operator introduced in [8] has never been investigated before, evaluating the BBP matching operator is interesting in its own. Therefore, we will compare different matching operators when used in support vector machines (SVMs) [20], a method that has become very popular for the classification of molecules [1,4,13]. The purpose of this experiment is investigating whether the BBP subgraph isomorphism has a larger predictive power compared to the general one, rather than obtaining state-of-the-art results.

4.1 Datasets

The NCI dataset has been made publicly available by the National Cancer Institute and provides screening results for the ability of more than 70,000 compounds to suppress or inhibit the growth of a panel of 60 human tumour cell lines. The datasets used here correspond to the parameter GI_{50}, the concentration that causes 50% growth inhibition. For each cell line, approximately 3,500 compounds are provided together with information on their cancer-inhibiting action, which defines a binary classification problem. We use the datasets of Swamidass et al. [13]; they are available from these authors upon request. From these datasets we have removed the non-outerplanar examples ($\sim$10%).

4.2 Method

First, we want to compare the performance of different metrics. Next to the proposed BBP-based metric (in which $w_{\lambda_G(x)}$ is set to 1 if $x \in V(G)$, 0 if $x \in E(G)$, defining the size of a graph as its number of vertices), we constructed a metric based on 2D fingerprints and a metric based on the WDK kernel. For each molecule, we constructed a 1024 FP2-fingerprint using OpenBabel v2.1.1[3] and we defined a metric on these fingerprints using the Tanimoto coefficient, which is still considered to produce state-of-the-art results for virtual screening [10]. Although the WDK kernel was not intended to be used in this way, for reasons of comparison we defined a metric according to the following formula: $d^2(x, y) = \kappa(x, x) - 2\kappa(x, y) + \kappa(y, y)$, with κ the WDK kernel function.

We used the 3 metrics in instance-based learning (IBL): in order to classify a given molecule, we selected the k neighbor molecules that are closest according to the metric (we chose $k = 11$, which was optimal for all metrics). For each molecule, we obtained a prediction equal to the percentage of positive votes of its neighbors. In this way, we can rank the predictions and compute the area under the ROC curve (AUROC). We used leave-one-out cross-validation.

For the second experiment involving SVMs, we have transformed molecules into bit-vectors using two approaches. The first approach generates frequent outerplanar subgraphs based on the BBP subgraph isomorphism using the mining algorithm FOG presented in [8]. The second approach generates frequent subgraphs based on the general subgraph isomorphism, using an efficient implementation [21] of the gSpan algorithm [12]. In both cases, the bit-vector encodes the occurrence of the frequent patterns. Since the earlier generated 2D fingerprints are also bit-vectors, we added them to the comparison. These vectors contain

[3] `http://openbabel.sourceforge.net`

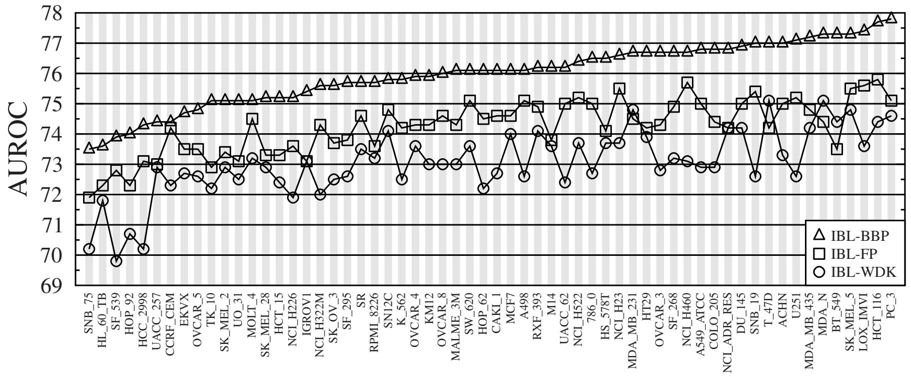


Fig. 3. Comparison of the performance of the IBL classifiers on NCI60

1024 features, and we have aimed at obtaining a similar number of features by selecting an appropriate frequency threshold for the mining algorithms (for FOG: 4% leading to 1376 patterns; for gSpan: 5% leading to 1292 patterns). Next to the propositionalization approaches, we have also compared to the WDK kernel, which we have run on our data using the settings provided by the authors of [1].

We used the SVM^{light} implementation [20]. For all methods we used exactly the same settings, which involved applying a polynomial kernel of degree 2 (used in [1]), using a 10-fold stratified cross-validation. For each fold, we have tuned the C parameter by holding out a development set from each training fold of the cross-validation. Finally, we have combined the predictions from each test fold, ranked all the predictions and computed again the AUROC.

4.3 Results

In Fig. 3 we have plotted the AUROC of the IBL classifiers. Under the null hypothesis that IBL-BBP is not better than the other classifiers, we expect an equal number of wins and losses. Instead, we found that IBL-BBP performed consistently better (60 wins out of 60) than the other two methods. Given these results[4], we can reject the null hypothesis with great confidence, and we conclude that the BBP-based metric is the best metric for IBL on molecular datasets.

Fig. 4 shows a similar comparison between the SVM-based classifiers. If we compare SVM-BBP to SVM-gSpan, we can conclude that the features generated by the BBP matching operator have a larger predictive power. This can be explained by the fact that SVM-BBP uses a more constrained language, leading to less redundant patterns. Next, SVM-BBP is also significantly better than SVM-WDK (48 wins/9 losses/3 ties). Finally, SVM-BBP performs equally well

[4] Generalization over cell lines follows from the win/loss-ratio; generalization to other molecules from the same population follows from the fact that the average AUROC is significantly better at the 1% level for samples of 3,500 molecules.

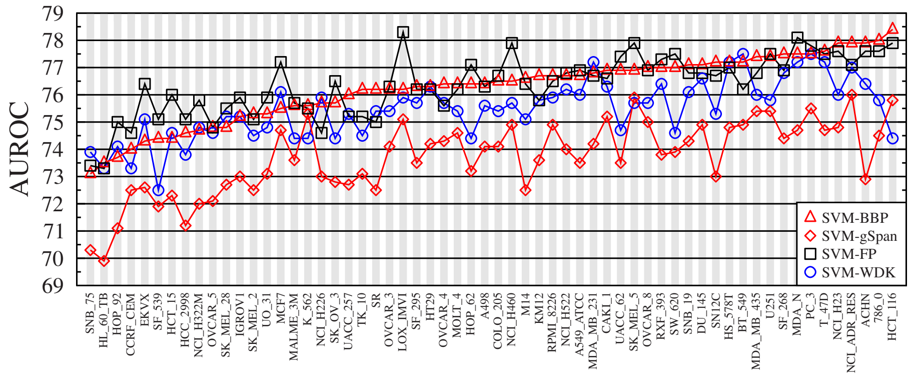


Fig. 4. Comparison of the performance of the SVM classifiers on NCI60

as SVM-FP (32 wins/28 losses). Additional experiments show that it is still possible to boost the performance of SVM-BBP and SVM-gSpan by lowering the support threshold (and in this way obtaining more patterns), but this does not change the above conclusions.

In summary, we have shown that, on these datasets, our metric outperforms previously published results, and more generally that the BBP matching operator shows good performance when used to generate features (instead of when used for a metric). Therefore, depending on the situation (e.g., the number of examples) and the user's preferences (e.g., the interpretability of the predictions), either a BBP-metric (IBL) or BBP-generated features can be good choices.

5 Conclusions and Further Work

In this paper, we have introduced a polynomial-time algorithm computing the MCCS of two outerplanar graphs under the BBP subgraph isomorphism. Second, we have investigated the performance of the BBP matching operator.

It turns out that this BBP-based metric outperforms previously published methods. One reason may be that dealing differently with cycles and linear fragments makes sense in chemical applications. Moreover, it is more intuitive than other metrics and graph kernels and although it uses the original graph structure, it is still efficiently computable. We can conclude that BBP subgraph isomorphism is an interesting matching operator for small molecules.

Since a fraction of the molecules in the NCI database cannot be represented by outerplanar graphs, it is useful to investigate how the ideas of the BBP matching operator can be extended to non-outerplanar graphs. In further future work, we plan to investigate other classes of graphs which would be suitable to represent molecules (e.g., graphs having a bounded treewidth) and for which we can design polynomial algorithms.

Acknowledgements. L.S. is supported by a PhD grant of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT). J.R. and H.B. are post-doctoral fellows of the Fund for Scientific Research (FWO) of Flanders. We thank Fabrizio Costa for valuable discussions.

References

1. Ceroni, A., Costa, F., Frasconi, P.: Classification of small molecules by two- and three-dimensional decomposition kernels. Bioinformatics 23(16), 2038–2045 (2007)
2. Shearer, K., Bunke, H., Venkatesh, S.: Video indexing and similarity retrieval by largest common subgraph detection using decision trees. Pattern Recognition Letters 34(5), 1075–1091 (2001)
3. Johnson, M., Maggiora, G.: Concepts and Applications of Molecular Similarity. John Wiley, Chichester (1990)
4. Deshpande, M., Kuramochi, M., Wale, N., Karypis, G.: Frequent substructure-based approaches for classifying chemical compounds. IEEE Transactions on Knowledge and Data Engineering 17(8), 1036–1050 (2005)
5. Raymond, J., Willett, P.: Maximum common subgraph isomorphism algorithms for the matching of chemical structures. Computer-Aided Molecular Design 16, 521–533 (2002)
6. Garey, M.R., Johnson, D.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman and Co., New York (1979)
7. Mitchell, S.L.: Linear algorithms to recognize outerplanar and maximal outerplanar graphs. Information Processing Letters 9(5), 229–232 (1979)
8. Horváth, T., Ramon, J., Wrobel, S.: Frequent subgraph mining in outerplanar graphs. In: Proceedings of the 12th ACM SIGKDD, pp. 197–206 (2006)
9. Hansch, C., Maolney, P., Fujita, T., R.M.: Correlation of biological activity of phenoxyacetic acids with hammett substituent constants and partition coefficients. Nature 194, 178–180 (1962)
10. Willett, P.: Similarity-based virtual screening using 2D fingerprints. Drug Discovery Today 11(23/24), 1046–1051 (2006)
11. King, R., Muggleton, S., Srinivasan, A., Sternberg, M.: Structure-activity relationships derived by machine learning: The use of atoms and their bond connectivities to predict mutagenicity by inductive logic programming. PNAS 93, 438–442 (1996)
12. Yan, X., Han, J.: gSpan: Graph-based substructure pattern mining. In: Proceedings of the IEEE Int. Conf. on Data Mining, pp. 721–724. IEEE Computer Society, Los Alamitos (2002)
13. Swamidass, S.J., Chen, J., Bruand, J., Phung, P., Ralaivola, L., Baldi, P.: Kernels for small molecules and the prediction of mutagenicity, toxicity and anti-cancer activity. Bioinformatics 21(suppl_1), 359–368 (2005)
14. Raymond, J., Gardiner, E., Willett, P.: Rascal: Calculation of graph similarity using maximum common edge subgraphs. Computer Journal 45, 631–644 (2002)
15. Diestel, R.: Graph Theory. Springer, Heidelberg (2000)
16. Syslo, M.: The subgraph isomorphism problem for outerplanar graphs. Theoretical Computer Science 17(1), 91–97 (1982)
17. Bunke, H., Shearer, K.: A graph distance metric based on the maximal common subgraph. Pattern Recognition Letters 18, 689–694 (1997)
18. Raymond, J., Willett, P.: Effectiveness of graph-based and fingerprint-based similarity measures for virtual screening of 2D chemical structure databases. Journal of Computer-Aided Design 16, 59–71 (2002)

19. Munkres, J.: Algorithms for the assignment and transportation problems. Journal of the Society for Industrial and Applied Mathematics 5, 32–38 (1957)
20. Joachims, T.: Learning to Classify Text using Support Vector Machines: Methods, Theory, and Algorithms. Springer, Heidelberg (2002)
21. Bringmann, B., Zimmermann, A., De Raedt, L., Nijssen, S.: Don't be afraid of simpler patterns. In: Proc. of the 10th European Conf. on Principles and Practice of Knowledge Discovery in Databases, pp. 55–66 (2006)

Mining Intervals of Graphs to Extract Characteristic Reaction Patterns

Frédéric Pennerath[1,3], Géraldine Polaillon[2], and Amedeo Napoli[3,⋆]

[1] Supélec, Campus de Metz, 2 rue Edouard Belin 57070 Metz, France
frederic.pennerath@supelec.fr
[2] Supélec, Campus de Gif-sur-Yvette, 3 rue Joliot-Curie 91192 Gif-sur-Yvette, France
geraldine.polaillon@supelec.fr
[3] Orpailleur team, LORIA, BP 239, 54506 Vandoeuvre-lès-Nancy Cedex, France
amedeo.napoli@loria.fr

Abstract. The article introduces an original problem of knowledge discovery from chemical reaction databases that consists in identifying the subset of atoms and bonds that play an effective role in a given chemical reaction. The extraction of the resulting *characteristic reaction pattern* is then reduced to a graph-mining problem: given lower and upper bound graphs g_l and g_u, the *search of best patterns in an interval of graphs* consists in finding among connected graphs isomorphic to a subgraph of g_u and containing a subgraph isomorphic to g_l, best patterns that maximize a scoring function and whose score depends on the frequency of the pattern in a set of examples. A method called `CrackReac` is then proposed to extract best patterns from intervals of graphs. Accuracy and scalability of the method are then evaluated by testing the method on the extraction of characteristic patterns from reaction databases.

1 Introduction

A possible model for representing molecules is to use molecular graphs: a *molecular graph* is a connected labeled graph whose vertices represent atoms labeled by their chemical element (C for carbon ...) and whose edges represent bonds labeled by their type (single, double, triple, or aromatic). *Chemical reactions* (or simply *reactions*) are physical transformations of molecular structures that consist in breaking, creating, or modifying types of bonds. A reaction can be represented by a *chemical equation*, as illustrated on Fig. 1. In order to setup new synthesis pathways, chemists use knowledge of reactions they learned from experiments. This knowledge consists of generic *synthesis methods* that can be applied to new synthesis contexts. Instances of a synthesis method are reactions that share a common *reaction pattern* along with specific environmental conditions. Figure 2(a) shows the reaction pattern of the synthesis method whose reaction of Fig. 1 is an instance of. Discovering new synthesis methods and organizing knowledge about known methods are two key issues for the organic

⋆ Authors wish to thank Gilles Niel at ENSCM in Montpellier, France that has selected synthesis methods and designed datasets.

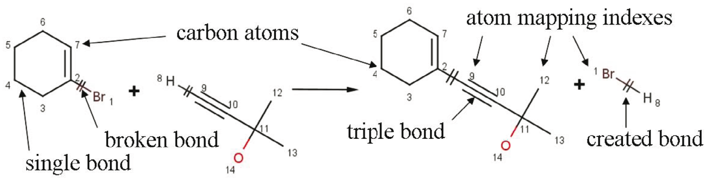
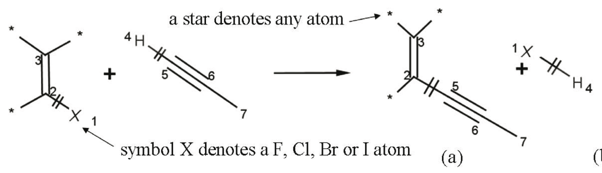


Fig. 1. A chemical equation of type $m_1 + m_2 \rightarrow m_3 + m_4$



(b)

n	$\theta_{R/P}(n)$
1	1
2	2
3	7
4	8
5	9
6	10

Fig. 2. Reaction pattern of the Sonogashira synthesis method (a) and atom mapping (b) of pattern with equation of Fig. 1

synthesis industry. Both tasks could be eased by knowledge discovery systems as reaction databases already describe millions of molecule synthesis. For this reason the underlying problem of reaction clustering has been a major research topic of computational chemistry (cf for instance [1,2,3]).

The present article introduces an original problem of knowledge discovery closely related to reaction clustering: the problem consists in identifying the subset of bonds and atoms that play an effective role in the course of a reaction. This subset forms the so called *characteristic pattern* of the reaction and represents a potential synthesis method applicable to other synthesis problems. This application is then reduced to a graph-mining problem: given lower and upper bound graphs g_l and g_u, the *search of best patterns in an interval of graphs* consists in finding among connected graph patterns contained in g_u and containing g_l, best patterns that maximize a scoring function and whose score depends on the frequency of the pattern in a set of examples.

This problem is related to *frequent graph-mining* that consists in determining the set of connected graph patterns that occur a minimum number of times in a set of examples of graphs. As existing frequent graph-mining algorithms [4,5,6,7] do not integrate the interval constraint, the article proposes a new method called `CrackReac` to address the problem of mining graph patterns bounded by an interval. While the method borrows some notions and data structures from frequent graph-mining algorithms, like embedding lists [7], the method is closer in its principles to selective pattern searching algorithms like `Subdue` [8], where the search is not exhaustive but rather the purpose is to output a limited number of informative patterns. Compared to existing graph searching algorithms, `CrackReac` brings an additional constraint as patterns are taken within intervals of graphs. This not only simplifies the pattern enumeration algorithm but also introduces a structural constraint on score maximization.

Main contributions of the article are an original graph-mining problem along with its application to chemistry. Whereas graph-mining (cf survey [9]) and Inductive Logic Programming (e.g [10,11]) have already been applied to molecular graphs, the article is one of the first, to the best of the authors'knowledge, to describe an application of graph-mining to chemical reaction databases. To this end, the article first formalizes in Sect. 2 the problem of extracting characteristic patterns of reactions and provides application-specific properties that are needed to reduce the problem to the search of best patterns in an interval of graphs. Section 3 defines the problem of searching best patterns in an interval of graphs, provides an example of scoring function and gives the outlines of the `CrackReac` method so that the problem of extracting characteristic patterns of reactions gets solvable. Section 4 details results of experiments on reaction databases run with the previous method and scoring function.

2 The Extraction of Characteristic Patterns of Reactions

Before developing the problem further, some common definitions are recalled: a *labeled graph* g is defined by a set $V(g)$ of vertices and a set $E(g)$ of pairs of vertices called edges. In addition, every vertex or edge is mapped to a label taken from a given set. Two labeled graphs g_1 and g_2 are *isomorphic* if there exists a bijection $\mu : V(g_1) \rightarrow V(g_2)$ that preserves vertex/edge labeling and adjacency (i.e $\{v_1; v_2\} \in E(g_1) \Leftrightarrow \{\mu(v_1); \mu(v_2)\} \in E(g_2)$). A graph g_1 is a *subgraph* of g_2 (denoted $g_1 \subseteq g_2$) if $V(g_1) \subseteq V(g_2)$ and $E(g_1) \subseteq E(g_2)$. A graph g_1 is *included* in graph g_2 wrt the *isomorphic subgraph relation*, which we denote by $g_1 \subseteq_G g_2$, if there exists a subgraph of g_2 that is isomorphic to g_1. A (closed) *interval of graphs* $[g_l; g_u]$ is the set of graphs g comprised between a *lower bound graph* g_l and an *upper bound graph* g_u: $[g_l; g_u] = \{g | g_l \subseteq_G g \subseteq_G g_u\}$. Given a dataset $\mathcal{D}$ of labeled graphs, the (relative) *frequency* of a graph g is the proportion of graphs g' in $\mathcal{D}$ such that g is included in g' wrt $\subseteq_G$. A graph pattern is *frequent* relatively to a threshold f_{min} if its frequency is larger or equal than f_{min}.

2.1 Data Representation

Reaction databases specify reactions by their reaction patterns:

Definition 1. *A* reaction pattern *is a 4-uplet* $(\mathcal{R}, \mathcal{P}, \lambda_\mathcal{R}, \lambda_\mathcal{P})$ *where connected components of graph* $\mathcal{R}$ *(resp.* $\mathcal{P}$*) are molecular graphs of initial (resp. resulting) molecules of a molecule transformation and where indexing functions* $\lambda_\mathcal{R} : V(\mathcal{R}) \rightarrow \mathbb{N}$ *and* $\lambda_\mathcal{P} : V(\mathcal{P}) \rightarrow \mathbb{N}$ *locate the same atom in* $\mathcal{R}$ *and* $\mathcal{P}$*: vertices* v_1 *in* $\mathcal{R}$ *and* v_2 *in* $\mathcal{P}$ *represent the same atom if and only if* $\lambda_\mathcal{R}(v_1) = \lambda_\mathcal{P}(v_2)$*.*

Atom mappings are specified on Fig. 1 by indexes next to each atom. The pattern of a reaction is called a *chemical equation.* Chemists also use reaction patterns to represent families of reactions thanks to a subsumption relation:

Definition 2. *Reaction pattern* $P_1 = (\mathcal{R}_1, \mathcal{P}_1, \lambda_{\mathcal{R}_1}, \lambda_{\mathcal{P}_1})$ subsumes *reaction pattern* $P_2 = (\mathcal{R}_2, \mathcal{P}_2, \lambda_{\mathcal{R}_2}, \lambda_{\mathcal{P}_2})$ *and is denoted by* $P_2 \sqsubseteq_R P_1$*, if and only if* $\mathcal{R}_1 \subseteq_G$

$\mathcal{R}_2$ and $\mathcal{P}_1 \subseteq_G \mathcal{P}_2$ and if associated injective morphisms $\theta_\mathcal{R} : V(\mathcal{R}_1) \to V(\mathcal{R}_2)$ and $\theta_\mathcal{P} : V(\mathcal{P}_1) \to V(\mathcal{P}_2)$ are consistent with atom mapping functions:

$$\forall s_1 \in V(\mathcal{R}_1), \forall s_2 \in V(\mathcal{P}_1), \lambda_{\mathcal{R}_1}(s_1) = \lambda_{\mathcal{P}_1}(s_2) \Leftrightarrow \lambda_{\mathcal{R}_2}(\theta_\mathcal{R}(s_1)) = \lambda_{\mathcal{P}_2}(\theta_\mathcal{P}(s_2))$$

Reaction pattern of Fig. 2(a) subsumes chemical equation of Fig. 1 wrt to atom mapping of Fig. 2(b). Reaction patterns can be equivalently represented by condensed reaction graphs introduced by Vladutz [12]:

Definition 3. *The* reaction graph $\mathcal{G}(P)$ *of a reaction pattern* $P = (\mathcal{R}, \mathcal{P}, \lambda_\mathcal{R}, \lambda_\mathcal{P})$ *is equal to the graph $\mathcal{R}$ where bonds created by the reaction (i.e in $E(\mathcal{P})$ but not in $E(\mathcal{R})$) have been added with respect to atom mappings and where every bond of $\mathcal{G}(P)$ has been labeled by the ordered pair $(l_\mathcal{R}, l_\mathcal{P})$ of labels of mapped bonds in $\mathcal{R}$ and $\mathcal{P}$ (with a special label* 0 *to denote non existing bond in $\mathcal{R}$ or $\mathcal{P}$).*

For instance, Fig. 3 gives the reaction graph of reaction pattern of Fig. 1. The representation model of reactions graphs is equivalent to the model of reaction patterns as the transformation is reversible. While reaction graphs have initially be introduced as a compact representation of reactions, reaction graphs hold interesting properties within the context of graph-mining:

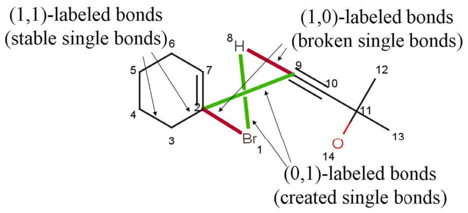


Fig. 3. Reaction graph of equation of Fig. 1

Property 1. *The order of reaction patterns relatively to $\sqsubseteq_R$ is equivalent to the order of equivalent reaction graphs ordered by the isomorphic subgraph relation $\subseteq_G$: $P_1 \sqsubseteq_R P_2 \Leftrightarrow \mathcal{G}(P_2) \subseteq_G \mathcal{G}(P_1)$.*

Property 2. *Reaction graphs are connected graphs.*

Because reaction graphs are fully compliant with graph-mining requirements (i.e connected and ordering relation of interest is $\subseteq_G$), the reaction patterns are subsequently modeled by their equivalent reaction graphs.

2.2 Problem Description

Every reaction is characterized by the pattern of its bond transformation, called the center of reaction: the *center* $P_c(E)$ of equation E is the reaction pattern made of bonds broken, created, or modified during the reaction along with incident atoms. The *reaction graph of the center* is denoted $G_c(E) = \mathcal{G}(P_c(E))$. Center of reaction of Fig. 1 is given on Fig. 4. Chemists have observed that, given

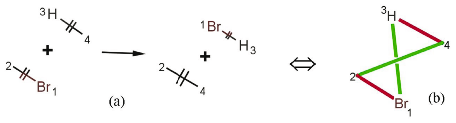


Fig. 4. Center (a) and reaction graph of the center (b) of reaction of Fig. 1

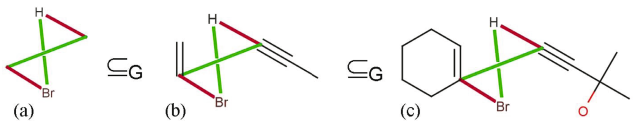


Fig. 5. The interval constraint: $G_c(E)$ (a), $G_m(E)$ (b) and $\mathcal{G}(E)$ (c)

a reaction of equation E, a set of molecules presenting the same topological environments as the ones that atoms of $P_c(E)$ have in the initial molecules of E, generally reacts according to the transformation pattern of $P_c(E)$, all other reaction conditions (e.g catalysts, solvents ...) being equal. The minimal environment of the center $P_c(E)$ that is required for the transformation to occur defines a reaction pattern called here a *characteristic reaction pattern* and denoted $P_m(E)$. Chemists search for these generic reaction patterns along with the required reaction conditions in order to reuse them as *synthesis methods* in new synthesis problems. As the characteristic pattern of a reaction necessarily contains the center of the reaction, the pattern $P_m(E)$ and thus the *characteristic reaction graph* $G_m(E) = \mathcal{G}(P_m(E))$ are constrained to be within intervals:

Property 3. *$E \sqsubseteq_R P_m(E) \sqsubseteq_R P_c(E)$ or equiv. $G_c(E) \subseteq_G G_m(E) \subseteq_G \mathcal{G}(E)$*

The interval of graphs associated to the example of Fig. 1 is given on Fig. 5. The motivation of the present work is to extract from the equation E of a reaction its characteristic reaction pattern $P_m(E)$, by mining reaction patterns in a dataset $\mathcal{D}$ of reaction graphs. For example, reaction pattern of Fig. 2(a) is to be extracted from equation of Fig. 1. Intuitively given an equation E of a reaction, the more a reaction pattern P subsuming E wrt $\sqsubseteq_R$ is simultaneously significant by its size and frequent in $\mathcal{D}$, the more likely P is to be characteristic of the synthesis method underlying E. However the larger P is, the less frequent it is in the dataset $\mathcal{D}$ so that the problem appears as finding the best compromise between size and frequency of patterns. Assuming the quality of the compromise can be assessed by an heuristic scoring function s that maps a reaction graph G to a score $s(G)$, the problem consists in searching $G_m(E)$ as the reaction graph of highest score within the interval $[G_c(E); \mathcal{G}(E)]$.

3 The Search of Best Patterns in Intervals of Graphs

3.1 Problem Definition

The previous section leads to the following graph searching problem:

Definition 4. *Given a list* $((g_{l_i}, g_{u_i}))$ *of pairs of lower and upper bound graphs and given a scoring function* s*, the* search of best patterns in intervals $[g_{l_i}, g_{u_i}]$ of graphs *consists in finding for every index* i *the* best graph pattern g_{best_i} *that maximizes* $s(g)$*, provided that graph* g *is connected and* $g_{l_i} \subseteq_G g \subseteq_G g_{u_i}$.

Various scoring functions can be considered in the context of the considered application. The present article focuses on a scoring function similar to the one used by `Subdue` [8]. This function is based on the *Minimum Description Length* principle that considers learning as a problem of compressing reality into models: given a graph pattern g as a model to describe a dataset $\mathcal{D}$, `Subdue` contracts every occurrence of g in $\mathcal{D}$ by a specific vertex. The best compressing pattern g is the one that maximises the saved space roughly equal to product $|g| \cdot n(g)$ of the size $|g|$ (e.g number of vertices and edges) of g by the number $n(g)$ of occurrences of g in $\mathcal{D}$. Similarly, the better a reaction pattern describes equations of the same reaction center, the more likely this pattern is characteristic of a synthesis method. This leads to the following definition of scoring function:

Definition 5. *The function* s_i *of a graph* g *relatively to a dataset* $\mathcal{D}$ *is:*

$$s_i(g, \mathcal{D}) \mapsto (I(g) - I(g_l)) \cdot f(g, \mathcal{D}) \text{ with}$$

$$I(g) = \sum_{v \in V(g)} i(l_v(v)) + \sum_{e \in E(g)} i(l_e(e)) \text{ where } i(l) = -\log_2 \left(\frac{n(l)}{\sum\limits_{l' \in \mathcal{L}_v \cup \mathcal{L}_e} n(l')} \right)$$

and where $f(g, \mathcal{D})$ *is the relative frequency of pattern* g *in dataset* $\mathcal{D}$ *wrt* $\subseteq_G$ *,*g_l *is the minimal graph that is known to be contained in* g *wrt* $\subseteq_G$*,* $I(g)$ *is the sum of information carried by every vertex* $v \in V(g)$ *of label* $l_v(v)$ *and by every edge* $e \in E(g)$ *of label* $l_e(e)$*,* $i(l)$ *is the information of label* l*, and* $n(l)$ *is the number of vertices or edges labeled by* l *in dataset* $\mathcal{D}$*.*

Compared to the function $|g| \cdot n(g)$ used in [8], the function s_i takes into account two additional effects: first the replacement of pattern size factor $s(g)$ by pattern information $I(g)$ increases the weight of naturally unfrequent labels. Second the term $I(g)$ is replaced by $I(g) - I(g_l)$ also equal to the information gain $I(g|g_l \subseteq_G g)$ as g is already known to contain g_l.

3.2 A First Solution Based on Existing Graph-Mining Algorithm

Given a list $(I_i) = ((g_{l_i}; g_{u_i}))$ of lower and upper bound graphs and given a scoring function s whose score $s(g)$ depends on the frequency of pattern g in a dataset $\mathcal{D}$ of graphs, this section shows how to compute the best pattern g_{best_i} of each interval I_i using any existing frequent graph-mining algorithm.

One obvious solution to search best patterns in an interval $[g_{l_i}, g_{u_i}]$ of graphs is to produce every connected graph pattern of frequency (at least) 1 in the singleton dataset $\{g_{u_i}\}$ using an existing frequent graph miner, then to remove patterns that do not contain a subgraph isomorphic to g_{l_i} and finally to compute frequency in $\mathcal{D}$ and then score of every remaining pattern. However this method does not work as a typical reaction graph has millions of subgraphs (e.g the small reaction graph of Fig. 3 has already about 685000 subgraphs). A better method is, given an arbitrary threshold f_{min}, to:

1. Partition the set of intervals $\mathcal{I} = \{I_i\}$ of graphs into parts $\mathcal{I}_j = \{(g_{l_j}, g_{u_{ij}})\}$ so that every interval in set $\mathcal{I}_j$ shares the same lower bound graph g_{l_j}.
2. For each part $\mathcal{I}_j$, extract from $\mathcal{D}$ the subset $\mathcal{D}_j$ of graphs that contain a subgraph isomorphic to g_{l_j}.
3. Mine frequent graphs $\mathcal{F}_j$ of $\mathcal{D}_j$ whose relative frequency is at least f_{min}.
4. Extract subset $\mathcal{F}'_j$ of $\mathcal{F}_j$ of graph patterns that contain at least one subgraph isomorphic to g_{l_j}.
5. Compute for each pattern $g \in \mathcal{F}'_j$ of frequency $f(g)$ its score $s(g, f(g))$.
6. For each upper bound graph $g_{u_{ij}}$ in $\mathcal{I}_j$, extract from $\mathcal{F}'_j$ the subset $\mathcal{F}''_{ij}$ of graph patterns that are isomorphic to a subgraph of $g_{u_{ij}}$.
7. For each interval I_i of $\mathcal{I}_j$, return pattern g_{best_i} of maximal score within $\mathcal{F}''_{ij}$.

Prefiltering of step 2 allows to keep f_{min} relatively high (e.g equal to 0.8) so that frequent graph-mining in step 3 remains tractable. However the observed performance bottleneck of the method is not step 2 but step 6, that requires for each interval to detect which of the many frequent patterns are isomorphic to subgraphs of g_{u_i}. This observation has raised the question whether searching directly the best patterns in an interval of graphs provides better performance.

3.3 The Algorithm `CrackReac` to Mine Intervals of Graphs

The main particularity of `CrackReac` is the way patterns are generated: the algorithm `CrackReac` replaces the interval constraint $g_l \subseteq_G g \subseteq_G g_u$ on pattern g by a disjunction of n constraints $g'_{l_i} \subseteq g \subseteq g_u$ where $\subseteq$ is the strict subgraph relation and where the n graphs $(g'_{l_i})_{1 \leq i \leq n}$ are all subgraphs of g_u isomorphic to g_l. This rewriting is possible by definition as for each pattern $g' \subseteq_G g_u$ there exists at least one subgraph g of g_u that is isomorphic to g'. This apparently insignificant change induces however strong consequences: generated patterns are not anymore isomorphically unique graph patterns, like with frequent graph miners, but simply connected subgraphs of g_u containing a subgraph isomorphic to g_l. The generation then gets simpler (see below) and faster (per generated pattern) than generating non isomorphic patterns as it does not require to compute any canonical representant of patterns. However the disadvantage is that distinct but isomorphic subgraphs of g_u induce duplicated frequency computations.

In practice the current pattern g is built by exploring the pattern space in a depth first search order, applying sequentially extensions to an initial pattern made of any subgraph g'_{l_i} of g_u isomorphic to g_l. Depth first search is a requirement to compute efficiently pattern frequencies using *embedding lists* as described in [7]. As pattern g must remain connected, possible extensions are either connected vertex extensions $\mathrm{CVE}(l_e, l_v, v)$ connecting a new l_v-labeled vertex to vertex v of g with a new l_e-labeled edge or edge extensions $\mathrm{EE}(l_e, v_1, v_2)$ connecting vertices v_1 and v_2 of g with a new l_e-labeled edge. Extensions of g can be efficiently enumerated by a constant time iterator: as g is a non-empty connected subgraph of g_u, every edge $\{v_1; v_2\} \in E(g_u) \setminus E(g)$ incident to g (i.e with $v_1 \in V(g)$) is in bijection with one extension $\text{ext}(\{v_1; v_2\})$ of g. Only the type of extension $\text{ext}(\{v_1; v_2\})$ varies with the current development of g: if $v_2 \in V(g)$ then $\text{ext}(\{v_1; v_2\})$ is $\text{EE}(l(\{v_1; v_2\}), v_1, v_2)$ else it is $\text{CVE}(l(\{v_1; v_2\}), l(v_2), v_1)$. The pseudo-code 1 details how the recursive procedure of `CrackReac` develops the current pattern g. The searching strategy extends g only if on line 1 current score is not less than a factor $1 - \varepsilon_{branch}$ of the best score on the current branch (i.e set of patterns along the recursive path up to g) and than a factor $1 - \varepsilon_{level}$ of the best score on the current level (i.e set of already mined patterns of the same number of edges as g). Line 2 forbids extensions whose branch has already been developed so that every valid subgraph of g_u is generated at most once.

Algorithm 1. The procedure `CrackReac` (g, s_g, M_{branch}, d)

```
   Data: Input interval $(g_l, g_u)$, dataset $\mathcal{D}$, scoring function $s$
   Input: Current subgraph $g$, score $s_g$, branch highest score $M_{branch}$ and depth $d$
   Result: Best pattern $g_{best}$ and its score $s_{best}$ are global variables
   Array $M_{level}[]$ of level maxima is a global variable ;
   Set $L \leftarrow \emptyset$ ;
   forall extension $e$ of $g$ in $G$ do
       Frequency $f \leftarrow \text{freq}(e(g), \mathcal{D})$ ; Score $s_e \leftarrow s(e(g), f)$ ;
       $L \leftarrow L \cup \{(e, s_e)\}$ ;
       $M_{level}[d] \leftarrow \max(M_{level}[d], s_e)$
   forall $(e, s_e) \in L$ do
1      if $s_e \geq \max((1 - \varepsilon_{branch}) \cdot M_{branch}, (1 - \varepsilon_{level}) \cdot M_{level})$ then
           CrackReac ($e(g)$, $s_e$, $\max(M_{branch}, s_e)$, d + 1) ;
2          disable($e$)
   forall $(e, s_e) \in L$ do
       enable($e$)
   if $s_g > s_{best}$ then
       $s_{best} \leftarrow s_g$ ; $g_{best} \leftarrow g$
```

4 Experiments

Experiments have been conducted on examples of synthesis methods of Sonogashira, asymmetric Sharpless epoxydation and aceto-acetic ester (cf patterns on Fig 6). Tests aim at assessing accuracy of the method and comparing performance of `CrackReac` with method proposed in Sect. 3.2 based on existing graph-mining methods. *Accuracy* of a function score is the property that its maxima occur for expected characteristic patterns. A function score is *robust* if it remains accurate when the variety of synthesis methods increase in the database. Accuracy is difficult to define formally from a random set of reactions as expected characteristic patterns are unknown. However it is possible to compute accuracy

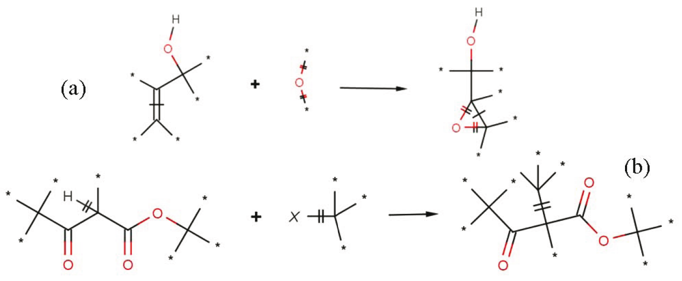


Fig. 6. Patterns of Sharpless epoxidation (a) and aceto-acetic ester synthesis (b)

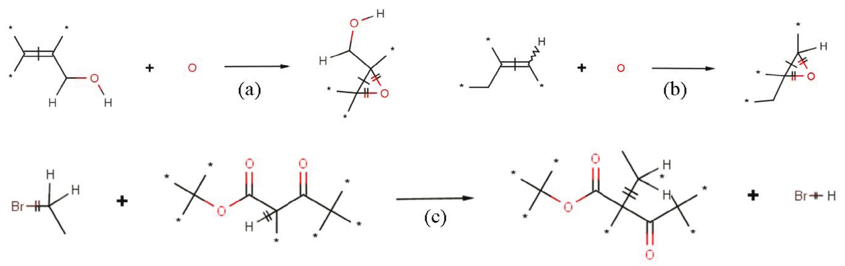


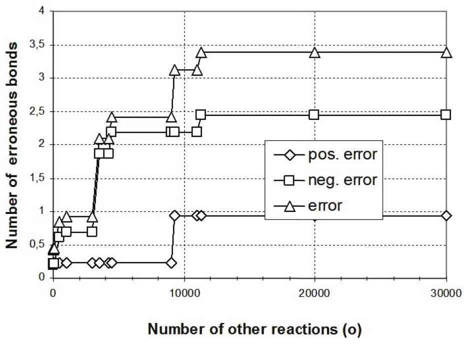
Fig. 7. Characteristic patterns of Sharpless epoxydation for $o = 500$ and $o = 10000$ (a and b) and of aceto-acetic ester synthesis for $o = 10000$ (c)

for a given synthesis method whose experts have provided the reaction pattern based on the following definition of error:

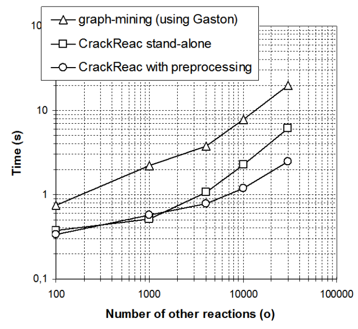
Definition 6. *Given a subgraph g of an input graph g_{input} and a reference graph g_{ref} that is assumed to be isomorphic to a number of subgraphs (g_{ref_i}) of g_{input}, the number of* false positive (resp.negative) bonds *of g relatively to g_{ref_i} is $\varepsilon_i^+ = |E(g) \setminus E(g_{ref_i})|$ (resp. $\varepsilon_i^- = |E(g_{ref_i}) \setminus E(g)|$). The* error *of g relatively to g_{ref} is then $\varepsilon(g, g_{ref}, g_{input}) = \min_i(\varepsilon_i^+ + \varepsilon_i^-)$. The* positive (resp. negative) error *is the value of ε_i^+ (resp. ε_i^-) for the graph g_{ref_i} that minimizes the error.*

Given the reaction graph g_{ref} of a given synthesis method, the accuracy of the characteristic subgraph g_{best} of g_{input} is assessed by the error $\varepsilon(g_{\text{best}}, g_{ref}, g_{input})$. Every elementary test consists in searching characteristic graphs of i input instances of a given method m from a dataset composed of e examples of method m merged to o other randomly selected reactions[1]. Number e of example reactions has been set to a low realistic value of 10 whereas number i of input graphs has been set to 100. Clustering of reactions is efficient as the 100 input reactions are distributed in average (rel. to o) over a number of 6, 3.5 and 7 characteristic patterns for each of the three synthesis methods. Positive, negative and total

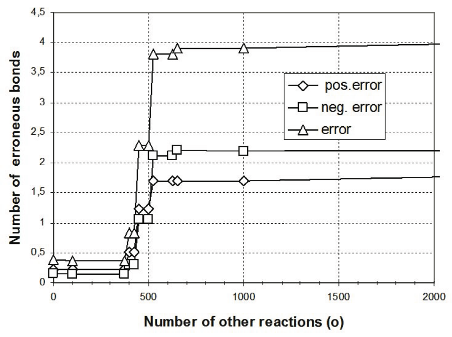
[1] Datasets can not be distributed but can be retrieved from Symyx®/MDL® commercial reaction databases *ChemInform* and *RefLib*, selecting mono-product reactions containing given synthesis method patterns and with an existing non-zero yield.



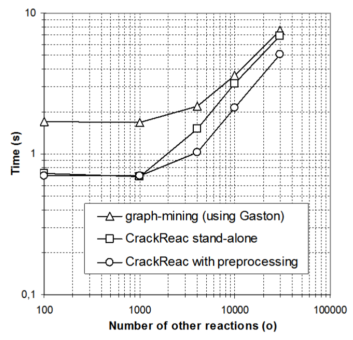
(a)



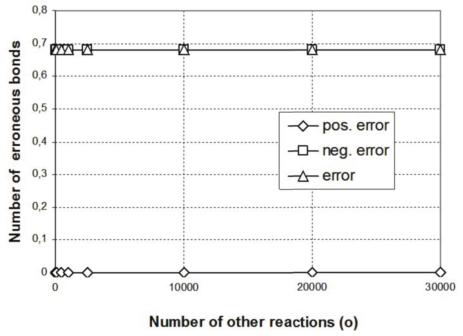
(b)



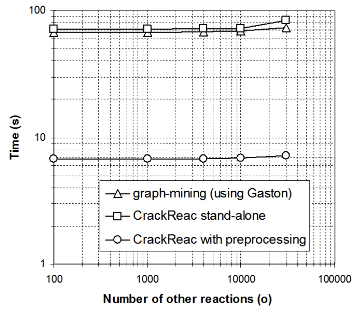
(c)



(d)



(e)



(f)

Fig. 8. Errors and processing times for Sonohashira (a and b), Sharpless epoxydation (c and d) and aceto-acetic ester synthesis (e and f)

errors are given on Fig. 8(a), Fig. 8(c) and Fig. 8(e) as functions of the number *o* of other reactions. Positive and negative errors grow step by step when the number *o* of other reactions increases. Characteristic patterns are found equal or very similar to the expected patterns for small *o* values but tend to shrink towards the center of reaction when o exceeds some thresholds as additions of other concurrent methods with the same reaction center increasingly perturbate values of scoring function: whereas the Sonogashira pattern is preserved up to 10000 other reactions, the epoxydation pattern is dismantled for $o \geq 750$ (cf Fig. 7(a) and Fig. 7(b)). Error of ester synthesis remains low and constant as no concurrent methods appear in other available reactions (cf Fig. 7(c)). From a performance perspective, Fig. 8(b), Fig. 8(d) and Fig. 8(f) (run on a T5600 1,8 Ghz for $i = 100, e = 10$) compare total processing times of `CrackReac` stand-alone, the method of Sect. 3.2, using `Gaston` [7] with $f_{min} = 0.8$, and the `CrackReac` method said with preprocessing, run on the same partitions $(\mathcal{I}_j)$ and datasets $(\mathcal{D}_j)$ of Sect. 3.2. `CrackReac` with preprocessing appears to be the fastest while it is not obvious which of the two other methods is second.

5 Conclusion

The problem of extracting characteristic patterns of reactions has lead to a graph-mining problem whose requirements do not match existing graph-mining algorithms. The proposed `CrackReac` method has given sensible results for the considered application without sacrificing speed or scalability: `CrackReac` provides the characteristic pattern of a reaction in hundredths to tenths of a second while mining thousands of reactions. However the tested scoring function provides limited robustness and suggests to search for a function that integrates domain knowledge to better reflect the notion of characteristic pattern.

References

1. Rose, J.R., Gasteiger, J.: Horace: An automatic system for the hierarchical classification of chemical reactions. Journal of Chemical Information and Computer Sciences 34(1), 74–90 (1994)
2. Hendrickson, J.B.: Comprehensive system for classification and nomenclature of organic reactions. Journal of Chemical Information and Computer Sciences 37(5), 852–860 (1997)
3. Satoh, H., Nakata, T.: Knowledge discovery on chemical reactivity from experimental reaction information. In: Grieser, G., Tanaka, Y., Yamamoto, A. (eds.) DS 2003. LNCS (LNAI), vol. 2843, pp. 470–477. Springer, Heidelberg (2003)
4. Inokuchi, A., Washio, T., Motoda, H.: An apriori-based algorithm for mining frequent substructures from graph data. In: Zighed, D.A., Komorowski, J., Żytkow, J.M. (eds.) PKDD 2000. LNCS (LNAI), vol. 1910, pp. 13–23. Springer, Heidelberg (2000)
5. Kuramochi, M., Karypis, G.: Frequent subgraph discovery. In: ICDM 2001: Proceedings of the 2001 IEEE International Conference on Data Mining, pp. 313–320 (2001)
6. Yan, X., Han, J.: Gspan: Graph-based substructure pattern mining. In: ICDM 2002: Proceedings of the 2002 IEEE International Conference on Data Mining, Washington, DC, USA, p. 721. IEEE Computer Society, Los Alamitos (2002)

7. Nijssen, S., Kok, J.N.: A quickstart in frequent structure mining can make a difference. In: KDD 2004: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 647–652. ACM Press, New York (2004)
8. Cook, D.J., Holder, L.B.: Substructure discovery using minimum description length and background knowledge. Journal of Artificial Intelligence Research 1, 231–255 (1994)
9. Fischer, I., Meinl, T.: Graph based molecular data mining - an overview. In: SMC (5), pp. 4578–4582. IEEE, Los Alamitos (2004)
10. Srinivasan, A., Muggleton, S.H., Sternberg, M.J.E., King, R.D.: Theories for mutagenicity: a study in first-order and feature-based induction. Artificial Intelligence 85(1-2), 277–299 (1996)
11. Dehaspe, L., Toivonen, H., King, R.D.: Finding frequent substructures in chemical compounds. In: Agrawal, R., Stolorz, P., Piatetsky-Shapiro, G. (eds.) 4th International Conference on Knowledge Discovery and Data Mining, pp. 30–36. AAAI Press, Menlo Park (1998)
12. Vladutz, G.: Do we still need a classification of reactions? In: Willet, P. (ed.) Modern Approaches to Chemical Reaction Searching, pp. 202–220. Gower Publishing (1986)

Refining Pairwise Similarity Matrix for Cluster Ensemble Problem with Cluster Relations

Natthakan Iam-on, Tossapon Boongoen, and Simon Garrett

Department of Computer Science, Aberystwyth University, UK
{nii07,tsb,smg}@aber.ac.uk
http://www.aber.ac.uk/compsci

Abstract. Cluster ensemble methods have recently emerged as powerful techniques, aggregating several input data clusterings to generate a single output clustering, with improved robustness and stability. This paper presents two new similarity matrices, which are empirically evaluated and compared against the standard co-association matrix on six datasets (both artificial and real data) using four different combination methods and six clustering validity criteria. In all cases, the results suggest the new link-based similarity matrices are able to extract efficiently the information embedded in the input clusterings, and regularly suggest higher clustering quality in comparison to their competitor.

