LNBI 3916

Jinyan Li
Qiang Yang
Ah-Hwee Tan (Eds.)

Data Mining for Biomedical Applications

PAKDD 2006 Workshop, BioDM 2006
Singapore, April 2006
Proceedings

Springer

Jinyan Li Qiang Yang Ah-Hwee Tan (Eds.)

Data Mining for Biomedical Applications

PAKDD 2006 Workshop, BioDM 2006
Singapore, April 9, 2006
Proceedings

Springer

Series Editors

Sorin Istrail, Brown University, Providence, RI, USA
Pavel Pevzner, University of California, San Diego, CA, USA
Michael Waterman, University of Southern California, Los Angeles, CA, USA

Volume Editors

Jinyan Li
Institute for Infocomm Research
21 Heng Mui Keng Terrace, Singapore 119613
E-mail: jinyan@i2r.a-star.edu.sg

Qiang Yang
Hong Kong University of Science and Technology, Department of Computer Science
Clearwater Bay, Kowloon, Hong Kong, China
E-mail: qyang@cs.ust.hk

Ah-Hwee Tan
Nanyang Technological University, School of Computer Engineering
Nanyang Avenue, Singapore 639789
E-mail: asahtan@ntu.edu.sg

Library of Congress Control Number: 2006922190

CR Subject Classification (1998): H.2.8, J.3, I.2, H.3, I.5, F.1

LNCS Sublibrary: SL 8 – Bioinformatics

ISSN	0302-9743
ISBN-10	3-540-33104-2 Springer Berlin Heidelberg New York
ISBN-13	978-3-540-33104-9 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11691730 06/3142 5 4 3 2 1 0

Preface

This edited volume contains the papers selected for presentation at the First Workshop on Data Mining for Biomedical Applications (BioDM 2006) held in Singapore on April 9, 2006. The workshop was held in conjunction with the 10th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2006), a leading international conference in the areas of data mining and knowledge discovery. The aim of this workshop was to provide a forum for discussing research topics related to biomedical applications where data mining techniques were found to be necessary and/or useful.

BioDM 2006 received a total of 35 full-length paper submissions from seven countries. Each submitted paper was rigorously reviewed by three Program Committee members. Although many papers were worthy of publication, only 14 regular papers can be accepted in the workshop for presentation and publication in this volume. The accepted papers were organized into three sessions according to their topics, with four papers on database & search, four papers on bio data clustering, and six papers on in-silico diagnosis. The distribution of the paper topics indicated that database query, search, similarity measure, feature selection, and supervised learning remained the current research issues in the field. In addition to the contributed presentation, the BioDM 2006 workshop featured a keynote talk delivered by Limsoon Wong, who shared his insightful vision on the bioinformatics research problems related to protein–protein interactions.

This workshop would not have been possible without the help of many colleagues. We would like to thank the Program Committee members for their invaluable review and comments. Given the extremely tight review schedule, their effort to complete the review reports before the deadline was greatly appreciated. In addition, we found some reviewers' comments were really excellent, as good as what is usually found in a survey paper—critical, constructive, and comprehensive. These comments were very helpful for us in selecting the papers.

Very importantly, we would like to acknowledge the PAKDD 2006 Conference Chair Lim Ee Peng for coordinating with the publisher. Without his effort, these proceedings may not have been published in time for the workshop. We also thank Elaine Koh and Chen Ling for their effort and time in workshop registration and website maintenance.

Thank you all and may the papers collected in the volume inspire your thoughts and research.

April 2006

Jinyan Li
Qiang Yang
Ah-Hwee Tan

Organization

BioDM 2006 workshop was organized by the School of Computer Engineering, Nanyang Technological Univeristy, and Institute for Infocomm Research in conjunction with PAKDD 2006.

Organizing Chair

Ah-Hwee Tan	Nanyang Technological University, Singapore

Program Co-chairs

Jinyan Li	Institute for Infocomm Research, Singapore
Qiang Yang	Hong Kong University of Science and Technology

Program Committee

Keun Ho Ryu	Chungbuk National University, Korea
Kenji Satou	JAIST, Japan
Kenta Nakai	University of Tokyo, Japan
Lusheng Wang	City University of Hong Kong, Hong Kong, China
Qiang Yang	Hong Kong University of Science and Technology, Hong Kong, China
Jingchu Luo	Peking University, China
Juan Liu	Wuhan University, China
Jagath C. Rajapakse	Nanyang Technological University, Singapore
Chee Keong Kwoh	Nanyang Technological University, Singapore
See-Kiong Ng	Institute for Infocomm Research, Singapore
Jinyan Li	Institute for Infocomm Research, Singapore
Vladimir Brusic	University of Queensland, Australia
Yi-Ping Phoebe Chen	Deakin University, Australia
Geoff McLachlan	University of Queensland, Australia
Jian Pei	Simon Fraser Univeristy, Canada
Jianbo Gao	University of Florida, USA
Alexander Statnikov	Vanderbilt University Medical Center, USA
Aik Choon Tan	Johns Hopkins University, USA
Mohammed Zaki	Rensselaer Polytechnic Institute, USA

Table of Contents

In-silico Diagnosis

Exploiting Indirect Neighbours and Topological Weight to Predict Protein Function from Protein-Protein Interactions

Hon Nian Chua, Wing-Kin Sung, and Limsoon Wong

School of Computing and Graduate School for Integrated Sciences and Engineering, National University of Singapore,
3 Science Drive 2, Singapore 117543
{g0306417, dcsswk, dcswls}@nus.edu.sg

Abstract. Most approaches in predicting protein function from protein-protein interaction data utilize the observation that a protein often share functions with proteins that interacts with it (its level-1 neighbours). However, proteins that interact with the same proteins (i.e. level-2 neighbours) may also have a greater likelihood of sharing similar physical or biochemical characteristics. We speculate that two separate forms of functional association accounts for such a phenomenon, and a protein is likely to share functions with its level-1 and/or level-2 neighbours. We are interested to find out how significant is functional association between level-2 neighbours and how they can be exploited for protein function prediction.

We made a statistical study on recent interaction data and observed that functional association between level-2 neighbours is clearly observable. A substantial number of proteins are observed to share functions with level-2 neighbours but not with level-1 neighbours. We develop an algorithm that predicts the functions of a protein in two steps: (1) assign a weight to each of its level-1 and level-2 neighbours by estimating its functional similarity with the protein using the local topology of the interaction network as well as the reliability of experimental sources; (2) scoring each function based on its weighted frequency in these neighbours. Using leave-one-out cross validation, we compare the performance of our method against that of several other existing approaches and show that our method performs well.

A Database Search Algorithm for Identification of Peptides with Multiple Charges Using Tandem Mass Spectrometry

Kang Ning, Ket Fah Chong, and Hon Wai Leong

Department of Computer Science, National University of Singapore,
3 Science Drive 2, Singapore 117543
{ningkang, chongket, leonghw}@comp.nus.edu.sg

Abstract. Peptide sequencing using tandem mass spectrometry is the process of interpreting the peptide sequence from a given mass spectrum. Peptide sequencing is an important but challenging problem in bioinformatics. The advancement in mass spectrometry machines has yielded great amount of high quality spectra data, but the methods to analyze these spectra to get peptide sequences are still accurate. There are two types of peptide sequencing methods –database search methods and the de novo methods. Much progress has been made, but the accuracy and efficiency of these methods are not satisfactory and improvements are urgently needed. In this paper, we will introduce a database search algorithm for sequencing of peptides using tandem mass spectrometry. This Peptide Sequence Pattern (PSP) algorithm first generates the peptide sequence patterns (PSPs) by connecting the strong tags with mass differences. Then a linear time database search process is used to search for candidate peptide sequences by PSPs, and the candidate peptide sequences are then scored by share peaks count. The PSP algorithm is designed for peptide sequencing from spectra with multiple charges, but it is also applicable for singly charged spectra. Experiments have shown that our algorithm can obtain better sequencing results than current database search algorithms for many multiply charged spectra, and comparative results for singly charged spectra against other algorithms.

1 Introduction

As the volume of MS/MS mass spectra grows, the accompanying algorithmic technology for automatically interpreting these spectra has to keep pace. An increasingly urgent problem is the interpretation of multi-charge spectra – MS/MS spectra with charge 3, 4, and 5 are available from the publicly accessible GPM (Global Proteome Machine) dataset [1]. It is foreseen that increasingly there will be more multi-charge spectra produced and so the problem of accurate interpretation of these spectra will become more important with time.

Most existing algorithms for peptide sequencing have been focused largely on interpreting spectra of charge 1. Even when dealing with multiply-charged spectrum, they assume each peak is of charge 1. Only a few algorithms take into account or explicitly make known that they taken into account spectra with charge 2 or higher [2-4].

Database searching algorithms [5-8] rely primarily on the completeness of databases, and the availability of a good scoring mechanism. Traditional database search methods have the common principle as this: the experimental spectrum is compared with the theoretical spectrum for each of the peptide in the database, and the peptide from the database with best match usually provides the sequence of the experimental peptide.

The most widely used database search algorithms for analyzing mass spectra of peptides has been software such as SEQUEST [5] and MASCOT [9]. These algorithms search a sequence database for peptides sequences which would produce ions of the mass observed for a particular spectrum, then score these candidate sequences against the observed spectrum. The best match between the peptide tandem mass spectrum and the database-derived peptide sequences is made via a combination of an ion intensity-based score plus a cross-correlation routine. The problems with these algorithms are that they only considered the ions of the mass observed for a particular spectrum, so they can work well for peptide sequences already in the database, but perform badly for peptides with post-translational modifications or other variations.

It is well known that it is almost impossible to find a peptide sequence that *matches exactly* (100% match) with an entry in the database. Instead, many methods rely on matching much shorter sequences called *tags* [6, 10]. However, for some of them [6], the simple assumptions limit the identification accuracy.

In [6], the authors use tag sequence for the search of the peptide sequence. A fragmentation spectrum usually contains a short, easily identifiable series of sequence ions, which yields a partial sequence. This partial sequence divides the peptide into three parts-regions 1, 2, and 3-characterized by the added mass m_1 of region 1, the partial sequence of region 2, and the added mass m_3 of region 3. The construct, *m_1 partial sequence m_3*, is called a "peptide sequence tag" and it is a highly specific identifier of the peptide. An algorithm then uses the sequence tag to find the peptide in a sequence database. The main problem of this approach is that the model used in this algorithm is too simple. A 3-segment peptide sequence tag is used, but not enough to capture several highly-confident fragment sequences. The database search may return several candidates peptide sequences, but further discriminations are very limited.

Because of these problems of the current database search algorithms, it is ideal if we can appropriately utilize all of (or as much as possible) the subsequences (tags) information in the spectrum, and find out the peptide sequence in the database, or detect the post-translational modification that has most support. Recently, the InsPecT algorithm has been developed by Tanner etc. [10], which use more tags information for database search. This algorithm has used score function similar to Dancik score [11] to generate highly reliable tags from spectrum graph, extend tags and use trie to search for candidate peptides in database, and evaluate candidate peptides by statistical analysis. Another database search algorithm based on a set of tags is SPIDER [12].

We have developed a new database algorithm that extend the idea of using tags [6], and we have concentrated on the multi-charge spectrum data. We have tried to utilize all of the tags information, and tried to get the best results based on this information. In our algorithm, we first find out some strong tags from the spectrum, and connect them by their mass differences; these tag-mass combinations are called patterns of the peptide sequence, and the peptide sequences in the database that best match the patterns are selected. This *peptide sequence pattern (PSP)* gives more flexibility and accuracy to the algorithm, especially for the multiply charged spectra that are very hard to interpret. Then a linear time database search process is used to search candidate peptides sequences by PSPs. These candidate peptide sequences are then scored by share peaks count, ranked and output.

In the following part, we will introduce our formulation of the problem and the database search algorithm. We will then describe our experiment settings and analysis of the results in details.

2 Problem Formulation of Multi-charge Peptide Sequencing

Consider an MS/MS spectrum for a peptide sequence $\rho = (a_1a_2...a_n)$ where a_j is the jth amino acid in the sequence. The *parent mass* of the peptide is given by $m(\rho) = M = \sum_{k=1}^{n} m(a_k)$. Consider a peptide fragment $\rho_k = (a_1a_2...a_k)$, for $k \leq n$ that has fragment mass $m(\rho_k) = \sum_{j=1}^{k} m(a_j)$. The peaks in the spectrum come from peptide fragmentation and each peak p can be characterized by the ion-type, specified by $(z, t, h) \in (\Delta_z \times \Delta_t \times \Delta_h)$, where z is the charge of the ion, t is the basic ion-type, and h is the neutral loss incurred by the ion. The set of ion-types considered is $\Delta = (\Delta_z \times \Delta_t \times \Delta_h)$, where $\Delta_z = \{1,2,...,\alpha\}$, $\Delta_t = \{a,b,y\}$, and $\Delta_h = \{\phi, -H_2O, -NH_3\}$. The (z, t, h)-*ion* of the peptide fragment ρ_k will produced an observed peak p in the spectrum S, that has a mass-to-charge ratio of $mz(p)$, that can be computed from the following formula [4]:

$$m(\rho_k) = mz(p) \cdot z + (\delta(t) + \delta(h)) + (z-1),$$

where δ(t) and δ(h) are the mass difference for the respective ion-type and neutral-loss, respectively. The *theoretical spectrum* for ρ is defined in [4] as the set $TS(\rho) = \{p : p$ is observed peak for (z, t, h)-*ion* of peptide fragment ρ_k, for all $(z, t, h) \in \Delta$ and $k=0,1,\ldots,n\}$ of all possible observed peaks for all ion types, and all possible fragments of ρ.

In peptide sequencing, we are given an experimental mass spectrum and the problem is to determine the sequence of the original peptide. A spectrum $S = \{p_1, p_2, \ldots, p_n\}$ of maximum charge α is a set of n peaks where each peak p_k is described by two parameters – $mz(p_k)$, the observed mass-to-charge ratio and $intensity(p_k)$, its intensity. To account for peaks that correspond to ions of charge 2, an "extension" process is performed to convert it to the equivalent peak of charge 1.

The *shared peaks count* between the experimental spectrum S and a peptide ρ is defined as the number of peaks in S that has the same mass as those in $TS(\rho)$, the theoretical spectrum of ρ.

We have followed the computational model in [4]. To account for the different ion-type (especially in multi-charge spectra), [4] introduced the concept of the ***extended spectrum*** S_β^α where α is the maximum charge of the spectrum S, and β is the largest charge considered for extension. In the extended spectrum S_β^α , we "extend" each peak by generating a set of pseudo-peaks (or guesses) that correspond to the different ion-types with charge $\leq \beta$. Namely, for each peak $p_j \in S$ and ion-type

$(z, t, h) \in (\{1,...,\beta\} \times \Delta_t \times \Delta_h)$, we generate pseudo-peak denoted by (p_j, (z, t, h)) with a corresponding *assumed* fragment mass given by
$m(p_j,(z,t,h)) = mz(p_j) \cdot z + (\delta(t) + \delta(h)) + (z-1)$. A corresponding ***extended spectrum graph*** of connectivity (defined below) d, $G_d(S_\beta^\alpha)$, is also introduced. Each vertex in this graph represents a pseudo-peak (p_j, (z, t, h)) in the extended spectrum S_β^α, namely to the (z, t, h)-*ions* for the peak p_j. For simplicity, we also denote the vertex by (v_j, (z, t, h)). Each vertex represents a possible peptide fragment mass $m(v_j, (z, t, h))$. An additional notion called the **PRM (*prefix residue mass*)** is also introduced. This mass refers to the prefix mass of the interpreted peptide fragment mass for vertex, and is defined as $PRM_v(v_i)$ = m(v_i) if t(v_i)∈{b-ion} else (a and y-ion) $PRM_v(v_i) = M -$ m(v_i).

In the "standard" spectrum graph, we have a directed edge (u, v) from vertex u to vertex v if $PRM(v)$ is larger than $PRM(u)$ by the mass of a single amino acid. In the extended spectrum graph of connectivity d, $G_d(S_\beta^\alpha)$, we extend the edge connectivity definition to mean "*a directed path of no more than d amino acids*". Thus, we connect vertex u and vertex v by a directed edge (u, v) if the $PRM(v)$ is larger than $PRM(u)$ by the total mass of d' amino acids, where $d' \le d$. In this case, we say that the edge (u,v) is connected by a path of length up to d amino edges. Note that the number of possible paths to be searched is 20^d and increased exponentially with d. In practice, we use d=2, except where it is explicitly stated otherwise. We illustrate the extended spectrum graph with an example shown in Fig. 1.

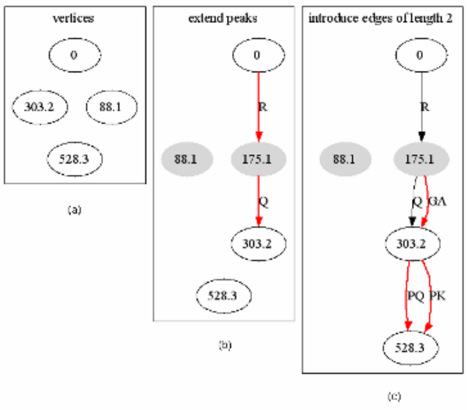


Fig. 1. The difference between $G_1(S_1^2)$ (left) and $G_2(S_2^2)$ (right). There are no paths from v_b to v_e in $G_1(S_1^2)$, but 4 paths in $G_2(S_2^2)$ due to extension.

3 Database Search Algorithm

In our algorithm, we first find out some strong tags from the spectrum, and connect them by their mass differences; these tag-mass combinations are called *Peptide*

Sequence Patterns (PSPs), and the peptide sequences in the database that best match the PSPs are selected for further process. These PSPs give more flexibility and accuracy to the algorithm. Our algorithm is called the PSP algorithm.

Peptide Sequence Patterns Algorithm

The PSP algorithm first compute a set, *BST*, of "best" strong tags. Informally, these strong tags are highly reliable tags found in the spectrum *S*. To find strong tags, we first restrict the possible ion-types those that appear most frequently. The restricted set of ion-type is given by $\Delta^R = (\Delta_z^R \times \Delta_t^R \times \Delta_h^R)$, where $\Delta_z^R = \{1\}$, $\Delta_t^R = \{b, y\}$, and $\Delta_h^R = \{\phi\}$. Namely, we consider only charge 1, an only *b-ions* and *y-ions* and no neutral loss. We also define $G_1(S_1^\alpha, \{\Delta^R\})$, the extended spectrum graph with ion-type restriction – namely, the spectrum graph $G_1(S_1^\alpha)$ where the ion types considered are restricted to those in Δ^R. Then a *strong tag T* of ion-type $(z, t, h) \in \Delta^R$ is a maximal path $\langle v_0, v_1, v_2, \ldots, v_r \rangle$ in the $G_1(S_1^\alpha, \{\Delta^R\})$, where every vertex $v_i \in T$ is of a (z, t, h)*-ion*. In each component of the graph, PSP algorithm computes a "best" tag with respect to some scoring function. Then the set *BST* is the set comprising the best tag for each component in the spectrum graph. Typically, the number of tags is much smaller than the number of peaks in *S*. (We refer the reader to [4] for more details.)

Given the set BST of best strong tag, the Peptide Sequence Patterns (PSPs) algorithm then proceeds to find the PSP that result from paths obtained by "connecting" the tags from *BST*. This is done by searching for paths in the graph *G*(*BST*) in which the vertices are the strong tags in *BST*, and we have an edge from the tail vertex *u* of T_1 to the head vertex *v* of T_2 if the *PRM*(*v*) is larger than *PRM*(*u*). We note two major difference between *G*(*BST*) and the extended spectrum graph – first, the number of vertices in *G*(*BST*) is small, and second, the number of edges is also very much smaller since we link strong tags in a head-to-tail manner.

The peptide sequence patterns (PSPs) that represent the paths compose of the tag fragments and mass fragments. Formally, $PSP_i = m_1t_1m_2t_2 \ldots m_nt_nm_{n+1}$, in which m_i and t_i refer to mass difference and tag, respectively. Each tag in the sequence composes of those consecutive amino acids with very high probability to be together. Each mass is the sequence represents the value of masses between tags.

After PSPs are retrieved, the PSPs are scored and ranked according to shared peaks count of the theoretical spectrum of the PSP and the experimental spectrum. Some top PSPs can be selected for database search.

The database search algorithm is essentially an approximation pattern matching in the database, with PSP (composed of tags and mass differences) as pattern. The detailed database search algorithm will be described later.

After database search based on PSPs, several candidate peptides are obtained. For each of candidate peptide sequences, score it by the shared peaks count of the theoretical spectrum of the candidate peptides and the experimental spectrum.

The scheme of the PSP algorithm and the description of the algorithm are illustrated in Fig. 2 and Fig. 3, respectively.

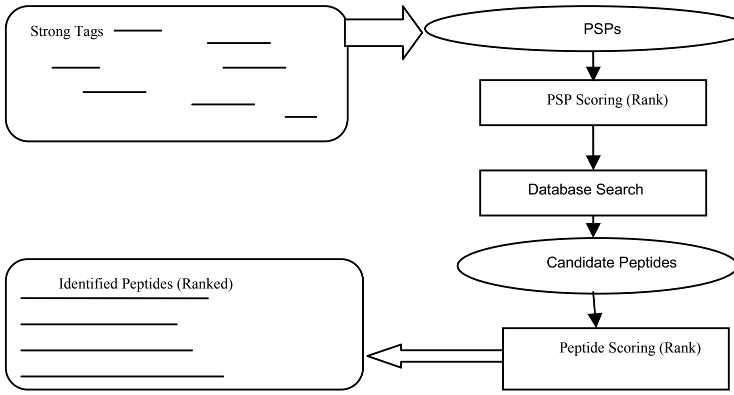


Fig. 2. The scheme of the database search algorithm

1. **Search for strong tags**
   - Transform spectrum to extended spectrum graph
   - Select all of the best strong tags (BST) in extended spectrum graph
2. **Generation of PSPs**
   - Connect BSTs by mass differences
   - Generate a graph G, every vertex is a BST, every edge is one mass difference. Starting and ending vertex represent 0 and parent masses, respectively
   - List all of the paths from start to end vertexes
   - For each of the path P_i, generate the peptide sequence pattern PSP_i
   - Score and rank PSPs by share peaks count score.
3. **Database search by PSP**
   - (details in later part)

Fig. 3. The description of the database search algorithm

Approximate Database Search Using PSP

The matched candidate peptides are searched in the database by PSP. By searching the database, we can find out those protein sequences that have a certain number of matched tagged (with 1 or 2 amino acids errors). But whether there is good match of one peptide sequence in the protein with the whole PSP is not clear. Therefore, it is also a very interesting pattern matching problem.

The approximate matching and pattern matching problem in the context of peptide sequencing is a special matching problem, since it involves both approximate tags matching and approximate masses matching. We have proposed a novel algorithm to solve this novel problem.

The research on string matching has been investigated by many researchers, and the theory and algorithms are quite developed now. It is known that inexact string matching with errors can be done in linear time, and exact string matching with wildcard can be done in linear time [13, 14]. Moreover, the semi-numerical inexact string matching algorithms [13, 14] can be very efficient if the patterns are relatively short. In the PSP algorithm, we have used the semi-numerical inexact string matching algorithms, and the database search process has been done in linear time.

The formal problem definition and the procedure of our algorithm are listed in Fig. 4.

Problem: Approximate database search using PSP

- Input:
    1) *peptide sequence pattern* (PSP)
       $PSP_i = m_1t_1m_2t_2...m_nt_nm_{n+1}$ (m_i and t_i refer to mass and tag, respectively)
    2) database sequence, *Seq*
- Output:
    1) Subsequence Seq_i (or subsequences) in *Seq* that fulfill the requirements
- Requirements:
    1) Approximate match with tags t_i in *Seq* in order, with strict tolerance (every tag with ≤2 amino acids error); if at most $m<n$ tags are present for every database sequences, then these *m* tags should be approximately matched
    2) Approximate match with masses m_i in *Seq* in order, with loose tolerance (every mass with ≤50 Da mass error)
    3) Efficient process

Procedure: Approximate database search using PSP

1. Select the top PSPs (currently top 3), search database for candidate peptides that approximately match with the tags and masses of these PSPs within certain tolerance.
2. Score and tank the candidate peptides by the share peaks count between their theoretical spectrum and experimental spectrum.
3. Output these peptide sequences.

Fig. 4. Formal description of the approximate pattern matching problem; and the procedure for the PSP algorithm

An illustration of approximate match of PSP to the peptide sequences in the database is in Fig. 5.

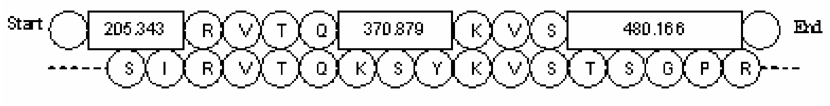


Fig. 5. An example of the match of the peptide sequence pattern (first row) and the peptide sequence in the database (second row)

As illustrated in Fig. 5, the PSP is "[205.343]RVTQ[370.879]KVS[480.166]", with numbers in brackets the mass differences between tags; and the matched peptide sequence is "SIRVTQKSYKVSTSGPR". In this example, the two tags "RVTQ" and "KVS" have matched the identical fragments in the peptide sequence (in other cases, 1 or 2 amino acids mismatches are tolerable). The three mass differences also match with the fragments having similar masses.

As to the running time, for one PSP having length of *m* and one peptide sequence in database having length *n*, the algorithm can operate in O(m+n) time. This is much better than the naïve sequence matching method, which requires O(m*n) time. Since there are thousands of peptide sequences in database, the efficiency improvement is very significant. If we load the peptide database into memory once, and search several PSPs against it, the average processing time for one PSP is even shorter.

4 Experiments

In these experiments, dataset being used is GPM (Global Proteome Machine) dataset [1]. The experimental data are selected from GPM dataset randomly, with different charges.

We have selected some spectra with different charges at random from GPM datasets for careful analysis. The methods to be compared are PeptideSearch [6], SPIDER [12], the 2 typical database search methods based on tags; MASCOT [9], one of the most popular database search methods; and the recent InsPecT [10] software.

Both of PeptideSearch and SPIDER need a tagged sequence (sequence composed of tags and masses) for the peptide search; we have used the PSP generated by us as such tagged sequence. For MASCOT and InsPecT, the input is original spectrum data. The PeptideSearch method uses the non-redundant database in FASTA format, which obtain the peptide sequences from various protein databases. SPIDER, MASCOT, InsPecT and our algorithm used the SWISS-PROT protein database. The SWISS-PROT protein database that we have used is Swiss-Prot Release 45.5 of 04-Jan-2005, which contains 167089 protein sequence entries. The default parameters have been used for the peptide sequencing for all of these algorithms, and the sequencing result with top rank is treated as the sequencing result.

Due to the space constraint, we have only compared PSP algorithm with MASCOT [9] and InsPecT [10] in details, and explain the comparison results against PeptideSearch [6] and SPIDER [12] briefly.

The comparison with MASCOT is meaningful. The MASCOT algorithm is currently regarded as one of the most accurate database search algorithms, so the comparisons with MASCOT can tell our PSP algorithm is very good or just another normal algorithm. More important is that MASCOT is not based on tags, and the input is the spectrum data, same as our algorithm's input. Therefore such comparison is fair. The results of the comparisons are shown in **Table 1**. The "accurate subsequences" refer to the subsequences of the correct sequences, and at the appropriate position of the sequences.

Table 1. Comparisons of MASCOT and PSP on selected spectra. The accurate subsequences are labeled in italics and red. A "-" means that there is no result.

M/Z	charge	correct	MASCOT	PSP
1219.8	2	VAQLEQVYIR	*VAQLEQVYIR*	*VAQLEQVYLR*
1397.9	2	ELEEIVQPIISK	*ELEEIVQPIISK*	*ELEEIVQPIISK*
1644.9	2	PAAPAAPAPAEKTPVKK	LHGGNAIGFMTLEGTK	*AAPA*ETSDLEFA*VKK*
881.5	2	SPRLRPR	LVIVAL*PR*	*SP*IVRG*PR*
1448.7	2	LPGAYFFSFTLGK	MLRAMVASGSE*LGK*	LVRGQNTVHI*LGK*
1888.1	3	VTHAVVTVPAYFNDAQR	*VTHAVVTVPAYFNDAQR*	I*VVT*QPRRISAVSV*AER*
1934.1	3	DNHLLGTFDLTGIPPAPR	*DNHLLGTFDLTGIPPAPR*	KNVA*LIG*LTVE*TG*SALVPK
1934.3	3	DNNLLGKFELTGIPPAPR	*DN*H*LLG*TFD*LTGIPPAPR*	*DNNLLGKFELTGIPPAPR*
1838.8	3	SSYSLSGWYENIYIR	*SS*L*SIS*SMFCNYDETR	*SSYSLSGWYENIYIR*
1761.0	3	PAAPAPAEKTPVKKKAR	LFFAFEKQESVPYR	-
1932.8	4	HKVYACEVTHQGLSSPVTK	VFFDNNFQCILWFLK	TLKVDGNDETFALSNISK
2000.2	4	PAAPAAPAPAEKTPVKKKAR	GQYEPV*AE*IGVGAYGTVYK	*PAAP*K*AAPA*TPAAPAPVYLR
1936.1	4	SIRVTQKSYKVSTSGPR	EGEYTGRTPSGADVTLQR	*SIRVTQKSYKVSTSGPR*
2101.1	4	KIETRDGKLVSESSDVLPK	MVQPDSSSLAEVLDR*VL*DK	*KIETRDGKLVSESSDVLPK*
2140.2	4	KASGPPVSELITKAVAASKER	GER*PP*DVETTVILPESVFR	*KASGPPVSELITKAVAASKER*
1933.3	4	VTIAQGGVLPNIQAVLLPK	DPEDGRPAPGVEHSNGLGK	*VTIAQGGVLPNIQAVLLPK*
3292.8	5	LLILEAGHRMSAGQALDHPWVITMAAGSSMK	EP*LE*LEDIPIEIDNDDDEDDED*GS*GVEYD	[387.26]WCGG[12.55]GD[1438.93]PIDIY*MK*
3291.8	5	LEILLHLTSLSQTFNHFFPEEKFETLR	QPIYPYGSPMGAHVYYPPPVAQPPVRGPVR	SPKVPRTL*LT*LDEQVLSFQRKVGILYCR
3151.2	5	MGSMFRSEEVALVQLFLPTAAAYTCVSR	*GS*GLPDLVLD*VA*GEFYKFGLEGIGAVLLG*SR*	D*EEV*DELYREAPIDKKGNFNYIEFTR
3752.0	5	LPPGEQCEGEEDTEYMTPSSRPLRPLDTSQSSR	CTPFRPSAMSPDFVAQVPLAPDLLPLAELFQRAR	RVEKNALKSQLRSMQ*EQ*LAEMQQKYVQLC*SR*
2359.0	5	CDKDLDTLSGYAMCLPNLTR	LGVMLVGWGGNNGSTLTAGVIANR	[1655.89]AGVPC*TR*

It is obvious from the table above that in these cases; our algorithm is more accurate than MASCOT. MASCOT can find exact match in only 4 cases, and ours can find exact match in 8 cases. In other cases, the results of our algorithm also have comparable number of matches to the true peptide sequences.

It is known that recent MASCOT has already incorporated the tags function similar to PeptideSearch [6], but our algorithm can still beat MASCOT for some spectrum data. This shows that our PSP algorithm, which contains the new strategies to find tags from spectra, as well as our database search techniques, is quite effective.

The comparisons of PSP algorithm against PeptideSearch [6] and SPIDER [12] (details not shown) show that the PSP algorithm has comparable or higher accuracies than these tag-based algorithms.

To evaluate the performance of PSP and InsPecT [10] algorithm, we use the following accuracy measures:

$$\text{Sensitivity} = \#\,\text{correct} / |\rho| \qquad \text{Tag-Sensitivity} = \#\,\text{tag-correct} / |\rho|$$
$$\text{Specificity} = \#\,\text{correct} / |P| \qquad \text{Tag-Specificity} = \#\,\text{tag-correct} / |P|$$

where #correct is the "*number of correctly sequenced amino acids*" and #tag-correct is "*the sum of lengths of correctly sequenced tags (of length > 1)*". The number of correctly sequence amino acids is computed as the longest common subsequence (lcs) of the correct peptide sequence ρ and the sequencing result P. The *sensitivity* indicates the quality of the sequence with respect to the correct peptide sequence and a high sensitivity means that the algorithm recovers a large portion of the correct peptide. The *tag-sensitivity* accuracy take into consideration of the continuity of the correctly sequences amino acids. For a fairer comparison with algorithms that only outputs the highest scoring tags (subsequences) we also use specificity and tag-specificity measures.

Results show that our database search algorithm has comparable accuracy results to Inspect based on our accuracy functions. Though the PSP algorithm has lower accuracies than Inspect for spectrum data with charge 1 and 2, it has comparable or higher accuracies compared with Inspect for spectrum with charge > 2. This shows the power of PSP for multiple charge spectrum data. In **Table 2**, the accuracies in cells are represented in a (specificity accuracy/sensitivity accuracy/[tag specificity accuracy/tag sensitivity accuracy]) format.

Table 2. The accuracy results of PSP and Inspect on GPM datasets

Charge	Number of spectrum	PSP	Inspect
1	756	0.301/0.285[0.110/0.108]	0.448/0.446[0.287/0.289]
2	874	0.412/0.400[0.213/0.212]	0.460/0.455[0.305/0.305]
3	454	0.338/0.339[0.143/0.144]	0.360/0.362[0.193/0.194]
4	207	0.302/0.322[0.099/0.109]	0.276/0.292[0.102/0.109]
5	37	0.286/0.340[0.088/0.120]	0.241/0.279[0.077/0.093]
Total	2328	0.350/0.343[0.153/0.152]	0.417/0.417[0.256/0.257]

We have calculated the ratios that the completely correct peptides are sequenced by the algorithms. Results (not shown here) show that Inspect has better performance that PSP algorithm based on this criteria.

We have also compared some of our sequencing results with those obtained from Inspect, and listed the sequencing results in details. From these results, we can see that both PSP and Inspect can correctly predict a large portion of the peptide sequences. Results are shown in **Table 3**. The “accurate subsequences” refer to the subsequences of the correct sequences, and at the appropriate position of the sequences.

Table 3. Comparisons of Inspect and PSP on selected spectra. The accurate subsequences are labeled in italics and red. A “-” means that there is no result.

M/Z	charge	correct	Inspect	PSP
1219.8	2	VAQLEQVYIR	*VAQLEQVYIR*	*VAQLEQVYLR*
1397.9	2	ELEEIVQPIISK	*ELEEIVQPIISK*	*ELEEIVQPIISK*
1644.9	2	PAAPAAPAPAEKTPVKK	*PAAPAAPAPAEKTPVKK*	*AAPA*ETSDLEFA*VKK*
881.5	2	SPRLRPR	PSIVG*RPR*	*SP*IVRG*PR*
1448.7	2	LPGAYFFSFTLGK	*LP*QSLKLHIIV*GK*	LVRGQNTVHI*LGK*
1888.1	3	VTHAVVTVPAYFNDAQR	*VTHAVVTVPAYFNDAQR*	I*VVT*QPRRISAVSV*AER*
1934.1	3	DNHLLGTFDLTGIPPAPR	*DNHLLGTFDLTGIPPAPR*	KNVA*LIG*LTVE*TG*SALVPK
1934.3	3	DNNLLGKFELTGIPPAPR	*DN*H*LLG*TFD*LTGIPPAPR*	*DNNLLGKFELTGIPPAPR*
1838.8	3	SSYSLSGWYENIYIR	SDGGLVMKRDPTE*YIR*	*SSYSLSGWYENIYIR*
1761.0	3	PAAPAPAEKTPVKKKAR	-	-
1932.8	4	HKVYACEVTHQGLSSPVTK	-	TLKVDGNDETFALSNISK
2000.2	4	PAAPAAPAPAEKTPVKKKAR	*PAAPAAPAPAEKTPVKKKAR*	*PAAP*K*AAPA*TPAAPAPVYLR
1936.1	4	SIRVTQKSYKVSTSGPR	YGKPFKLIFH*VST*LQR	*SIRVTQKSYKVSTSGPR*
2101.1	4	KIETRDGKLVSESSDVLPK	*KIETRDGKLVSESSDVLPK*	*KIETRDGKLVSESSDVLPK*
2140.2	4	KASGPPVSELITKAVAASKER	*KASGPPVSELITKAVAASKER*	*KASGPPVSELITKAVAASKER*
1933.3	4	VTIAQGGVLPNIQAVLLPK	*VAQLEQVYIR*	*VTIAQGGVLPNIQAVLLPK*
3292.8	5	LLILEAGHRMSAGQALDHPWVITMAAGSSMK	*ELEEIVQPIISK*	[387.26]WCGG[12.55]GD[1438.93]PIDIY*MK*
3291.8	5	LEILLHLTSLSQTFNHFFPEEKFETLR	*PAAPAAPAPAEKTPVKK*	SPKVPRTL*LT*LDEQVLSFQRKVGILYCR
3151.2	5	MGSMFRSEEVALVQLFLPTAAAYTCVSR	PSIVG*RPR*	D*EEV*DELYREAPIDKKGNFNYIEFTR
3752.0	5	LPPGEQCEGEEDTEYMTPSSRPLRPLDTSQSSR	*LP*QSLKLHIIV*GK*	RVEKNALKSQLRSMQ*EQ*LAEMQQKYVQLC*SR*
2359.0	5	CDKDLDTLSGYAMCLPNLTR	*VTHAVVTVPAYFNDAQR*	[1655.89]AGVPC*TR*

The experiments on ISB datasets [15] are also performed. The results show that our results are not as accurate as the results of Inspect, but comparable to Mascot’s (results not shown here).

Other experimental results (details not shown) clearly show that the processing time of our PSP algorithm is moderate. Running on a PC with 3GHz of CPU and 1GB of RAM, it uses about 10 seconds for the sequencing of one spectrum (the average of 50 PSPs checked). The time is also dependent on the quality of the spectrum data, especially the accuracy of the parent mass, so high quality data may result in fast process as well as high accuracy. For example, the running time is about 60 seconds for a spectrum data, for which we have generated more than 300 PSPs in step 2 of the PSP algorithm (refer to Fig. 4). The running time is comparable with typical methods such as Sequest [5], but slower than ImPacT[10].

Since the process is fast, we have established a web-based portal for the peptide sequencing based on PSP algorithm. It is located at http://ras-0.comp.nus.edu.sg/~msms/.

5 Conclusions

We have developed a database search algorithm for peptide sequencing using tandem mass spectrometry. The key steps of the algorithm are the selection of the tags from the spectrum of the peptide, and the approximate match of the PSP against the peptides in the database. Our algorithm does not need the comparison of the experimental spectrum to the theoretical spectrum of the peptide in the database; and in most of the cases, it does not even need to check all of the peptides in the database. Since our algorithm can output results that contain uninterrupted mass values, it has the potential to cope with the post-translational modifications. Experiments show that our algorithm is comparable to or more accurate than other database search algorithms, including those based on tags.

Currently, we have used a linear time approximate pattern matching algorithm to search the PSP against the database in linear time, and we have not tried other efficient pattern matching algorithms. Also, the scoring of the candidate peptide sequences against the theoretical spectrum is simply based on share peaks count. We have seen room for further improvements in efficiency and accuracy based on our current computational model, and we have been working on them.

The database search algorithm is essentially a pattern matching algorithm in the context of the peptides sequencing, so it is interesting to see if this program has any other applications such as text mining.

Acknowledgement

The authors would like to thank Pavel Pavzner for insightful discussion. This work was partially supported by the National University of Singapore under grant R252-000-199-112.