Keywords: cluster ensembles, pairwise similarity matrix, cluster relation, link analysis.

1 Introduction

1.1 Motivation

Data clustering sets out to discover data groupings, or clusters, such that data in the same cluster are more similar to each other than to those in different clusters. Application domains include bioinformatics, machine learning, data mining, information retrieval and pattern recognition, and the ability to produce high quality clusters is highly valued. Since there are a large number of clustering algorithms [1], and since the No Free Lunch theorem [2] suggests there is no single, supreme algorithm for discovering all cluster shapes and structures, it is extremely difficult for users to decide which algorithm would be the most effective for a given set of data. *Cluster ensemble* methods set out to mitigate this problem by allowing the user to perform various clusterings, and then uses them all to produce a higher quality output than any individual clustering. This paper suggests two new cluster ensemble methods for improving cluster value, and validates the improvement against several known datasets.

1.2 Overview

This paper build on the ideas of cluster ensmbles, including the feature-based approach that transforms the problem of cluster ensembles to clustering categorical data [3], graph-based algorithms that employ a graph partitioning methodology [4], and the pairwise approach that makes use of the co-association matrix to represent relationships between all pairs of data points [4], [5].

Methods for generating two new pairwise similarity matrices are presented, named *Connected-Triple based similarity (CTS) matrix* and *SimRank based similarity (SRS) matrix*. Both are informed by the basic conjecture of taking into consideration as much information, embedded in a cluster ensemble, as possible when finding similarity between data points. To discover similarity values, they consider both the associations among data points as well as those among clusters in the ensemble using link-based similarity measures.

The paper is organized as follows. Section 2 contains a formal definition of the cluster ensemble problem, and a review of related works on both the pairwise cluster ensemble and link analysis. Section 3 describes two powerful new similarity matrices. Section 4 provides the results of the experimental evaluation of the new matrix methods under a variety of conditions. Section 5 contains analysis and suggestions for further work.

2 Background

2.1 Foundational Concepts

Problem Formulation. Let $X = \{x_1, x_2, \ldots, x_N\}$ be a set of N data points and let $\Pi = \{\pi_1, \pi_2, \ldots, \pi_M\}$ be a set of M base clustering results, which will be referred to as a *cluster ensemble*. Each base clustering result (called an *ensemble member*) returns a set of clusters $\pi_i = \{C_1^i, C_2^i, \ldots, C_{k_i}^i\}$, such that $\bigcup_{j=1}^{k_i} C_j^i = X$, where k_i is the number of clusters in the i-th clustering. For each $x \in X$, $C(x)$ denotes the cluster label to which the data point x belongs. In the i-th clustering, $C(x) = j$ if $x \in C_j^i$. The problem is to find a new partition π^* of a data set X that summarizes the information from the cluster ensemble Π.

Cluster Ensemble Framework. The general process for creating a cluster ensemble is shown in Figure 1. Multiple input clusterings, known as *ensemble members*, are aggregated to form a final partition. There are two stages: (i) generating the cluster ensemble, and (ii) producing the final partition (normally referred to as *consensus function*).

The cluster ensemble is typically built by exploiting both different cluster models and different data partitions. Methods may use different cluster algorithms or a single algorithm with several sets of parameter initialization, such as cluster centers and number of clusters used in k-means method [3], [5], [6]. A cluster ensemble can also be achieved by applying manifold subsets of initial data to base clusterings [7]. In addition to using one of these methods, any combination of them can be applied as well [4].

The *consensus functions* can be categorized into: (i) feature based, (ii) graph based and (iii) pairwise approaches. The first technique transforms the problem of cluster ensembles to clustering categorical data [3], [6]. The second methodology represents an ensemble as a graph, which is divided into a definite number of

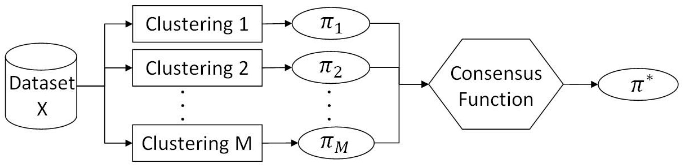


Fig. 1. The basic process of cluster ensembles. It first applies multiple base clusterings to a dataset X to obtain diverse clustering decisions ($\pi_1 \ldots \pi_M$). Then, these solutions are combined to establish the final clustering result (π^*) using a consensus function.

approximately equal-sized partitions [4], [8]. The last approach creates a matrix, containing the pairwise similarity among data points, to which similarity-based clustering algorithms can be applied [4], [5], [7].

Pairwise Cluster Ensemble. [5] Given a cluster ensemble Π, a $N \times N$ similarity matrix is constructed for each ensemble member, denoted as $S_m, m = 1 \ldots M$. Matrix entries represent the relationship between data points, x_i and x_j (Equation 1), and the matrices are effectively merged to form the *co-association (CO)* matrix (Equation 2):

$$S_m(x_i, x_j) = \begin{cases} 1 & if C(x_i) = C(x_j) \\ 0 & otherwise \end{cases} \tag{1}$$

$$CO(x_i, x_j) = \frac{1}{M} \sum_{m=1}^{M} S_m(x_i, x_j) \tag{2}$$

Having obtained the CO matrix, Fred and Jain [5] used the agglomerative clustering to derive the final partitions. In contrast, Strehl and Ghosh [4] proposed Cluster-based Similarity Partitioning Algorithm (CSPA) that generates a similarity graph whose vertices represent data points and edges' weights represent similarity scores obtained from the CO matrix. Afterwards, a graph partitioning algorithm called METIS [9] is used to divide this graph into k clusters of approximately equal size.

2.2 Link Analysis

When objects are connected according to their relations, it is possible to estimate the similarity of any object pair by using the underlying link information. Various link-based similarity measures have proven effective for the classification of web documents [10]. Two of these approaches, namely the "Connected-Triple algorithm" and the "SimRank algorithm", have proven to be particularly useful in the development of new similarity matrices that have improved cluster accuracy. Their fundamental concepts and applications to similarity matrices are thoroughly explained in Section 3.

3 Methodology: New Similarity Matrices for Consensus Functions

Despite the advantage of its simplicity, the CO matrix fails drastically to handle *unknown* relations between data points, whose similarity is zero. Investigations revealed zero-similarity values of 75%, +/- 5%, for the real-world datasets used here. The CO matrix can expose only a small proportion of pairwise similarity between data points, which may be better discovered by bringing in additional information regarding relations between clusters in an ensemble. As a result, two novel link-based methods are proposed to estimate similarity values: the Connected-Triple based similarity (CTS) matrix and the SimRank based similarity (SRS) matrix .

3.1 Connected-Triple Based Similarity (CTS) Matrix

The Connected-Triple approach is used in finding duplicates of author names in the bibliographic database, DBLP [11]. It works on the basis that if two nodes share a link to a third node then this is indicative of similarity between those two nodes. Its application to the cluster ensemble problem is illustrated in Figure 2. The circle vertices denote data points and the square nodes represent clusters in each clustering. There exists an edge between a data point x_i and a cluster C_j if x_i belongs to C_j. In particular, data points x_1 and x_2 are considered to be similar in clustering 2 and 3, in which they are assigned to the same clusters (clusters C and D, respectively). In contrary, their similarity is denoted as zero using information in the clustering 1 alone. Intuitively, despite being assigned to different clusters, their similarity may be revealed if these clusters are seemingly similar. Using the Connected-Triple approach, cluster A and B are justified similar due to the fact that they possess 2 Connected-Triples in which cluster C and D are centers of the triples.

Originally, the amount of triples associated with any two objects is summed up as a whole number. This simple counting might be sufficient for data points or other indivisible objects. However, to evaluate the similarity between clusters, it is crucial to take into account the characteristics like shared data members among clusters. Inspired by this idea, the new Weighted Connected-Triple algorithm for the problem of cluster ensembles is introduced as follows.

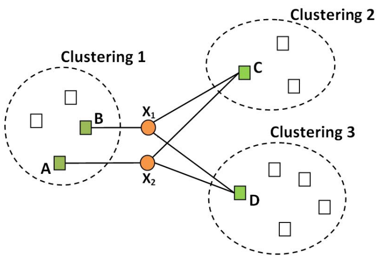


Fig. 2. A graphical representation of a cluster ensemble

Weighted Connected-Triple. Given a cluster ensemble Π, a graph $G = (V, W)$ can be constructed where V is the set of vertices each representing a cluster in Π and W is a set of weighted edges between clusters. Formally, the weight assigned to the edge connecting clusters i and j is estimated in accordance with the proportion of their overlapping members.

$$w_{ij} = \frac{|X_i \cap X_j|}{|X_i \cup X_j|} \tag{3}$$

where X_A denotes the set of data points belonging to cluster A. Instead of counting the number of triples as a whole number, the Weighted Connected-Triple regards each triple as the minimum weight of the two involving edges.

$$C_{ij}^k = \min(w_{ik}, w_{jk}) \tag{4}$$

where C_{ij}^k is the count of the triple between clusters i and j whose common neighbor is cluster k. The count of all triples $(1 \ldots q)$ between cluster i and cluster j can be calculated as follows:

$$C_{ij} = \sum_{k=1}^{q} C_{ij}^k \tag{5}$$

The similarity between clusters i and j can be estimated as follows, where C_{max} is the maximum C_{ij} value of any two clusters i and j.

$$S_{WT}(i, j) = \frac{C_{ij}}{C_{max}} \tag{6}$$

Connected-Triple Based Similarity (CTS) Matrix. This matrix adopts the cluster-oriented approach previously described to enhance the quality of the pairwise similarity matrix. Specifically, for the m-th ensemble member, the similarity of data points x_i and x_j is estimated using Equation 7, where DC is a constant decay factor (i.e. confidence level of accepting two non-identical objects as being similar) whose value range is in [0,1].

$$S_m(x_i, x_j) = \begin{cases} 1 & if C(x_i) = C(x_j) \\ S_{WT}(C(x_i), C(x_j)) \times DC & otherwise \end{cases} \tag{7}$$

Following that, each entry in the CTS matrix can be computed as,

$$CTS(x_i, x_j) = \frac{1}{M} \sum_{m=1}^{M} S_m(x_i, x_j) \tag{8}$$

3.2 SimRank Based Similarity (SRS) Matrix

SimRank reflects the underlying assumption that *neighbors are similar if their neighbors are similar as well.* Essentially, the similarity of any two objects, g_1 and g_2, can be calculated as follows:

$$s(g_1, g_2) = \frac{DC}{|P_{g_1}||P_{g_2}|} \sum_{i=1}^{|P_{g_1}|} \sum_{j=1}^{|P_{g_2}|} s(P_{g_1}^i, P_{g_2}^j) \tag{9}$$

where DC is a decay factor and $DC \in [0, 1]$, P_{g_1} and P_{g_2} are the sets of neighbors of objects g_1 and g_2. Individual neighbors of these objects are specified as $P_{g_1}^i$ and $P_{g_2}^j$, for $1 \leq i \leq |P_{g_1}|$ and $1 \leq j \leq |P_{g_2}|$. Note that $s(g_1, g_2) = 0$ when $P_{g_1} = \emptyset$ or $P_{g_2} = \emptyset$. It is suggested by Jeh and Widom [12] that the optimal similarity measures could be obtained through iterative refinement of similarity values to a fixed-point (i.e. after k iterations).

$$\lim_{k \to \infty} R_k(g_1, g_2) = s(g_1, g_2) \tag{10}$$

$$R_{k+1}(g_1, g_2) = \frac{DC}{|P_{g_1}||P_{g_2}|} \sum_{i=1}^{|P_{g_1}|} \sum_{j=1}^{|P_{g_2}|} R_k(P_{g_1}^i, P_{g_2}^j) \tag{11}$$

At the outset, this iterative process starts off using the lower of: $R_0(g_1, g_2) = 1$ if $g_1 = g_2$, and 0 otherwise.

Applying SimRank to Cluster Ensemble Problem. Besides considering a cluster ensemble as a network of clusters only (the CTS algorithm), a bipartite representation can be utilized to reveal more hidden relations. Figure 3(a) and 3(b) show the cluster results of two base clusterings, and the corresponding bipartite graph is presented in Figure 3(c).

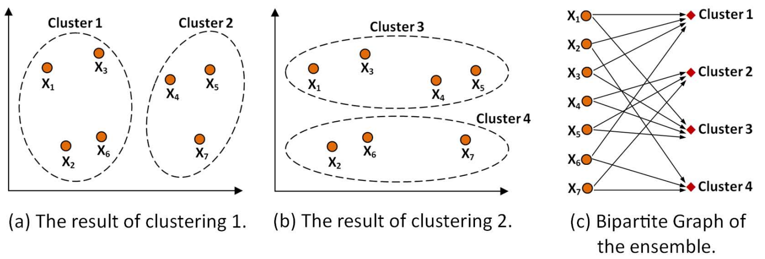

Cluster 1
Cluster 2
X1
X3
X4
X5
X2
X6
X7
(a) The result of clustering 1.
Cluster 3
Cluster 4
(b) The result of clustering 2.
Cluster 1
Cluster 2
Cluster 3
Cluster 4
(c) Bipartite Graph of the ensemble.

Fig. 3. Representing a cluster ensemble as a bipartite graph

Given a cluster ensemble Π, a graph $G = (V, E)$ can be constructed, where V is a set of vertices representing both data points and clusters in the ensemble and E denotes a set of edges between data points and their clusters. Let $SRS(a, b)$ is the entry in the SRS matrix, which represents the similarity between any pair of data points or the similarity between any two clusters in the ensemble. For $a = b, SRS(a, b) = 1$. Otherwise,

$$SRS(a, b) = \frac{DC}{|N_a||N_b|} \sum_{a' \in N_a} \sum_{b' \in N_b} SRS(a', b') \tag{12}$$

where DC is constant decay factor within the interval [0, 1], N_x denotes the set of vertices connecting to x. Accordingly, the similarity between data points x_i and x_j is the average similarity between the clusters to which they belong, and the similarity between clusters is the average similarity between their members.

Iterative Computation of SimRank. The similarity scores between any pair of vertices can be computed through the iteration process. Let $SRS_r(a, b)$ be a similarity score between a and b at iteration r, the estimation of the similarity score at the next iteration $r + 1$ is shown below. Note that, at the outset, $SRS_0(a, b) = 1$ if $a = b$ and 0 otherwise.

$$SRS_{r+1}(a,b) = \frac{DC}{|N_a||N_b|} \sum_{a' \in N_a} \sum_{b' \in N_b} SRS_r(a', b') \qquad (13)$$

4 Experiments, Results and Analysis

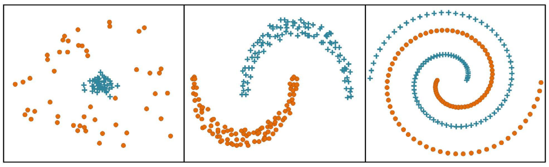
In this section, the experimental evaluation of two new pairwise similarity matrices is presented. Their performances are compared with that of the CO matrix over both synthetic and real-world datasets, using a variety of validity indices.

4.1 Datasets

The proposed similarity matrices are experimentally evaluated over six datasets, where true natural clusters are known but are not explicitly used by the cluster ensemble process (see Figure 4(a)). Three real-world datasets obtained from UCI benchmark repository[1] are used: Iris, Wine and Glass. In addition, three synthetic datasets are included in the experiments: Difficult Doughnut [13], 2-banana and 2-spiral [14] datasets, shown in Figure 4(b) to 4(d), respectively.

Datasets	Data points	Features	Classes
Difficult Doughnut	100	12	2
2-banana	200	2	2
2-spiral	190	2	2
Iris	150	4	3
Wine	178	13	3
Glass	214	9	6

(a) Dataset details.



(b) Difficult Doughnut. (c) 2-banana. (d) 2-spiral.

Fig. 4. Synthetic datasets

4.2 Evaluation Criteria

Since the external class labels are available for all experimented datasets, the results of final clustering are evaluated using four label-oriented validity indices: Classification Error, Normalized Mutual Information [4], Rand index [15] and Adjusted Rand index [15]. In particular, the Classification Error (CE) measures

[1] www.ics.uci.edu/∼mlearn/MLRepository.html

the number of misclassified data points within a clustering solution compared with known class labels. Formally, the CE score is estimated by [6] using the following equation; where N and K are the number of data points and clusters in the dataset, respectively, n_i is the number of data points in cluster i and m_i is the number of data points with the *majority* class label in cluster i.

$$CE = \frac{\sum_{i=1}^{K}(n_i - m_i)}{N} \tag{14}$$

In addition to label-oriented criteria, the *goodness* of a clustering solution can be measured using only quantities and features inherited from the dataset. Thus, two other label-irrelevant validity criteria, namely Compactness [16] and Dunn index [17], are also employed to measure the quality of clustering results.

4.3 Experimental Results

Unknown Relations (Zero-Similarity Values). Comparing to the CO matrix, Figure 5 presents much lower percentages of unknown relations achieved in the proposed link-based similarity matrices, the CTS and SRS matrices. This empirical evidence signifies that link-based similarity measures can help discovering implicit relationship amongst data points, which is not possible using the original co-occurrence statistical approach.

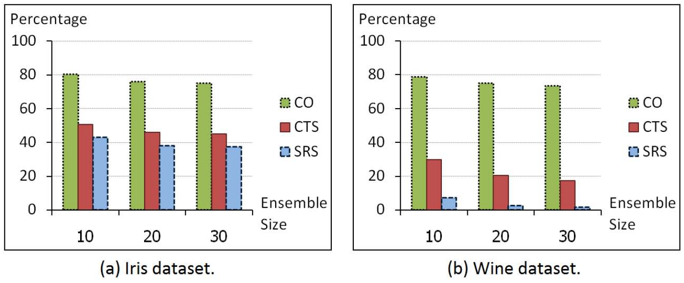


Fig. 5. Percentages of zero-similarity values in two proposed similarity matrices, comparing to those of the CO matrix. This set of statistics is the average figures of 50 runs achieved on Iris and Wine datasets, with three different ensemble sizes (10, 20 and 30).

Quality of Pairwise Similarity Matrices. The quality of CTS and SRS matrices are empirically compared against the conventional CO matrix using several settings of cluster ensembles exhibited below.

- The k-means clustering algorithm is specifically used to generate the base clusterings, with random initialization of cluster centers.
- Three schemes for selecting the number of clusters (k) in each base clustering are: fixed $k = \sqrt{N}$, random k in $[2, \sqrt{N}]$, and random k in $[2, N/2]$, where N is the number of data points.

– Three different ensemble sizes of 10, 20 and 30 base clusterings, respectively.
– The constant decay factor (DC) are set to be 0.5 and 0.8 for the Connected-Triple and SimRank algorithms, respectively.
– Consensus methods: three agglomerative approaches (single-linkage, complete-linkage and average-linkage) and a graph partitioning method (METIS). Note that, applying METIS to the CO matrix is the technique named *Cluster-based Similarity Partitioning Algorithm (CSPA)* [4]. For comparison purpose, as in [6] and [8], these consensus functions divide data points into K (the number of *true classes* for each dataset) partitions in accordance with the underlying similarity matrix (CO, CTS or SRS).
– The ultimate clustering results are evaluated using six validity indices emphasized in Section 4.3. The quality of each similarity matrix with each specific ensemble setting is generalized as the average of 50 runs.

Table 1 presents a specific subset of experimental results where each final clustering result produced by consensus methods is evaluated using only the Classification Error (CE) measure and the ensemble size is 30. Accordingly, the CTS and SRS approaches achieve the minimum CE score for most experimented settings of cluster ensembles. This effectively implies the better quality of two link-based similarity matrices comparing to the traditional co-association method. Especially, the CO matrix causes the worst performance with the complete-linkage algorithm across all datasets, and real-world datasets in particular.

Table 1. Classification errors (in percentage) of CTS, SRS and CO pairwise methods. Represented figures are the averages across 50 runs of cluster ensemble (size = 30), using four different combination techniques over six datasets. The lowest CE score of each specific ensemble setting is highlighted in **boldface**.

Dataset	k	Single-Linkage			Complete-Linkage			Average-Linkage			METIS		
		CTS	SRS	CO	CTS	SRS	CO	CTS	SRS	CO	CTS	SRS	CO
Difficult	10	36.18	**6.96**	37.10	28.78	**25.20**	39.96	11.98	**2.34**	6.40	**1.00**	**1.00**	1.32
Doughnut	[2,10]	7.60	**3.42**	9.50	24.00	**23.42**	32.70	25.12	**20.16**	27.58	1.00	1.00	**0.98**
	[2,50]	38.50	**36.28**	39.20	26.92	**26.26**	36.82	24.18	**19.92**	24.28	**1.00**	**1.00**	1.08
2-banana	15	**0.00**	**0.00**	**0.00**	33.82	**24.91**	41.94	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	[2,15]	**0.00**	**0.00**	**0.00**	12.41	**11.22**	29.45	**2.08**	2.35	4.93	**5.86**	7.07	13.38
	[2,100]	**0.00**	**0.00**	**0.00**	21.56	**18.56**	35.02	5.02	**4.72**	7.02	7.56	**7.50**	8.32
2-spiral	14	40.99	42.51	**38.38**	**44.07**	49.31	47.54	47.59	**45.87**	46.84	39.27	**36.51**	40.64
	[2,14]	46.41	**38.39**	46.06	**38.63**	39.92	42.80	37.88	**37.53**	38.15	**35.00**	35.53	35.94
	[2,95]	**8.89**	18.98	11.34	41.63	**41.16**	45.42	38.71	**38.23**	39.68	37.05	**36.08**	38.18
Iris	13	20.91	**16.91**	23.40	16.07	**14.92**	47.43	16.11	**12.75**	13.19	3.96	**3.92**	4.36
	[2,13]	18.69	**9.57**	18.20	**11.53**	12.29	17.11	**14.15**	14.27	14.55	**4.41**	4.52	4.56
	[2,75]	**31.63**	32.00	31.75	**12.45**	13.35	31.57	**13.01**	13.71	13.79	**4.40**	4.53	5.11
Wine	14	41.87	**39.94**	42.47	8.53	**7.66**	49.89	16.53	**8.94**	11.00	**7.70**	7.94	7.84
	[2,14]	46.89	**32.27**	48.58	**7.13**	7.96	20.25	4.36	5.01	**4.17**	7.31	7.29	7.18
	[2,89]	56.49	**55.63**	57.07	**9.94**	10.20	37.91	**6.94**	7.66	7.57	6.81	**6.71**	7.04
Glass	15	48.74	**40.67**	48.82	46.06	**45.51**	50.08	48.90	**47.60**	48.75	40.01	40.17	**38.82**
	[2,15]	53.81	**49.45**	53.88	**47.65**	47.83	48.04	47.69	47.32	**47.00**	40.42	**40.03**	41.19
	[2,107]	52.10	**49.76**	53.36	47.90	**47.48**	55.64	46.88	**45.34**	47.48	40.30	**37.79**	40.10

Table 2. The number of times that each matrix provides the best performance, measured by six validity indices, across six datasets (Difficult Doughnut, 2-banana, 2-spiral, Iris, Wine and Glass), four combination methods (SL, CL, AL and METIS), three different ensemble distributions (fixed $k = \sqrt{N}$, random k in $[2, \sqrt{N}]$ and random k in $[2, N/2]$) and three ensemble sizes (10, 20 and 30).

Validity Indices	Single-Linkage			Complete-Linkage			Average-Linkage			METIS		
	CTS	SRS	CO	CTS	SRS	CO	CTS	SRS	CO	CTS	SRS	CO
Classification Error	10	45	12	18	36	0	13	35	12	27	27	9
Normalized Mutual Information	19	35	13	20	32	2	21	29	10	27	28	8
Rand Index	13	41	13	16	38	0	14	37	9	28	27	8
Adjusted Rand Index	19	35	13	21	33	0	20	31	9	28	27	8
Compactness	10	43	14	15	36	3	14	36	10	9	23	30
Dunn Index	26	20	21	14	28	12	20	16	24	20	20	22
Total	97	219	86	104	203	17	102	184	74	139	152	85

In order to compare the quality of CO, CTS and SRS similarity matrices across all settings of cluster ensemble aforementioned emphasized, the winning statistics is exploited. Particularly, Table 2 summarizes the number of times that each pairwise method achieves the best performance across all experiment settings. Based on four label-oriented validity measures, both CTS and SRS approaches perform much better than the other counterpart. However, without label information, the goodness of clusters generated from three matrices are rather competitive.

Analysis of the SRS Matrix. Due to the recursive computation of the SRS approach, it is important to investigate the implication of iterations on the clustering quality. With the ensemble size of 10, Figure 6 depicts the classification error (CE) scores (averages of 50 runs) of the CO and SRS methods over Iris dataset (similar trends obtained with other datasets). It is obvious with all consensus functions that the SRS method usually achieves lower CE scores than the CO approach in its second iteration. Additionally, for all datasets, the convergence of the SRS method is typically achieved within 3 or 4 iterations, which is analogous to the typical convergence of the SimRank algorithm [12].

Complexity Analysis. Besides quality assessment, this section presents computational requirements of link-based similarity approaches. Primarily, the time complexity of creating the CO matrix is $O(N^2M)$, while that of the CTS matrix is $O(N^2MT)$, where T is the average of $|N_{C_a}||N_{C_b}|$, C_a and C_b are the clusters to which data points a and b belong, $|N_{C_a}|$ and $|N_{C_b}|$ are the number of clusters that share members with C_a and C_b, respectively. In addition, the same requirement of the SRS matrix is $O(nr(N^2 + C^2))$, where n is the average of $|N_a||N_b|$ over all pairs (a,b), r denotes the iterations of the SimRank algorithm and C is the total number of clusters in the ensemble.

In essence, two link-based matrices are more computationally expensive than the original CO matrix and they may be impractical for very large datasets.

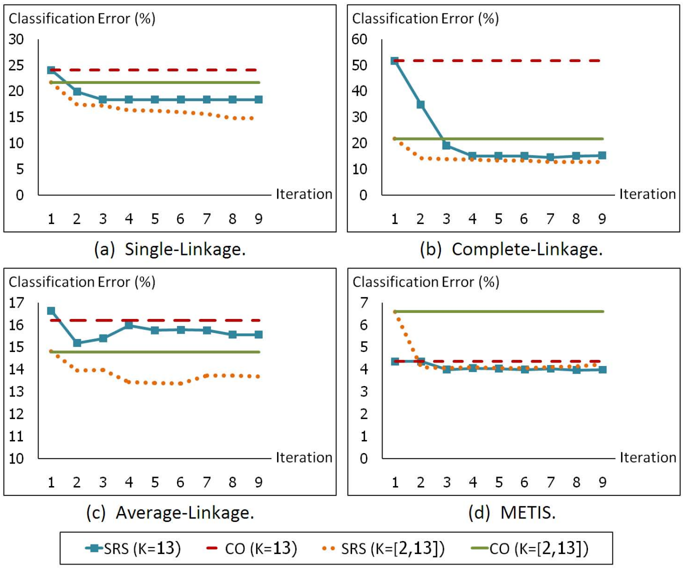


Fig. 6. Classification errors of the SRS pairwise approach at various iterations, compared to the CO matrix over the Iris dataset, two ensemble distributions (fixed $k = \sqrt{N}$ and random k in $[2, \sqrt{N}]$) and four consensus functions (SL, CL, AL and METIS).

However, as empirically demonstrated, they greatly improve robustness and quality of clustering results, by being able to recover additional hidden relations among data points that are completely neglected in the CO approach.

5 Conclusion

This paper presents two novel link-based similarity matrices for the problem of cluster ensembles, Connected-Triple based similarity matrix and SimRank based similarity matrix. In these pairwise matrices, similarity among data points is evaluated using structural-context embedded within cluster ensembles. The empirical studies, with several ensemble settings and validity measures over real-world and synthetic datasets, suggests that the proposed methods achieve superior clustering results comparing to the traditional co-association approach. This achievement is greatly due to the fact that link-based algorithms are able to reveal more hidden relationship among data points, with which the original method fail utterly to cope. Despite such promising practice, the prominent future work includes the reduction of time complexity inherited from link-oriented analysis. To this extend, subspace and sampling techniques may make link-based methods much more effective.

References

1. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: A review. ACM Comput. Surv. 31(3), 264–323 (1999)
2. Wolpert, D.H., Macready, W.G.: No free lunch theorems for search. Technical Report SFI-TR-95-02-010, Santa Fe Institute (1995)
3. Topchy, A.P., Jain, A.K., Punch, W.F.: A mixture model for clustering ensembles. In: Berry, M.W., Dayal, U., Kamath, C., Skillicorn, D.B. (eds.) SDM. SIAM, Philadelphia (2004)
4. Strehl, A., Ghosh, J.: Cluster ensembles — a knowledge reuse framework for combining multiple partitions. Journal of Machine Learning Research 3, 583–617 (2002)
5. Fred, A.L.N., Jain, A.K.: Combining multiple clusterings using evidence accumulation. IEEE Trans. Pattern Anal. Mach. Intell. 27(6), 835–850 (2005)
6. Gionis, A., Mannila, H., Tsaparas, P.: Clustering aggregation. In: ICDE, pp. 341–352. IEEE Computer Society, Los Alamitos (2005)
7. Fern, X.Z., Brodley, C.E.: Random projection for high dimensional data clustering: A cluster ensemble approach. In: Fawcett, T., Mishra, N. (eds.) ICML, pp. 186–193. AAAI Press, Menlo Park (2003)
8. Fern, X.Z., Brodley, C.E.: Solving cluster ensemble problems by bipartite graph partitioning. In: Brodley, C.E. (ed.) ICML. ACM International Conference Proceeding Series, vol. 69. ACM, New York (2004)
9. Karypis, G., Kumar, V.: Multilevel k-way partitioning scheme for irregular graphs. J. Parallel Distrib. Comput. 48(1), 96–129 (1998)
10. Calado, P., Cristo, M., Gonçalves, M.A., de Moura, E.S., Ribeiro-Neto, B.A., Ziviani, N.: Link-based similarity measures for the classification of web documents. JASIST 57(2), 208–221 (2006)
11. Klink, S., Reuther, P., Weber, A., Walter, B., Ley, M.: Analysing social networks within bibliographical data. In: Bressan, S., Küng, J., Wagner, R. (eds.) DEXA 2006. LNCS, vol. 4080, pp. 234–243. Springer, Heidelberg (2006)
12. Jeh, G., Widom, J.: Simrank: a measure of structural-context similarity. In: KDD, pp. 538–543. ACM, New York (2002)
13. Kuncheva, L.I., Vetrov, D.: Evaluation of stability of k-means cluster ensembles with respect to random initialization. IEEE Trans. Pattern Anal. Mach. Intell. 28(11), 1798–1808 (2006)
14. de Castro, L.N.: Immune Engineering: Development of Computational Tools Inspired by the Artificial Immune Systems. Ph.d. thesis, DCA - FEEC/UNICAMP, Campinas/SP, Brazil (2001)
15. Campello, R.J.G.B.: A fuzzy extension of the rand index and other related indexes for clustering and classification assessment. Pattern Recognition Letters 28(7), 833–841 (2007)
16. Nguyen, N., Caruana, R.: Consensus clusterings. In: ICDM, pp. 607–612. IEEE Computer Society, Los Alamitos (2007)
17. Dunn, J.C.: Well separated clusters and optimal fuzzy partitions. Journal of Cybernetica 4, 95–104 (1974)

Input Noise Robustness and Sensitivity Analysis to Improve Large Datasets Clustering by Using the GRID

Alberto Faro, Daniela Giordano, and Francesco Maiorana

University of Catania, Dipartimento di Ingegneria Informatica e Telecomunicazioni,
Via A. Doria, 6
95125 Catania Italy
{afaro,dgiordan,fmaioran}@diit.unict.it

Abstract. In this paper we investigate the performance of a refined version of the Kohonen self organizing feature maps algorithm in terms of classification correctness when we inject in a sparse input matrix different kinds of noise and compared these classification results with the one without noise. The analysis not only gives indications on the classification errors due to noisy data, but also let a methodology to emerge in order to identify the portion of the input matrix that must be controlled with great care for avoiding classification errors. The methodology also suggests a suitable data partitioning approach for a GRID implementation of the described algorithm. The methodological indications were successfully verified by a case study belonging to the bioinformatics field.

Keywords: Clustering algorithm, Kohonen SOFM, noise robustness, noise sensitivity, Grid, Globus toolkit.

1 Introduction

As is known, cluster analysis is an important field of study that according to [1] is the "task of organizing a set of objects into meaningful groups. The group can be disjoint, overlapping or organized in some hierarchical fashion". Clustering analysis has been used according with [2] in numerous fields ranging from marketing, pattern recognition earthquake studies, spatial data analysis, image processing, economic science, document classification and so on.

According to [3] the optimal clustering algorithms should have many properties, in this study we will focus our attention on the robustness and sensitivity to input noise of an unsupervised clustering algorithm derived from the Kohonen Self Organized Feature Map (SOFM) algorithm.

The paper is organized as follows: section 2 briefly recalls the Kohonen self organizing feature maps algorithms and the modifications designed by the authors to this algorithms, section 3 revises the literature about input noise robustness of the clustering algorithms, section 4 describes our case study dealing with a noise robustness and sensitivity methodology of particular interest to improve the execution of clustering algorithms of sparse dataset using the GRID infrastructure. The validity of the methodology is tested to classify a large bibliographical dataset in search of new gene-disease associations. Section 5 summarizes the lessons that can be derived from the case study to apply our methodology for the parallel clustering of general datasets.

2 From SOFM to Parallel Clustering Algorithms

The Kohonen Self Organizing Feature Maps (SOFM) [4], [5], [6] are often used to cluster datasets in an unsupervised manner. In its original version SOFM is composed by two layers: an input layer with as many nodes as the number of dimensions of the dataset we want to cluster, and an output layer with as many nodes as the number of classes we are looking for.

In this paper we deal with the online SOFM since the batch version has some disadvantages [7], as, for example, the fact that it is often a rough approximation of the online algorithm. In particular, in the paper we will take into account our modified version of SOFM, described in [8] and [9], characterized by a different topology of the network in which the output neurons are arranged along a mono-dimensional layer. In this configuration the classes are given by the output neurons. This means that if, at the final cycle, the neuron mostly activated by the i^{th} object is the j^{th} neuron, then the input object belongs to class j.

Our algorithm works on both the feature space and the similarity space. If we use the similarity space the software tool allows us to perform a final step in which, for each class, we find both the most relevant features common to the majority of class elements, called *positive features*, and the features few present in the majority of class elements, called *negative features*. The authors have also suggested in [8] and [9] an automatic strategy to find the optimal number of classes.

With the advent of the GRID, the interest towards the parallel version of SOFM or its variants is increased more and more in order to classify large datasets and discovery new associations in emerging fields such as biomedicine and nutritional genomics characterized by very large amount of both experimental and bibliographical data. The interested reader may find a concise survey of parallel unsupervised clustering algorithms in [16] and [17], where two effective parallel algorithms are also proposed: one dealing with the parallelization of the K-Means algorithm and the other of our refined version of SOFM.

3 Brief Review of Input Noise Sensitivity of the SOFM

Several studies can be found in literature concerning the evaluation of neural network robustness to input noise. In [10] the results of the noise sensitivity for various kinds of classifiers are given by taking into account two types of noise. To this aim the initial set of the n-dimensional vectors v_i representing the i^{th} input item has been modified by adding two entries $v_i(n+1)$ and $v_i(n+2)$ for each vector containing respectively a noise uniformly distributed in the interval [-1, 1] and [0, 0.1], whereas a third entry given by $[v_i(n+1)+ v_i(n+2)]/2$ for redundant information simulation has been also added.

In [11], a robustness analysis of a radial base function (RBF) and a multi-layered feed forward neural network used to approximate a series of simulated function has been done. From the original data set a noisy data set was generated by adding a small amount of noise sampled from a normal distribution N (μ, $\sigma/100$) where μ and σ are taken from the original dataset, yielding an error ratio of approximately 1%. This analysis was done in practice by taking into account a dataset derived from the chemistry. The robustness of the two networks, configured almost identically was compared with a slightly better result for the RBF neural net. In [11] it is pointed out that the robustness analysis depends on the dataset and on the network configuration used and that consequently it is not possible to draw general conclusions about intrinsic robustness properties. Thus a robustness analysis can give an indication on the property of a neural classifier with respect to a specific type of dataset.

In [12], the performances of the SOFM in features extraction by using time series generated from synthetic, linear progressive data are given. The sensitivity to map size, map lattice structure, initialization, neighbourhood function and noise has been evaluated. In particular the question about the ability of SOFM to extract known patterns in the presence of white noise with a signal to noise ratio equal to one has been investigated.

In [13], several clustering algorithms have been compared with respect to different parameters such as homogeneity and separation of clusters, silhouette width and robustness. For evaluating the robustness, the data were perturbed by adding a random vector. Each element of the random vector was generated from a Gaussian distribution ($\mu = 1$, $\sigma = 0.01$). For each cluster the index proposed in [14], named Weighted Average Discrepant Pair (WADP), has been calculated. This has been obtained by first computing for each cluster the ratio D/M, where M is the number of pairs of elements in the original cluster and D is the number of pairs of elements that don't remain together in the clustering of perturbed data. A weighted average of this index is then computed over all the clusters. This process was repeated many times to calculate the overall average as the WADP. The SOFM obtained the best value for the index with values ranging from 0.05 for 20 clusters to 0.18 for 60 clusters. (a value of zero of the index indicate perfect matching).

In [15], the SOFM has been compared against seven hierarchical clustering methods on 252 synthetic datasets with various level of imperfection on the input data such as: data dispersion, outliers, irrelevant variables and non uniform cluster densities pointing out the superior performance of SOFM. For example when one or two irrelevant variables are inserted in the dataset the performances of the SOFM decay respectively to 84% and 79.2%. The values reported are the average obtained with datasets containing from one to five variables.

In the following we study the problem of the noise robustness and sensitivity having in mind a clustering problem of large bibliographical data to be solved in a GRID environment in which the problem of the term identification and of how to subdivide the dataset among the computational units may be critical for achieving acceptable time performances and effective clustering precision.

4 Robustness and Sensitivity to Noise for a Sparse Input Matrix to be Classified by a Parallel Algorithm

In [16] we proposed a novel method for discovering and evaluating hopefully new relationships between genes and genetic diseases from Pubmed abstracts using the GRID. This method was based on the repetitive application of a clustering algorithm. In [16], the clustering algorithm at the core of the method is a parallel version of the K-means clustering, whereas in [17] we have proposed to use a parallel version of our modified version of SOFM (i.e. the one presented in [8] and [9]) to attain higher performances in processing time and to achieve a better precision in clustering.

We worked with real word data obtained from indexing 3528 Pubmed abstracts. The rows of the matrices M_k given as input to the mentioned clustering algorithms are related to the abstracts, whereas their columns give the frequency in the abstracts of the terms related to a certain gene or disease, e.g., $M_D(i, j)$ gives the frequency of the genetic disease labeled by j in the i^{th} abstract. Four matrices were used for discovering the gene-disease associations, i.e., $M_D(i, j)$ which gives the frequency of the genetic disease labeled by j in the i^{th} abstract assuming that some genes are also referred in the abstract; $M_G(i, j)$ which gives the frequency of the gene labeled by j in the i^{th} abstract assuming that some genetic diseases are also referred in the abstract; $D_D(i, j)$ which gives the frequency of the genetic disease labeled by j in the i^{th} abstract assuming that no genes are referred in the abstract; and $G_G(i, j)$ which gives the frequency of the gene labeled by j in the i^{th} abstract assuming that no genetic diseases are referred in the abstract.

Let us note that all these input matrices are sparse. From fig.1a we can observe also that more or less all the documents contain few terms, e.g., 2,204 documents out of 3,528 refer only one term in the abstract. The distribution of documents with respect to the term contained in the abstract is logarithmic. Fig.1b displays the number of columns for different numbers of elements greater than zero. From this figure we can observe that almost all the columns have a very few number of entries whose values differs from zero, e.g., for 230 columns the number of entries different from zero is less than 1% (35) of the total number of rows (3528). Fig.1b shows that the distribution of the terms with respect to the documents is an exponential distribution fast decaying to zero.

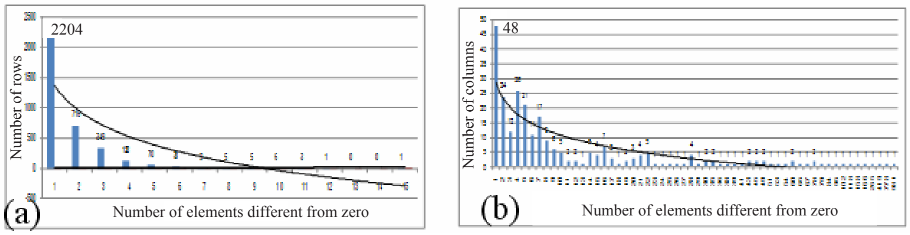


Fig. 1. Number of rows in a) and of columns in b) with respect to the number of elements different from zero

Meaningful associations and hopefully new ones have been extracted by using the method proposed in [16], however, two relevant problems which may affect precision and processing time of the resulting classification have been left open, i.e., how much effort should be dedicated to the term identification in order to limit the effect of the false positives and false negatives in the resulting classification, and how the input matrix should be optimally subdivided into sub-matrices to be elaborated by the GRID nodes so that the classification resulting in a parallel environment few differs from the one obtainable by a single supercomputing server.

For these reasons, in the following we present a study of the input noise robustness and sensitivity able to give useful indications on how to distribute the data among the computational elements of the GRID and on what columns and rows it is convenient to carry out an indexing with more sophisticated algorithms. Although the results of the study are valid for sparse matrices having the distributions shown in the previous figures 1a and 1b, the proposed methodology can be applied to solve the mentioned problems with respect to matrices having other type of distributions.

We have injected two kinds of noise in the four above mentioned feature matrices, to simulate both false positives and false negatives. In the former case some cells that are zero are set to a random value ranging from the minimum to the maximum value of the modified row or columns. The latter case behaves in opposite direction, i.e., some cells that are greater than zero are set to zero.

In each experiment we have fixed a threshold given as the percentage of the rows or columns to modify. This percentage is calculated against the elements greater than (or equal to) zero. Hence we have two main types of noise, i.e., noises which lead to false positives and the ones which lead to false negatives, and for each type of noise we have two ways to generate it (by row or by column). Injecting a noise in a row means that the document is characterized by a fuzzy abstract, whereas injecting a noise in a column means that the term identification algorithm adopted for indexing the term related to this column is few efficient.

We chose to modify: all rows (columns) or a selection of rows (columns) or a fixed percentage of rows (columns). We have also ordered the rows (columns) in decreasing importance order: the most important rows (columns) are the one with more values grater than zero. This will allow us to investigate the noise effects in the most important row or columns, thus clarifying if the presence of noise in important rows (columns) spawns more or less classification errors. A similar analysis was carried out on the less important rows.

As said before, in our analyses we injected noise as a percentage of elements greater than zero. We started by injecting false positives changing all the columns (rows) and modifying 1, 5, 10 and 15% of the rows (columns). The rows (columns) to be modified have been chosen randomly among the elements in the considered column (row) equal to zero. We repeated the process on the original dataset three times in order to mediate the random procedure. The results are reported in figure 2 for noise injected by modifying all the rows (a) and all the columns (b). For a more robust analysis we generated five synthetic datasets with the same distribution of figure 1. The averaged results are reported in figure 2, where we notice that the injection of noise row-wise produces a greater correct classification rate with respect to the noise column-wise since in the former case the number of modified elements are reduced. In fact, if we inject noise in all the rows by changing 15% of the elements we change 270 elements. This represents 0.04% of the elements of the input matrix. On the contrary, if we inject noise in all the columns and for each column we modify 15% of the elements greater than zero, we modify 1,459 elements, and this is about 0.1% of the elements of the input matrix.

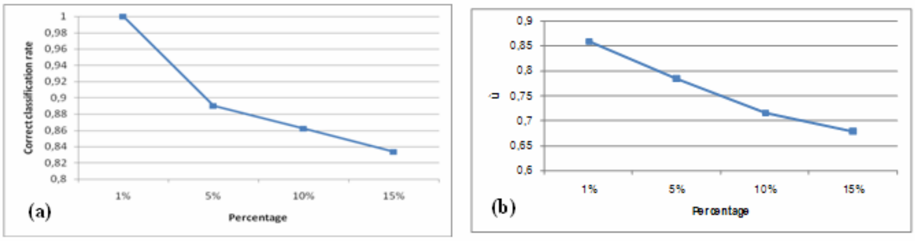


Fig. 2. Average correct classification rate for different percentage of injected noise on the original dataset: all the rows are modified; b) all the columns are modified

We also investigated the classification results when the false positive error is injected in the most important rows (columns). In this study we chose the first five hundreds rows with many ones and the first fifty columns with many ones. In this case for each column (rows) we chose randomly the percentage of rows (columns) to modify among the most important rows (columns). With a similar method we repeated the analysis for the less important rows (columns). Also in this case the process is repeated five times with synthetic datasets and the averaged results are reported in figure 3 where the correct classification rate for injected noise row-wise (i.e., by modifying all the rows) (a) are column-wise (b) (i.e., by modifying all the columns) are drawn. The graph compares the results when the changes are randomly distributed on all the columns (rows), or on the most important columns (rows), or in the less important columns (rows).