References

[1] R. Craig, J. P. Cortens and R. C. Beavis. Open source system for analyzing, validating, and storing protein identification data. *J Proteome Res.*, 3:1234-1242, 2004.

[2] A. Frank and P. Pevzner. PepNovo: De Novo Peptide Sequencing via Probabilistic Network Modeling. *Anal. Chem.*, 77:964 -973, 2005.

[3] B. Ma, K. Zhang, C. Hendrie, C. Liang, M. Li, A. Doherty-Kirby and G. Lajoie. PEAKS: Powerful Software for Peptide De Novo Sequencing by MS/MS. *Rapid Communications in Mass Spectrometry*, 17:2337-2342, 2003.

[4] K. F. Chong, K. Ning and H. W. Leong. De Novo Peptide Sequencing For Multiply Charged Mass Spectra. *To appear APBC2006*, 2006.

[5] J. K. Eng, A. L. McCormack and I. John R. Yates. An approach to correlate tandem mass spectral data of peptides with amino acid sequences in a protein database. *JASMS*, 5: 976-989, 1994.

[6] M. Mann and M. Wilm. Error-tolerant identification of peptides in sequence databases by peptide sequence tags. *Analytical Chemistry*, 66:4390-4399, 1994.

[7] D. Fenyo, J. Qin and B. T. Chait. Protein identification using mass spectrometric information. *Electrophoresis*, 19:998-1005, 1998.

[8] P. A. Pevzner, V. Dancik and C. L. Tang. Mutation-tolerant protein identification by mass-spectrometry. *International Conference on Computational Molecular Biology (RECOMB 2000)*, 231-236, 2000.

[9] D. N. Perkins, D. J. C. Pappin, D. M. Creasy and J. S. Cottrell. Probability-based protein identification by searching sequence databases using mass spectrometry data. *Electrophoresis*, 20:3551-3567, 1999.

[10] S. Tanner, H. Shu, A. Frank, M. Mumby, P. Pevzner and V. Bafna. Inspect: Fast and accurate identification of post-translationally modified peptides from tandem mass spectra. *submitted*, 2005.

[11] V. Dancik, T. Addona, K. Clauser, J. Vath and P. Pevzner. De novo protein sequencing via tandem mass-spectrometry. *J. Comp. Biol.*, 6:327-341, 1999.

[12] Y. Han, B. Ma and K. Zhang. SPIDER: Software for Protein Identification from Sequence Tags with De Novo Sequencing Error. *2004 IEEE Computational Systems Bioinformatics Conference (CSB'04)*, 2004.

[13] D. Gusfield. Algorithm on Strings, Trees, and Sequences: Computer Science and Computational Biology. Cambridge University Press, 1st edition, 1997.

[14] S. Wu and U. Manber. AGREP - A Fast Approximate Pattern-matching Tool. *Proceedings of the Winter 1992 USENIX Conference*, 153-162, 1992.

[15] A. Keller, S. Purvine, A. I. Nesvizhskii, S. Stolyar, D. R. Goodlett and E. Kolker. Experimental protein mixture for validating tandem mass spectral analysis. *OMICS*, 6:207-212, 2002.

Filtering Bio-sequence Based on Sequence Descriptor

Te-Wen Hsieh[1], Huang-Cheng Kuo[1], and Jen-Peng Huang[2]

[1] Department of Computer Science and Information Engineering,
National Chiayi University, Taiwan
{s0930295, hckuo}@mail.ncyu.edu.tw
[2] Department of Information Management,
Southern Taiwan University of Technology, Taiwan
jehuang@mail.stut.edu.tw

Abstract. Study on biological sequence database similarity searching has received substantial attention in the past decade, especially after the sequencing of the human genome. As a result, with larger and larger increases in database sizes, fast similarity search is becoming an important issue. Transforming sequences into numerical vectors, called sequence descriptors, for storing in a multidimensional data structure is becoming a promising method for indexing bio-sequences. In this paper, we present an effective sequence transformation method, called SD (Sequence Descriptor) which uses multiple features of a sequence including Count, RPD (Relative Position Dispersion), and APD (Absolute Position Dispersion) to represent the original sequence data. In contrast to the q-gram transformation method, this avoids the problem of exponentially growing vector size. Also, we present a transformation, called ST (Segment Transformation), which recursively divides sequence data into equal length subsequences, and concatenates them after transformation of the subsequences. Experiments on human genome data show that our transformation method is more effective than the q-gram transformation method.

Keywords: Bio-sequence, Sequence Descriptor, Similarity Searching, KNN.

1 Introduction

In biological databases, such as genome and protein databases, searching for sequences with high alignment scores in relationship to the query sequence is an essential task. Researchers have discovered lots of information of interest via the sequence similarity analysis. Similar bio-sequences of different species usually imply the relationship of function and evolution. Moreover, there is implicit structural information for protein data. For instance, approximate sequence analysis has assisted the detection of certain strains of Escherichia coli (E. coli) bacteria, often responsible for infant diarrhea and gastroenteritis.

Although the traditional DP (Dynamic Programming) alignment algorithm guarantees the accurate alignment result, it is impractical because of its requiring both $O(n^2)$ in time and space. Therefore, several heuristics have been developed, such as BLAST [1, 17], Pattern Hunter [11, 12], and FASTA [14]. These methods not only provide good accuracy, but also execute efficiently. Particularly, BLAST is the most popular method because of its excellent accuracy and speed. However, a study of the performance of BLAST shows that BLAST is becoming 64% slower each year because of the exponentially growth of biological databases [7].

Filtering has been suggested to solve the problem. Researchers transform the sequences into numerical vector space [15] for storing in a multidimensional index [16]. Then DP or another heuristic algorithm is used to align the small number of selected sequences with the query sequence.

In this paper, we present an effective transformation method, called SD (Sequence Descriptor), which uses multiple features including Count, RPD (Relative Position Dispersion) and APD (Absolute Position Dispersion). In contrast to the q-gram transformation method [13], in which vector size grows exponentially with q, we use fewer dimensions and receive better accuracy. Additionally, we also present a transformation method, called ST (Segment Transformation), which is more powerful than Wavelet Transformation. It recursively divides sequence data into equal length subsequences and concatenates them after transformation.

The rest of the paper is organized as follows. Section 2 discusses the background and related work. Terminology and formulation are in section 3. Section 4 discusses the proposed transformation techniques and their integration. Section 5 demonstrates a concise empirical performance analysis and the simulation results. Finally, section 6 contains the conclusion and future work.

2 Background and Related Work

In a typical sequence similarity search application, there are two commonly used query types: (i) k-Nearest Neighbor (k-NN), which asks for the most similar k sequences to the query sequence; and the (ii) ε-range query, which asks for sequences sufficiently similar to the query sequence. In ε-range queries, users must have enough domain knowledge to predetermine a threshold ε to check whether the sequences in the database and the query sequence are similar or not. In contrast to ε-range queries, k-NN queries are more popular because of its convenience. Therefore, we perform k-NN queries for illustrating our methods.

As the biological databases become larger and larger, even the heuristics are no longer feasible. Therefore, a new method of filtration has been developed. Researchers transform original sequence data into numerical vector space, and filter out the distant potential dataset via metric distance functions. Transforming sequence data into a frequency domain is the most popular method. Some variants of it are still constructed on frequency domains. For instance, a DNA sequence S = “ACGGTCAGAA” on $\Sigma = \{A, C, T, G\}$ can be transformed into a frequency domain as [4, 2, 1, 3].

In [13], frequency transformation of a sequence S into N-gram, FT#N(S) is defined as a vector of frequencies of N-grams. For the above DNA sequence S, FT1(S) = [4, 2, 1, 3], and FT2(S) = [1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1].

Moreover, they also combine their FT#N(S) with Wavelet Transformation [13]. N-gram Frequency Wavelet Transformation is represented as WT#N(S). After dividing the sequence S into S_α and S_β, they compute the transformations V_α=FT#N(S_α) and V_β=FT#N(S_β). Then they concatenate the addition and subtraction of those two vectors, i.e., [$V_\alpha + V_\beta$, $V_\alpha - V_\beta$]. The definition given here is the first two wavelet coefficients of the last recursion in that formal definition. This definition promotes efficiency with the elimination of recursion. In their experiment results WT#N(S) is better than FT#N(S). For the above DNA sequence S, WT1(S) = [4, 2, 1, 3, -2, 0, 1, 1].

However, in q-gram transformation the vector size grows exponentially with q. And the Wavelet Transformation also makes the vector size double. The tradeoff between accuracy and vector size must be considered. In their research they show N=2 producing satisfactory results.

We present a transformation with multiple features to represent the original sequence data. As such, we can use fewer dimensions to create better accuracy. In addition, we present a better transformation method, Segment Transformation, in contrast to Wavelet Transformation.

3 Terminology and Formulation

Definition 1 (Count). Let S = $s_1, \dots, s_n$ be a sequence over the alphabet $\Sigma_u = \{\alpha_1, \dots, \alpha_u\}$, then the Count of S, called C(S) is defined as: C(S) = [$C_1, \dots, C_u$], where C_i ($\geq$ 0) corresponds to the total number of α_i in S, and $\sum_{i=1}^{u} C_i = |S| = n$. For example, for the above DNA sequence S, C(S) = [4, 2, 1, 3].

Definition 2 (Relative Position Dispersion, RPD). Let S = $s_1, \dots, s_n$ be a sequence over the alphabet $\Sigma_u = \{\alpha_1, \dots, \alpha_u\}$, and N_{α_i} be the number of α_i in S. $\sum_{i=1}^{u} N_{\alpha_i} = n$. For any symbol α_i in S, let $l_{\alpha_i,j}$ be the difference of positions of the jth α_i and (j+1)th α_i. So, there are $N_{\alpha_i} - 1$ values of position differences for symbol α_i. Relative Position Dispersion of S, called RPD(S), is defined as: RPD(S) = [$\frac{(l_{\alpha_1,1})^2 + \dots + (l_{\alpha_1,N_{\alpha_1}-1})^2}{N_{\alpha_1} - 1}$, … , $\frac{(l_{\alpha_u,1})^2 + \dots + (l_{\alpha_u,N_{\alpha_u}-1})^2}{N_{\alpha_u} - 1}$]. If N_{α_i} is 1, the corresponding term in RPD is set to 0. For example, for the above DNA sequence S, RPD(S) = [$(6^2+2^2+1^2)/(4-1)$, $4^2/(2-1)$, 0, $(1^2+4^2)/(3-1)$] = [13.67, 16, 0, 8.5].

Definition 3 (Absolute Position Dispersion, APD). Let S = $s_1, \dots, s_n$ be a sequence over the alphabet $\Sigma_u = \{\alpha_1, \dots, \alpha_u\}$, and N_{α_i} be the number of α_i in S. Let $p_{\alpha_i,1}, \dots, p_{\alpha_i,N_{\alpha_i}}$ represent the positions of α_i in S. Absolute Position Dispersion of S, called APD(S), is defined as: APD(S) =[$\frac{(p_{\alpha_1,1})^2 + \dots + (p_{\alpha_1,N_{\alpha_1}})^2}{N_{\alpha_1}}$, … , $\frac{(p_{\alpha_u,1})^2 + \dots + (p_{\alpha_u,N_{\alpha_u}})^2}{N_{\alpha_u}}$].

Definition 4 (k^{th}-level Segment Transformation, k^{th}-ST). Let S = $s_1, \ldots, s_n$ be a sequence over the alphabet $\Sigma_u = \{\alpha_1, \ldots, \alpha_u\}$, k^{th}-level Segment Transformation recursively divides S into 2^k equal-length subsequences of S, $S = \left\{S_{1,2^k}, \ldots, S_{2^k,2^k}\right\}$, k^{th}-ST(S) = $\left\{V_{1,2^k}, \ldots, V_{2^k,2^k}\right\}$, $0 \leqq k \leqq \log_2 n$ ($V_{i,2}{}^k$ indicates the vector of $S_{i,2}{}^k$ after transformation.) Due to the vector size limitation, and we mainly compare with the first and second coefficients of Wavelet Transformation, our Segment Transformation indicates the 1st-level Segment Transformation throughout this paper.

4 Transformation Techniques

Given a sequence database, blocking is initiated first. It means that we partition the initial sequences into equal length subsequences (blocks). There are mainly three different blocking methods: i) incremental blocking: Each of the consecutive blocks of length l, overlap by l – 1 residues; ii) half overlap blocking: Each of the consecutive blocks of length l, overlap by l/2 residues; and iii) non-overlapping blocking. As more blocks are extracted, we observed better accuracy, however resulting in a higher computational cost. Therefore, there is a tradeoff between cost and accuracy.

Consequently, we can execute our SD (Sequence Descriptor) transformation on each subsequence (block). In SD transformation, we first compute the C(S), RPD(S), and APD(S) which were defined in the above section.

Count of S, C(S), counts total occurrences of each kind of base in S. Because each subsequence has the same length, it indicates frequencies of occurrences of each kind of base. Relative Position Dispersion of S, RPD(S), computes the average squared distance between each pair of the nearest similar base. It indicates the degree of relative dispersion of each kind of base in S. Absolute Position Dispersion of S, APD(S), computes the average square distance from the first base of each kind of base in S. It indicates the degree of absolute dispersion of each base in S.

In contrast to q-gram transformation using only one feature, SD uses multiple features in addition to frequency. We can expect that our method will result in a better order preservation because using multiple features can represent the original sequence data better if each feature is feasible. Additionally, our method make the vector size grow linearly according to the number of features. Therefore, we avoid the problem of exponential growth in vector size as found in q-gram transformation.

Because we use multiple features and suppose presently that each feature has the same weight, we must normalize each feature vector first. Then we concatenate them to represent the original sequence S, SD(S) = [C(S), RPD(S), APD(S)]. Moreover, we need to normalize each element in SD(S) to be an integer between 0 and *Max* in order to reduce storage space. For example, for the above DNA sequence S, SD(S) = [4/10, 2/10, 1/10, 3/10, 16.67/41.17, 16/41.17, 0/41.17, 8.5/41.17, 57.75/132.42, 20/132.42, 25/132.42, 29.67/132.42] = [0.4, 0.2, 0.1, 0.3, 0.4, 0.39, 0, 0.21, 0.44, 0.15, 0.19, 0.22]. Then we normalize each element to be an integer between 0 and 65535, SD(S) = [26214, 13107, 6554, 19661, 26214, 25559, 0, 13762, 28835, 9830, 12452, 14418].

SSD (Segment Sequence Descriptor), combination of SD (Sequence Descriptor) and ST (Segment Transformation), partitions the sequences (blocks) into two equal length subsequences before SD transformation, i.e., SSD(S) = [C($S_{1,2}$), C($S_{2,2}$), RPD($S_{1,2}$), RPD($S_{2,2}$), APD($S_{1,2}$), APD($S_{2,2}$)].

After all subsequences (blocks) in the database have been transformed into numerical vector space using SD (or SSD) transformation, they can be stored in a multidimensional indexing structure, such as R-tree [8], to speed up query procedure. All of the above procedures are completed in the offline phase, and the searching procedure is executed online.

In the query procedure we also block the query sequence first. Then each query subsequence (block) is transformed into its corresponding numerical vector. Accordingly, we discard the distant sequences.

5 Performance Analysis

In our experiments, considering the vector size, we mainly compare FT#2(S) with our SD(S), WT#2(S) with our SSD(S), and Wavelet Transformation with our Segment Transformation. We take a sample from 18th chromosome of the human genome [NCBI, http://www.ncbi.nlm.nih.gov/] as our testing data for performing k-NN searching. We incorporate a non-overlapping blocking method with block size = 500 and use L_1 distance function to compute the distance between two vectors. First, we randomly select a start point in that sample sequence and extract 1000 subsequences as our data sequences. Then we extract 30 query subsequences from the same start point in the same sequence as our query sequences.

Table 1. Vector size (# of dimensions) in each transformation method

	SD	FT#2	SSD	WT#2
vector size(# dimension)	12	16	24	32

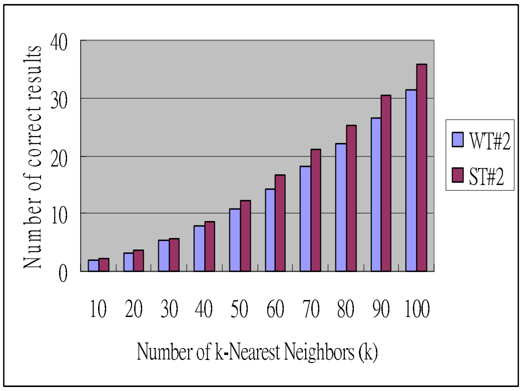


Fig. 1a. True Positives for k-NN on human chromosome18 dataset when comparing WT#2 with ST#2

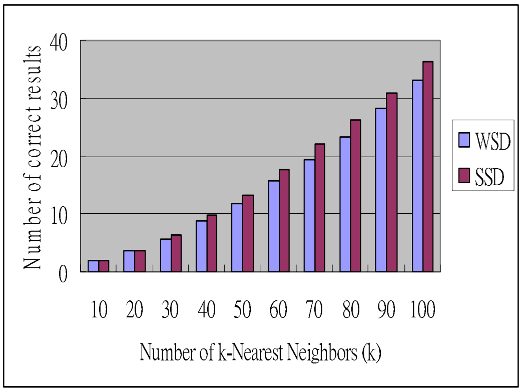


Fig. 1b. True Positives for k-NN on human chromosome18 dataset when compared WSD with SSD

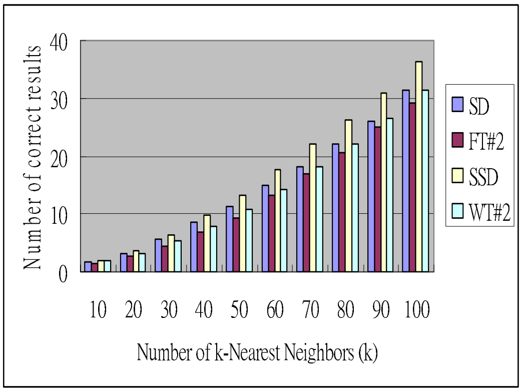


Fig. 2. True Positives for k-NN on human chromosome18 dataset

In contrast to FT#2(S) and WT#2(S) respectively, our transformation methods only use 75% of the vector size dimensions as shown in Table 1. Therefore, we can theoretically save 25% running time (query time) and memory space; especially, when we take into account the I/O time. The I/O time affects the total execution time because I/O is usually the most time consumed during the execution of a process.

In Figure 1a, we observe that ST#2 (2-gram Frequency Segment Transformation) has a better TPR (True Positive Rate) than WT#2 (2-gram Frequency Wavelet Transformation). Also we can see that SSD (Segment Sequence Descriptor) has a better TPR than WSD (Wavelet Sequence Descriptor) in Figure 1b. It indicates that whether we use SD or q-gram transformation, Segment Transformation is better than Wavelet Transformation.

In Figure 2, we observe that our SD Transformation is better than 2-gram Transformation, and our SSD is better than WT#2(S). When comparing our SD transformation with the 2-gram transformation, there is about 3.3% average improvement in

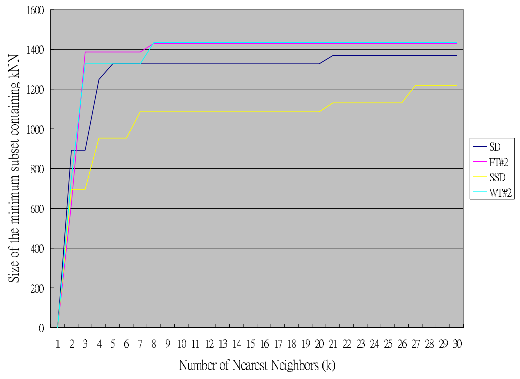


Fig. 3. Filtration for k-NN on human chromosome18 dataset

TPR. And our SSD also has about a 4.8% average improvement in TPR when compared to WT#2(S). Even our SD transformation is better than WT#2(S) in most k values.

We also use filtration ratio to compare our transformation method with q-gram transformation method and change the dataset size to be 10000 subsequences. In Figure 3, we can see that our SD and SSD have a better filtration ratio in almost all k values than FT#2 and WT#2 respectively. It means that when we search the correct k nearest neighbors for a query sequence, we can discard more distant potential dataset as compared to q-gram transformation methods. Therefore we can reduce a larger amount of time consuming alignments in second phase. Take k=30 as an example, our SD has a better filtration ratio 13.68% than 14.3% of FT#2. And our SSD has a better filtration ratio 12.2% than 14.35% of WT#2. In other words, we only need to align no more than 1368 subsequences when we want to search 30 correct nearest neighbors for a query using SD or SSD.

In order to understand how well each transformation method reflects the distance between a pair of sequences, we compute the distance between vectors (sequence descriptors) of sequences, and perform sequence alignments using dynamic

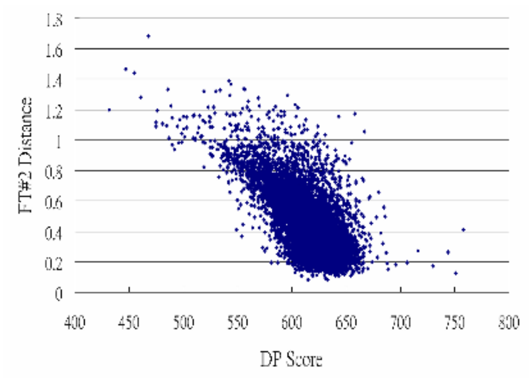


Fig. 4a. DP Scores and FT#2 Distances

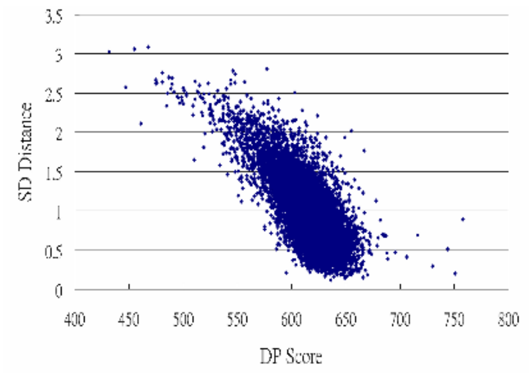


Fig. 4b. DP Scores and SD Distances

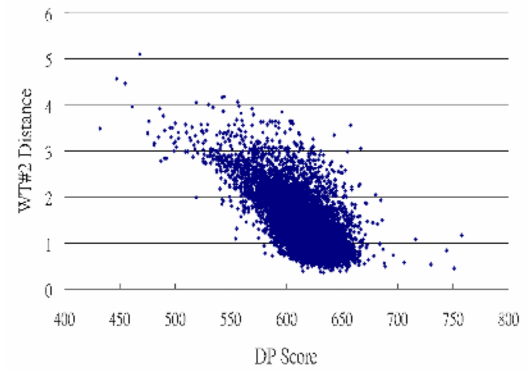


Fig. 4c. DP Scores and WT#2 Distances

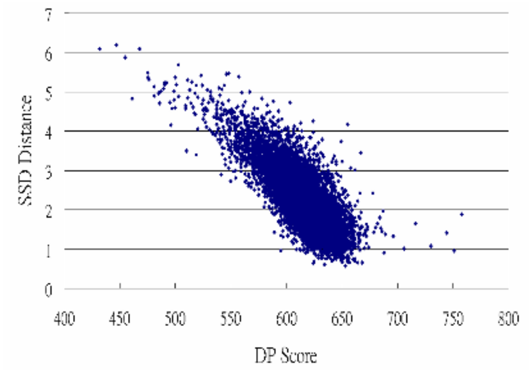


Fig. 4d. DP Scores and SSD Distances

programming to obtain DP scores between the pair of sequences. For more precise experimental results, we randomly selected 10,000 pairs of subsequences from the same sequences used in the TPR measurement. Figure 4a to figure 4d show the relationship between sequence descriptor distances and DP scores. The correlation coefficients are -0.68, -0.75, -0.68, and -0.78, for FT#2, SD, WT#2, and SSD transformations, respectively. The correlation coefficient improves 0.07 from FT#2 to SD. The improvement is about 0.1 from WT#2 to SSD. Even SD, with vector size 12, has a better correlation coefficient than WT#2, with vector size 32.

Figure 5 shows that our SD has better correlation coefficients than FT#2 on each block size except block size equals to 100 and 300. However, too small block size will result in too many blocks. On each block size, our SD has a 0.02 average improvement in correlation coefficient as compared to FT#2. And our SSD has better correlation coefficients on all tested block sizes. It has a 0.06 average improvement in correlation coefficient as compared to WT#2.

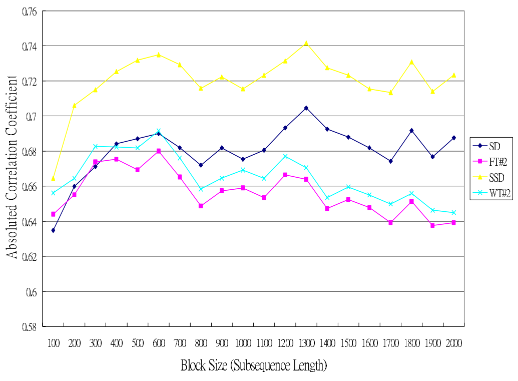


Fig. 5. Correlation Coefficients of Varied Block Size (Subsequence Length)

6 Conclusion and Future Work

In this paper, we present an effective transformation method, called SD (Sequence Descriptor). It uses multiple features to represent the original sequences. The vector size of SD is smaller than that of q-gram transformation, which grows exponentially. In addition, we present a Segment Transformation which has better accuracy as compared with Wavelet Transformation. In our experiments, we demonstrate a better method than the q-gram transformation.

In the future, we will test a large protein dataset and DNA sequences of other species. Finally, we hope to build a complete framework for a multidimensional indexing approach for stringing similarity search problems, which can be helpful for understanding the "boundary effect" on transformations.

Acknowledgement

This work was supported by grant NSC 93-2213-E-415-005 from the National Science Council, Taiwan.

References

1. S. F. Altschul, W Gish, W Miller, E. W. Myers, D. J. Lipman, "Basic local alignment search tool," J. Mol. Biol., 1990, pp. 403-410.
2. S. F. Altschul, T. L. Madden, A. A. Schaffer, J Zhang, Z Zhang, W Miller, D. J. Lipman, "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs," Nucleic Acids Res, 1997, pp. 3389-3402.
3. S. Alireza Aghili, Divyakant Agrawal, Amr El Abbadi, "Filtration of string proximity search via transformation," IEEE International Symposium on Bioinformatics and Bioengineering, 2003, pp. 149-157.
4. S. Alireza Aghili, Ozgur D. Sahin, Divyakant Agrawal, Amr El Abbadi, "Efficient filtration of sequence similarity search through singular value decomposition," IEEE International Symposium on Bioinformatics and Bioengineering, 2004, pp. 403-410.
5. N. Beckmann, H. P. Kriegel, R. Schneider, B. Seeger, "The R* -tree: an efficient and robust access method for points and rectangles," ACM SIGMOD, 1990, pp. 322-331.
6. T. Bozkaya, N. Yazdani, Z. M. Ozsoyoglu, "Matching and indexing sequences of different lengths," the Sixth International Conference on Information and Knowledge Management, 1997, pp. 128-135.
7. Michael Cameron, Hugh E. Williams, Adam Cannane, "Improved gapped alignment in BLAST," IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB) archive Vol. 1, No. 3, 2004, pp. 116-129.
8. A. Guttman, "R-Trees: A dynamic index structure for spatial searching," ACM SIGMOD, 1984, pp. 47-57.
9. T. Kahveci, A. Singh, "An efficient index structure for string databases," VLDB Conference, 2001, pp. 351-360.
10. Emre Karakoc, Z. Meral Ozsoyoglu, S. Cenk Sahinalp, Murat Tasan, Xiang Zhang, "Novel approaches to biomolecular sequence indexing," Bulletin of the IEEE Technical Committee on Data Engineering, 2004, pp. 37-44.
11. M. Li, B. Ma, D. Kisman, J. Tromp, "Patternhunter II: Highly sensitive and fast homology search," J. Bioinformatics and Computational Biology, Vol. 2, No. 3, 2004, pp. 417-439.
12. B. Ma, J. Tromp, M. Li, "Patternhunter: Faster and more sensitive homology search," Bioinformatics, Vol. 18, No. 3, 2002, pp. 440-445.
13. O. Ozturk, H. Ferhatosmanoglu, "Effective indexing and filtering for similarity search in large biosequence databases," IEEE International Symposium on Bioinformatics and Bioengineering, 2003, pp. 359-366.
14. W. R. Pearson, "Flexible sequence similarity searching with the FASTA3 program package," in *Bioinformatics Methods and Protocols*, 1999, Humana Press, pp. 185-219.
15. Kim R. Rasmussen, Jens Stoye, Eugene W. Myers, "Efficient q-gram filters for finding all epsilon-matches over a given length," International Conference on Research in Computational Molecular Biology (RECOMB), 2005, pp. 189-203.
16. T. Sellis, N. Roussopoulos, C. Faloutsos, "The R+ -Tree: A dynamic index for multidimensional objects," VLDB Conference, 1987, pp. 507-518.
17. J. Zhang, T. L. Madden, "PowerBLAST: a new network BLAST application for interactive or automated sequence analysis and annotation," Genome Research, Vol. 7, No. 6, 1997, pp. 649-656.

Automatic Extraction of Genomic Glossary Triggered by Query

Jiao Li and Xiaoyan Zhu

State Key Laboratory of Intelligent Technology and Systems (LITS),
Department of Computer Science and Technology,
Tsinghua University, Beijing 100084, China
jiao-li04@mails.tsinghua.edu.cn, zxy-dcs@tsinghua.edu.cn

Abstract. In the domain of genomic research, the understanding of specific gene name is a portal to most Information Retrieval (IR) and Information Extraction (IE) systems. In this paper we present an automatic method to extract genomic glossary triggered by the initial gene name in query. LocusLink gene names and MEDLINE abstracts are employed in our system, playing the roles of query triggers and genomic corpus respectively. The evaluation of the extracted glossary is through query expansion in TREC2003 Genomics Track ad hoc retrieval task, and the experiment results yield evidence that 90.15% recall can be achieved.

1 Introduction

Biomedicine is really an active academic discipline, the amount of literatures in which is exploding with recording the new discoveries of gene, gene regulation, and protein-protein interaction. Biomedical entities (e.g., genes, proteins) are usually nominated and represented in different forms by authors within their articles, thus, there is a special section naming "Glossary" in most biomedical journals (e.g., Journal of Cell Biology, Journal of Biological Chemistry). With the help of glossary, reader can easily understand the meaning of the unfamiliar symbol, and get the linking knowledge in the article. The above glossary depends on authors to build. However, for a large amount of biological textual resources distributed in the web and databases, such as MEDLINE which contains approximately 15 million records back to 1950s in life sciences [1], it is impossible to organize such kind of glossary manually. Furthermore, in gene-based information systems, both DNA sequence databases (e.g., GenBank [2]) and genomic literature analysis systems (e.g., MeKE [3]), gene name becomes the entry to access the knowledge, while synonyms and homonyms of gene pose a great challenge for these systems. Thus, the ability to enhance gene symbol explanation is becoming increasingly important.

During the past decades, many efforts have been made to facilitate the understanding of gene-related information. They include databases containing gene names as well as their synonyms (e.g., LocusLink [4, 5], euGenes [8, 9]), thesauruses involving gene lexicons (e.g. MeSH [10], UMLS [11, 12]), and systems extracting meaningful acronym pairs from biomedical literature (e.g., Abbreviation Online [14, 15], AcroMed [6, 7]).

***LocusLink*,** developed by National Center for Biotechnology Information (NCBI), provides detailed information about function and position of gene, some special fields of which are related with gene name such as: OFFICIAL SYSMBOL (the officially approved symbol for gene), OFFICAL_GENE_NAME (the official name of the gene), and ALIAS_SYMBOL (an alias that is sometimes used to refer to the gene) etc. [4].

***euGenes*,** developed by Indiana University Biology Department, provides a common summary of gene and genomic information from eukaryotic organism databases, including gene symbol, gene full name, and gene product information etc. [8].

***MeSH (Medical Subject Headings)*,** the controlled vocabulary thesaurus of the National Library of Medicine (NLM), consists of 22,997 biomedical terms naming descriptors in a hierarchical structure conjunct with 83 topical qualifiers [10].

***UMLS (Unified Medical Language System)*,** developed by NLM, includes SPECIALIST lexicon, Semantic Network, and Metathesaurus three parts. Where Metathesaurus is built from the electronic version of many different thesauri, lists of controlled terms, and contains information about biomedical and health related concepts, their various names, and the relationships among them [11].

***Abbreviation Online*,** developed by Stanford University, uses statistical learning algorithm (logistic regression) to score abbreviation expansions based on their resemblance to a training set, and creates an online dictionary of abbreviations from MEDLINE [14, 15].

AcroMed, developed by Brandeis University, uses regular expression algorithm to generate a database which contains 480,000 biomedical acronyms and the associated long forms extracted from MEDLINE abstracts [6].

In contrast with the popular databases, thesauruses, and extraction systems mentioned above, our work is to build a genomic glossary for the specific document collection rather than a global biomedical dictionary for any cases. We hypothesize author defines gene symbol in his/her article if its meaning is new in the literature, before it's used widely in the rest part of the literature. We also hypothesize that the above definition can be reused by the other literatures in the same category. The two hypotheses are tested in the study of gene/protein term identification [16].

Another significant feature of our work is that the glossary generation is triggered by query, which is a thoroughly different workflow from traditional information extraction systems, in which database technologies are employed to store the extracted knowledge and support the search function. The input to our system are a set of biomedical articles and one or more gene names from user, and the output glossary is able to interpret the confusing name appearing at the collection in its own language and provide the related linkage to this interpretation (e.g. ING1 (inhibitor of growth 1) 11966686. Where, *"inhibitor of growth 1"* is the interpretation to *ING1*, and this glossary item is extracted from Doc*11966686.*).

The rest of this paper is organized as follows: in Section 2, we review algorithms for biomedical terms extracted from unstructured files. In section 3, we present our extraction methods. Section 4 states the experimental settings and result analysis, in which we utilize TREC 2003 Genomics Track to evaluate our methods. Section 5 is the conclusions and future works.

2 Related Works

Due to its importance, the understanding of gene name has been the focus of research for a long time. Previous works, introduced briefly in the last section, can be roughly divided into two groups: manual construction and automatic extraction. As our target is to extract glossary (i.e. the definition for the gene symbol) automatically, this review section emphasizes the automatic extraction and matching algorithms, including rule-based, linguistic, statistical, machine learning, and hybrid approaches.

Hisamitsu et al. (1998) extracted technical term and its definition though bi-gram statistic, selected phrases associated with parentheses (the parenthetical phrase and the outer phrase) co-occur more frequently than random firstly, and then they applied a set of rules to identify whether the parenthetical phrase was an explanation of the outer phrase. For example, a rule indicated that a phrase was an abbreviation of a full form if the letters of the phrase appeared in order in the full form. Their evaluation of this approach demonstrated 97% precision [17].

Pustejovsky et al. (2001) introduced the regular expression algorithm to extract acronym form MEDLINE, and used the shallow parser technology to enhance the algorithm [13]. Their evaluation on precision was 98%, but the performance on recall was not significant because of the strict regular expression limited. Another contribution was the presentation of POLYFIND algorithm, which solved the storage and index problems in the database caused by an acronym that has several possible long forms associated with it (named as Polynym).

Chang et al. (2002) created a robust method for identifying biomedical abbreviation using supervised machine learning theory. They decomposed the problem into four components: scanning text for occurrences of possible abbreviations; aligning the candidates to the preceding text; converting the abbreviations and alignments into feature vector; scoring the feature vector using a statistical machine learning algorithm. AcroMed [6] was used as golden standard in their evaluation step, and they achieved 80% precision and 83% recall [14].

Satou et al. (2004) gathered biomedical terms from different resources and presented dictionary building and matching approach [18]. Their matching algorithm succeeded in looking up generative biomedical terms and substrings surrounded by non-space characters with the help of meaningful segmentation.

All the four methods contributed in the definition for domain-specific term, but the limitations might affect their use in the biomedical domain. Hisamitsu et al.'s algorithm depends on statistical significance of the two terms that are associated with parentheses, which leads to missing the terms that are newly introduced into the literature. In Pustejovsky et al.'s study, the strict regular expression rules do not apply to many biomedical terms. Chang et al.'s machine learning method relies on a large scale training set annotated manually. Satou et al.'s dictionary is built on the top of many biomedical resources and may fail to understand the terms that are not involved in the resources.

In our methods, newly introduced gene name can get its proper definition from the latest publication, as our query-triggered glossary extraction depends on the document collection, i.e. document quantity and quality determine the glossary content.

Moreover, triggering term in the query acts as an anchor in the extraction procedure, which avoids a series of problems caused by term identification in the information extraction systems. We introduce guidelines about genome nomenclature that are useful for applying computational approaches to generate genomic glossary [19-24]. Besides considering the above features, we also flexibly loose the rule restriction according to the configuration, taking gene *E2F1* for example,

At the strict level, it can be explained as:

E2F1: E2F transcription factor 1.

At the loose level, it can be explained as:

E2F1: a transcription factor involved in both cell cycle progression and apoptosis.

The detailed descriptions about how the extraction is triggered, how the nomenclature guidelines are utilized, how the restriction is configured and the other methods involved in our system are discussed in the next section.

3 Methods

As our genomics glossary extraction is for specific document collection, and the novel idea of query trigger needs to frequent and flexible interaction with the document collection. Thus, the management of document collection becomes an unavoidable problem in our glossary extraction system. We decompose the query-triggered genomic glossary extraction problem into two main components: (1) Document Indexing, (2) Glossary Extracting.

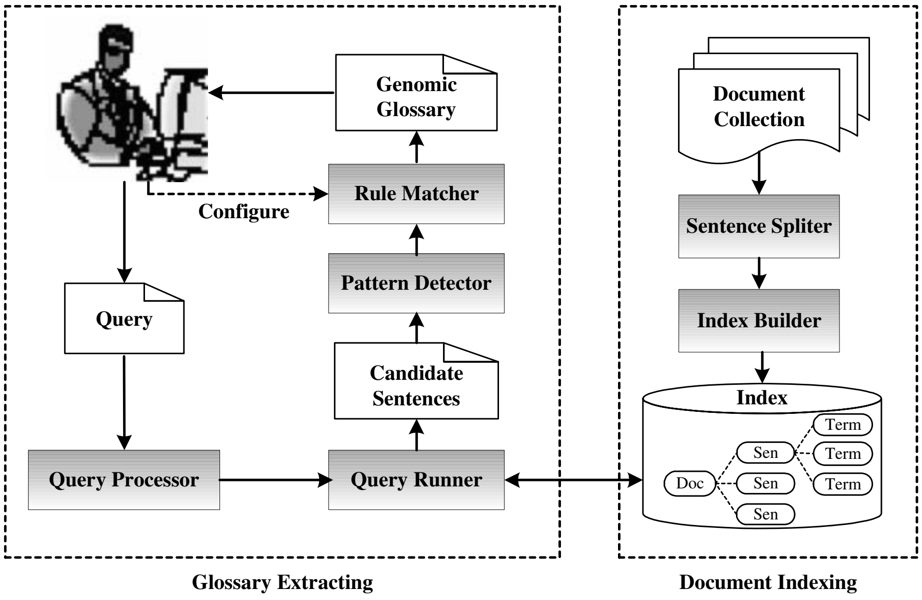


Fig. 1. System Architecture of Query-Triggered Genomic Glossary Extraction

Fig.1 shows the system architecture of query-triggered genomic glossary extraction. Document is split into meaningful unit, sentence, and then pushed into index, which is the foundation work in the workflow. The following workflow would be comprehensive through this scenario: user finds an unfamiliar gene name when he reads a biomedical book that is already included in the index, and he sends a query to our system. Then the system responds as follows: expand the query; retrieve the query and its expansions in the index; get the candidate sentences which contain query terms; recognize the sentences which satisfy certain patterns; match them with the rule set whose restriction level can be configurable; and finally return user the matching result (i.e. our genomic glossary items). The rest of this section focuses on discussing the key steps in our system architecture.