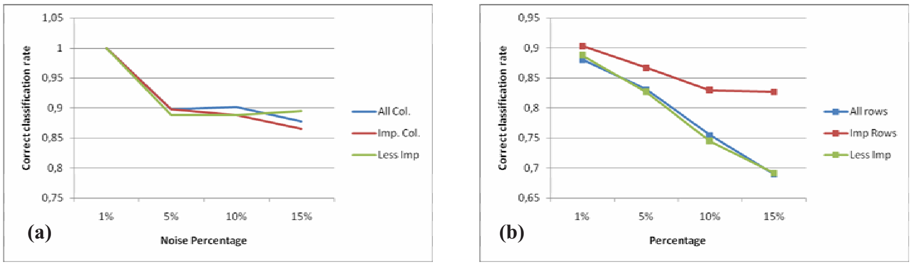


Fig. 3. Correct classification rate for different percentage of injected noise when we change all the rows a) or all the columns (b)

Let's consider first an injection of noise row-wise (figure 3a). The results are comparable between them for a low percentage of noise (since we change very few elements) but for a greater percentage of noise (i.e., 15%) we see that, if we take as reference the case in which, for each row, the modified elements are randomly distributed among all the columns, whose correct classification rate (CCR) is 0.8773, we obtain a better results if the noise is concentrated into the less important columns (CCR = 0.8950) and the worse result if the noise is concentrated into the most important columns (CCR = 0.8658). This is mainly due to the fact that the most frequent terms tend to characterize the document class, thus if many of such terms are, by mistake, present in non pertinent documents, this may produce with greater probability relevant modifications of the document classes. This imply that in indexing the document *we should put more attention to the most frequent terms by applying the more sophisticated term identification algorithms even at the cost of a more computational power.*

If we inject noise column-wise (figure 3b) we can observe that if, for each column, the noise is more concentrated into the most important rows we obtain the best classification rate. The difference is visible even for smaller percentage of noise. This is mainly due to the fact that the most important rows few modify their classification even if we change some frequency in the terms referred by their abstract. *This implies that we must put special care in the indexing method for the less important rows.*

From these results we may derive a methodology that can be applied to different datasets with the objective of quantifying the importance of different groups of rows or columns (i.e., the most important or the less important rows) with respect to the problem of the false positives or negatives. To this aim, we denote the set of less important rows (columns) as R^- (C^-) and the one of the most important rows (columns) as R^+ (C^+). The methodology consists of the following five steps:

1. analyze the performance of the classification algorithm of the original dataset to establish the class membership of each documents. This resulting classification will be used as *gold standard.*
2. inject noise by modifying all the rows (columns). For each row (columns) modify a percentage of the columns (rows). Selects the columns (rows) to modify among all the columns (rows) with a value equal to zero. Compare the classification results to the gold standard. This correct classification rate R_{ref} (C_{ref}) will be used as reference, i.e., the correctness rate of the *reference case.*
3. Inject the noise in the same percentage as step 2, but in this case the rows to modify are selected first among R^- and then among R^+. Compare the classification results to the gold standard thus obtaining the correct classification rate for either the less important and most important case, let say R_{less} *and* R_{most} such values. Compare these results with the results of reference case obtained in step 2, thus obtaining the following matrix R consisting of one column:

$$R = \begin{array}{|c|c|} \hline R^- & R_{less} / R_{ref} \\ \hline R^+ & R_{most} / R_{ref} \\ \hline \end{array}$$

4. Repeat the process in step 3 by substituting R^-, R^+, R_{less} *and* R_{most} by C^-, C^+, C_{less} *and* C_{most}, thus obtaining the following matrix C consisting of one row:

$$C = \begin{array}{|c|c|} \hline C^- & C^+ \\ \hline C_{less} / C_{ref} & C_{most} / C_{ref} \\ \hline \end{array}$$

5. Perform the following analysis for both the rows and the columns:
   - For the rows in R^- (for the columns in C^-) change the elements in C^- (R^-). For the rows in R^+ (for the columns in C^+) selects the elements to change randomly among the zero elements in all the columns (rows). Compare the classification results to the gold standard thus obtaining the correct classification rate $Q1_R$ ($Q1_C$).

- For the rows in R^+ (for the columns in C^+) change C^- (R^-). For the rows in R^- (for the columns in C^-) selects the columns (rows) randomly among the zero elements in all the columns (rows). Compare the classification results to the gold standard thus obtaining the correct classification rate $Q2_R$ ($Q2_C$).
- For the rows in R^- (for the columns in C^-) change C^+ (R^+). For the rows in R^+ (for the columns in C^+) selects the columns (rows) randomly among the zero elements in all the columns (rows). Compare the classification results to the gold standard thus obtaining the correct classification rate $Q3_R$ ($Q3_C$).
- For the rows in R^+ (for the columns in C^+) change C^+ (R^+). For the rows in R^- (for the columns in C^-) selects the columns (rows) randomly among the zero elements in all the columns (rows). Compare the classification results to the gold standard thus obtaining the correct classification rate $Q4_R$ ($Q4_C$).
- Compare the results against the reference case by using the following matrix:

$$Q_R = \begin{array}{|c|c|c|} \hline \blacktriangleright & C^- & C^+ \\ \hline R^- & Q1_R & Q2_R \\ \hline R^+ & Q3_R & Q4_R \\ \hline \end{array} \qquad Q_C = \begin{array}{|c|c|c|} \hline \blacktriangledown & C^- & C^+ \\ \hline R^- & Q1_C & Q2_C \\ \hline R^+ & Q3_C & Q4_C \\ \hline \end{array}$$

In particular if $R_{less} / R_{ref} < R_{most} / R_{ref}$ (i.e., the less important rows are the most sensitive ones to the noise), then the row R^- of the matrix Q_R has to be taken into account to identify if the most sensitive columns of the most sensitive rows are C^- or C^+ depending on if $Q1_R < Q2_R$ or not. If $R_{less} / R_{ref} > R_{most} / R_{ref}$ (i.e., the most important rows are most sensitive to the noise) we have to consider $Q3_R$ and $Q4_R$ in the row R^+ of the matrix Q_R to identify if the most sensitive columns of the most sensitive rows are the ones belonging to C^- or C^+. Similarly we have to proceed to identify the most sensitive columns of the matrix C and then the most sensitive rows of the most sensitive columns by comparing $Q1_C$ with $Q3_C$ or $Q2_C$ with $Q4_C$ of the matrix Q_C. For example if $C_{less} / C_{ref} < C_{most} / C_{ref}$, then columns C^- are the most sensitive and most sensitive rows of the most sensitive columns are the rows of R- or R+ depending on if $Q1_C < Q3_C$ or not.

To confirm such methodological indications experimentally we repeated the mentioned analysis for the five synthetic data sets characterized by the same distribution of figure 1. The results are reported in the following figures. Figures 4a and 4b refer to noise generated in column order (i.e., column-wise); Figures 4c and 4d refer to noise generated in row order (i.e., row-wise).

Figure 4a gives a strong indication that the less (most) important rows are most (less) sensitive to noise as reported earlier. According to the proposed methodology the most critical case is obtained when we associate the columns C^+ with rows of R^- (see fig.4b), whereas the best results are obtained when we associate the columns, especially C^+, with R^+. *Consequently, the resulting classification can be improved if we previously check the terms C^+ with the best indexing algorithms especially for the documents R^-.*

According to the proposed methodology figure 4c indicates that C^+ (C^-) are most (less) sensitive to noise as reported earlier. Moreover, the most critical case is obtained when we associate the rows R^+ with the columns C^+ (see fig.4d), and the best results are obtained when we associate the rows, especially R^-, with C^-. *Consequently, the resulting classification can be improved if we previously check the documents labeled by many terms with respect to the most frequent terms C^+ in the dataset.* Figure 5 reports the above results for different percentage of injected noise.

R⁻	$\underline{0.6919}$ 0.6900
R⁺	$\underline{0.8271}$ 0.6900

a)

▼	C⁻	C⁺
R⁻	$\underline{0.6870}$ 0.6900	$\underline{0.6840}$ 0.6900
R⁺	$\underline{0.7050}$ 0.6900	$\underline{0.7840}$ 0.6900

b)

C⁻	C⁺
$\underline{0.8950}$ 0.8773	$\underline{0.8658}$ 0.8773

c)

▶	C⁻	C⁺
R⁻	$\underline{0.8963}$ 0.8773	$\underline{0.8883}$ 0.8773
R⁺	$\underline{0.8893}$ 0.8773	$\underline{0.8748}$ 0.8773

d)

Fig. 4. Correct classification rate: a) noise concentrated in R^+ or R^- for column-wise noise, b) various combinations of importance to be used depending on matrix in (a), c) noise concentrated in C^+ and C^- for row-wise noise, d) various combinations of importance to be used depending on matrix in (c)

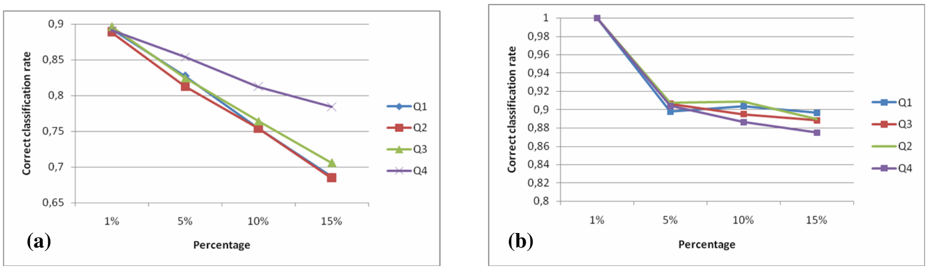


Fig. 5. Correct classification rate varying the noise percentage: a) column-wise noise, b) row-wise noise

The results generated by the analysis above have important implications in developing a suitable neural classification algorithm for the GRID since any parallel classification algorithm is greatly influenced on the data partitioning strategy adopted by a master node to divide the input matrix among the different computational elements. The simplest approach is to divide the rows randomly among the computational GRID. However, from the previous results we obtain a strong indication that R^- should be treated with great care since an error on the indexing of these documents will change, with greater probability, their class membership. Moreover each node should have an adequate sample of R^+ since they tend to greatly characterize the dataset. From these indications we find that a suitable strategy to divide the dataset among the GRID nodes is the one of splitting R^- and R^+ equally among the computational GRID so that each node has a suitable sample of the dataset.

To get an experimental evidence of this rule we performed two different classifications of the same data set on a GRID infrastructure based on Globus toolkit composed on three computational elements connected by a local area network: in the first case we distributed the rows randomly, in the second case we divided equally the less and most important rows among the nodes. We performed various classifications using different algorithms to merge the weights. The results are reported in table 1. The interested reader may find how the referred merging strategies work in [17].

From the above table we can argue that with the proposed strategy of data distribution we obtain the best result in term of correct classification rate. The improvements are visible in almost all the merging algorithms.

Table 1. Correct classification rate for various GRID implementations of the SOFM algorithm with a random row split or with a suitable distribution of R^- and R^+ among the nodes

	Mean of the weights	Mean with weight of different stages half weighted	Mean with weight of different stages double weighted	Weight rotation
Random split	0,8293	0,8276	0,8325	0,8195
R^- distributed	0,8124	0,934	0,9558	0,8934

We also investigated the correct classification rate for different number of R^+. Figure 6a reports the gain in correct classification rate between the reference case and the important case in which we change all columns and select the elements to change among the first 500, 750 and 1,000 most important rows. As we can see from the figure the improvements due to selection of R^+ decrease as we increase the set of R^+.

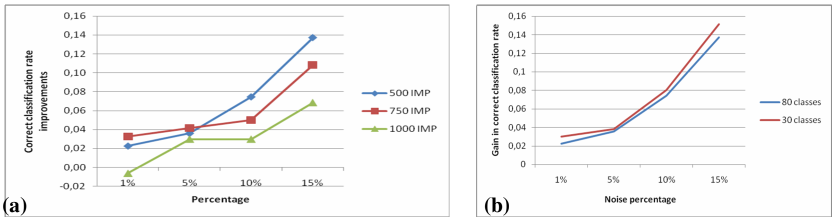


Fig. 6. Gain in correct classification rate between the reference case and the important case: a) for different numbers of important rows; b) classification with different numbers of classes

Another analysis was done on classifications performed on the same dataset but with thirty classes instead of eighty. We measured the correct classification improvement between the important case and the reference case. The results are reported on figure 6b where the gain is reported for different percentage of injected noise. As we can see from the figure with less are the classes and less is the chance for an element to change class since the boundaries between classes are sharper. If we increase the number of classes their boundary becomes fuzzy. Less classes we have more the classification is robust to noise.

We further investigated the performance classification for a synthetic dataset where we doubled the initial number of ones, leaving the same distribution. With the same percentage of injected noise we modify a double number of elements than the previous case. The analysis confirmed the previous results with a shift downward of the correct classification rate. For example in the reference case by changing 15 % of random columns we obtain a correct classification rate equal to 0,72 (row order); by changing 15% of random rows we obtain a classification rate equal to 0.52.

We also performed an analysis for noise injected as false negative. We used the original data set and repeated the analysis three times in order to average the random selection of the rows and columns. We performed the analysis only for the reference case. This is due to the fact that, if we try to modify all the rows (columns) and inject noise on the most important columns (rows), we are not sure to find the given percentage of elements greater than zero in this small set. We recall that the matrix is sparse so the elements greater than zero are a rarity. The results have a similar trend as in the reference case for false positive noise, with a little shift downward (1%) both for noise generated in both row and column orders. The shift is more evident for a percentage of noise equal to 15%. This could be due to the fact that we eliminate ones that are a rarity in the dataset and hence we produce more modifications.

5 Concluding Remarks

In this paper we have proposed a methodology to evaluate the robustness and sensitivity of a refined version of the SOFM algorithm to noise in the input data. The analysis was carried out on a real word dataset, coming from a bioinformatics application, and on synthetic datasets. From the analysis we can draw the following conclusions:

- the injection of false positive noise generated column-wise gives rise to greater errors. This is due to the fact that in this manner we modify a lot of documents. In dictionary based indexing if a term has many synonyms this could lead to false positive hence particular attention should be given to such terms.
- the false positive noise generated row-wise is more conservative since less abstract are modified. In automatic document indexing, making some mistake in processing some documents is not a big problem.
- the false negative effects tend to have a bit greater effect than false positive noise especially for greater percentage of noise. This means that when indexing documents we should take care also of alias.
- the noise concentrated into the most important rows has a less important effect than the one randomly distributed over all the rows. The difference is sharp. The situation reverses when we inject false positive noise on the most important columns: in fact errors on indexing the most important columns (i.e., the most frequent terms) have a mild greater negative impact on the classification performance.
- terms in C^+ should be treated with great care before starting the classification steps, in particular, we can split the indexing of the most common terms among the computational elements to perform the indexing of such terms with better algorithms even if they require more time.
- the best data partitioning scheme for a grid implementation of the clustering algorithm is the one of equally distributing the most and less important rows among the computational elements to obtain a strong gain in performance.

References

1. Zhao, Y., Karypis, G.: Data clustering in life science. Molecular Biotechnology 31(1), 55–80 (2005)
2. Xu, R., Wunsch, D.: Survey of Clustering Algorithms. IEEE Transactions on Neural Networks 16(3) (2005)
3. Tirozzi, B., Bianchi, D., Ferraro, E.: Introduction to computational neurobiology and clustering. World Scientific, Singapore (2007)
4. Kohonen, T.: Self Organizing Maps. Springer, Heidelberg (1995)

5. Kaski, S., Kangas, J., Kohonen, T.: Bibliography of self organizing map (SOM) papers: 1981 – 1997. Neural Computing Survey 1(3), 102–350 (1998)
6. Oja, M., Kaski, S., Kohonen, T.: Bibliography of self organizing map (SOM) papers: 1998 – 2001 Addendum. Neural Computing Survey 3(1), 1–156 (2003)
7. Cottrel, M., Fort, J.C., Letremy, P.: Advantages and drawbacks of the batch Kohonen Algorithm. In: 10th European Symposium On Artificial Neural Network, Bruges, Belgium, pp. 223–230 (2005)
8. Faro, A., Giordano, D., Maiorana, F.: Discovering complex regularities by adaptive Self Organizing classification. Enformatika I, 27–30 (2005)
9. Faro, A., Giordano, D., Maiorana, F.: Discovering complex regularities from tree to semi – lattice classifications. International Journal of Computational Intelligence 2(1), 34–39 (2005)
10. Beck, S., Ghosh, J.: Noise sensitivity of static neural network classifiers. In: Rogers, S.K. (ed.) Proceding of SPIE Conference on applications of Artificial Neural Networks III, vol. 1709, pp. 770–779 (1992)
11. Derks, E.P.P.A., Pastor, M.S.S., Buydens, L.M.C.: Robustness analysis of radial base function and multi-layered feed forward neural network models. Chemometrics and Intelligent Lab. System 28(1), 46–60 (1995)
12. Liu, Y., Weisberg, R.H., Mooers, C.N.K.: Performance evaluation of the self organizing map for feature extraction. Journal of geophysical Research 111 (2006)
13. Chen, G., Jaradat, S.A., Banerjee, N.: Evaluation and comparison of clustering algorithms in analyzing cell gene expression data. Statistica Sinica 12, 241–262 (2002)
14. Bittner, M., et al.: Molecular classification of cutaneous malignant melanoma by gene expression profiling. Nature 406, 536–540 (2000)
15. Mangiameli, P., Chen, S.K., West, D.: A comparison of SOM neural network and hierarchical clustering methods. Eur. Journal of Operational Research 93(2) (1996)
16. Faro, A., Giordano, D., Maiorana, F., Spampinato, C.: Discovering Genes–Diseases Associations from Specialized Literature using the GRID. IEEE Transactions on Information Technology in Biomedicine 18(5) (2008)
17. Faro, A., Giordano, D., Maiorana, F.: Optimizing the execution of parallel clustering algorithms over the GRID (2007), http://www.ing.campusone.it/FGM01.pdf

An Integrated Graph and Probability Based Clustering Framework for Sequential Data

Haytham Elghazel[1], Tetsuya Yoshida[2], and Mohand-Said Hacid[3]

[1] Université de Lyon, Lyon, F-69003, France ; université Lyon 1, EA4125, LIESP, Villeurbanne, F-69622, France
elghazel@bat710.univ-lyon1.fr
[2] Grad. School of Information Science and Technology, Hokkaido University
N-14 W-9, Sapporo 060-0814, Japan
yoshida@meme.hokudai.ac.jp
[3] Université de Lyon, Lyon, F-69003, France; université Lyon 1, CNRS UMR5205, LIRIS, Villeurbanne, F-69622, France
mshacid@liris.cnrs.fr

Abstract. This paper proposes a new integrated sequential data clustering framework based on an iterative process which alternates between the EM process and a modified b-coloring clustering algorithm. It exhibits two important features: Firstly, the proposed framework allows to give an assignment of clusters to the sequences where the b-coloring properties are maintained as long as the clustering process runs. Secondly, it gives each cluster a twofold representation by a generative model (Markov chains) as well as dominant members which ensure the global stability of the returned partition. The proposed framework is evaluated against benchmark datasets in UCI repository and its effectiveness is confirmed.

Keywords: Clustering, sequential data, graph *b-coloring*, EM algorithm.

1 Introduction

Data clustering, also called unsupervised classification, is a method of creating groups of objects, or clusters, in such a way that objects in one cluster are very similar and objects in different clusters are quite distinct. Clustering is one of the most frequently performed analyses on sequential data (or time series) [1]. Sequence clustering algorithms are widely used in many areas and play an important role in a broad range of applications. Specific applications are: clustering users based on their web navigation patterns [2], clustering patients based on Red Blood Cell Cytograms [3] and so on.

There are two major categories of strategy for the clustering of sequential data: *similarity based* approaches and *probabilistic model based* approaches. In the first approach, to cluster sequential data set, the similarity measures between sequences become important. In the latter approach, statistical models are built to describe the dynamics of each group of sequences based on their observations.

In [4], we have proposed a clustering method for sequential data. Based on the notion of b-coloring of a graph [5], it trains a Mixture of Markov chains models (EM process), where each sequence can be used to estimate the parameters of more than one cluster depending on their graph characteristics. Ultimately the process converges to a final clustering of the data and a generative model (the Markov chains) for each of the clusters, as well as dominant members which ensure the global stability of the partition. The framework worked well in discovering a typology of clinical pathways from medical data of the French health information system. However, this scheme suffers from two weaknesses that are: (1) It doesn't guarantee the b-coloring property (proper coloring and dominance) of the returned partition and (2) the reliance of the b-coloring approach to the selection colors manner for some vertices: especially in the initialization step where a randomly strategy is used.

This paper proposes a new integrated framework for sequential data *clustering, which integrates both graph-based and probability-based ones, in order* to complement each other. The proposed framework is based on an iterative process which alternates between the *EM process* and a *modified b-coloring*. The main goal here is to give an assignment of clusters to the sequences when the number of clusters is not specified in advance and so that:

- The b-coloring properties are maintained in the returned partition: *proper coloring* and *dominance.*
- Each cluster has a set of dominant sequences which reflects its properties and also guarantees that the cluster has a distinct separation from all other clusters of the partition.
- Each cluster is governed by a probabilistic generative model (*Markov chain*) for sequences peculiar to that group. The behavior of these models can be used for classification and predicting possible paths for new arrival sequences.

The proposed framework is evaluated against benchmark datasets in UCI repository and its effectiveness is confirmed.

2 Related Work

As described in Section 1, there are two major categories of strategy for the clustering of sequential data: *similarity based* approaches and *probabilistic model based* approaches. In the first approach, to cluster sequential data set, the similarity measures between sequences become important. In this case, the comparison between two sequences is viewed as a process to transform a given sequence into another, namely, *sequence alignment.* In [6], Gunopulos and Das present a tutorial for time series similarity measures. Two of the well-known similarity functions between a pair of sequences are: *Longest Common Subsequence* (LCS), and *Dynamic Time Warping* (DTW), both systematic and efficient methods be based on the sequence alignment.

When the proximity measure between sequences is computed, a variety of similarity-based techniques for grouping sequences can be found in literature [7] in order to discover a partition of data such that the sequences within the same cluster are similar to each other (*intracluster cohesion*) while sequences from different clusters are dissimilar (*intercluster separation*).The weak point of similarity-based approaches is that the returned clusters are not easily interpretable since most of these methods fail to give a clear idea about the *relationships* between sequences of clusters. Consequently, such methods don't facilitate the prediction and classification making process.

The other broad class in sequence clustering builds probabilistic models like Markov chains and HMM to describe the dynamics of each group of sequences. Among them are the Learning Mixture Markov models that constitute the most widespread popular tool for clustering time series. Given the number *k* of clusters, Cadez *et al.* [3] propose a unifying probabilistic framework for clustering individuals into *k* clusters, when the available data measurements are not multivariate vectors of fixed dimensionality. They provide a general Expectation-Maximization algorithm (EM) [8] for clustering such data set and demonstrate its usefulness on three applications. The key idea in [3] is the fact that each cluster in the partition is depicted by a Markov chain model and the EM approach is used to generate parameter estimates (the initial state probability vector and the transition matrix for each Markov cluster) so that cluster models are constructed in a straightforward and consistent manner.

Another probabilistic model-based approach for clustering sequences is proposed in [9] using a mixture of Hidden Markov Models (HMM). A real advantage of this method is that the behavior of each cluster of the partition is governed by a HMM. HMM is a powerful stochastic method for modeling sequential data, and has been successfully used in many tasks such as speech recognition and DNA sequence analysis. To estimate the models parameters, the HMM model-based approaches adopt a k-means formulation, which is extended with soft memberships via an Expectation Maximization (EM) procedure and applied to video data in [10]. While probabilistic clustering approaches exhibit some important features: the discovered models can be used to assign sequences to clusters (*online property)*, and the given clusters are easily interpretable. Nevertheless, the training process suffers from some weaknesses: The number of clusters *k* must be set beforehand and an initial partitioning of data is required in advance as an initialization step which generally affects the quality of the returned clustering.

In order to remedy the problems of both probabilistic model-based and similarity-based approaches, Oates *et al.* [11] find that the hybridization of dynamic time warping and hidden Markov model forms better clusters than either approach alone on artificial data. Their idea is that DTW and HMM methods complement each other: DTW produces a rough initial clustering and the HMM dynamically redeploys the sequences that do not belong to suitable clusters. The downside is that the HMM may transfer some good sequences along with the bad ones. In [4], we have proposed a new framework for clustering time series which is based on a hybrid model that uses a recently proposed *graph b-coloring based clustering approach* [5] and *Markov chain models*.

The *graph b-coloring* [12] is the assignment of colors (clusters) to the vertices of the graph such that: (*i*) no two adjacent vertices (vertices joined by an weighted edge representing the dissimilarity between sequences) have the same color (*proper coloring*), (*ii*) for each color, there exists at least one vertex (called *dominating vertex*) which is adjacent (has a sufficient dissimilarity degree) to all other colors. The *b-coloring based clustering method* enables to build a fine partition of the sequential data set in clusters when the number of clusters is not specified in advance. It consists of two steps: 1) generate an *initial proper coloring* of vertices using a maximum number of colors, and 2) removing each color that has no dominating vertices yet using a *greedy procedure*.

3 An Integrated Clustering Framework

In this section, we discuss our *probabilistic b-coloring* based clustering framework. Our approach formulates the sequence clustering problem as a kind of graph partitioning problem in a weighted linkage graph $G = (\boldsymbol{V}, \boldsymbol{E})$, where $\boldsymbol{V} = \{v_1,v_2,...,v_n\}$ is the vertex set which correspond to the sequences set $\boldsymbol{S} = \{S_1,S_2,...,S_n\}$, and $\boldsymbol{E} = \boldsymbol{V} \times \boldsymbol{V}$ is the edge set which correspond to higher dissimilarities than given threshold θ and are weighted by their dissimilarities. The graph G is traditionally represented with the corresponding weighted dissimilarity matrix, which is the $n \times n$ symmetric matrix $D=\{d_{i,j}| S_i,S_j \in \boldsymbol{S}\}$.

The goal is to divide the vertex set $\boldsymbol{V}$ of the superior threshold graph G into a partition $\boldsymbol{P_k}=\{C_1,C_2,..,C_k\}$ where for $\forall\ C_i,C_j \in \boldsymbol{P_k},\ C_i \cap C_j=\varnothing$ *for* $i\neq j$ (when the number of clusters k is not pre-defined) and each cluster C_i is governed by a Markov chain with parameters $\phi_i=(\pi_i,a_i)$. The different steps of our framework are illustrated in the following figure and detailed in the sequel of this section.

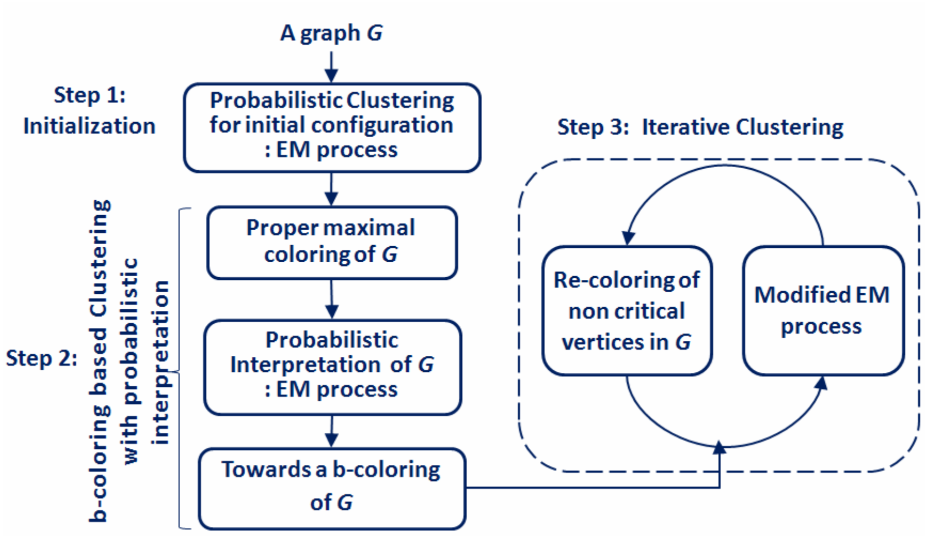

A graph G
Step 1: Initialization
Probabilistic Clustering for initial configuration : EM process
Step 2:
b-coloring based Clustering with probabilistic interpretation
Proper maximal coloring of G
Probabilistic Interpretation of G : EM process
Towards a b-coloring of G
Step 3: Iterative Clustering
Re-coloring of non critical vertices in G
Modified EM process

Fig. 1. The different steps of our integrated clustering framework

3.1 Notations

In the remaining of the paper, assuming that the vertices of G are colored, the following notations will be used:

The following routines are proposed for the rest of procedures:

- *Update($N_c(v_i)$)* is the method which updates the neighborhood colors of the vertex v_i when the color of at least one of its neighbors has changed.
- *Enqueue(x,X)* is the method which adds the vertex (resp. color) x into the vertex (resp. color) set X.
- *Dequeue(x,X)* is the method which removes the object x from the set X.

Table 1. Notations table 1

Symbol	Description
Δ	Maximum degree of a graph *G*.
$c(v_i)$:	Color (integer value) of the vertex v_i in *G*.
$N(v_i)$:	Neighborhood of vertex v_i in *G*.
$N_c(v_i)$:	Neighborhood colors of vertex v_i.
C:	Set of colors used in the graph (one set of integer values) .
D_m	Set of colors which have dominating vertices.
ND_m	Set of colors that have no dominating vertex.
$ac(v_i)$	The appropriate color to containing v_i among all the colors in *G*. It is the color which gives the highest probability of that sequence being generated by its associated Markov chain (eq.(1)).

3.2 Step 1: Initialization with Probabilistic Interpretation

Since the *initialization step* of our original b-coloring clustering approach in [5] provides an initial proper coloring of *G* using the maximum number of colors available (Δ+1), we propose to, as a first of our framework to:

1. Run, *before the initialization coloring step*, the *EM algorithm*, in order to estimate the parameters of the (Δ+1) Markov chains associated to the (Δ+1) required colors where:
   - o the number of clusters is Δ+1,
   - o The initial partition is given using a random operation.
2. Compute the n* Δ+1 matrix **P** where:
   - o **P(i,j)** is the *conditional probability* that the sequence $S_i=\{e_{i,1}, e_{i,2},..., e_{i,Ti}\}$ is generated by the Markov chain $\phi_j.=(\pi_j,a_j)$ associated to the cluster C_j:

$$P(i,j)=P(S_i|c(S_i)=j,\phi_j)=P(S_i|\phi_j)=\pi_j(e_{i,1})\prod_{t=1}^{T_i-1}a_j(e_{i,t},e_{i,t+1}) \qquad (1)$$

where:

- π_j: the initial state probability vector (probability to emit a symbol *e* at *t*=0).
- a_j: the transition matrix $m*m$.

3.3 Step 2: b-Coloring Based Clustering with Probabilistic Interpretation

3.3.1 Step 2.1: The Proper Maximal Coloring of G

Let us consider an additional notation ***T*** to denote the set of treated (colored) vertices which are sorted on descending order of their degrees and ascending order of their probability to being generated by the Markov chain of their clusters (eq.(1)). Initially, since vertices of *G* are not already colored, ***T*** is an *empty set* $\{\varnothing\}$ and will be updated as long as *Step 2.1* runs, as shown below.

This step provides an initial proper coloring of the superior threshold graph *G* using the maximum number of colors available, i.e. Δ+1 and based on the returned results from the previous step. In order to reduce the sensitivity of the initialization step (for original b-coloring clustering approach) to the selection colors manner for some vertices, we propose to incorporate the probability measure in the algorithm. Indeed, the following rules are taken into account:

- If there is a choice between some vertices with maximum degree for starting, the one having maximum probability to being generated by the Markov chain of its appropriate cluster will be selected.
- If there is a choice between many colors for one vertex v_i, the color generating the maximum probability for v_i (eq.(1)) will be selected to color it.

The procedure *proper_maximal_coloring* starts from one vertex of $\boldsymbol{V}$ which has the maximum degree Δ (let v be such a vertex). The algorithm puts $c(v)=1$ and adds v to the set $\boldsymbol{T}$. Then *proper_maximal_coloring* tries to color the remaining vertices according to the following principle: for each vertex v_i belonging to $\boldsymbol{T}$ (initially $\boldsymbol{T}=\{v\}$), a new color is assigned to each one of its neighbors v_j if it is not already colored and which will be then added to $\boldsymbol{T}$. In order to give a proper vertex-coloring of G with a maximum number of colors, the color of v_j should be different from those of its neighbors (which gives what is called a *proper vertex-coloring* of G) and -if possible- different from the one of v_i neighbors (for allowing to x to become dominating vertex). If all existing colors in G do not satisfy both constraints, the selected color for v_j must at least verify the first one (i.e. the color must be different from those of its neighbors). For maximizing the probability $\boldsymbol{p_c}(h)$ of each cluster (color) C_h (eq.(2)), the color giving the *maximal probability* for v_j (eq.(1)) will be selected if there is a choice between many colors for v_j.

$$p_c(h)=\prod_{i=1,\dots,n|c(S_i)=h}^{n} P(S_i|\phi_h) \tag{2}$$

After the coloring of every neighbor v_j of the vertex v_i, the procedure checks if the color $c(v_i)$ is a new dominating color. After that, the vertex v_i is removed from $\boldsymbol{T}$.

```
Procedure  proper_maximal_coloring()
BEGIN
C := {1,2,.., Δ+1};
for each vertex $v_i$
   $ac(v_i)$=$\mathrm{argmax}_{h\in C}$(P(i,h));//$ac(v_i)$ is the appropriate color for $v_i$
v=$\mathrm{argmax}_{v_i\in V/d(v_i)=\Delta}$(P(i,ac($v_i$)); // d($v_i$) is the degree of $v_i$
c(v):=ac(v);
Repeat
  Select $v_i$ from T;
  M := $N_c(v_i)\cup\{c(v_i)\}$;
  for each vertex $v_j \in N(v_i)$ such that $c(v_j)=\varnothing$ do
    H := {h| h∈C and h∉M and h∉ $N_c(v_j)$};
    if (H≠∅) then c($v_j$):= $\mathrm{argmax}_{h\in H}$(P(j,h));
    else H := C\$N_c(v_j)$; c($v_j$):= $\mathrm{argmax}_{h\in H}$(P(j,h)); endif.
    Enqueue($v_j$,T);
    for each vertex $v_h \in N(v_j)$ Enqueue(c($v_j$),$N_c(v_h)$);
  enddo.
  if $N_c(v_i)$=C\{c($v_i$)}then Enqueue(c($v_i$),$D_m$); endif.
  Dequeue($v_i$,T);
Until(T=∅)
END.
```

3.3.2 Step 2.2: Probabilistic Interpretation of Proper Coloring

In order to re-estimate the parameters of the Marko chain associated to each cluster and the matrix **P**, the clusters returned from the *proper maximal coloring step* is considered as an input to a new training EM process. Indeed, we propose to:

1. Run the EM algorithm where:
   - the number of clusters is Δ+1**,**
   - The initial partition is the one returned from the initial coloring step (initial configuration).
2. Compute the n* Δ+1 matrix **P**.
3. Compute the Δ+1 vector $\boldsymbol{p_c}$ where:
   - $\boldsymbol{p_c}(h)$ is the probability of the cluster h given by formula in eq.(2).

3.3.3 Step 2.3: Towards a b-Coloring of *G*

After performing *the initial coloring step 2.1*, some assigned colors remain without any *dominating vertex*. Our objective now is to find a *b-coloring* of graph G where all colors are *dominating colors*. The idea is the following: each non-dominating color can be changed. In fact, after choosing one non-dominating color c (If there is many non dominating colors, the one having the minimum probability $p_c(c)$ is selected) and removing it from the graph G, for each vertex v_i colored with c (i.e. $c(v_i)=c$), a new color is assigned to v_i which is different from those of its neighborhood. When there is a choice between many colors to color v_i, the color giving the maximum probability (eq.(1)) for the sequence S_i will be selected for it. Before starting again with another non dominating color c', the procedure verifies if some non dominating colors became dominating (in such a case, these colors will not be modified in the remaining of the step).

```
Procedure find_b-coloring()
BEGIN
Repeat
  $ND_m$:= $C\backslash D_m$;
  c:=$argmin_{h \in ND_m}(p_c(h))$;
  C := C\{c};
  $ND_m$:= $C\backslash D_m$;
  for each vertex $v_i$ such that $c(v_i)=c$ do
       H := $C\backslash N_c(v_j)$;
       $c(v_j)$:=$argmax_{h \in H}(P(j,h))$;
  enddo.
  for each vertex $v_j$ such that $c(v_j) \in ND_m$ do
    Update($N_c(v_j)$);
    if $N_c(v_j)=C\backslash\{c(v_j)\}$then Enqueue($c(v_j)$,$D_m$);  endif.
  enddo.
Until($ND_m$ =$\emptyset$)
END.
```

3.4 Step 3: Iterative Clustering

In order to improve the quality of the returned partition, we try, using a greedy procedure, to re-color some vertices providing that the b-coloring properties are maintained. The main idea is to alternate between the EM and a re-coloring process until the coloring is stable, i.e. no change between two successive partitions.

3.4.1 Definitions and Notations

The following definitions are introduced:

Definition 1. A vertex v_s is called "*supporting vertex*" if v_s is the only vertex colored with $c(v_s)$ in the neighborhood ($N(v_d)$) of one dominating vertex v_d. Thus, v_s cannot be re-colored.

Definition 2. A vertex v_f is called "*finished vertex*" if v_f is already checked for re-coloring.

Definition 3. A vertex v_c is called "*critical vertex*" if v_c is a *dominating*, *finished* or *supporting* vertex. Thus, v_c cannot be re-colored.

Under these definitions, we use the following additional notations for this step:

Table 2. Notations table 2

Symbol	Description
V_d	Dominating vertices set.
V_s	Supporting vertices set.
V_f	Finished vertices set. It contains the vertices which are already checked for re-coloring.
V_c	Critical vertices set. $V_c=V_d \cup V_s \cup V_f$.

3.4.2 Modified EM Process: It Is Performed, Where:

- The number of clusters is k (the returned clusters number from the find-b-coloring procedure).
- The initial partition is the one returned from the previous iteration denoted by P_k. For the first iteration, P_k corresponds to the partition returned from the *find-b-coloring* procedure.

Since the dominating vertices reflect the properties of their clusters and also guarantee that their clusters has a *distinct separation* from other clusters and cannot be re-colored (they must be assigned to only their colors returned from the *find-b-coloring* step), we assume that only the colors of *non dominating sequences* can be changed. The idea is to allow these sequences to be assigned to more than one model. Formally, this proposition is given as follow: At each iteration of the EM process, every *dominating sequence* S_d is assigned to the Model (Markov chain) of its cluster returned from the *find-b-coloring process* (i.e. $c(S_d)$). This condition is fulfilled by considering, in the *Expectation step*, that the probability that the sequence S_d belongs to the cluster $c(S_d)$ is equal to 1 and 0 to all other clusters, during the execution of the EM process. Consequently, the sequence S_d is used to re-estimate only the parameters of the cluster $c(S_d)$ (in the *Maximisation step*). However, the probability that a *non dominating sequence* S_i belongs to one cluster $c=1,2,\ldots,k$ is given using the original formula in *eq.*(3) and this sequence is used in the re-estimation of the parameters of all cluster models.

$$P(c_i = c|S_i, \phi) = \frac{P(S_i|c_i = c, \phi_c) * P(c)}{\sum_{u=1}^{k} P(S_i|c_i = u, \phi_u) * P(u)} \quad (3)$$

where $\phi = \{\phi_1, \phi_2, ..., \phi_k\}$ represents the parameters of the partition ***P*** and $P(c)$ is the *membership probability* of cluster c.

Our idea is summarized in the following procedure. Generally, the termination condition of EM process is reached when there is low change of the log likelihood, given by the following formula.

$$log(P(S|\phi)) = \sum_{i=1}^{n} log(P(S_i|\phi)) = \sum_{i=1}^{n} log\left(\sum_{i=1}^{n} P(S_i|\phi_k) * P(k)\right) \quad (4)$$

The time complexity of this procedure is linear in the sum of the lengths of all sequences in ***S***. At the end of *modified EM procedure*, the new **n* k** matrix **P** is computed.

```
Procedure Modified_EM()
Require: $P_k$; //a partition which is a b-coloring of $G$=(V,E)
BEGIN
Initialize EM with $P_k$;
Repeat
Expectation step:
  for each sequence $S_i$ related to a vertex $v_i \in V$ do
      if $v_i \in V_d$ then // $v_i$ is dominating
         $P(c_i=c(S_i)|S_i,\phi) := 1$;
         for each cluster $c \in \{1,2,..,k\}$ such that $c \neq c(S_i)$
              $P(c_i=c|S_i,\phi) := 0$;
      else
         for each cluster $c \in \{1,2,..,k\}$

      $P(c_i=c|S_i,\phi) := P(S_i|c_i=c,\phi_c)*P(c)/Sum_{u=1,2,...,k}(P(S_i|c_i=u,\phi_u)*P(u))$;
      endif.
  enddo.
Maximization step: The same as for original EM
Until(EM termination condition)
END.
```

3.4.3 Greedy Re-Coloring Process: It Is Performed under the Following Arguments

- The colors of all dominating vertices V_d should be kept. Regarding the definition of a dominating vertex, each $v_d \in V_d$ is connected to the vertices with all the other colors. Consequently, such vertex is considered as important to ensure a large cluster disparity and its color should not be changed.
- The colors of all supporting vertices $v_s \in V_s$ should not be changed, because changing c(v_s) to other color can make some $v_d \in V_d$ as non-dominating and thus deteriorates the quality of a partition. The idea to not to re-color these vertices allows to guarantee the dominance property of b-coloring of G as long as the re-coloring process runs.
- Under the previous hypothesis, only the vertices in $V \backslash \{V_d \cup V_s\}$ should be considered for re-coloring. In order to guarantee the termination of our

re-coloring process, each vertex considered for re-coloring is checked at once and moved into $\boldsymbol{V_f}$. Thus, in each step, we consider the re-coloring of only *non critical vertices* in $\boldsymbol{V_{nc}=V\backslash\{V_d \cup V_s \cup V_f\}}$.

- Among $\boldsymbol{v \in V_{nc}}$, we select $\boldsymbol{v}$ with the minimal $\boldsymbol{P}(v,c(v))$ and we re-color it providing that this transformation maintains the b-coloring properties. Indeed, when the vertex $\boldsymbol{v}$ is selected for re-coloring, we check the colors in $\boldsymbol{C_p(v)=P_k \backslash N_c(v)}$ and select the one $\boldsymbol{c}$ with the maximal $\boldsymbol{P}(v,c)$. The vertex $\boldsymbol{v}$ is then moved to $\boldsymbol{V_f}$ and the algorithm tries to select another vertex in $\boldsymbol{V_{nc}}$ until it became empty. Since, this re-coloring is repeated for each $v \in \boldsymbol{V_{nc}}$, when algorithm terminates, the following index monotonically increases:

$$f=\prod_{i=1}^{n} P\left(S_i \middle| \phi_{c(i)}\right)=\prod_{i=1}^{n}\left(\pi_{c(i)}\left(e_{i,1}\right)\prod_{t=1}^{T_i-1} a_{c(i)}\left(e_{i,t},e_{i,t+1}\right)\right) \tag{5}$$

- When the color $c(v)$ of $\boldsymbol{v \in V_{nc}}$ is re-colored to c, some $\boldsymbol{v_{nc} \in V_{nc}}$ might become new critical vertices, because (1) some vertices can become dominating (these vertices are in the neighborhood of $\boldsymbol{v}$ and needed only one neighbor in the new color of v to become dominating and are checked using the *find_new_dominating()* routine), or (2) some vertices can become supporting ones (checked using the *find_new_ supporting()* routine).

```
Procedure re-coloring()
Require: G=(V,E);//A graph with a set of vertices and a set of edges.
Require: $P_k$; //a partition which is a b-coloring of G =(V,E)
BEGIN
C=$P_k$;
Divide V into $V_c \cup V_{nc}$
while $V_{nc} \neq \emptyset$ do
   v:=$\text{argmin}_{v' \in V_{nc}}$(P(v',c(v')));
   c(v):=$\text{argmax}_{c \in C_p(v)}$(P(v,c));//re-coloring of v
   $V_f := V_f \cup \{v\}$;
   find_new_dominating();
   find_new_supporting();
   $V_c := V_d \cup V_s \cup V_f$.
   $V_{nc} := V \backslash V_c$;
endwhile
END.
```

4 Evaluations

Experiments have been made using two relevant benchmark sequential data sets chosen from UCI machine learning repository [13]. The first data set (*promoter*) consists of strings that represent nucleotides (one of A, G, T or C). The input features are 57 sequential DNA nucleotides and the total number of instances is 106. The second data set (*splice*) contains 3190 instances of 60 sequential DNA nucleotides (A, G, T or C).

Since we deal with sequences of categorical observations, we note that the *normalized Edit distance* (as defined in [4]) is applied to define the dissimilarity level between two sequences S_i and S_j:

$$D(S_i,S_j)=\frac{|S_i|+|S_j|-2*|LCS(S_i,S_j)|}{|S_i|+|S_j|} \tag{6}$$

On the other hand, the dissimilarity threshold θ related to the graph G is computed using to our *b-coloring based clustering approach* in [5].

The quality of the cluster obtained from *our framework* is compared to the results from the mixture Markov models framework introduced in [3] (with randomly initial partitioning and a same number of clusters k as returned from our framework).

For an interesting assess of the results gained with both clustering frameworks, our evaluation will be based on two quality indices:

Intracluster homogeneity index (*IH*): such index is fundamental in the cluster validation problem. Considered as a probability scheme, the *intracluster homogeneity* is used to reflect the compactness of the discovered clusters. The greater this value, the more cohesive are clusters of partition. For a partition $\boldsymbol{P_k}=\{C_1,C_2,...,C_k\}$ of $\boldsymbol{S}=\{S_1,S_2,...,S_n\}$, this function is defined as the average *intracluster homogeneity* of all the k clusters of $\boldsymbol{P_k}$ ($\eta_i = |C_i|$ is the cardinality of one cluster C_i) as follows:

$$IH=\frac{\sum_{c=1}^{k} IH_c}{k} \quad such\ that \quad IH_c=\frac{\sum_{S_i\in c}\delta_i}{n_c} \tag{7}$$

where:

- $\begin{cases}\delta_i=0; if\ P(c(S_i)=c|S_i,\phi)<0.5\\ \delta_i=1; if\ P(c(S_i)=c|S_i,\phi)\geq 0.5\end{cases}$
- $P(c_i=c|S_i,\phi)$ is the probability that the sequence S_i belongs to the cluster c (eq.(3)).

Prediction performance index (*PP*): The key idea is to select the sequence $S_i=\{e_{i,1}, e_{i,2},..., e_{i,Ti}\}$ within a data set X sequences separately and to:

- Eliminate the final state $e_{i,Ti}$ from the sequence S_i.
- Classify the truncated sequence (called $S_{trucated}$) in one of the k current clusters using the *online property* given by the formula in *eq.*(6). The chosen cluster is denoted by c_i.

$$c_i = argmax_{1\leq c\leq k}\{P(S_{truncated},c)\} \tag{8}$$

- Predict the next state z in the sequence by using the transition matrix of the selected cluster c_i. This state will be compared with the original eliminated state $e_{i,Ti}$. The comparison value called ω_i is given as follow:

$$\omega_i=\begin{cases}1; if\ e_{i,T_i}=argmax_{1\leq z\leq m}\{a_{c_i}(e_{i,T_i-1},z)\}\\ 0; \quad otherwise\end{cases} \tag{9}$$

The *Prediction performance index* PP_X of a data set X is defined as:

$$PP_X=\frac{\sum_{i\in X}\omega_i}{|X|} \tag{10}$$

We used 5-fold cross validation for the evaluation with the previous indices. Table 3 provides the clustering results (*intracluster homogeneity* and *prediction performance*) obtained over the two samples mentioned above, using our integrated framework and the mixture Markov models framework. Both *indices* indicate better partitioning for our framework with a large variety on the *intracluster homogeneity*. The clusters are compact and well-interpretable. This confirms the pertinence of our framework to offer a compromise between the *intracluster cohesion* and the interpretability of the returned partition.

Table 3. Frameworks Performances on benchmark data sets

Data set	Framework	# instances	*k*	IH	PP (training)	PP (test)
Promoter	Our	106	2	0.91	0.35	0.43
	Cadez			0.26	0.33	0.29
Splice	Our	3190	3	0.83	0.37	0.34
	Cadez			0.006	0.36	0.29

5 Conclusion

In this paper, a new integrated framework for sequential data clustering was presented. It is based on a hybrid model that uses a modified *b-coloring based clustering approach* and the iterative EM process. The underlying clustering method result is that each cluster is described by dominant members, as well as a finite-state Markov chain model linked to each cluster, where the b-coloring properties are maintained as long as the clustering process runs. We have implemented, performed experiments, and compared our framework to other mixture Markov models framework, and illustrated its efficiency on benchmark data sets.

There are many interesting issues to pursue: (1) leading more experiments and comparison for our framework, and (2) extending our framework to deal with the clustering of semantic web services based on their behaviour modelling, to name a few.

Acknowledgments

The first author was supported by JSPS, Japan (PE 07555) for his stay in Japan. The second author was supported by Canon Foundation Fellowship for his stay in France. The authors are grateful to these grants. This work is partially supported by the grant-in-aid for scientific research (No. 20500123) funded by MEXT, Japan.

References

[1] Antunes, C., Oliveira, A.: Temporal data mining: an overview. In: KDD Workshop on Temporal Data Mining, pp. 1–13 (2001)

[2] Cadez, I.V., Heckerman, D., Meek, C., Smyth, P., White, S.: Visualization of navigation patterns on a Web site using model-based clustering. In: Knowledge Discovery and Data Mining, pp. 280–284 (2000)

[3] Cadez, I.V., Gaffney, S., Smyth, P.: A general probabilistic framework for clustering individuals and objects. In: Knowledge Discovery and Data Mining, pp. 140–149 (2000)
[4] Elghazel, H., Deslandres, V., Kallel, K., Dussauchoy, A.: Clinical Pathway Analysis Using Graph-Based Approach and Markov Models. In: The Second IEEE/ACM International Conference on Digital Information Management, Lyon, France, pp. 279–284 (2007)
[5] Elghazel, H., Deslandres, V., Hacid, M.S., Dussauchoy, A., Kheddouci, H.: A new clustering approach for symbolic data and its validation: Application to the healthcare data. In: Esposito, F., Raś, Z.W., Malerba, D., Semeraro, G. (eds.) ISMIS 2006. LNCS (LNAI), vol. 4203, pp. 473–482. Springer, Heidelberg (2006)
[6] Gunopulos, D., Das, G.: Time series similarity measures (tutorial pm-2). In: Tutorial notes of the 6th ACM SIGKDD (2000)
[7] Jain, A.K., Murty, M.N., Flynn, P.J.: Data Clustering: A Review. ACM Computing Surveys 31, 264–323 (1999)
[8] Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM-Algorithm. Journal of the Royal Statistical Society, Series B 39, 1–38 (1977)
[9] Smyth, P.: Clustering sequences with hidden Markov models. Advances in Neural Information Processing 9, 648–654 (1997)
[10] Alon, J., Sclaroff, S., Kollios, G., Pavlovic, V.: Discovering clusters in motion time-series data. In: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 375–381 (2003)
[11] Oates, T., Firoiu, L., Cohen, P.: Clustering time series with hidden Markov models and dynamic time warping. In: Proceedings of the IJCAI 1999 Workshop on Neural, Symbolic and Reinforcement Learning Methods for Sequence Learning, pp. 17–21 (1999)
[12] Irving, W., Manlove, D.F.: The b-chromatic number of a graph. Discrete Applied Mathematics 91, 127–141 (1999)
[13] Blake, C.L., Merz, C.J.: UCI repository of machine learning databases, 1998. University of California, Irvine (1998), `http://www.ics.uci.edu/~mlearn/MLRepository.html`

Cluster Analysis in Remote Sensing Spectral Imagery through Graph Representation and Advanced SOM Visualization

Kadim Taşdemir and Erzsébet Merényi[⋆]

Rice University, Electrical and Computer Engineering
6100 Main Street, Houston, TX, 77005 - USA
tasdemir@rice.edu, erzsebet@rice.edu

Abstract. The Self-Organizing Map (SOM), a powerful method for clustering and knowledge discovery, has been used effectively for remote sensing spectral images which often have high-dimensional feature vectors (spectra) and many meaningful clusters with varying statistics. However, a learned SOM needs postprocessing to identify the clusters, which is typically done interactively from various visualizations. What aspects of the SOM's knowledge are presented by a visualization has great importance for cluster capture. We present our recent scheme, CONNvis, which achieves detailed delineation of cluster boundaries by rendering data topology on the SOM lattice. We show discovery through CONNvis clustering in a remote sensing spectral image from the Mars Exploration Rover Spirit.

1 Introduction

Self-Organizing Maps (SOMs) are widely and successfully used neural paradigms for clustering and data mining. They perform an iterative learning process which has two advantageous properties: an adaptive vector quantization that results in optimal placement of the prototypes in the data space; and ordering of those prototypes on a rigid low-dimensional lattice according to their similarities. Due to these properties, SOMs facilitate informative visualization of the structure of a higher-dimensional data space in lower (usually two) dimensions, which in turn aids interactive capture of the cluster boundaries.

Many visualization schemes have been proposed to represent distance based (dis)similarities of prototypes or the size of the receptive fields of neural units in various ways such as by screen intensity levels, or by the sizes or shapes of the SOM grid cells. The most commonly used methods include the U-matrix [1] and its variants, which represent the (Euclidean) distance of the prototypes that are neighbors in the SOM lattice. These are useful when relatively large SOM grid

⋆ This work was partially supported by grant NNG05GA94G from the Applied Information Systems Research Program, NASA, Science Mission Directorate. We thank Dr. William Farrand from the Space Science Institute, Boulder, CO, USA for his valuable contributions to the analysis of the Martian scene discussed here.

accomodates small data sets with a low number of clusters (e.g., [2], [3]) but, because of averaging of prototype distances over neighbours or thresholding, they tend to miss finer structure in complicated data [4]. Alternatively, the distances between neighboring prototypes are shown through the adaptation of the size or the shape of the grid cells [5,6]. Another approach for visual inspection is a modified SOM algorithm which updates the grid positions of prototypes based on their similarities instead of having a rigid grid (Adaptive Coordinates [7], Double SOM [8], visualization induced SOM (ViSOM) [9]). However, these either have hard-to-set parameters or are only applicable to small data sets because they require relatively large number of prototypes. Automated color assignments are also used for exploration of the coarse cluster structure [10,11,12,13]. It is popular to analyze individual component planes of the SOM to discover information specific to the corresponding component [12,14]. We point the reader to [14] and [15] for more review.

A recent graph-based SOM visualization, CONNvis, proposed by these authors [16], represents the data topology by a weighted version of the Delaunay graph and drapes this graph over the SOM. In this paper, we show how CONNvis helps detailed interactive cluster capture and knowledge discovery from remote sensing spectral imagery, through an example of a Martian scene. Section 2 briefly explains CONNvis and clustering. Section 3 discusses the application to Martian geology. Section 4 concludes and provides future directions.

2 CONNvis: A Graph Based SOM Visualization

CONNvis is a rendering of data topology – represented by a "connectivity matrix" CONN – on the SOM lattice [15]. Below we first describe CONN, then how it is used as a SOM visualization to assist interactive clustering.

2.1 Connectivity Matrix (CONN)

The connectivity matrix [15], CONN, is a weighted Delaunay triangulation, where the weight of an edge connecting two prototypes is the number of data vectors for which these prototypes are the best matching unit (BMU) and the second BMU. Formally, each element of CONN, $CONN(i, j)$, *the connectivity strength* between prototypes i and j is defined as

$$CONN(i, j) = |RF_{ij}| + |RF_{ji}| \tag{1}$$

where RF_{ij} is that section of the receptive field (Voronoi polyhedron) of the prototype i where j is the second BMU, and $|RF_{ij}|$ is the number of data vectors in RF_{ij}. CONN thus shows how the data is distributed within the receptive fields with respect to neighbor prototypes. This produces a finer density distribution than other existing density representations which show the distribution on the receptive field level. CONN also indicates the neighborhood relations of the prototypes with respect to the data manifold because a binarized CONN is equivalent to the *induced* Delaunay graph, which is the intersection of the

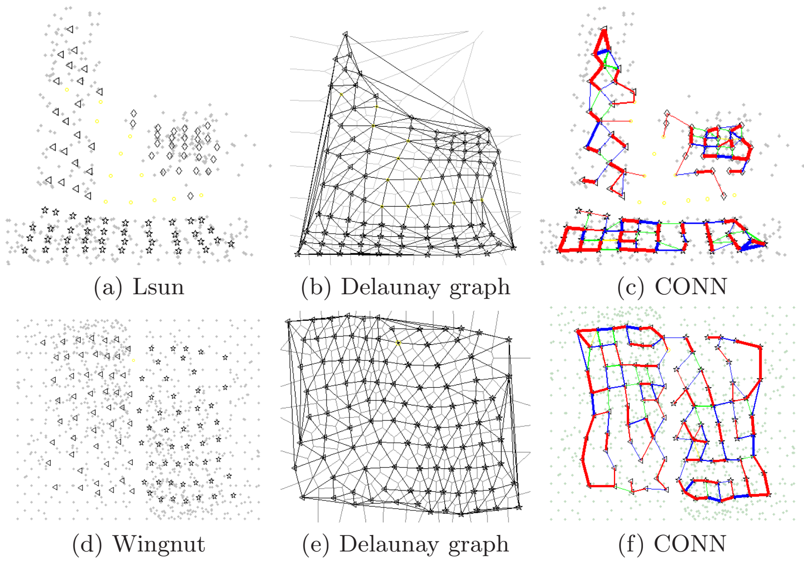


Fig. 1. Illustration of how the connectivity matrix CONN shows neighborhood relations in the data space, with two simple 2-d data sets from [18]. A 10 × 10 SOM is used to obtain prototypes. Top: (a) Lsun (3 clusters) and its prototypes with true labels in data space. (b) Delaunay triangulation of the Lsun prototypes. Empty prototypes do not have symbols. (c) CONN of Lsun prototypes. Bottom: (d) Wingnut (2 clusters with inhomogeneous density distribution) (e) Delaunay triangulation of the Wingnut prototypes (f) CONN of Wingnut prototypes. The clusters of Lsun and Wingnut can be seen through the CONN.

Delaunay triangulation with the data manifold [17]. This results in making the separations within the data set visible.

Fig. 1 shows examples of CONN for two simple 2-d data sets constructed by [18]. The first one is called “Lsun” which has three well-separated clusters (two rectangular and one spherical). The second one, “Wingnut”, has two rectangular clusters with inhomogeneous density distribution within clusters and similar intra-cluster and inter-cluster distances. For both cases, the cluster structure can be seen through CONN regardless of the variations in cluster characteristics. CONN can be visualized in the data space for low-d (1-d to 3-d) data sets. For high-d data sets, it can be rendered on the 2-d SOM lattice to represent the data topology just as informatively as in the data space, as shown in Fig. 2.

2.2 CONNvis: Rendering CONN on the SOM

CONN can be visualized on the SOM by connecting the lattice locations of the prototypes with lines of various widths and colors. The line widths are made proportional to the connectivity strengths, $CONN(i, j)$, thus show the density distribution among the connected units. The line width conveys a sense of the *global* importance of each connection by allowing comparison with all others,

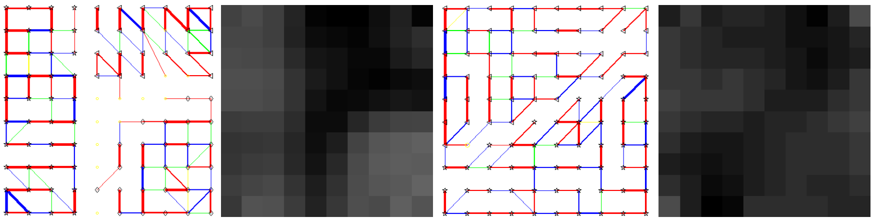


Fig. 2. CONNvis and U-matrix visualization of the 10x10 SOM prototypes of the two datasets in Fig. 1. (a) CONNvis of the Lsun. 3 clusters are visually separated through CONNvis. (b) U-matrix of Lsun. A darker grey indicates more dissimilarity. It is hard to see the 3 clusters from this visualization. (c) CONNvis of the Wingnut. 2 clusters are made visible by strong connectivity within themselves but only one weak connection across them. (d) U-matrix of Wingnut. Due to equal distances between prototypes within and across clusters, it is hard to see the separation between two clusters.

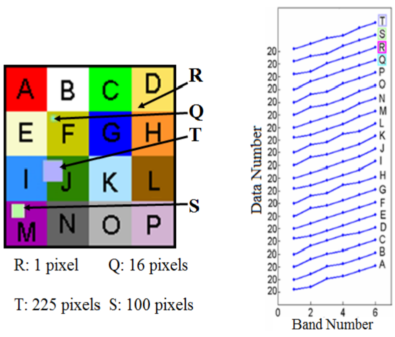
in this visualization. Line colors are used to express a ranking of the Voronoi neighbors of a prototype i in terms of the strengths of their connectivity to i : the most-to-least connected neighbors are shown with red, blue, green, yellow, and dark to light grey colors, in this order. Since this ranking does not depend on the size of the receptive field of i, but only on the relative contribution of each neighbor, the line color indicates the *local* importance of a connection. It also defines a similarity measure among the neighbors.

Fig. 2 shows examples of CONN rendered on the 10×10 SOM of the two small data sets in Fig. 1. The separations (clusters) in the data can be clearly seen through the CONNvis in Figures 2.a and 2.c. For the same data sets, the U-matrix does not resolve sufficient detail for visual extraction of the clusters in Lsun and Wingnut (Figures 2.b and 2.d.). A larger SOM (100×100) can discover these clusters through a U-matrix as shown in [3], at significantly larger computational cost. Explanatory examples of CONNvis for a more complicated 2-d data set are given in [15].

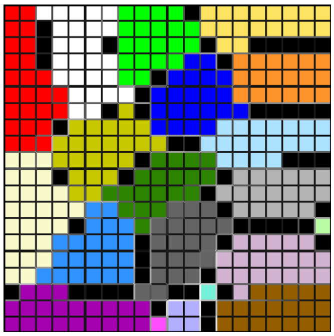
In CONNvis, using a different line width for each connectivity strength becomes infeasible, due to limitations by screen resolution and the discrimination capability of the human eye, when the number of data samples is much larger than the number of prototypes and consequently connectivity strengths span a large range of values.. To help this, a 4-level binning scheme is applied for line widths as follows:

$$width(i,j) = \begin{cases} 1 & \mu_3 > CONN(i,j) \geq \{\mu_4, 0\} \\ 2 & \mu_2 > CONN(i,j) \geq \mu_3 \\ 3 & \mu_1 > CONN(i,j) \geq \mu_2 \\ 4 & CONN(i,j) \geq \mu_1 \end{cases} \quad (2)$$

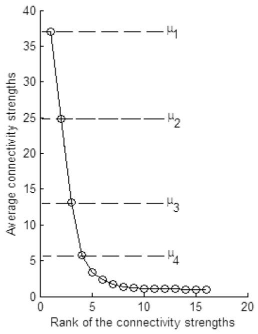
where $width(i,j)$ is the width of the line between prototypes i and j, μ_n is the mean strength of the n^{th} ranking connections. This choice provides an automated selection of thresholds based on internal data characteristics and also employs the limited number of bins effectively, because each bin reflects the global



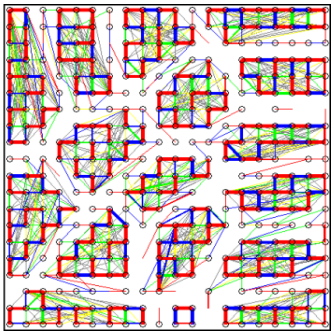
(a) 20-class, 6-d data



(b) Known labels on the SOM



(c) The connectivity statistics



(d) CONNvis

Fig. 3. (a) Left: the spatial distribution (class map) of the 20 different types of 6-d feature vectors in the 128×128 pixel image. The known types are labeled by both colors and letters. Right: the mean signatures of the 20 classes, offset for clarity. (b) Known class labels overlain on the SOM (c) Mean connectivity strengths for each rank of the connections. (d) CONNVis on the SOM. Coarse clusters (strongly connected groups of prototypes) can be detected in spite of the presence of many topology violations, *i.e.*, connections between prototypes that are not neighbors in the SOM lattice.

importance of one rank. The resolution of this binning not only distinguishes strong connections but also reveals weak connections across densely connected SOM regions, thus those regions can be recognized as clusters as shown in Figure 3.

In Fig. 3, we analyze a 128 x 128 pixel synthetic image, where each pixel is a 6-d feature vector (the analog of a spectrum). This image has 20 classes, 4 of which are relatively small and one class has only one pixel. The signatures of these 20 classes are also similar to each other, to make clustering challenging (Fig. 3.a). A 20×20 SOM is used to obtain the quantization prototypes of this data set. The statistics of the ranked connectivity strengths in Fig. 3.c indicate that the maximum number of connections for a prototype is 16. The average connectivity strength is as high as 37 ($= \mu_1$) for the first ranking (red) connections and it drops sharply after the fourth ranking (yellow) connections

($\mu_4 = 6$). The CONNvis of this data, shown in Fig. 3.d, is obtained by using the 4-level binning scheme in eq. 2.2 with $\{\mu_1, \mu_2, \mu_3, 0\}$, given in Fig. 3.c, as the thresholds. It makes strongly connected groups of prototypes (coarse clusters) emerge.

Due to the representation of data topology on the SOM lattice, CONNvis also helps in a detailed assessment of topology preservation. For example, there are many grey connections in Fig. 3 which are between pairs of prototypes that are not immediate neighbors in the SOM lattice (*forward* topology violations). There are also prototypes neighbored in the SOM lattice but unconnected (which means they are not Voronoi neighbors in the data space), indicating *backward* topology violations. The more data vectors contribute to a given connection that expresses topology violation (thus termed *violating connection*), the more severe the violation. For a topology violating connection, low strength (thin line) usually indicates outliers or noise while a greater strength is due to data complexity or badly formed SOM. The *folding length* of the violating connection, that is the maximum norm distance between the connected neural units in the SOM lattice, describes whether the topology violation is local (short ranged) or global (long ranged). In [15] we define a connection as "global" if the placement of the corresponding Voronoi-neighbor prototype occurs outside of the "tightest possible" SOM neighborhood.

Similarly to the CONNvis in Fig. 3.d, perfect topology preservation is not necessary for cluster extraction in most cases. Weak global violations, or violations that remain within clusters may not affect the delineation of boundaries. However, accurate assessment of such conditions for a trained SOM is important. The connectivity matrix and its visualization, introduced above, is a useful tool for such analysis.

2.3 Interactive Clustering from CONN Visualization

Interactive clustering is done from CONNvis by pruning. First, topology violations are investigated and weak global ones are removed. Strong global violations require further analysis to see whether they are caused by data complexity or by bad SOM learning. For example, the CONNvis of Lsun and Wingnut, shown in Fig. 2, have no topology violations (as expected). The CONNvis of the 20-class data set (Fig. 3) has only weak violations. The global violations (as defined in [15]) will be those connections with *length* > 2 since a prototype has at most 16 neighbors in the data space (from Fig. 3.c), and these can fit in the tightest possible SOM neighborhood: 8 of them in the immediate square SOM neighborhood (at length=1) and the remaining 8 in the next tier of SOM neighborhood (at length=2). After removal of those global violations, strongly connected groups of prototypes (coarse clusters), can be seen, usually with some number of weak connections across them. The prototypes with these weak connections to the coarse clusters, i.e. prototypes at the cluster boundaries (such as shown by black dots in Fig. 4.a), are visually identified by the human analyst.

Crisp delineation of the coarse clusters is determined by evaluating the connections of those prototypes at the cluster boundaries. For any prototype at the cluster boundary, one of the three situations can occur: 1) it may have different number of connections to each cluster; 2) it may have one connection to each cluster with different strengths (widths); 3) it may have one connection to each cluster with different ranking (color). For each situation, a corresponding criterion is applied to remove connections as follows and as illustrated in Fig. 4.a-b.

1. If the number of connections to the two cluster differs, remove the connections to the cluster with fewer connections
2. If the prototype has the same number of connections to each, remove the weaker set of connections.
3. If the prototype has the same number and strength of connections to each cluster, remove the lower ranking connection.

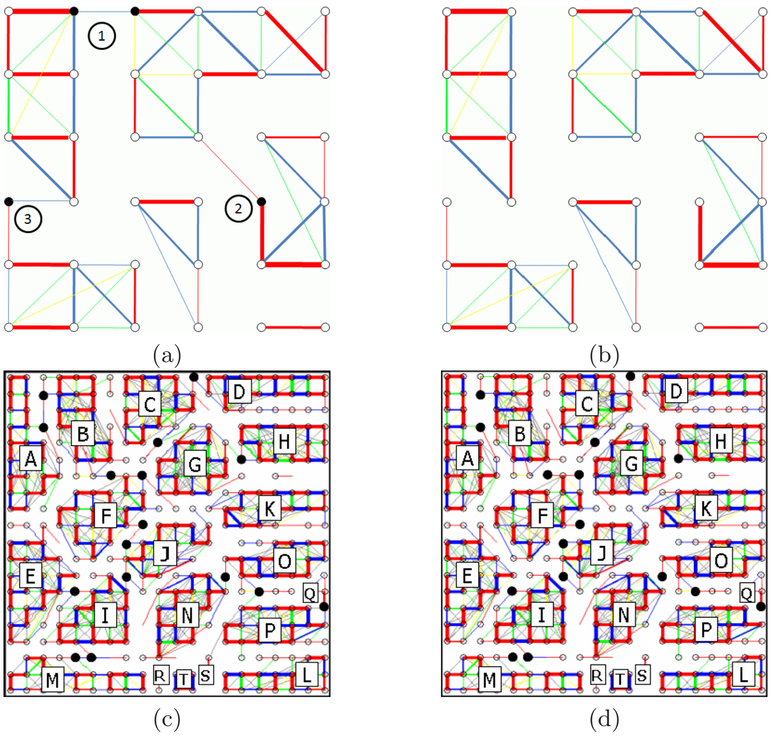


Fig. 4. Illustration of interactive clustering from CONNvis. Some groups of prototypes are outlined with the lack of connections. The prototypes at the cluster boundaries are shown by black dots. (a) The three situations at coarse cluster boundaries, for which one has to decide how to cut connections. 1: prototypes with different number of connections to each coarse cluster; 2: a prototype with the same number of connection to each cluster but with differentconnectivity strengths; 3: a prototype that has a connection to eachcluster with the same strength but different ranking. (b) Extracted clusters (separated groups of prototypes) (c) CONNvis of the 20-class data with global violations (*length* > 2) removed (d) Resulting clusters after interactive clustering. All classes, including one-pixel class R, are captured.

Figures 4.c-d illustrate the interactive clustering of the 20-class data with the above three criteria. Removal of connections across the coarse clusters results in extraction of the 20 known classes in the data, including the one pixel class R.

3 Interactive Clustering of Surface Materials in a Mars Exploration Rover Spectral Image

From sol 159 of its mission to the present day, the Spirit rover has been exploring a region dubbed the Columbia Hills (in honor of the space shuttle Columbia astronauts). Rocks found in the Columbia Hills generally have clastic textures and several chemical classes have been identified [19]. These rock types have also been analyzed in terms of their visible and near infrared (VNIR) multispectral properties [20,21] and thermal infrared emittance [22]. The scene analyzed here was collected on sol 608 of Spirit's mission near the summit of Husband Hill (named after shuttle commander Rick Husband), by the multispectral Pancam instrument. It is a 700×450 pixel 7-band image from the Pancams right "eye" (one band centered at 432 nm and six more bands with central wavelengths from 754 to 1009 nm). The image includes the rock outcrop named "Tenzing" and the rock target "Whittaker" as well as other unnamed rocks.

We use a 40×40 SOM and cluster the SOM prototypes interactively through CONNvis (Fig. 5) by visual determination of the coarse clusters and removal of the weak connections at the boundaries of these coarse clusters, as described in Section 2.2. The cluster map of the scene is shown in Fig. 6. 17 geologically meaningful clusters are extracted. Shadows and shaded regions, which do not provide geologically meaningful spectral characterization, are also shown by different grey color codes. Some clusters match well known surface units (spectral classes) identified in previous analyses [21] whereas some could be subtypes of the more representative spectral classes. Mean spectra of these 17 clusters are shown in Fig. 7 where similar spectra are placed next to each other for easy comparison of subcluster characteristics.

The far field of the scene (upper part of the image, most extensively covered by the orange class, H) is a mixture, most closely resembling the spectral signature of soil with heavy influence from rock fragments and an airfall dust component. There are also soils with presumed basaltic compositions. Some of these presumed basaltic soils (J, e) are darker-toned and coarse-grained whereas some (I, L, S) are lighter-toned and finer-grained due to much more oxidization. Among the lighter-toned soils, some (bright drift, N) are higher in albedo and are likely mobile drifts whereas some (intermediate albedo soils, L) are likely immobile. The dune near the summit of Husband Hill, which is mostly covered with bright drift (N) and intermediate albedo soil (L), is separated from the far field with a border (C, Q) that is possibly a mixture of sand grains.

The big rock (mostly covered by the orange class due to airfall dust) near the horizon is the Tenzing outcrop. Tenzing is a subtype of the Jibsheet

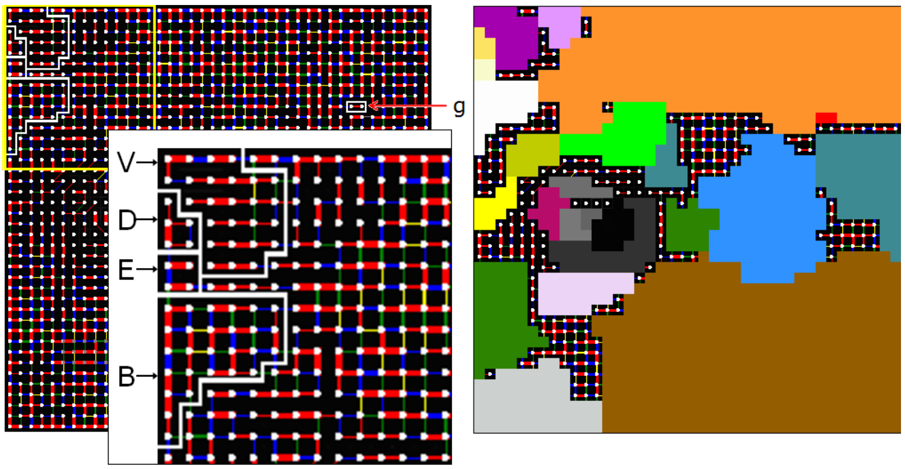


Fig. 5. Left: CONNvis of the 40 × 40 SOM of the Husband Hill image. The boundaries of some clusters are outlined to illustrate the results of interactive cluster extraction from CONNvis. An enlarged view of the upper left area in the yellow rectangle is shown in the inset for a clearer view. Right: Extracted clusters, color coded according to the color wedge in Fig. 6, overlain on the CONNvis. The clusters overlain on the Husband Hill image is shown in Fig. 6.

spectral class, characterized by a shallow infrared absorption feature (centered between 900 and 934 nm) [21]. The white rocks (B) in the scene are Jibsheet type rocks whereas yellowish (D, E) rocks are rocks with a variant of the typical Jibsheet spectral signature. The slight but consistent spectral differences (Fig. 7) of two subclusters (D and E) of Tenzing may reflect different textural properties (rougher/smoother textures) of some rocks near the summit of Husband Hill (an observation by the MER team, for example, [23]). Fig. 7.b shows the slight differences among the spectra of subclusters B, D and E and their comparison to the Jibsheet superclass in [21]. There are also rocks in the scene which have spectra close to that of the nearby rock target Bowline. The clusters V (purple) and P (olive green) correspond to the Bowline-type rocks in the scene where P has a shallower near-infrared band compared to V and appears only in one rock. Dark purple (c) areas are partly shaded Bowline rocks whereas light pink (h) areas are a mixture of small Bowline rocks and soil.

These 17 clusters are spatially coherent and the spectral signatures of many of them have been associated with geologic meaning (either in earlier analyses, or in this analysis as geologically interpretable variations within formerly recognized classes). One cluster, g (color coded red, and shown in the white oval at the top left in Fig. 6), is a new discovery in this image. Its signature is sufficiently distinct to warrant a separate cluster, however, by analyzing the spectral relations with other classes one can arrive at a hypothesis as to the physical nature of this unit.

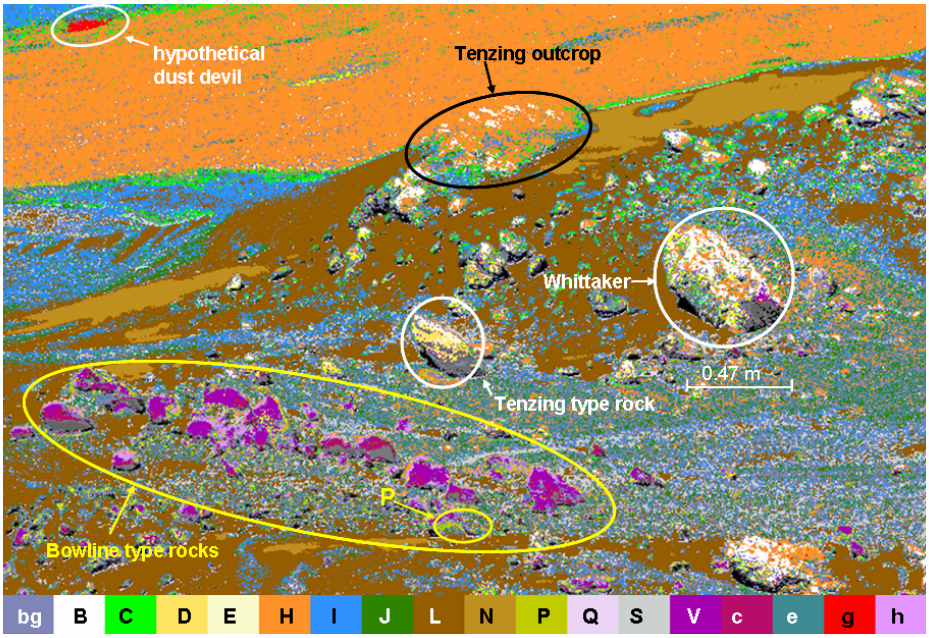


Fig. 6. Cluster map of the scene near the summit of Husband Hill (MER Pancam image taken on Sol 608) with clusters extracted from CONN visualization in Fig 5. 17 geologically meaningful clusters are color coded according to color wedge. These correspond to or refine previously identified spectral classes as explained in the text. Shaded regions are shown in various grey color codes (not included in the color wedge).

The spectrum of g is most similar to those of sand grains (C, Q) but inherently brighter, which might be an indication of finer grains and more light scattering such as caused by fine sand stirred up in the air. The image was collected at a time of extensive dust devil activity over the far plains so one might surmise that (in spite of the poor spatial resolution that does not allow visual verification) that the unit g might be a dust devil.

In summary, through interactive CONNvis clustering, we find the geological units identified in previous analyses and segment some units into subtypes which may be geologically meaningful. Our analysis provides a comprehensive mapping of the scene with clusters that reliably match the prototypical surface materials described by geologists. Relatively large areas, identified through our clustering as belonging to the same spectral type, have average signatures very close to the spectra of smaller type locations that were hand selected and extensively studied by geologists. By reliably showing the spatial occurrences of those materials verified from the type locations, our clusters yield trustworthy statistics for the whole scene. The reliable mapping also means potential for autonomous data analysis. This gives us confidence that CONNvis clustering can be a viable candidate for detailed information extraction and scientific discoveries.

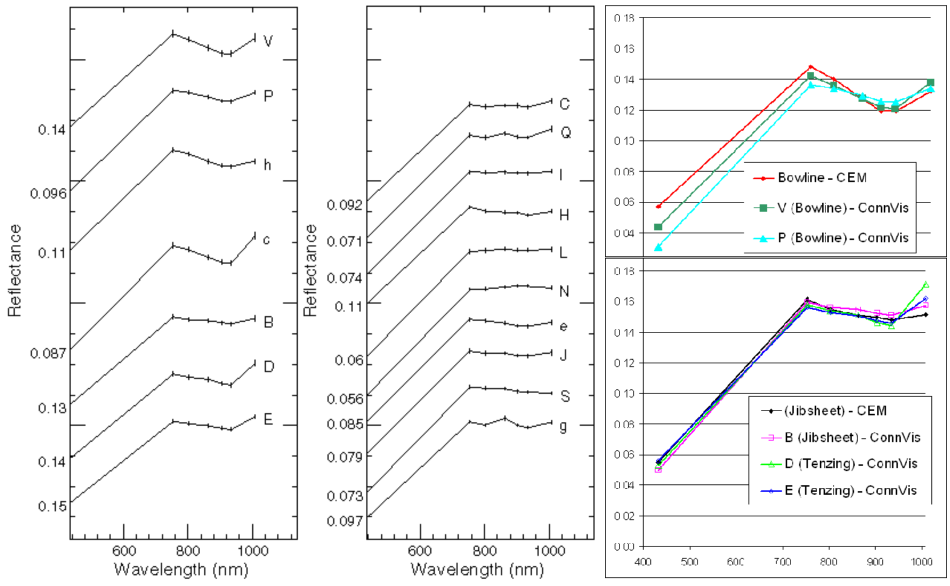


Fig. 7. Left and Center: Mean spectra of the 17 clusters in Fig. 6. The spectra of the subclusters grouped by their major types: Bowline (V, P, h), Jibsheet and Tenzing (B, D, E), soil and soil mixtures (I, H, L, N, e, J, S) and hypothetical dust devil (g). The subclusters have slight but consistent differences whereas different major types have greater spectral dissimilarities. Right: Comparison of subclusters to the corresponding superclasses in [21]. Top: Bowline subclusters. Bottom: Jibsheet and Tenzing.

4 Conclusions and Future Work

We show that CONNvis [16], a graph based visualization of data topology on the SOM lattice, successfully aids capture of accurate and fine details in remote sensing spectral imagery. This graph based visualization can also be useful for knowledge discovery from many other types of data including but not limited to genetic microarray data, genome profiles or other biological data, medical data, and financial data. Given that the parameters of CONNvis are derived automatically from the data characteristics, one might be able to apply CONN for automated cluster extraction from the SOM, with the same detail as with interactive clustering methods. This would be a significant achievement for structure discovery since current automated SOM clustering schemes produce results inferior to results from interactive procedures. This may open further possibilities for autonomous on-board science applications.

References

1. Ultsch, A.: Self-organizing neural networks for visualization and classification. In: Lausen, O.B., Klar, R. (eds.) Information and Classification-Concepts, Methods and Applications, pp. 307–313. Springer, Berlin (1993)

2. Kraaijveld, M.A., Mao, J., Jain, A.K.: A nonlinear projection method based on Kohonen's topology preserving maps. IEEE Trans. on Neural Networks 6(3), 548–559 (1995)
3. Ultsch, A.: Maps for the visualization of high-dimensional data spaces. In: Proc. 4th Workshop on Self-Organizing Maps (WSOM 2003), vol. 3, pp. 225–230 (2003)
4. Merényi, E., Jain, A.: Forbidden magnification? II.. In: Proc. 12th European Symposium on Artificial Neural Networks (ESANN 2004), Bruges, Belgium, D-Facto, April 28-30, 2004, pp. 57–62 (2004)
5. Cottrell, M., de Bodt, E.: A Kohonen map representation to avoid misleading interpretations. In: Proc. 4th European Symposium on Artificial Neural Networks (ESANN 1996), Bruges, Belgium, D-Facto, pp. 103–110 (1996)
6. Hakkinen, E., Koikkalainen, P.: The neural data analysis environment. In: Proc. 1st Workshop on Self-Organizing Maps (WSOM 1997), Espoo, Finland, June 4-6, 1997, pp. 69–74 (1997)
7. Merkl, D., Rauber, A.: Alternative ways for cluster visualization in Self-Organizing Maps. In: Proc. 1st Workshop on Self-Organizing Maps (WSOM 2005), June 4-6, pp. 106–111. Helsinki University of Technology, Neural Networks Research Centre, Espoo (1997)
8. Su, M.-C., Chang, H.-T.: A new model of self-organizing neural networks and its applications. IEEE Transactions on Neural Networks 12(1), 153–158 (2001)
9. Yin, H.: ViSOM- A novel Method for Multivariate Data Projection and Structure Visualization. IEEE Transactions on Neural Networks 13(1), 237–243 (2002)
10. Villmann, T., Merényi, E.: Extensions and modifications of the Kohonen SOM and applications in remote sensing image analysis. In: Seiffert, U., Jain, L.C. (eds.) Self-Organizing Maps: Recent Advances and Applications, pp. 121–145. Springer, Heidelberg (2001)
11. Himberg, J.: A SOM based cluster visualization and its application for false colouring. In: Proc. IEEE-INNS-ENNS International Joint Conf. on Neural Networks, Como, Italy, vol. 3, pp. 587–592 (2000)
12. Kaski, S., Kohonen, T., Venna, J.: Tips for SOM processing and colourcoding of maps. In: Deboeck, T.K.G. (ed.) Visual Explorations in Finance Using Self-Organizing Maps, London (1998)
13. Kaski, S., Venna, J., Kohonen, T.: Coloring that reveals cluster structures in multivariate data. Australian Journal of Intelligent Information Processing Systems 6, 82–88 (2000)
14. Vesanto, J.: SOM-based data visualization methods. Intelligent Data Analysis 3(2), 111–126 (1999)
15. Taşdemir, K., Merényi, E.: Exploiting data topology in visualization and clustering of Self-Organizing Maps. IEEE Transactions on Neural Networks (submitted, 2007)
16. Taşdemir, K., Merényi, E.: Data topology visualization for the Self-Organizing Maps. In: Proc. 14th European Symposium on Artificial Neural Networks (ESANN 2006), Bruges, Belgium, D-Facto, April 26-28, 2006, pp. 277–282 (2006)
17. Martinetz, T., Schulten, K.: Topology representing networks. Neural Networks 7(3), 507–522 (1993)
18. Ultsch, A.: Clustering with som: U*c. In: Proc. 5th Workshop on Self-Organizing Maps (WSOM 2005), Paris, France, September 5-8, 2005, pp. 75–82 (2005)
19. Squyres, S.W., Arvidson, R.E., Blaney, D.L., Clark, B.C., Crumpler, L., Farrand, W.H., Gorevan, S., Herkenhoff, K.E., Hurowitz, J., Kusack, A., McSween, H.Y., Ming, D.W., Morris, R.V., Ruff, S.W., Wang, A., Yen, A.: The rocks of the columbia hills. Journal of Geophys. Res.: Planets 111, E02S11, 10.1029/2005JE002562

20. Farrand, W.H., Bell III, J.F., Johnson, J.R., Squyres, S.W., Soderblom, J., Ming, D.W.: Spectral variability among rocks in visible and near infrared multispectral pancam data collected at gusev crater: Examinations using spectral mixture analysis and related techniques. Journal of Geophys. Res.: Planets 111, E02S15, 10.1029/2005JE002495
21. Farrand, W.H., Bell III, J.F., Johnson, J.R., Blaney, D.L.: Multispectral reflectance of rocks in the columbia hills examined by the mars exploration rover spirit: Cumberland ridge to home plate. Lunar and Planeary. Science XXXVIII (1957)
22. Ruff, S.W., Christensen, P.R., Blaney, D.L., Farrand, W.H., Johnson, J.R., Moersch, J.E., Wright, S.P., Squyres, S.W.: The rocks of guser crater as viewed by the mini-tes instrument. Journal of Geophys. Res.: Planets 111, E12S18, 10,1029/2006JE002747
23. Herkenhoff, K.E., Squyres, S., Arvidson, R.: The Athena Science Team, Overview of recent athena microscopic imager results. In: Lunar and Planetary Science XXXVIII, abstract 1421 (2007)

Mining Unordered Distance-Constrained Embedded Subtrees

Fedja Hadzic, Henry Tan, and Tharam Dillon

DEBII, Curtin University of Technology, Perth, Australia
{f.hadzic,h.tan,t.dillon}@curtin.edu.au

Abstract. Frequent subtree mining is an important problem in the area of association rule mining from semi-structured or tree structured documents, often found in many commercial, web and scientific domains. This paper presents the u3Razor algorithm, for mining unordered embedded subtrees where the distance of nodes relative to the root of the subtree needs to be considered. Mining distance-constrained unordered embedded subtrees will have important applications in web information systems, conceptual model analysis and more sophisticated knowledge matching. An encoding strategy is presented to efficiently enumerate candidate unordered embedded subtrees taking the distance of nodes relative to the root of the subtree into account. Both synthetic and real-world datasets were used for experimental evaluation and discussion.

1 Introduction

To express more complex and meaningful relationships between the data objects, many organizations represent their domain knowledge using semi-structured documents. Semi-structured documents such as XML possess a hierarchical document structure, where an element may contain further embedded elements, and each element can be attached with a number of attributes. It is therefore frequently modeled using a rooted ordered labeled tree. To support effective and efficient data analysis from tree structured documents algorithms have been developed to extract all subtree patterns that occur in the database of ordered labeled trees as many times as the user supplied support threshold. This is known as the frequent subtree mining (FSM) problem and is the first and most important and complex problem to consider when discovering useful associations among data objects from a tree structured document [1,2,3]. In many biological data analysis tasks, the aim is to find frequent structured patterns, such as frequent protein or chemical compound structures from the data. For example, the work presented in [4] demonstrates the potential of the tree mining algorithms for discovering substructures from Protein data that could be useful for discovering interesting similarities and differences in protein datasets taken across protein families and species. Tree mining has also been successfully applied in [5] for the analysis of phylogenetic databases. Driven by different application needs many algorithms have been developed that can mine different subtree types. HybridTreeMiner [6], uFreqt [7], and uNot[8], mine induced unordered trees. Treeminer [9] is an efficient algorithm for discovering all frequent embedded subtrees in a forest using a data structure called the vertical scope-list. The SLEUTH [10] algorithm extracts all frequent unordered embedded subtrees by using unordered scope-list joins via the descendant and cousin tests. Our contribution to the area of frequent subtree mining is the introduction of a tree model guided (TMG) candidate subtree enumeration framework which was used for developing efficient algorithms for mining of ordered induced/embedded [2], ordered distance-constrained embedded subtrees [11] unordered induced [12] and unordered embedded [13] subtrees. TMG ensures that only those subtrees are enumerated which conform to the underlying tree structure of the document [3]. For a more extensive overview of the state-of-the-art of tree mining please refer to [14].

In an ordered subtree the left-to right order among the sibling nodes needs to be preserved while in an unordered subtree the order of the sibling nodes (and the subtrees rooted at those nodes) can be exchanged and the resulting subtree is still considered the same. This causes the enumeration and counting of unordered subtrees more difficult, since each enumerated subtree needs to be ordered into one logical and consistent form, so that all its variants that have different order among sibling nodes are considered as the same subtree. An induced subtree preserves the parent-child relationships from the original tree while in an embedded subtree the parent-child relationship are allowed to be ancestor-descendant relationships in the original tree. By mining embedded subtrees one can detect commonly occurring relationships between data objects in spite of the difference in the level where the relationship in the document occurred. Certain concepts may be represented in a more specific/general way in certain documents. This specific information is often in the form of additional child nodes of the concept, and hence, two general and related concepts may be separated by a number of levels in the document tree. If the user is only interested in the relationship between these two general concepts, such a relationship could be directly found in an embedded subtree set, while if induced subtrees were extracted, the information irrelevant to the user may be present in the patterns of interest.

While mining of embedded subtrees is a generalization over induced subtrees, one limitation is that the context information may be lost in some patterns. For example, when analyzing a biomedical database containing patient records of potential illness causing factors, one would be interested in common set of data object properties that have frequently been associated with a particular disease. By allowing ancestor-descendant relationships it may be possible to loose some information about the context in which particular disease characteristic occurred. This is mainly due to the fact that some attributes of the dataset may have a similar set of values and hence indicating which value belonged to which particular attribute is necessary. There appears to be a trade-off here and in this case allowing ancestor-descendant relationships can result in unnecessary and misleading information, but in other cases, it proves useful as it detects common patterns in spite of the difference in granularity of the information presented. A difficulty with embedded subtrees appears then to be that there is too much freedom allowed with respect to the difference in the distances between the nodes. All occurrence of one particular relationship are considered the same and valid even if the distance between the related data objects is so different that it is possible that it occurred in a different context and has a different intended meaning. One way to avoid this characteristic is to further distinguish the embedded subtrees based upon the distance between the nodes. Making this distinction will have important applications in web information systems, conceptual model analysis, knowledge matching and for general knowledge management tasks by allowing for more specialized queries.

In this study we extend our past work by developing the first algorithm that will extract all unordered embedded subtrees with node distance information. It adds more granularity to the problem as the occurrences of the embedded subtrees with different distances among the nodes are now considered as different candidates (hence the name distance-constrained). Overall, mining of unordered distance-constrained subtrees is more expensive in terms of space and time required than when mining any of other subtree types. In Section 2 we define some tree mining related concepts and provide a motivating example for mining of unordered distance-constrained embedded subtrees. The proposed u3Razor algorithm is described in Section 3 together with a suitable encoding strategy for enumerating unordered embedded subtrees that take the node distance information from the database tree into account. Section 4 presents some experiments to test the scalability of the approach and compare with the results obtained by not imposing the distance constraint. Section 5 concludes the paper.

2 Problem Definition and Motivation

A tree T is an acyclic connected graph with the node at the top defined as the *root[T]*. The *Paren*t of node v (*parent[v]*) is defined as its predecessor. Two nodes that share the same parent are referred to as *sibling nodes*. The fan-out or degree of a node corresponds to the number of children of that node. A *leaf node* is a node without a child; otherwise, it is an internal node. A path from vertex v_i to v_j, is defined as the finite sequence of edges that connects v_i to v_j. The length of a path p is the number of edges in p. The distance between two nodes v_i and v_j can then be defined as the length of the path connecting v_i and v_j. If p is an ancestor of q and q is a descendant of p, then there exists a path from p to q. The *rightmost path* (RMP) of T is defined as the (shortest) path connecting the rightmost leaf with the root node. The *Depth/level* of a node is the length of the path from root to that node. The *size* of a tree equals to the total number of nodes in the tree. In this paper, the term 'k-subtree' refers to a subtree that consists of k number of nodes. A tree can be denoted as $T(V,L,E)$, where (1) V is the set of vertices or nodes; (2) L is the set of labels of vertices, for any vertex $v \in V$, $L(v)$ is the label of v; and (4) $E = \{(x,y) | x,y \in V\}$ is the set of edges in the tree. The problem of **frequent subtree mining** can be generally stated as: Given a tree database T_{db} and minimum support (σ) extract all candidate subtrees that occur at least σ times in T_{db}.

Within the tree mining framework, two support definitions often used are transaction-based and occurrence match support. When using the ***transaction-based support*** definition, the transactional support of a subtree t, denoted as $\sigma_{tr}(t)$ in a tree database T_{db} is equal to the number of transactions in T_{db} that contain at least one occurrence of subtree t. Let the notation $t \prec k$, denote the support of subtree t by transaction k, then $t \prec k = 1$ whenever k contains at least one occurrence of t, and 0 otherwise. Given N transactions k_1 to k_N of tree in Tdb, the $\sigma_{tr}(t)$ in T_{db} is defined as $\sum_{i=1}^{N} t \prec k_i$.

The ***occurrence-match support*** takes the repetition of items in a transaction into account and counts the subtree occurrences in the database as a whole. Hence, the occurrence-match support of a subtree *t*, denoted as $\sigma_{oc}(t)$, in a tree database T_{db} is equal to the total number of occurrences of *t* in all transactions in T_{db}. Let function *g(t,k)* denote the total number of occurrences of subtree *t* in transaction *k*. If there are *N* transactions k_1 to k_N of tree in T_{db}, $\sigma_{oc}(t)$ in T_{db} can be defined as $\sum_{i=1}^{N} g(t, k_i)$.