3.1 Document Indexing

Different from traditional IE systems which scan document collection one time and store the extracted information in database, our system is to extract a glossary tailed for document collection, and provide the explanation to the unpredictable query from reader. The simple I/O operations and string matching technologies are not competent any more when the document collection scale explodes rapidly and content updates frequently. We introduce the inverted index, a popular technology applied in search engine [25], to solve the document storage and match problem.

Sentence is a more meaningful unit for term definition compared with passage and document, thus, before being pushed into the index, textual resource is split into sentences while still keeping document information that could provide evidence and linkage for the extracted glossary. In contrast with the classic inverted index, *doc-term* list, there is an additional level *sen* (short for sentence) in our index part (see Fig. 1).

3.2 Query Processing

In the context of biomedical articles, gene name may appear in an abundant number of lexical variants, the main reason of which is the different preferences of hyphenation, spacing and Greek letters. For example, the *tropomyosin-alpha* gene is referred in four different ways: *tropomyosin alpha, tropomyosin 1, tropomyosin-1, tropomyosin1.*

This problem is solved by the query processor in our system. According to the rules (see Table 1.), it creates a list (l_q) of possible variants ($v_{q1}, v_{q2}, v_{q3}\ldots$) for the triggering query (t_q) that contains at least one hyphen, digit, or Greek letter.

Table 1. Sample Rules Used in Query Processor

1. Hyphens are removed from the term.
2. At every transition form alphabetic characters to digit or Greek letter, a hyphen is inserted.
3. Digit is removed when they appear at the end of the term.
4. Greek letters are translated to their numerical and Latin equivalents.

3.3 Pattern Detecting

With the help of inverted index, we efficiently get the candidate sentences, each of which contains at least one t_q or v_{qi} in the l_q, but not all these candidate sentences can provide the evidence for the t_q's definition. Therefore, the pattern detector is used to select the most possible sentences for term definition by some pattern limitations.

Parenthesis Detection: The content in the parenthesis always explains and annotates the content before it, for example, *ATF-2 (activating transcription factor 2).* However, certain parentheses are not associated with reasonable definition, such as parentheses containing only numbers, numbers with percentage symbol, and certain keywords (e.g. fig, table, and PH). If it is detected as containing parenthesis within above features, sentence would be filtered out of the sentence set S_{par}.

Position Detection: The t_q or its lexical variant v_{qi} appears at the head of one sentence, in most cases it plays a subject role in the sentence (i.e. all the sentence content may encompass it). Taking gene name *EAAT2* for instance, the sentence "*EAAT2 is the major carrier of glutamate in the mammalian brain*" is a good interpretation for it. Thus the position factor is also considered by the pattern detector, and the detected sentence set S_{pos} is involved in our loose level glossary.

3.4 Rule Matching

Query term t_q and its lexical variant v_{qi} act as anchors in our system. They help us to find the candidate sentences from the huge documents firstly, and then select two sentence sets, S_{par} and S_{pos}, by the pattern detector. The rule-based algorithm for the anchor related glossary extraction from S_{par} is presented in this section.

Sentence s_{par1} in S_{par} can be formalized as follow:

PreComponent (MidComponent) PosComponent.

Where, PreComponent, MidComponent, and PosComponent are three ordinal components around the parentheses in s_{par1}.

```
ExGlossary ($anchor_g$, $s_{par1}$) {
    if $anchor_g \subset$ PreComponent {
        $def_g'$ = MidComponent;
        Forward_Match ($anchor_g$, $def_g'$);
    }
    elsif $anchor_g \subset$ MidComponent {
        $def_g'$ = PreComponent;
        Backward_Match ($anchor_g$, $def_g'$);
    }
    Return ($anchor_g$, $def_g$);
}
```

Algorithm 1. Rule-based Genomic Glossary Extraction Algorithm

In Algorithm 1, gene name t_q or its lexical variant v_{qi} appearing at $s_{par1}$is expressed as *anchor*$_g$, and the definition for the gene (i.e. the item will be involved in the genomic glossary) is expressed as *def*$_g$ while the candidate definition, the uncertain one before the rule matching applied, is expressed as *def*$_g'$.

The algorithm never considers the PosComponent, as the components before and after the left parenthesis construct the definition relationship. For different positions of *anchor*$_g$, PreComponent and MidComponent, our algorithm adopts Forward_Match and Backward_Match strategy respectively, because the right boundary of PreComponent and the left boundary of MidComponent are fixed.

Take one sentence of MEDLINE abstract (PMID = 12541321) for instance to explain the algorithm, *"We also evaluated the expression of the CCR1 ligand macrophage inflammatory protein-1alpha (MIP-1alpha/CCL3)."* If the *anchor*$_g$ is "*macrophage inflammatory protein-1alpha*", the MidComponent, "*MIP-1alpha/ CCL3*", becomes the definition candidate *def*$_g'$ for the *anchor*$_g$, and then ***m**acrophage* matches ***M***, ***i**nflammatory* matches ***I***, ***p**rotein* matches ***P***, and ***alpha*** matches ***alpha*** from left to right, naming Forward_Match. On the other hand, if the *anchor*$_g$ is "*MIP-1alpha*", the PreComponent, "*We also evaluated the expression of the CCR1 ligand macrophage inflammatory protein-1alpha"*, becomes the definition candidate *def*$_g'$ for the *anchor*$_g$, and then ***alpha*** matches ***alpha***, ***P*** matches ***p**rotein*, ***I*** matches ***i**nflammatory*, and ***M*** matches ***m**acrophage* from right to left, naming Backward_Match. No matter which cases above are processed by the algorithm, the pair "*macrophage inflammatory protein-1alpha*"-"*MIP-1alpha*" is returned as the result of ExGlossary.

In order to illuminate the algorithm, the matching rule applied in the above example is quite simple (i.e. matching the first letters of the words), however, much more rules guided by genomic nomenclature (see Table 2.) are designed in our system, such as (1) any number and special character (e.g. +, -, /) are ignored for matching; (2) match consecutive letters, e.g. ***ACO1***, "***aco**nitase 1*"; (3) skip one word if matching the first letter of following word, e.g. ***TCP10L***, "***t-c**om**p**lex 10 mouse **l**ike*"; (4) match a middle letter of a word if the first letter of the word is matched and the first letter of the following word is not, e.g. ***IgM***, "***I**mmuno**g**lobulin **M***". All the rules are applied iteratively until the components are completely matched. The matched pairs become the strict result of genomic glossary while the dis-matched sentences conjunct with S_{pos} become the loose result.

Table 2. Guidelines for Genome Nomenclature

1.	Where a complete alternative gene name is being included as part of the name, this should be in the parentheses [21].
2.	Gene symbol should be short-form representation for the descriptive gene name [20, 21].
3.	Initial character of gene symbol should always be a letter [19-24].
4.	Gene symbol should not contain punctuation [20, 21].
5.	Greek symbols in gene name should be changed to their Latin alphabet [21].
6.	Gene symbol should not contain "G" for gene [20].
7.	Modifier of gene name should follow the main part of the name, separated by commas [21].
8.	Gene name of other species should be in parentheses at the end [21].

4 Experiment Settings and Result Analysis

4.1 Experiment Settings

In the genome nomenclature criteria, gene full name should be included in the abstract of any relevant papers [19]. The abstracts of MEDLINE, a famous biomedical bibliographic resource which covers 4,800 worldwide leading journals on biosciences [1], become the document collection in our experiment. We use LocusLink to simulate query sender in the experiment. The evaluation for the extracted glossary is through ad hoc retrieval task of Text REtrieval Conference (TREC) Genomics Track in year 2003 [26].

Topic of the ad hoc retrieval task can be described as: For gene X, find all MEDLINE references that focus on the basic biology of the gene or its protein products from the designated organism. The correct understanding of the gene X becomes a key point in this task.

In the following experiment, gene X triggers the glossary extraction firstly, and then the generated glossary items together with the original topic are input into the search engine [27] to get the relevant document list. 50 batch topics in test data set and relevance judgments (called “*qrels*” in TREC jargon) for them are involved in our experiment, thanks to TREC release.

Fig. 2. shows our genomic glossary evaluation through RECALL performance at each topic in TREC2003 ad hoc retrieval task.

$$\text{Recall} = |Ra|/|R|$$

Recall is the fraction of the relevant documents which has been retrieved [25]. Where, |R| is the number of relevant documents, and |Ra| is the number of relevant and retrieved documents.

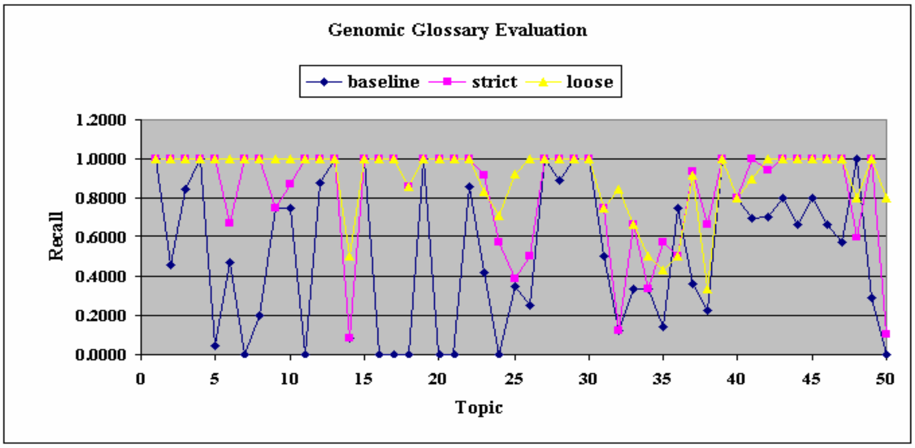


Fig. 2. Genomic Glossary Evaluation through TREC2003 Genomics Track Ad Hoc Task

4.2 Result Analysis

In Fig. 2, “baseline”, “strict” and “loose” represent the retrieval results of the gene names in LocusLink (naming *initial*$_q$), extracted glossary at strict level conjunct with *initial*$_q$ (naming *Instrict*$_q$), and extracted glossary at loose level conjunct with *initial*$_q$ (naming *Inlosse*$_q$) respectively. Query-triggered genomic glossary technology improves the recall performance of IR system, especially at the point that *initial*$_q$ fails to hit any documents in the collection, taking topic 11 for example to explain how our methods succeed in breaking the bottleneck of such topic. Table 3. shows the gene names (the third column) and their name types (the second column) in the topic 11. None of the four names are mentioned in MEDLINE (4/1/2002 -- 4/1/2003), so topic 11 gets 0 recall at baseline.

Table 3. Topic 11 in the Test Topic Set of TREC2003 Genomics Track Ad Hoc Retrieval Task

11	OFFICIAL_GENE_NAME	tissue inhibitor of metalloproteinase 2
11	OFFICIAL_SYMBOL	TIMP2
11	PREFERRED_PRODUCT	tissue inhibitor of metalloproteinase 2 precursor
11	PRODUCT	tissue inhibitor of metalloproteinase 2 precursor

As certain rules in our query processor ignore numeric factor, “*tissue inhibitor of metalloproteinase*” becomes one variant of triggering query “*tissue inhibitor of metalloproteinase 2*” after query processing (see section 3.2); the following sentence is returned as candidate sentence “*tissue inhibitor of metalloproteinase-2 (TIMP-2) in ectopic and eutopic endometrium from women with and without endometriosis throughout the menstrual cycle.*” (PMID = 12372458); pattern detector selects the above sentence as one member of sentence set S_{par} (see section 3.3); in the ExGlossary algorithm, “*tissue inhibitor of metalloproteinase*” is the *anchor*$_g$, and “*TIMP-2*” is mined out as *def*$_g$ (see section 3.4). This is one of effective procedures for topic 11, as “*TIMP-2*”is the common expression in document collection, and is used in all the relevant documents of this topic. After the strict-level glossary applied, the average recall of 50 topics improves to 83.20% from *baseline* 52%.

The difference between the results of *Instrict*$_q$ and *Inlosse*$_q$ is not as significant as the one between *baseline* and *Instrict*$_q$. Although the average recall of *Inlosse*$_q$ increases to 90.15%, its performances at some topics are not better than *Instrict*$_q$ (e.g. topic 24, topic 35). The reason is that *Inlosse*$_q$ consists of all the terms in *anchor*$_g$, S_{par} and S_{pos}, and brings in so many noises that the relevant documents cannot rank into retrieved document set. For ranking algorithm and query term weighting strategy, they are the more complex problems in IR field [25]. In order to focus on the evaluation of extracted glossary, all above three retrieval results are based on the same ranking algorithm and term weighting strategy.

From the above experiment settings and result analysis, we can conclude that our query-triggered genomics glossary improves the recall performance of information retrieval system in TREC2003 Genomics Track ad hoc retrieval task.

5 Conclusions and Future Works

In this paper we present a novel framework to extract genomic glossary automatically, where a series of document indexing methods and rule-based algorithms are applied to support the query-triggered and document-centralized biomedical text mining. The evaluation trough TREC Genomics Track indicates our glossary's contribution in IR task (83.20% recall at strict level, 90.15% recall at loose level). Our system may provide a useful supplement for the public genome resource (e.g. LocusLink, MeSH) to identify the new expressions of gene in the latest literature, and assist the manual work for curators in genomic field.

In our future works, more computational linguistic techniques will be imported, especially at the steps of the pattern detecting and the loose-level result processing. Some helpless rules, which may lead to low system efficiency, are to be optimized in next phase. On the other hand, the glossary evaluation experiments will be enhanced, such as attempt to compare with other works in KDD Cup 2002 Task1: information extraction from biomedical article [29].

Acknowledgements

The work was supported by National Natural Science Foundation of China under grant No.60572084 and 60321002.

References

[1] MEDLINE, http://www.nlm.nih.gov/pubs/factsheets/medline.html, 2005
[2] GenBank, http://www.ncbi.nlm.nih.gov/Genbank/, 2005
[3] J.H.Chiang and H.H. Yu. (2003) MeKE: discovering th functions of gene products form biomedical literature via sentence alignment, Bioinformatics, 19(11), pp. 1417-1422.
[4] Pruitt KD et al. Introducing RefSeq and LocusLink: curated human genome resource at the NCBI. Trends Genet 2000;16(1):44-7.
[5] LocusLink Home Page. http://www.ncbi.nih.gov/LocusLink, 2004
[6] J. Pustejovsky, J. Castaño, R. Saurí, A. Rumshisky, J. Zhang, W. Luo. Medstract: Creating Large-scale Information Servers for Biomedical Libraries. ACL 2002 Workshop on Natural Language Processing in the Biomedical Domain. Philadelphia, PA.
[7] The Medstract Project - AcroMed 1.1 http://medstract.med.tufts.edu/acro1.1/, 2005
[8] Donal G. Gilbert. euGenes: A Eukaryote Genome Information System. Nucletic Acids Research, 30(1):145-148, 2002.
[9] Genomic Information for Eukaryotic Organisms. http://eugenes.org/, 2005
[10] U.S. National Library of Medicine Medical Subject Headings (MeSH) Home Page. http://www.nlm.nih.gov/mesh/meshhome.html 2005
[11] L. Humphreys, D.A.B. Lindberg, H.M. Schoolman, and G.O. Barnett. 1998. The Unified Medical Language System: An Informatics Collaboration. Journal of the American Medical Informatics Association, 1(5):1-13.
[12] Unified Medical Language System (UMLS), http://www.nlm.nih.gov/research/umls/
[13] J. Pustejovsky, J. Castaño, B. Cochran, M. Kotecki, M. Morrell, A. Rumshisky. Linguistic Knowledge Extraction from Medline: Automatic Construction of an Acronym Database. Medinfo, 2001

[14] Chang JT, Schütze H, and Altman RB (2002). Creating an Online Dictionary of Abbreviations from MEDLINE. The Journal of the American Medical Informatics Association. 9(6): 612-20.
[15] Biomedical Abbreviation, http://abbreviation.stanford.edu/, 2005
[16] Yu, H., V. Hatzivassiloglou, A. Rzhetsky, and W.J. Wilbur, Automatically identifying gene/protein terms in MEDLINE abstracts. J Biomed Inform, 2003. 35(5-6): p. 322-330.
[17] Hisamitsu T, Niwa Y. Extraction of useful terms form parenthetical expression by using simple rules and statistical measures. Proceedings of the First Workshop on Computational Terminology, Compu Term '98 (Montreal, Ontario; Aug 15, 1998). 1998: 36-42.
[18] Kenji Satou, Kaoru Yamamoto, Utilizing weakly controlled vocabulary for sentence segmentation in biomedical literature. In Silico Biology 5: (2004)
[19] Kohli J. Genetic, Nomenclature and Gene List of the Fission Yeast, Schizosaccharomyces pombe. Curr Genet 1987; 11(8): 575-89.
[20] Hester M. Wain, Elspeth A. Bruford, Ruth C. Lovering, Michael J. Lush, Mathew W. Wright and Sue Povey, Guidelines for Human Gene Nomenclature, Genomics 2002, 79(4):464-470.
[21] HUGO Gene Nomenclature Committee, http://www.gene.ucl.ac.uk/nomenclature/. 2005
[22] Maltais LJ et al. Rules and Guidelines for mouse gene nomenclature: a condensed version. International committee on standardized genetic nomenclature for mice. Genomics 1997, 45(2): 471-6
[23] Antonarakis SE. Recommendations for a nomenclature system for human gene mutations. Nomenclature working group. Hum Mutat 1998, 11(1):1-3
[24] Horvitz HR et al. A Uniform Genetic Nomenclature for the Nematode Caenorhabditis Elegans. Mol Gen Genet 1979; 175 (2):129-33.
[25] Ricardo Baeza-Yates, Berthier Riberiro-Neto, Modern Information Retrieval, ACM Press, 1999, p24-138.
[26] William R. Hersh, Ravi TB. TREC Genomics Track Overview. The Twelfth Text Retrieval Conference: TREC2003. 2003. Gaithersburg, MD: National Institute of Standards and Technology.
[27] Jiao Li, Xian Zhang, Min Zhang and Xiaoyan Zhu:" THUIR at TREC 2004: Genomics Track", Proceedings of 13th Text Retrireval Conference (TREC2004), pp 571-575, Gaithersburg, USA, November 2004.
[28] Klavans J, Muresan S. Evaluation of the DEFINDER System for Full Automatic Glossary Construction. Proceedings of the AMIA Symposium, 2001.
[29] Alexander S. Yeh, L.H., Alexander A. Morgan, Background and Overview for KDD Cup 2002 Task 1: Information Extraction from Biomedical Articles. SIGKDD Explorations, 2002. 4(2): p. 87-89.

Frequent Subsequence-Based Protein Localization*

Osmar R. Zaïane, Yang Wang, Randy Goebel, and Gregory Taylor

University of Alberta, Edmonton ALB, Canada
{zaiane, wyang, goebel}@cs.ualberta.ca

Abstract. Extracellular plant proteins are involved in numerous processes including nutrient acquisition, communication with other soil organisms, protection from pathogens, and resistance to disease and toxic metals. Insofar as these proteins are strategically positioned to play a role in resistance to environmental stress, biologists are interested in proteomic tools in analyzing extracellular proteins. In this paper, we present three methods using frequent subsequences of amino acids: one based on support vector machines (SVM), one based on boosting and FSP, a new frequent subsequence pattern method. We test our methods on a plant dataset and the experimental results show that our methods perform better than the existing approaches based on amino acid composition.

1 Introduction

Proteins are the molecules that accomplish most of the functions of the living cell. All proteins are composed of linear sequences of smaller molecules called amino acids. There are twenty naturally occurring amino acids. Long proteins may contain a chain of as many as 4500 amino acids. Finding the proteins that make up a creature and understanding their functions is the foundation of explanation in molecular biology [11]. With the introduction of large-scale sequencing, biologists have accumulated an immense volume of raw biological sequences that are publicly available. In order to better understand the functions and structures of these protein sequences, a vitally important problem facing the biology community is to classify these sequences into different families based on the properties of the sequences, such as functions, structures, etc.

Protein sub-cellular localization is a key functional characteristic of proteins. In order to execute a common physiological function, proteins must be localized in the same cellular compartment. Proteins may be localized at various locations within the cell or be transported to the extracellular space. The process through which proteins are routed to their proper sub-cellular localizations is called sub-cellular protein sorting. Protein sorting is the simplest in gram positive prokaryotes, where proteins are only directed to the cytoplasm, the plasma membrane, the cell wall, or secreted to the extracellular space. Gram negative protein localization sites include the cytoplasm, the inner membrane, the

* Research funded in part by the Alberta Ingenuity Funds and NSERC Canada.

periplasm, the outer membrane, and the extracellular space. Sub-cellular localizations in eukaryotic proteins are much more complex due to the presence of membrane-bound organelles. The major location sites for eukaryotic proteins include the plasma membrane, the nucleus, the mitochondria, the peroxisome, the endoplasmic reticulum, the Golgi apparatus, the lysosome, the endosome, and others (such as chloroplasts, vacuoles, and the cell wall in plant cells).

The sub-cellular localization of a protein plays an important role with regard to its function. Knowledge of sub-cellular localization can provide valuable information concerning its possible functions. It can also help in analyzing and annotating sequences of hypothetical or known gene products. In addition, it can influence the design of experimental strategies for functional characterization [5].

Since the number of collected sequences has been rapidly increasing, it is time consuming and costly to approach this problem of predicting the sub-cellular localization of a protein entirely by performing various biological experimental tests. In view of this, it is highly desirable to develop some algorithms to rapidly predict the sub-cellular localizations of proteins.

Herein, we are particularly interested in identifying those proteins that are secreted to the extracellular environment (called *extracellular proteins*), versus proteins localized at various locations within the cell (called *intracellular proteins*) in plants. Extracellular plant proteins are involved in numerous processes including nutrient acquisition, communication with other soil organisms, protection from pathogens, and resistance to disease and toxic metals. Insofar as these proteins are strategically positioned to play a role in resistance to environmental stress, biologists are interested in proteomic tools in analyzing them [26].

A number of methods have been developed in the bioinformatics community for predicting protein sub-cellular localizations. They can be classified into three major approaches based on the features used in the learning algorithms. The first approach is based on "sorting signals", which are short subsequences of approximately 3 to 70 amino acids. For example, SignalP [17, 18] and TargetP [6] use neural networks to identify the sorting signals. The accuracy of SignalP is 68% for human proteins, 70.2% in Eukaryote, 83.7% in E.coli, 79.3% in Gram-negative bacteria and 67.9% in Gram-positive bacteria. TargetP achieves an accuracy of 85% in plants and 90% in non-plant proteins. The second approach is based the amino acid composition. The amino acid composition of a protein sequence refers to the relative frequencies of 20 different amino acids. Each protein is represented by a histogram with 20 bins, regardless of the length of the protein. NNPSL [19] uses neural network and SubLoc [10] uses support vector machines(SVM) to learn the predictors based on amino acid composition. The accuracy of NNPSL is 66% in Eukaryotes excluding plants, and 81% in prokaryotes. The accuracy of SubLoc is 91.4% in prokaryotes and 79.4% on eukaryotes. The third approach, e.g. LOCKey [15] and PA-sub [14], uses the textual information associated with a protein (available in Swiss-Prot [3]) to learn the predictor. Some tools (e.g. PSORT [16]) take an integrative approach by combining several different methods. LOCKey achieves an accuracy of 87% on their test data extracted from Swiss-Prot. PA-sub achieves an overall accuracy of about 98% in all of their datasets (including animal, archea, fungi, plant, Gram-positive bacteria and Gram-negative bacteria).

Recently, She *et al.* [22] proposed some methods for outer membrane protein classification based on frequent subsequences. Their results have shown that frequent-subsequence-based methods perform better than other methods in the biological domain using precision as a measure. In this paper, we use similar ideas for the problem of extracellular plant protein prediction.

In our work, we use support vector machines as well as boosting using frequent subsequences of amino acids, then combine them with amino acid composition of proteins to improve accuracy. We also introduce a promising new approach FSP specifically designed for frequent subsequences.

2 Predicting Extracellular Proteins

While modeling proteins with histograms representing the amino acid composition has been shown successful [19, 10, 2, 8], we found that the amino acid composition loses discriminant power for plant proteins. Instead we model a protein by a set of frequent subsequences it contains. Our hypothesis is that frequent subsequences of amino acids are better descriptors to discriminate between extracellular and intracellular plant proteins. In this section, after introducing the features used in the training algorithms, we introduce three different methods for extracellular plant protein prediction.

2.1 Feature Extraction

We use *frequent subsequences* as the features for the learning algorithms. A frequent subsequence is a subsequence made up of consecutive amino acids that occurs in more than a certain fraction (*MinSup*) of extracellular proteins. The reason we choose frequent subsequences is based on the following observations:

– Subsequences that appear frequently in extracellular proteins and rarely appear in intracellular proteins have very good discriminative power for identifying extracellular proteins and can be of great interest to biologists.
– It has been known that common subsequences among related proteins may perform similar functions via related biochemical mechanisms [7].
– Frequent subsequences capture the local similarity that may relate to important functional or structural information of extracellular proteins.

There are algorithms for finding frequent subsequences in a set of sequences using generalized suffix trees (GST) [25]. A GST is a trie-like structure designed for compactly representing a set of strings. Each suffix of the string is represented by a leaf in the GST. Each leaf is associated with an index i. The edges are labeled with character strings such that the concatenation of the edge labels on the path from the root to the leaf with index i is a suffix of the ith string in the set. There are algorithms that can construct the GST for a set of strings in linear time [9]. After a GST is constructed, it is traversed in order to find frequent subsequences.

2.2 SVM Method

The first method we use is based on support vector machines (SVM) [24]. SVM is well founded theoretically because it is based on well developed statistical learning theory. It has also been shown to be very effective in real-world applications.

In order to use SVM, the input data have to be in the form of vectors. Each protein sequence is transformed into an n-dimensional vector $\boldsymbol{x} = (a_1, a_2, ..., a_n)$, where n is the number of frequent subsequences found from extracellular proteins, and $a_j (1 \leq j \leq n)$ is the feature corresponding to the ith subsequence. A binary representation is used. If the ith subsequence appears in protein sequence $\boldsymbol{x}$, the value of a_j is set to 1. Otherwise, it is set to 0. For the class label, +1 is used to indicate extracellular proteins and -1 for intracellular proteins.

We can train the SVM using different kernel functions. A kernel function $\Phi(\boldsymbol{x})$ maps the input vector $\boldsymbol{x}$ into a higher dimensional space. Nonlinear separators for the original data can be found by a linear separator in this higher dimensional space. Classical kernel functions include:
Linear Kernel Function: $K(\boldsymbol{x}_i, \boldsymbol{x}) = \boldsymbol{x}_i \cdot \boldsymbol{x}$; Polynomial Kernel Function: $K(\boldsymbol{x}_i, \boldsymbol{x}) = (\boldsymbol{x}_i \cdot \boldsymbol{x} + 1)^d$; and Radial Basic Function(RBF): $K(\boldsymbol{x}_i, \boldsymbol{x}) = exp(-\gamma \parallel \boldsymbol{x}_i - \boldsymbol{x} \parallel^2)$.

2.3 Boosting Method

Boosting is a meta-learning method that has a theoretically justified ability to improve the performance of any *weak classifier*. A weak classifier is an algorithm that, given $\epsilon, \delta > 0$ and access to random examples, can achieve at least slightly better error rate ϵ than random guessing ($\epsilon > 1/2 - \gamma$, where $\gamma > 0$), with a probability $(1 - \delta)$. The purpose of boosting is to build a highly accurate classifier by combining many *weak* or *base* hypotheses, each of the weak hypothesis may be only moderately accurate. Various different boosting algorithms have been proposed in the literature [4, 20, 23].

Boosting algorithms work iteratively. During each iteration, a classifier is learned based on a different weighted distribution of the training examples. The main intuition behind boosting algorithms is to increase the weights of the incorrectly classified examples and decrease the weights of the correctly classified examples. This forces the learning algorithm to focus on those examples that are not correctly classified in the next iteration. The algorithm usually stops after a pre-specified number of iterations, or it can stop when some measurement of the quality of the classifier based on certain measurement (such as error rate) starts to deteriorate. The set of classifiers obtained after these iterations are combined together for the final prediction of unseen examples.

In our application of extracellular protein prediction, we use AdaBoost [20] with simple rule-based classifiers as the weak hypotheses. Every rule is a simple check for the presence or absence of a frequent subsequence in a protein primary sequence. Based only on the outcome of this test, the weak hypothesis outputs the prediction and the confidence that each label ("extracellular" or "intracellular") is associated with the protein sequence.

If we denote the possible class label for a protein sequence x by l and define $a \in x$ to represent the fact that subsequence a appears in protein sequence x, the weak hypothesis corresponding to this subsequence has the following form:

$$h(x,l) = \begin{cases} c_{0l} \text{ if a} \in \text{x} \\ c_{1l} \text{ if a} \notin \text{x} \end{cases}$$

where the c_{jl} are real numbers. The weak learner searches all possible frequent subsequences. For each subsequence, it calculates the values c_{jl} and assigns a score. Once all the subsequences are searched, the weak hypothesis with the lowest score is returned by the weak learner. In our case, the score is an exact calculation of Z_t (refer to [20] for details). The score is calculated as follows (refer to [21] for details):

Let $X_0 = \{x : w \notin x\}$ and $X_1 = \{x : w \in x\}$. For $j \in \{0,1\}$ and for $b \in \{-1,+1\}$, we calculate the following based on the current distribution D_t:

$$W_b^{jl} = \sum_{i=1}^{m} D_t(i,l)\{x_i \in X_j \wedge Y_i[l] = b\}$$

Z_t is minimized for a particular term by choosing $c_{jl} = \frac{1}{2}ln(\frac{W_{+1}^{jl}}{W_{-1}^{jl}})$ and by setting α_t=1. These settings imply that

$$Z_t = 2 \sum_{j\in\{0,1\}} \sum_{l\in\mathcal{Y}} \sqrt{W_{+1}^{jl} W_{-1}^{jl}}$$

After all frequent subsequences are searched, the weak learner returns the one for which the value of Z_t is the smallest.

2.4 Frequent Subsequence Pattern (FSP) Method

She et al. proposed a rule-based classification based on *frequent patterns*, which have the form $*X_1 * X_2 * ...$, where $X_1, X_2, ...$ are frequent subsequences made up of consecutive amino acids, and "*" is a variable-length-don't-care (VLDC) that can substitute for zero or more letters when matching the pattern against a protein sequence [22]. Their method finds a set of frequent patterns that discriminate outer membrane proteins (OMP) from non-OMPs. In the classification stage, if a protein matches one of the frequent patterns, it is classified as OMP. Otherwise it is classified as non-OMP. We adopt a similar idea in our method, but with the following modification.

Consider a pattern P=$*X_1 * X_2*$ that appears in two different sequences S_1 and S_2 such that X_1 and X_2 are close to each other in S_1 while they are too far apart in S_2. Intuitively, the match in S_1 is more likely to be biologically significant. In our algorithm, we introduce another parameter called $MaxGap$. When matching a pattern against a protein sequence, if the distance (in terms of number of amino acids) of two adjacent subsequences are too far apart, we do not consider it to be a match. For example, if $MaxGap$ is set to be 3, the pattern "*ABC*DEF*" does not match the sequence "ABCMNOPQDEF", since the gap between subsequence "ABC" and "DEF" is 5 (see Figure 1(a)). However the pattern "*ABC*DEF*" matches the sequence "ABCABCPQDEF", since we can find a way to align them, so that the gap between "ABC" and "DEF" is 2. In this paper, we call the pattern with $MaxGap$ to be *frequent subsequence pattern*, and the pattern without $MaxGap$ to be *frequent pattern*.

Algorithm 1. FSP: Algorithm for Finding Patterns

```
Input: Training set $D = P \cup N$. ($P$ and $N$ are the sets of extracellular and intracellular proteins)
Output: $R$ a set of patterns in the format of $*X_1 * X_2 * \dots$ for predicting extracellular proteins
Parameters: $\alpha$: rate of weight decrease; $\delta$: threshold total weight; $min\_Znumber$: minimum acceptable Z-number; $MaxGap$: maximum gap between two subsequences.
Method:
set the weight of every example in P to 1
pattern set $R \leftarrow \emptyset$
$totalWeight \leftarrow TotalWeight(P)$
while $totalWeight > \delta \cdot totalWeight$ do
  $N^{'} \leftarrow N, P^{'} \leftarrow P$
  pattern $r \leftarrow empty\_rule$
  while true do
    Choose the subsequence $p$ with the largest Z-number, according to $N^{'}$ and $P^{'}$
    if $Z\_number(p) < min\_Znumber$ then
      break
    end if
    append $p$ to $r$
    for each example $t$ in $P^{'} \cup N^{'}$ do
      if not Match($t$, $r$, $MaxGap$) then
        remove $t$ from $P^{'} \cup N^{'}$
      end if
    end for
  end while
  $R \leftarrow R \cup \{r\}$
  for each example $t$ in $P$ do
    if Match($t$, $r$, $MaxGap$) then
      $t.weight \leftarrow \alpha \cdot t.weight$
    end if
  end for
  $totalWeight \leftarrow TotalWeight(P)$
end while
return $R$
```

She et al. use an exhaustive search to build frequent patterns to identify outer membrane proteins by concatenating two or more frequent subsequences [22]. However, since there could be thousands of subsequences found in the training set, an exhaustive search produces an explosive number of candidate patterns. To deal with this problem, we exploit a greedy algorithm to find those patterns. We search for the current best rule and reduce the weights of the positive examples that are covered by this rule, until the total weight of the positive examples are less than a certain threshold (Algorithm 1). The procedure $Match(t, r, MaxGap)$ in Algorithm 1 is implemented by enumerating all the possible alignment of the subsequences in the pattern r to the sequence t. The pattern r is considered to "match" sequence t, if there is one possible alignment, such that the distances between two adjacent subsequences are all less than $MaxGap$.

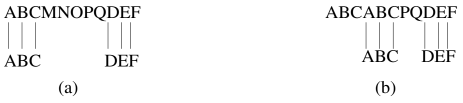


Fig. 1. Matching pattern against sequence

The *Z-number* in Algorithm 1 is calculated as follows. Given a rule R and s_R denotes its support, let a_C denote the mean of the target class C, defined as $a_C = |S_C|/|S|$, where S is the current training set and S_C is the subset of S where C is the class label. Let σ_C denote the standard deviation of the target class C. In the binary classification problem, it is calculated as $\sigma_C = \sqrt{a_C(1 - a_C)}$. Using these notions, Z-number is defined as $Z_R = \sqrt{s_R}(a_R - a_C)/\sigma_C$. The Z-number measures how well a rule R discriminates examples of class C [13]. It is similar to the z-test or t-test in statistics. A rule with high positive Z-number predicts the presence of C with high confidence. A rule with high negative Z-number predicts absence of C with high confidence. A rule with Z-number close to zero does not have much power of discriminating examples of class C.

After the set of patterns are generated, we filter them in order to keep those patterns with good predictive power. Only those patterns with support greater than a threshold $MinSup$ and confidence greater than $MinConf$ are kept for predicting unseen protein examples. The prediction process is relatively easy. Given an unseen example t, every pattern r in the pattern set is tested. If there exist a pattern r that matches t, t is predicted to be an extracellular protein, otherwise it is predicted to be an intracellular protein.

3 Experimental Results

Our hypothesis is that frequent subsequences of amino acids are better discriminant than amino acid composition for distinguishing between intracellular and extracellular plant proteins. We compare our methods including SVM based on frequent sequences, boosting based on frequent subsequence, and our frequent subsequence pattern method (FSP) with SVM based on amino acid composition and boosting based on amino acid composition. We also investigate the effect of combining subsequences and amino acid composition in the same classifier.

3.1 Dataset and Evaluation

We tested the performance of our methods on a plant protein dataset that we received from the Proteome Analyst project [14] at the University of Alberta. This dataset contains 3293 proteins, among which 171 are extracellular proteins.

We performed 5-fold cross validation, i.e., each run takes one of the 5 folds as the test set and the remaining 4 folds as the training set. To ensure fair comparisons, all the methods are evaluated using the same folding.

The performance of a classification algorithm is usually evaluated by its *overall accuracy*. However, in our application, overall accuracy is not a good evaluation metric since in our dataset, only about 5% of the proteins are extracellular proteins. A high accuracy (95%) can easily be achieved by classifying every protein to be intracellular. Instead, we choose to use *precision*, *recall* and *F-measure* with respect to the rare class (extracellular proteins) as our evaluation metrics. They are based on the confusion matrix shown in Table 1. Using the notions in Table 1, precision (P) and recall (R) of extracellular class can be defined as:

$$P = \frac{TP}{TP + FP}, R = \frac{TP}{TP + FN}$$

Table 1. Confusion Matrix

	Actual Extracellular	Actual Intracellular
Predicted as Extracellular	TP	FP
Predicted as Intracellular	FN	TN

The F-measure is a harmonic average of precision and recall: $F = \frac{2PR}{P+R}$.

For all the experiments, the subsequences are obtained by setting the minimum support threshold to be 5%. The numbers of subsequences in each fold are: 2658, 2605, 2532, 2817, 2722 for folds 1 to 5 respectively.

3.2 Experimental Result of SVM

We used the SVM^{light} implementation [12] since it is well-known and has been used extensively in previous research. We tried with three different kernels, including the linear kernel, the polynomial kernel with degree of 2 and the radial basis function kernel with γ=0.005. For each kernel, we tried different values for C (the regularization parameter that controls the trade-off between margin and misclassification error). The best result (in terms of F-measure) using frequent subsequences is **0.804** with a linear kernel. We compared our method with SubLoc [10]. SubLoc uses SVM with amino acid composition as its features. The authors show that SubLoc performs better in terms of accuracy compared with other methods based on amino acid composition. It also performs better than methods based on N-terminal signals. SubLoc is not specifically designed for predicting extracellular proteins, but since its implementation is based on SVM^{light}, we re-implemented it with SVM^{light} and tested it on our dataset.

We tried the same parameter settings as we did in SVM with subsequences. The best result obtained is only **0.522** with a polynomial kernel (d=2) and C=1000.

3.3 Experimental Result of Boosting

In the experiments with boosting, we chose the number of iterations to be 500, 1000 and 2000. The results show that the boosting algorithm is robust with respect to the number of iterations. The best result obtained by boosting using frequent subsequences is **0.729** with 1000 iterations. For the purpose of comparison, we also tried AdaBoost based on amino acid composition. Since the attributes are continuous values in this case, the weak hypothesis used is a single test of whether the composition of an amino acid is above or below some threshold (see [21] for details). The best result obtained is a mediocre **0.574** with 1000 iterations.

3.4 Experimental Result of the FSP Method

For the experiments using the frequent subsequence pattern (FSP) method, there are quite a few parameters to be tuned. In order to tune those parameters, we took a portion of the training examples and tried our algorithm with different parameter settings, then tested the learned model on another portion of the examples. Through trial and error, we identified the following parameter setting: $MinLen$ set to 3, min_gain to 0.1, δ to 0.03 and α to 0.8. The $MinSup$ is set to 5%, $MinConf$ to 80%, and $MaxGap$ to 300. We obtained a precision of **0.765**, a recall of **0.614** and an F-measure of **0.681**.

Even though SVM based on frequent subsequences achieves the best experimental result, there are some advantages in using the FSP method. The reason is that the decision functions learned by SVM algorithms are difficult for people to understand. The discovered hyperplane is difficult to manipulate. However, the decision rules found by the FSP method can be easily interpreted and modified by human experts. Figure 2 shows some examples of the rules found by the FSP method. Biologists can easily read these rules and determine whether they are biologically meaningful. They can also incorporate their biological knowledge and modify the patterns, e.g., by adding or removing subsequences in the patterns, to get even better classification models. This study is currently in progress.

```
IF (sequence contains *CKN*CGPGHGIS*) THEN (extracellular)
IF (sequence contains *YWGQNG*EIN*) THEN (extracellular)
IF (sequence contains *QVY*AGH*NVT*) THEN (extracellular)
...
ELSE (intracellular)
```

Fig. 2. Examples of patterns found by the FSP method

3.5 Combining Frequent Subsequences and Amino Acid Composition

It is clear that the methods based on frequent subsequences perform better than those based on amino acid composition. However, we can still take advantage of the information represented in the amino acid composition histograms by combining these two features. We investigated this possibility. Interestingly, we found that SVM does not improve at all. The result shows that there is no obvious benefits of combined features for SVM. In other words, SVM could not take advantage of the additional information. In the case of the RBF kernel, SVM based on combined features deteriorated. The additional data (amino acid composition) created noise.

Contrary to SVM, the performance of boosting (measured by F-measure) can be improved significantly by combining frequent subsequences and amino acid composition. As can be seen in Table 2 and Table 3 Boosting using combined features gives a better result than the best result of SVM with frequent subsequences reaching **0.831**. Boosting better exploits this additional data regarding the amino acid composition when added to the frequent subsequences.

Table 2. AdaBoost classification with combined features

Number of iterations	Recall	Precision	F-measure
500	0.685	0.967	0.802
1000	0.717	0.989	**0.831**
2000	0.708	0.989	0.826

Table 3. Comparison of AdaBoost based on different features

Number of iterations	Combined feature	Subsequence	Composition
500	0.802	0.714	0.562
1000	**0.831**	**0.729**	**0.574**
2000	0.826	0.726	0.548

Since amino acid composition is represented by fractional numbers (i.e. the histogram), there is no easy way to merge frequent subsequences and amino acid composition in the FSP method. Thus, we did not combine amino acid compositions in our FSP algorithm. However, a new model to represent this information is worth investigating.

For cross comparison, we chose the best (in terms of F-measure) result generated by each algorithm (i.e., 0.804 for SVM with subsequences, 0.729 for boosting with subsequences, 0.522 for SVM with amino acid composition, 0.574 for boosting with amino acid composition). The comparison of different algorithms is shown in Figure 3. Our methods based on frequent subsequences are better than methods based on amino acid composition. In particular, Boosting with a combination of frequent subsequences and amino acid composition performs the best among the different approaches reaching an F-measure of **0.831**.

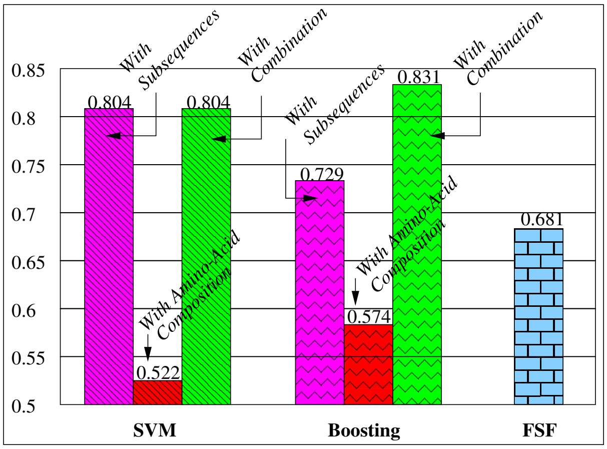


Fig. 3. F-measures of different algorithms

4 Conclusion and Future Work

We present in this paper several methods for identifying extracellular plant proteins using frequent amino acid subsequences. Our experimental results show that our methods perform better than amino acid composition-based methods. The best result is actually achieved by combining frequent amino acid subsequences and amino acid composition using Boosting. Combining these two features is not always beneficial. It was indeed detrimental in the case of SVM.

Even though the experimental results show SVM and boosting based on frequent subsequences as being the best approaches, there are advantages in using our new FSP method. The main reason being that contrary to SVM for example, the decision functions of the FSP method are easily readable rules that can be easily understood, interpreted and edited by human experts. Moreover, FSP is extendable. While it can not accommodate amino acid composition for the moment, additional information such as location of frequent subsequences, and constraints on their sizes could be combined in the algorithm.

There are a number of directions for possible future research. First of all, we only use the protein primary sequences for training the predictor of extracellular proteins. If additional properties of proteins (e.g., secondary structures, functions) are available, future research can take these characteristics into account to make a more accurate prediction.

With respect to frequent subsequences of amino acid, one important feature is the location of the subsequence within the protein. Biologists believe that the position of a frequent subsequence, in the beginning, the end, or other, within the protein can provide some indication regarding the protein itself. We are currently investigating the combination of frequent subsequences, amino acid composition, and the relative subsequence positions to build a more robust classifier. In particular, we are dividing a protein sequence into percentiles

(50%, 25%, and 10%) and labeling the relative position of a frequent subsequence by the portion in the protein where the subsequence starts. One good model that lends itself to this type of combinations is the associative classifier [1]. Based on associations rules, classification rules can be learned from proteins modeled into transactions of features. These rules are also easily understood and potentially modifiable by human experts in include additional domain knowledge.

References

1. M.-L. Antonie, O. R. Zaïane, and A. Coman. *Mining Multimedia and Complex Data*, chapter Associative Classifiers for Medical Images, pages 68–83. Lecture Notes in Artificial Intelligence 2797. Springer-Verlag, 2003.
2. M. Bhasin and G. Raghava. Eslpred: SVM-based method for subcellular localization of eukaryotic proteins using dipeptide composition and psi-blast. *Nucleic Acids Research*, 32:W414 – W419, July 2004.
3. B. Boeckmann, A. Bairoch, R. Apweiler, M.-C. Blatter, A. Estreicher, E. Gasteiger, M. J. Martin, K. Michoud, C. O'Donovan, I. Phan, S. Pilbout, and M. Schneider. The Swiss-Prot protein knowledgebase and its supplement TrEMBL in 2003. *Nucleic Acids Research*, 31:365–370, 2003.
4. W. Cohen and Y. Singer. A simple, fast and effective rule learner. In *Proceedings of Annual Conference of American Association for Artificial Intelligence*, pages 335–342, 1999.
5. F. Eisenhaber and P. Bork. Wanted: subcellular localization of proteins based on sequence. *Trends in Cell Biology*, 8:169–170, 1998.
6. O. Emanuelsson, H. Nielsen, S. Brunak, and G. von Heijne. Predicting subcellular localization of proteins based on their N-terminal amino acid sequence. *Journal of Molecular Biology*, 300:1005–1016, 2000.
7. K. A. Frenkel. The human genome project and informatics. *Communications of the ACM*, 34(11):41–51, 1991.
8. A. Garg, M. Bhasin, and G. Raghava. Support vector machine-based method for subcellular localization of human proteins using amino acid compositions, their order, and similarity search. *Journal of Biological Chemistry*, 280(15):14427 – 14432, April 2005. Epub 2005 Jan 12.
9. D. Gusfield. *Algorithms on Strings, Trees and Sequences.* Cambridge University Press, 1997.
10. S. Hua and Z. Sun. Support vector machine approach for protein subcellular localization prediction. *Bioinformatics*, 17(8):721–728, 2001.
11. L. Hunter. *Artificial Intelligence and Molecular Biology.* AAAI Press, 1993.
12. T. Joachims. *Learning to Classify Text Using Support Vector Machines.* Kluwer, 2002.
13. M. V. Joshi, R. C. Agarwal, and V. Kumar. Mining needles in a haystack: Classifying rare classes via two-phase rule induction. In *Proceedings of ACM SIGMOD Conference*, pages 91–102, Santa Barbara, CA, 2001.
14. Z. Lu. Predicting protein sub-cellular localization from homologs using machine learning algorithms. Master thesis, 2002. Department of Computing Science, University of Alberta.
15. R. Nair and B. Rost. Inferring sub-cellular localization through automatic lexical analysis. In *Proceedings of the tenth International Conference on Intelligent Syetems for Molecular Biology*, pages 78–86. Oxford University Press, 2002.

16. K. Nakai. A knowledge base for predicting protein localization sites in eukaryotic cells. *Genomics*, 14:897–911, 1992.
17. H. Nielsen, J. Engelbrecht, and S. Brunak. A neural network method for identification of prokaryotic and eukaryotic signal peptides and prediction of their cleavage sites. *International Journal of Neural Systems*, 8:581–599, 1997.
18. H. Nielsen, J. Engelbrecht, S. Brunak, and G. von Heijne. Identification of prokaryotic and eukaryotic signal peptides and prediction of their cleavage sites. *Protein Engineering*, 10(1):1–6, 1997.
19. A. Reinhardt and T.Hubbard. Using neural networks for prediction of the subcellular location of proteins. *Nucleic Acids Research*, 26(9):2230–2236, 1998.
20. R. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.
21. R. Schapire and Y. Singer. BoosTexter: A boosting-based system for text categorization. *Machine Learning*, 39(2):135–168, 2000.
22. R. She, F. Chen, K. Wang, M. Ester, J. L. Gardy, and F. S. L. Brinkman. Frequent-subsequence-based prediction of outer membrane proteins. In *Proceedings of ACM SIGKDD Conference*, Washington, DC, USA, 2003.
23. K. M. Ting. A comparative study of cost-sensitive boosting algorithms. In *Proceedings of Intl. Conference on Machine Learning*, pages 983–990, 2000.
24. V. Vapnik. *Statistical Learning Theory.* Wiley, New York, 1998.
25. J. Wang, G. Chirn, T. Marr, B. Shapiro, D. Shasha, and K. Zhang. Combinatorial pattern discovery for scientific data: Some preliminary results. In *Proceedings of ACM SIGMOD Conference*, Minnesota, USA, 1994.
26. Y. Wang. EPPdb: a database for proteomic analysis of extracytosolic plant proteins. Master thesis, 2004. Department of Computing Science, University of Alberta.

gTRICLUSTER: A More General and Effective 3D Clustering Algorithm for Gene-Sample-Time Microarray Data[⋆]

Haoliang Jiang[1], Shuigeng Zhou[1,2], Jihong Guan[3], and Ying Zheng[1]

[1] Department of Computer Science and Engineering, Fudan University
[2] Shanghai Key Lab of Intelligent Information Processing, Fudan University,
220 Handan Road, Shanghai 200433, China
{hljiang, sgzhou}@fudan.edu.cn
[3] Dept. of Computer Sci. and Technol., Tongji University, Shanghai 200092, China
jhguan@mail.tongji.edu.cn

Abstract. Clustering is an important technique in microarray data analysis, and mining three-dimensional (3D) clusters in gene-sample-time (simply GST) microarray data is emerging as a hot research topic in this area. A 3D cluster consists of a subset of genes that are coherent on a subset of samples along a segment of time series. This kind of coherent clusters may contain information for the users to identify useful phenotypes, potential genes related to these phenotypes and their expression rules. TRICLUSTER is the state-of-the-art 3D clustering algorithm for GST microarray data. In this paper, we propose a new algorithm to mine 3D clusters over GST microarray data. We term the new algorithm gTRICLUSTER because it is based on a more general 3D cluster model than the one that TRICLUSTER is based on. gTRICLUSTER can find more biologically meaningful coherent gene clusters than TRICLUSTER can do. It also outperforms TRICLUSTER in robustness to noise. Experimental results on a real-world microarray dataset validate the effectiveness of the proposed new algorithm.

1 Introduction

Microarray technology can measure the expression level of a large number of genes interesting to the biologists, within a number of different experimental conditions. These conditions may be different time points, different environmental conditions, and different experimental samples. The data output by this technology is called gene expression data. A number of important bioinformatics and biomedical research problems are based on the analysis of gene expression data. It is an interesting and challenging task to efficiently and effectively mining meaningful clusters in gene expression data.

[⋆] The work was supported by the National Natural Science Foundation of China under grant no. 60373019, Open Research Foundation of Shanghai Key Lab of Intelligent Information Processing under grant no. IIPL-04-005, and partially supported by the Shuguang Scholar Program of Shanghai Education Development Foundation.

Up to date, a number of clustering algorithms for microarray data analysis have been developed [1], [2], [3], [4], [5], [6], in which most of the early developed clustering algorithms work in the full dimensional space [5]. However, in many applications, subspace clusters are more useful and meaningful than full space clusters. Biclustering algorithms [1], [6] or coclustering algorithms [7] were proposed to find groups of genes and conditions. Specifically, this kind of algorithms mine genes clusters defined with respect to a subset of the conditions, or conditions clusters defined with respect to a subset of the genes.

Nowadays, a newly developed microarray technology can monitor the expression levels of a set of genes under a set of samples during a series of time points. Data generated by this technology is called gene-sample-time microarray data (GST data for short) or three dimensional microarray data [2]. To exploit the power of GST data, some pioneering work on GST data clustering have been reported [2], [3].

TRICLUSTER is the first and the state-of-the-art 3D clustering algorithm over GST data. It mines the maximal 3D clusters (or triclusters) satisfying the following homogeneity criterion: any 2×2 submatrix of a tricluster must obey a constant *multiplicative* or *additive* relationship (or symmetry property in [3]). However, such a strict constraint considerably limits the capability of TRICLUSTER to find some useful patterns. Certainly, TRICLUSTER allows a certain deviation from this strict constraint, and the deviation degree is determined by the parameter ε. In practice, the allowed values of ε are very small (usually ε=0.003 in [3]), which still cannot prevent TRICLUSTER from missing some important cluster patterns hidden in the data. Let us take the data illustrated in Fig.1 as an example to demonstrate the weakness of TRICLUSTER caused by the imposed symmetry property.

Fig. 1 shows a set of synthetic data indicating the expression level profiles of one gene under two samples over six time points. Though the absolute values of these two profiles are quite different, to a biologist, the overall trends of these two profiles are highly consistent, which may mean that these two samples are biologically associated. However, to capture this kind of patterns, the parameter

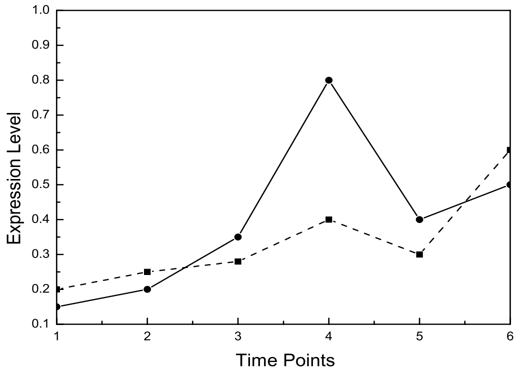


Fig. 1. An example to illustrate TRICLUSTER's weakness

ε of TRICLUSTER has to be 3.33, which is not only difficult for the users to select exactly such a value in advance, but also not allowed by TRICLUSTER (reminding in TRICLUSTER the parameter ε is usually less than 0.01).

Considering the weakness of TRICLUSTER, in this paper, we propose a new and effective deterministic 3D clustering algorithm, which is called gTRICLUSTER. Similar to TRICLUSTER, gTRICLUSETR also mines triclusters, but it is based on a more general tricluster model. In gTRICLUSTER, we give up the symmetry property imposed on TRICLUSTER, and use the Spearman rank correlation (SRC) as the basic similarity metric to evaluate the local similarity of two expression level profiles. This enables our approach to capture more cluster patterns that may be omitted by TRICLUSTER, and be more robust to noise than TRICLUSTER. Experiments on a real-world dataset validate the effectiveness of gTRICLUSTER.

Major contributions of this paper are as follows:

– we present a new 3D cluster model over GST microarray data. The new model avoids the symmetry property of TRICLUSTER, and is consistent with biological observation.
– we develop a new algorithm named gTRICLUSTER to mine effectively 3D cluster, based on the new cluster model mentioned above.
– we conduct experiments on a real-world data set to validate the effectiveness of the proposed algorithm.

The rest of the paper is organized as follows. Section 2 describes related work. Section 3 defines the problem. Section 4 presents the gTRICLUSTER algorithm. Section 5 gives experimental results. Section 6 concludes this paper.

2 Related Work

Previous work related to this research includes various algorithms of biclustering (or coclustering) [1], subspace clustering, project clustering on microarary data, and some pioneering work on GST data clustering [2], [3].

Cheng and Church first proposed the biclustering model [1]. The clusters are mined according to coherent gene and condition simultaneously. Biclustering model has been proved to be a NP-hard problem. Later, a number of algorithms are proposed to efficiently find biclusters based on different heuristic search schemes [4].

J. Liu, et al. [8] proposed OP-Cluster model to mine clusters that share similar tendency, and utilized prefix-tree structure to exhaustively enumerate all possibly patterns.

D. Jiang, et al. [2] first proposed a clustering technique applied to GST microarray data. They focused on mining the *maximal coherent gene clusters*. However, the cluster model is defined on global time series, i.e., the gene and sample dimensions.

Recently, Zhao and Zaki [3] presented the first and the state-of-the-art 3D clustering algorithm on GST microarray data, which is called TRICLUSTER.

They sliced the original GST microarray data along the time dimension to get a series gene$\times$sample tables (each time point corresponds to a gene$\times$sample table). By utilizing the imposed *symmetry property* in matrix computation, cost search on gene dimension can be converted to search on sample dimension. For each time point, i.e., a gene$\times$sample table, TRICLUSTER constructs a range multigraph to record all similar value ranges between any two sample columns. Then it searches for constrained maximal cliques in this multigraph to generate a set of biclusters for this time point. After all time points are processed, TRICLUSTER advances to mine the 3D clusters (triclusters) by merging the biclusters of different time points to generate the maximal cliques.

Our paper also deals with triclusters mining. Different from TRICLUSTER, 1) we use a more general 3D cluster model. Concretely, we release the symmetry constraint imposed on TRICLUSTER, which enables our algorithm to mine some biologically useful patterns that TRICLUSTER may omit; 2) we use Spearman rank correlation (SRC) to evaluate the local similarity of arbitrary two expression level profiles, which makes our algorithm be more robust to noise than TRICLUSTER; 3) we use set enumeration tree to find biclusters in sample $\times$ time matrices, and merge the biclusters to generate the maximal cliques by inverted list, while TRICLUSTER uses graph-based method to generate biclusters in the gene $\times$ sample tables and obtain the triclusters eventually by merging the biclusters.

3 Problem Statement

Given a set of n genes $G = \{g_0, g_1, \ldots, g_{n-1}\}$, a set of m biological samples $S = \{s_0, s_1, \ldots, s_{m-1}\}$ and a series of l time points $T = \{t_0, t_1, \ldots, t_{l-1}\}$, a *GST* microarray dataset is a real-valued three-dimensional $n \times m \times l$ matrix, $D = G \times S \times T = \{d_{ijk}\}$. Each cell d_{ijk} represents the expression level of gene g_i in sample s_j at time t_k.

In this paper, a *GST* data set is viewed as a two-dimensional $n \times m$ matrix, in which each cell m_{ij} represents the expression level of gene g_i under samples s_j during the whole time series, *i.e.*, a real value vector. While in [3], the *GST* three-dimensional matrix is partitioned into a series of two-dimensional tables, each of which represents the expression level of the whole gene set on the whole sample set at a concrete time point.

Some existing work on *GST* data clustering has investigated how to find those gene that are coherent on a subset of the samples during the whole time series [2]. A basic biological observation is: the genes that are biologically associated may behave in similar expression patterns only over a segment of the time series, and beyond this time range their expression patterns could be completely irrelevant. Here, we aim to find those genes coherent on a subset of the samples within a segment of the time series.

Usually, the users of microarray data are more concerned about the *qualitative behavior* (overall trend of the expression levels) rather than the absolute values. Here, a crucial issue is how to choose a proper metric to accurately evaluate the similarity between two arbitrary expression level profiles, and the metric should be robust to noise, data shifting and scaling. We will give our similarity metric in next section.

Definition 1. Given a gene g_i, for any pair of samples $s_{j1}, s_{j2} \in S$ (S is a subset of the whole samples set), denote the similarity between s_{j1} and s_{j2} as $Sim(m_{i,j1}, m_{i,j2})$. We say gene g_i is *coherent* across the subset of samples S during the time series segment T_i if the following two conditions hold.
1) $Sim(m_{i,j1}, m_{i,j2}) \geq \delta$, where δ is the *minimum similarity threshold* predefined by the users, and
2) $|T_i| \geq min_t$, here $|T_i|$ is the length of the time series segment T_i, min_t is the *minimum length of time series segment*, which is also pre-specified by the users.

Definition 2. For a subset of genes G with size not smaller than the *minimum gene size threshold* (denoted min_g) and a subset of samples S with size not smaller than the *minimum sample size threshold* (denoted min_s), if every gene $g_i \in G$ is coherent across the samples in S during the time series segment T_i, and $T = \bigcap_{i=1\sim|G|} T_i$, $|T| \geq min_t$, then $(G \times S \times T)$ is a *coherent 3D cluster*. In practice, we specify that a *coherent 3D cluster* must contain at least two genes and two sample under two time points.

Definition 3. A coherent three-dimensional cluster $G \times S$ is *maximal* if there exists not any other coherent three-dimensional cluster $G' \times S'$ such that $G \subseteq G'$ and $S \subseteq S'$.

Definition 4. Given a *GST* microarray matrix *M*, and pre-specified parameters δ, min_s, min_s and min_t, the 3D cluster (or tricluster) mining problem with M is to find all the *maximal coherent 3D clusters* in *M*.

4 gTRICLUSTER: A New 3D Clustering Algorithm over GST Microarray Data

In this Section, we will first give the similarity metric for gene expression level profiles, then describe the algorithm to finding biclusters in sample $\times$ time matrices, and finally describe the process to mine triclusters by merging biclusters on the gene dimension of GST data.

4.1 Similarity Metric

To better capture the local similarity of two profiles X and Y of one gene on two samples, we define the similarity of X and Y, denoted $Sim(X,Y)$, as the $Sim_k(X,Y)$ that satisfies the following two conditions: 1) $Sim_k(X,Y) \geq \delta$ (δ is defined in last section), k is the length of the time series segment in which the similarity is computed; 2) there is no any other time series segment of length $k' > k$, having $Sim_{k'}(X,Y) \geq \delta$. That is, the similarity of two profiles is determined by the longest time segment of these two profiles, which satisfies the basic similarity restriction (condition 1 above). Concretely, $Sim_k(X,Y)$ is given as follows:

$$\max_{1 \leq i \leq n-k+1} S(X[i,i+k-1], Y[i,i+k-1]). \qquad (1)$$

Here, S is a basic similarity measure. We choose the *Spearman rank correlation* (SRC) as the basic similarity measurement. The SRC between two profiles X and Y is evaluated as follows:

$$SRC(X,Y) = 1 - \frac{6}{n(n^2-1)} \sum_{i=1}^{n} (r_X(x_i) - r_Y(y_i))^2 \qquad (2)$$

where $r_X(x_i)$ is the rank of x_i in the profile $(x_1, \ldots, x_n)$: $r_X(x_i) = k \Leftrightarrow |\{j | x_j < x_i\}| = k-1$.

To demonstrate the advantage of SRC, let us go back to Fig.1, in which there are two synthetic expression level profiles with six time points. By using SRC, their similarity is 0.94, while using the Pearson correlation their similarity is 0.61. As we mentioned in the introduction section, by observation these two profiles show similar trend pattern, while TRICLUSTER cannot find such pattern.

4.2 Mining the Maximal Coherent Sample Subsets

To search a maximal coherent 3D cluster, the first step we take is to identify the *maximal coherent samples subset* for each gene. Given a gene g_k, a maximal coherent subset of samples S satisfies the following conditions: (1) $|S| \geq min_s$; (2) g_k is coherent on S according to Definition 1; and (3) g_k is not coherent on any superset of S. Certainly, a gene could have more than one maximal coherent sample sets. We search the maximal coherent sample subsets by using a method similar to that in [2]. First, for each gene g_k, we test each pair of samples (s_i, s_j) and construct a binary triangle similarity matrix (c_{ij}): $c_{ij} = 1$ if gene g_k is coherent on the sample pair (s_i, s_j); otherwise $c_{ij} = 0$.

After the construction of the matrix (c_{ij}), we convert the original problem to find the complete set of maximal cliques. Though the problem is NP-complete, in real microarray datasets the number of samples is usually less than one hundred. Thus, we can use a depth-first search in the sample space to enumerate all the possible maximal cliques. To further improve the performance, we associate the time series range information in the similarity matrix. Consequently, the global search space is partitioned into a number of sample subspaces, each of which includes only those samples whose overlayed time series' length is larger than min_t. The following algorithm describes the above process.

```
**Algorithm 1: Computing maximal coherent sample sets.**
**Input:** the $GST$ data set, $\delta$, $min_s$, $min_t$;
**Output:** the maximal coherent samples sets in any possible time series segment
for each gene;
**foreach** gene $g_k$
    compute similarity matrix $c_{ij}$ of $g_k$;
    Initiate $EC$ (Existed Maximal Cliques);
    //partition the search space into different subspaces
    for i=1 to (time series length+$min_t$-1)
        extract the samples set $S = \{s_1, \ldots, s_n\}$ that the pattern of
        each sample pair in S covers the time series segment $(i, i + min_t - 1)$;
        for j=1 to $(n - min_s + 1)$
            CliqueSearch($\{s_i\}, \{s_{i+1}, \ldots, s_n]\}$);
        end for
    end for
end foreach

Function: CliqueSearch(current, extend)
    Let $s_i$ be the last sample in current, if $c_{ij} = 0$, delete sample $s_j$ from extend;
    if ($|current \cup extend| < min_s$) // the number of samples is too small
        then return;
    if ($current \cup extend \subset C$) // $C \in EC$, $EC$ is existed maximal cliques set
        then return; // further search won't lead to new maximal cliques
    if (extend is empty) then
        take current as a new maximal clique and move it to EC;
    else
        while (extend is not empty)
            move the first sample of extend to current;
            CliqueSearch(current,extend);
        end while
    return;
```

4.3 Finding 3D Clusters

After obtaining the maximal coherent sample subset for each gene, the process of mining triclusters is straightforward. First, we apply a depth-search strategy similar to that in Algorithm 1 to enumerated subsets of samples. Then, for each subset(combination) of samples, we need to find the *maximal coherent gene set* G_s such that the genes in G_s are coherent on S. An efficient solution is to use *inverted list.* In advance, we generate the *inverted list* L_s for each sample, which consists of all the maximal coherent sample sets containing s. For example, we have $g_1 = \{s_1, s_2, s_4\}, g_2 = \{s_1, s_3, s_4\}$ as the maximal coherent sample sets for genes, then the inverted lists for samples are $s_1 = \{g_1, g_2\}, s_2 = \{g_1\}, s_3 = \{g_2\}, s_4 = \{g_1, g_2\}$. When we want to computer the maximal coherent gene sets G_s for a subset of samples S, we only need to get the intersection of the inverted lists. Last, we test whether $(G_s \times S)$ is a maximal coherent gene cluster and combine with time segments information, thus we obtain a tricluster.

5 Experimental Evaluation

We implement gTRICLUSTER in JAVA and evaluate it on a real GST dataset. To compare the performance of TRICLUSTER and gTRICLUSTER, we also implement TRICLUSTER from scratch. The experiments are done on a PC (2.0GHz Pentium-IV and 512M memory) running Windows XP.

5.1 The Dataset

We use the real microarray dataset reported in [9], which studies the relationship between the mRNA expression level regulation and the cell cycle. Among the whole dataset, we focus on the time series for the Pheromone experiments. There are totally 6178 genes whose expression values are measured from 0 to 119 minutes with 7-minute intervals. So there are totally 18 time points. We conduct the routine data preprocessing procedure and filtered 255 genes with errors or overfull missing values. Then we use 9 attributes of the original dataset as the samples. Finally we get a GST microarray data set with size: $G \times S \times T = 5923 \times 9 \times 18$. We apply gTRICLUSTER on the generated data with parameters as follows: $min_g = 20$, $min_s = 3$, $min_t = 6$, $\delta = 0.9$. In total, we obtain 41 clusters that cover 5278 genes in the dataset. The overlap ratio is 7.1%. On average each cluster contains 138 genes and 5 samples, and the average time series span is 12 points. For TRICLUSTER, we set parameters as follows: $mx = 50$(genes), $my = 4$(samples), $mz = 5$(time pints), and $\varepsilon = 0.01$.

5.2 Effectiveness

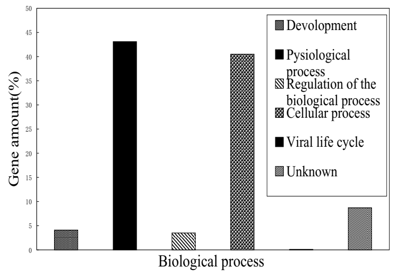
The gene ontology (GO) project (www.geneontology.org) provides a controlled vocabulary to describe gene and gene product attributes in any organism. We use the Onto-Express (http://vortex.cs.wayne.edu/Projects.html) to verify the biological significance of gTRICLUSTER's clustering results. For each of the following three gene function categories: biological processes, cellular components and gene functions, we construct a hierarchy of GO terms for each gene in the clusters found by gTRICLUSTER. Table 1 lists six significant clusters. From column 1 to column 5, the data means the cluster number (Cluster No.), the number of genes in the cluster (open reading frame, ORF), biological process, molecular function, and cellular component respectively. For example, the genes in cluster No. 0 are mainly related to conjugation. The tuple (n=34, p=0.000951) means that out of the 73 genes in cluster 0, 34 belong to the conjugation process, and the statistical significance is measured by the p-value of 0.000951.

From Table 1, we can see that gTRICLUSTER can find biologically meaningful clusters. For example, in cluster No. 4, 14 genes belong to mRNA catabolism, and in cluster No. 22, 27 genes belong to intracellular signaling cascade. However, most clusters in Table 1 cannot be identified by TRICLUSTER. In cluster No. 4, 42 genes belong to the modification-dependent protein catabolism, while TRICLUSTER can group only 3 such genes to one cluster.

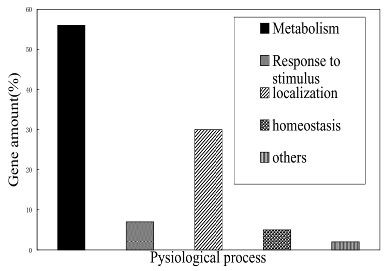
Fig. 2 depicts the distribution of biological process in cluster 10. From Fig. 2a, we can see that the majority of the genes in this cluster are involved in cellular processes and physiological process, while genes involved in other biological processes (e.g., development, regulation, viral life cycle) are very few. Furthermore, examining Fig. 2b, it is noticeable that among the genes involved in physiological process, those involved in metabolism and localization hold a large portion. This result may help the users to predict the biological characteristics of the genes

Table 1. Some biologically meaningful clusters mined by gTRICLUSTER

Cluster NO.	Number of ORFs	Biological Process	Molecular Function	Cellular Component
0	73	conjugation (n=34, p=0.000951), cell budding (n=21,0.000477)	ubiqutin conjugating enzyme activity (n=9, p=0.00439)	ubiqutin ligase complex (n=13, p=0.000765)
4	103	modification-dependent protein catabolism (n=42, p=0.00000263), mRNA catabolism (n=14, p=0.0000589)	ligase activity (n=27, p=0.00112), iosmeratse activity (n=15, p=0.00275)	mitochondrial nucleoid (n=7, p=0.00837)
17	66	DNA damage response, signal transduction (n=8, p=0.00272), response to oxidatate stress (n=9, p=0.00992)	general RNA polymerase II transcription factor activity (n=20, p=0.00498), transcription regulator activity (n=12, p=0.0000783)	nucleotide excision repair complex (n=6, p=0.00904)
22	71	intracellular signaling cascade (n=27, p=0.000754), membrane fusion (n=15, p=0.000534)	signal transducer activity (n=20, p=0.000343)	incipient budsit (n=7, p=0.000365), signalosome complex (n=12, p=0.000207)
29	202	protein targeting (n=52, p=0.00124), RNA-nucleus export (n=20, p=0.000935)	structural constituent of ribosome (n=62, p=0.000986), structural constituent cytoskeleton (n=12, p=0.00431)	cytoplasmic vesicle (n=11, p=0.00509)
36	157	negative regulation of cellular physiological process (n=50, p=0.000541)	translation regulator activity (n=12, p=0.00565)	bud tip (n=10, p=0.00823), mating projection tip (n=8, p=0.00476)



(a) The distribution of biological processes in Cluster 10



(b) The distribution of physiological pro- cesses in Cluster 10

Fig. 2. The biological process distribution of cluster 10

in cluster No 10 even when the users has little knowledge of those genes. It also shows that gTRICLUSTER can find potentially biologically significant clusters in GST microarray data.

5.3 Robustness to Noise

With the inherent inaccuracy of microarray experiments, the expression data may contain noise such as missing value or numerical value fluctuation. So a practical clustering algorithm should be robust to noise. We use the Adjusted Rand Index (ARI) [5] as the performance metric for robustness to noise. ARI is used to evaluate the similarity between the clustering result based on domain knowledge (simply K) and that obtained by TRICLUSTER or gTRICLUSTER (simply R). ARI is computed as follows:

$$ARI(K, R) = \frac{2(ad - bc)}{(a+b)(b+d) + (a+c)(c+d)} \tag{3}$$

where a, b, c and d respectively represents the number of object pairs that belong to the same cluster in both K and R, belong to the same cluster in K but not R, belong to the same cluster in R but not K, and belong to different clusters in both K and R. ARI's values lies between 0 and 1, and larger value means higher similarity between the clustering results. If the experimental result is perfectly consistent to the domain knowledge, the index value will be 1. If a clustering is no more than a random choice, the index will be zero.

We apply gTRICLUSTER on the original dataset and take the result as the domain knowledge. We add 1%, 2%, 4%, 6%, 8% and 10% noise (missing values and fluctuation of values) into the original dataset respectively, then compute the ARI values between the results on noisy datasets and the domain knowledge (corresponding to the original dataset). Such a process is repeated ten times. Similar experiments are also conducted with TRICLUSTER. The results are illustrated in Fig. 3, where the *best*, *average* and *worst* curves correspond to the best, average and worst results respectively. It can be seen that the ARI values of gTRICLUSTER are larger that of TRICLUSTER for all experiment setting, which means that gTRICLUSTER is more robust to noise than TRICLUSTER.

We then use the *precision* and *recall* metric to evaluate the stability of gTRICLUSTER's clustering results in time and sample dimensions. Similarly, we take

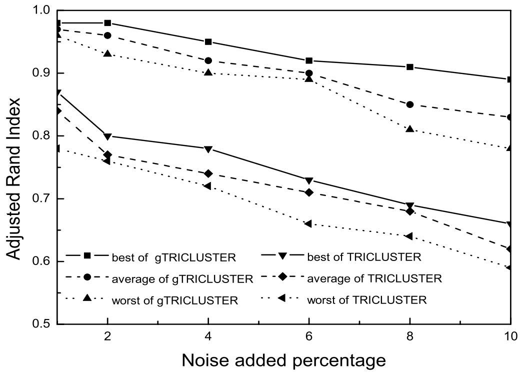
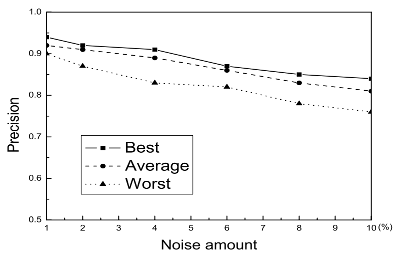
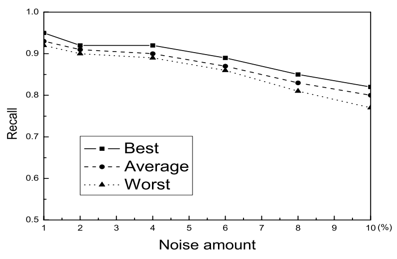


Fig. 3. Comparison of robustness to noise: gTRICLUSTER vs. TRICLUSTER

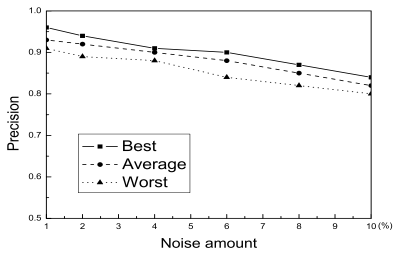


(a) The precision of sample dimension

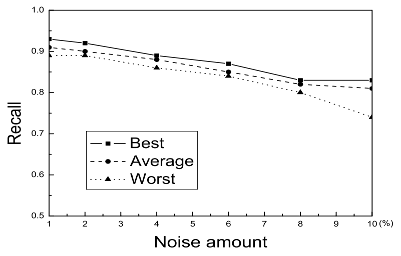


(b) The recall of sample dimension

Fig. 4. Precision and recall of sample dimension



(a) The precision of time dimension



(b) The recall of time dimension

Fig. 5. Precision and recall of time dimension

the clustering results without noise as domain knowledge. For each clusters pair, denote a, b, c as the numbers of samples (time points) selected both by K and R, the number of samples (time points) selected by R, the number of samples (time points) selected by K. The precision of sample/time dimension is a/b and the recall of sample/time dimension is a/c. Experimental results are shown in Fig. 4 and Fig. 5 respectively. We can see that even in noisy condition gTRICLUSTER keeps high precision and recall in both sample and time dimensions.

6 Conclusion

In this paper, we present a novel clustering algorithm called gTRICLUSTER, to mine 3D clusters in GST microarray data. Our experiments on a real-world microarray data set show that gTRICLUSTER can mine biologically meaningful clusters effectively and has good robustness to noise.

Recent studies have noticed the time shifting phenomenon in gene expression data: two similar patterns may appear in different time ranges, i.e., there is a time latency between two similar patterns. Some naïve techniques have been developed to handle such a problem, but the computation cost is very high. For future work, we plan to develop new algorithms to mine this kind of patterns in GST data based on dynamic programming.

References

1. Y.Cheng and G.M.Church, Biclustering gene expression patterns. In Proc. of the 3rd Annual Int'l Conference on Computational Biology (RECOMB'99), 1999.
2. D.Jiang, J.Pei, M.Ramanathany, C.Tang and A.Zhang, Mining coherent gene clusters from gene-sample-time microarray data. In Proc. of the 10th ACM SIGKDD Conference (KDD'04), 2004.
3. Lizhuang Zhao and Mohammed J.Zaki, TRICLUSTER: An effective algorithm for mining coherent clusters in 3D microarray Data. In Proc. of SIGMOD'05.
4. Yang. J., et al. -cluster: Capturing Subspace Correlation in a large Data Set. In Proc. of ICDE'02.
5. K. Yeung and W. Ruzzo, An Empirical Study on Principal Component Analysis for Clustering Gene Expression Data, Bioinformatics, 17(9):763-774, 2001.
6. R. Balasubramaniyan, E. Hullermeier, N. Weskamp and J. Kamper, Clustering of gene expression data using a local shape-based similary measure, Bioinformatics, vol.21, no.7, 2005.
7. I. S. Dhillon, S. Mallela, and D. S. Modha, Information-Theoretical Coclustering. In Proc. of the Ninth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD'03), pp.89-98, 2003
8. J. Liu, J. Yang, and W. Wang. Biclustering of gene expression data by tendency. In Proceedings of the IEEE Computational Systems Bioinformatics Conference (CSB), pp. 182-193, 2004.
9. P.T.Spellmean, G.Sherlock, M.Q.Zhang, V.R.Iyer, et al. Comprehensvie identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. Molecular Biology of the Cell, 9(12):3273-3297, Dec. 1998.