Next, we provide some formal definitions of commonly mined subtree types.

Given a tree $S = (V_S,L_S,E_S)$ and tree $T = (V_T,L_T,E_T)$, *S* is an ***induced subtree*** of *T,* iff (1) $V_S \subseteq V_T$;(2) $L_S \subseteq L_T$, *and* $L_S(v)=L_T(v)$; and (3) $E_S \subseteq E_T$.

Given a tree $S = (V_S,L_S,E_S)$ and tree $T = (V_T,L_T,E_T)$, *S* is an ***embedded subtree*** of *T*, iff (1) $V_S \subseteq V_T$;(2) $L_S \subseteq L_T$, *and* $L_S(v)=L_T(v)$; (3) *if* $(v_1,v_2) \in E_S$ then $parent(v_2) = v_1$ in *S* and v_1 is ancestor of v_2 *in T.* Hence, the main difference between an induced and an embedded subtree is that, while an induced subtree keeps the parent-child relationships from the original tree, an embedded subtree allows a parent in the subtree to be an ancestor in the original tree. All the definitions provided above do not take into account the order among the sibling nodes. This is what makes them ***unordered*** subtrees. In an ***ordered*** subtree the left to right ordering of sibling nodes in the original tree is preserved. As mentioned in the introduction, when a distance constraint is imposed on an embedded subtree, the distance information between the nodes in the original subtree needs to be stored and used as an additional candidate grouping criterion.

Given a tree $S = (V_S,L_S,E_S)$ and tree $T = (V_T,L_T,E_T)$, *S* is an ***unordered distance-constrained embedded subtree*** of *T* iff (1) $V_S \subseteq V_T$;(2) $L_S \subseteq L_T$, and $L_S(v)=L_T(v)$; (3) if $(v_1,v_2) \in E_S$ then $parent(v_2) = v_1$ in *S* and v_1 is ancestor of v_2 in T; and (4) $\forall\ v \in V_S$ there is an integer stored indicating the distance between *v* and the root node of *S* in the original tree *T*.

For an ordered distance-constrained embedded subtree, in addition to the above pre-conditions the left-to-right ordering among the sibling nodes in the original tree would also need to be preserved [11]. To illustrate the difference in mining of different subtree types please consider the example tree in Fig. 1 where the label of each node is shown with its pre-order position on the left. In this paper, the term 'occurrence coordinate(s) *(oc)*' will be used to refer to the position(s) of a particular node or a subtree in the tree database. In the case of a node, *oc* corresponds to the pre-order position of that node in the tree database, whereas for a subtree, *oc* is a sequence of *ocs* from nodes that belong to that particular subtree. If ordered or unordered induced subtrees are mined *st* occurs only once in *T* with *oc*:01569, while for ordered embedded subtrees it also occurs at *oc*:01589 since the ancestor-descendant relationships between nodes c (*oc*:5) and d (*oc*:8) are allowed. With unordered embedded subtrees the order can also be exchanged and hence *st* also occurs at *oc*:01587. If unordered distance constrained subtrees are mined each of the three occurrences of *st* will be considered as a separate subtree depending on the distance of the nodes to the root of the subtree as detected in *T* (i.e. st_1, st_2 and st_3 in Fig.1.). The numbers next to the link indicates the distance between the nodes connected by that link in *T*. Hence, st_1 is a representative of the subtree with *oc*:01569, st_2 of *oc*:01589 and st_3 of *oc*:01587.

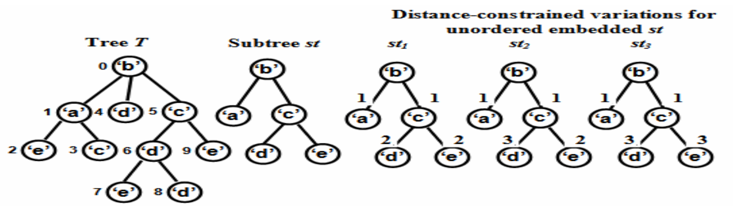


Fig. 1. Example tree *T* and subtree *st* with distance-constrained variants

For unordered subtrees the enumeration and counting phase is more difficult than for the ordered case, since each enumerated subtree needs to be ordered into one logical and consistent form, so that all its variants that have different order among sibling nodes are considered as the same subtree. The group of possible trees obtained by permuting the sibling nodes in all possible ways is referred to as the automorphism group of a tree [10]. During the pre-order traversal of a database, ordered subtrees are generated by default. It is necessary to identify which of these ordered subtrees form an automorphism group of an unordered subtree. One tree needs to be selected to uniquely represent the unordered tree. This selected tree is known as the canonical form (CF) of an unordered tree. A canonical form (CF) of an entity is in general a representative form (or a function) for which many equivalent variations of an entity can be represented (mapped) into one standard, conventional, logical form in a consistent manner [15]. The CF used by the proposed algorithm will be explained in Section 3.

To conclude this section, we provide an example that illustrates a case where adding the distance constraint is important for effective data analysis with respect to the application needs. In Fig. 2, two example trees are displayed that indicate a part of the ancestor family tree from two ill patients (the examples come from an image of a disease family tree obtained from [16]. Such information is used for linkage analysis of an illness by performing gene testing which can provide information about one having a disease-related gene mutation. When looking for a disease gene, scientists often start by studying DNA samples from family members over several generations who have a number of relatives who have developed an illness [16].

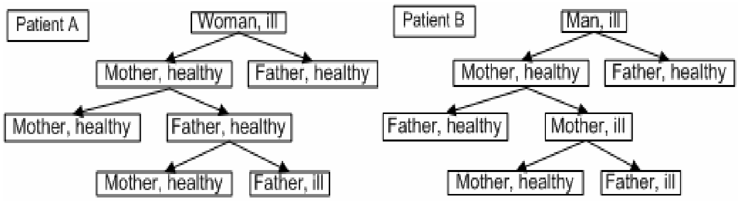


Fig. 2. Example ancestor representation of two ill patients (A and B)

For example, a scientist may want to discover how many ill relatives an ill patient has had and to discover the number of generations that separates them. Using the traditional embedded subtree definition, we can extract information only about the number of ill relatives, but cannot have the information about the number of generations that separate the patient and the relatives that have a common disease. This is because the traditional embedded subtree definition does not have this kind of expressive capability. In contrast, by utilizing the distance-constrained embedded subtrees, we can find out exactly how many generations they are separated by, by inspecting the distance information stored between the nodes. From Fig. 2, patient A has only one diseased ancestor and it is her great-grandfather, while patient B has two diseased ancestors, a grandmother and great-grandfather. Even though we do not have such an example in the figure, it is worth noting that it could well be the case that an ill patient will have two ancestors of the same gender that have the illness. In this case, the traditional embedded subtree definition would group these subtree occurrences as one candidate and indicate wrongly that there is only one ancestor with a disease. On the other hand, by mining distance-constrained embedded subtrees, both occurrences will be considered as separate entities due to the difference in the distance to the root node which is used as an additional candidate grouping criterion. Generally speaking, an algorithm for mining of unordered distance-constrained embedded subtrees will have some important applications in analysis of biological sequences, web information systems and conceptual model analysis.

3 u3Razor Algorithm

The steps taken by the u3Razor algorithm are presented in Fig. 3. The tree database is first transformed into a database of rooted integer-labelled trees as hashing integer-labelled trees is much faster than hashing of string-labelled trees. It is then ordered into its canonical form (CF) to reduce the average number of candidate trees that need to be ordered. Recursive List (*RL*) is constructed which is a global sequence of encountered nodes in the pre-order traversal together with the necessary node information. During this process the node labels are hashed to obtain the set of frequent 1-subtrees (*F1*). TMG candidate generation using the *RL* structure takes place and the string representatives of candidate subtrees with the distance information between the nodes are hashed to the Ck hash table and their occurrences are stored. Prior to hashing the string representation of each candidate subtree, it is first ordered into its CF, if necessary. The process repeats until all frequent k-subtrees are enumerated. To enable the mining of unordered distance constrained embedded subtrees the major change to our general TMG framework [2, 3, 12, 13] took place in the way that candidate subtrees are represented at the implementation level to take into account the distance information and the CF used, which is explained next. We then explain the *RL* structure and the TMG process for enumerating a complete set of unordered distance-constrained embedded subtrees.

Tree Representation and CF Ordering. Our work utilizes the pre-order string encoding (φ) as described in [2,9], which is a sequential representation of the nodes of a tree as encountered during the pre-order traversal of that tree. The backtrack symbol ('/') is used whenever moving up a node in the tree during the pre-order traversal. To take the distance between the nodes into account, the encoding of a subtree is obtained by reading the nodes in the pre-order traversal and for each node storing the

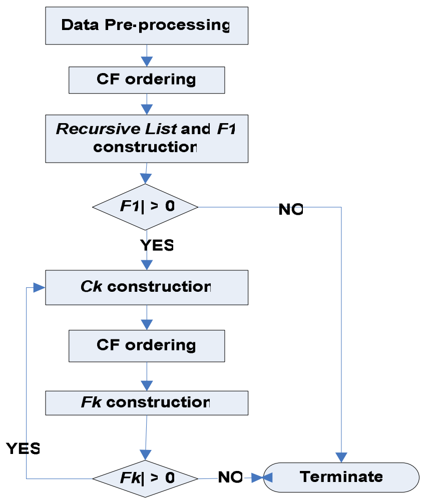


Fig. 3. General description of the steps taken in the proposed approach

distance to the root of the subtree (node depth). The distance to the root is worked out from the node levels stored in the *RL* structure, where the root of the subtree is assigned the depth of 0 and all other nodes are assigned the difference between their level and the original level of the new subtree root. Further modification of the encoding consists in storing a number next to each backtrack '/' symbol indicating the number of backtracks in the subtree, as opposed to storing each of those backtracks as a separate symbol. This representation allows easier string manipulation due to uniform block size. We denote encoding of a subtree *T* as *φ(T)* and eg. from Fig. 1, *φ(T):* 'b0 a1 e2 /1 c2 /2 d1 /1 c1 d2 e3 /1 d3 /2 e2 /2', *φ(st1):* 'b0 a1 /1 c1 d2 /1 e2 /2', and *φ(st2)*: ''b0 a1 /1 c1 d3 /1 e2 /2'. The backtrack symbols can be omitted after the last node eg. *φ(st3)*: ''b0 a1 /1 c1 d3 /1 e3'.

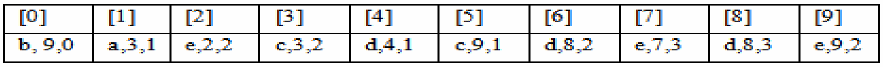
The canonical form ordering occurs at the start where the whole tree database is ordered into its canonical form and later where candidate subtrees are ordered so that unordered subtrees are correctly enumerated. The canonical form according to which we order the trees uses the idea of the DFCF [1] where the nodes are sorted at each level of the subtree in a bottom up fashion (i.e. starting from the leaf nodes), and the nodes with labels that sort lexicographically smaller are placed to the left of the subtree. The ordering process is determined by the means used for comparing nodes or subtrees so that they are placed at the right position in the tree. At the implementation level the process can be formally explained as: given two trees T1 and T2, with root[T1] = r1 and root[T2] = r2, let C(r1) and C(r2) denote the children sets of r1 and r2, respectively. Further, let $\varphi(Tx)_k$ denote the kth element of the pre-order string encoding of tree Tx (x = 1 or 2)(this can be either a node label or the special backtrack ('/') symbol which is considered smaller than any other label). In case the node labels are the same the distance information associated with each node will be considered so that the nodes with smaller distances to the root of the tree are placed to the left. T1 is considered smaller than T2 iff either:

a.) L(r1) < L(r2), or

b.) L(r1) = L(r2) and either size(C(r1)) < size(C(r2)) and $\varphi(T1)_k = \varphi(T2)_k$ for all $1 \le k \le$ length(φ(T1)), or $\varphi(T1)_k < \varphi(T2)_k$ for some $1 \le k <$ length(φ(T1))

This ordering scheme will ensure that all the instances of unordered distance-constrained embedded subtrees are correctly represented and counted.

Recursive List (RL) and F1 Construction. The tree database, $T_{db,}$ is scanned once to create the global sequence RL in memory, through which nodes' related information can be directly accessed. Each node is stored following the pre-order traversal of the T_{db}. *Position*, *label*, *scope*, and *level* information are stored for each node. The scope of a node refers to the position of its rightmost leaf node or its own position if it is a leaf node itself [2,9] whereas the level refers here to the level in the T_{db} tree, where this node occurs. An item in RL at position *i* is referred to as *RL[i]*. Every time a node is inserted into the *RL*, we generate a candidate 1-subtree. Based on its label, we increment its support count in the C_1 hash table. If its support count is $\ge \sigma$ (user-specified minimum support count), we insert the candidate 1-subtree to the frequent 1-subtree set, F_1. An example RL structure representing the tree *T* from Fig. 1 is displayed in Fig. 4. The pre-order position of a node in the tree database is equal to the index of the RL at which that nodes is stored, and the label, scope and level are shown in that order underneath the entry. All this information is necessary to enumerate only valid subtree candidates and is accessed in the TMG candidate enumeration process explained next.



[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
b, 9,0	a,3,1	e,2,2	c,3,2	d,4,1	c,9,1	d,8,2	e,7,3	d,8,3	e,9,2

Fig. 4. Recursive List representation of tree *T* (Fig. 1)

A particular subtree, as defined by its encoding can be found at many places in the database and these different occurrences need to be stored so that subsequent set of candidates can be generated. We only store the occurrence coordinates of the nodes in the right-most path of the subtree (referred to as *RMP-oc*). Within our framework, this information is sufficient for enumerating candidate *(k+1)*-subtrees from a frequent *k*-subtree. Given a k-subtree *T* with *oc* $[e_0,e_1,\ldots e_{k-1}]$, the *RMP-oc* of *T*, denoted by $\Psi(T)$, is defined by $[e_0,e_1,\ldots,e_j]$ such that $\Psi(T) \subseteq oc(T)$; $e_j = e_{k-1}$; and $j \le k$-1 and the path from e_j to e_0 is the *RMP* of tree *T*. Vertical Occurrence List (*VOL*) is used to store all $\Psi(T)$ of a subtree *T* represented by its pre-order string encoding $\varphi(T)$, and to determine the occurrence-match and transaction-based support. A transaction identifier (*tid*) is stored for each $\Psi(T)$ so that the occurrence match of *T* equals to |*VOL*| while the transaction-based support equals to the number of unique *tid*s in *VOL.*

TMG Candidate Subtree Generation. TMG is a specialization of the right most path extension method which has been reported to be complete and non-redundant [2,9]. To enumerate all embedded k-subtrees from a (*k-1*)-subtree, the TMG enumeration approach extends all the nodes in the *RMP* of a (*k-1*)-subtree, by one node at a time. Hence, it is a breadth-first (BF) enumeration strategy. Suppose that nodes in the

RMP of a subtree are defined as *extension points*. The TMG can be formulated as follows. Let $\Psi(T_{k-1})$:$[e_0,e_1,\ldots e_j]$ denote the *RMP-oc* of a frequent (k-1)-subtree T_{k-1}, and Φ the *scope* of the root node e_0. TMG generates k-subtrees by extending each extension point $n \in \Psi(T_{k-1})$ with a node with *oc t* iff $n < t \leq \Phi$. Suppose that the encoding of T_{k-1} is denoted by $\varphi(T_{k-1})$ and $l(e_j,t)$ is a labeling function for extending extension point *n (i.e.* e_j) with a node at position *t*. $\varphi(T_k)$ would be defined as $\varphi(T_{k-1})+l(e_j,t)$, where $l(e_j,t)$ determines the number of backtrack symbols '/' to be appended before the label of the new node is added to $\varphi(T_k)$. The number of backtrack symbols is calculated as the shortest path length between the extension point *n* and the right-most-node *r*, (notation *pl(n,r)*). To generate RMP at each step of candidate generation, we utilize the computed number of backtrack symbols *b* that need to be appended before the new node with *oc t* is added to the encoding. Given that the $\Psi(T_{k-1})$ is $[e_0,e_1,\ldots,e_j]$, the RMP of the k-subtree ($\Psi(T_k)$) is generated by appending *t* at position $(j+1) - b$ of the ($\Psi(T_{k-1})$) and removing any *RMP-oc* that occur after *t*, thereby making *t* the right most node of T_k. This will make sure that at each extension of (k-1)-subtree, *RMP-oc* of k-subtree are appropriately stored.

To provide an illustrative example let us say that we are extending the T_{k-1} subtree from Fig. 5, with $\Psi(T_{k-1})$:$[0,4,5]$ (*oc*:0145) and $\varphi(T_{k-1})$:'a0 b1 /1 b1 c2'. The label, level and scope information is obtained from the *RL* entries corresponding to the *oc* of a node as is shown on the right of figure. For example at *RL*[10], we would have the label 'c', scope 10 and level 2. If extending T_{k-1} from extension point node 'b' (*oc*:4) with node 'e' (*oc*:8) then $l(5,8)$ will append one backtrack symbol ($pl(4,5) = 1$) and the label 'e' to $\varphi(T_{k-1})$ together with the distance to the root node obtained from *RL* (i.e. level of node 'e' - level of root node 'a', i.e. 3 – 0 = 3) . The new encoding $\varphi(T_k)$ becomes 'a0 b1 /1 b1 c2 /1 e3', and $\Psi(T_k)$:$[0,4,8]$ (i.e. inserting 8 in entry (j+1) – b = (3+1) – 1 = 3 of $\Psi(T_{k-1})$).

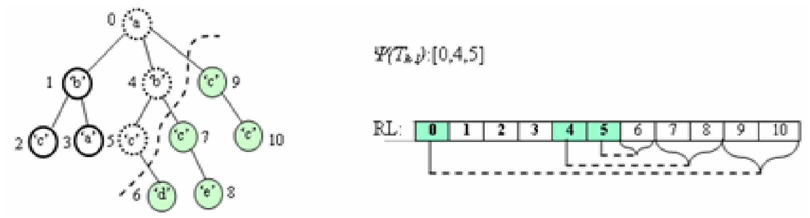


Fig. 5. TMG enumeration: extending (k-1)-subtree T_{k-1} (where $\varphi(T_{k-1})$:'a b / b c' occurs at position (0,1,4,5)) with nodes at positions 6, 7, 8, 9, and 10

In the case of unordered subtrees, the right-most-node may not always correspond to the last node (tail position) in the encoding as it does for the ordered subtree case. We refer to this case as *non-tail expansion*. A notion of pivot position ς is used to denote the position in the subtree encoding that corresponds to the right-most-node. Each RMP-OC of a subtree will store an integer indicating the pivot position ς in the encoding for that particular occurrence of the subtree. Hence, for a *non-tail expansion* of a subtree T_{k-1}, if we are appending a new node with label *l* and OC *t*, rather than appending the backtrack symbols (if any) and *l* to the last node in $\varphi(T_{k-1})$, it will be appended to the pivot position ς by the function $l(\varsigma, t)$, in order to obtain $\varphi(T_k)$. Please note that if there are b backtrack symbols to be appended with l and there were already some backtrack symbols after the pivot position ς in $\varphi(T_{k-1})$, then l will be appended after the bth backtrack symbol. Furthermore, an additional backtrack symbol will be appended after the position in the encoding where l has been appended. To illustrate this please consider the subtree *st3* from tree *T* in Fig. 1, with *oc*:01587, $\Psi(st3):[0,5,8]$ *and* $\varphi(st3)$:'b0 a1 /1 c1 d3 /1 e3' As can be seen the right-most node does not correspond to the last node 'e' in the encoding with oc:7, but rather to node 'd' with *oc*:8. Therefore, if we are extending *st3* from extension point node 'c' (*oc*:5) with node 'e' (*oc*:9) then *l(5,9)* will append the label 'e' to $\varphi(st3)$ at pivot position ς and add '/1' after 'd' (from *pl(5,8)*. The new encoding becomes 'b0 a1 /1 c1 d3 /1 e2 /1 e3'.]. Whenever a new candidate k-subtree is generated, full (k-1) pruning [2,3,9] is performed where a k-subtree is pruned if at least one of its (k-1)-subtree is infrequent. The whole process of TMG candidate enumeration is repeated until all frequent k-subtrees are enumerated.

4 Experimental Results

The experiments were run on Intel Xeon E5345 at 2.33 GHz with 8 cores, 8 GB RAM and 4MB Cashe Open SUSE 10.2. The purpose of the first experiment is to test the scalability of the proposed u3Razor algorithm with respect to the increasing number of transactions present in a database. An artificial database was created, where the size of the transactions for each test was varied from 100,000, 500,000 to 1 million with minimum support 50, 250, and 500, respectively. Occurrence match support definition was used and the result displayed in Fig. 6, shows that the time to complete the task approximately scales linearly with the increase in transaction size.

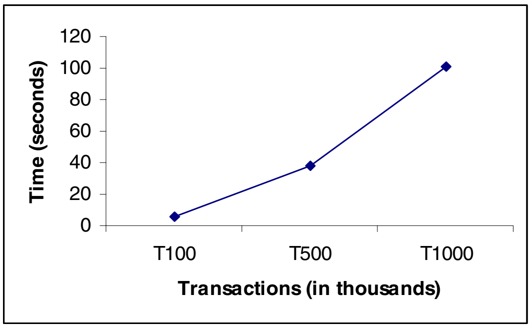


Fig. 6. Scalability – time performance / number of transactions

The second experiment was performed to examine compare the number of frequent subtrees detected between the proposed algorithm and the UNI3 [12] and U3 [13] algorithms for mining of unordered induced and embedded subtrees, respectively. Real world CSLogs data set [9] consisting of 32421 transactions was used, and the transactional support definition was used. The number of frequent subtrees detected by the u3Razor and UNI3 algorithms is equal for support thresholds of 1000, 800 and 600 (Figure 7(a)). At these supports, the U3 algorithm detects additional subtrees as frequent, which implies that those additional embedded subtrees occur with a different distance among the nodes in the original tree. Otherwise, a number of them would have been detected by the u3Razor algorithm, where the distances have to be the same. It should be noted here that there may be some embedded subtrees that occur with the same distance among the nodes but the number of occurrences of such subtrees is not sufficient to be considered as frequent by the u3Razor algorithm for the given support. Since there are no embedded levels among the nodes in induced subtrees (i.e. the distance between all the nodes is equal to 1), both u3Razor and UNI3 detect the same frequent subtrees in this scenario. For lower support thresholds, more subtrees will be considered as frequent. The difference between the number detected by u3Razor and UNI3 in Fig. 7(a) indicates that there are some sufficient occurrences of embedded subtrees where the distance among the nodes is different in the original tree. There are 4 such embedded subtrees for s400 and 39 for s200. For some applications it may be of interest to the user to analyze such patterns to reveal some specific dataset characteristics. If the difference in the level is caused by same information being stored differently then they can be considered as valid patterns, while if the difference is due to the items with same labels being used in different contexts then they should be considered invalid. They are valid with respect to the support threshold. Fig. 7(b) shows the time taken by the algorithms for completing this task. For most support thresholds, the u3Razor algorithm takes slightly longer and this is due to the fact that the level information needs to be stored for all the nodes of the enumerated subtrees. At s200, the U3 algorithm takes the longest which is explained by the additional 54 subtrees that it considers as frequent in comparison with u3Razor (see Fig. 7(a)).

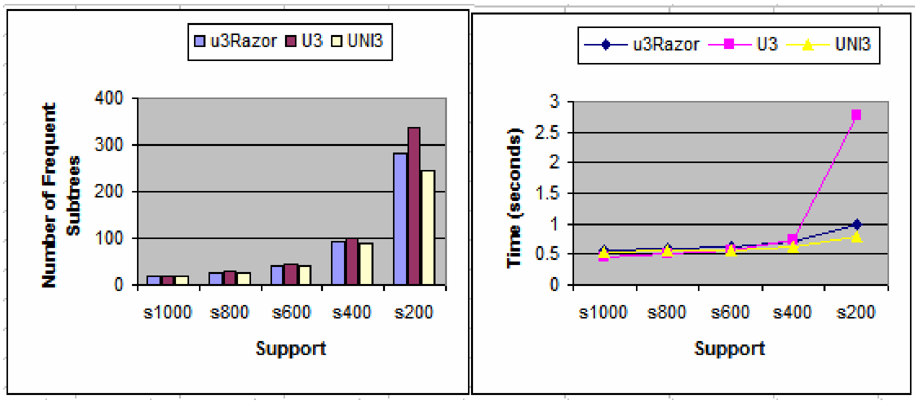


Fig. 7. (a) Number of frequent subtrees **(b)** time taken

5 Conclusions

In this paper we have discussed the motivation and some important applications for mining of unordered distance-constrained embedded subtrees. The first algorithm that solves this problem was presented as the extension to our general TMG candidate subtree enumeration framework. A number of experiments were performed using both synthetic and real-world datasets. The comparison of the results with the algorithms mining traditional subtree types, indicate the potential for more specific data analysis from tree-structured documents by considering the distance between the nodes.

References

1. Chi, Y., Yirong, Y., Muntz, R.R.: Canonical Forms for Labeled Trees and Their Applications in Frequent Subtree Mining, Knowledge and Information Systems (2004)
2. Tan, H., Dillon, T.S., Hadzic, F., Feng, L., Chang, E.: IMB3-Miner: Mining Induced/Embedded Subtrees by Constraining the Level of Embedding. In: Ng, W.K., Kitsuregawa, M., Li, J., Chang, K. (eds.) PAKDD 2006. LNCS (LNAI), vol. 3918, pp. 450–461. Springer, Heidelberg (2006)
3. Tan, H., Hadzic, F., Dillon, T.S., Feng, L., Chang, E.: Tree Model Guided Candidate Generation for Mining Frequent Subtrees from XML. ACM Transactions on Knowledge Discovery from Data (TKDD) (to appear, 2007)
4. Hadzic, F., Dillon, T.S., Sidhu, A., Chang, E., Tan, H.: Mining Substructures in Protein Data. In: IEEE ICDM DMB Workshop, Hong Kong, 18-22 December (2006)
5. Shasha, D., Wang, J.T.L., Zhang, S.: Unordered Tree Mining with Applications to Phylogeny. In: 20th International Conference on Data Engineering (2004)
6. Chi, Y., Yang, Y., Muntz, R.R.: HybridTreeMiner: An efficient algorihtm for mining frequent rooted trees and free trees using canonical forms. In: Proc. of the 16th Int'l Conf. on Scientific and Statistical Database Management, Santorini Island, Greece (2004)
7. Nijssen, S., Kok, J.N.: Efficient discovery of frequent unordered trees. In: Int'l Workshop on Mining Graphs, Trees, and Sequences (MGTS 2003), Dubrovnik, Croatia (2003)
8. Asai, T., Arimura, H., Uno, T., Nakano, S.: Discovering Frequent Substructures in Large Unordered Trees. In: 6th Int'l Conf. on Discovery Science (2003)
9. Zaki, M.J.: Efficiently Mining Frequent Trees in a Forest: Algorithms and Applications. IEEE Transactions on Knowledge and Data Engineering 17(8), 1021–1035 (2005)
10. Zaki, M.J.: Efficiently Mining Frequent Embedded Unordered Trees. Fundamenta Informaticae 65, pp. 1–20. IOS Press, Amsterdam (2005)
11. Tan, H., Dillon, T.S., Hadzic, F., Chang, E.: Razor: mining distance-constrained embedded subtrees. In: Workshop on Ontology Mining and Knowledge Discovery from Semistructured documents (MSD 2006), in conjunction with ICDM 2006, Hong Kong, December 28-22 (2006)
12. Hadzic, F., Tan, H., Dillon, T.S.: UNI3: efficient algorithm for mining unordered induced subtrees using TMG candidate generation. In: IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2007), Honolulu, Hawaii, April 1-5 (2007)
13. Hadzic, F., Tan, H., Dillon, T.S.: U3 – Mining Unordered Embedded Subtrees Using TMG Candidate Generation. In: ECML PKDD, Antwerp, Belgium, September 15-19 (submitted, 2008)
14. Tan, H., Hadzic, F., Dillon, T.S., Chang, E.: State of the art of data mining of tree structured information. International CSSE Journal 23(2) (March 2008)
15. Valentine, G.: Algorithms on Trees and Graphs. Springer, Berlin (2002)
16. Access Excellence @ the national health museum, Understanding gene testing: What does a predictive gene tell you, `http://www.accessexcellence.org/AE/AEPC/NIH/gene14.html`

Finding Frequent Patterns from Compressed Tree-Structured Data

Seiji Murakami, Koichiro Doi, and Akihiro Yamamoto

Graduate School of Informatics, Kyoto University
Yoshida-Honmachi, Sakyo-ku, Kyoto 606-8501, Japan
murakami@iip.ist.i.kyoto-u.ac.jp,
{doi,akihiro}@i.kyoto-u.ac.jp

Abstract. In this paper we present a new method for finding frequent patterns from tree-structured data, where a frequent pattern means a subgraph which frequently occurs in a given tree-structured data. We make use of a data compression method called TGCA for tree-structured data. Improving manipulation of large scaled data by compressing them has been investigated in previous studies, such as keyword search in plain texts, and frequent itemset mining from transaction data, but it has not been applied to finding frequent patterns from tree-structured data in the best of our knowledge. The TGCA algorithm is obtained by modifying the SEQUITUR algorithm for plain texts so that it can compress tree-structured data, and we show that we can count occurrences of patterns in the original data by using the data compressed by TGCA without expanding it. This is the reason why our method improves the efficiency of finding frequent patterns. The advantage of our method is shown in some experiments in the case that the data can be compressed in some good compression ratios.

1 Introduction

A semi-structured document is a text document which is given hierarchical structures with tags and is represented in the form of a tree. We call such a document *tree-structured data*. Typical examples of tree-structured data are HTML/XML documents. The aim of this paper is to present a new method for finding frequent patterns from a tree-structured data, where a frequent pattern means a subgraph which frequently occurs in a given tree-structured data.

Improving manipulation of large scaled data by compressing them has been investigated in previous studies. Takeda et al.[12] achieved more efficient string retrieval from compressed text data than from uncompressed. Han et al.[5] made finding frequent itemsets more efficient by using a method which is similar to data compression. However, in the best of our knowledge, improving finding patterns with data compression from tree-structured data has not been investigated yet.

For XML documents, compression algorithms such as XMill[8], XGrind[14], and XPress[6] have been proposed for making query processing more friendly and so on. In this study, we use the TGCA algorithm[11] which is obtained by modifying the SEQUITUR algorithm[9] so that it can compress tree-structured data. TGCA enables us to count occurrences of patterns in the original data by using the data compressed by TGCA without expanding it.

Various algorithms have been proposed for finding frequent patterns efficiently from tree-structured data, e.g. [1][2][4][10][13][16], but each of them follows its own formalization of the tree-structured data mining problem. We follow the formalization for the FREQT algorithm by Asai et al.[1]. We also extend FREQT so that it may take as its input data compressed by TGCA. We show that our method is more efficient than the original FREQT, by complexity analysis of algorithm. We also made some experiments with practical data to see whether or not our method works well.

This paper is organized as follows: We introduce the definition of tree-structured data mining problem and the compression algorithm TGCA in Section 2. Section 3 presents our method of finding frequent patterns and its merits. Section 4 shows our experimental results and analysis, and in Section 5 we give our conclusion.

2 Preliminaries

2.1 Tree-Structured Data Mining Problem

According to the formalization by Asai et al.[1], we model tree-structured data and patterns over them with rooted ordered trees.

A rooted tree T is a tuple of $(V, \mathcal{L}, v_0, E, \preceq, label)$, where V and $\mathcal{L}$ are distinct finite sets, $E \subseteq V^2$, $\preceq$ is a partial order of elements in V, and *label* is a mapping $V \rightarrow \mathcal{L}$. Each element of V is called a *node* of T, and $v_0 \in V$ is called its *root*. Each element of E is called a *directed edge*. If $label(v) = l$, we say that the *label* of the node v is l. For each $v \in V$ except v_0, there is a unique v' s.t. $(v', v) \in E$ and we call v' the *parent* of v, and write $v' = \pi(v)$. We inductively define $\pi^n(v)$ as $\pi^0(v) = v$ and $\pi^n(v) = \pi(\pi^{n-1}(v))$. We let $\pi(v_0) = \lambda$, and $\pi(\lambda) = \lambda$ where λ stands for empty. The nodes $v_1, ..., v_n$ are *siblings* if $\pi(v_1) = \cdots = \pi(v_n)$, and we assume that $v_1 \preceq v_2 \preceq \cdots \preceq v_n$. We write $v_{i+1} = NextSibling(v_i)$, and let $\lambda = NextSibling(v_n)$. The leftmost child of v is denoted by $Child(v)$. If v has no child, then we write $Child(v) = \lambda$. The size of T, which is denoted by $|T|$, is the total number of the nodes in T.

A *data tree* is a rooted ordered tree which intends to express a tree-structured data. A *pattern tree* or simply *pattern* is a rooted ordered tree. A pattern tree T *occurs in* a data tree D if a set $\{d_1, ..., d_n\}$ of the nodes in D exists for the set $\{t_1, ..., t_n\}$ of all nodes in T, and satisfies all of the following three conditions.

1. For all $1 \leq i \leq n$, $label(t_i) = label(d_i)$.
2. For all $1 \leq i, j \leq n$, if $t_i = \pi(t_j)$, then $d_i = \pi(d_j)$.
3. For all $1 \leq i, j \leq n$, if $t_j = NextSibling(t_i)$, then $d_i \preceq d_j$.

We call $d_i(\{d_1, ..., d_n\})$ an *occurrence* of $t_i(T, resp.)$ in D. The set of all occurrences of t_i in D is denoted by $Occ_D(t_i)$, and its length by $|Occ_D(t_i)|$. The root of

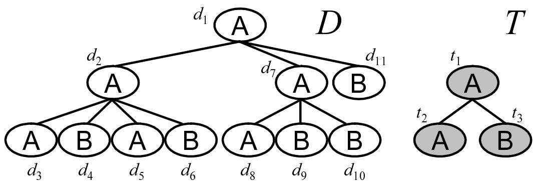


Fig. 1. Examples of a data tree and a pattern tree

T is denoted by $Root(T)$, and the number of occurrences of T in D is defined as $|Occ_D(Root(T))|$. Note that even if $\{d_1, d_2, ..., d_n\}$ and $\{d_1, d'_2, ..., d'_n\}$ where occurrence of T is d_1 are different occurrence of T in D, we do not take into account the duplication in counting $|Occ_D(Root(T))|$. We define its occurrence frequency $freq_D(T)$ as $freq_D(T) = |Occ_D(Root(T))|/|D|$. We fix σ s.t. $0 < \sigma \leq 1$ and call it the *minimum support*. If $freq_D(T) \geq \sigma$, then T is *σ-frequent* in D.

We show examples D of a data tree and T of a pattern in Fig. 1. The occurrences of T in D are $\{d_1, d_2, d_{11}\}$, $\{d_1, d_7, d_{11}\}$, $\{d_2, d_3, d_4\}$, $\{d_2, d_3, d_6\}$, $\{d_2, d_5, d_6\}$, $\{d_7, d_8, d_9\}$, and $\{d_7, d_8, d_{10}\}$. Those of $Root(T)$ are d_1, d_2, and d_7, and therefore $freq_D(T) = 3/11$.

We define the tree-structured data mining problem as follows:

Find all pattern trees which are σ-frequent in a given data tree D for a given minimum support σ.

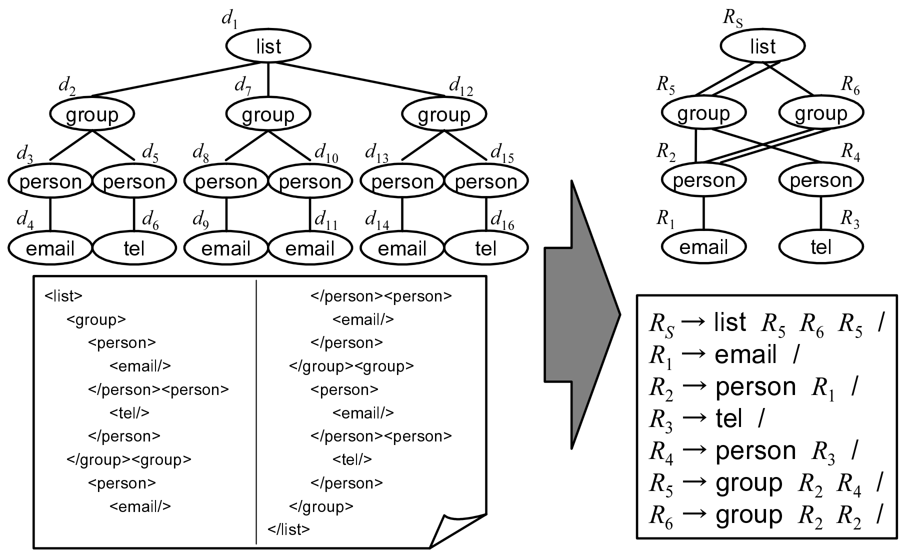
2.2 Compression Algorithm TGCA for Tree-Structured Data

The TGCA algorithm is a compression algorithm for semi-structured documents with tree grammars.

A tree grammar is a grammar generating trees. If we obtain a tree grammar which generates exactly one given tree, then the grammar can be regarded as a representation of the tree. TGCA generates such a grammar for the tree-structured data representing a given semi-structured document. Each rule in TGCA is of the form $R \rightarrow L\ \Psi\ /$, where R is a non-terminal symbol, $L \in \mathcal{L}$ is a terminal symbol which means starting tag, Ψ is a finite sequence of non-terminal symbols which refer to other rules, and '/' is a terminal symbol which means a finishing tag of L.

TGCA reads a sequence of tags in a semi-structured document from start to end, and generates a tree grammar G with a stack and a list of rules. TGCA achieves compression by representing a subtree occurring more than twice by one rule. Compressing a tree-structured data by TGCA is equivalent to naming each of the subtrees. The same subtrees are represented by one rule, and no two rules have the same non-terminal symbol in their lhs, so we can identify the subtree with the non-terminal symbol in the lhs of the rule. We use the non-terminal in the lhs of every rule as an identifier of the rules and subtrees.

Figure 2 illustrates the tree grammar generated by TGCA from a semi-structured document. From the document, TGCA generates rules in the order



```
<list>
  <group>
    <person>
      <email/>
    </person><person>
      <tel/>
    </person>
  </group><group>
    <person>
      <email/>
    </person><person>
      <email/>
    </person>
  </group><group>
    <person>
      <email/>
    </person><person>
      <tel/>
    </person>
  </group>
</list>
```

$R_S \rightarrow$ list R_5 R_6 R_5 /
$R_1 \rightarrow$ email /
$R_2 \rightarrow$ person R_1 /
$R_3 \rightarrow$ tel /
$R_4 \rightarrow$ person R_3 /
$R_5 \rightarrow$ group R_2 R_4 /
$R_6 \rightarrow$ group R_2 R_2 /

Fig. 2. Example of compression by TGCA

$R_1, ..., R_6, R_S$. The rule R_S which is generated last is called the *starting rule*. The tag sequence which represents the original tree is generated by applying other rules to the starting rule until all non-terminals are replaced. If a non-terminal R_j appears in the rhs of R_i, then R_j is called a *child rule* of R_i. A non-terminal R_j of the child rule of R_i may appear more than twice in the rhs of R_i. We write $|(R_i, R_j)| = n$ if a non-terminal R_j appears in the rhs of R_i n times. $|(R_i, R_j)| = 0$ means that R_j is not a child rule of R_i. The number of rules in G is denoted by $|G|$, and we define the ratio of the compression from D to G as $|G|/|D|$. The compression ratio of the example in Fig. 2 is 7/16.

3 Algorithms

3.1 FREQT and the Rightmost Expansion Method

The FREQT algorithm[1] is one for solving the tree-structured data mining problem. In this subsection, we explain FREQT and the rightmost expansion method, an efficient method for enumerating tree patterns in FREQT.

Rightmost expansion of a pattern tree T means to add a new node to T as the rightmost child of a node on the rightmost path, where the *rightmost path* means the one from the root to the rightmost node, and the *rightmost node* is the last one in the preorder traversal. The rightmost node of T is denoted by $rm(T)$. If a pattern T' is obtained by rightmost expansion of T, then we say T' is the *rightmost expansion* of T, and if $\pi(rm(T')) = \pi^p(rm(T))$ and $label(rm(T')) = l$, then we say that T' is (p, l)-*expansion* of T.

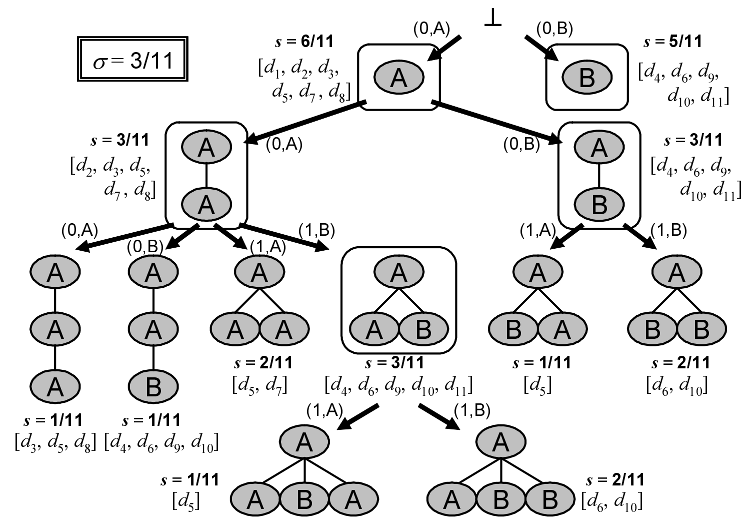


Fig. 3. The rightmost expansion tree for the data tree D in Fig. 1

For our convenience, we consider that each pattern of size 1 is generated by rightmost expansion of the empty tree $\perp$ whose size is 0. Then any T can be generated by applying rightmost expansion $|T|$ times to $\perp$, and the process of the rightmost expansion from $\perp$ to T is unique because it follows the preorder traversal of T. Therefore, the processes of generating patterns with the rightmost expansion can be expressed as a tree which is called a *rightmost expansion tree.* The rightmost expansion tree $\mathcal{R}$ is defined as $\mathcal{R} = (\mathcal{T}, \perp, \mathcal{E})$, where $\mathcal{T} = \{T \mid T$ is a pattern tree$\}$, $\perp$ is the root of $\mathcal{R}$, and $\mathcal{E} \subseteq \mathcal{T}^2$ where an edge $(T, T') \in \mathcal{E}$ iff T' is obtained by rightmost expansion of T. By using the rightmost expansion method, we can enumerate any pattern without duplication.

In FREQT, all σ-frequent patterns are found with rightmost expansion. For a tree pattern T and its every rightmost expansion T' of T, it holds that

$$Occ_D(Root(T)) \supseteq Occ_D(Root(T')) \Rightarrow freq_D(T) \geq freq_D(T') .$$

This means that if T' is σ-frequent, then T is σ-frequent. From this property, all σ-frequent patterns can be found by applying rightmost expansion repeatedly. The property also ensures that if we meet T which is not σ-frequent, we need not expand it anymore.

We can easily get $Occ_D(T')$ without scanning the whole of D if we know $Occ_D(T)$, where T' is the rightmost expansion of T. Furthermore, only if we know $Occ_D(rm(T))$ and depth of $rm(T)$ from $Root(T)$, then we can get $Occ_D(rm(T'))$ and $Occ_D(Root(T'))$ for any rightmost expansion T' of T. FREQT embraces strategy which keeps occurrences of each pattern in a data tree as a list which is called the *rightmost occurrence list.* Using the rightmost occurrence list, we can obtain space efficiency.

Algorithm 1. *Count_Rules*

Input: a TGCA-compressed text G
Output: A list $noslist_G$ of the number of the occurrences of the subtrees corresponding to each rule in G

```
 1: for each $R_k \in G$ do
 2:     $|R_k| := 0$ ;
 3: end for
 4: $R_j := R_S$ ;   $|R_S| := 1$ ;
 5: while $R_j \neq null$ do
 6:     for each $R_i$ s.t. a child rule of $R_j$ do
 7:         $|R_i| := |R_i| + |R_j| \cdot |(R_j, R_i)|$ ;
 8:     end for
 9:     $R_j := prev(R_j)$ ;    // the previous rule of $R_j$ in the generating order
10: end while
11: $noslist_G := [|R_1|, ..., |R_S|]$ ;
12: return $noslist_G$ ;
```

In Fig. 3, we show the rightmost expansion tree for the data tree D in Fig. 1 when $\sigma = 3/11$ is given. The patterns which are surrounded by a box are σ-frequent, and FREQT finds all of them. To each pattern tree in the figure, the value s of its support and its rightmost occurrence lists are attached.

The computational complexity of FREQT is $O(k^2 blN)$, where k is the size of the maximal frequent pattern, b is the maximum number of separation in D, l is the number of different labels, and N is the total length of the rightmost occurrence list of all frequent patterns.

3.2 Our Algorithm for Compressed Tree-Structured Data

We extend FREQT to apply for tree-structured data compressed by TGCA(called *TGCA data*). Note that in order to find frequent patterns, we have to count the number of occurrences of subtrees from the given TGCA data. We claim the following proposition.

Proposition 1. *Suppose a TGCA data G is obtained from a data tree D. Let $R_i \in G$, and $|Occ_D^{sub}(R_i)|$ denote the number of the subtrees which correspond to a rule R_i in D. Then it holds that*

$$|Occ_D^{sub}(R_i)| = \sum_{R_j \in G} |Occ_D^{sub}(R_j)| \cdot |(R_j, R_i)| \ .$$

This proposition means that $|Occ_D^{sub}(R_i)|$ can be computed by all of the numbers of the occurrences of the subtrees which correspond to the rules taking R_i as a child rule. Algorithm 1 shows how to count the occurrences of subtrees from a given TGCA data. The algorithm uses the property that the list $R_1, \ldots, R_{|G|-1}, R_s$ is in the reverse topological order.

The method for counting numbers of occurrences of each subtree has not been shown in [11], but it is possible to compute with considering the number of occurrences directly from TGCA data by introducing Algorithm 1.

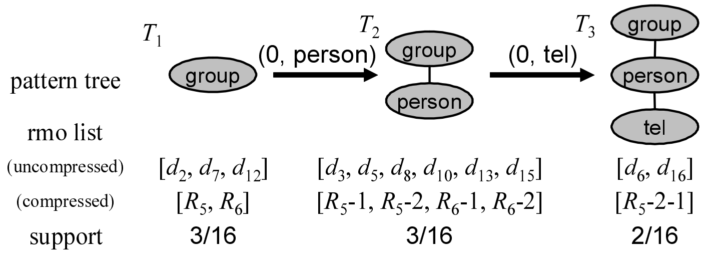


Fig. 4. Example of rightmost occurrence list(rmo list) in the original FREQT and our method in a part of the rightmost expansion tree in Fig. 2

Algorithm 2. FREQT for TGCA data

```
Input: a TGCA data $G$, a minimum support $\sigma$
Output: All $\sigma$-frequent patterns in the original data tree of $G$
1: $noslist_G := Count\_Rules(G)$ ;
2: $RMOlist_G(\bot) := [R_1, ..., R_S]$ ;
3: $RightMostExpand(RMOlist_G(\bot), noslist_G, \bot, \sigma)$ ;
```

We can apply the rightmost expansion in FREQT[1] directly to the TGCA data by using the number of the occurrences of the subtrees. We modify the rightmost expansion so that it can be applied to TGCA data. Moreover, because a TGCA data is composed of a set of rules, we also modify the rightmost occurrence list so that it keeps the rule which corresponds to the root of the pattern and the path from it to the rightmost node. Though we can not identify the only parent of a rule unlike a node in a tree, we can handle rules in the list as if they are nodes in a tree by this modification. Finally, we modify how the algorithm counts the number of the occurrences of a pattern in the data tree. Let G be a TGCA data obtained from the data tree D, and $RMOlist_G(T)$ be the rightmost occurrence list of T in D in our method. $RMOlist_G(T)$ keeps rules in it if $rm(T) = Root(T)$, and keeps paths from $Root(T)$ to $rm(T)$ otherwise. We count $freq_D(T)$ in our method as follows:

$$freq_D(T) = \left(\sum_{R_i \in RMOlist_G(T)} |Occ_D^{sub}(R_i)| \right) / \ |D|$$

We show an example of a modified rightmost occurrence list in Fig. 4. In our method, in the case of T_2 in Fig. 4, R_5 and R_6 are in $RMOlist_G(T_2)$, $|Occ_D^{sub}(R_5)| = 2$, and $|Occ_D^{sub}(R_6)| = 1$, therefore $freq_D(T_2)$ is 3/16. Because equal subtrees are processed only once, the sum of the length of the rightmost occurrence lists in our method is shorter than that in the original algorithm.