Automatic Orthologous-Protein-Clustering from Multiple Complete-Genomes by the Best Reciprocal BLAST Hits*

Sunshin Kim, Kwang Su Jung, and Keun Ho Ryu

Database/Bioinformatics Laboratory, Department of Computer Science,
Chungbuk National University, Cheongju, South Korea
{sskim04, ksjung, khryu}@dblab.chungbuk.ac.kr

Abstract. Though the number of completely sequenced genomes quickly grows in recent years, the methods to predict protein functions by homology from the genomes have not been used sufficiently. It has been a successful technique to construct an OPCs(Orthologous Protein Clusters) with the best reciprocal BLAST hits from multiple complete-genomes. But it takes time-consuming-processes to make the OPCs with manual work. We, here, propose an automatic method that clusters OPs(Orthologous Proteins) from multiple complete-genomes, which is, to be extended, based on INPARANOID which is an automatic program to detect OPs between two complete-genomes. We also prove all possible clustering mathematically.

1 Introduction

Although an unprecedented amount of amino acid sequences has been accumulated in databases, the most protein functions have not been known. Though the number of completely sequenced genomes have especially been increased above 150 species, the methods to extract functional information from the species have not been used sufficiently. Predicting protein functions by homology is one of the fundamental researches in bioinformatics. When recognizing the unknown functions of protein sequences, it is very useful to infer the functions of the new sequences from the protein sequences with known functions.

One of techniques to predict the unknown protein functions from the known protein functions is to construct an OPCs. It is possible to infer the unknown functions of protein sequences from the known protein functions by clustering the proteins with same functions into same groups as distinguished from other proteins with different functions. When inferring the functions of the new protein sequences recognized by biological experiments, we can also predict the functions by comparing the sequences with the OPCs.

It could be supposed that organisms are evolving through gene duplication and speciation. The functions of the duplicated proteins are thought to be changed each other as evolving with time after the species split. Those of the others are believed not to be changed with time. As there happen mutations with time, the sequence similarity

* This work was supported by the Regional Research Centers Program of Ministry of Education & Human Resources Development in South Korea.

of proteins may become less. The closer the time the species split is, the higher the similarity will be. The farther the time speciation happens is, the less sequence similarity may become. At that time it could be supposed that the evolving rates of the whole proteins in a genome are almost the same. The sequence similarity of specific genes between two genomes, which have the same functions each other, is therefore expected to be higher than the others. By clustering the protein pairs with one another's reciprocal best hits by comparing protein pairs among the completely sequenced genome pairs, since the similarity of the protein sequences with the same functions is the highest of all protein pairs in each genome pairs, we can make an OPCs. Such genes evolved by only speciation are called orthologs[1] and have been classified as a group with the same function. The speciating genes after gene duplication are called paralogs[1] and have been studied as taking different functions each other[2].

When constructing an OPCs, we face an obstacle. It takes time-consuming-processes to deal with large datasets of protein sequences by hand. We propose an automatic method which clusters OPs automatically from multiple complete-genomes without almost manual work.

Section 2 discusses related work. Our approach of constructing an OPCs is represented in Section 3. The proof of all possible clustering is given in Section 4. We conclude with a discussion of our work.

2 Related Work

The COGs(Clusters of Orthologous Groups of proteins)[3] were clustered from 21 complete genomes of bacteria, archaea and eukaryotes, between which proteins with the best reciprocal BLAST[4] hits are supposed to be orthologous each other. The method to be used is to detect triangles formed from protein lines(pairs) with mutually best hits among genomes and merge triangles with a common side of a protein line(pair) through biological analysis without an arbitrary threshold.

The COG database[3, 5] in NCBI(National Center for Biotechnology Information) is constructed from 66 complete genomes using the gapped BLAST program. The BLAST is very efficient and fast by using the heuristic method[6, 7, 8, 9, 10], but it has been known that it is very difficult to detect the homology between evolutionarily distant genomes[11, 12, 13, 14].

KO(KEGG Orthology)[15] is a database with the ortholog group tables, which contain orthologous genes extracted from metabolic pathways or regulatory pathways. KO is almost exactly classified because it is manually edited from the pathways which show the functional relations of proteins clearly.

INPARANOID[16] is a fully automatic program that detects proteins with the best reciprocal BLAST hits between only two genomes and decides a main protein pair of a and b , fixed as a center point, from which additional in-paralogs(orthologs) are clustered. To avoid the relations of the evolutionarily distant genomes and short-domain-level matches, a score cut-off of 50 bits and an overlap longer than 50% of the longer sequence were only used. They compared their results with those of the phylogenetic method. INPARANOID detected the additional orthologs of 7% than the method, and discovered the false positives of 9%, the false negatives of 3%, and the true positives of 84%.

Montague *et al.*[17] disclosed differences in the degree of conservation between functional classes of genes by introducing COG Stringency Number in addition to the COG technique. Clues for genes with related functions may appear by their method.

Stuart *et al.*[18] made an OPCs which is defined as metagenes clustered from multiple eucaryote organisms for explaining the fact that the functions of genes conserved through evolution are globally related with each other. The metagenes are clustered by the similar way as constructing COGs except an E-value less than 10^{-5} to make a purer OPCs.

In this paper, our aim on construction of an OPCs is to prove all possible clus-tering mathematically. Our approach begins with the results produced by INPARANOID[16]. We, therefore, settle on the quality of orthologs detected by the INPARANOID program.

3 Our Approach

3.1 The Algorithm's Basis

Sequence space can be represented as a graph. The nodes of the graph represent the sequences and the links among nodes represent the degree of similarity[2]. As shown in Fig. 1, genomes consist of proteins(genes), which can be represented as $G_1=\{g_{11}, g_{12}, g_{13}, ..., g_{1a}\}$ and $G_2=\{g_{21}, g_{22}, g_{23}, ..., g_{2b}\}$ when G_1 and G_2 are genomes, and g_{1i} and g_{2j} are proteins. If a protein pair with the reciprocal best hit between two genomes is identified, the protein pair has better similarity than other protein pairs, and each protein of the pair is supposed to be orthologous. The orthologous protein lines are respectively stored in a table.

Definition 1. *If a protein pair has the best reciprocal BLAST hits between two complete genomes, the protein pair is a protein line.*

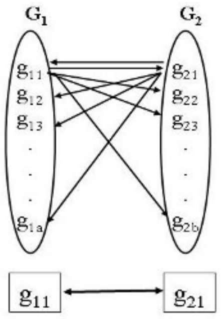


Fig. 1. A protein pair with the reciprocal best hit

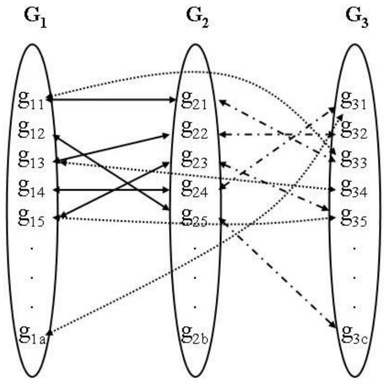


Fig. 2. Protein pairs relationships with the reciprocal best hits among three genomes

Definition 2. *If a cluster has ones more than or equal to two protein lines with common genes, the cluster is an orthologous protein cluster.*

Let's consider a simple example. There are three genomes, which have protein pairs relationships as shown in Fig. 2. The protein pairs with the reciprocal BLAST best hits between two genomes can be stored as the tables of Fig. 3. From the protein pairs with common proteins of the three tables, which are flagged not to search repeatedly, each orthologous-protein-cluster can be constructed separately as in Fig. 4. Each cluster in the Γ table has its inherent protein function different from the others.

When considering *n* genomes, we can, in general, have *n(n-1)/2* tables with the lines of protein pairs as Fig. 5.

T_{12}

G_1	G_2	Flag
g_{11}	g_{21}	yes
g_{12}	g_{25}	yes
g_{13}	g_{22}	yes
g_{14}	g_{24}	none
g_{15}	g_{23}	none

T_{13}

G_1	G_3	Flag
g_{11}	g_{33}	yes
g_{13}	g_{34}	yes
g_{15}	g_{35}	yes
g_{1a}	g_{31}	yes

T_{23}

G_2	G_3	Flag
g_{21}	g_{33}	yes
g_{22}	g_{32}	yes
g_{23}	g_{35}	yes
g_{24}	g_{31}	yes
g_{25}	g_{3c}	yes

Fig. 3. Proteins pairs tables with the reciprocal best hits among three genomes

Γ_{123}

OPC001	g_{11}, g_{21}, g_{33}
OPC002	g_{13}, g_{22}, g_{32}, g_{34}
OPC003	g_{12}, g_{25}, g_{3c}
OPC004	g_{15}, g_{23}, g_{35}
OPC005	g_{1a}, g_{24}, g_{31}

Fig. 4. A total table with the reciprocal best hits among three genomes

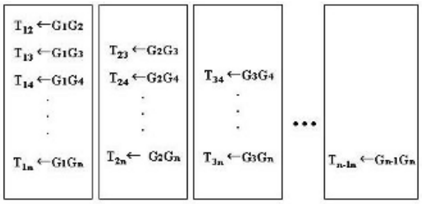

$T_{12} \leftarrow G_1G_2$
$T_{13} \leftarrow G_1G_3$
$T_{14} \leftarrow G_1G_4$
$T_{1n} \leftarrow G_1G_n$
$T_{23} \leftarrow G_2G_3$
$T_{24} \leftarrow G_2G_4$
$T_{2n} \leftarrow G_2G_n$
$T_{34} \leftarrow G_3G_4$
$T_{3n} \leftarrow G_3G_n$
$T_{n\text{-}1n} \leftarrow G_{n\text{-}1}G_n$

Fig. 5. All possible genome pairs and tables from *n* genomes

3.2 Clustering Algorithm

The suggested algorithm has two procedures. The first procedure is to cluster all possible OPs from *n* genomes. The second procedure is to cluster additional OPs into the already grouped clusters.

First stage: Detect and save all possible protein pairs with the best reciprocal BLAST hits among *n* genomes, and merge all possible lines(pairs) with common proteins.

Second stage: To add new proteins of a new genome to the clusters, detect and save all possible protein pairs with the best reciprocal BLAST hits between the new genome and *n* genomes, and merge all possible lines(pairs) with common proteins.

When clustering *n* Genomes
Algorithm ClusteringGenomes

```
PairwiseSpeciesClustering
InitialOrthologClustering
for m=4 ... n
  OrthologClustering
end for
```

Fig. 6. Algorithm to cluster *n* genomes

When adding the *(n+1)th* genome
Algorithm AddingNewGenome

```
PairwiseSpeciesAddClustering
m=n+1
OrthologClustering
```

Fig. 7. Algorithm to add an additional genome to the previously grouped clusters

Algorithm PairwiseSpeciesClustering
Require: *n* selected species

```
for i = 1 ... i = n-1 do
    for j=i+1 ... j=n do
      search the reciprocal best hits of two proteins in both $Genome_i$ and $Genome_j$
      save the proteins in $T_{ij}$
    end for
 end for
```

Fig. 8. Algorithm to get all possible protein pairs with BLAST from *n* genomes

Algorithm PairwiseSpeciesAddClustering
Require: the *(n+1) th* selected species

```
j = n+1
for i = 1 ... i = n do
      search the reciprocal best hits of two proteins in both $Genome_i$ and $Genome_j$
      save the proteins in $T_{ij}$
end for
```

Fig. 9. Algorithm to get all possible protein pairs with BLAST from a new genome and *n* genomes

Fig. 6 shows the subroutines of the first stage to cluster the protein pairs among *n* genomes and Fig. 7 shows the second stage. Fig. 8 and Fig. 9 show the steps to get all possible protein lines with the best reciprocal BLAST hits among *n* selected genomes, and between a new genome and *n* genomes. The results of both Fig. 8 and Fig. 9 can be drawn from INPARANOID program. Fig. 10 shows the tables with all possible protein lines(pairs) among three genomes by BLAST. Fig. 15, using the tables from Figure 10, shows all possible steps to search all possible protein lines(pairs) with common proteins, and save the proteins into the total table of Γ. Next, Fig. 16 presents all possible steps to search all protein lines with common proteins from the tables gotten due to adding genomes 4th – *n*th , and save the detected proteins into Γ.

T_{12}	T_{13}
	T_{23}

Fig. 10. The tables to be taken from three genomes

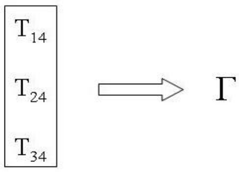


Fig. 11. The relationship between the three tables due to the 4th genome and the total table gotten from the three genomes

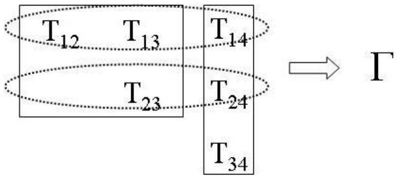


Fig. 12. The horizontal relationship between the tables from three genomes and the tables due to the 4th genome

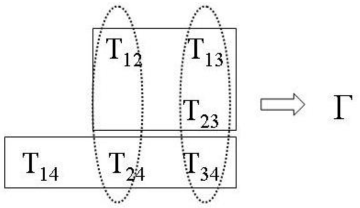


Fig. 13. The vertical relationship between the tables from three genomes and the tables due to the 4th genome

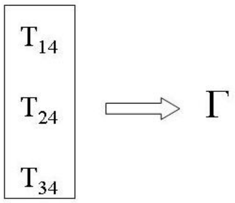
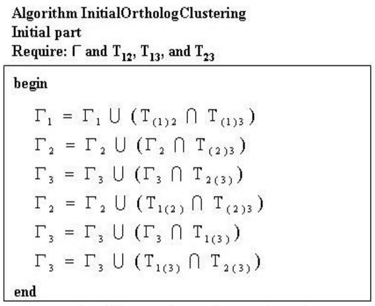


Fig. 14. The relationship among only the tables due to the 4th genome

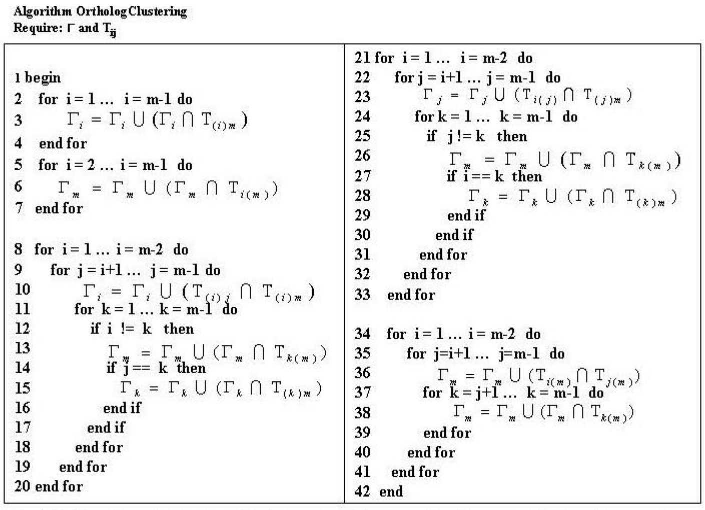


```
Algorithm InitialOrthologClustering
Initial part
Require: $\Gamma$ and $T_{12}$, $T_{13}$, and $T_{23}$
begin
    $\Gamma_1 = \Gamma_1 \cup (T_{(1)2} \cap T_{(1)3})$
    $\Gamma_2 = \Gamma_2 \cup (\Gamma_2 \cap T_{(2)3})$
    $\Gamma_3 = \Gamma_3 \cup (\Gamma_3 \cap T_{2(3)})$
    $\Gamma_2 = \Gamma_2 \cup (T_{1(2)} \cap T_{(2)3})$
    $\Gamma_3 = \Gamma_3 \cup (\Gamma_3 \cap T_{1(3)})$
    $\Gamma_3 = \Gamma_3 \cup (T_{1(3)} \cap T_{2(3)})$
end
```

Fig. 15. Algorithm to cluster three selected genomes

We, here, explain all the steps in detail to cluster the 4th genome to *n*th into the total table with the OPs of the three genome. All the protein lines(pairs) with common proteins among three tables are represented as in Fig. 15. The compared common genomes in the table of T are represented as the subscript numbers in parentheses. In the total table of Γ, the common genomes are represented as the subscript numbers.

From the new three tables due to the additional 4th genome, all possible steps to search and save protein lines with common proteins, when compared with the total table of Γ, are represented as Fig. 11 and the following expressions in general, which have the same meaning as the statements of the lines 1-7 in Fig. 16. All the steps can be generalized as case (a) and case (b) of formula (1).



```
Algorithm OrthologClustering
Require: $\Gamma$ and $T_{ij}$

1 begin
2   for i = 1 ... i = m-1 do
3       $\Gamma_i = \Gamma_i \cup (\Gamma_i \cap T_{(i)m})$
4   end for
5   for i = 2 ... i = m-1 do
6       $\Gamma_m = \Gamma_m \cup (\Gamma_m \cap T_{i(m)})$
7   end for

8   for i = 1 ... i = m-2 do
9     for j = i+1 ... j = m-1 do
10        $\Gamma_i = \Gamma_i \cup (T_{(i)j} \cap T_{(i)m})$
11      for k = 1 ... k = m-1 do
12        if i != k then
13          $\Gamma_m = \Gamma_m \cup (\Gamma_m \cap T_{k(m)})$
14          if j == k then
15            $\Gamma_k = \Gamma_k \cup (\Gamma_k \cap T_{(k)m})$
16          end if
17        end if
18      end for
19    end for
20  end for
21  for i = 1 ... i = m-2 do
22    for j = i+1 ... j = m-1 do
23        $\Gamma_j = \Gamma_j \cup (T_{i(j)} \cap T_{(j)m})$
24      for k = 1 ... k = m-1 do
25        if j != k then
26          $\Gamma_m = \Gamma_m \cup (\Gamma_m \cap T_{k(m)})$
27          if i == k then
28            $\Gamma_k = \Gamma_k \cup (\Gamma_k \cap T_{(k)m})$
29          end if
30        end if
31      end for
32    end for
33  end for

34  for i = 1 ... i = m-2 do
35    for j=i+1 ... j=m-1 do
36        $\Gamma_m = \Gamma_m \cup (T_{i(m)} \cap T_{j(m)})$
37      for k = j+1 ... k = m-1 do
38          $\Gamma_m = \Gamma_m \cup (\Gamma_m \cap T_{k(m)})$
39      end for
40    end for
41  end for
42 end
```

Fig. 16. Algorithm to cluster orthologous protein from the 4th genome to the *nth* genome

$$
\begin{aligned}
&Case\ \ (a) : \Gamma_i^{n+1} = \Gamma_i^n \cup (\Gamma_i^n \cap \mathrm{T}_{(i)m}) \\
&where \qquad (i) : 1 \le i \le m-1 \\
&Case\ \ (b) : \Gamma_m^{n+1} = \Gamma_m^n \cup (\Gamma_m^n \cap \mathrm{T}_{i(m)}) \\
&where \qquad (m) : 2 \le i \le m-1 .
\end{aligned}
\tag{1}
$$

After the above phase, all possible steps to search and store the protein lines with common proteins from the tables are considered as Fig. 12 in case of the 4th genome and the following expressions in general. In this case the lines 8-20 of Fig. 16 have the same steps as those. All the steps are generalized as case (a) and case (b) of formula (2).

$$
\begin{aligned}
&Case\ (a): \quad \Gamma_i^{n+1} = \Gamma_i^n \cup (\mathrm{T}_{(i)j} \cap \mathrm{T}_{(i)m}) \\
&where \quad (i): 1 \le i \le m-2, \quad i+1 \le j \le m-1. \\
&Case\ (b): \begin{cases} \Gamma_m^{n+1} = \Gamma_m^n \cup (\Gamma_m^n \cap \mathrm{T}_{k(m)}) \quad when \quad (m): \quad if \quad i \ne k \\ \Gamma_k^{n+1} = \Gamma_k^n \cup (\Gamma_k^n \cap \mathrm{T}_{(k)m}) \quad when \quad (k): \quad if \quad j = k \end{cases} \\
&where \quad 1 \le i \le m-2, \quad i+1 \le j \le m-1, \quad and \quad 1 \le k \le m-1.
\end{aligned}
\tag{2}
$$

After finishing the above phase, Fig. 13 in case of the 4th genome and in general formula (3) follows, which are also expressed by the lines 21-33 of Fig. 16.

$$\begin{aligned}
&Case\ (a): \Gamma_j^{n+1} = \Gamma_j^n \cup (\mathrm{T}_{i(j)} \cap \mathrm{T}_{(j)m}) \\
&where \quad (j): 1 \le i \le m-2, \quad i+1 \le j \le m-1. \\
&Case\ (b): \begin{cases} \Gamma_m^{n+1} = \Gamma_m^n \cup (\Gamma_m^n \cap \mathrm{T}_{k(m)}) & when \quad (m): \quad if \quad j \ne k \\ \Gamma_k^{n+1} = \Gamma_k^n \cup (\Gamma_k^n \cap \mathrm{T}_{(k)m}) & when \quad (k): \quad if \quad i = k \end{cases} \\
&where \quad 1 \le i \le m-2, \quad i+1 \le j \le m-1, \quad and \quad 1 \le k \le m-1.
\end{aligned} \tag{3}$$

Finally, all possible steps among just the tables due to the fourth genome are represented as Fig. 14 in case of the 4th genome, and the following expressions in general, which have the same steps as the lines 34-42 of Fig. 16. All the steps are expressed as case(a) and case(b) of formula (4).

$$\begin{aligned}
&Case\,(a): \Gamma_m^{n+1} = \Gamma_m^n \cup (\mathrm{T}_{i(m)} \cap \mathrm{T}_{j(m)}) \\
&where\,(m): 1 \le i \le m-2, \quad i+1 \le j \le m-1. \\
&Case\,(b): \Gamma_m^{n+1} = \Gamma_m^n \cup (\Gamma_m^n \cap \mathrm{T}_{k(m)}) \\
&where\,(m): 1 \le i \le m-2, \quad i+1 \le j \le m-1, \quad and \quad j+1 \le k \le m-1.
\end{aligned} \tag{4}$$

4 Proof of All Possible Clustering

Lemma 4.1. *If there are n(n-1)/2 tables with OPs from n complete-genomes, then the total or partial unions of the tables consist of orthologous proteins.*
Proof. It is trivial to prove this lemma. Since *n(n-1)/2* tables drawn from *n* complete-genomes include only OPs, the total or partial sums of the tables must consist of only OPs.

Lemma 4.2. *If there are lines(orthologous protein-pairs) with common proteins, then an orthologous protein cluster consists of the total or partial unions of the lines.*

Proof. It is also trivial to prove this lemma. If the lines have common proteins, then the total or partial sums of the pairs make an orthologous protein cluster.

Theorem 4.1. *Formula (1) clusters all possible OPs among the total table Γ and the new tables due to additional genomes.*

Proof. Let's consider Fig. 15, Fig. 11, and the following expressions.

$$\begin{aligned}
\Gamma_1^{n+1} &= \Gamma_1^n \cup (\Gamma_1^n \cap \mathrm{T}_{(1)4}) \\
\Gamma_2^{n+1} &= \Gamma_2^n \cup (\Gamma_2^n \cap \mathrm{T}_{(2)4}) \\
\Gamma_3^{n+1} &= \Gamma_3^n \cup (\Gamma_3^n \cap \mathrm{T}_{(3)4}) \\
\Gamma_4^{n+1} &= \Gamma_4^n \cup (\Gamma_4^n \cap \mathrm{T}_{2(4)}) \\
\Gamma_4^{n+1} &= \Gamma_4^n \cup (\Gamma_4^n \cap \mathrm{T}_{3(4)})
\end{aligned}$$

Algorithm Fig. 15 represents all possible combinations of tables drawn from three complete-genomes to get OPCs. By Lemma 4.1 and 4.2, the above expressions explain all possible combinations of the total table Γ and the new tables due to the 4th genome. From this specific fact, we can easily generalize the above equations into formula (1).

Theorem 4.2. *Formula (2) clusters all possible OPs with horizontal relationships among the prior tables and the new tables due to additional genomes, and all possible OPs among the total table Γ and the rest of the new tables due to additional genomes.*

Proof. Let's consider Fig. 12, and the following expressions.

$$\Gamma_1^{n+1} = \Gamma_1^n \cup (T_{(1)2} \cap T_{(1)4}) \quad \Gamma_1^{n+1} = \Gamma_1^n \cup (T_{(1)3} \cap T_{(1)4}) \quad \Gamma_2^{n+1} = \Gamma_2^n \cup (T_{(2)3} \cap T_{(2)4})$$
$$\Gamma_4^{n+1} = \Gamma_4^n \cup (\Gamma_4^n \cap T_{2(4)}) \text{ ,} \quad \Gamma_4^{n+1} = \Gamma_4^n \cup (\Gamma_4^n \cap T_{2(4)}) \text{ ,} \quad \Gamma_4^{n+1} = \Gamma_4^n \cup (\Gamma_4^n \cap T_{1(4)})$$
$$\Gamma_2^{n+1} = \Gamma_2^n \cup (\Gamma_2^n \cap T_{(2)4)}) \quad \Gamma_3^{n+1} = \Gamma_3^n \cup (\Gamma_3^n \cap T_{(3)4)}) \quad \Gamma_3^{n+1} = \Gamma_3^n \cup (\Gamma_3^n \cap T_{(3)4)})$$
$$\Gamma_4^{n+1} = \Gamma_4^n \cup (\Gamma_4^n \cap T_{3(4)}) \quad \Gamma_4^{n+1} = \Gamma_4^n \cup (\Gamma_4^n \cap T_{3(4)}) \quad \Gamma_4^{n+1} = \Gamma_4^n \cup (\Gamma_4^n \cap T_{3(4)})$$

By Lemma 4.1, 4.2 and the above equations, we can represent the horizontal relationship between the tables drawn from three genomes and the tables due to the 4th genome, and all possible OPs among the total Γ and the rest of the new tables due to the 4th genome. These equations are generalized into formula (2).

Theorem 4.3. *Formula (3) clusters all possible OPs with vertical relationships among the prior tables and the new tables due to additional genomes, and all possible OPs among the total table Γ and the rest of the new tables due to additional genomes.*

Proof. Let's consider Fig. 13, and the following expressions.

$$\Gamma_2^{n+1} = \Gamma_2^n \cup (T_{1(2)} \cap T_{(2)4}) \quad \Gamma_3^{n+1} = \Gamma_3^n \cup (T_{1(3)} \cap T_{(3)4}) \quad \Gamma_3^{n+1} = \Gamma_3^n \cup (T_{2(3)} \cap T_{(3)4})$$
$$\Gamma_4^{n+1} = \Gamma_4^n \cup (\Gamma_4^n \cap T_{1(4)}) \text{ ,} \quad \Gamma_4^{n+1} = \Gamma_4^n \cup (\Gamma_4^n \cap T_{1(4)}) \text{ ,} \quad \Gamma_4^{n+1} = \Gamma_4^n \cup (\Gamma_4^n \cap T_{1(4)})$$
$$\Gamma_1^{n+1} = \Gamma_1^n \cup (\Gamma_1^n \cap T_{(1)4)}) \quad \Gamma_1^{n+1} = \Gamma_1^n \cup (\Gamma_1^n \cap T_{(1)4)}) \quad \Gamma_2^{n+1} = \Gamma_2^n \cup (\Gamma_2^n \cap T_{(2)4)})$$
$$\Gamma_4^{n+1} = \Gamma_4^n \cup (\Gamma_4^n \cap T_{3(4)}) \quad \Gamma_4^{n+1} = \Gamma_4^n \cup (\Gamma_4^n \cap T_{2(4)}) \quad \Gamma_4^{n+1} = \Gamma_4^n \cup (\Gamma_4^n \cap T_{2(4)})$$

By Lemma 4.1, 4.2, and the above expressions, we can represent the vertical relationship between the tables drawn from three genomes and the tables due to the 4th genome, and all possible OPs among the total Γ and the rest of the new tables due to the 4th genome. If these equations are generalized, formula (3) is formed.

Theorem 4.4. *Formula (4) clusters all possible OPs among the rest of the new tables due to additional genomes, and all possible OPs among the total table Γ and the rest of the new table due to additional genomes.*

Proof. Let's consider Fig. 14, and the following expressions.

$$\Gamma_4^{n+1} = \Gamma_4^n \cup (T_{1(4)} \cap T_{2(4)})$$
$$\Gamma_4^{n+1} = \Gamma_4^n \cup (\Gamma_4^n \cap T_{3(4)})$$
$$\Gamma_4^{n+1} = \Gamma_4^n \cup (T_{1(4)} \cap T_{3(4)})$$
$$\Gamma_4^{n+1} = \Gamma_4^n \cup (T_{2(4)} \cap T_{3(4)})$$

By Lemma 4.1, 4.2 and the above equations, we can represents all possible combination between only the tables due to the 4th genome, and all possible OPs among the total table Γ and the rest of the new table due to the 4th genome. The equations are generalized as formula (4).

5 Conclusion

To infer the unknown protein functions, it is very useful to cluster orthologous proteins from the completely sequenced genomes. COG database is an OPCs constructed with the best reciprocal BLAST hits with biological analysis. Unfortunately, it needs much time to detect homology between multiple complete-genomes by using BLAST algorithm with manual work. As the number of complete genomes especially rises, it needs much more time and effort to cluster them. Remm *et. al.*[16] made a fully automatic program to detect OPs between two genomes to attack this problem. For extending their work about two complete-genomes to multiple complete-genomes, we suggested an automatic method to cluster OPs without almost manual work from multiple complete-genomes. We also proved, mathematically, all possible combinations of orthologous protein lines with common genes, which are drawn from multiple complete-genomes by using INPARANOID.

The ClusteringGenomes algorithm of Fig. 6 is $O(n^4)$. This is not reasonable for a large n value. Future work will address this problem by using parallel computing or others.

References

1. Fitch, W. M.: Distinguishing homologous from analogous proteins. Syst. Zool., 19 (1970) 99-113.
2. Tatusov, R. L., Koonin, E. V., Lipman, D. J.: A genomic perspective on protein families. Science, 278(5338) (1997) 631-637.
3. Tatusov, R. L., Galperin, M. Y., Natale, D. A., Koonin, E. V., et al.: The COG database: a tool for genome-scale analysis of protein functions and evolution. Nucleic Acids Research, 28 (2000) 33-36.
4. Altschul, S. F., Gish, W., Miller, W., Myers, E. W. et al.: Basic local alignment search tool. J. Mol. Biol., 215 (1990) 403-410.
5. Tatusov, R. L., Fedorova, N. D., Jackson, J. D., Aviva R Jacobs, A. R. et al.: The COG database: an updated version includes eukaryotes. BMC Bioinformatics, 4, (2003) 41.
6. Chervitz, S. A., Aravind, L., Sherlock, G., Ball C. A. et al.: Comparison of the complete protein set of worm and yeast:orthology and divergence. Science, 282 (1998) 2022-2028.
7. Rubin, G. M., Yandell, M. D., Wortman, J. R., Gabor Miklos, G. L. et al.: Comparative genomics of the eukaryotes. Science, 287 (2000) 2204-2215.
8. Wheelan, S. J., Boguski, M. S., Duret, L., Makalowski, W.: Human and nematode orthologs – lessons from the analysis of 1800 human genes and the proteome of Caenorhabditis elegans. Gene, 238 (1999) 163-170.
9. Mushegian, A. R., Garey, J. R., Martin, J., Liu, L. X.: Large-scale taxonomic profiling of eukaryotic model organisms: a comparison of orthologous proteins enclosed by the human, fly, nematode, and yeast genomes. Genome Res., 8 (1998) 590-598.

10. Kanehisa M., Peer B.: Bioinformatics in the post-sequences era. nature genetics supplement, 33 (2003) 305-310.
11. Bork P., Koonin E. V.: Predicting functions from protein sequence-where are the bottlenecks?. Nat. Genet., 18 (1998) 313-318.
12. Eisen J. A.: Phylogenomics:improving functional predictions for uncharacterized genes by evolutionary analysis. Genome Res., 8 (1998) 163-167.
13. Galperin M. Y., Koonin E. V.: Source of systematic error in functional annotation of genomes: domain rearrangement, nonorthologous gene displacement and operon disruption. In Silico Biol., 1 (1998) 55-67.
14. Kimmen S.: Phylogenomic inference of protein molecular function: advances and challenges. Bioinformatics, 20 (2004) 170-179.
15. Bono, H., Goto, S., Fujibuchi, W., Ogata, H. et al.: Systematic Prediction of Orthologous Units of Genes in the Complete Genomes. Genome Inform Ser Workshop Genome Inform., 9 (1998) 32-40.
16. Remm, M., Storm, C. E., Sonnhammer, E. L.: Automatic Clustering of Orthologs and in-paralogs from Pairwise Species Comparisons. J. Mol. Biol., 314 (2001) 1041-1052.
17. Montague, M. G., Hutchison III, C. A.: Gene content phylogeny of herpersviruses. PNAS, (2000) 5334-5339.
18. Stuart, J. M., Segal, E, Koller, D., Kim, S. K.: A Gene-Coexpression Network for Global Discovery of Conserved genetic Modules. Science, 302 (2003) 249-255.

A Novel Clustering Method for Analysis of Gene Microarray Expression Data

Fei Luo and Juan Liu*

School of Computer Science, Wuhan University, Wuhan 430079, China
Luofei_whu@126.com, liujuan@whu.edu.cn

Abstract. As the gene sequencing technology has been maturating, genomes of more and more organisms and genetic sequences are available in international repositories. However, the biological functions of most of these genes remain unknown. Microarray technology opens a door to discover information of underlying genes and is widely used in basic research and target discovery, biomarker determination, pharmacology, target selectivity, development of prognostic tests and disease-subclass determination. Clustering is one of the typical methods of analyzing microarray data. By clustering the gene microarray expression data into categories with similar profiles, genes with similar function can be focused on. There are many clustering methods used for the analysis of gene microarray data. However, they usually suffer from some shortcomings, such as sensitive to initial input, inappropriate grouping, difficult to discover natural or near optimal clusters, and so on. In this paper, we propose a novel clustering method to discover the optimal clusters by searching PPVs (Pair of Prototype Vector). The experiment results show that our method works very well.

1 Introduction

In the post-genome era, with the great improvement of sequencing technology, genomes of more and more organisms have been sequenced, which results in that millions of genetic sequences have been deposited in international repositories. However, the biological functions of most of these genes contained in such large amount of raw data remain unknown. Therefore, one main task in post-genome era is to find the functions of the genes. One way to determine the functions of these genes is through repeated measurements of their RNA transcripts. Since microarray technology, developed in 1995 by Dari Shelton as his PhD thesis at Stanford University, can give global information on transcriptions of genes and measure the genome-wide mRNA abundance in the cellular process under all kinds of experimental environment [1]. Analysis of microarray data (also called as gene expression data) produced by this technology can help to discover the gene functions. Clustering and class prediction are typical methods currently used in gene expression data analysis. By clustering analysis, one can discover 1) the gene groups with similar function, which is based on the biological premise that the biological coexpressed

* The corresponding author.

genes are likely to be coregulated and are therefore likely to perform similar biological functions; 2) the static information of gene expression such as gene expression condition in each time spot or organism, and the dynamic information that how one gene expression pattern gets related to another one as well.

Although a lot of clustering algorithms have been used in the microarray data analysis, for example, SOM [2] [3], K-Means, hierarchical [4], and some other clustering methods such as those based on information theory [5] and statistic modal [6][7], they still confront with some problems. First, some algorithms require the users to predetermine some parameter values. For example, how to set the K value in K-Means clustering and when to stop the hierarchical clustering procedure, and so forth. However, it is difficult for biologists to give suitable parameters. Second, some algorithms are sensitive to the input order of data. Take the K-Means method as an example once again, if the initial partitions are not chosen carefully enough, it will run the risk of converging to a local minimum rather than the global minimum solution. Third but the most important, there are seldom algorithms can discover natural clustering number or near-natural one automatically. Recently some efforts have been made to come up with such shortcomings [8] [9]. They often assume that the whole set of microarray data is a finite mixture of a certain type of distributions, usually Gaussian mixture [10], and propose a static model. However, such model faces two key challenges: one is that the microarray data usually have small number of samples but with a very large amount of genes, which makes it difficult for the model to fit the data very well in the statistical way; the other one is that the expression levels for an individual gene may not be independent, which conflicts with the independent assumption of the model.

Here we notice that the natural groups of the genes have the characteristic that each group has distinguished patterns and usually the shape of patterns can be represented approximately by several typical gene patterns. If we can find some sets of typical representative genes, such that the genes in the same set have most similar expression patterns and genes from different sets have very different expression patterns, then we can easily get the clusters. Moreover, finding several representative genes, instead of directly clustering the whole expression data, also makes the time cost of the computation lower.

How many representatives should be selected? Enlightened by the idea of [11] that finding more representatives for each cluster is superior to finding only one, in this paper, we choose two, which we call as a PPV (pair of prototype vectors), to represent a cluster. According to above ideas, we propose a novel clustering method based on PPV Search Technology (PPVST), called as PPVTOC (Pair of Prototype Vector Takes One Cluster).

The measurement method of (dis)similarity between genes is very important for gene clustering. Since the widely used Euclidean distance and Pearson Correlation have some disadvantages that will be analyzed in the next section, we define a new simple distance measurement, which we call Curvature Distance.

According to the Curvature Distance, PPVTOC first iteratively finds PPVs such that the intra-distance of PPV is minimal and inter-PPV distance is maximal as possible as it can. The number of final automatically found PPVs is the number of the clusters that the genes will be partitioned into. Then each PPV competes for all rest genes into its own cluster until it reaches an optimal grouping state.

The rest of the paper is organized as following. Section 2 presents the distance measurement of gene dissimilarity. In section 3, we describe the proposed PPVTOC clustering method, and the evaluation experiment results are shown in section 4. Finally, the conclusions are drawn in section 5.

2 Curvature Distance

In microarray data, a gene expression profile can be viewed as a vector with *n* dimensions of the expression values, where *n* is the number of experiments or time points for that gene. By comparing these vectors, we can find which genes show similar (or dissimilar) data profiles across a series of experiments or time points. Given two gene expression profiles $X = (X_i)$, $Y = (Y_i)$, the most widely used method is Euclidean distance, which is given by formula (1):

$$dis_{Euclidean}(X,Y) = \frac{1}{n}\sqrt{\sum_{i=1}^{n}(X_i - Y_i)^2} \tag{1}$$

According to the Euclidean distance, the smaller the distance, the more similar the two genes are. So the data must be normalized before calculating their distance. Even though, the Euclidean distance does not consider the changing trends of the genes, which would make the distance between two completely different genes is equal to the distance between two similar genes. For example, Fig. 1 illustrates two cases (a) and (b) with equal Euclidean distances between data A and data B, however, they have similar profiles in the Fig. 1(a), yet they have completely different profiles in the Fig. 1(b). Therefore, the dynamic information that includes changing trend of two gene profiles is also what we are interested in, whereas Euclidean distance neglects such key information. So in this paper we don't adopt Euclidean distance to measure the dissimilarity between gene profiles.

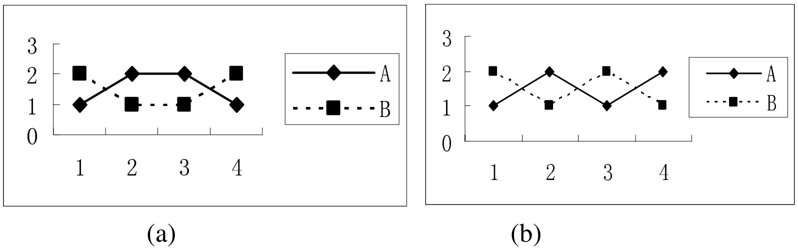


Fig. 1. A and B have equal Euclidean distance, while in (a) A and B have similar profiles, and in (b) they have completely different profiles

Pearson Correlation, which in shown as formula 2, is also widely used to measure the similarity of two gene profiles.

$$\mathrm{S(X,Y)}=\frac{1}{\mathrm{n}}\sum_{i=1}^{n}\left(\frac{X_i - X_{offset}}{\Phi_X}\right)\left(\frac{Y_i - Y_{offset}}{\Phi_Y}\right) \tag{2}$$

Where $\Phi_G = \sqrt{\sum_{i=1}^{n}\frac{(G_i - G_{offset})^2}{n}}$ and G_{offset} is the mean of vector *G*.

Pearson Correlation does not require that the data should be normalized, and it also considers the change trend of the data. The value of Pearson Correlation varies from -1 to 1. Positive value means positive correlation (similar) and negative value means negative correlation (dissimilar). However, the negative values are inconvenient to compute and understand. We hope that there is one way that not only has the value range like Euclidean distance from 0 to *x* ($x \geq 0$), but also can reflect the changing trend of gene profiles like Pearson Correlation.

Here, we define a new measurement called as Curvature Distance to measure the dissimilarity between two gene expression profiles. Curvature Distance meets the requirements of ordinary distance function definition:

$$\begin{aligned}&(I) dis(X,Y) \geq 0, dis(X,X) = 0\\&(II) dis(X,Y) = dis(Y,X)\\&(III) dis(X,Y) \leq dis(X,Z) + dis(Z,Y)\end{aligned} \tag{3}$$

In our method, supposing the microarray data S has *n* genes with *m* experiments. $X_i = (x_{i1}, x_{i2} \ldots x_{im})$ is the gene expression profile of gene *i*:

$$S = \begin{pmatrix} x_{11} & \cdots & x_{1m} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nm} \end{pmatrix}$$

In order to reflect the change of consecutive spots of a gene X_i, we define a vector $VX_i = (v_{i1}, v_{i2} \ldots v_{i(m-1)})$ corresponding to X_i, where

$$v_{ij} = \frac{(x_{i(j+1)} - x_{ij})}{h},\ 1 \leq j \leq m-1,\ h = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} | x_{ij} |}{m * n}$$

h is the mean of S, which reflects the average expression level. Then, we get a new matrix *VS* derived from matrix *S*,

$$VS = \begin{pmatrix} v_{11} & \cdots & v_{1(m-1)} \\ \vdots & \ddots & \vdots \\ v_{n1} & \cdots & v_{n(m-1)} \end{pmatrix}$$

Finally, we can define the Curvature Distance between two genes as formula (4).

$$dis_{curvature}(X_i, X_k) = \frac{\sum_{j=1}^{m-1} | v_{ij} - v_{kj} |}{m-1} \tag{4}$$

Obviously, Curvature Distance satisfies the requirements (I), (II), (III).

Applying the Curvature Distance to measure dissimilarity between A and B in Fig.1, the value of $\mathrm{dis}_{\mathrm{curvature}}$ between A and B in Fig. 1(a) is 0.89, while the value of $\mathrm{dis}_{\mathrm{curvature}}$ between A and B in Fig. 1(b) is 1.33. Comparing two distance values, the former one is less than the later one, which reflects A and B in Fig. 1(a) have better similarity than those in Fig. 1(b) and $\mathrm{dis}_{\mathrm{curvature}}$ can distinguish them.

3 PPVTOC Clustering Method

Although various clustering algorithms have been used for gene expression data analysis, they suffer from some drawbacks such as difficulty to discover natural clustering number, sensitivity to initialization of input data, and so on. In [12], the authors suggested that using one-prototype-to-take-one-cluster (OPTOC) is superior to using one-prototype-take-multiple-cluster (OPTMC). Fig. 2 shows the main ideas of OPTMC and OPTOC mentioned in [12]. OPTMC means that if the number of prototypes is less than number of the natural clusters in the dataset, there must be at least one prototype to win patterns from more than two clusters. Obviously, OPTMC can't get an ideal clustering result. In contrast, the OPTOC paradigm allows one prototype to characterize only one natural cluster in the dataset. Therefore, one natural cluster can be found by searching its relevant prototype. However, using only one prototype to represent the cluster is not enough [11]. Since there are more than one typical characteristic element to represent their relevant natural cluster, it is more possible to find natural clusters by discovering more representative information. So in this paper, we use two representatives (PPV) to represent the natural cluster.

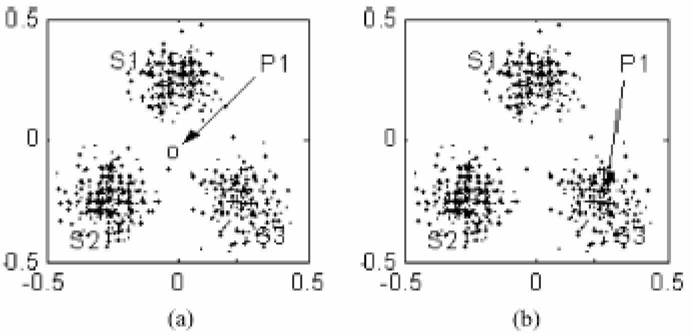


Fig. 2. Two learning methods: OPTMC versus OPTOC. (a) One prototype takes the center of three clusters (OPTMC). (b) One prototype takes one cluster (OPTOC) and ignores the other two clusters.

3.1 Finding PPVs

A basic principle of clustering is to make distance of intra-cluster minimal and distance of inter-cluster maximal as possible as we can. Therefore, PPV should be vectors that are most possibly grouped into the same cluster and are the most representative vectors of their cluster. Let S denote the whole dataset. So we first find two vectors X_k, X_l with $dis_{curvature}(X_k, X_l) = \underset{1\le i\le n, 1\le j\le n}{Min}\{dis_{curvature}(X_i, X_j)\}$. Undoubtedly, they should be a PPV according to intra-cluster distance minimization principle and added into V_s, for they are most similar vectors in S. We denote this pair as PPV_0. Next, in order to find another new PPV, we first search a vector $X_{furthest}$ that is farthest to PPV_0 based on the principle of inter-cluster distance maximization. viz. $X_{furthest} = \underset{1\le i\le n}{Max}\{dis_{curvature}(X_i, PPV_0)\}$, considering that $X_{furthest}$ will be finally grouped into some cluster, we attempt to find a new PPV of the cluster that $X_{furthest}$ belongs to according to the method described below.

Method to discover new PPV

```
Input: $V_s$
1. Initialization:
   D=10000
   Dt=10000                      /*Record distance and 10000 is great enough*/
2. Find FV of $V_s$ by $||FV-V_s||=Max\{||X-V_s||\}$, $X,FV \in V_{us}$
   D=$||FV-V_s||$                /*$||X-V_s||$ indicates the distance between X and $V_s$*/
3. Find NV of FV by $||NV-FV||=Min\{||X-FV||\}$, $X \in S$
   Dt=$||FV-NV||$
4. If(Dt<D)
   {
   D=Dt
   FV=NV
   Go to step 3
   }
5. If(NV$\in V_s$ or Stop( )=TRUE)
     Go to step 7
6. Add NV and its nearest vector into $V_s$ and they are the new PPV
   Remove them away from $V_{us}$
7. End
```

Note: V_s: the set of vectors that have been selected as a member of some PPVs, at beginning $V_s = \varnothing$. Initially, the most similar vectors in dataset will be selected into V_s as the PPV_0.
V_{us}: the set of vectors that have not been selected as a member of any PPV, and at beginning V_{us}=S.

Given V_s, above procedure tries to find out new possible PPV. It should be far from all existed PPVs in V_s for inter-cluster distance should be maximal as possible as it can, so the method finds the vector, called FV, which is furthest from all PPVs in V_s. Here we denote the minimum value among distance between X and each PPV in V_s as the distance between X and V_s, viz. $\|X-V_s\|=\text{Min}\ \{\|X-PPV_k\ \|\}$, $PPV_k \in V_s$, $X \in V_{us}$. Through defining in this way, it can put FV in the area far from all existed PPVs to search new possible PPV. Based on the logic premise that the FV should be divided into some cluster at last and the FV is most possible to be grouped into the same cluster together with vector NV that is nearest to FV. There are two conditions that can decide the searching procedure to stop. One is if there is no new NV found, the procedure stops searching. The other one depends on the function of stop( ) which is operated by users through observing the judging diagram that describes the distance between the new found possible PPV and V_s.

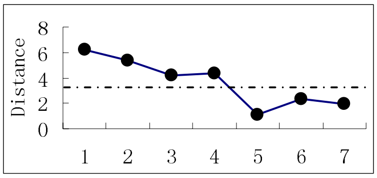


Fig. 3. Judging diagram

After finding a new possible PPV every time, we compute the distance between it and V_s. According to the algorithm, this PPV must be far from all existed PPVs in V_s as possible as it can. Therefore, if all natural clusters are not found, the new possible PPV should represent a natural cluster different from those that PPVs in V_s represent. The distance between new PPV and V_s is approximately equivalent to inter-cluster distance, whereas, if all natural clusters are found, the distance between new PPV and V_s is approximately equivalent to intra-cluster distance for there is no other natural cluster can be represented by the new possible PPV. Obviously, the inter-cluster distance is always greater than intra-cluster distance. So the Judging diagram will show like the Fig. 3. At point 4, it is a dividing point which can indicate the natural number. Because the series of distance before the dividing point reflect inter-cluster distance, while series of distance after it reflect intra-cluster distance and the dividing point divides the diagram into two distinct parts. In Fig. 3, the dataset should contain five natural clusters, which are represented by four searched PPVs and one initial PPV. Therefore, the searching procedure usually stops according to function of stop( ) when user observes the dividing point in the judging diagram.

For large dataset, beginning with an initial PPV, the method can finally discover several PPVs. The number of found PPVs is the final cluster number that the genes will be grouped into.

3.2 Clustering Based on PPVs

The found PPVs, each of which represents a cluster, will compete for rest vectors in V_{us}. For example, a vector X in V_{us} will be divided into cluster whose PPV has nearest distance with X.

In order to avoid losing any PPV in S, after finding out a set of PPVs and finishing the final clustering, we can also use (5) to assist us to validate the accuracy of the natural cluster number decided by the dividing point in the judging diagram.

$$J_e = \sum_{i=1}^{c} \sum_{y \in \Gamma_i} \| y - PPV_i \| \qquad (5)$$

Where, C is the number of PPVs, Γ_i represents cluster i.

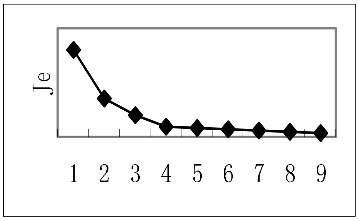


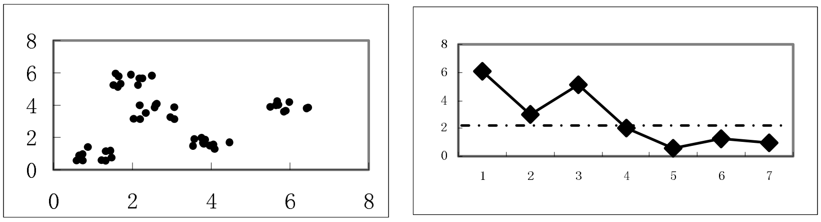
Fig. 4. The number of well grouped clusters is inflexion

In Fig. 4, the value of Je decreases along with C increasing. Supposing that the natural cluster number of dataset S is K, Je will sharply decrease when the cluster number increase from K-1 to K. but if one natural cluster is artificially split into two clusters, *Je* will decreases little. So the separation procedure should stop at the inflexion point, which should accord with natural cluster number decided by the dividing point in the judging diagram.

4 Evaluation Experiments

4.1 Experiment on Artificial Data

In order to evaluate the performance of our PPVTOC clustering method, we generate a testing dataset including five clusters and the size of each cluster is 10. We number data in the first cluster from 1 to10, data in second cluster from 11 to 20 and the rest can be deduced by analogy. Each data in the testing dataset is two dimensions so that the distribution of the dataset can be easily observed.



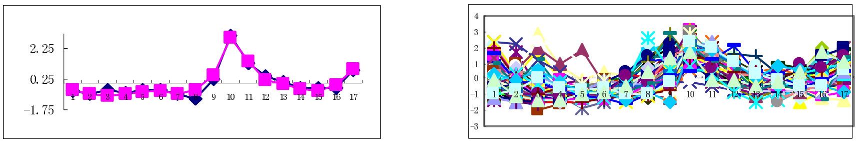
(a) (b)

Fig. 5. (a) The distribution of dataset and (b) the judging diagram

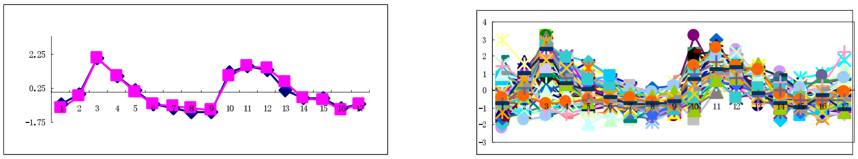
Appling PPVTOC clustering method to the testing dataset and through seven times iterative searching, we get the judging diagram shown in Fig. 5(b). From the judging diagram, we can clearly find the point 4 is the dividing point. It means the dataset should be separated into five natural clusters, which accords with the distribution of dataset shown in Fig. 5(a). At the same time, we record the order of FV of V_s once a new PPV is added into it. They are 24, 17, 32, 48, 15, 42, and 27, beginning with the initial PPV, whose vectors are 1 and 4. FV falls in different cluster area each time so that it won't be able to compete with exited PPVs, which proves the validity of the mechanism of searching possible PPVs.

4.2 Experiment on Yeast Cell Cycle Data

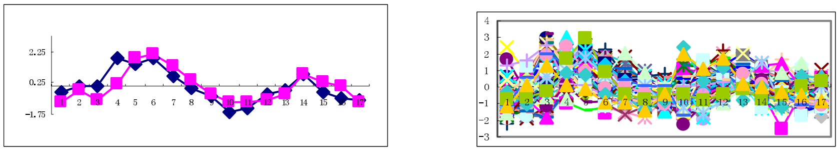
In this part, we use the yeast cell cycle data to evaluate our method. The yeast cell cycle dataset was published by Cho et al. [13]. It contains the expression profiles of



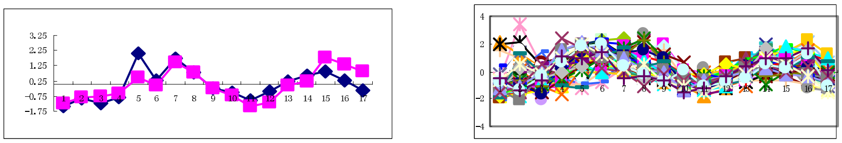
(a) Cluster 1



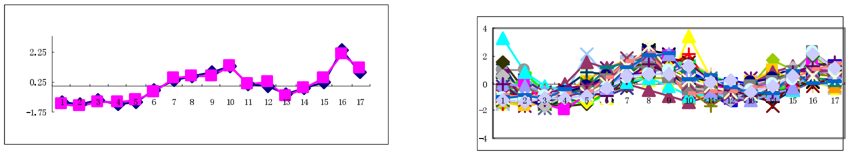
(b) Cluster 2



(c) Cluster 3



(d) Cluster 4



(e) Cluster 5

Fig. 6. Left part is profiles of 5 PPVs found by PPVST and right part is clustering results

6220 genes over 17 time points (treatments) taken at 10 minute intervals, covering nearly two cell cycles. In fact, the yeast cell cycle data set has established itself as a standard for the assessment of newly developed clustering algorithms. This dataset is very attractive because a large number of genes contained in it are biologically characterized and have been assigned to different phases of the cell cycle. The entire dataset is available at http://cellcycle-www.stanford.edu. For comparison reason, here we use a subset of 384 genes, which was ever studied by Yeung et al. [6] and is available at http://www.cs.washington.edu/homes/kayee/model. All the 384 genes were assigned to one of the 5 clusters by the original investigators Yeung et al. [6].

Through computing the distance between each data using the curvature distance and applying the PPVTOC clustering method, the found PPVs and the clustering results are shown in Fig. 6. From the subset of 384 genes, our method discovered 5 PPVs. Comparing our results with those of Yeung et al., the clustering number is the same as Yeung et al. and the profile of each PPV basically matches the profile of its cluster. However, we also noticed that the third PPV shown in Fig. 6(c) is different from Yeung et al. In Yeung's experiment, although the third PPV is divided into different clusters, from the gene microarray profile they are so similar that should be in the same cluster.

5 Conclusions

In this paper, we proposed a new clustering method PPVTOC for analysis of gene microarray data. In order to discover dynamic information underlying gene microarray data such as gene coordinate expression, we discussed the disadvantages of the popular (dis)similarity measurement methods and gave a new simple distance function; this new type of distance can properly reflect the dissimilarity and coordinateness among genes. According to the principle of intra-cluster distance minimization and inter-cluster distance maximization, we proposed PPVST to search PPVs that are the representatives of natural clusters. With the judging diagram, users can easily obtain the natural clustering number automatically without any parameters given. The evaluation results on artificial data and yeast cell cycle data show that our method can discover optimal clustering number and get the proper results.

Next step, our method needs to be validated with larger dataset and under more complex situation. Moreover, we will make use of the clustering results to do further researches [14] [15]. For example, we can examine those genes that cluster together and assign a function or value to the cluster and combine it with other knowledge about known transcription factors, regulatory elements, sequence or structure information, or assigned gene functions to search new associations in biology.

Acknowledgments

This work was supported by the National Nature Science Foundation of China (60301009), Chenguang Project of Wuhan city (211121009).

Reference

1. M. Schena, D. Shalon, R.W. Davis, "Quantitative monitoring of gene expression patterns with a DNA microarray," Science, vol. 270, pp.467-470, 1995
2. T. Kohonen, "The self-organizing map," Proceedings of the IEEE, vol. 78, pp. 1464-1479, 1990
3. J. Wang, J. Delabie, H. Aasheim, "Clustering of the SOM easily reveals distinct gene expression patterns: results of a reanalysis of lymphoma study," BMC Bioinformatics 3, 36, 2002
4. J. Quackenbush, "Computational analysis of microarray data," Nature Reviews Genetics, vol. 2, pp.418-427, 2001
5. ALN Fred, A.K. Jain, "Data Clustering using Evidence Accumulation," ICPR 2002 , pp. 276 -280, 2002
6. K.Y. Yeung, C. Fraley, A. Murua, A.E Raftery, "Model-based clustering and data transformations for gene expression data," Bioinformatics, vol. 17, pp. 977-987,2001
7. D.B. Allison, G.L. Gadbury, M Heo, "A mixture model approach for the analysis of microarray gene expression data," Computational Statistics and Data Analysis, vol. 39, pp. 1-20, 2002
8. R. Tibshirani, G. Walther, T. Hastie, "Estimating the number of data clusters via the gap statistic," Journal of the Royal Statistical Society Series B, pp. 411-423 ,2001
9. R.J. Hathaway, J.C. Bezdek, Y. Hu, "Generalized Fuzzy c-Means Clustering Strategies Using Lp Norm Distances," IEEE Trans. on Fuzzy Systems, vol. 8, pp. 576-582, 2000
10. Y. Qu, S. Xu, "Supervised cluster analysis for microarray data based on multivariate Gaussian mixture," Bioinformatics, vol. 20, pp.1905-1913,2004
11. M.Halkidi, M.Vazirgiannis, "Clustering validity assessment using multi representatives," Proceedings of SETN Conference, Thessaloniki, Greece, 2002
12. Y. J. Zhang, Z. Q. Liu, "Self-Splittng competitive learning: A new on-line clustering paradigm," IEEE Trans. Neural Networks, vol. 13, pp.369-380, 2002.
13. R.J. Cho, M.J. Campbell, E.A Winzeler, "A genome-wide transcriptional analysis of the mitotic cell cycle,"Molecular Cell, vol. 2, pp. 65-73, 1998
14. R.B. Altman, S. Raychaudhuri, "Whole-genome expression analysis: challenges beyond clustering," Curr. Opin. Struct. Biol, vol.11, pp.340-347, 2001
15. C.C. Gu, D.C. Rao, G. Stormo, "Role of Gene Expression Microarray Analysis in Finding Complex Disease Genes," Genetic Epidemiology, vol. 23, pp. 37-56, 2002

Heterogeneous Clustering Ensemble Method for Combining Different Cluster Results

Hye-Sung Yoon[1], Sun-Young Ahn[1], Sang-Ho Lee[1],
Sung-Bum Cho[2], and Ju Han Kim[2]

[1] Ewha Womans University, Department of Computer Science and Engineering,
Seoul 120-750, Korea
comet@ewhain.net, lovesy@ewhain.net, shlee@ewha.ac.kr
[2] Seoul National University Biomedical Informatics (SNUBI),
Seoul National University College of Medicine, Seoul 110-799, Korea
csb@medigate.net, juhan@snu.ac.kr

Abstract. Biological data set sizes have been growing rapidly with the technological advances that have occurred in bioinformatics. Data mining techniques have been used extensively as approaches to detect interesting patterns in large databases. In bioinformatics, clustering algorithm technique for data mining can be applied to find underlying genetic and biological interactions, without considering prior information from datasets. However, many clustering algorithms are practically available, and different clustering algorithms may generate dissimilar clustering results due to bio-data characteristics and experimental assumptions. In this paper, we propose a novel heterogeneous clustering ensemble scheme that uses a genetic algorithm to generate high quality and robust clustering results with characteristics of bio-data. The proposed method combines results of various clustering algorithms and crossover operation of genetic algorithm, and is founded on the concept of using the evolutionary processes to select the most commonly-inherited characteristics. Our framework proved to be available on real data set and the optimal clustering results generated by means of our proposed method are detailed in this paper. Experimental results demonstrate that the proposed method yields better clustering results than applying a single best clustering algorithm.

1 Introduction

Bioinformatics is a combined interdisciplinary subject focused on the use of computational techniques to assist the understanding and organization of information associated with biological macromolecules. Genome sequencing projects and high-throughput technologies, like microarray experimental data, have resulted in a tremendous amount of information-rich data [4], [6].

Data mining techniques have been used extensively as approaches to uncover interesting patterns from large databases [1]. Of these, clustering analysis is one of the most important approaches, because it groups elements in a data set in terms of their similarities and does not require class label information. Genomic researchers are willing to apply clustering algorithms to gain better genetic understanding and biological information in the bio-data, because most bio-data are associated with insufficient prior knowledge. However, clustering techniques can be applied to analyze bio-data with their different characteristics. The challenge selecting the best algorithm, because variety clustering methods often lead to inconsistent results due to their own methodological bias and varying function criteria [12], [13]. In this paper, we describe a novel approach that digresses from using a single clustering algorithm for bio-data analysis.

The clustering ensemble problem recently has been introduced that partitions a set of objects without accessing its original features. This process demonstrated usefulness in improving the scalability and reliability of cluster results [5]. Rather than merely selecting a winning partition, we want to show that combining the clustering results of different clustering algorithms yields a better clustering solution than selecting results from a single clustering process alone. We also show a new heterogeneous clustering ensemble (HCE) method based on a genetic algorithm (GA) that combines different clustering results from diverse clustering algorithms. The use of GA is a probabilistic search approach that is founded on the concepts of evolutionary processes. Hence, we used GA approach to further improve clustering results in a HCE problem.

The paper is organized as follows. The prior clustering ensemble methods are reviewed in Section 2, along with a description of combined methods, a review of the importance of clustering results and a presentation of reasons to consider applying GA. Section 3 explains the proposed HCE method based on GA for bio-data applications. Section 4 reviews significant experimental results obtained by applying the proposed method. Finally, section 5 contains concluding remarks and future research ideas.

2 Related Works and Background

Generating high quality cluster results is a challenging problem in bio-data analysis because of the inherent noise that exists in experimental data and the inconsistency that exists among the different clustering algorithms. In the past, clustering analysis often has repeated execution of a clustering procedure, followed by selection of an individual solution that maximizes a user-defined criterion [2]. However, recent research has shown that combining of clustering results often yields better results.

Clustering ensemble techniques have recently been successfully applied to increase the accuracy and stability of classification in data mining [3], [10]. That being said, it remains difficult to say which clustering result is best because the same algorithm can lead to different results as a result of various repetitions and random initialization. The goal of cluster ensemble methods is to combine the results of multiple clustering algorithms to obtain higher-quality and more robust cluster results [8], [9]. One of the major issues of clustering ensemble is how to combine different clustering results. Previous studies regulated clustering results from clustering algorithms into the same number of clusters [13].

However, directly combining the same number of clustering results cannot generate a meaningful result. Therefore, a new mechanism to combine the different numbers of cluster results is needed to obtain better clustering results.

In this paper, we assume that effectively combining of clustering algorithms is an important method to improve cluster quality. We have focused on optimally exploiting the information provided by a collection of different clustering results by combining them into one final result, using a variety of methods. Applying GA is highly advantages for tasks requiring optimization and is highly effective in any situation in which many inputs (variables) interact to produce a large number of possible outputs (solutions) [8]. GA constitutes search method that also can be used both for solving problems and modeling evolutionary systems. Since it is heuristic, on one can know if the solution is totally accurate. However, most scientific problems are addressed via estimates, rather than assuming 100% accuracy.

Approach methods using GA can be classified broadly into two basic categories. The first category consists of generational GA that uses typical parameters such as roulette selection with elitism. This is a method by which the fittest potential parents are selected from a population; however, this does not guarantee that the fittest member proceeds to the next generation. The second method is the steady-state genetic algorithm that selects two individual parents by rank selection then combines them to produce one offspring, thereby replacing the worst characteristics (or traits) of a population with better characteristics. Unfortunately, the steady-state GA method has the potential of premature convergence, which occurs by quickly converging the solution set. The major difference between steady-state and generational GAs is that, for each parent of the population generated in the generational GA, there are two parents selected by means of the steady state method. Consequently, selection drifts appear twice as fast within a steady-state GA because this method first determines rank in the population and then every member receives fitness from as a result of this ranking.

Combining the strengths of the various methods counteracts the weaknesses of each system. Therefore, in this paper, we compromised with these two methods that first determined ranks of members and selected two parents for using crossover operation according to highly-overlapped objects.

3 Methods

In this section, the experimental data and methods applied in this paper are explained in detail. The overall experimental framework is illustrated in Figure 1.

3.1 Data

In this paper, CAMDA (Critical Assessment of Techniques for Microarray Data Analysis) 2006 conference data set (http://www.camda.duke.edu/camda06/datasets) were used in the current study as data set for the application of the proposed method. This data set is derived from the CDC (Center for Disease Control and Prevention) chronic fatigue syndrome (CFS) research group and contains

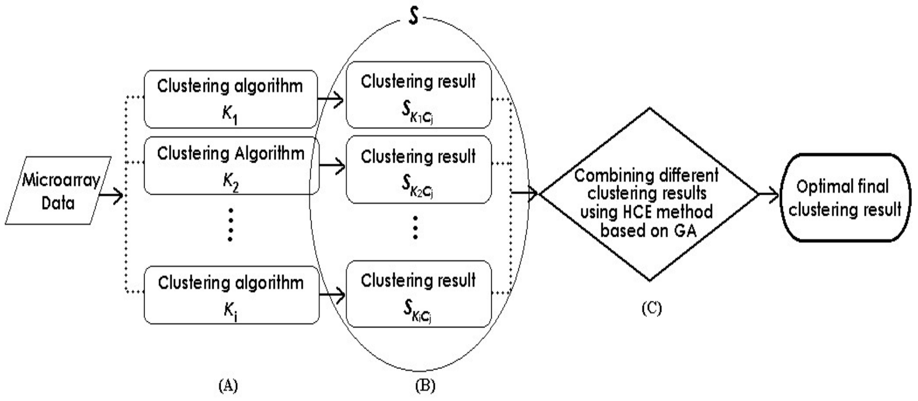

S
Microarray Data
Clustering algorithm K_1
Clustering Algorithm K_2
Clustering algorithm K_i
Clustering result $S_{K_1C_j}$
Clustering result $S_{K_2C_j}$
Clustering result $S_{K_iC_j}$
Combining different clustering results using HCE method based on GA
Optimal final clustering result
(A)
(B)
(C)

Fig. 1. Flowchart of the experimental method. (A) Apply the different types of algorithms. (B) Generate different clustering results by means of these algorithms. (C) Combine the different numbers of clustering results based on GA.

microarray, proteomics, single nucleotide polymorphism (SNP) and clinical data. In our experiments, two categories of data, microarray and clinical, were used for application and verification. The first microarray data set is a single-channel experimental data set that is composed of 20,160 genes using DNA from 177 patients. The second data set is classified 227 patients into three CFS patient subgroups (categorized by degree of clinical severity- least, middle and worst) from the CDC human subjects committee. Prior to analysis, we deal with missing values by assuming that the ratio of expression of given genes is greater than that of background intensity among microarray data, and we replaced the missing values by means of the k-nearest neighbor (kNN) method. In addition, we created a final experimental data set consisting of 19,592 genes from 169 patients; this was done by removing repeats and controls after transforming to a logarithmic ratio.

To estimate the effectiveness of the proposed method, we analyzed from 118 patient data set, which includes identical partitions about a broad range of clinical severity and microarray data, and compared our multi-dimensional clustering technique with other single clustering approaches.

CFS is a syndrome that is diagnosed on the basis of classification criteria that, mostly, are highly subjective. The illness has no diagnostic clinical signs or laboratory abnormalities, and it is unclear if it represents a single entity or a spectrum of many. Prior analyses into CFS pathogenesis have not yield further insights into the nature of this condition [7], [11]. Our own previous attempts at analysis, to data, have not yielded further insights into CFS pathogenesis either. An objective of the current study was to observe how our multi-dimensional application method deals with a condition like CFS, in which both the clinical parameters and the pathogenesis of disease is unclear. Recall that we propose to combine the strengths of different clustering algorithms to offset the weaknesses of any single algorithm.

3.2 HCE Method Based on GA Operation

Based on the work presented in Section 2, we proposed a HCE method based on GA operations to achieve optimization between different types of algorithms, K_i, and different numbers of clustering results, C_j.

The proposed HCE method must be differentiated from previous ensemble approaches. First, previous methods referred to the importance of ensemble algorithms but they were methods that did not consider the characteristics of each algorithm and dataset. Therefore, the methods fixed clustering results with the same number of clustering algorithms. In addition, highly-overlapped clustering results were assumed to indicate the final clustering result among these C_j. It goes without saying that papers applying different numbers of cluster results existed, but these investigators invariably searched for the optimal cluster number as well and reapplied the cluster number to all algorithms as a parameter.

Algorithm. HCE method based on GA operation

```
Input :

(1) The data set of $N$ data points $D = X_1, X_2,.., X_N$
(2) A set of clustering algorithms $K_i$
        - $i$ : the number of clustering algorithms available for analysis
(3) The cluster numbers $C_j$
        - the $K_i$ generates different cluster numbers $C_j$ for the data set $D$
(4) The clustering result is $S= \{Sk_1c_j, Sk_2c_j,....., Sk_ic_j\}$
        - $Sk_ic_j$ are clustering results consisting of $C_j$ numbers of the $i^{th}$ algorithm

Output :
The optimal clustering result on the data set $D$

1. Run clustering algorithm $K_i$ on the $D$
2. Construct a disjoint non-empty subsets, $SM^{(g)}$ with only 2 elements, from
   the clustering result $S$
3. Iterate $n$ until convergence (permute the clustering result of the data every
   iteration) :
   3.1 Compute fitness $F$(t) to select two parents/subsets from $SM^{(g)}$
   3.2 Crossover two parents
         - compare between the first parent clusters and the second parent
           clusters
         - use the first parent to replace the cluster of the second parent, which
           has the largest number of highly-overlapped objects
         - repeat once by borrowing a cluster from the second parent
   3.3 Replace parents by offspring from $SM^{(g)}$
```

Second, prior ensemble methods generally selected one best algorithm among application algorithms and indicated clustering results using this one application. We wish to address both of these problems in this paper.

The premise of our proposed HCE method based on GA operation is as follows. Different types of clustering algorithms initially are applied to the data. We then generate optimal clustering result sets by means of multiple crossover repetitions based on GA, so as to generate different clustering results. GA is a probabilistic search approach that is founded on the concept of evolutionary processes [8] and applied to further improve clustering results in our method. Our proposed algorithm, HCE method based on GA operation, is outlined as follows. In the current experiment, we aim to find associations between patients. Therefore, the input data of this algorithm executed a vector for each gene base on patients (samples). The output shows similar patient clusters for CFS.

The first stage of the algorithm is applying different types of clustering algorithms to the input data. From that result, we construct $SM^{(g)}$, a disjointed non-empty subset as a pair with only two elements from clustering results, S, of different clustering algorithms. The third stage is the GA application stage of the HCE method. We selected two parents as a couple, which has the largest number of highly-overlapped objects to fitness function $F(\mathrm{t})$ for crossover operation within the population $SM^{(g)}$. In clustering analysis, the objective of the crossover operation is to produce offspring from two parents such that the offspring inherit as much meaningful parental information as possible. That is, the clustering results convey important information and we need to find a way to effectively transmit meaningful information from parents onto their offspring. However, most traditional crossover operators were designed to deal with objects traits rather than clusters traits.

Hence, we present a novel crossover operation using data gleaned from multiple clustering processes, so as to exchange meaningful information among clusters efficiently and effectively. Our selection and use of fitness operation is elaborated in Section 3.3. In the 3.2 stage, the prior process is repeated by replacing two parents of the population to generate offspring after the crossover operation until an optimal $SM^{(g)}$ is formed.

The reason we used GA is that it allows for selection of more reliable clustering results and better extraction of optimal clustering. The algorithm replaces different clustering results by allowing the fitness function to identify similar cluster data subsets, dependent on the degree of influence that data should has on optimal final clustering. This fitness operation provides prior conditions by which to select two parents among clustering results from different algorithms.

3.3 Crossover Operation

We applied three clustering algorithms. To implement the initial population and comparison with existing algorithms, we applied k-means, hierarchical methods and principle component analysis (PCA) based clustering algorithm. The more complementary clustering algorithms also can be added without any changes to the architecture of the proposed framework. Thus far, we generated a population

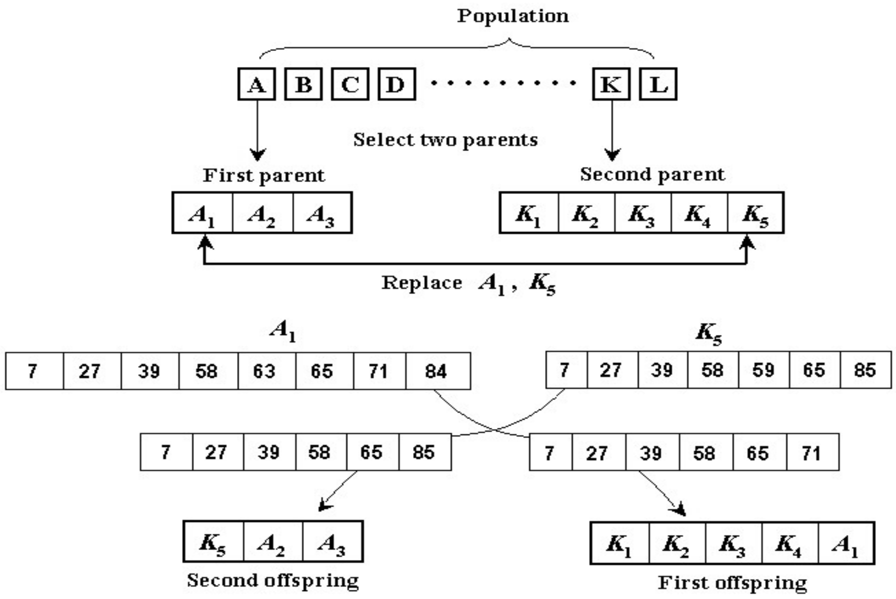


Fig. 2. Crossover operation to exchange the clustering results

totaling nine parents. That is, we created three different clustering results via the iteration and change of clusters k (3, 4 and 5) using k-means. Subsequently, the remaining two clustering algorithms were applied to yield three different clustering results.

We took the nine total different clustering results, generated by means of three clustering algorithms, and combined them with our proposed method to generate a final cluster results. We first computed the fitness function, which selects two parents, and briefly composed disjoint non-empty subsets with only two elements among nine different clustering results. For example, we can use 36 disjoint subsets with two clustering results as a pair if we have nine different clustering results. The pair with highly-overlapped objects then generates the selection of two parents during the crossover process stage.

Figure 2 explains a novel crossover approach. If we directly apply crossover operations to the ensemble problem, it may be inherited without considering clustering structures of parents, thereby eventually producing less optimal offspring [9]. For example, A and $K$are two selected parents in the initial population (Fig. 2). One parent has three clustering results (A_1, A_2, and A_3) and the other five clustering results (K_1, K_2, K_3, K_4, and K_5). First, we select one cluster, say cluster A_1, from the first parent and see that it has more highly-overlapped traits than the other two clusters (A_2 and A_3) when compared to clusters of the second parent, K. Then, we use A_1 to replace a cluster from the second parent, say K_5, which has the largest number of similarities to A_1 (objects 7, 27, 39, 58, 63, 65, 71 and 84). With replacement, those objects in A_1 (objects 63, 71 and 84) do not appear as overlapping objects in K_5, for example. However, object 63 and 84 in

A_1 appear as objects in K_2 and K_4, respectively. Consequently, objects 63 and 84 are removed so that each object belongs only to one cluster. The remaining objects in A_1 (object 71) are taken from K_5 until these objects do not appear in any other cluster. Finally, the new clustering solution is represented by the first offspring possessing traits K_1, K_2, K_3, K_4 and revised A_1. This crossover operation is repeated once by selecting a cluster from the second parent to generate the second offspring. Figure 2 shows the third stage of the proposed algorithm. Two parents are replaced by new offspring in the population in the final stage. After the replacement, we again compute fitness with the disjoint non-empty subsets using only two elements; then determine a pair of new candidates for the following parent selection; and finally repeat the stages above.

These procedures exchanges cluster traits of different clustering results and objects with highly-overlapped and meaningful information being inherited by offspring until finally we achieve an optimal final clustering result. Hence, we believe that the crossover operation we propose is a stable approach because of the invariable population of subsets and the process of combining highly-overlapped objects.

4 Experimental Results

The clinical data set from CAMDA is classified into three cluster groups: least, middle, and worst (most symptomatic) for CFS. In this paper, the AVADIS analysis tool (http://avadis.strandgenomics.com) was applied to different clustering algorithms and several parameters of the AVADIS analysis tool were applied to generate several clustering results. We also compared the results generated using AVADIS to those of our proposed method.

For data analysis and validity testing, we used 118 patients who were in common between the clinical data and microarray data sets. Table 1 represents the true classified clusters of the clinical data set.

Table 1. Classified clusters of the clinical data set. L, M and W mean least symptomatic, moderately symptomatic and most symptomatic patients number for CFS, respectively.

L	**M**	**W**	**Total**
42	51	25	**118**

Using the proposed algorithm, we discovered a final optimal result was composed of four clusters (cluster set # in Table 2) that have the largest number of fitness values among 36 disjoint subsets by means of 10,000 crossover operation repetitions. Using different fitness operations, it goes without saying that different cluster results may be captured. Four cluster results, those generated using three clustering algorithms and our proposed method, are compared.