Our method is represented as the pair of Algorithm 2 and 3. Though the computational complexity of our algorithm is $O(k^2 blN)$, same as that of the original FREQT, we expect that our algorithm reduces the sum N of the rightmost occurrence lists. In the example illustrated in Fig. 4, N for the original FREQT is

Algorithm 3. $RightMostExpand(RMOlist_G(\perp), noslist_G, T, \sigma)$

```
Input: a rightmost occurrence list $RMOlist_G(\perp)$, a pattern tree $T$
1: $L := \{\}$ ;   $d := 0$ ;
2: while $d \leq Depth(rm(T))$ do
3:    for each $r \in RMOlist_G(\perp)$ do
4:       if $d = 0$ then
5:          $s := Child(r)$ ;
6:       else
7:          $s := NextSibling(\pi^{d-1}(r))$ ;
8:       end if
9:       while $s \neq \lambda$ do
10:         $T' := TreeExpand(T, d, label(s))$ ;   // $(d, label(s))$-expansion of $T$
11:         $L := L \cup \{T'\}$ ;   $Addlist(RMOlist_G(T'), s)$ ;
12:         $s := NextSibling(s)$ ;
13:      end while
14:   end for
15:   for each $T' \in L$ do
16:      if $freq_D(T') \geq \sigma$ then
17:         $output(T')$ ;
18:         $RightMostExpand(RMOlist_G(T'), noslist_G, T', \sigma)$ ;
19:      end if
20:   end for
21:   $d := d + 1$ ;
22: end while
```

11, while N for our algorithm is 7. We also expect that the better compression ratio is, the less N is and the more reduced computing time is.

4 Experimental Results

We use a FREQT program which is coded by Kudo in C++[7]. Because the program lacks the duplicate detection in [1], we implement it by ourselves. In the following, we refer to the modified program as "the original FREQT". Our method is also based on Kudo's program. Our experiments were on a PC with Xeon5150x2 and 2GB memory. We prepared five XML documents from Washington University XML Data Repository[1] for our experiments.

Table 1 shows the results of the computing time of the original FREQT and our method. Our method is faster than the original FREQT for dblp.xml and wsu.xml. On the other three XML documents, nasa.xml, swissprot.xml, and treebank.xml, we can see that the compression ratios are worse than dblp.xml and wsu.xml.

Our method takes three times slower than the original FREQT in the case of processing the rightmost occurrence lists of same length. We guess that this is because every rightmost occurrence in the list of our method has more complicated data structure than those of the original FREQT.

[1] http://www.cs.washington.edu/research/xmldatasets/

Table 1. Comparison of the original method and our method

dblp.xml ($\sigma = 1\%$)
4,700 rules / 3,332,130 nodes
time for TGCA compression: 12.205 sec.

	original method	our method
$ub = 1$ $p = 14$	11.029 sec. (3,332,130)	0.292 sec. (4,700)
$ub = 3$ $p = 88$	26.550 sec. (15,757,351)	3.312 sec. (850,608)
$ub = 5$ $p = 209$	52.371 sec. (35,601,047)	11.045 sec. (3,044,584)
$ub = \infty$ $p = 753$	173.739 sec. (120,890,452)	68.496 sec. (19,600,054)

nasa.xml ($\sigma = 0.8\%$)
8,738 rules / 476,646 nodes
time for TGCA compression: 1.836 sec.

	original method	our method
$ub = 1$ $p = 20$	1.500 sec. (476,646)	0.112 sec. (8,738)
$ub = 3$ $p = 54$	2.612 sec. (1,422,756)	1.008 sec. (239,442)
$ub = 5$ $p = 87$	4.548 sec. (2,854,316)	5.116 sec. (1,023,665)
$ub = 10$ $p = 424$	28.686 sec. (20,882,015)	62.808 sec. (12,258,891)

swissprot.xml ($\sigma = 1\%$)
58,610 rules / 2,977,031 nodes
time for TGCA compression: 12.481 sec.

	original method	our method
$ub = 1$ $p = 22$	10.069 sec. (2,977,031)	1.324 sec. (58,610)
$ub = 2$ $p = 42$	13.569 sec. (5,709,171)	7.108 sec. (1,600,454)
$ub = 3$ $p = 135$	29.094 sec. (18,433,014)	46.763 sec. (11,237,164)
$ub = \infty$ $p = 496$	91.594 sec. (68,534,369)	229.034 sec. (54,755,158)

treebank.xml ($\sigma = 1\%$)
471,312 rules / 2,437,666 nodes
time for TGCA compression: 14.805 sec.

	original method	our method
$ub = 1$ $p = 24$	8.345 sec. (2,437,666)	3.984 sec. (471,312)
$ub = 2$ $p = 47$	11.853 sec. (4,748,040)	9.257 sec. (1,687,053)
$ub = 3$ $p = 80$	17.253 sec. (7,927,094)	20.077 sec. (3,829,584)
$ub = \infty$ $p = 111$	27.766 sec. (13,342,182)	45.279 sec. (8,141,190)

wsu.xml ($\sigma = 1\%$)
20 rules / 74,557 nodes
time for TGCA compression: 0.196 sec.

	original method	our method
$ub = 1$ $p = 19$	0.228 sec. (74,557)	0.004 sec. (20)
$ub = 3$ $p = 134$	0.740 sec. (525,817)	0.004 sec. (135)
$ub = 10$ $p = 7,263$	33.114 sec. (28,500,013)	0.080 sec. (7,264)
$ub = \infty$ $p = 102,424$	549.274 sec. (401,911,777)	1.780 sec. (102,425)

ub means the upper bound of the size of patterns. We find every frequent pattern whose size is no more than n if $ub = n$. If $ub = \infty$, then we find all frequent patterns. p is the number of frequent patterns found by FREQT at its ub. Every integer with parentheses is the sum of the length of the rightmost occurrence lists about all processed patterns.

In the cases of nasa.xml and swissprot.xml, though TGCA compression works better than treebank.xml, the efficiency of computing for the cases in our method is not improved so well from that in the original method as we expected. Moreover, though both of the compression ratios of the two data are about 10 times as good as that of treebank.xml, the difference between the efficiency of the original method and that of our method is not so improved from treebank.xml. However, the length of the rightmost occurrence lists for our method approaches

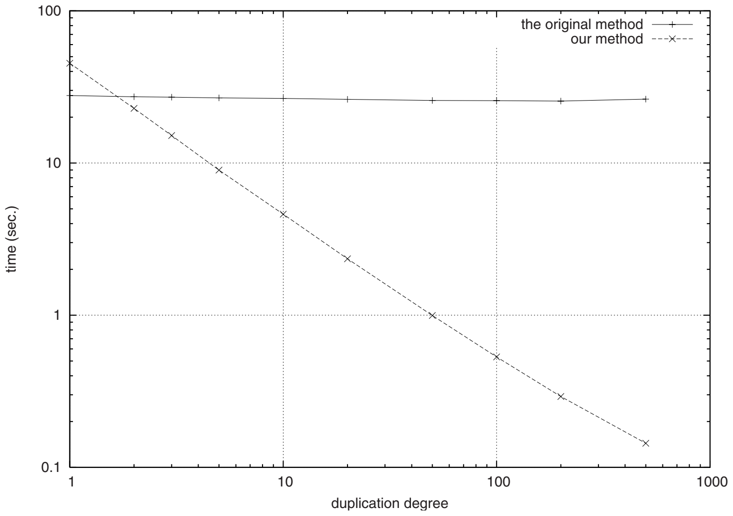


Fig. 5. Comparison of computing time in varying compression ratio

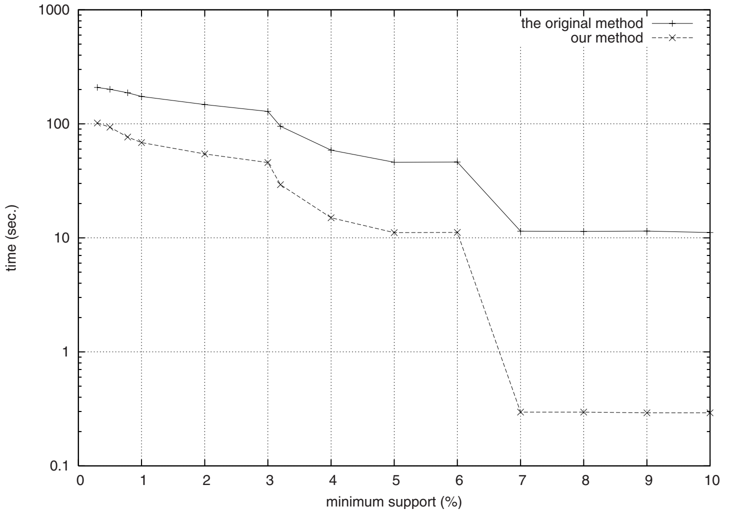


Fig. 6. Comparison of computing time in varying minimum support σ

gradually that of the list of the original one in all of the three data with worse compression ratios. We conjecture that in nasa.xml and swissprot.xml only small subtrees are duplicated in the data tree and contribute to the compression ratio.

Next, we compared computing time in various compression ratios of treebank.xml. The treebank.xml data has subtrees $s_1, \ldots, s_{56384}$ as the children of the root node. We define "duplication degree n" as the degree of duplication of these subtrees. These subtrees $s_{nm+1}, \ldots, s_{nm+n}$ are made from the same subtree as s_{nm+1} for $0 \leq m \leq \lfloor 56384/n \rfloor$. The larger the duplication degree n is, the better the compression ratio becomes.

We can expect that the length of the rightmost occurrence lists in our method is saved as the compression ratio is improved, and that computing time also becomes shorter. Figure 5 shows the graph of the result of this experiment with minimum support $\sigma = 1\%$ at $ub = \infty$. In Fig. 5, both the vertical and the horizontal axes are logarithm scale of time and duplication degree. The original method takes constant time. On the other hand, in our method the computing time becomes shorter when the duplication degree is larger.

When the minimum support σ is small, we have to consider the non-terminal symbols which have small number of occurrences. Therefore, the effect of compression becomes smaller in the case of smaller σ. We compare computing time of $ub = \infty$ for dblp.xml in varying the minimum support σ. Figure 6 shows the result of the experiment. The vertical axis in Fig. 6 is a logarithm scale of computation time. The difference of the computing time between our method and the original FREQT grows when the value of σ is enlarged. Our method has advantage when σ is larger.

5 Conclusions

FREQT and TGCA were not related to each other before. In this study, we find the method of computing with considering the number of occurrence without expanding the TGCA data. We also extend FREQT for applying to TGCA data input, and show the efficiency of our method at finding frequent patterns from tree-structured data in Section 3, and in Section 4 we confirmed the advantages of our method in the case that the data can be compressed with some good compression ratio by experiments.

Though we proposed our method on the assumption that the data tree is an ordered tree, TGCA can work also on unordered trees. In unordered tree model, since we need not keep sibling relations and can shift nodes which are siblings, better compression ratio can be expected than the case in ordered tree model. It follows that we could expect more efficiency of finding patterns.

The research which applies compression to making discovery efficient have not existed in data structures except for itemset databases of sparse data. By showing the possibility of making discovery more efficient with compression in tree-structured data in this research, the same achievements in other data structures which represent dense data can be expected.

In future work, we will give more analysis about the reason why our method is less efficient in some cases, for making the relationship between data compression and knowledge discovery clearer. Furthermore, we will consider applying the method which makes discovery efficiently with compression to finding maximal and closed frequent patterns from tree-structured data, e.g. [3][15].

Acknowledgment

This work was partially supported by Grant-in-Aid for Scientific Research (19300046 and 20700135) from JSPS and MEXT.

References

1. Asai, T., Abe, K., Kawasoe, S., Arimura, H., Sakamoto, H., Arikawa, S.: Efficient substructure discovery from large semi-structured data. In: Proc. of the 2nd SIAM International Conference on Data Mining (SDM 2002), pp. 158–174. SIAM, Philadelphia (2002)
2. Asai, T., Arimura, H., Uno, T., Nakano, S.: Discovering frequent substructures in large unordered trees. In: Grieser, G., Tanaka, Y., Yamamoto, A. (eds.) DS 2003. LNCS (LNAI), vol. 2843, pp. 47–61. Springer, Heidelberg (2003)
3. Chi, Y., Xia, Y., Yang, Y., Muntz, R.R.: Mining closed and maximal frequent subtrees from databeses of labeled rooted trees. IEEE Transactions on Knowledge and Data Engineering 17(2), 190–202 (2005)
4. Chi, Y., Yang, Y., Munts, R.R.: HybridTreeMiner: An efficient algorithm for mining frequent rooted trees and free trees using canonical forms. In: Proc. of the 16th International Conference on Scientific and Statistical Database Management (SSDBM 2004), June 2004, pp. 11–20 (2004)
5. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: Proc. of the ACM SIGMOD International Conference on Management of Data, pp. 1–12 (2000)
6. Park, M.J., Min, J.K., Chung, C.W.: XPRESS: A queriable compression for XML data. In: Proc. of the ACM SIGMOD International Conference on Management of Data, pp. 122–133 (2003)
7. Kudo, T.: FREQT: An implementation of FREQT (2003), `http://www.chasen.org/%7etaku/software/freqt/`
8. Liefke, H., Suciu, D.: XMill: An efficient compressor for XML data. In: Proc. of the ACM SIGMOD International Conference on Management of Data, vol. 29(2), pp. 153–164 (2000)
9. Manning, C.N., Witten, I.: Identifying hierarchical structure in sequences: A linear-time algorithm. Journal of Artificial Intelligence Research 7, 67–82 (1997)
10. Nijssen, S., Kok, J.N.: Efficient discovery of frequent unordered trees. In: 1st International Workshop on Mining Graphs, Trees and Sequences (MGTS 2003), pp. 55–64 (2003)
11. Onuma, J., Doi, K., Yamamoto, A.: Data compression and anti-unification for semi-structured documents with tree grammars (in Japanese). IEICE Technical Report AI2006-9, pp. 45–50 (2006)
12. Takeda, M., Shibata, Y., Matsumoto, T., Kida, T., Shinohara, A., Fukamachi, S., Shinohara, T., Arikawa, S.: Speeding up string pattern matching by text compression: The dawn of a new era. Transactions of Information Processing Society of Japan 42(3), 370–384 (2001)
13. Termier, A., Rousset, M.-C., Sebag, M.: DRYADE: a new approach for discovering closed frequent trees in heterogeneous tree databases. In: Proc. of the 4th IEEE International Conference on Data Mining (ICDM 2004), pp. 543–546 (2004)
14. Tolani, P.M., Haritosa, J.R.: XGrind: A query-friendly XML compressor. In: Proc. of the 18th International Conference on Data Engineering (ICDE 2002), pp. 225–234 (2002)
15. Xiao, Y., Yao, J.F., Li, Z., Dunham, M.: Efficient data mining for maximal frequent subtrees. In: Proc. of the 3rd IEEE International Conference on Data Mining (ICDM 2003), pp. 379–386 (2003)
16. Zaki, M.J.: Efficiently mining frequent trees in a forest. In: Proc. of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 71–80 (2002)

A Modeling Approach Using Multiple Graphs for Semi-Supervised Learning

Akihiko Izutani and Kuniaki Uehara

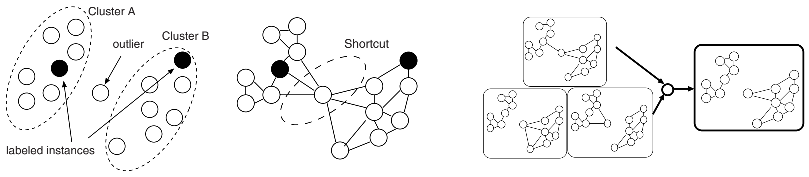
Graduate School of Engineering, Kobe University,
1-1 Rokkodai, Nada, Kobe 657-8501, Japan
izutani@ai.cs.kobe-u.ac.jp, uehara@kobe-u.ac.jp

Abstract. Most graph-based semi-supervised learning methods model the structure of a dataset as a single k-NN graph. Although graph construction is an important task, many existing graph-based methods build a graph from a dataset directly and naively. While the resulting k-NN graph provides relatively a good representation of the dataset,it generally produces inappropriate shortcuts on cluster boundaries. In this paper, we propose a novel approach for modeling and combining multiple graphs with different edge weights to avoid such undesirable behavior. Using the combination of those graphs, we can systematically reduce the effect of noise in conceptually similar fashion to an ensemble approach. Experimental results demonstrate that our approach improves classification accuracy on both benchmark and artificial datasets.

1 Introduction

In many classification domains, it is expensive to obtain a large amount of labeled data. For most supervised algorithms, however, insufficient labeled data will result in lower prediction accuracy. Meanwhile, it is often easy to collect a large amount of data without labels. Although such unlabeled data do not indicate their classes by definition, their distribution may reveal the implict structure of the dataset, which could help us to solve a classification task. Thus semi-supervised learning (SSL), which leverages both labeled and unlabeled data, has been attracting much attention in recent years.

Among various SSL approaches, graph-based approaches has been most extensively studied[1][3][4]. These approaches represent a dataset as a graph where nodes correspond to instances and weighted edges reflect pair-wise similarities between instances. Although these graph-based approaches have been proven effective for many cases, it also has a drawback that their accuracy heavily depend on an initially constructed graph. Although the value of an instance has been suffered from noise in many cases, many methods calculate the graph from a dataset naively and directly. Such graphs allow inaccurate structure (e.g. shortcut nodes between clusters), and the classification error occurs due to noise in the dataset. For example, Fig. 1 (a) presents a tiny example of dataset. The *shortcut* between clusters is generated in Fig. 1 (b) if an outlier exists in the dataset. This structure of the dataset will worsen the classification accuracy on



(a) A tiny dataset (b) A graph representation (c) Combining multiple graphs

Fig. 1. A tiny example of dataset and its graph representations

cluster boundaries, because it is difficult to predict which cluster that a neighboring instance of the *shortcut* belongs to from that graph. Therefore, a robust modeling approach is desirable for further accuracy improvement.

To resolve the above issue, we introduce a model containing multiple graphs. In this paper, we assume that the combination of different graphs covers the drawback of representations of each graph. Fig. 1 (c) illustrate the idea of our assumption. Firstly we generate different graph structures from a dataset, and the idea combine to eliminate the influences of noise. Fig. 1(c) shows three different graphs, where two graphs are not affected by the influence of the outlier instance. By combining these graphs by majority voting, a new graph representation can eliminate the shortcut (the RHS of Fig. 1(c)). Therefore, the classification accuracy could improve as compared with the single graph representation (Fig. 1 (b)).

In this paper, to implement the model, we define the different graph as follows: "graphs which have the same structure and the different weights of the edge". This is because, it is difficult to generate graphs which consist of different connections automatically. In fact, a dataset can be represented $2^{{}_nC_2}$ patterns (n is the number of instances). Hence, it has been next to impossible to select useful graphs from the vast patterns. In this paper, we emulate the model using graphs which have different weights.

The remainder of this paper is organized as follows: Section 2 explains the basic definition of graph-based SSL and summarizes the problem. Section 3 describes our proposed algorithm in detail. Section 4 presents the experimental results on both benchmark and artificial datasets, followed by the discussions in Section 5. Section 6 concludes this paper.

2 Background and Problem Setting

In recent years, graph-based SSL has drawn much attention due to its effectiveness in the real-world problem. The method constructs a graph $g = (V, E)$ where node set $V = \{v_1, v_2, \ldots, v_n\}$ represents n instances in the given dataset $\mathbf{D} = \{\mathbf{d}_1, \ldots, \mathbf{d}_n\}$ and edge set $E = \{\ldots, e_{ij}, \ldots\}$ represents relations between two instances, d_i and d_j. Most of graph-based approaches employ weighted graphs: each edge e_{ij} has a weight w_{ij} which usually represents the similarity between $\mathbf{d}_i$ and $\mathbf{d}_j$. We define the weight matrix (i.e., adjacency matrix) of g as $\mathbf{W}$ where each element w_{ij} is given by the Gaussian function:

$$w_{ij} = \exp\{-\frac{\|\mathbf{d}_i - \mathbf{d}_j\|}{2\sigma^2}\} \tag{1}$$

where σ is the scaling parameter. Especially, most methods create a k nearest neighbor graph (k-NN graph) which defined as $w_{ij} = 0$ iff $\mathbf{d}_j$ is not in the k nearest neighbors of $\mathbf{d}_i$. The graph-based SSL approach uses $\mathbf{W}$ as a learning model. In the ideal dataset, a high value is given to the weight w_{ij} if nodes v_i and v_j belong to the same cluster, and low ones are given to other edges. Thus the model can reveal the similarity of instances and the cluster.

Although many classification algorithms for the graph-based model have been proposed, but they are not essentially different. They have the same assumption that the graph are linked thickly in the inner cluster and sparsely in the others (known as *cluster assumption*). From this assumption, they predict labels of unlabeled nodes using the smoothness on a graph. The smoothness is given by [4]:

$$\epsilon(f) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(\mathbf{f}_i - \mathbf{f}_j)^2 \tag{2}$$

where $\mathbf{f}$ is C-dimensional vector (C is the number of classes) The vector of i-th node of g is denoted by $\mathbf{f}_i$, and the d-th element of $\mathbf{f}_i$ is denoted by $\mathbf{f}_{id}$. C-dimensional vectors represent the predicted class, so we call them *label vectors.* Minimizing Eq. 2, we estimate labels of unlabeled instances.

However, the graph-based approach make misclassified if the dataset is affected by noise. This is because such approaches compute a k-NN graph from a dataset directly and naively. Therefore, their structures are possible to predict wrong clusters. If a graph is generated as Fig. 1 (b), for example, an outlier node is connected to a instance labeled A. This graph indicates wrong clusters because an outlier and its neighbor nodes may be estimated as the cluster A. In spite of this problem, the graph construction method is not enough studied [2].

3 Graph Modeling Approach Using Backoff Process

In this paper, we propose a graph model which consists of multiple different graphs to deal the above problem. This idea is inspired from ensemble approaches [8] which can reduce the noise effect. Thus, we consider the idea as the ensemble of multiple graph structure and assume this model can cover drawbacks of a single k-NN graph.

To implement the idea, we define the different graph as graphs which has the same structure and the different weight of edges (e.g. Fig .2 (right)). This is because the selection of graph is difficult from computational cost and the structure of a k-NN graph is useful [1] except in the boundary between clusters. Therefore, in this paper, we emulate the basic idea using different weights. For generate different weights, we apply random values to the weight of edges. To

Dataset $\mathbf{D} =$
$$\begin{pmatrix} 1 : 0.2, 0.3, \ldots \\ 2 : 0.4, 0.2, \ldots \\ 3 : 0.3, 0.3, \ldots \end{pmatrix}$$

Weight matrix $\hat{\mathbf{W}} =$
$$\begin{bmatrix} 0 & 0.5 + 0.5p(e_{21}) & 0.4 + 0.4p(e_{31}) \\ 0.5 + 0.5p(e_{12}) & 0 & 0.6 + 0.6p(e_{32}) \\ 0.4 + 0.4p(e_{13}) & 0.6 + 0.6p(e_{23}) & 0 \end{bmatrix}$$

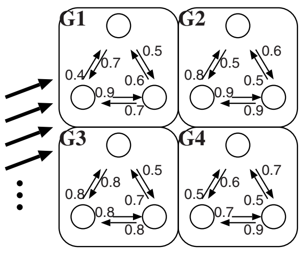


Fig. 2. A simple example of the proposed model: the graph with the backoff process can represent many different graphs

attend the problems of isolated nodes (any nodes must have at least one path to labeled nodes for classification), we focus on the *backoff process* [5]. The backoff process was introduced by Fagin et al. in Markov chain study. They propose the novel Markov chain model which has the probabilistic backward transition. The model of the backoff process adds *backward transition* and permits to return from current state to previous state probabilistically. Applying the backoff process, we obtain the graph whose nodes have at least one path to labeled nodes. Thus, we introduce the ensemble of graphs for reducing effects of noise. In this section, we explain the graph model with the backoff process and a classification algorithm in detail.

3.1 A Model for the Multiple Graphs

In this subsection, we propose a procedure for constructing a graph which includes multiple graphs. The proposed method creates a directed k-NN graph, then we apply the idea of backoff process to the initial graph.

Firstly, we construct a complete graph $g_{full} = (V, E)$. Next, we construct a k-NN graph from this graph. In this study, we assume that the graph does not have any loop edges nor multi-edges [6], thus set $w_{ii} = 0$. After the construction of the k-NN graph, we normalize the rows of the matrix as $\sum_j w_{ij} = 1$ because the sum of the transition probability must be 1. Note that g is a directed graph and thus generally $w_{ij} \neq w_{ji}$.

Secondly, we add *the probabilistic backward weight* to $\mathbf{W}$ and update $\mathbf{W}$ to $\hat{\mathbf{W}}$. The probabilistic backward weight is inspired from the idea of the backoff process, and we define the weight as $w_{ji} + p(e_{ij})w_{ij}$ on each e_{ji}. $p(e_{ij})$ is defined by the $[0, 1]$-uniform random value[1]. Thus we update the edge weight w_{ji} to $\hat{w}_{ji}$:

$$\hat{w}_{ji} = w_{ji} + p(e_{ij})w_{ij}. \tag{3}$$

Considering the above equation, weights are determined when the probabilistic term $p(e_{ij})$ is fixed. Therefore, we can generate two or more graphs that

[1] In our preliminary-experiment, the uniform distribution showed more robust performance than Gaussian or Bernoulli distribution.

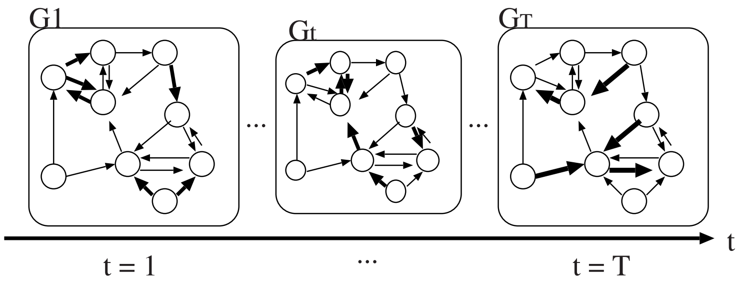


Fig. 3. An example of the procedure of the proposed label propagation: Our method use different weights by sampling from the model $\hat{\mathbf{W}}$ for the label propagation on each step t. It can be viewed as the combination of multiple graphs.

have different weights by other $p(e_ij)$ values. For example, Fig. 2 presents the proposed model. An example of the basic weight matrix $\hat{\mathbf{W}}$ is shown in Fig. 2 (left). Accordingly, we can generate different weights from $\hat{\mathbf{W}}$ because $p(e_{ij})$ is a random value. In the figure (right), for example, there is four graphs (G_1, G_2, G_3, G_4) that have the same structure and different weights. Hence, we obtain a model which includes multiple graphs.

3.2 Label Propagation on the Graph with Backoff Process

We also propose the classification method for the proposed model. In this paper, we propose an iterative algorithm that propagates the label information for all unlabeled nodes. Note that we define the propagation from v_i to v_j as the product of the label vector and its edge's weight $w_{ij}\mathbf{f}_i$. In this subsection, the proposed classification algorithm and its intuitive explanation are stated.

Firstly, we initialize the label vector of labeled nodes v_l:

$$\mathbf{f}_{ld} = \begin{cases} 1 & \text{iff } d = \text{c } (c \text{ represents the class index of } v_l) \\ 0 & \text{otherwise,} \end{cases} \tag{4}$$

Furthermore, the vectors of unlabeled nodes are set to the zero vector.

After the initialization, we start the iterative procedure to predict the label vector of unlabeled nodes. The algorithm is divided into two steps. Firstly, we calculate the propagation value from connected nodes of v_i as the sum of the propagation $\hat{w}_{ji}\mathbf{f}_j$. Note that we introduce the probability function $p(e_{ij})$, then $(w_{ji} + w_{ij}p(e_{ij}))$ represents the weight $\hat{w}_{ji}$. Secondly, we update the label vector $\mathbf{f}_i$ by the sum of the propagation. In this step, we propagate the label information (propagation values) and update the label vectors. Note that we do not change the label vectors of labeled nodes because they have true information. Therefore, we update the label vector $\mathbf{f}_i^t$ to $\mathbf{f}_i^{t+1}$:

$$\mathbf{f}_i^{t+1} = \alpha \sum_j (w_{ij}p^t(e_{ij}) + w_{ji})\mathbf{f}_j^t, \tag{5}$$

where α ($0 \leq \alpha \leq 1$) tunes the propagation speed (in this paper we set $\alpha = 0.5$). Iterating the above procedure until $t = T$ ($1 \leq t \leq T$) on each node, we determine the label vectors of unlabeled data. Finally, we output the most accurate class index m of v_i as the estimated class label of the unlabeled data $\mathbf{d}_i$. That m is calculated by

$$\arg\max_m \mathbf{f}_{im}^T. \tag{6}$$

This algorithm can be interpreted as the combination of graphs that contain different weights, because the random value $p^t(e_{ij})$ is changed in each iteration t independently. Intuitively, Fig. 3 shows the aspect of our algorithm. In our method, we probabilistically and independently determine weights of edges at each iteration (T times), and we propagate and update the label vectors on each nodes simultaneously. Thus, as shown in Fig. 3, There are different propagation values at each iteration step. Therefore, our method can be viewed as the algorithm that uses T graphs. Using this classification method, the drawback of a single graph is removed and the robustness for noise can be improved.

4 Experiments

To investigate the effectiveness of the proposed method, we perform several experiments on various datasets. Next, the effectiveness of the backoff process is examined using synthetic and real datasets.

The proposed algorithm is implemented by C++, and the random value $p(e_{ij})$ is generated by *mt19937* from the Boost C++ Libraries[2]. In all experiments, we do not set the number of iteration T. Instead of T, we stop the iteration when $\mathbf{f}_i$ (in Eq. 5) is changed less than 10^{-6}. Other parameters are set as follows: $\alpha = 0.5$, $\sigma = 1.0$ and $k = 5$. All experiments are performed 20 trials by randomly sampling labeled instances. The classification accuracy is defined as the average of all trials, and its significance is computed by the 5% t-test.

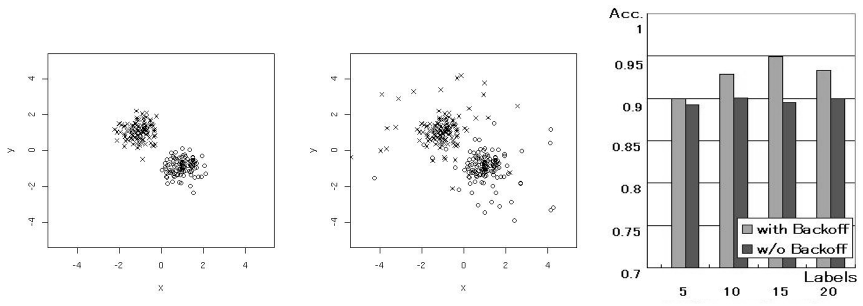
4.1 Evaluation of the Robustness to Noise on Artificial Dataset

In this subsection, we chose a simple experimental setup in order to check whether the proposed method is robust at noisy data. This experiment use the artificial dataset. Initial (no-noise) state of the artificial data (see Fig. 4(a)) includes 100 positive instances sampled from Gaussian distribution $N((1, -1), 0.5)$ and 100 negative instances from $N((-1, 1), 0.5)$ respectively[3]. Then we add 60 noise instances (sampled from the same mean and different deviation, $\sigma = 2.0$) to the above dataset (see Fig. 4(b)). We perform the label propagation with and without the backoff process on the above noisy-dataset and set the number of randomly labeled instances is $\{5, 10, 15, 20\}$. The results are presented in Fig. 4 (c).

From Fig. 4 (c), the result shows that the algorithm with the backoff process is more robust for noise. This dataset is affected by noise and a simple k-NN

[2] http://www.boost.org/

[3] The initial state of the artificial dataset is classified perfectly.



(a) (b) (c)

Fig. 4. (a): The initial dataset (b): the dataset including 30% of noise (c): the results on the dataset including 30% of noise. Note that $\circ$ and $\times$ denote the positive and negative labels, respectively.

graph has many edges in the boundary of clusters. Therefore, the classification accuracy is lower than the result on the initial dataset. However, the proposed algorithm with the backoff process reduced the classification error than one without the backoff process. This result indicate that the proposed algorithm have the potential to avoids the noise effect. Hence, the proposed method restrains the harmful influence of outliers in the dataset.

4.2 Comparison with Existing Methods

In order to verify the performance of the proposed method, we have experiments using benchmark classification datasets provided by UCI Machine Learning Repository.

In the experiment, the proposed method is compared with the following conventional methods: SGT [3], Harmonic [4], TSVM [7], SVM[4] and k-Nearest Neighbor (k-NN). SGT and Harmonic use graph structures, SGT is a kind of transductive k-NN using the graph spectral and Harmonic uses Gaussian fields over the continuous state space. TSVM is a transductive version of SVM and does not use a graph representation. In this experiment, Harmonic use 50-NN graph and k-NN set $k = 5$. SVM and k-NN are trained by using only labeled data. Unlabeled data are treated as test data. Because SGT and TSVM are binary classifiers, we show their experimental results for only binary datasets. The mean accuracy and its standard deviation on 10, 30 and 50 labels are presented in Table 1, and bold digits indicate the best results (include non-significant results).

From Table 1, the proposed method obtain the best results on some settings. Especially, our method basically performs best classification accuracy at the setting of 10 labels. Moreover, other graph-based approaches (Harmonic and SGT) are sensitive to the datasets. This is because we set the parameter of graph for all datasets and settings. Other graph-based method use a single representation

[4] In this experiment, we use LIBSVM using RBF kernel, and its σ parameter is set to the default value.

Table 1. Results for effectiveness of proposal method on UCI datasets

Dataset	Labels	Harmonic	SGT	TSVM	SVM	kNN	**Proposal**
breast	10	74.1 (±12.2)	83.8 (±5.7)	**94.5** (±2.0)	92.0 (±9.6)	80.2 (±16.8)	90.6 (±5.3)
	30	76.7 (±11.7)	85.9 (±2.3)	95.2 (±2.1)	**96.2** (±0.6)	94.6 (±1.4)	92.8 (±2.1)
	50	76.1 (±10.6)	86.5 (±1.9)	94.8 (±1.5)	**96.3** (±0.5)	95.7 (±0.7)	94.6 (±1.2)
bupa	10	**59.2** (±0.4)	56.9 (±4.4)	57.1 (±6.4)	53.7 (±6.0)	55.1 (±4.8)	**57.9** (±4.1)
	30	**61.7** (±1.8)	**59.7** (±4.4)	60.8(±4.7)	55.0 (±5.1)	59.1 (±2.9)	**60.6** (±3.1)
	50	**64.2** (±1.2)	61.8 (±3.8)	62.3 (±4.0)	55.5 (±5.4)	62.2 (±3.8)	**63.9** (±2.6)
ionosphere	10	65.1 (±0.4)	**74.9** (±5.8)	**74.4** (±7.6)	64.1 (±11.8)	65.2 (±6.7)	**72.7** (±8.3)
	30	67.2 (±1.0)	**81.4** (±4.6)	79.7 (±5.3)	78.0 (±6.3)	73.9 (±5.3)	76.6 (±4.7)
	50	69.4 (±0.8)	**84.4** (±2.7)	**84.5**(±1.8)	82.4 (±3.9)	78.2 (±6.3)	80.6 (±2.0)
pima	10	**65.6**(±1.9)	56.8 (±4.3)	62.3 (±7.3)	**66.2** (±3.7)	**64.1** (±6.4)	63.2 (±4.9)
	30	66.7 (±9.6)	61.1 (±2.6)	67.4 (±4.3)	**69.2** (±3.5)	**70.0** (±2.4)	67.1 (±2.5)
	50	67.3 (±4.2)	59.2 (±3.5)	68.3 (±5.1)	69.2 (±3.0)	**72.1** (±1.9)	68.7 (±2.3)
ecoli	10	62.0 (±9.1)	N/A	N/A	58.6 (±10.9)	54.8 (±10.2)	**67.5** (±8.6)
	30	72.9 (±5.0)			67.1 (±8.1)	**75.2** (±4.2)	**76.0** (±7.7)
	50	76.2 (±6.8)			72.4 (±6.5)	**82.0** (±2.1)	**81.1** (±2.2)
glass	10	18.7 (±4.5)	N/A	N/A	38.5 (±6.7)	38.4 (±7.1)	**46.4** (±8.1)
	30	31.2 (±9.8)			44.0 (±6.5)	**60.0** (±5.4)	**62.1** (±4.4)
	50	46.8 (±1.4)			47.6 (±5.1)	**68.5** (±2.7)	**70.0** (±3.8)
iris	10	76.2 (±11.7)	N/A	N/A	73.5 (±10.2)	71.1 (±12.5)	**91.1** (±1.8)
	30	92.3 (±1.4)			88.8 (±10.4)	**93.7**(±2.9)	**93.7** (±2.0)
	50	93.5 (±1.8)			**94.8** (±2.4)	**96.4** (±1.5)	**95.3** (±1.0)
wine	10	69.4 (±14.9)	N/A	N/A	73.2 (±14.2)	65.6 (±11.3)	**84.4** (±5.8)
	30	91.4 (±5.0)			**94.4** (±4.4)	**95.4** (±2.0)	92.7 (±1.6)
	50	95.6 (±2.1)			**97.2** (±1.3)	**96.1** (±1.5)	93.8 (±2.1)

of the graph, then they get worse results if a constructed graph is unfitted for the setting. Meanwhile, our method shows stable results because of the ensemble of multiple graphs. Consequently, the proposed method achieves high performance and stability than conventional methods.

4.3 Comparing the Effectiveness on Benchmark Dataset

From the above experiment, the proposed method achieves the better result. To examine whether the improvement of the performance is caused by the backoff process, we present the comparison of the graph with backoff process (*backoff*) and the graph without the backoff process (*non-backoff*). Table 2 shows the experimental result of 10, 30 and 50 labels settings on UCI dataset, and the statistically better results are represented in bold.

Table 2 shows that the backoff achieve better or equivalent accuracy than the non-backoff in many settings. Moreover, the standard deviation of the backoff is smaller than the non-backoff in many datasets. This is because the proposed method generates many graphs and its combination reduces the bias of noise. Thus such a combination also reduces the variance of a learning model (graph), then the results of standard deviation have been increased. Accordingly, the backoff process is effective to the real-world dataset and can reduce the variance of classification results.

4.4 Parameter Stability

In this subsection, we discuss the stability of the scaling parameter σ and k. According to [10], both experiments are performed by using the USPS

Table 2. Results for effectiveness of the backoff process on UCI datasets

Dataset	Labels	non-backoff	**backoff**	Dataset	Labels	non-backoff	**backoff**
breast	10	94.1 (±2.7)	92.1 (±5.3)	ecoli	10	66.4 (±9.1)	68.4 (±8.5)
	30	94.2 (±2.2)	94.0 (±2.1)		30	76.9 (±4.3)	75.6 (±7.7)
	50	95.4 (±1.0)	95.0 (±1.2)		50	79.7 (±2.9)	**82.0** (±2.2)
bupa	10	55.2 (±4.4)	**57.9** (±4.1)	glass	10	44.0 (±7.8)	44.3 (±8.1)
	30	59.6 (±3.6)	60.2 (±3.1)		30	53.5 (±4.3)	**57.0** (±4.4)
	50	63.5 (±2.4)	64.7 (±2.6)		50	61.8 (±2.8)	63.0 (±3.9)
ionosphere	10	67.1 (±8.7)	**73.3** (±8.3)	iris	10	89.0 (±8.3)	**93.2** (±1.8)
	30	73.3 (±6.5)	**80.0** (±4.7)		30	95.0 (±1.7)	94.7 (±2.0)
	50	83.7 (±1.9)	83.4 (±1.9)		50	96.2 (±1.5)	96.3 (±1.0)
pima	10	63.7 (±5.1)	63.4 (±4.9)	wine	10	87.8 (±7.0)	88.4 (±5.7)
	30	68.1 (±3.1)	67.2 (±2.5)		30	93.7 (±2.3)	93.7 (±1.6)
	50	69.9 (±2.3)	69.5 (±2.3)		50	**95.9** (±1.3)	94.4 (±2.0)

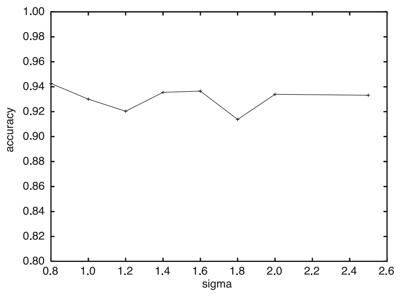


Fig. 5. Parameter stability of σ

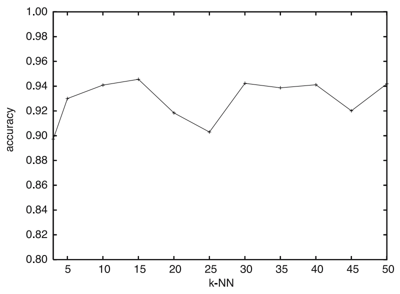


Fig. 6. Parameter stability of k-NN

handwritten digits dataset [5,6]. The experimental settings are equal to the previous experiment. The parameters are set to:

$$\sigma;\ \{0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0, 2.5\},$$
$$k;\ \{3, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50\}.$$

The mean of classification accuracy over 50 trials using different σ and k are presented in Fig. 5 and Fig. 6 respectively.

These figures show the difference of the accuracy between the best and the worst results is less than about 4 points. This result means that the proposed method has a potential of stability for both parameters. Moreover, it is appeared that k is more sensitive than σ. This result suggests the that the connection of nodes is more important than the measure of edge's weight. This is because the proposed method changes the weight of edges, but it does not change k-value.

[5] http://www.kernel-machines.org/data/

[6] According to [10], we reconstruct the dataset by the characters '1', '2', '3', '4'.

5 Discussion

We presented several experiments and show the effectiveness of the proposed method in the above section. In this section, we discuss the relationship between the proposed method and other machine learning paradigms; ensemble approach, graph smoothing method, stochastic resonance and random matrix theory. The ensemble approach improves the performance by creating and combining multiple learners. Graph smoothing method is proposed as the pre-processing method for graph construction. Stochastic resonance is a phenomenon that amplifies the weak information by the assistance of noise. Random matrix theory is a mathematic scheme which has elements sampled from the probability distribution. In the following, we discuss the relationships between the proposed method and the other approach.

Relation to Ensemble Approach: Ensemble approach is the most famous meta-learning paradigm in the machine learning domain. It creates multiple classifiers (called weak learners) which are specialized to solve a part of the learning problem. Then it makes the highly accurate learner by combining the weak leaners. This approach is successful in many cases and has been proven to be effective.

As suggested in Section 3.2, the proposed method is interpreted as the combination of graphs. That is to say, the proposed method is an ensemble approach. Especially, the proposed method is similar to bagging [8]. Bagging is a random sampling method for generating a combination of classifiers, and the randomness of its algorithm avoids local-minimum solution [9]. Because the proposed method propagates the labels on many graphs which are generated randomly, the proposed algorithm is a kind of bagging. Bagging can reduce the variance of a model and the generalization error. Therefore, the accuracy improvement of the proposed method can be interpreted as the reduction of the variance of graph construction.

Relation to Graph Smoothing methods: Another paradigm is the graph smoothing which has been proposed in recent years [10][11][12]. This algorithm modifies the weights or edges of an initial graph, in the same way as our algorithm. Especially, [11] gave interesting idea to us. They assume that the higher frequency spectral of the distribution are more likely to noise, and they use the power of weight matrix $\mathbf{W}^m$ for reducing high frequency spectral. This aspect is similar to our method, thus the improvement of our method can be interpreted as a kind of graph smoothing. However, it can only use the modification for prepossessing the classification task. The novelty of the proposed method is the graph (model) including the randomness, and weights of edges are changed in each propagation step.

Relation to Stochastic Resonance: Stochastic Resonance is researched in neuroscience and physics [13][14]. Stochastic Resonance is the generic model that improves the system performance using noise. Usually noise is considered to be worthless in the machine learning, but Stochastic Resonance improves its performance using noise.

If we assume the probabilistic backward weight as noise, we can consider the proposed method as the implementation of stochastic resonance for graph-based SSL. However, it is difficult to verify this hypothesis.

Relation to the Random Matrix Theory: Random matrix is defined as the matrix whose elements are sampled from a probability distribution [15]. Recently, this mathematical scheme has been used in many research areas, e.g., biology, physics and network modeling. Furthermore, it is applicable to the machine learning problems [16].

Meanwhile, the weight matrix of a graph with the backoff process can translate the sum of initial weight matrix and random matrix. Because the weight matrix $\hat{\mathbf{W}}^t$ after t iterations can be divided into two matrices; $\hat{\mathbf{W}}^t = \mathbf{W} + \mathbf{U}\mathbf{W}'$ where $\mathbf{W}$ is an initial graph, and $\mathbf{U}$ is a random matrix generated by $\mathbf{U}^t_{ij}$ (the element of i, j) as $[0, w_{ji}]$-uniform random value. The distribution of eigenvalues of random matrix $\mathbf{U}$ is perturbative. Affecting its perturbation to the graph spectral which calculated from $\mathbf{W}$, the proposed method may cancel the noise effects.

6 Conclusion and Future Work

In this paper, we proposed a model for the multiple graph representation using the backoff process. Furthermore, the label propagation algorithm on the graph was introduced. We evaluated the effectiveness of our method on artificial and benchmark datasets. The proposed method performed better or equivalent result than other methods in some settings. Moreover, additional experiments showed the effectiveness of the proposed method. Therefore, we conclude that the proposed method contributes to improve the performance of graph-based SSL.

In future, we will develop a more theoretical view of our method. The theoretical aspect can be mainly divided into three-ways: the ensemble approach, the Stochastic Resonance phenomenon and the random matrix theory. Especially the random matrix which has elements sampled from the probability distribution is studied theoretically [15][16], and its property is similar to the proposed method.