Table 2 lists the comparisons between four clusters created by our method and four clusters of three clustering algorithms created by the parameter change.

Table 2. Clustering results comparison of the three clustering algorithms and HCE method based on GA

Microarray data set for CFS		**Clustering Results**	
Method	**Cluster set #**	**Algorithms**	**True clusters**
KM	Cluster 1	M	W
	Cluster 2	M	W
	Cluster 3	L	W
	Cluster 4	**L**	**L**
HC	Cluster 1	L	M
	Cluster 2	**L**	**L**
	Cluster 3	M	W
	Cluster 4	L	W
PCA	**Cluster 1**	**M**	**M**
	Cluster 2	L	W
	Cluster 3	M	W
	Cluster 4	**M**	**M**
HCE	**Cluster 1**	**L**	**L/M**
	Cluster 2	**M**	**M**
	Cluster 3	**M**	**M/W**
	Cluster 4	**L**	**L**

KM, HC, PCA and HCE mean k-means, hierarchical clustering, PCA-based clustering and our proposed method, respectively.

This demonstrate that the results using a clustering algorithm when we have no previously defined clusters, are not consistent with the classified three symptomatic of the clinical data set than the proposed method. To validity testing, we chose to the representative symptomatic among the largest number of similarities. The similar representative value between the proposed method and three different algorithms are written to bold characters. However, we discover that our HCE method mostly agrees with the clusters classified by the clinical data. Here, L/M and M/W are said to be clustering in the same ratio as the number of patients classified as least/middle and middle/worst.

The proposed algorithm shows that four clusters have the best fitness in disjoint non-empty subsets with two elements, and we compared them to different clustering results with four clusters. However, the clustering results of our proposed algorithm also outperformed three and five cluster results of the remaining clustering results, even though their fitness is not the best.

5 Conclusions and Future Work

Since a huge amount of gene expression data is produced by microarray experiments, a clustering technique that combines similar samples can be highly effective. The combined cluster results can find better clustering results than those obtained when using single cluster results alone.

In this paper, we considered characteristics of bio-data and clustering algorithms to present optimal clustering results by combining different types of clustering algorithms. Additionally, we proposed a HCE approach to generate optimal clusters, by newly-designing and applying the crossover operation of the genetic algorithm. The proposed method appears useful for understanding clustering results by combining several clustering algorithms for a related bio-data set.

Experiments with real microarray data show that this method can search for possible solutions effectively and improve the effectiveness of cluster analysis using crossover operations, which generate clusters of highly-overlapped traits.

We also observed that the proposed HCE method increases performance as more repetitions are added. We need not remove objects for preprocessing and fix the same cluster numbers to the first application step because the genetic algorithm is rapidly executed. Therefore, it can extract more reliable results than other clustering algorithms. In addition, clustering algorithm is an unsupervised learning method that appears useful in identifying experimental results in the absence of prior knowledge. Thus, combining different clustering algorithms by considering bio-data characteristics and analysis of clustering results also can overcome the instability inherent in clustering algorithm problems.

The experimental methods introduced in this paper suggest several avenues for future research. One direction would be to optimize cluster results by combining different bio-data sources in multi-source bio-data sets. Another would be applying different clustering algorithms under the assumption of no prior knowledge, since only one data source is used for the fitness operation. Therefore, we plan to design a proper fitness operation and novel analysis method for analysis of combined multi-source bio-data. Lastly, another important task would be to develop a theoretically and experimentally justified verification system to handle disparate data.

References

1. Alexander, P. T., Behrouz, M-B., Anil, K. J., William, F. P.: Adaptive clustering ensembles. Proceedings of the International Conference on Pattern Recognition, **1** (2004) 272–275
2. Banerjee, A., Krumpelman, C., Basu, S., Mooney, R., Ghosh, J.: Model-based overlapping clustering. Proceedings of the International Conference on Knowledge Discovery and Data Mining, (2005) 532–537
3. Greene, D., Tsymbal, A., Bolshakova, N., Cunningham, P.: Ensemble clustering in medical diagnostics. Proceedings of the 17th IEEE Symposium on Computer-Based Medical Systems, (2004) 576–581
4. Jaewoo, K., Jiong, Y., Wanhong, X., Pankaj, C.: Integrating heterogeneous microarray data sources using correlation signatures. Proceedings of the Data Integration in the Life Sciences , **LNBI 3615** (2005) 105–120
5. Jouve, P. E., Nicoloyannis, N.: A new method for combining partitions, applications for distributed clustering. Proceedings of the International Workshop on Parallel and Distributed Machine Learning and Data Mining, (2003)
6. Kasturi, J., Acharya, R.: Clustering of diverse genomic data using information fusion. Bioinformatics, **21** (2005) 423–429

7. Kenneth, J. R., Suzanne, D. V., Ellen, B., William C. R.: The economic impact of chronic fatigue syndrome. Cost Effectiveness and Resource Allocation, **2** (2004)
8. Liu, J. J., Cutler, G., Li, W., Pan, Z., Peng, S., Hoey, T., Chen, L., Ling X. B.: Multiclass cancer classification and biomarker discovery using GA-based algorithms. Bioinformatics, **21** (2005) 2691–2697
9. Patrick, C.H. Ma., Keith, C.C. Chan.: Discovering clusters in gene expression data using evolutionary approach. Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence, (2003) 459–466
10. Qiu, P., Wang, Z.J., Liu, K.J.: Ensemble dependence model for classification and prediction of cancer and normal gene expression data. Bioinformatics, **21** (2005) 3114–3121
11. Whistler, T., Unger, E. R., Nisenbaum, R., Vernon, S. D.: Integration of gene expression, clinical, and epidemiologic data to characterize Chronic Fatigue Syndrome. Journal of Translational Medicine, **1** (2003)
12. Xiaohua, H.: Integration of cluster ensemble and text summarization for gene. Proceedings of the 4th IEEE Symposium on Bioinformatics and Bioengineering, (2004) 251–258
13. Xiaohua, H., Illhoi, Y.: Cluster ensemble and its applications in gene expression. Proceedings of the Asia-Pacific Bioinformatics Conference, **29** (2004) 297–302

Rule Learning for Disease-Specific Biomarker Discovery from Clinical Proteomic Mass Spectra

Vanathi Gopalakrishnan[1], Philip Ganchev[1], Srikanth Ranganathan[2], and Robert Bowser[2]

[1] Center for Biomedical Informatics, University of Pittsburgh, Pittsburgh, PA 15260, USA
{vanathi@cbmi.pitt.edu, philip@cs.pitt.edu}
[2] Department of Pathology, University of Pittsburgh, Pittsburgh, PA 15260, USA
bowserrp@upmc.edu

Abstract. A major goal of clinical proteomics is the identification of protein biomarkers from mass spectral analyses of fairly easily obtainable samples such as blood serum, urine or cerebrospinal fluid from patient populations. It is hoped that such protein biomarkers can be utilized for early detection of disease and examined further for potential therapeutic use. In this paper, we present the process for successful discovery of biomarkers that are indicators of a chronic neurodegenerative disease of motor neurons, called Amyotrophic Lateral Sclerosis; from application of rule learning to the analysis of proteomic mass spectra from cerebrospinal fluid samples. We have implemented a wrapper-based rule learning framework within which the massive number of features that accumulate from mass spectral analyses of clinical samples can be evaluated by repeated invocation of a rule learner. Our framework facilitates evidence gathering as indicated in this case study, and can speed up disease-specific biomarker discovery from clinical proteomic mass spectra.

1 Introduction

It is widely believed that early detection of fatal and chronic diseases leads to better disease-management options for the diagnosed patient. High-throughput proteomic mass spectral technologies are rapidly becoming popular choices for this task. In clinical proteomics, biological samples such as blood serum or cerebrospinal fluid (CSF) are collected from populations of patients with and without a particular disease; and analyzed to create protein and peptide expression profiles for that disease. Apart from disease profiling, an enormous challenge in biomedical research involves the search for a set of protein biomarkers that can be used for early identification of the disease. Biomarkers are biological molecules that are indicators of physiologic state and also of change during a disease process. The utility of a biomarker lies in its ability to provide an early indication of the disease, to monitor disease progression, to provide ease of detection, and/or to provide a factor measurable across populations [1]. The problem of accurate detection of biomarkers from a multitude of data is therefore of utmost significance to biomedical researchers. The number of samples is often very small in comparison to the number of features present within the mass spectrum of each sample, making it difficult to draw meaningful conclusions from the data.

Mass spectrometric (MS) technology now allows for the simultaneous resolution of many tens of thousands of protein and peptide species in body fluids, leading to a revolution in protein-based diagnostics [2]. A recent MS-based technology is called surface-enhanced laser desorption/ionization (SELDI), where protein chip arrays are used to provide a variety of surface chemistries for researchers to optimize protein/peptide segregation, capture and analysis. The chemistries on these chips include classical chromatographic surfaces such as hydrophobic for reversed-phase capture, cation and anion exchange surfaces, and immobilized metal affinity capture (IMAC) for capturing metal-binding (e.g., transcription factors and other zinc/copper/nickel-binding) proteins. Bound proteins are liberated from the chips by ionization. Mass separation is achieved by Time-of-Flight (TOF). The mass spectrometer measures the mass-to-charge (m/z) of the protein or peptide (it is hoped that whole proteins are being ionized). The mass spectrum from analysis of each sample then comprises of tens of thousands of m/z values for each of which a relative abundance measure is assigned depending on the number of analytes detected at each m/z value based on the time-of-flight.

ProteinChip SELDI-MS technology has been utilized in several recent studies aimed at disease profiling and identification of biomarkers [3-7]. Software techniques that can analyze these massive mass spectral datasets are fast accumulating [8, 9]. A recent review of the literature on promising results from surface-enhanced laser desorption/ionisation time of flight (SELDI-TOF-MS) for cancer proteomics examines proteomic profiles of control and disease states to find biomarkers for early diagnosis [10]. In this paper, we present a rule-based analysis of a small sample dataset obtained from SELDI-TOF-MS analysis of cerebrospinal fluid (CSF) samples leading to identification of three biomarkers for Amyotrophic Lateral Sclerosis (ALS) in CSF. Currently in the United States, there is only one FDA approved drug that prolongs life by 2 to 3 months for this debilitating motor neuron disease. The median survival rate for diagnosed patients with this disease is three to five years. Our analysis comprised of proteins/peptides only in the low molecular weight range between 2 and 20 kilo daltons (kDa). The data from SELDI-TOF-MS is considered to be more accurate for this range of molecular weight.

2 Methods and Dataset

The overall framework for biomarker discovery that we adopted is presented in Figure 1. Clinical cerebrospinal fluid (CSF) samples obtained by lumbar puncture from control subjects and patients with ALS are subjected to mass spectral analysis using Ciphergen ProteinChips (Ciphergen Biosystems, Inc. Palo Alto, CA, USA). The good quality spectral data from the chips are then analyzed using wrapper-based rule learning and other statistical methods. The results along with the rule-based models are then examined by both computational and experimental users to gather evidence for potential biomarkers as indicated by the m/z values. Once sufficient evidence is obtained, identification of the specific proteins or peptides is carried out by enriching peaks and separating proteins using ion-exchange chromatography. The eluted proteins are digested with an enzyme called trypsin that cuts proteins and peptides in specific places. The tryptic digests are then subjected to two methods of protein

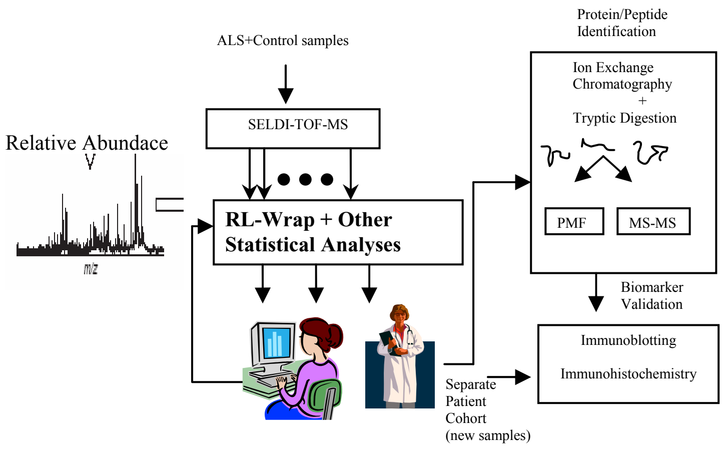

ALS+Control samples
SELDI-TOF-MS
Relative Abundace
m/z
RL-Wrap + Other Statistical Analyses
Protein/Peptide Identification
Ion Exchange Chromatography + Tryptic Digestion
PMF
MS-MS
Biomarker Validation
Immunoblotting
Immunohistochemistry
Separate Patient Cohort (new samples)

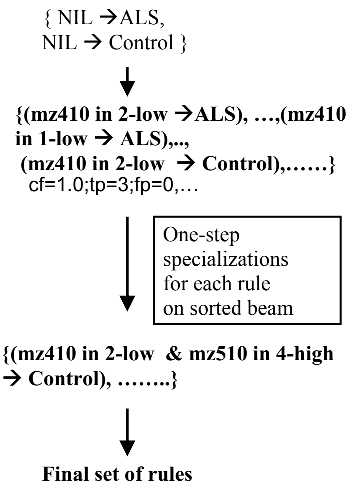
Fig. 1. Overview of our methodology for biomarker discovery

identification – Peptide Mass Fingerprinting (PMF) and Peptide sequencing using tandem mass spectrometry (MS-MS). Peptide Mass Fingerprints are obtained by subjecting the tryptic digests to mass spectrometry by MALDI (matrix-assisted laser desoption/ionization) technology, and the ProFound protein database search engine was used to identify proteins that match the experimental mass spectrum. To confirm the identity of the proteins thus identified, the typtic fragments for each biomarker is analyzed by peptide sequencing using QSTAR tandem MS (MS-MS) with a ProteinChip Interface (Applied Biosystems Inc., Foster City, CA, USA). Furthermore, to validate the identities of the discovered protein biomarkers, samples from a separate cohort of subjects are used in conjunction with immunoblotting and immunohistochemistry. Detailed experimental analyses were previously described [11]. In this paper, we focus on the rule learning aspects as they pertain to influencing the decision for which m/z values are likely to add value to predictive biomarker panels for ALS.

2.1 Rule Induction and Rule Learner (RL)

Rule induction constitutes the learning of a set of classification rules (model) from training data containing a set of subjects (or objects), whose features are described as (attribute, value) pairs. Each training example is associated with a class label. This is a well studied problem in supervised machine learning and data mining research, and various kinds of rule induction algorithms have been developed over the years such as C4.5, CART and RIPPER [12, 13]. Most data mining approaches to the induction of rules falls into two categories: divide-and-conquer (used by decision tree approaches such as C4.5 and CART) and separate-and-conquer (the set covering approach used by RIPPER). The former recursively partitions the instance space of training examples until the remaining small instance space is explained by the model; while the latter induces one rule at a time and removes instances covered by this rule until no more rules can be generated. Such methods suffer from the 'splitting problem' caused by the dwindling sample size resulting in creation of classification rules with lesser statistical support. Our rule learner (RL) described below is particularly well suited for clinical applications because it samples the instance space with replacement, thereby finding classification rules that have good support in the training data. Another advantage of RL is that it can be agnostic, in that class predictions on test data can be withheld for those cases that yield insufficient evidence for any of the classes under consideration.

The RL program [14] views inductive learning as a knowledge-based problem solving activity that could be implemented in the heuristic beam search paradigm. It was first used for predicting mass spectra of complex organic molecules [15]. RL creates and searches possible rules by successive specialization, guided by data in a training set and by prior knowledge about the data (e.g., clinical diagnosis or symptoms) to define diagnostic rules [16]. RL learns a set of conditional rules of the form IF-THEN or $P_1,..,P_k$ => C, where the left-hand side (LHS) are premise clauses that describe tests on values of one or more attributes of the training set (e.g., spectral peaks). The concept conclusion (C) is an assertion that any subject with features matching the conditions on the LHS of rule is a member of a class (e.g., ALS).


{ NIL →ALS,
NIL → Control }
{(mz410 in 2-low →ALS), …,(mz410 in 1-low → ALS),..,
(mz410 in 2-low → Control),……}
cf=1.0;tp=3;fp=0,…
One-step specializations for each rule on sorted beam
{(mz410 in 2-low & mz510 in 4-high → Control), ……..}
Final set of rules

0. ((sax2404.9228 in 0.335..0.516)) ==>(GROUP = Control)
CF=0.91, TP=12, FP=0, Pos=22,Neg=18
1. ((sax13647.294 in 0.185..0.335)) ==> (GROUP = Control)
CF=0.91, TP=12, FP=0, Pos=22,Neg=18
2. ((zn8931.1149 in 0.274..0.482)) ==> (GROUP = Control)
CF=0.91,TP=12, FP=0, Pos=22, Neg=18
3. ((zn3010.6341 in 0.302..0.481)) ==> (GROUP = ALS)
CF=0.873, TP=10, FP=0,Pos=18,Neg=22
4. ((sax6879.6187 in -0.113..0.091)) ==> (GROUP = ALS)
CF=0.862, TP=9, FP=0, Pos=18,Neg=22
5. ((zn3415.6208 in 0.332..0.488)) ==> (GROUP = ALS)
CF=0.862, TP=9, FP=0, Pos=18,Neg=22
6. ((sax2459.2504 in 0.731..1.018)) ==> (GROUP = ALS)
CF=0.862, TP=9, FP=0, Pos=18,Neg=22
7. ((zn19805.987 in 0.051..0.251)) ==> (GROUP = ALS)
CF=0.764, p= 0.033, TP=4,FP=0,Pos=18, Neg=22

Fig. 2. (Left) An example of beam search in RL. Each rule on the beam has associated statistics from training data. (Right) A sample final rule set. *cf* is certainty factor, *p*-value for rules 0 to 6 is less than 0.001 (not shown), *tp* and *fp* are the number of positive and negative training examples covered by the rule.

RL learns predictive patterns by starting from rules with a single feature and adding one feature at a time to partial rules that look most promising. Each partial rule is matched against the training cases to see how many of the positive cases are correctly predicted and how many of the negative cases are incorrectly predicted, thus providing statistical guidance to the search. Rules are ordered according to statistical significance and placed on the beam according to rank. The beam width is specified apriori and is a heuristic by which unpromising partial rules are eliminated from the search.

A new rule is considered for placement on the beam as long as it matches at least one new training instance that is currently not matched by other rules on the beam (that is, the partial model).

Figure 2 depicts an example of RL's beam search with promising partial rules on the beam being specialized successively by adding conjuncts. In the example, mz410 refers to an attribute whose *m/z* value is 410 daltons; 2-low refers to its relative abundance value being 2 standard deviations below the mean. Since the relative abundance of ionized analytes is a continuous valued attribute, RL discretizes its range based on an assumption of a normal distribution; and bins the values of each data item based on the mean and number of standard deviations away from the mean.

The need for biases in learning generalizations is well described in [17]. Bias refers to the set of assumptions made by the learner regarding the target function in order to be able to generalize from observed training instances and make predictions on new instances. This includes choice of the hypothesis space, which needs to be large enough to contain a solution to the problem at hand, yet small enough to ensure good generalization [17]. Bias space search is the search for an appropriate learning bias [18]. It is called inductive bias when learning happens through inductive generalization. Some learners, such as RL, have parameters that allow some variation in their inductive bias. The set of parameter combinations is the *bias space* of the learner. RL has 12 parameters that comprise its bias space. The main four are:

1. Certainty Factor (CF) Function: A function that computes the degree of belief of a rule. It is used for evaluating rules and ranking them on the beam. Several choices of certainty factor functions are available, such as PPV or positive predictive value, PPV + Yates' correction, PPV + normalization, signal/noise ratio, Laplace accuracy estimate and Laplace + bias toward short rules.
2. Minimum CF: The minimum CF value which rules must have in order to stay on the beam.
3. Beam Width: The number of rules kept on the beam at each iteration after all the rules are evaluated. Default value is 1000.
4. Maximum Number of Conjuncts: The maximum number of attribute-value pairs in any rule's condition. This is a very useful parameter as it limits the cardinality of non-linear relationships to be examined.

RL can perform automatic bias search on parts of the bias space. For each bias, learning and testing is done using 5-fold cross validation. The model with the greatest Fisher exact test score and the bias RL used to learn it are shown to the user. What part of the bias space is explored depends on a user-definable parameter *exhaustiveness* (values are low, medium or high). A low value implies a coarser search, with fewer values tried for the minimum CF and maximum number of conjuncts parameters. Default settings imply for low exhaustiveness, a search over 21 parameter combinations with 105 models learned over 5-fold cross-validation; 105 parameter combinations creating 525 models for medium; and 504 parameter combinations creating 2520 models for high exhaustiveness. Bias search has a running time $t_{bs}(n, a, v)$ ~= $t_x(n, a, v)\ m$, where $t_x(n, a, v)$ is the time for running x-fold cross-validation on n data with a attributes with an average of v values, and m is the number of models generated during the bias search.

Once a model is learned, it can be applied to the test set using different criteria for classification. One commonly used criterion is “weighted voting”, where the predictions on the test data are made by weighting each matching rule by its certainty factor. Another criterion that can be used is to apply the matching rule with the highest certainty factor value. These criteria perform automatic conflict resolution among rules.

2.2 RL-Wrap

Due to the large number of attributes in mass spectral data, we developed and implemented an attribute selection method based on a greedy wrapper around RL, called RL-Wrap. The algorithm is as follows:

```
Phase I: Select Attributes
1. Randomly partition the set of attributes into a specified number of attribute
   subsets $\{A_i\}$. $A_i$ corresponds to a section of data.
2. Find a bias for feature selection, by performing bias space search on an
   arbitrary section $A_i$ of training data.
3. For each section $A_i$:
        a. $M_i$ = model learned on $A_i$, using the bias found in (2)
        b. Select interesting attributes into $I_i$
           $I_i$ = {attributes that appeared in $M_i$}
        c. Combine $I_i$ with all interesting attributes $I$. ($I = I \cup I_i$)
Phase II: Learn final model
4. Find a bias for learning final model, by performing bias space search on the
   training data and selected attributes I
5. Learn a model M on the training data with the bias found in (4)

Phase III: Apply model M to test data
```

In our current implementation, I_i in (step 3b) consists of all attributes referred to by the learned model, while the *Combine* step (3c) is the union of all I_i. The number of attribute subsets (or equivalently the size K of the maximal subset) is specified by the user; the default size is 1000 attributes. A Java implementation of RL is used for learning models from the training data that are subsequently applied to the test data.

2.2.1 Attribute Interestingness Criteria

We have identified three heuristic interestingness criteria for attributes in a rule-based model, described below. RL-Wrap displays information regarding these criteria to help the user evaluate sets of attributes.

(a) Sensitivity analysis by removing attribute(s): One heuristic interestingness measure we have used is based on testing how sensitive the model is to removing one or more attributes. Given a rule-based classification model, the attribute is removed from any rule whose condition contains the attribute (thus generalizing the rule), and if that leads to an empty condition, the rule is completely ignored. If the performance greatly degrades, this suggests that the attribute is very important for the model. The measures of classification performance we used in this study were classification accuracy, sensitivity, and specificity. Results from sensitivity analysis on models are presented to the user. In the current version of RL-Wrap, combinations of up to five attributes can be removed from each model to perform sensitivity analysis.

(b) Proportion correct use (PCU) in rules: The Proportion Correct Use (PCU) interestingness criterion prefers attributes appearing in rules that make many correct (*c*) and few incorrect (*w*) predictions on the test data. This measure of interestingness of an attribute *A* can be described as, $I(A) = c/(c+w+1)$ where *c* is the total number of correct predictions on the test set of rules that refer to *A*, and *w* is the total number of incorrect predictions on the test set of rules that refer to *A*.

(c) Number of appearances in models: In addition, we have used an interestingness criterion for an attribute in a set of models. The interestingness *I* of an attribute *A*, $I(A) = k/n$, where *k* is the number of models in which the attribute appears, and *n* is the total number of models.

2.3 Dataset: Amyotrophic Lateral Sclerosis

Amyotrophic lateral sclerosis (ALS) is a rapidly progressive and fatal neurodegenerative disease with complex genetic contribution that is difficult to diagnose at the early stages. Samples of cerebrospinal fluid (CSF) were obtained by lumbar puncture from 54 individuals – 23 patients with ALS and 31 negative controls (initial dataset had 34 training samples and 20 blinded samples used as testing). CSF was used since this fluid circulates within the nervous system and is therefore in proximity to cells affected by the disease. Therefore CSF likely harbors a high concentration of potential biomarkers and represents an excellent starting material for proteomic study. The data for each spectrum consisted of 18,396 SELDI-TOF points. Two Ciphergen ProteinChips [6] were used, strong anion exchange (SAX2) and zinc cation-loaded IMAC (Zn-IMAC30). These chips have different binding affinities and capture different subsets of proteins/peptides from the CSF. The variability of replicate experiments was well within 10%; therefore, we only analyze data for each sample from one chip each of SAX-2 and Zn-IMAC30.

We combined the data from the two chips for each sample, since they capture different proteins and peptides. Previous analyses [19] indicated better results from combined data, indicating that a disease-specific biomarker panel would need to be constructed based on evidence gathered from different sources. Each mass spectrum ranged from 2000.1936 kDa to 19964.553 kDa, for a total of 36778 input attributes per sample. The data were normalized by scaling the values for each m/z for all samples, to the interval [0, 1] as in ([20]). That is, the relative amplitude of the intensity at each *m/z* identity in the spectral data was normalized against the most intense and the least intense values in the data stream according to the formula: NV = (V - Min) / (Max - Min), where NV is the normalized value, V the raw value, Min the minimum intensity and Max the maximum intensity.

3 Results

An initial analysis using the wrapper approach on a training set of 34 subjects and 20 blinded test samples resulted in 496 features being identified as interesting attributes of which 264 m/z values were from SAX2 chip and 232 m/z values from the Zn-IMAC30 chip. These features were used as input for the final RL run that resulted in a model containing 10 rules (with total of 10 features). These ten rules were then applied to make predictions on the blinded test set of 20 samples that included four healthy subjects and six neurologic disease controls. Eight of the 10 ALS samples in the test set were correctly identified, as also six of the 10 control subjects. RL made predictions for 19 of the 20, resulting in overall coverage of 95%, with 74% accuracy, 80% sensitivity and 60% specificity. The same training data were analyzed using Ciphergen's Biowizard software that implements a version of CART [21]. This resulted in a decision tree model containing 9 features. Since the software would not permit application of the model to blinded test data, we utilized the version of CART available in Clementine software package (SPSS, Inc.) to learn a model with just the 9 m/z features, and apply it to the 20 test samples. This resulted in complete coverage of the test data with 65% accuracy, 30% sensitivity and 100% specificity. We also gave the 9 m/z features directly to the two-step clustering program in Clementine, and obtained 75% accuracy, 60% sensitivity and 90% specificity based on separation of the test data.

Additionally, the Ciphergen clustering software was also used to perform a univariate analysis of both the SAX2 and Zn-IMAC30 datasets to identify m/z's, whose relative abundances were statistically significantly different between the ALS and Control CSF samples. A non-parameteric Mann Whitney test was used on 366 m/z values that were autodetected by the Biowizard software to have differential signal intensities in ALS and Control spectra. This yielded 15 significant m/z's from each of the two chips ($p < 0.01$). One m/z value (13.38 kDa) that was detected as significant in all of the above analyses has been identified and validated experimentally.

Next, we combined all the samples to get 54, and from these we removed 2 samples that exhibited poor quality spectra that had low signal-to-noise ratios. From the remaining 52 samples, we randomly picked 12 as test examples; and used the remaining 40 as training data. The test examples had 7 ALS and 5 Control. We ran RL-Wrap twelve times on the training data, with different sub-sampling of the feature subsets. The best model had 12 features in the rules. This model yielded 91% accuracy (11 out of 12 test subjects), 80% sensitivity and 100% specificity on test set. One of the m/z values (6.88 kDa) found in this model was hypothesized to be the doubly charged peak of one of the significant m/z's (13.78 kDa) found in the univariate analysis. Two of the m/z values (6.88 and 3.42 kDa) from this set of 12 features have been identified and validated experimentally. One of these m/z's (3.42 kDa) has only been identified by our RL-Wrap method.

In order to evaluate the overall consistency of RL-Wrap analyses, we ran RL-Wrap multiple times with different attribute subsetting criteria. For these analyses, we worked on a subset of the dataset produced by applying a univariate filter (chi-square value equal to zero) to remove attributes that were unlikely to be significant. This yielded a dataset containing 1567 attributes. We also compared RL-Wrap analyses on this dataset to models learned from C4.5 and RIPPER implementations available in the WEKA machine learning environment. Even though overall performance accuracies are comparable when averaged across several runs, the models produced are not. Particularly, because the instance space is not resampled by C4.5 and RIPPER, the models produced often contain only 2 or 3 features. Also, they have a rule that just predicts the default class. Hence, we do not report our comparison with these classification algorithms as we believe that several highly accurate m/z's that classify subsets of the instance space are missing in their models.

We ran RL-Wrap using 50, 100, 400 and 1000 attributes per subset and bias space search. The bias search parameters were Exhaustiveness=Low and 5-fold cross validation, thus each bias search iterated over 21 parameter combinations, creating 105 models, and used an approximate 40-10 train-test split for each iteration. The RL-Wrap algorithm selected attributes by partitioning the attribute set, learning a rule set on each partition, and unioning the attributes that appear in the rule sets. After the attributes were selected, a model was learned using bias search, again with Exhaustiveness=Low and 5 fold cross-validation.

3.1 Sensitivity Analysis

The results from a sensitivity analysis on a model by removing all combinations of attributes (up to 5 attributes) from the model and testing the reduced models on the test set are shown in Table 1. A model from RL-Wrap analysis constitutes a potential panel of biomarkers for disease classification from mass spectra. The results from this analysis seem to indicate that removing one or two biomarkers from this panel may not result in a significant drop in classification accuracy and coverage on the test set. This kind of analysis reveals which biomarkers are crucial predictors on the test set, as the report includes which m/z (s) were dropped along with performance statistics. On other models, we see sharper drops in accuracy upon removing even a single m/z.

Table 1. Sensitivity Analysis Summary. The number of attributes in the reduced model, together with the number of models that have the given number of attributes, mean accuracies of these models, mean coverage of the test data, and mean specificity and sensitivity of the models are reported. A sample of the summary produced is depicted here. Mean differences of accuracy, coverage and other metrics of the reduced model are reported based on the metrics obtained from the original unreduced model.

# Atts in reduced model	# Models	Mean accuracy (acc)	Mean acc Diff	Mean Coverage (cov)	Mean cov Diff	Mean Spec	Mean Spec Diff	Mean Sens	Mean Sens Diff
10	1	1	.	.818	.	.8	.	.833	.
9	10	1	.	.8	-.018	.8	.	.8	-.033
8	45	.997	-.003	.774	-.045	.787	-.013	.759	-.074
7	120	.991	-.009	.738	-.08	.758	-.042	.711	-.122
6	210	.984	-.016	.691	-.127	.713	-.087	.656	-.178
5	252	.976	-.024	.631	-.187	.65	-.15	.592	-.242

3.2 Proportion Correct Use Criterion

The Proportion Correct Use (PCU) interestingness criterion prefers attributes appearing in rules that make many correct (c) and few incorrect (w) predictions on the test data. For example, one such function is $c(a) / (c(a) + w(a) + 1)$. RL-Wrap shows c

Table 2. Example of attribute interestingness by Proportion Correct Use (PCU) criterion. The range of values (in this case normalized) that each attribute takes on in the training data are shown along with the number of correct and incorrect uses in prediction on the test data.

Attribute	Min	Max	Mean	Std Dev	# Correct uses in predictions	# Incorrect uses in predictions
sax13682.916	0.0	1.0	0.27	0.24	**11**	.
zn14537.848	0.0	1.0	0.34	0.26	**1**	**1**
zn9969.4349	0.0	1.0	0.37	0.21	.	.
sax9556.2863	0.0	1.0	0.23	0.16	.	**1**

and *w* for attributes in each learned model. Part of the table it generated for the final RLW100 (using 100 attributes per sampled subset in the first iteration) model is shown in Table 2, sorted by decreasing PCU. The tables also show the minimum, maximum, mean and standard deviation of values in the training data.

3.3 Number of Appearances in Models

RL-Wrap can be run repeatedly to gather statistics regarding the number of times a particular attribute appears in the various final models generated. A sample graph plotting the statistics obtained from running RL-Wrap 12 times is shown in Figure 3. The mean accuracy and coverage of the models were .77 and .88, and the modes were .9 and 1.0. A total of 12 attributes appeared in the 12 models, 7 of which appeared in 10 or more models: SAX12280, SAX13647, SAX2459, SAX2404 and SAX6878 and Zn3010 and Zn8931 (Figure 3).

Next, a sensitivity analysis was performed on each model, by ignoring each attribute in turn, and recording the model's accuracy, coverage, sensitivity and specificity. For each attribute, the mean values of the metrics for the models with the attribute were compared to the mean values for those models when that attribute was ignored. RL-wrap automatically computes these statistics for each attribute (as also for attribute subsets). The two attribute interestingness criteria mostly agree on this data.

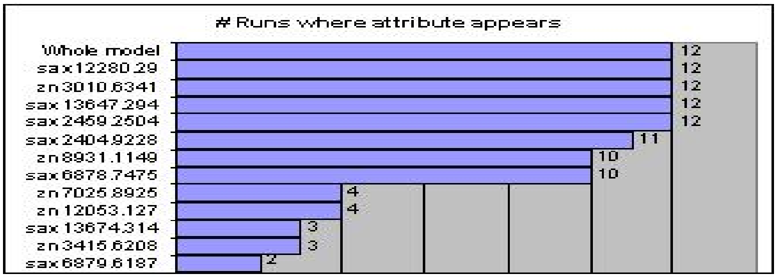


Fig. 3. Plot of number of times an attribute occurred within 12 models learned from repeated RL-Wrap runs

3.4 Evidence Gathering for Site of Disease Onset

The site of disease onset often determines disease prognosis. In our ALS dataset, we have clinical information about the region of disease onset. *Bulbar* onset is rarer and more rapid than *limb* onset. Out of 22 ALS patients, 15 were reported to have site of onset in the limb region, with the remaining 7 in the bulbar region. We trained RL using 5-fold cross validation on this small sample dataset and obtained 95% overall predictive accuracy. Limb onset samples were classified accurately with just 2 m/z's.

4 Biomarker Validation

Three m/z values (3.42, 6.88 and 13.38 kDa) obtained from RL-Wrap analyses were further evaluated and the proteins were identified using experimental procedures described in [11]. The three proteins are believed to be: 7B2CT, a carboxy-terminal fragment of the neuroendocrine protein 7B2 (3.42 kDa peak); cystatin C (13.38 kDa peak), and the 6.88 kDa peak as a monomer of Transthyretin (TTR). We obtained a 100% match of the sequence of the 7B2CT protein sequence with the sequences of the 12 tryptic peptides obtained from our 3.42 kDa peak. Both Cystatin C and TTR are already known to be involved in Alzhiemer's disease. We performed immunoblot and immunohistochemistry on separate cohorts of age-matched ALS and control subjects, using commercially available antibodies to TTR and cystatin C to validate our findings. TTR protein levels as measured by immunoblot predicted ALS with 70% sensitivity and 60% specificity on a set of 20 coded test subjects. We used 14 additional CSF samples for the immunoblot that were from patients whose samples we had already used for mass spectrometry. Detailed experimental verification is described in [11].

5 Conclusions and Future Work

In this paper, we show how biomarkers can be successfully identified using a rule learning methodology that samples the instance space of training examples with replacement. Even though some biomarkers have been successfully identified, the small number of samples from which the rule-based models have been learned for classification of ALS leads us to believe two things: (a) it is indeed remarkable to have experimental identification and confirmation of protein identities, indicating that there are indeed biomarkers within CSF that can help identify diseases such as ALS, and (b) these analyses must be repeated with a lot more data in order to be able to identify disease-specific biomarker panels that are truly worthy of application in clinical proteomics. Nevertheless, there seems to be real value in utilizing rule-based methods that provide easier means for evidence gathering from multitudes of features present within clinical proteomic mass spectra.

Acknowledgements

We are grateful to Dr. Bruce Buchanan for providing us with the JAVA version of Rule Learner, and Dr. Merit Cudkowicz at Massachussets General Hospital, Boston, MA for CSF samples. Funding support was provided by NIH/NIGMS GM 071951 (VG, PG), NIH/NIEHS ES013469 (RB) and the ALS foundation.

References

1. Srinivas PR, Verma M, Zhao Y, Srivastava S: Proteomics for cancer biomarker discovery. Clin Chem 2002, 48(8):1160-1169.
2. Tyers M, Mann M: From genomics to proteomics. Nature 2003, 422(6928):193-197.
3. Cazares LH, Adam BL, Ward MD, Nasim S, Schellhammer PF, Semmes OJ, Wright GL, Jr.: Normal, benign, preneoplastic, and malignant prostate cells have distinct protein expression profiles resolved by surface enhanced laser desorption/ionization mass spectrometry. Clin Cancer Res 2002, 8(8):2541-2552.
4. Wright GL, Cazares LH, Leung SM, Nasim S, Adam BL, Yip TT, Schellhammer PF, Gong L, Vlahou A: Proteinchip(R) surface enhanced laser desorption/ionization (SELDI) mass spectrometry: a novel protein biochip technology for detection of prostate cancer biomarkers in complex protein mixtures. Prostate Cancer Prostatic Dis 1999, 2(5/6): 264-276.
5. Adam BL, Qu Y, Davis JW, Ward MD, Clements MA, Cazares LH, Semmes OJ, Schellhammer PF, Yasui Y, Feng Z, Wright GL Jr.: Serum protein fingerprinting coupled with a pattern-matching algorithm distinguishes prostate cancer from benign prostate hyperplasia and healthy men. Cancer Res 2002, 62(13):3609-3614.
6. Petricoin EF, Ardekani AM, Hitt BA, Levine PJ, Fusaro VA, Steinberg SM, Mills GB, Simone C, Fishman DA, Kohn EC, Liotta LA: Use of proteomic patterns in serum to identify ovarian cancer. Lancet 2002, 359(9306):572-577.
7. Coombes KR, Morris JS, Hu J, Edmonson SR, Baggerly KA: Serum proteomics profiling--a young technology begins to mature. Nat Biotechnol 2005, 23(3):291-292.
8. Bensmail H, Golek J, Moody MM, Semmes JO, Haoudi A: A novel approach for clustering proteomics data using Bayesian fast Fourier transform. Bioinformatics 2005, 21(10):2210-2224.
9. Fung ET, Weinberger SR, Gavin E, Zhang F: Bioinformatics approaches in clinical proteomics. Expert Rev Proteomics 2005, 2(6):847-862.
10. Seibert V, Ebert MP, Buschmann T: Advances in clinical cancer proteomics: SELDI-ToF-mass spectrometry and biomarker discovery. Brief Funct Genomic Prot 2005, 4(1):16-26.
11. Ranganathan S, Williams E, Ganchev P, Gopalakrishnan V, Lacomis D, Urbinelli L, Newhall K, Cudkowicz ME, Brown RH, Jr., Bowser R: Proteomic profiling of cerebrospinal fluid identifies biomarkers for amyotrophic lateral sclerosis. J Neurochem 2005, 95(5):1461-1471.
12. Frank E, Hall M, Trigg L, Holmes G, Witten IH: Data mining in bioinformatics using Weka. Bioinformatics 2004, 20(15):2479-2481.
13. Witten IH, Frank E: Data Mining: Practical machine learning tools and techniques, Second edn. San Francisco: Morgan Kaufmann; 2005.
14. Clearwater S, Provost F: RL4: A Tool for Knowledge-Based Induction. In: Proceedings of the Second International IEEE Conference on Tools for Artificial Intelligence (TAI-90): 1990.
15. Feigenbaum EA, Buchanan BG: Dendral and Meta-Dendral - Roots of Knowledge Systems and Expert System Applications. Artif Intell 1993, 59(1-2):223-240.
16. Provost F, Fawcett, T: Robust classification for imprecise environments. Machine Learning 2001, 42:203-231.