References

1. Chapelle, O., Schölkopf, B., Zien, A.: Semi-Supervised Learning. MIT Press, Cambridge (2006)
2. Zhu, X., Ghahamani, Z.: Semi-supervised learning using gaussian fields and harmonic functions. In: Proc. of the 20th ICML, pp. 912–919 (2003)
3. Joachims, T.: Transductive learning via spectral graph partitioning. In: Proc. of the 20th ICML, pp. 290–297 (2003)
4. Zhu, X.: Semi-supervised learning literature survey (2005), `http://www.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf`
5. Fagin, R., Karlin, A.R., Kleinberg, J., Raghavan, P., Rajagopalan, S., Rubinfeld, R., Sudan, M., Tomkins, A.: Random walks with back buttons. Annals of Applied Probability 11(3), 810–862 (2001)

6. Diestel, R.: Graph Theory. Springer, New York (2004)
7. Joachims, T.: Transductive inference for text classification using support vector machines. In: Proc. of the 16th ICML, pp. 200–209 (1999)
8. Alpaydin, E.: Introduction to Machine Learning. MIT Press, Cambridge (2004)
9. García-Pedrajas, N., García-Osorio, C., Fyfe, C.: Nonlinear boosting projections for ensemble construction. J. Mach. Learn. Res. 8, 1–33 (2007)
10. Wang, F., Zhang, C.: Label popagation through linear neighborhoods. In: Proc. of the 23rd ICML, pp. 985–992 (2006)
11. Zhou, X., Li, C.: Combining smooth graphs with semi-supervised classification. In: Ng, W.-K., Kitsuregawa, M., Li, J., Chang, K. (eds.) PAKDD 2006. LNCS (LNAI), vol. 3918, pp. 400–409. Springer, Heidelberg (2006)
12. Shin, H., Hill, N.J., Rätsch, G.: Graph based semi-supervised learning with sharper edges. In: Proc. of 17th ECML, pp. 401–412 (2006)
13. Moss, F., Ward, L.M., Sannita, W.G.: Stochastic resonance and sensory information processing: a tutorial and review of application. Clinical Neurophysiology 115, 267–281 (2004)
14. Nobori, T., Matsui, N.: Stochastic resonance neural network and its performance. In: Proc. of IJCNN 2000, vol. 2, p. 2013 (2000)
15. Nagao, T.: Random Matrices: An Introduction. University of Tokyo Press (2005)
16. Achlioptas, D.: Random matrices in data analysis. In: Proc. of the 8th PKDD, pp. 1–7 (2004)

String Kernels Based on Variable-Length-Don't-Care Patterns

Kazuyuki Narisawa[1], Hideo Bannai[1], Kohei Hatano[1], Shunsuke Inenaga[2], and Masayuki Takeda[1]

[1] Department of Informatics, Kyushu University
[2] Graduate School of Information Science and Electrical Engineering, Kyushu University
744 Motooka, Nishiku, Fukuoka, 819–0395 Japan
{k-nari,bannai,hatano,takeda}@i.kyushu-u.ac.jp,
inenaga@c.csce.kyushu-u.ac.jp

Abstract. We propose a new string kernel based on *variable-length-don't-care patterns* (*VLDC patterns*). A VLDC pattern is an element of $(\Sigma \cup \{\star\})^*$, where Σ is an alphabet and $\star$ is the variable-length-don't-care symbol that matches any string in Σ^*. The number of VLDC patterns matching a given string s of length n is $O(2^{2n})$. We present an $O(n^5)$ algorithm for computing the kernel value. We also propose variations of the kernel which modify the relative weights of each pattern. We evaluate our kernels using a support vector machine to classify spam data.

1 Introduction

Text classification techniques in machine learning have a wide range of applications such as natural language processing and bioinformatics. These techniques are based on the similarity of words, sequences or other string units with a dictionary of natural language, biological actions and periodicities. It is a challenging task to efficiently compute these similarity measures, and the kernel method is one of the classical approaches for evaluating string similarity.

Several string kernels have been developed for various types of data. Mismatch kernels [1] are used for biological data such as DNA and protein sequences. This kernel allows mismatches between sequences. Rational kernels [2,3] can separate regular languages, using weighted transducers or rational relations.

The string subsequence kernels (SSKs) and N-gram kernels (NGKs) [4] are popular string kernels. They are easy to compute, and can be applied to a variety of data since they do not make too many assumptions about the underlying text structure. SSKs map strings to a feature space where each dimension corresponds to a subsequence of length n. The value of the dimension depends on a subsequence gap weight and how the subsequence occurs in each string. NGKs map strings to a feature space where each dimension corresponds to a substring of length n, or n-gram. The value of the dimension depends on how many of each n-gram the string contains.

These kernels, however, have limitations. That is, these kernels are not suitable when the label of the texts depend not only on some relevant pattern (substring or subsequence) appearing in the texts, but on the order of such relevant patterns. For example, assume that labels of texts are positive if the word "*aab*" appears, followed by another word "*aba*". For such tests, NGKs cannot capture the order of strings in the texts. SSKs takes into account the orders of subsequences, but cannot capture the order of strings, either. In the case of SSKs, dissimilar strings may be judged similar due to the dimensions of subsequence. For example, strings "*xxxabcxxx*" and "*yayybyycy*" are clearly different. However, they may be judged similar because they have the same subsequence dimension "*abc*". Although the SSK resolves this issue by the gap weight, preferring that each character of the subsequence occur *closer* together, it cannot handle the order of two (near-substring) subsequences as in the first example.

In this paper, we propose a new string kernels based on *variable-length-don't-care* (*VLDC*) patterns [5,6,7]. A VLDC pattern is an element of $\Pi = (\Sigma \cup \{\star\})^*$, where Σ is alphabet and $\star$ is a wildcard that matches any string. In the running example, the VLDC pattern $ab \star bb \star ba$ matches $abaabbaba$ by replacing the $\star$'s with aa and a, respectively. The VLDC pattern gives the best of both worlds and more, since it contains both the set of substrings and subsequences as a subset. Unlike NGKs and SSKs, our kernel can handle the situations where the order of relevant strings influence the labels. Our kernel is not limited to exact matching substrings as in NGKs, and can also handle the order of two substrings.

The number of VLDC patterns that matches a given string s of length n is $O(2^{2n})$. We define several versions of kernels based on the VLDC pattern, and propose algorithms that compute kernels, for example, for a given a pair of strings in $O(max\{n^4, \ell 2^\ell |\Sigma|^\ell\})$ time and $O(n^4)$ space, where ℓ is the length of VLDC patterns and n is the maximum length of given strings. Unfortunately, the time and space complexity of computing VLDC kernels are still quite high to apply for large data, but our kernels can be applied for data of small size in reasonable computation time.

We show the running times of VLDC kernel computation in a preliminary experiment, and compare with that of SSKs, using random text data. Our main experiment is a comparison of the performance in text classification, using spam data. The experiment shows that VLDC kernels outperform both SSKs and NGKs.

2 Preliminaries

2.1 VLDC Patterns

Let Σ be a finite *alphabet* of size σ. An element of Σ^* is called a *string*. Strings x, y and z are said to be a *prefix*, *substring*, and *suffix* of the string $u = xyz$. The length of any string u is denoted by $|u|$. Let ε denote the empty string, that is, $|\varepsilon| = 0$. Let $\Sigma^+ = \Sigma^* - \{\varepsilon\}$. The i-th character of a string u is denoted by $u[i]$ for $1 \leq i \leq |u|$, and the substring of u that begins at position i and ends at position j is denoted by $u[i, j]$ for $1 \leq i \leq j \leq |u|$.

Let $\star$ denote a special symbol called a *variable length don't care* (*VLDC*) or a *wildcard*, which matches any string in Σ^*. Let $\Pi = (\Sigma \cup \{\star\})^*$. Any element p of Π is called a *variable-length-don't-care pattern* (or a *VLDC pattern* in short). The length of any VLDC pattern p, denoted $|p|$, is the number of characters plus the number of $\star$'s contained in p.

Definition 1. *For any string $x \in \Sigma^*$ and any VLDC pattern $p \in \Pi$, p is said to* match *x iff x is obtained by replacing the $\star$'s in p with some strings in Σ^*. We write $p \preceq x$ when p matches x.*

Since $\star$ matches the empty string ε, for any string $x \in \Sigma^*$ there are an infinite number of VLDC patterns matching x. To limit the number of VLDC patterns matching x, consider the subset $\Pi' \subset \Pi$ of VLDC patterns which start and end with $\star$ and contain no consecutive $\star$'s. For instance, $\star a \star b \star \in \Pi'$ but $\star a \star\star b \star \notin \Pi'$ and $a \star b \star \notin \Pi'$.

Definition 2. *For any string $x \in \Sigma^*$, we define $P(x)$ by*

$$P(x) = \{p \in \Pi' \mid p \preceq x\}.$$

For any string x of length n, the size of $P(x)$ is $O(2^{n-1}\sigma^n)$. Note that $P(x)$ still contains all interesting VLDC patterns matching x, and contains only those. Using the running example, $\star a \star b \star \in \Pi'$ matches a string x if and only if $\star a \star\star b \star \notin \Pi'$ matches x. Also, we have that $\star a \star b \star \in \Pi'$ matches x if $a \star b \star \notin \Pi'$ matches x (the opposite is not true).

For each VLDC pattern p, we consider the multiset of the "maximal run" of characters of Σ in p, as follows.

Definition 3. *For any VLDC pattern $p \in \Pi'$, we define the multiset $Con(p)$ of strings as*

$$Con(p) = \begin{cases} \{w_1, \dots, w_n\} & \text{if } p = \star w_1 \star \cdots \star w_n \star \text{ and } w_i \in \Sigma^+ \ (1 \le i \le n), \\ \emptyset & \text{if } p = \star. \end{cases}$$

2.2 Support Vector Machines and Kernels

In this subsection, we review the outline of Support Vector Machines(SVMs) and kernels. SVMs are a machine learning algorithm based on the idea of a kernel mapping, and were introduced by Boser et al [8]. This methods learn the hyperplane in order to separate two sets of points so as to maximize the margin distance, using several statistic properties. One of its properties is the mapping to high dimensional feature space, that shape the performance of SVM. So called kernels, denoted by

$$K(s,t) = \phi(s) \cdot \phi(t),$$

are used in this method. The function $K(s,t)$ calculates the inner product between instance s and t in a high dimensional feature space defined by a mapping ϕ.

3 VLDC Kernels

For a given string, we first define the VLDC kernel which considers the hyper-space where each dimension represents a VLDC pattern and its value is 1 if the pattern matches the string, and 0 otherwise. Thus, the kernel can then be defined as the number of VLDC patterns that match both strings.

Definition 4 (VLDC Kernel). *For any strings $s, t \in \Sigma^*$, we define the* VLDC kernel $K(s,t)$ *by*

$$K(s,t) = \sum_{p \in \Pi'} \delta(p,s,t),$$

where

$$\delta(p,s,t) = \begin{cases} 1 & \textit{if } p \in P(s) \cap P(t), \\ 0 & \textit{otherwise.} \end{cases}$$

In many situations, it is more natural to prefer patterns where the occurrence of each character is closer together, in order to better capture the characteristics of the text. In the SSKs, this is achieved through gap weights. For the VLDC kernel, we can achieve this by preferring longer contiguous constant parts in the VLDC pattern, as follows:

Definition 5 (Weighted VLDC Kernel). *For any strings $s, t \in \Sigma^*$, we define the* weighted VLDC kernel $K'(s,t)$ *by*

$$K'(s,t) = \sum_{p \in \Pi'} \delta'(p,s,t),$$

where

$$\delta'(p,s,t) = \begin{cases} \sum_{x \in Con(p)} \lambda^{|x|} & \textit{if } p \in P(s) \cap P(t) \textit{ and } Con(p) \neq \emptyset, \\ 0 & \textit{otherwise,} \end{cases}$$

for some $\lambda > 1$.

For the weighted VLDC kernel, we also consider a version where the length of the VLDC pattern is limited, keeping the calculation tractable.

Definition 6 (Length Restricted Weighted VLDC Kernel). *For any strings $s, t \in \Sigma^*$, we define the* length restricted weighted VLDC kernel $K''_\ell(s,t)$ *by*

$$K''_\ell(s,t) = \sum_{p \in \Pi'} \delta''(p,s,t,\ell),$$

where

$$\delta''(p,s,t,\ell) = \begin{cases} \sum_{x \in Con(p)} \lambda^{|x|} & \textit{if } p \in P(s) \cap P(t),\ Con(p) \neq \emptyset \textit{ and } |p| \leq \ell, \\ 0 & \textit{otherwise,} \end{cases}$$

for some $\lambda > 1$ and $\ell \geq 1$.

We can also restrict the number of wildcards (VLDCs) used in the pattern, as follows.

Definition 7 (Wildcard Restricted Weighted VLDC Kernel). *For any strings $s, t \in \Sigma^*$, we define the* wildcard restricted weighted VLDC kernel *$K_r'''(s,t)$ by*

$$K_r'''(s,t) = \sum_{p \in \Pi'} \delta'''(p,s,t,r),$$

where

$$\delta'''(p,s,t,r) = \begin{cases} \sum_{x \in Con(p)} \lambda^{|x|} & \text{if } p \in P(s) \cap P(t),\ 0 < |Con(p)| \leq r+1, \\ 0 & \text{otherwise,} \end{cases}$$

for some $\lambda > 1$ and $r \geq 1$.

4 Algorithms to Compute VLDC Kernels

In this section we present our algorithms to compute string kernels based on VLDC patterns introduced in the previous section.

We use the following data structure in our algorithms.

Definition 8 (WDAWG). *The* Wildcard DAWG *(*WDAWG*) of a string x, denoted $WDAWG(x)$, is the smallest automaton that accepts all VLDC patterns $p \in \Pi$ such that $p \preceq x$.*

$WDAWG(x)$ with $x = abbab$ is shown in Fig. 1.

By definition, every VLDC pattern $q \in P(x)$ is accepted by $WDAWG(x)$. This implies that there is always a path spelling out q from the initial state of $WDAWG(x)$.

Theorem 1 ([9,10]). *For any string x of length n, $WDAWG(x)$ requires $O(n^2)$ space and can be constructed in linear time in the output size.*

Using WDAWGs we obtain the following results:

Theorem 2. *For any strings s and t, $K(s,t)$ can be computed in $O(n^5)$ time and space, where $n = \max\{|s|,|t|\}$.*

Proof. It follows from Theorem 1 that $WDAWG(s)$ and $WDAWG(t)$ can be constructed in $O(n^2)$ time and space. Then we can easily construct a DFA which accepts every element of $P(s) \cap P(t)$ by combining $WDAWG(s)$ and $WDAWG(t)$. The size of such a DFA will be $O(n^4)$ in the worst case. Furthermore, since $K(s,t) = \sum_{p \in \Pi} \delta(p,s,t) = |P(s) \cap P(t)|$, $K(s,t)$ can be computed by a simple linear-time depth-first traversal on the combined DFA, to count the number of paths from the initial state to all accepting states which start and end with $\star$ and contain no consecutive $\star$'s. However, since the number of such paths can be very large, that is, up to $O(2^n \sigma^n)$, treating these numbers at each state can require up to $O(n)$ time and space. Therefore, $K(s,t)$ can be calculated in a total of $O(n^5)$ time and space in the worst case. $\square$

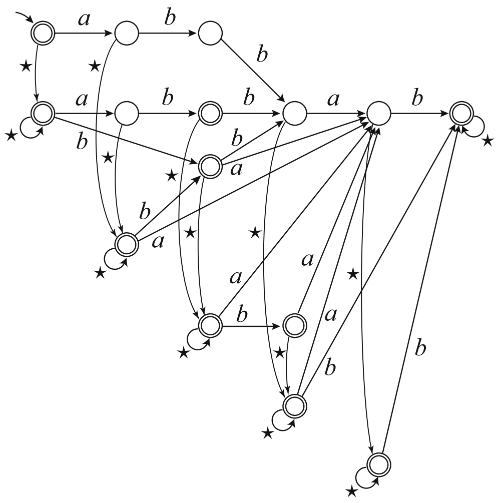


Fig. 1. *WDAWG*(x) with $x = abbab$

Theorem 3. *For any strings s and t, $K'(s,t)$ can be computed in $O(n2^{2n})$ time and $O(n^4)$ space, where $n = \max\{|s|, |t|\}$.*

Proof. For each VLDC pattern $p \in P(s) \cap P(t)$ with $Con(p) \neq \emptyset$, we need to compute $\delta'(p,s,t) = \sum_{x \in Con(p)} \lambda^{|x|}$. Hence we execute a depth-first traversal on the DFA for all the paths from the initial state to all accepting states which start and end with $\star$ and contain no consecutive $\star$'s. There are $O(2^{2n})$ such paths. At each state corresponding to VLDC pattern p, the value of $\delta'(p,s,t)$ can be computed in $O(n)$ time utilizing information of the parent state in the traversal. Overall, the total time cost is $O(n2^n\sigma^n)$. The value of $K'(s,t)$ grows up to $O(2^{2n}\lambda^n)$ which requires $O(n)$ bits. Thus the space requirement is bounded by $O(n^4)$, the size of the combined DFA. $\square$

Due to the above theorem, computing the weighted VLDC kernel $K'(s,t)$ is not feasible for large n (long strings). However, the next theorem states that the length restricted weighted VLDC kernel $K''_\ell(s,t)$ is computable in cases where ℓ is small.

Theorem 4. *For any strings s, t and integer ℓ, $K''_\ell(s,t)$ can be computed in $O(\max\{n^4, \ell 2^\ell \sigma^\ell\})$ time and $O(n^4)$ space, where $n = \max\{|s|, |t|\}$.*

Proof. Since we have only to consider the VLDC patterns of length at most ℓ, the total number of paths we traverse in the DFA is $O(2^\ell\sigma^\ell)$. At each state p we need $O(\ell)$ time to compute $\delta''(p,s,t,\ell)$. Hence the time cost is $O(\max\{n^4, \ell 2^\ell \sigma^\ell\})$. The space requirement is $O(n^4)$ since $\ell \leq n$. $\square$

The wildcard restricted VLDC kernel $K'''_r(s,t)$ is also computable when r is small, since:

Theorem 5. *For any strings s, t and integer r, $K_r'''(s,t)$ can be computed in $O(n^{r+1}r)$ time and $O(n^4)$ space, where $n = \max\{|s|,|t|\}$.*

Proof. The number of VLDC patterns which have at most r wildcard symbols $\star$ and match both s and t is

$$O(\sum_{i=2}^{r} \times {}_{n-1}C_{i-2} \times n^2(i-1)) = O(n^r r).$$

Since at each state p we need $O(n)$ time to compute $\delta'''(p,s,t,r)$, the total time cost is $O(n^{r+1}r)$. The space requirement is $O(n^4)$ since $r \leq n$. $\square$

5 Computational Experiments

In this section, we compare our VLDC kernels with SSK and NGK by experiments.

5.1 Time Comparison

Firstly, we show our experiments for comparison of computational times with randomly generated strings. The parameters of random strings are the alphabet size such as 4, 25 and 52 and string length such as 100, 200, $\cdots$, 500. We compare the length restricted VLDC kernel K'' with SSK. The lengths of the VLDC patterns for K'' and the subsequence for SSK are set to 5, 10 and 15. We utilized the SSK algorithm of [4]. We used a CentOS Linux desktop computer with two 3GHz dual core Xeon processors and 16GB memory.

Table 1 shows the running times (sec) for computing the length restricted VLDC kernel K'' for each VLDC pattern length. We have not computed kernel values for the cells with "-", since it will take too much time. Since our algorithm traverses all possible paths in the DFA where each state has at most σ transitions, the running time grows exponentially with respect to the alphabet size σ (see Theorem 4). Table 2 shows the running times (sec) for computing the SSK for each subsequence length. The running time of the SSK algorithm does not depend on the alphabet size.

5.2 Performance Comparison

We evaluated the performances of our VLDC kernels K and K'', compared to other kernels SSK and NGK. The classification algorithm we used was a free software SVM^{light}[1].

The values of our VLDC kernels may be huge, and thus we used the normalized value $\overline{K}$ for K, as follows:

$$\overline{K}(s,t) = \frac{K(s,t)}{\sqrt{K(s,s)K(t,t)}},$$

where $s,t \in \Sigma^*$. The normalized value $\overline{K''}$ for K'' is defined similarly.

[1] http://svmlight.joachims.org/

Table 1. Computational times (sec) of the VLDC Kernel K'' values for random strings, with alphabet sizes 4, 26 and 52 and string lengths 100, 200, $\cdots$, 500. The the VLDC pattern length parameter ℓ was set to 5, 10 and 15. We did not compute the kernel value for each cell marked with "-", as it will apparently take too much time.

alphabet size	VLDC pattern length	string length				
		100	200	300	400	500
4	5	0.1125	0.7650	1.7250	2.9150	4.7250
	10	0.1156	0.7356	1.4467	2.7500	4.5633
	15	0.3500	1.3500	1.900	3.3100	5.1300
26	5	0.1550	0.9525	2.1350	3.8725	6.4475
	10	6.6400	34.2989	41.5500	49.6400	58.1067
	15	7468.7900	-	-	-	-
52	5	0.2175	1.445	3.5825	6.8050	11.2750
	10	24.2500	307.26	859.6867	1530.8200	2098.3830
	15	-	-	-	-	-

Table 2. Computational times (sec) of the SSK value for random strings, with alphabet sizes 4, 26 and 52 and string lengths 100, 200, $\cdots$, 500. The subsequence length parameter in SSK was set to 5, 10 and 15.

alphabet size	subsequence length	string length				
		100	200	300	400	500
4	5	0.0004	0.0040	0.0132	0.0244	0.0384
	10	0.0012	0.0120	0.0256	0.0504	0.0736
	15	0.0036	0.0184	0.0452	0.0780	0.1216
26	5	0.0000	0.0040	0.0116	0.0212	0.0356
	10	0.0020	0.0108	0.0272	0.0500	0.0792
	15	0.0020	0.0180	0.0404	0.0712	0.1180
52	5	0.0004	0.0040	0.0116	0.0228	0.0368
	10	0.0012	0.0108	0.0264	0.0488	0.0760
	15	0.0020	0.0172	0.0388	0.0720	0.1144

In our experiments, we used spam data collected from the forum of Yahoo! Japan Finance[2], which were also used in [11]. We used 4 data sets of forum ID "4936", "4974", "6830", and "8473". The website contains a lot of spam messages, and the administrators watch over the forums and delete spam messages. We classified the deleted messages as spam and other remaining messages as non-spam. In order for our algorithms to run in a reasonable amount of time, we only used messages of length at most 200 in each data set. Table 3 shows the number of instances (messages) in each data set.

In Table 4, we show the accuracy (%) of classification with our VLDC kernels K, K'', SSK and NGK. The length restricted VLDC kernel K'', SSK and NGK require the following parameters; K'': VLDC pattern length ℓ and weight λ, SSK: subsequence length and gap weight, NGK: substring length.

[2] `http://quote.yahoo.co.jp`

Table 3. The number of instances in each data set

ID	spam	nonspam
4296	24	48
4974	4	14
6830	28	102
8473	22	84

Table 4. The performances of VLDC kernels K, K'', SSK and NGK. Each cell shows the accuracy(%) by SVMlight. In the results of the length restricted VLDC kernel K'', "length" means the VLDC pattern length ℓ and "weight" means consecutive string weight $\lambda > 1$. In the results of SSK, "length" means subsequence length, and "weight" means the gap weight $0 < \delta < 1$. In the results of NGK, "length" means substring length, and NGK needs no string weight parameters.

			data set ID			
Kernel	Length	Weight	4296	4974	6830	8473
VLDC K			66.67	**77.78**	80.00	79.25
VLDC K''	5	1.1	68.06	**77.78**	91.54	95.29
		1.5	68.06	**77.78**	91.54	95.29
		2.0	**76.39**	72.23	**93.08**	**96.23**
	10	1.1	66.67	72.23	89.23	93.40
		1.5	66.67	72.23	89.23	93.40
		2.0	66.67	72.23	89.23	93.40
SSK	5	0.1	63.89	**77.78**	70.77	79.25
		0.3	66.67	72.23	76.93	86.79
		0.5	66.67	72.23	83.08	89.62
		0.7	69.45	72.23	67.69	86.80
		0.9	69.44	66.67	62.31	67.92
	10	0.1	66.67	50.00	79.24	51.89
		0.3	50.00	72.23	74.62	78.31
		0.5	66.67	72.23	76.92	81.14
		0.7	66.67	72.23	82.31	83.97
		0.9	66.67	72.23	78.46	79.25
	15	0.1	66.67	**77.78**	21.54	20.75
		0.3	66.67	50.00	78.46	79.25
		0.5	66.67	72.23	76.92	79.25
		0.7	65.28	72.23	83.08	79.25
		0.9	66.67	77.78	82.31	79.25
NGK	5		70.84	72.23	90.00	92.46
	6		70.84	72.23	90.00	89.63
	7		69.45	72.23	89.23	88.68
	8		69.45	72.23	90.00	87.74
	9		69.45	72.23	88.46	85.85
	10		69.45	72.23	86.92	84.91

Shorter patterns tend to give good performances for all kernels except for K which has no length parameter. The weight parameter λ of K'' was the best with $\lambda = 2.0$. Observe that, for all the forums, our length restricted VLDC kernel K'' shows the best performance.

6 Conclusions and Future Work

In this paper we proposed four types of string kernels based on VLDC patterns. Firstly, we introduced the VLDC kernel based on the all common VLDC patterns for a given pair of strings. This kernel is computable in $O(n^5)$ time and space, where n is the input string length. The needlessness of parameters for pattern length or weights is a merit of this kernel. Secondly, the weighted VLDC kernel was introduced, in which the consecutive constant characters are weighted according to the length of the consecutiveness. This kernel requires $O(n2^{2n})$ time and $O(n^4)$ space. The third kernel, the length restricted weighted VLDC kernel, has a parameter ℓ for the length of VLDC patterns, and thus is computable in $O(\max\{n^4, \ell 2^\ell \sigma^\ell\})$ time and $O(n^4)$ space. Lastly, we also proposed the wildcard restricted weighted VLDC kernel where the number of $\star$'s in each VLDC pattern is restricted by a parameter r. This kernel can be computed in $O(n^{r+1}r)$ time and $O(n^4)$ space. We evaluated performances of our VLDC kernels for the computation time and text classification ability by experiments. Computing our VLDC kernels took longer time than SSKs and NGK. VLDC kernels, however, outperformed SSK and NGK in the classification accuracy in the spam detecting experiments.

Our VLDC kernels are shown to have high potential to classify text data. However, their computational complexities may be too large to apply to large text datasets. We are currently investigating ways to lower this complexity, while still retaining the high classification accuracy.

Rational Kernels [2,3] are known to be able to classify regular languages. We would also like to investigate the expressiveness of our VLDC kernels in this direction, and determine the class of languages that they can classify.

References

1. Leslie, C.S., Eskin, E., Weston, J., Noble, W.S.: Mismatch string kernels for svm protein classification. In: Advance in Neural Information Processing Systems 15, pp. 1417–1424 (2002)
2. Cortes, C., Haffner, P., Mohri, M.: Rational kernels: Theory and algorithms. Journal of Machine Learning Research 5, 1035–1062 (2004)
3. Cortes, C., Kontorovich, L., Mohri, M.: Learning languages with rational kernels. In: Proceedings of the 20th Annual Conference on Learning Theory, pp. 349–364 (2007)
4. Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., Watkins, C.: Text classification using string kernels. Journal of Machine Learning Research 2, 419–444 (2002)

5. Inenaga, S., Bannai, H., Shinohara, A., Takeda, M., Arikawa, S.: Discovering best variable-length-don't-care patterns. In: Lange, S., Satoh, K., Smith, C.H. (eds.) DS 2002. LNCS, vol. 2534, pp. 86–97. Springer, Heidelberg (2002)
6. Rahman, M.S., Iliopoulos, C.S., Lee, I., Mohamed, M., Smyth, W.F.: Finding patterns with variable length gaps or don't cares. In: Chen, D.Z., Lee, D.T. (eds.) COCOON 2006. LNCS, vol. 4112, pp. 146–155. Springer, Heidelberg (2006)
7. Navarro, G., Raffinot, M.: Fast and simple character classes and bounded gaps pattern matching. Journal of Computational Biology 10(6), 903–923 (2003)
8. Boser, B.E., Guyon, I.M., Vapnik, V.N.: A training algorithm for optimal margin classifiers. In: COLT 1992: Proceedings of the fifth annual workshop on Computational learning theory, pp. 144–152. ACM, New York (1992)
9. Inenaga, S., Takeda, M., Shinohara, A., Hoshino, H., Arikawa, S.: The minimum DAWG for all suffixes of a string and its applications. In: Apostolico, A., Takeda, M. (eds.) CPM 2002. LNCS, vol. 2373, pp. 153–167. Springer, Heidelberg (2002)
10. Inenaga, S., Shinohara, A., Takeda, M., Bannai, H., Arikawa, S.: Space-economical construction of index structures for all suffixes of a string. In: Diks, K., Rytter, W. (eds.) MFCS 2002. LNCS, vol. 2420, pp. 341–352. Springer, Heidelberg (2002)
11. Narisawa, K., Bannai, H., Hatano, K., Takeda, M.: Unsupervised spam detection based on string alienness measures. In: Corruble, V., Takeda, M., Suzuki, E. (eds.) DS 2007. LNCS (LNAI), vol. 4755, pp. 159–172. Springer, Heidelberg (2007)

Unsupervised Spam Detection by Document Complexity Estimation

Takashi Uemura[1], Daisuke Ikeda[2], and Hiroki Arimura[1]

[1] Hokkaido University, Kita 14, Nishi 9, Kita-ku, Sapporo 060-0814, Japan
[2] Kyushu University, 744 Motooka Nishi-ku, Fukuoka 819-0395, Japan

Abstract. In this paper, we study a content-based spam detection for a specific type of spams, called *blog* and *bulletin board spams*. We develop an efficient unsupervised algorithm DCE that detects spam documents from a mixture of spam and non-spam documents using an entropy-like measure, called the *document complexity*. Using suffix trees, the algorithm computes the document complexity for all documents in linear time w.r.t. the total length of input documents. Experimental results showed that our algorithm especially works well for detecting word salad spams, which are believed to be difficult to detect automatically.

1 Introduction

Spam messages are ubiquitous in the diversified media of the Internet. Recently, more than 90 percent of all emails have been reported as spam. As new communication media, such as blogs, have appeared, new types of spam messages adapted for these media have also arisen. For instance, more than 75 percent of all blog entries are blog spams, called splogs [1]. Spam messages cause great damage to our infrastructures: They wastefully consume network resources, unnecessarily burden the server systems of target media, such as blog servers, and degrade the accuracy of search results. According to a report, spam emails cause damage worths 39 billion euros worldwide. It is therefore essential to develop methods that can detect and remove spam messages automatically.

A large number of automatic spam detection methods have been developed. In the early stages of the history of spam detection, rule-based methods were used, such as the compilation blacklists of IP addresses. Since then, machine learning-based methods have been used, e.g., a Bayesian method in [2]. In general, these methods learn probabilistic distributions of features from given training examples and then judge a new coming email on whether it is *spam* or non-spam, say *ham*, using these learned distributions. Since it is too costly to create the necessary training data, unsupervised methods were proposed [3,4,5]. However, to circumvent detection through these methods, spammers created spam messages in more sophisticated ways, such as image spams and word salads.

Spam detection methods are categorized into two groups: non-content-based methods and content-based ones. A method based on a blacklist is a typical example of the former. A Bayesian method outlined in [2] is a content-based method. Algorithms in [3,4] find characteristic substrings in spam messages and so they are also content-based method. The algorithm in [5] judges some emails as spam if the number of similar emails is larger than a given threshold value. The algorithm in [6] estimates language models of spam and ham messages. Hyperlinks form part of content, and algorithms in [7,8] make full use of their link structures.

In this paper, we study efficient content-based spam detection methods that can work with a large collection of input documents. We assume that spam documents are generated as copies of seed documents allowing some random operations [9] by reflecting the observation that the aim of spammers is to obtain large rewards with little effort and hence they have to create a large number of copies.

The main contribution of this paper is to propose an unsupervised algorithm for spam detection. The basic idea is to measure the *cost* of each document d relative to an input document collection D by an entropy-like measure $L(d \mid \theta_D)$, called *document complexity*. We expect that if documents are normal (ham) then its generation costs are sufficiently high. On the other hand, if they are spam document generated from seed documents, then their document complexity is quite low. The model parameter θ_D is computed by the suffix tree data structure, and the document complexity is estimated by an extension of a string probability model called MO method proposed by Jagadish *et al.* [12] based on the *leave-one-out suffix trees*.

However, a straightforward method based on the original MO takes almost quadratic time in the total input size N. To overcome this difficulty, by extending MO, we develop an efficient algorithm DCE (Document Complexity Estimation) that computes the document complexity for all documents d together with all leave-one-out suffix trees in in linear time in N. The keys of the algorithm is full exploitation of the suffix tree and suffix links.

We ran experiments on the real data from popular bulletin boards and synthetic data. For the real data, the results of our algorithm are comparable to the previous unsupervised algorithm in [4]. Experimental results on the synthetic data showed that our algorithm particularly works well for those spams such as *word salad spams* based on random replacement of inconsecutive regions.

The rest of this paper is organized as follows. Section 2 reviews basic notions. Section 3 gives our spam detection algorithm DCE. Section 4 reports experimental results. Section 5 concludes this paper.

2 Preliminaries

2.1 Strings

Let $\mathsf{N} = \{0, 1, 2, \ldots\}$ and let Σ be a finite *alphabet*. We denote by Σ^* the set of all finite *strings* over Σ. Then, the *length* of x is denoted by $|x| = n$. The *empty string* is the string of length zero and denoted by ε. If $s = a_1 \cdots a_n \in \Sigma^*$ $(a_i \in \Sigma)$ is a string of length $n = |s|$, then for every $1 \leq i \leq n$, the i-th letter of s is $s[i] = a_i$, and the *substring* from i to j is $s[i..j] = a_i \cdots a_j$ if $i \leq j$ and $s[i..j] = \varepsilon$

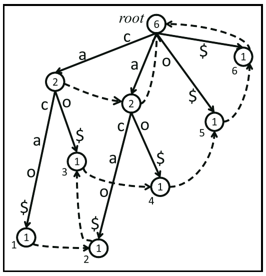


Fig. 1. The suffix tree for a string $s = cacao\$$

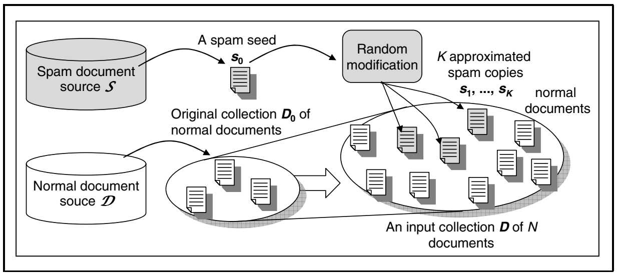

Spam document source $\mathcal{S}$
A spam seed s_0
Random modification
K approximated spam copies $s_1, \ldots, s_K$
normal documents
Original collection D_0 of normal documents
Normal document souce $\mathcal{D}$
An input collection D of N documents

Fig. 2. Blog and Bulletin Board Spam-Generation Process

otherwise. The *concatenation* of strings s and t is denoted by $s{\cdot}t$ or st. For a string s, if $s = xyz \in \Sigma^*$ for some strings $x, y, z \in \Sigma^*$ then x, y, and z are called, resp., a *prefix*, a *substring*, and a *suffix* of s. For a set $D = \{s_1, \ldots, s_M\} \subseteq \Sigma^*$ $(M \geq 0)$ of M strings, we define $|D| = M$ and $||D|| = \sum_{s \in D} |s|$. We define the sets of *all substrings* and *all suffices* of D by $Sub(D) = \{\ s[i..j] : s \in D, 1 \leq i \leq |s|\ \}$ and $Suf(D) = \{\ s[i..|s|] : s \in D, 1 \leq i \leq |s|\ \}$, resp.

2.2 Suffix Trees

Let $D = \{s_1, \ldots, s_M\}$ $(M \geq 0)$ be a set of M strings over Σ with total size $N = ||D||$. Then, the *suffix tree* for D, denoted by $ST(D)$, is a compacted trie $ST(D) = (V, E, root, suf, lab, fr)$ [10] for the sets of all suffices of the strings in D as shown in Fig. 1. Formally, the suffix tree for D is a rooted tree , where V is the vertex set, $E \subseteq V^2$ is the edge set, $suf : V \to V$ is a suffix link, $lab : E \to \Sigma^*$ is an edge-labeling, and $fr : V \to \mathsf{N}$ is a frequency counter. All the out-going edges leaving from a vertex always start with mutually different letters. Each vertex $v \in V$ represents the unique substring $\alpha = [v] \in Sub(D)$, where $[v]$ is the concatenation of the labels on the unique path from the root to v. For every internal vertex $[c\alpha]$ starting with a symbol $c \in \Sigma$, there always exists a suffix link from $[c\alpha]$ to the unique vertex $suf([c\alpha]) = [\alpha]$. Finally, the leaves of $ST(D)$ represent the set $Suf(D)$, and thus, $ST(D)$ stores $Sub(D)$. In Fig. 1, we show the suffix tree for a string $s = cacao\$$, where solid and broken lines represent the edges and the suffix links, respectively. Numbers in vertices represent their frequency counters. We note that \$ is the *end-marker* which does not occur in $s[1..|s|-1]$, and then $ST(s)$ strictly has $|s|$ leaves.

$ST(D)$ has at most $2N - 1$ vertices since the common prefix of paths can be shared [10]. Ukkonen [10] presented an elegant algorithm that build $ST(D)$ in $O(N)$ time by traversing the tree using suffix links. We augment each vertex $v = [\alpha]$ with the counter $fr(v) \in \mathsf{N}$, which keeps the frequency of α as a substring in strings of D. We can show that $fr([\alpha])$ equals the number of leaves that are descendants of $[\alpha]$.

2.3 Blog and Bulletin Board Spam-Generation Process

We focus on a specific type of spams called *blog* and *bulletin board spams*. Fig. 2 shows an outline of a spam-generating mechanism for this type of spam that we assumed in this paper. We next assume a large collection D_0 consisting of normal documents randomly drawn from the document source $\mathcal{D}$. In typical situations, normal documents are posted by human users.

On the other hand, to generate spam documents, we first randomly draw a document s_0, called a *spam seed*, from the source $\mathcal{S}$ for spam documents. For some appropriately chosen number $K \geq 1$, we then generate K approximate copies $s_1, \ldots, s_K$ of the original seed s_0 by applying the following perturbations to s_0 that are common to blog spams and bulletin board: **Mutation:** Changing randomly selected letters in a document. **Crossover:** Exchanging randomly selected parts of a pair of documents. **Word Salad:** Replacing a set of randomly selected words in a sentence with randomly selected words drawn from other sentences to have a grammatically correct, meaningless sentence.

Then, we add these K generated spam documents $s_1, \ldots, s_K$ to the original document set D_0. Repeating this process for different spam seeds, we create a mixture of many normal and some spam documents, called an *collection of input documents* $D = (d_1, \ldots, d_n)$, $n \geq 0$. A document $d \in D$ is *dirty* if it is a spam and *clean* if it belongs to the original collection D_0.

Although we have described a hypothetical spam-generation process, we note that our detection algorithm in Section 3 is *adaptive* and thus does not need to know a priori the characteristics of probabilistic document sources $\mathcal{D}$ and $\mathcal{S}$, classes of modifications, or the number K of approximate copies per seed.

2.4 The Spam Detection Problem

We consider the following problem: given an input collection D of normal and spam documents, to classify all documents into clean or dirty. Our goal is to devise a mapping $f : D \to \{1, 0\}$, called a *decision function*, which makes classification, where values 1 and 0 indicates the states that d is dirty and d not, respectively. We measure the *performance* of f on D by some cost functions. Let $N = |D|$ and let M_+ (or M_-) be the number of detected spam (normal) documents, and N_+ (N_-,) be the total number of spam (normal) documents in D. Then, the *recall* is $R = M_+/N_+$ and the *precision* is $P = M_+/(M_+ + M_-)$. The *F-score* is defined by $F = 2PR/(P + R)$.

3 The Proposed Spam Detection Method

In this section, we describe our proposed method for blog and bulletin board spam detection, called *DCE* (Document Complexity Estimation).

3.1 Outline of Our Decision Method

In our framework, each *document* is represented as a *sequence of letters* over an alphabet Σ rather than a *bag-of-words*. Our method is parameterized by a

```
Algorithm DCE1($D$):
Input: An input collection $D \subseteq \mathcal{D}$ of $M$ documents, a threshold $\gamma$.
Output: A set $A \subseteq D$ of dirty documents in $D$.
1: $A := \emptyset$;
2: foreach $d \in D$ do
3:     $D' := D \backslash \{d\}$;
4:     Build a model $\theta' = \theta_{D'} \in \Theta$ for $D'$ by minimizing $L(D' \mid \theta')$;
5:     Estimate $L := L(d \mid \theta') \simeq \lceil -\log Q(d \mid \theta') \rceil$;
6:     if $L/|d| \leq \gamma$ then
7:         $A := A \cup \{d\}$; //$d$ is detected as a spam.
8: end for
9: return $A$;
```

Fig. 3. An outline of our spam detection algorithm

given probabilistic model $\mathcal{M}$ for probability distribution over Σ^*. We assume a *probabilistic model* $\mathcal{M} = \{ Q(\cdot \mid \theta) : \theta \in \Theta \}$, which is a family of probability distributions over strings $Q(\cdot \mid \theta) : \Sigma^* \to [0, 1]$ with parameters θ drawn from some parameter space Θ. A parameter θ can be a description of a non-parametric model, e.g., Markov chains, as well as a parametric model.

In Fig. 3, we show the algorithm DCE1, which is an algorithmic schema of our method DCE. The basic idea of our method is as follows. Let $D \subseteq \Sigma^*$ be an input collection(or a *sample*) of M documents . If a given document $d \in D$ is *dirty* then d is a copy of some spam seed $s_0 \in \mathcal{S}$ that has a certain number of copies in D. In this case, we expect that d is statistically similar to some portion of D, and thus, the contribution of d to the whole of D will be small. Let $d \in D$ be any target document in the sample D. If we model D by some probability distribution $Q(\cdot \mid \theta)$ for some $\theta \in \Theta$, then we can encode D in an encoding of the code length

$$L(D \mid \theta) \simeq \lceil -\log Q(D \mid \theta) \rceil,$$

so that best compression can be achieved [11]. In what follows, we denote by θ_D this best parameter θ for a given collection D within Θ.

On the other hand, we suppose that the sample D of M documents is obtained from the collection $D' = D \backslash \{d\}$ of $M - 1$ documents by adding the document d. D' can again be encoded by some $\theta' = \theta_{D'} \in \Theta$ in length $L(D' \mid \theta')$. Then, the contribution of d can be measured by how much this addition of d increases the code length of the sample D, which is bounded above by

$$\begin{aligned} \Delta L(D, d) &\overset{\text{def}}{=} L(D' \cup \{d\} \mid \theta) - L(D' \mid \theta') \\ &= L(D \mid \theta) - L(D' \mid \theta') \ \leq \ L(d \mid \theta'), \end{aligned}$$

where $\theta = \theta_D$ and $\theta' = \theta_{D'}$. In the above equations, the inequality in the last row follows from the following observation: A description of D can be obtained by appending to the description for D' of length $L(D' \mid \theta')$ an encoding for d of length $L(d \mid \theta')$ for d conditioned by $D' = D \backslash \{d\}$. Hence, we have an inequality $L(D \mid \theta) \leq L(D' \mid \theta') + L(d \mid \theta')$.

Algorithm MO:
Input: Any data structure $Sub(D)$ for a set $D \subseteq \Sigma^*$ of input strings, frequency counter $fr : Sub(D) \to \mathsf{N}$, and a string $s \in \Sigma^*$ of length L.
Output: an estimated probability of $Q(s|\theta_D)$.

```
(i)   Let $\gamma_0 = \varepsilon$ be the context, $fr(\varepsilon) = ||D||$, $\theta_D = (Sub(D), fr)$, and $i := 2$.
(ii)  While $s \neq \varepsilon$ holds, we repeat the following process: Find the unique longest substring $\gamma_i \in Sub(D)$ that maximizes $\gamma_{i-1} \otimes \gamma_i$ such that $\gamma_i \otimes s \neq \varepsilon$. If there are more than two substrings with the same overlap, then take the longer one. In case that there exists no such $\gamma_i$, set $\alpha_i = \varepsilon$, $\beta_i = \gamma_i$ to be the initial letter of $s$, $P(\beta_i \,|\, \alpha_i) = 1/||D||$. Let $\alpha_i = \gamma_{i-1} \otimes \gamma_i$ and $\beta_i = \gamma_i \otimes s$. Remove the overlap $\gamma_i \otimes s$ from $s$. Define $P(\beta_i \mid \alpha_i) = fr(\alpha\beta_i)/fr(\alpha_i)$.
(iii) Set $m = i$. Return $Q(s \mid \theta_D) = P(\beta_1) \prod_{i=2}^{m} P(\beta_i \mid \alpha_i)$.
```

Fig. 4. The original MO algorithm for estimating the probability $Q(s \mid \theta_D)$ [12]

We expect that if the target document d has more copies in D then the contribution $\Delta L(D, d)$ will be smaller and thus more likely to be spam. Then, $\Delta L(D, d)$ can be estimated by the upperbound $\Delta L(D, d) \simeq L(d \mid \theta') \simeq \lceil -\log Q(d \mid \theta') \rceil$. This is our decision procedure for spam detection:

$$f(d) = \begin{cases} 1 & \text{if } \lceil -\log Q(d \mid \theta') \rceil / |d| \leq \gamma \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

where $\gamma > 0$ is a threshold parameter, which will be given in the following section. This gives the algorithm DCE1 in Fig. 3.

3.2 Probability Estimation for Strings by MO Method

To model $Q(d \,|\, \theta_D)$ for a sample D, we use the *MO* (Maximal Overlap) method, introduced by Jagadish *et al.* [12] and built on top of an index structure based on the suffix tree [10].

The MO method estimates the probability $P(x)$ of a long string $x \in \Sigma^*$ based on a set of shorter substrings appearing in the original string in D as follows. The *conditional probability* for a string α, given string β, is given by $P(\beta \,|\, \alpha) = P(\alpha\beta)/P(\beta)$. Let $s \in \Sigma^*$ be a *target* string to estimate $Q(s \,|\, \theta_D)$. In general, for any model $Q(\cdot \,|\, \theta)$, we can write the probability of the string $s = \beta_1 \cdots \beta_m$ ($m \geq 1$) where $s = \beta_1, \ldots, \beta_m \in \Sigma^*$, as $P(s) = P(\beta_1) \prod_{i=1}^{m} P(\beta_i \mid \beta_1 \cdots \beta_{i-1})$ For each $i = 1, \ldots, m$, the MO method approximates the conditional probability $P(\beta_i \mid \beta_{i-1} \cdots \beta_1)$ by the conditional probability $P(\beta_i \mid \alpha_i)$ with the unique longest suffix α_i of $\beta_1 \cdots \beta_{i-1}$, called the *longest context*, such that $\alpha_i\beta_i$ appears in D. For substrings u and v of an input string s, we define the *maximal overlap*, denoted by $u \otimes v$, as the maximal string $w \in \Sigma^*$ that is both a suffix of u and a prefix of v.

Recall that $Sub(D)$ is the set of strings that appears as substrings in D. In Fig. 4, we show the original MO method that computes an estimation of

```
Algorithm LinearMO:
Input: a string $s$, the suffix tree $ST(D)$for a set of strings $D$.
Output: an estimated probability of $Q(s|\theta_D)$.
 1: $v :=$ the root of $ST(D)$;
 2: for $i := 1, \dots, \ell = |s|$ do
 3:    $x_i := s[i]$;
 4:    while $v$ has no out-going edge $(v, w)$ labeled with $x_i$ do
 5:       if $(v = root)$ then $Q := Q \cdot (1/N)$ with $N = ||D||$;
 6:       else $v := suf(v)$ by following the suffix link;
 7:    end while
 8:    $Q := Q \cdot fr(w)/fr(v)$;  /* $\alpha_i = str(v)$ and $P(x_i|\alpha_i) = fr(w)/fr(v)$ */
 9:    $v := w$ by following the edge;
10: end for
11: return $Q$;  /* estimation of $Q(s|\theta_D)$ */
```

Fig. 5. A linear time MO-score estimation algorithm

$P(s)$ in the greedy way. For every $i = 1, \dots, |s|$ ($|s| \geq 1$), MO finds maximally overlapping substrings $\gamma_i = \alpha_i\beta_i \in Sub(\{s\})$ such that $\gamma_i \in Sub(D)$ and $\beta_i \neq \varepsilon$. Then, we define the associated probabilities by $P(\beta_i \mid \alpha_i) = fr(\beta_i\alpha_i)/fr(\beta_i)$. If there exists no such a γ_i then the next letter $s[k] \in \Sigma$ has never appeared in D, where $k = |\beta_1 \cdots \beta_{i-1}|$. Then in this zero-probability case, we set $\alpha_i = \varepsilon$, $\gamma_i = \beta_i$ to be the un-seen letter $s[k] \in \Sigma$, and define $P(\beta_i \mid \alpha_i) = 1/N$, where $N = ||D||$ is the total letter frequency. Note that $s = \beta_1 \cdots \beta_{|}s|$. Finally, we compute the estimated probability by $Q(s \mid \theta_D) = P(\beta_1)\prod_{i=2}^{|s|} P(\beta_i \mid \alpha_i)$.

Jagadish *et al.* [12] show that MO is computable in $O(\ell q)$ time for given the suffix tree $ST(D)$, where $\ell = |s|$ and q is the depth of $ST(D)$. That is, MO takes $O(\ell^2)$ time in worst case. They also show that MO outperforms the previous method KVI [13], which uses the non-overlapping decomposition of the input string in estimation.

3.3 Efficient Computation of MO by a Suffix Tree

In Fig. 6, we show the outline of algorithm DCE2 that runs in $O(N)$ to detect all spam candidates based on DCE, where $N = ||D||$ is the total length of documents. The crucial part of the original algorithm DCE1 in Fig. 3 is the suffix tree construction phase $\theta_{D\setminus\{d\}}$ at Line 4 and the MO-score estimation phase for $-\log Q(d \mid \theta_{D\setminus\{d\}})$ at Line 5 for each value of $d \in D$. This is a time-comsuming process since it iteratively builds the suffix tree $ST(D\setminus\{d\})$ M times for all except the current document. A straightforward implementation of DCE1 requires $O(MN + M\ell^2) \simeq O(MN + N\ell) = O(MN)$ time, where $M = |D|$, $N = ||D||$, and $\ell = \max_{d\in D} |d|$.

Speed-up of MO estimatoin. Fig. 5 shows a linear-time algorithm LinearMO for estimating $Q(s|\theta_D)$ using $ST(D)$ that quickly find the longest context β_i. The

```
Algorithm DCE2($D$):
Input: An input collection $D \subseteq \mathcal{D}$, a threshold $\gamma > 0$.
Output: A set $A \subseteq D$ of dirty documents in $D$.
1: Construct the suffix tree $ST(D)$ for the whole document $D$;
2: $A := \emptyset$;
3: foreach $d \in D$ do
4:     Simulate $Q :=$ LinearMO$(d, ST(D \backslash \{d\}))$ by running copies of LinearMO
       simultaneously on $ST(D)$ and $ST(\{d\})$;
5:     $L := -\log Q$;
6:     if $L/|d| \leq \gamma$ then $A := A \cup \{d\}$; /* $d$ is detected as a spam */
7: end for
8: return $A$;
```

Fig. 6. An outline of our spam detection algorithm

algorithm quickly finds the longest context α_i for each factor β_i by traversing the suffix tree using the suffix links via a technique similar to [10]. By the next lemma, LinearMO improves computation time of Line 4 of DCE1 from $O(M\ell^2)$ time to $O(M\ell) \simeq O(N)$ time.

Lemma 1. *Let Σ be an alphabet of constant size. The algorithm* LinearMO *in Fig 5 correctly implements the algorithm* MO *in Fig 4 to estimate $Q(s \,|\, \theta_D)$, and runs in $O(\ell)$ time and $O(N)$ space, where $N = ||D||$ is the total size of the sample D and $\ell = |s|$ is the length of the string s.*

Proof. Let $\gamma_i = \alpha_i\beta_i$ $(m \geq 1, 1 \leq i \leq m)$ be the maximally overlapping substrings computed in the original MO. Assume that $\beta_j = x_h \cdots x_i$ $(h \leq i)$ with $k = |\beta_i|$. Then, we can show that $P(\beta_i \,|\, \alpha_i) = P(x_h \,|\, \alpha_i)P(x_{h+1} \,|\, \alpha_i x_h) \cdots P(x_i \,|\, \alpha_i x_h \cdots x_{i-1})$ since the longest context of x_j in s relative to $ST(D)$ is $\alpha x_h \cdots x_{j-1}$ for each $h \leq j \leq i$. Therefore, the correctness follows. The time complexity is derived from arguments similar to [10]. □

Speed-up of suffix tree construction. In the building phase at Line 3 of Algorithm DCE1, a straightforward implementation takes $O(MN + M\ell^2)$ time by building the suffix tree $ST(D\backslash\{d\})$, called a *leave-one-out suffix tree* for D, for each document $d \in D$. Instead of this, we simulate traversal of the leave-one-out suffix tree $ST(D\backslash\{d\})$ directly on $ST(D)$ for D and $ST(\{d\})$. A *virtual leave-one-out suffix tree* for D and d is a suffix tree $\widetilde{ST}(D \setminus \{d\}) = (V', E', root', suf', lab', fr')$ defined by $V' = \{[\alpha] : \alpha \in \Sigma, fr_D(\alpha) \geq 1, fr_d(\alpha) \geq 1\}$ and $E' = \{([\alpha],[\beta]) \in V' \times V' : ([\alpha],[\beta]) \in E_D\}$. We define the frequency of a vertex $[\alpha]$ by $fr'([\alpha]) = fr_D([\alpha]) - fr_d([\alpha])$.

Lemma 2. *Let $D \subseteq \Sigma^*$ and $d \in \Sigma^*$ be any collection and a document. Suppose that $d \in D$. Then, $\widetilde{ST}(D\backslash\{d\})$ is isomorphic to the original $ST(D\backslash\{d\})$.*

Lemma 3. *Let $D \subseteq \Sigma^*$ be any sample. For any document $d \in D$ of length ℓ, we can simulate the algorithm* LinearMO *in Fig 5 over the virtual suffix tree*

$\widetilde{ST}(D\backslash\{d\})$ by runnning the copies of LinearMO *simultaneously over $ST(D)$ and $ST(\{d\})$ by the above rule. The obtained algorithm runs in $O(\ell)$ time with pre-process $O(||D|| + \ell)$ for constructing $ST(D)$ and $ST(\{d\})$.*

Proof. The algorithm has a pair (v_D, v_d) of two pointers $v_D \in V_D$ and $v_d \in V_d$, respectively, on vertices of $ST(D)$ and $ST(\{d\})$. When the algorithm receives the next letter $c = s[i]$, it moves from a combined state (v_D, v_d) to (v'_D, v'_d) such that $v'_D = [str_D(v_D) \cdot c]$ and $v'_d = [str_d(v_d) \cdot c]$ if $fr_D(v'_D) - fr_d(v'_d) \geq 1$ hold. For the zero-probability case (at line 5), we set $N := ||D|| - |d|$. To see that the moves are always well-defined, we define the pair for v by $\pi(\alpha) = (\mathbf{sgn}\, fr_D(\alpha), \mathbf{sgn}\, fr_d(\alpha)) \in \{+, 0\} \times \{+, 0\}$ for any vertex $v = [\alpha]$ $(\alpha \in \Sigma^*)$. Since $d \in D$ and $\alpha \in Sub(d)$ hold by assumption, Then, we can show that $(+, +)$ is only possible state for $\pi(\alpha)$. Hence,the result immediately follows. □

From Lemma 1 and Lemma 3, we show the main theorem of this paper.

Theorem 1. *The algorithm* DCE2 *in Fig 6 runs in $O(N)$ time and $O(N)$ space, where $N = ||D||$ is the total size of D.*

3.4 Determining the Threshold

Our unsupervised spam detection method DCE2 in Fig. 6 takes a decision threshold $\gamma > 0$ as a learning parameter. To determine an appropriate value of γ, we first draw the histogram of the normalized document complexity L over unlabeled examples only, and then adaptively define γ to be the point that takes the minimal value in an appropriate range, say, $0.0 \sim 1.0$ (bit).

To justify this selection of γ, we show in Fig. 7 the histograms of the normalized document complexity L over all documents in a test collection YF (described later). In this experiment, documents are manually classified into three classes, the spam, the normal, and the whole document set to see the validity of the above selection method. In the figure, we see that the distribution for spam documents (spams) is concentrated at the values $L = 0.0 \sim 0.2$ (bit) per letter, while the normal documents (hams) spread at $L = 1.0 \sim 3.5$ (bit) per letter obeys a bell-shaped distribution with the peak around 2.2 (bit). Thus, it is a straightforward strategy to minimize classification risk by taking γ to be the point that takes the minimal value. This justifies the choice of γ.

4 Experimental Results

4.1 Dataset

We used a test collection of forums from Yahoo Japan Finance[1], collected by Narisawa *et al.* [4]. The collection consists of four sections of forum data: YF4314, YF4974, YF6830, and YF8473. All posts from each forum's data are labeled if they are spam. Table 1 shows the details of them. In the following experiments, we used only the body texts of documents, including the HTML tags.

[1] http://quote.yahoo.co.jp

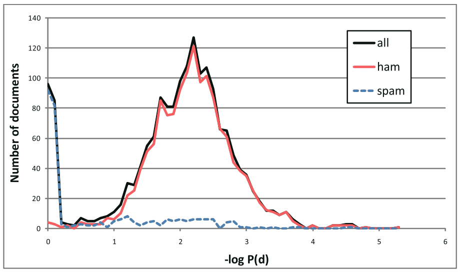


Fig. 7. Histogram of the document complexity of Web data

Table 1. Details of the datasetsD

Dataset	# of ham	# of spam	Total length
YF4314	291	1424	184,775
YF4974	331	1315	211,505
YF6830	317	1613	252,324
YF8473	264	1597	239,756

4.2 Method

We implemented the proposed method DCE and the following methods.

Naive method: Let D be the input document set. Then, the Naive method regards a document d as spam if d is a substring of another document in D. That is, Naive is a type of copy-detection method.

Alienness measure: Narisawa *et al.* [4] propose a spam detection method which uses the *representatives* of substring classes, which are characteristic strings extracted from the document set. There are three measures for representatives, called *Length*, *Size*, and *Maximin*, and we call the methods with them the AM_Len, the AM_Siz, and the AM_Max, respectively. They also propose a method for determining the threshold for their measures. Their methods then regard a document as spam if it contains a representative such as alien, which is an outlier of substrings in nomal documents.

4.3 Results

Exp. 1: Basic Performance. Table 2 shows the Recalls, the Precisions, and the F-scores for all methods. As shown in the table, DCE and AM_Siz achieved high Recall, DCE and Naive achieved high Precision, and DCE achieved the highest F-score in all the datasets. Overall, DCE shows slightly better performance compared to other methods.

Exp. 2: Recall and Precision. In this experiment, we did not use the methods to determine thresholds of each method. Instead, we output all documents as

Table 2. Performance of spam detection methods

Forum	Method	Recall	Precision	F-score
4314	DCE	0.59	**0.95**	**0.73**
	AM_Len	0.62	0.72	0.67
	AM_Siz	**0.82**	0.50	0.62
	AM_Max	0.68	0.62	0.65
	Naive	0.54	0.94	0.68
4974	DCE	**0.63**	0.69	**0.66**
	AM_Len	0.35	0.70	0.47
	AM_Siz	**0.63**	0.69	**0.66**
	AM_Max	0.57	**0.71**	0.63
	Naive	0.52	0.66	0.58
6830	DCE	**0.68**	0.67	**0.67**
	AM_Len	0.41	0.66	0.51
	AM_Siz	0.75	0.58	0.65
	AM_Max	0.67	0.67	**0.67**
	Naive	0.54	**0.71**	0.62
8473	DCE	0.63	0.69	**0.66**
	AM_Len	0.53	0.72	0.61
	AM_Siz	**0.70**	0.61	0.65
	AM_Max	0.67	0.66	**0.66**
	Naive	0.54	**0.79**	0.64

spam by ascending order of its document complexity and computed Recalls and Precisions. The graph in Fig. 8 shows the Recall and the Precision curves of the dataset YF6830. To the left of the intersection of the Precision and the Recall of DCE, Precision and Recall are clearly higher than for any other methods. However, to the right of the intersection, Precision decreases faster than for the other methods.

Exp. 3: Performance Against Noise Density. We created *noise spam* by using a seed as follows: For each position $1 \leq i \leq |d|$ of d, we replaced $d[i]$ by $d[j]$ with probability p, where $1 \leq j \leq |d|$ is a random number. We called the probability the *density* of noise. To examine the effect of the density, we set the number of spams to 20. Figure 9 shows Recall rates against density. Although DCE has higher Recall than Naive, its performance is insufficient at a higher density.

Exp. 4: Performance Against the Number of Spams. In Fig. 10, we show the performance against the number of inserted spams. We used the density $p = 0.02$. According to the increase in the number of inserted spams, the performance of DCE showed improvement. However, Naive did not improve.

Exp. 5: Detecting Word Salads. *Word salad* is a new type of spam that is difficult to detect. Spammers create word salads by replacing words in a *skeleton* document with some interesting *keywords*. In this experiment, we created word salads and inserted them into the document set. The dataset used was YF4974. We created the keyword set by manually selecting from the dataset and selected a spam as the skeleton in YF8473. The keyword set consisted of 15 nouns, and the skeleton consisted of 45 words. All nouns in the skeleton were replaced in the creation of the word salads. We then created these word salads and insert them into the dataset. Figure 11 shows the result. AM_Siz detected about 20% of word salads despite only a few word salads being inserted. However, the Recall increased up to a final figure of only 30%. On the other hand, DCE detected approximately 100% of word salads when the number of word salads inserted was greater than 30.

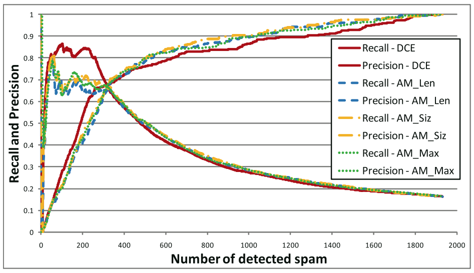


Fig. 8. Recall and Precision curve

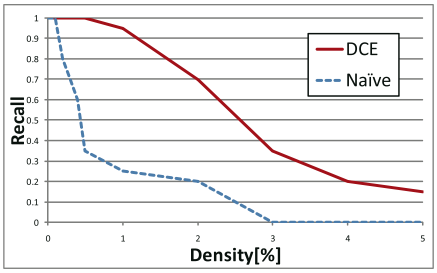


Fig. 9. Recall vs. density

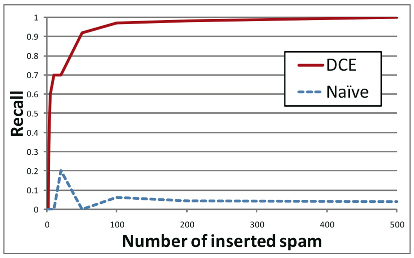


Fig. 10. Recall vs. number of inserted noise spams

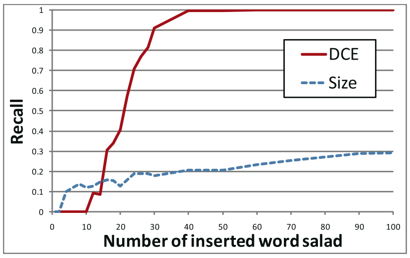


Fig. 11. Recall vs. number of inserted word salads

5 Conclusion

In this paper, we have proposed an unsupervised spam detection algorithm that calculates document complexity. The algorithm runs in linear time w.r.t. the total length of the input documents. We also conducted experiments using both real and synthetic data, where the real data was collected from popular bulletin boards and the synthetic data was generated as word salad spam documents.

References

1. Kolari, P., Java, A., Finin, T.: Characterizing the splogosphere. In: Proc. WWE 2006 (2006)
2. Graham, P.: A Plan for Spam (2002), `http://paulgraham.com/spam.html`
3. Narisawa, K., Yamada, Y., Ikeda, D., Takeda, M.: Detecting Blog Spams using the Vocabulary Size of All Substrings in Their Copies. In: Proc. WWE 2006 (2006)
4. Narisawa, K., Bannai, H., Hatano, K., Takeda, M.: Unsupervised spam detection based on string alienness measures. In: Corruble, V., Takeda, M., Suzuki, E. (eds.) DS 2007. LNCS (LNAI), vol. 4755, pp. 161–172. Springer, Heidelberg (2007)
5. Yoshida, K., Adachi, F., Washio, T., Motoda, H., Homma, T., Nakashima, A., Fujikawa, H., Yamazaki, K.: Density-based spam detector. In: Proc. ACM KDD 2004, pp. 486–493 (2004)

6. Mishne, G., Carmel, D., Lempel, R.: Blocking Blog Spam with Language Model Disagreement. In: Proc. AIRWeb 2005 (2005)
7. Gyöngyi, Z., Garcia-Molina, H., Pedersen, J.: Combating Web Spam with TrustRank. In: Proc. VLDB 2004, pp. 576–587 (2004)
8. Benczúr, A.A., Csalogány, K., Tamás Sarlós, M.U.: SpamRank – Fully Automatic Link Spam Detection. In: Proc. AIRWeb 2005 (2005)
9. Bratko, A., Filipič, B., Cormack, G.V., Lynam, T.R., Zupan, B.: Spam filtering using statistical data compression models. JMLR 7, 2673–2698 (2006)
10. Ukkonen, E.: On-line construction of suffix-trees. Algorithmica 14(3), 249–260 (1995)
11. Cover, T.M., Thomas, J.A.: Elements of Information Theory, 2nd edn. Wiley, Chichester (1938)
12. Jagadish, H.V., Ng, R.T., Srivastava, D.: Substring selectivity estimation. In: Proc. PODS 1999, pp. 249–260 (1999)
13. Krishnan, P., Vitter, J.S., Iyer, B.: Estimating alphanumeric selectivity in the presence of wildcards. In: Proc. SIGMOD 1996, pp. 282–293 (1996)

A Probabilistic Neighbourhood Translation Approach for Non-standard Text Categorisation

Ata Kabán

School of Computer Science, The University of Birmingham,
Birmingham, B15 2TT, UK
A.Kaban@cs.bham.ac.uk

Abstract. The need for non-standard text categorisation, i.e. based on some subtle criterion other than topics, may arise in various circumstances. In this study, we consider written responses to a standardised psychometric test for determining the personality trait of human subjects. A number of state-of-the-art text classifiers that have been very successful in standard topic-based classification problems turn out to perform poorly in this task. Here we propose a very simple probabilistic approach, which is able to achieve accurate predictions, and demonstrates this peculiar problem is still solvable by simple statistical text representation means. We then extend this approach to include a latent variable, in order to obtain additional explanatory information beyond a black-box prediction.

1 Introduction

Automatic text classification has been a highly researched topic over the past decade and many successful methods have been devised. However, in benchmark test beds, the notion of class is most often associated with that of a topic. Vector-space representations [10] of text collections tend to be well separable w.r.t. topics, and so methods like linear SVM and Naive Bayes [14,1] typically obtain high accuracy.

However, there are cases when text categorisation needs to be based on some non-standard and more subtle criteria, other than topics. For example, in this work, we deal with responses of human subjects to a standardised psychometric test, for determining their personality traits. All subjects take the same test, and the data consists of their written English prose responses to the same set of questions. The topics are therefore common to all documents. Instead, in this case, the classification criterion of interest is the personality trait of each subject. The question is this: Is it possible to automatically predict the personality trait of human subjects based on their written responses and a set of hand-labelled examples?

Several methods that have been highly successful on topic based text classification turn out to perform poorly on this task. One may even worry that perhaps a bag-of-words representation is inappropriate and perhaps a more sophisticated representation, involving syntactic and semantic characteristics, and / or natural language processing approaches might be required. Naturally, searching for another feature representation in an infinite space of possibilities is not a practical prospect. Therefore, in this work we set out to investigate whether a statistical approach would still bear any fruit.

We devise a fairly simple probabilistic approach that works on standard bag-of-words features, and has a natural and clear probabilistic formulation. On some inspection, our model has some resemblance with both tf-idf and nearest-neighbour classification, and for the latter, we have chosen to call it nearest neighbour translation. Despite its simplicity, our method is able to deal with peculiar non-linear separation boundaries and, based on the data set tested, it appears to be highly suited to the non-standard text classification task under study. We then extend this approach to include a latent variable in order to obtain some explanatory information in addition to black-box prediction.

In the remainder of the paper, Section 2 introduces our method. Section 3 presents its latent variable modelling extension, including an iterative procedure for parameter estimation. Section 4 presents comparative prediction results, as well as interpretability results. Finally, Section 5 concludes the paper.

2 Probabilistic Neighbourhood Translation

Consider a training set of N text documents, together with their associated labels. Each document will be represented as a discrete distribution over all the terms of the dictionary. For document n, this is denoted by $P(.|n)$. Similarly, each term is a distribution over all the documents of the corpus. For term t, this is denoted by $P(.|t)$.

Using the above representation, we define the probabilistic neighbourhood of document n by marginalising over the terms, as the following:

$$P(n'|n) = \sum_{t \in \text{Dictionary}} P(n'|t)P(t|n) \tag{1}$$

Document n' is a neighbour of document n with probability $P(n'|n)$. In general, all documents are neighbours of all other documents with some (possibly zero) probability.

Further, let us denote the label of document n by $P(z|n)$. This is a vector of length C, where C is the number of classes. For the training set instances, the labels are known, so $P(z_{true}|n)$ is just a 1-of-C label encoding — i.e. $P(z = c|n) = 1$ if document n belongs to class c and zero otherwise.

Now, we can write the label probability distribution of a previously unseen document, as the following:

$$\begin{aligned}
P(z|n_{new}) &= \sum_{n \in \text{TrainingSet}} P(z|n)P(n|n_{new}) && (2)\\
&= \sum_{n=1}^{N} P(z|n) \sum_{t=1}^{T} P(n|t)P(t|n_{new}) && (3)\\
&= \sum_{t=1}^{T} P(z|t)P(t|n_{new}) && (4)
\end{aligned}$$

where $P(t|n_{new})$ is the term distribution representation of the new document, and all other quantities are pre-computed from the training set.

The training procedure then only requires us to prepare the probability matrix $P(n|t)$ by appropriately normalising the documents-by-terms matrix, and $P(z|n)$ are the given label assignments. Moreover, the marginalisation over training documents can also be pre-computed, as in (4), which essentially results in each term being assigned a label probability. Observe in addition, the obtained label probabilities $P(z|n_{new})$ are guaranteed to be properly normalised to sum to one w.r.t. z, without any further effort. We may threshold this label probability to obtain a hard partitioning of the data into classes, but at the same time the actual value of the predicted label probability tells us about the confidence of the prediction. A probability close to 0 or 1 indicates a highly confident prediction, whereas, in two-class problems, a value close to 0.5 predicts a low confidence of the class prediction. Thus we have set up a non-parametric and fully probabilistic predictor, capable of making class predictions as well as uncertainty estimates.

To make some connection with related methods, let us inspect the formulations (1)-(4) and note a resemblance with the popular 'term frequency inverse document frequency' (tf-idf) method [10]. In (3), we have $P(t|n)$ exactly the term-frequencies (tf), and an analogy between the term $P(n|t)$ and inverse document frequencies may also be seen, since $P(n|t) = \#(t,n)/\sum_n \#(t,n)$ is inversely proportional to the frequency of term t throughout the entire corpus. Contrary to tf-idf however, our formulation has a very clear probabilistic foundation.

Secondly, from the form (2), we can see an analogy to a nearest neighbour model, with a neighbourhood kernel defined by (1). Because of this analogy, and since the neighbourhood kernel is a stochastic translation matrix, and the entire model is formulated in probability terms, we call this approach ' probabilistic nearest neighbour translation'. The probabilistic nature of our formulation makes it fairly straightforward to incorporate extensions as appropriate and the next section demonstrates this with the purpose of allowing us to gain, besides prediction, some additional explanatory information from the data.

3 An Extension for Uncovering Term-Associations

Though the simple approach presented in the previous section can be readily used to perform classification, i.e. to predict class labels for previously unseen data, one would often prefer to be able to map text documents from the space of words into a more conceptual space. In the present case, this conceptual space would be necessarily other than a topical space, for it being defined by some non-standard labelling that we try to accommodate. Technically, this would be useful e.g. in the cases when documents happen to have no overlapping words, despite they share content w.r.t. to the given labelling — and in addition it may provide additional insights in terms of interpretability. To achieve this, we extend our model by introducing a latent 'bottleneck' variable having K different discrete values, $k = 1, ..., K$, and aim for a version of eq. (2) where the space of terms is replaced with the space of our new latent variable:

$$P(z|n_{new}) = \sum_{n=1}^{N} P(z|n) \sum_{k=1}^{K} P(n|k)P(k|n_{new}) \tag{5}$$

To make the connection to the actual words space, we write:

$$P(n|k) = \sum_{t=1}^{T} P(n|t)P(t|k); \quad \text{and} \quad P(k|n_{new}) = \sum_{t'=1}^{T} P(k|t')P(t'|n_{new}) \tag{6}$$

We can think of this extension as having added a probabilistic translation of both terms and documents into the latent space. The use of a 'bottleneck' latent variable is quite common in generative latent variable modelling of text, especially in unsupervised learning [6,11]. Here in turn, we employ this technique as part of our conditional model, in the supervised learning context.

Now, replacing into (5) we have:

$$P(z|n_{new}) = \sum_{n=1}^{N} P(z|n) \sum_{t=1}^{T} P(n|t) \sum_{k=1}^{K} P(t|k) \sum_{t'=1}^{T} P(k|t')P(t'|n_{new}) \tag{7}$$

and we may additionally also interpret the part $\sum_k P(t|k)P(k|t')$ as a compressed (aggregated) term association probability matrix $P(t|t')$ — somewhat in the spirit of [11] — which we are going to estimate.

3.1 Parameter Estimation by Maximising the Leave-One-Out Error

For the ease of the subsequent manipulations, let us introduce the following notations for elements precomputed from the training set: $\boldsymbol{U}_{-n} \in \mathbb{R}^{C \times T}$ with elements $P(z|t) = \sum_{n' \neq n} P(z|n')P(n'|t)$, $\boldsymbol{v}_n \in \mathbb{R}^{T \times 1}$ with elements $P(t|n)$, and $\boldsymbol{y}_n \in \mathbb{R}^{K \times 1}$ with elements $P(z_{true}|n), \forall n \in$ TrainingSet. The unknown parameters will be denoted by $\boldsymbol{A}_1 \in \mathbb{R}^{T \times K}$ with elements $P(t|k)$ and $\boldsymbol{A}_2 \in \mathbb{R}^{K \times T}$ with elements $P(k|t')$ respectively. Using these notations, the r.h.s. of the model (7) may now be written compactly as $\boldsymbol{U}_{-n_{new}} \boldsymbol{A}_1 \boldsymbol{A}_2 \boldsymbol{v}_{n_{new}}$.

Now, to estimate the parameters $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$, we maximise the *leave-one-out error*, using the training set. The idea of maximising the leave-one-out error was previously proposed in [5] for a predictive k-nearest-neighbour model termed the 'neighbourhood component analysis'. Our approach in this section may be seen to have some analogies with that model, though an obvious difference is that our parameters are all probabilities, which leads to different (and simpler) estimation equations and allows straightforward interpretation in the context of our text analysis application.

Thus, we formulate the objective to minimise the sum of Kullback-Leibler divergences between the given labels $\boldsymbol{y}$ and their predictions[1],

$$\{\boldsymbol{A}_1, \boldsymbol{A}_2\} = \underset{\boldsymbol{A}_1, \boldsymbol{A}_2}{\operatorname{argmin}} \sum_{n=1}^{N} KL(\boldsymbol{y}_n || \boldsymbol{U}_{-n} \boldsymbol{A}_1 \boldsymbol{A}_2 \boldsymbol{v}_n) \tag{8}$$

which, up to an additive constant (i.e. the sum of entropies of $\boldsymbol{y}_n$), is equivalent to maximising the following objective:

$$Obj(\boldsymbol{A}_1, \boldsymbol{A}_2) = \sum_n \boldsymbol{y}_n \log \{\boldsymbol{U}_{-n} \boldsymbol{A}_1 \boldsymbol{A}_2 \boldsymbol{v}_n\} \tag{9}$$

We add Lagrange multipliers to ensure that all columns of both $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ are proper probabilities i.e. sum up to one.

After straightforward algebra, the stationary equations will have the form of fixed point non-linear equations that can be solved iteratively. These iterative updates are the following:

$$\boldsymbol{A}_1^{new} \propto \boldsymbol{A}_1^{old} \odot \sum_n \boldsymbol{U}_{-n}^T \frac{\boldsymbol{y}_n}{\boldsymbol{U}_{-n} \boldsymbol{A}_1^{old} \boldsymbol{A}_2 \boldsymbol{v}_n} (\boldsymbol{A}_2 \boldsymbol{v}_n)^T \tag{10}$$

$$\boldsymbol{A}_2^{new} \propto \boldsymbol{A}_2^{old} \odot \sum_n (\boldsymbol{U}_{-n} \boldsymbol{A}_1)^T \frac{\boldsymbol{y}_n}{\boldsymbol{U}_{-n} \boldsymbol{A}_1 \boldsymbol{A}_2^{old} \boldsymbol{v}_n} \boldsymbol{v}_n^T \tag{11}$$

where $\propto$ denotes proportionality, $\odot$ stands for element-wise multiplication and the division above also operates element-wise.

The algorithm is then to alternate the above updates to convergence. Converge to a local optimum of the objective (9) is guaranteed, for similar arguments as in E-M algorithms.

A remaining issue is to estimate the optimal dimensionality of the latent space, i.e. K. This may be done e.g. by cross-validation. However, in our experiments a wide range of K yielded very similar results, therefore for the next section we decided to use a $K = 3$, for best serving interpretability. The reason is, we are then able to use $p(k|n)$ for visualising the text corpus. In addition, we may also use $P(t|k)$ to inspect the latent concepts inferred, as a result of the word associations captured, in the form of ordered lists of representative words. Finally, since the neighbourhood kernel is now parametrised, visualising those is also likely to be meaningful.

4 Results

4.1 The Data

The input to our two algorithms consist of responses of 669 human subjects to a psychometric test that encompasses four standardised questions, the same for

[1] The KL-divergence is a natural choice for measuring the dissimilarity between two probability distributions. Other divergence functions may also be employed alternatively.

all subjects tested. The actual questions are not part of the data. Further, the training data was hand-labelled by domain experts and comprised 281 dominant and 388 submissive examples[2].

For this study, we assemble the four pieces of texts produced by each subject into a subject-specific text document, so each document will have four paragraphs. Topically, all of these are necessarily strongly overlapping, since the topical subject has been set and fixed through the use of the same set of four questions within the standardised test. The aim is then to investigate the possibility of automatically deciding the 'dominant' versus 'submissive' personality traits of the subjects on the basis of their written answers.

First, we perform a standard preprocessing the raw text to produce a term-document frequency based bag-of-words representation, using the Rainbow toolkit [8]. In this process, we kept all words that occurred more than once, and we switched off both stemming and stop-word removal, because unlike in topic-based discrimination, the use of these language features may well contribute to expressing one or the other of the personality traits.

This preprocessing resulted in a dictionary size of 4030 distinct terms. We did not attempt to correct for spelling mistakes, so misspelled versions of the same word occasionally coexist as distinct words in the dictionary.

4.2 Illustration

As already mentioned in Sec. 2, the basic version of our method is a lazy-learner, i.e. it does not require any effort for training. The latent variable extension (Sec. 3) in turn, requires an iterative algorithm for parameter estimation at the training stage. The evolution of the objective function Eq (9) through the iterations is shown in Figure 1. As expected, we observe a monotonic increase to convergence.

Fig. 2 further illustrates the working of the method, showing the probabilistic label predictions for each document, in a leave-one-out experiment, in comparison to the true labels. The accurate matching is quite apparent.

4.3 Personality Trait Prediction Results

To measure the performance of the automated categorisation methods that we investigate, we create 100 independent random splits of the data into 320 training examples and 349 examples set aside for testing. We measure the classification accuracy in terms of the percentage of correct predictions, as well as in terms of area under the Receiver Operating Characteristic curve (AUC) [4]. The results are presented in Table 1.

In the table, pNN refers to the approach presented in Section 2 and pNN-aggr3 is the extension detailed in Section 3, where $K = 3$ was set. We see, both variants of our method produced highly accurate predictions. The results

[2] The data was anonymised and kindly provided for research purposes by Dr Marco de Boni, Unilever Ltd.

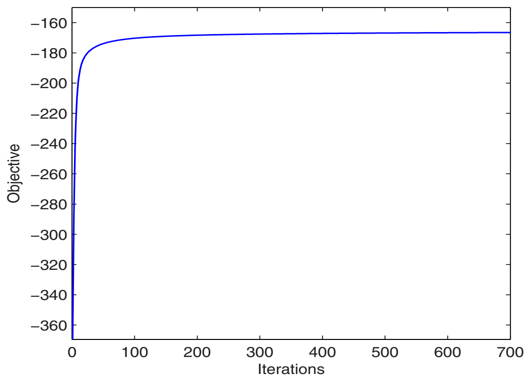


Fig. 1. Evolution of the objective function Eq (9) through iterations

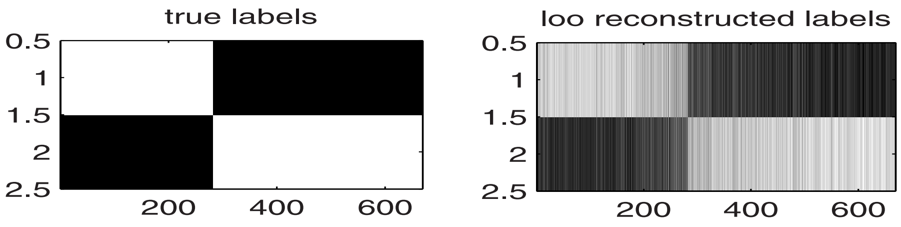


Fig. 2. Left: the true labels. Right: Leave-one-out predictions.

of these two methods are comparable to each other, there is a very slight (and statistically insignificant) performance sacrificed for interpretability in the latter method. We see the AUC values are high and even closer, which reflects the methods' ability to also produce good uncertainty estimates.

In turn, in the remainder of the table we see results obtained on the same data, by some of the most successful existing text classification methods, which all turned out to perform quite poorly on this non-standard classification task. We found it useless to give the AUC values for these — SVM does not give probabilistic predictions or uncertainty estimates, and for the other three methods the probabilistic predictions were close to 0 or 1 in all cases. It is clear already from the 0-1 errors that these methods are barely above a random guessing (though the difference from random guessing was found statistically significant in all cases).

In these comparisons, the multinomial Naive Bayes (implemented in Rainbow) may be considered as a baseline for its simplicity, nevertheless its previous good performance in topic-based classification has been quite remarkable (see e.g. [9]). The Dirichlet Compound Multinomial is a fairly recent enhancement on Naive Bayes [2], endowed with an ability to model word burstiness in the language, which offers a more realistic word distribution than the multinomial model. Despite its remarkable previous success, it does not excel in the non-standard categorisation task investigated here. The linear SVM has been among the best

Table 1. Classification results over 100 random splits into 320 subjects for training and 349 subjects for testing (mean ± standard deviation)

Method	% classification accuracy	AUC
pNN	0.9822 ± 0.0120	0.9996 ± 0.0004
pNN-aggr3	0.9722 ± 0.0119	0.9977 ± 0.0015
Multinomial Naive-Bayes	0.6002 ± 0.0152	
Dirichlet Compound Multinomial	0.6098 ± 0.0082	
linear-SVM	0.6183 ± 0.0131	
sparse (L1) Logistic Regression	0.5977 ± 0.0682	

performing and most popular text classifiers [7]. It has an in-built capability of avoiding overfitting in the presence of large numbers of relevant features (words). However, from the results obtained, its limitation in this peculiar problem is most apparent[3] Next, we tested the possibility that perhaps the poor performance in our task is because too many words may be irrelevant to the target. We employed the sparse logistic regression, which again has been previously found to have a state-of-the-art performance for text categorisation [3]. We used the efficient implementation available in [13], which can deal with high dimensional data. However, the results obtained in this problem have been again not much better than random.

4.4 Discussion

It is not trivial to generalise and trace ultimate conclusions from these results, nevertheless, we are able to answer the most concerning of the questions. Namely, it is certainly the case, for this data set, that a statistical approach using just word frequency information (without any more sophisticated NLP technique) is still capable to predict the classes of interest. Thus, the automated prediction of personality traits from psychometric tests seems feasible to a reasonably high degree of accuracy.

As for the large difference in performance between our simple approach versus several of the previously most successful methods, we conjecture at this point this may be because — unlike in topic-based classification, where topic classes tend to be well separated — here we may encounter peculiar non-linear boundaries between the representatives of the two personality traits in the data space. Our rather simple approach, with its kNN flavour is able to deal with this successfully. Further investigations using other, non-linear classifiers may shed more light on this issue. However, one advantage of our approach, besides of being extremely simple and computationally inexpensive, is its probabilistically clear foundation. This enables extensions in a straightforward manner, as we have already seen in Section 3. Rather than employing a universal black-box predictor and having to

[3] The result given in the table is with the C parameter optimised using an internal leave-one-out validation. However, we have not experimented with other kernels so far.

search for the appropriate non-linearity to suit the problem at hand, we have a natural way of defining similarities and optimising the leave-one-out prediction error directly, in order to gain further explanatory information in support of a subsequent interpretation of the results.

4.5 Interpretability

Naturally, the extended (parametrised) version of our method is computationally more demanding than the basic variant. Let us inspect therefore the additional information that it provides. Figure 3 shows the visualisation of the text collection in the coordinate basis defined by the estimated parameters $P(k|n)$. Contrary to any unsupervised or topic-based visualisation method, by our model construction, the groupings are necessarily w.r.t. the particular label specification. The colours and markers reflect the true labels for the ease of visual evaluation. Indeed the two fairly distinct clouds of points have a good correspondence to the true personality trait labels. Beyond just label predictions, such a visualisation may have the benefit of revealing the more detailed topological proximity / similarity relationships between the subjects w.r.t. the categorisation studied, beyond the hard partitioning into disjoint classes.

Further, we may inspect the three latent variables through their associated lists of probable words. As discussed earlier, these are unlikely to be topical aspects, but instead, aspects that define the imposed non-topical grouping of the documents. In the present case, these will be correlated lists of words whose usage characterises the two personality traits. The top end of these lists are shown in Table 2.

To find out how these factors relate to the known classes, we look at the probabilities $P(z|k) = \sum_n P(z|n)P(n|k)$. In this case, we find a probability very close to 1 for the 'submissive' class in the case of the aspects $k = 1$ and $k = 2$, and very close to 1 for the 'dominant' class in the case of $k = 3$. So the interpretation of the above lists of words is clear. We should note though that, in

Table 2. The ordered lists of the most probable words associated with the three possible values of the latent variable in our model

ran 0.020; peoples 0.019; ends 0.018; excersise 0.017; discretion 0.016; diseased 0.016; encourages 0.015; motivate 0.014; becasue 0.013; appeal 0.012; achoice 0.012; questioning 0.012; steadied 0.011; epecially 0.011; turns 0.011; puffing 0.011; energy 0.011; baulk 0.010; expression 0.010; ...
throats 0.030; ran 0.029; tieing 0.028; lung 0.026; speak 0.026; encourages 0.023; ready 0.023; wanna 0.021; atmospheres 0.018; wee 0.018; pregnant 0.018; previous 0.017; attitude 0.016; greed 0.016; circumstances 0.016; achoice 0.015; garden 0.015; epecially 0.015; closely 0.015; ...
countries 0.146; home 0.109; easily 0.103; restrictions 0.050; burger 0.050; hours 0.045; things 0.044; health 0.036; social 0.031; longer 0.029; clubs 0.027; matter 0.024; unlike 0.023; arrival 0.021; media 0.019; bombarded 0.017; boys 0.016; public 0.013; sec 0.012; ...

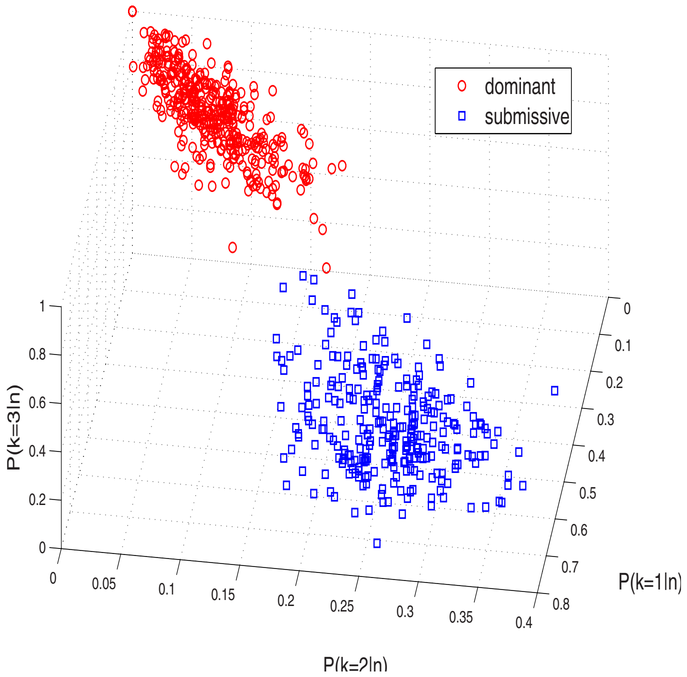


Fig. 3. Visualisation of the text collection in the coordinate basis defined by $P(k|n)$. Each points represents one subject. The superimposed true labels represented by the markers have a good correspondence with the shape of the density.

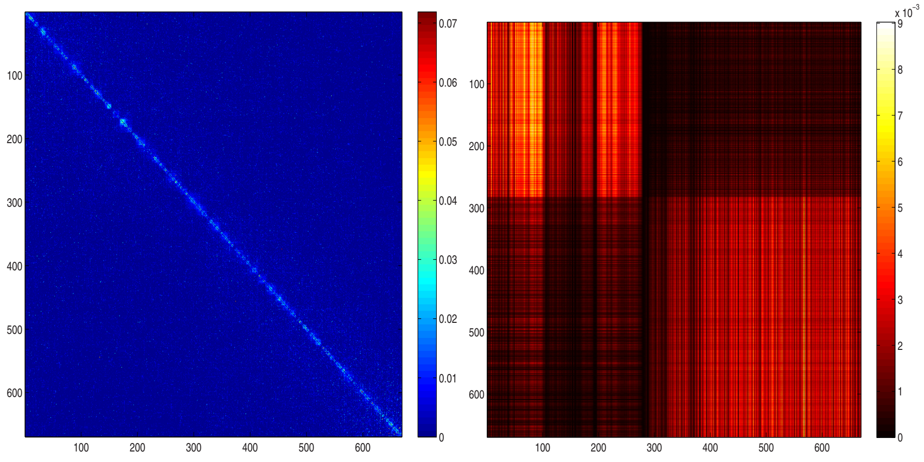


Fig. 4. The obtained neighbourhood probabilities (kernels) $p(n|n')$ in pNN and the 3D aggregated version of pNN

general, these estimated 'conceptual aspects' need not be exclusively tied with one of the classes but they may be shared by all classes with a certain probability $P(z|k)$ that we can calculate.

Finally, to conclude the discussion on parameter interpretability, Figure 4 shows the neighbourhood kernels as obtained by our kNN and aggregated kNN respectively, highlighting the advantage of the latter in terms of bringing out the hidden structure from the data. Since in the aggregated version the neighbourhood probabilities become a function of the additional latent variable and its associated parameters, this allows us essentially to learn the similarities implied by the given labelling. This is most apparent from the right-hand plot, where we clearly see the 2-class structure in the aggregated kNN neighbourhood. By contrary, this global structure is not readily seen in the simpler kNN neighbourhood (left-hand plot), as this model extracts the predictive information in a 'local', i.e. instance-specific manner without distilling any global structural information.

5 Conclusions

We presented a novel probabilistic approach for text categorisation and text analysis for automating the prediction of personality traits on the basis of standardised psychometric tests taken by human subjects. This is a non-topical, non-standard text categorisation problem, which presents difficulties to a number of state of the art text classifiers. Through our approach, we demonstrated that such a peculiar task can still be successfully automated within a simple statistical approach using a standard word frequency representation of documents. Further work may consider more data sets of this kind to test the so far well-performing method further. In addition, other existing or novel classifiers could be included in the investigation to further pin down the reasons that make some methods more suitable than others for this problem.

Acknowledgement

Thanks to Dr Marco de Boni from Unilever Ltd. for sharing the data and the problem.

References

1. Colas, F., Brazdil, P.: Comparison of SVM and Some Other Classification Algorithms in Text Classification Tasks. Artificial Intelligence in Theory and Practice 217, 169–178 (2006)
2. Madsen, R.E., Kauchak, D., Elkan, C.: Modeling Word Burstiness Using the Dirichlet Distribution. In: Proceedings of the Twenty-Second International Conference on Machine Learning (2005)
3. Eyheramendy, S., Genkin, A., Ju, W.-H., Lewis, D.D., Madigan, D.: Sparse Bayesian Classifiers for Text Categorization. Technical Report, Department of Statistics, Rutgers University (2003)

4. Fawcett, T.: ROC graphs: Notes and practical considerations for researchers, Technical report, HP Laboratories, MS 1143, 1501 Page Mill Road, Palo Alto CA 94304, USA (April 2004)
5. Goldberger, J., Roweis, S., Hinton, G., Salakhutdinov, R.: Neighbourhood Component Analysis. In: Neural Information Processing Systems (NIPS 2004) 17, pp. 513–520 (2004)
6. Hofmann, T.: Probabilistic Latent Semantic Analysis. In: Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI 1999) (1999)
7. Joachims, T.: Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In: Proceedings of the European Conference on Machine Learning (1998)
8. McCallum, A.K.: Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering (1996), www.cs.cmu.edu/~mccallum/bow
9. Mitchell, T.: Machine Learning, ch. 6. McGraw Hill, New York (1997)
10. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Communications of the ACM 18, 613–620 (1975)
11. Saul, L., Pereira, F.: Aggregate Markov Models for statistical language processing. In: Proc. of the Second Conference on Empirical Methods in Natural Language Processing, pp. 81–89 (1997)
12. Sebastiani, F.: Machine learning in automated text categorization. ACM Computing Surveys 34, 1–47 (2002)
13. Shevade, S.K., Keerthi, S.S.: A Simple and Efficient Algorithm for Gene Selection using Sparse Logistic Regression, Technical Report No. CD-02-22, Control Division, Department of Mechanical Engineering, National University of Singapore, Singapore - 117 576 (2002)
14. Yang, Y.: An Evaluation of Statistical Approaches to Text Categorization. Information Retrieval 1(1/2), 69–90 (1999)

Author Index