17. Mitchell T: The need for biases in learning generalizations. In: Readings in Machine Learning. Edited by Dietterich TG, Shavlik J: Morgan Kaufmann; 1991.
18. Provost F, Buchanan BG: Inductive policy: the pragmatics of bias selection. Machine Learning 1995, 20:35-61.
19. Gopalakrishnan V, Williams E, Ranganathan S, Bowser R, Cudkowic ME, Novelli M, Lattanzi W, Ganbotto A, Day BW: Proteomic Data Mining Challenges in Identification of Disease-Specific Biomarkers from Variable Resolution Mass Spectra. In Proceedings of SIAM Bioinformatics Workshop 2004, Society of Industrial and Applied Mathematics International Conference on Data Mining, April 2004, 1-10.
20. Liu H, Li J, Wong L: A Comparative Study on Feature Selection and Classification methods Using Gene Expression Profiles and Proteomic Patterns. Genome Informatics 2002, 13:51-60.
21. Breiman L, Friedman JH, Olshen RA, Stone CJ: Classification and Regression Trees. Belmont, CA: Wadsworth International Group; 1984.

Machine Learning Techniques and Chi-Square Feature Selection for Cancer Classification Using SAGE Gene Expression Profiles

Xin Jin[1], Anbang Xu[1,2], Rongfang Bie[1,*], Ping Guo[1,2]

[1] College of Information Science and Technology,Beijing Normal University, Beijing 100875, P.R. China
xinjin796@126.com, rfbie@bnu.edu.cn

[2] Image Processing & Pattern Recognition Laboratory, Beijing Normal University
anbangxu@mail.bnu.edu.cn, pguo@bnu.edu.cn

Abstract. Recently developed Serial Analysis of Gene Expression (SAGE) technology enables us to simultaneously quantify the expression levels of tens of thousands of genes in a population of cells. SAGE is better than Microarray in that SAGE can monitor both known and unknown genes while Microarray can only measure known genes. SAGE gene expression profiling based cancer classification is a better choice since cancers may be due to some unknown genes. Whereas a wide range of methods has been applied to traditional Microarray based cancer classification, relatively few studies have been done on SAGE based cancer classification. In our study we evaluate popular machine learning methods (SVM, Naive Bayes, Nearest Neighbor, C4.5 and RIPPER) for classifying cancers based on SAGE data. In order to deal with the high dimensional problem, we propose to use Chi-square for tag/gene selection. Both binary classification and multicategory classification are investigated. The experiments are based on two human SAGE datasets: brain and breast. The results show that SVM and Naive Bayes are the top-performing SAGE classifiers and that Chi-square based gene selection can improve the performance of all the five classifiers investigated.

1 Introduction

Serial Analysis of Gene Expression (SAGE) is a relatively new method of measurement for gene expression data [3] and has been used in studies of a wide range of biological systems [4, 5]. It identifies a short mRNA tag from each individual transcript and concatenates them into long DNA molecules, which are then sequenced. By counting these tags one can estimate, for example, the expression of genes in a cell. The information gained from performing this technique on a tissue sample is called a SAGE library. SAGE method is better than *microarray* technique,

[*] Corresponding author.

another method for measuring gene expression, in that both known and unknown the mRNAs in a tissue sample has a chance of being tagged and sequenced by SAGE while *microarray* can only measure the sequences of known mRNAs. This advantage is important, especially for cancer profiling on the gene expression level, because the sequence and purpose of most mRNAs has not been discovered yet.

The large amount of SAGE data would greatly help doctors in their endeavor to better understand what goes on inside the human body. Since a human investigation of the data does not extract much meaningful information, an automated approach is needed. Automated classification of cancer data is crucial in making diagnoses quick, more reliable, and less prone to human error. However, the main challenges in the SAGE data classification task are the availability of a smaller number of libraries/samples compared to huge number of tags/genes, many of which are irrelevant for classification, and the noisy nature of SAGE data. Therefore, any good classifier needs to first remove errors, reduce the number of tags and filter out genes that are not strongly correlated with the cancer classes as indicated by the samples in the provided training data. The process of reducing the number of tags is called feature selection.

Several SAGE data analysis methods have been developed, primarily for extracting SAGE tags and identifying differences in mRNA levels between two libraries. Cai Li et al. cluster the SAGE tags using a Poisson distribution approach [1]. Raymond T. Ng et al. cluster the SAGE libraries using the OPTICS algorithm [2]. As for SAGE data classification, J. Sander et al. adopt a Nearest Neighbors (NN) approach for *binary* (two-class) classification [10]. They randomly selected part of cancerous and normal libraries of a specific tissue as the training libraries, and then randomly picked a testing library from the remaining ones. They try to predict whether the testing library is normal or cancerous based on the NN classifier. In this article, we extend the focus to classifying cancer types, both binary and multicategory. We evaluate five different kinds of popular machine learning algorithms for classification of SAGE data and propose to use Chi-square for SAGE tag selection before building classification model.

This paper proceeds as follows. Section 2 outlines the five learners we investigated. Section 3 presents the gene selection method. Section 4 reports on performance evaluation of those classifiers using micro averaged accuracy and F_1 measure. Section 5 concludes the findings.

2 Machine Learning Algorithms

In this section, we will give a brief description to each of the five well-known machine learning mechanisms investigated in this work. We begin with a kernel based learner: Support Vector Machine [9], a learning paradigm that is based on Structured Risk Minimization principle. Then we proceed with a probabilistic leaner: Naive Bayes [22]. It is probabilistic in the sense that it is able to estimate the probability of each class being predicted. After that, we introduce Nearest Neighbor: an instance based learning approach that labels a new sample according to its similarity between stored instance base. Then we proceed with a tree based learner: C4.5 [19]. Finally, we discuss a rule based learner: RIPPER [11].

2.1 Kernel Based Learning: Support Vector Machine

The Support Vector Machine (SVM) is a powerful classifier, originally proposed by Vapnik, which finds a maximal margin separating hyperplane between two classes of data [9]. A SVM selects a small number of critical boundary samples from each class and builds a linear discriminant function (also called maximum margin hyperplane) that separates them as widely as possible. In the case that linear separation is impossible, the technique of kernel will be used to automatically inject the training samples into a higher-dimensional space, and to learn a separator in that space. A maximum margin hyperplane *H(T)* for a test sample *T* is a linear combination of kernels computed at the training data points and is constructed as

$$H(T) = sign(\sum_i [\alpha]_i * [C]_i * k(T, [X]_i) + b) \tag{1}$$

where $[X]_i$ are the training data points, $[C]_i$ are the class labels of these data points, $k()$ is the kernel function, b and $[\alpha]_i$ are parameters that determine the hyperplane and can be learned from the training data.

There are several ways for training a SVM. One of the fastest algorithms is developed by Platt, which solves the above quadratic programming problem by sequential minimal optimization (SMO). In our experiments, we use SMO with the polynomial kernel function and the transformation of the output of SVM into probabilities is conducted by a standard sigmoid function.

2.2 Probability Based Learning: Naive Bayes

Naive Bayes, a simple Bayesian classification algorithm, has been gaining popularity lately, and has been found to perform surprisingly well in text category and email filtering [13].

Consider the task of SAGE data classification in a Bayesian learning framework. A parametric model is assumed to have generated the data, and Bayes-optimal estimates of the model parameters are calculated using the training data. We classifies new test library using Bayes rule to turn the generative model around and calculate the posterior probability that a class would have generated the test library. Then, classification becomes a simple matter of selecting the most probable class.

We assume that SAGE libraries are generated by a mixture model parameterized by θ. The mixture model consists of mixture components $C = \{c_1 \ldots c_M$, M is the number of classes$\}$ that correspond to the classes. Each component $c_i \in C$ is parameterized by a disjoint subset of θ. A library l_j is generated by first selecting a mixture component c_i according to the prior distribution $P(c_i|\theta)$ and then having the component generate a library according to its own parameters, with distribution $P(l_j|c_i;\theta)$. The likelihood of a library is given by a sum of probability over all mixture components:

$$P(l_j \mid \theta) = \sum_i P(c_i \mid \theta) P(l_j \mid c_i; \theta) \quad (2)$$

where i= 1, 2,···, M. Each SAGE library has been manually annotated with their correct class. Since the true parameters θ of the mixture model are not known, we need to estimate the parameters from labeled training libraries. If $\theta^{'}$ denotes the estimated parameters, given a set of training libraries L={l_1,..., l_N, N is the number of training samples}, we use maximum likelihood to estimate the class prior parameters as the fraction of training libraries is c_i:

$$\theta^{'}_{c_i} = P(c_i \mid \theta^{'}) = \frac{\sum_{j=1}^{N} P(c_i \mid l_j)}{N} \quad (3)$$

where $P(c_i|l_j)$ is 1 if l_j correspond to class c_i and 0 otherwise.

In general, the SAGE data classification problem can be described as follows. Taking into account that one library only belongs to one class (type of cancer), for a given library l we search for a class c_i that maximizes the posterior probability $P(c_i| l;\theta^{'})$, by applying Bayes rule:

$$P(c_i \mid l; \theta^{'}) = \frac{P(c_i \mid \theta^{'}) P(l \mid c_i; \theta^{'})}{P(l \mid \theta^{'})} \quad (4)$$

Note that $P(l|\theta^{'})$ is the same for all classes, thus l can be classified by computing.

$$c_l = \arg\max_{c_i \in C} P(c_i \mid \theta^{'}) P(l \mid c_i; \theta^{'}) \quad (5)$$

See [16] for more information on estimating continuous distributions in Bayesian classifiers.

2.3 Instance Based Learning: Nearest Neighbor

Instance based learning (also called memory-based Learning) is a non-parametric inductive learning paradigm that stores training instances in a memory structure on which predictions of new instances are based. The approach assumes that reasoning is based on direct reuse of stored experiences rather than on the application of knowledge (such as models or decision trees) abstracted from experience. The similarity between the new instance and an example in memory is computed using a distance metric. In our experiment, we used IB1 [20], a Nearest Neighbor (NN) classifier that uses Euclidian distance metric [14].

The main idea of NN for SAGE data is that it treats all libraries as points in the m-dimensional space (where m is the number of tags/genes in the library set) and given an unseen library l, the algorithm classifies it by the nearest training library.

2.4 Decision Tree Based Learning: C4.5

Decision tree (DT) is one of the most popular inductive learning algorithms. The nodes of the tree correspond to attribute (in our case gene) test, the links (to attribute values and the leaves) to the classes. To induce a DT, the most important attribute (according to an attribute selection criteria, such as information gain, GainRatio, etc.) is selected and placed at the root; one branch is made for each possible attribute value. This divides the examples into subsets, one for each possible attribute value. The process is repeated recursively for each subset until all instances at a node have the same classification, in which case a leaf is created. To classify an example we start at the root of the tree and follow the path corresponding to the example's values until a leaf node is reached and the classification is obtained. To prevent overtraining DT is typically pruned. The most popular DT learning algorithm is Quinlan's C4.5 [19]. We have used the Weka [18] implementation of the C4.5 algorithm in our experiments.

2.5 Rules Based Learning: RIPPER

RIPPER [11], a well-known rule based learning algorithm, builds a ruleset by repeatedly adding rules to an empty ruleset until all positive examples are covered. Rules are formed by greedily adding conditions to the antecedent of a rule (starting with an empty antecedent) until no negative examples are covered. After a ruleset is constructed, an optimization postpass massages the ruleset so as to reduce its size and improve its fit to the training data. A combination of cross-validation and minimum-description length techniques are used to prevent overfitting.

3 Gene Selection

It is common to use gene selection for gene expression data classification, in order to reduce over-fitting to the training data and to speed up the classification process [6]. Following [7] we list four subset selection methods/metrics: (1) ratio of features Between-categories to Within-category sums of squares (BW); (2-3) Signal-to-Noise (S2N) scores [8] applied in a One-Versus-Rest (S2N-OVR) and One-Versus-One (S2N-OVO) fashion; and (4) Kruskal-Wallis nonparametric one-way ANOVA (KW).

For SAGE data clustering, J. Sander et al. used the Wilcoxon rank sum test, which can tests whether two libraries are taken from the same population, to exclude SAGE tags/genes that have similar expression levels [10]. We propose to select useful genes/features by Chi-square for SAGE data classification problem. The Chi-square, which is a popular feature selection method, will evaluate genes individually with respect to the classes. The range of continuous valued features needs to be discretized into intervals. Chi-squared is based on comparing the obtained values of the frequency of a class because of the split to the expected frequency of the class. Of the *N* examples, let *Nij* be the number of samples of the *Ci* class within the *jth* interval and M_{Ij} is the number of samples in the *j*th interval. The expected frequency of *Nij* is $Eij = M_{Ij} \, |Ci|/N$. The Chi-squared statistic of a gene is then defined as:

$$\chi^2 = \sum_{i=1}^{C} \sum_{j=1}^{I} \frac{(N_{ij} - E_{ij})^2}{E_{ij}} \tag{6}$$

where I is the number of intervals. The larger the χ^2 value, the more informative the corresponding gene is.

4 Experiments

In this section, we present the experiments performed on five machine learning algorithms for SAGE data based cancer classification. First, we will give the dataset descriptions and the SAGE data preprocessing method. Next, we present the empirical results of the experiments on two SAGE datasets (*brain* and *breast*).

4.1 Data and Preprocessing

The experiments are based on two SAGE data sets, the raw data is available on the NCBI SAGE website (http://www.ncbi.nlm.nih.gov/SAGE). One problem with the raw SAGE data is that many tags in each library are expected to contain sequencing errors, and since these errors result in noise and increase the dimensionality of the data, error removal is a must need. Within one library, some tags have a frequency of 1; these unique tags are either sequencing errors or representations of very low expression level genes. In our experiment, we just remove these single frequency tags to filter out data noise.

Brain Dataset: The dataset is based on 52 Hs (human sapiens) SAGE brain libraries. These libraries are made of samples from human brain and fall in to four categories: Astrocytoma (11 libraries), Ependymoma (9 libraries), Glioblastoma (8 libraries) and Medulloblastoma (24 libraries). There are 64558 genes (after noise removal) in the dataset. We used the dataset for multicategory classification experiments.

Breast Dataset: The dataset is based on 26 Hs (human sapiens) SAGE breast libraries. These libraries are made of samples from human breast and fall in to two classes: Normal (10 libraries) and Cancer (16 libraries). There are 36087 genes (after noise removal) in the dataset. We used this dataset for binary classification experiments.

4.2 Performance Measures

Our experiments adopt the most commonly used performance measures, including the accuracy and F_1 measures.

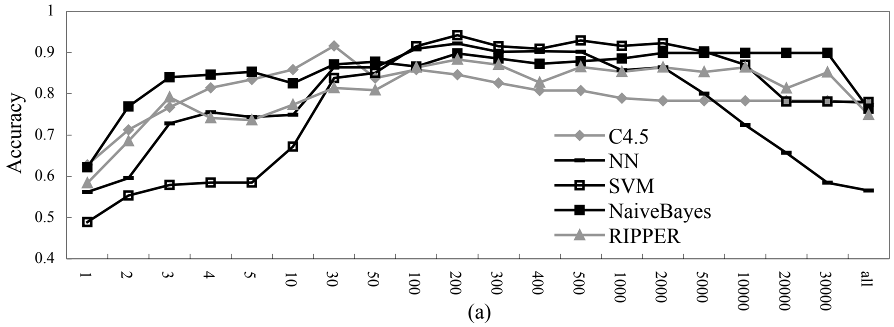
Accuracy is defined by the ratio of the number of correct predictions and the number of all predictions (both correct and incorrect): $Acc = N_{cp}N_p$, where N_{cp} is the number of correct predictions and N_p is the number of all predictions (i.e. the number of test samples).

F_1 rating is defined as $F_1=2R*P/(R+P)$. Recall (R) is the percentage of the libraries for a given category that are classified correctly. Precision (P) is the percentage of the predicted libraries for a given category that are classified correctly. It is a normal practice to combine recall and precision to F_1 measure so that classifiers can be compared in terms of a single rating.

4.3 Results

K-fold cross-validation is used for estimating classifier performance. We choose k=5 for all our experiments, we do not use k=10 because there are two classes Ependymoma and Glioblastoma which have less than 10 samples. We performed thirty runs for each 5-fold cross-validation and averaged the results. The number of selected genes varies from 1 to all.

Multicategory classification: Fig. 1 shows the results of multicategory classification on the human *brain* libraries. Without gene selection, Naive Bayes, SVM, C4.5 and RIPPER have similar performance, RIPPER are the worst. The performances of five classifiers are improved by Chi-square based gene selection. SVM reaches a maximum of 94% accuracy at 200 genes. Naive Bayes reaches a maximum of 0.91 F_1 measure at 200 genes. The figure shows that in order to successfully classify



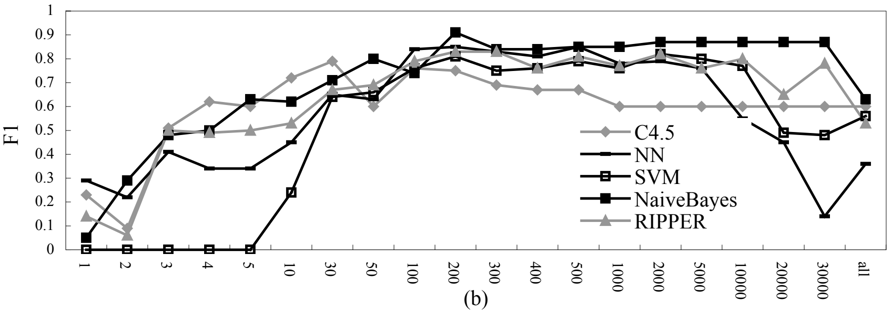


Fig. 1. Micro average accuracy (a) and F_1 measure (b) curves of five classifiers for different feature sizes on the *brain* SAGE libraries: multicategory classification. The X axis denotes the number of selected genes according to Chi-square ranking.

multicategory cancers we just need to select several hundred genes accompanied with SVM or Naïve Bayes classifiers.

Binary classification: Fig. 2 shows the results of binary classification on the human *breast* libraries. Without gene selection, SVM and Naive Bayes are the top two classifiers, RIPPER is the worst. Gene selection can improve the performance of all five classifers. Naive Bayes reaches a maximum of 98% accuracy at 500, 1000 and 2000 genes and a maximum of 0.96 F_1 measure at 500 and 1000 genes. With only a few genes, C4.5, NN, and RIPPER can still achieve high accuracy and F_1 measure. This show that only few Chi-square selected genes can distinguish between cancer and normal samples.

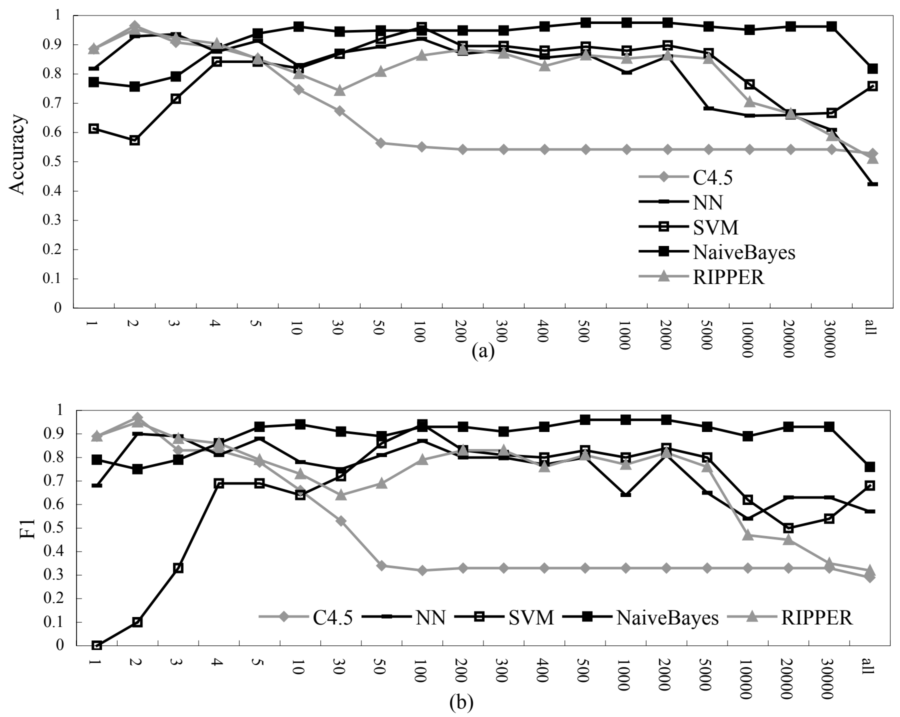


Fig. 2. Micro average accuracy (a) and F_1 measure (b) curves of five classifiers for different feature sizes on the *breast* SAGE libraries: binary classification (between normal and cancerous tissues). The X-axis denotes the number of selected genes according to Chi-square ranking.

5 Conclusions

In this study we evaluate popular machine learning methods (SVM, Naive Bayes, Nearest Neighbor, C4.5 and RIPPER) for classifying cancers based on SAGE gene expression profiles. From the analysis of the experimental human *brain* and *breast* SAGE data, we found that SVM and Naïve Bayes can achieve better performance than the other three classifiers with suitable number of selected genes. We also found that with Chi-square, several hundred tags/genes are sufficient for successful multicategory cancer classification and only a few genes are sufficient to distinguish between cancer and normal samples.

Acknowledgment

The research work described in this paper was fully supported by grants from the National Natural Science Foundation of China (Project No. 60273015, No. 10001006 and No. 60275002).

References

1. Cai Li, Huang Haiyan, Blackshaw Seth, Liu Jun, Cepko Connie, Wong Wing: Clustering Analysis of SAGE Data Using a Poisson Approach. Genome Biology, 5:R51 (2004)
2. Raymond T. Ng, Jörg Sander, Monica C. Sleumer: Hierarchical Cluster Analysis of SAGE Data for Cancer Profiling. BIOKDD, 65-72 (2001)
3. Velculescu V. E., Zhang L., Vogelstein B., Kinzler K.W.: Serial Analysis of Gene Expression. Science, Vol. 270, Oct 20 484-487 (1995)
4. Buckhaults P., et al.: Identifying Tumor Origin Using a Gene Expression-based Classification Map. Cancer Res, (63):4144-4149 (2003)
5. Porter D., et al.: A Neural Survival Factor is a Candidate Oncogene in Breast Cancer. Proc Natl Acad Sci USA (100):10931-10936 (2003)
6. I. Guyon, J. Weston, S. Barnhill, V. Vapnik: Gene Selection for Cancer Classification Using Support Vector Machines. Machine Learning, 46(1/3):389–422 (2002)
7. Alexander Statnikov, et al.: A Comprehensive Evaluation of Multicategory Classification Methods for Microarray Gene Expression Cancer Diagnosis. Bioinformatics Advance Access, September 16 (2004)
8. Golub T., et al.: Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring. Science, Vol. 286, October 15 (1999)
9. Corinna Cortes and Vladimir Vapnik: Support-vector Networks. Machine Learning, 20(3):273-297 (1995)
10. J. Sander, R.T. Ng, M.C. Sleumer, M. Saint Yuen, and S.J. Jones: A Methodology for Analyzing SAGE Libraries for Cancer Profiling. ACM Transactions on Information Systems, 23(1):35-60 (2005)
11. William W. Cohen: Fast Effective Rule Induction. Machine Learning. In Proceedings of the Twelfth International Conference (ML95) (1995)
12. SAGEMap: http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GPL4 (2005)
13. Karl-Michael Schneider: A Comparison of Event Models for Naive Bayes Anti-Spam E-Mail Filtering. In Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics, Budapest, Hungary, 307-314, April (2003)
14. A. Statnikov, C. F. Aliferis, I. Tsamardinos, D. P. Hardin, S. Levy: A Comprehensive Evaluation of Multicategory Classification Methods for Microarray Gene Expression Cancer Diagnosis. Bioinformatics (2004)

15. I. Androutsopoulos et al.: Learning to Filter Spam E-mail: A Comparison of a Naive Bayesian and a Memory-based Approach. In Proceedings of the Workshop on Machine Learning and Textual Information Access, 1-13 (2000)
16. George H. John and Pat Langley: Estimating Continuous Distributions in Bayesian Classifiers. In Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence. Page 338-345. Morgan Kaufmann, San Mateo (1995)
17. Leo Breiman: Bagging Predictors. Machine Learning, 24(2):123-140. (1996)
18. I.Witten and E.Frank: Data Mining –Practical Machine Learning Tools and Techniques with Java Implementation. Morgan Kaufmann (2000)
19. Ross Quinlan: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, San Mateo, CA. (1993)
20. Aha, D., and D. Kibler: Instance-based Learning Algorithms, Machine Learning, Vol.6, 37-66 (1991)
21. S.S. Keerthi, S.K. Shevade, C. Bhattacharyya, K.R.K. Murthy: Improvements to Platt's SMO Algorithm for SVM Classifier Design. Neural Computation, 13(3):637-649 (2001)
22. Le Zhang, Jingbo Zhu, Yao Tianshun: An Evaluation of Statistical Spam Filtering Techniques. ACM Trans. Asian Lang. Inf. Process. 3(4):243-269 (2004)

Generation of Comprehensible Hypotheses from Gene Expression Data

Yuan Jiang, Ming Li, and Zhi-Hua Zhou

National Laboratory for Novel Software Technology,
Nanjing University, Nanjing 210093, China
{jiangy, lim, zhouzh}@lamda.nju.edu.cn

Abstract. Machine learning techniques have been recognized as powerful tools for the analysis of gene expression data. However, most learning techniques used in class prediction in gene expression analysis during the past years generate black-box models. Although the prediction accuracy of these models could be very well, they provide little insight into the biological facts. This paper holds the recognition that a more reasonable role for machine learning techniques is to generate hypotheses that can be verified or refined by human experts instead of making decisions for human experts. Based on this recognition, a general approach to generate comprehensible hypotheses from gene expression data is described and applied to human acute leukemias as a test case. The results demonstrate the feasibility of using machine learning techniques to help form hypotheses on the relationship between genes and certain diseases.

1 Introduction

DNA arrays consist of a large number of DNA molecules spotted in a systemic order on a solid substrate. When the diameter of the DNA spot is less than $250\mu m$, DNA arrays can be categorized as microarrays [14][19]. With the development of microarray technology, the simultaneous measurement of gene-expression levels for thousands of genes is now possible. Analyzing gene expression data could be helpful for medical treatment. For example, systematic and unbiased approaches could be developed for cancer classification through analyzing gene expression data, which is very important for cancer treatment [8]. However, as keeping track of thousands of measurements and their relationships is overwhelmingly complicated, gene expression data is difficult to analyze without the help of computers.

During the past years, machine learning techniques have been recognized as powerful tools for gene expression analysis [16]. Machine learning [15] is the study of computer algorithms capable of learning to improve their performance of a task on the basis of their own previous experience. It is closely related to pattern recognition and statistical inference, and has data mining as its engineering application aspect. Machine learning techniques such as clustering, neural networks, hidden Markov models, and nonlinear regression have already been widely used in the practice of engineering, business, and science.

In analyzing gene expression data, most work had primarily been descriptive rather than analytical and had focused on cell culture rather than primary patient material, in which genetic noise might obscure an underlying reproducible expression pattern. Since Golub et al.'s work [8], learning techniques such as neural networks [1][8], support vector machines [2][7], ensemble learning methods [1][2][4][5][22], nearest neighbor classifiers [2], logistic regression [13], linear discriminant analysis [5][17], emerging patterns [12], etc. have been applied to gene expression data, where the goal is to classify cases into diagnostic or prognostic categories. However, almost all the learning techniques used in gene expression analysis during the past years generate black-box models. Although the prediction accuracy of these models could be very well, they provide little insight into the biological facts and can hardly provide explicit explanations for their predictions.

Imagine the scenario where a patient asked the doctor why he made a specific diagnosis, the doctor said he could not explain because he did not know either. Of course such a diagnosis is unacceptable. This somewhat reflects the problem of prediction with black-box models. In fact, it is not reasonable to anticipate computers replace experienced human medical experts. Therefore a more reasonable role for machine learning techniques is to generate hypotheses that could be verified or refined by human experts, which might be the basis for further understanding of the relationship between specific genes and diseases. It is evident that black-box models are helpless to this purpose.

In fact, there are many works devoted to the improving of the comprehensibility of black-box models [24], some of which has already been applied to medical diagnosis [10][21]. Unfortunately, as mentioned before, in the analysis of gene expression data, almost all the methods used before generate black-box models.

In this paper, a general approach to generate comprehensible hypotheses from gene expression data is described, which is based on a recent achievement of machine learning, i.e. the C4.5Rule-PANE method [25]. This paper applies this approach to human acute leukemias as a test case. The results demonstrate the feasibility of using machine learning techniques to help disclose the relationship between genes and certain diseases.

The rest of this paper is organized as follows. Section 2 briefly introduces the C4.5Rule-PANE method. Section 3 describes the case study on generating comprehensible hypotheses for human acute leukemias from gene expression data. Section 4 concludes.

2 The Method

Although the main purpose of this paper is to report on the application of the C4.5Rule-PANE method to human acute leukemias, for the self-containness of this paper, here a brief introduction on the method is given. Interested readers can refer [25] for more details.

Suppose there is a gene expression data set $S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_l, y_l)\}$, where $\mathbf{x}_i$ is a feature vector coding the gene measurements on a case, y_i is the known outcome of the case corresponding to $\mathbf{x}_i$, l is the number of cases with known outcomes.

At first, bootstrap sampling [6] is employed to produce N data sets with the same size as $|S|$, i.e. the number of cases in S. A neural network classifier is then trained from each of these new data sets. Therefore a neural network ensemble [27] is obtained.

After that, the neural network ensemble is used to judge the cases in S. In detail, all the feature vectors $\mathbf{x}_i$ $(i = 1, 2, \cdots, l)$ are fed to the neural network classifiers. For $\mathbf{x}_i$, N predictions will be provided, each by one neural network classifier. Then, the majority of these N predictions is regarded as the outcome of the neural network ensemble on $\mathbf{x}_i$, which is denoted as y'_i. Therefore, after processing all the cases in S, a new data set $S' = \{(\mathbf{x}_1, y'_1), (\mathbf{x}_2, y'_2), \cdots, (\mathbf{x}_l, y'_l)\}$ is generated.

Moreover, a set of random vectors $\mathbf{x}^*_j$ $(j = 1, 2, \cdots, m)$ can be generated, where the k-th element of $\mathbf{x}^*_j$ is a value randomly chosen from the values that could appear on the k-th gene measurement. These vectors are fed to the neural network ensemble. Let the outcome of the neural network ensemble on $\mathbf{x}^*_j$ be denoted as y^*_j. Then a data collection $S^* = \{(\mathbf{x}^*_1, y^*_1), (\mathbf{x}^*_2, y^*_2), \cdots, (\mathbf{x}^*_m, y^*_m)\}$ is generated. Through combining S' and S^*, a new data set S^{**} is obtained. The size of S^{**} can be controlled by the parameter $\mu = m/l$. Note that S^{**} could be far bigger than the original data set S because m could be far bigger than l.

Then, a C4.5 decision tree [20] is trained from S^{**} and every path from the root to a leaf is converted to an initial propositional rule by regarding all the test conditions appearing in the path as the conjunctive rule antecedents while regarding the classification held by the leaf as the rule consequence. All the initial rules are generalized by removing antecedents that do not seem helpful for distinguishing a specific class from other classes. Rules that do not contribute to the accuracy of the rule set are also removed. A default rule is created for dealing with cases that have not been covered by any of the generated rules, which has no antecedent and the consequence is the biggest class among these cases. The resulting rules could finally be organized into an 'IF-THEN-ELSE' format with embedding 'IF-THEN-ELSE' structures, which corresponds to a hypothesis generated from the gene expression data.

The details of C4.5Rule-PANE can be found in [25], whose computational cost is very close to that of the neural network ensemble. It has been proven [26] that training a neural network ensemble and then using it to predict the training data, which is then given to another learning approach, could be beneficial. The premise is that the original training data set contains much noise and has not fully captured the target distribution, and the neural network ensemble is more accurate than the model directly trained from the original training data set by the second learning approach. It is evident that the first condition is very easy to meet because in rare applications the training data set does not contain much noise and does fully capture the target distribution.

Note that the C4.5Rule-PANE algorithm is not an algorithm which simply extracts faithful rules from neural network ensembles [24]. Instead, some mistakes made by the neural network ensemble could be corrected during the learning of the rules. Thus, the generalization ability of C4.5Rule-PANE can be even higher than that of the neural network ensemble [25][26].

It is also worth noting that in analyzing gene expression data, an often encountered problem is that there are only a small number of cases with known outcomes. Thus, models built by machine learning techniques are not very reliable since the training set is too small. C4.5Rule-PANE has two keys to relax this limitation. The first is the generation of S^* with the help of neural network ensemble, which could greatly enlarge the training set for the rule generation process. The second is the comprehensible rules it generated, which could be verified by human experts instead of could only be used alone.

In fact, different rules can be produced if C4.5Rule-PANE is run for several times. These rules can be validated if there are cases with known outcomes that have not been used in training, and the rule with the highest validating accuracy can be regarded as the final hypothesis which might be helpful in disclosing the relationship between certain genes and diseases.

3 Case Study

Chemotherapy regimens for acute lymphoblastic leukemia (ALL) generally contain corticosteroids, vincristine, methotrexate, and L-asparaginase, whereas most acute myeloid leukemia (AML) regimens rely on a backbone of daunorubicin and cytarabine [3][18]. Although remissions can be achieved using ALL therapy for

Table 1. Measurements appear in the hypotheses

ID#	Gene accession number	Gene description
5	AFFX-BioC-3_at	AFFX-BioC-3_at (endogenous control)
8	AFFX-CreX-5_at	AFFX-CreX-5_at (endogenous control)
13	AFFX-BioC-5_st	AFFX-BioC-5_st (endogenous control)
22	AFFX-DapX-3_at	AFFX-DapX-3_at (endogenous control)
461	D49950_at	Liver mRNA for interferon-gamma inducing factor (IGIF)
1834	M23197_at	CD33 CD33 antigen (differentiation antigen)
1882	M27891_at	CST3 Cystatin C (amyloid angiopathy and cerebral hemorrhage)
2020	M55150_at	FAH Fumarylacetoacetate
2242	M80254_at	PEPTIDYL-PROLYL CIS-TRANS ISOMERASE, MITOCHONDRIAL PRECURSOR
2402	M96326_rnal_at	Azurocidin gene
2759	U12471_cds1_at	Thrombospondin-p50 gene extracted from Human thrombospondin-1 gene, partial cds
3258	U46751_at	Phosphotyrosine independent ligand p62 for the Lck SH2 domain mRNA
3320	U50136_rnal_at	Leukotriene C4 synthase (LTC4S) gene
4847	X95735_at	Zyxin
5039	Y12670_at	LEPR Leptin receptor
6201	Y00787_s_at	INTERLEUKIN-8 PRECURSOR

AML and vice versa, cure rates are markedly diminished, and unwarranted toxicities are encountered. So, distinguishing ALL from AML is critical for successful treatment.

The data were taken from [8]. The training data set consists of 11 AML cases and 27 ALL cases, while the test data set consists of 14 AML cases and 20 ALL cases. Each case is composed of 7,129 gene expressions.

Since the number of gene measurements is very large compared to the number of cases, feature selection in the context of gene expression analysis has been investigated and found beneficial [1][4][9][11]. In a previous work where seven different feature selection methods were applied to the concerned data, it was

Table 2. Hypotheses generated on the human acute leukemias data set

Hypothesis 1:	
IF (M27891_at $>$ 820) AND (U12471_cds1_at $>$ 145) THEN class = AML	(11/11)
ELSE IF (X95735_at $\leq$ 1380.257218) THEN class = ALL	(18/18)
ELSE IF (U50136_rna1_at $\leq$ 1464.485389) THEN class = ALL	(2/3)
ELSE class = AML	(2/2)
Hypothesis 2:	
IF (Y00787_s_at $\leq$ 523) AND (M23197_at $\leq$ 472.769728) THEN class = ALL	(16/17)
ELSE IF (D49950_at $>$ 58.110118) AND (M23197_at $>$ 191.728789) AND (U12471_cds1_at $>$ 56.992413) THEN class = AML	(13/13)
ELSE IF (U46751_at $>$ 5402.227657) AND (M96326_rna1_at $\leq$ 17475.565446) THEN class = AML	(0/0)
ELSE class = ALL	(4/4)
Hypothesis 3:	
IF (M27891_at $\leq$ 1358) AND (Y00787_s_at $\leq$ 11858.081462) THEN class = ALL	(19/19)
ELSE IF (U12471_cds1_at $>$ 173.296678) THEN class = AML	(9/9)
ELSE IF (M27891_at $>$ 7090.800758) AND (M80254_at $\leq$ 693) AND (AFFX-BioC-3_st $\leq$ -225) AND (M27891_at $\leq$ 14708.236206) THEN class = ALL	(0/0)
ELSE IF (Y12670_at $\leq$ 2017.153487) THEN class = AML	(5/6)
ELSE IF (AFFX-CreX-5_at $\leq$ -393.142106) THEN class = AML	(0/0)
ELSE class = ALL	(0/0)
Hypothesis 4:	
IF (M27891_at $>$ 851.428976) AND (U12471_cds1_at $>$ 139.014026) AND (D49950_at $>$ 19.992492) THEN class = AML	(12/12)
ELSE IF (Y12670_at $>$ 1233.788534) AND (M55150_at $>$ 284.216049) THEN class = AML	(0/0)
ELSE IF (M23197_at $\leq$ 288) THEN class = ALL	(19/19)
ELSE IF (AFFX-DapX-3_at $\leq$ -94.34693) THEN class = ALL	(1/2)
ELSE IF (AFFX-BioC-5_st $\leq$ -264.931499) THEN class = AML	(1/1)
ELSE class = ALL	(0/0)

found that among the 7,129 measurements, there were only 30 being chosen by more than two methods [4]. Therefore, here only these 30 measurements are used. The results reported below show that such a feature selection scheme is quite effective. Note that among these 30 measurements only 16 measurements finally appear in the hypotheses generated by C4.5Rule-PANE, as shown in Table 1.

Through setting the parameter μ to 1, 2, 3 and 4, respectively, four different hypotheses have been generated by C4.5Rule-PANE, as listed in Table 2. Here (x/y) at the end of each line indicates that among the 34 test cases, y cases were judged by this line while x cases were correctly judged.

The test accuracy of any of these hypotheses is 97.1%. In fact, each hypothesis made only one misclassification. Note that endogenous control measurements appear in Hypothesis 3 and Hypothesis 4, and therefore Hypothesis 1 and Hypothesis 2 are more preferable.

Since the data used in this case study has been widely investigated, the test accuracy of the above hypotheses can be compared with the best accuracy reported by different researchers, as shown in Table 3. Note that since the machine learning techniques used in most previous research on human acute leukemias generate black-box models, the comprehensibility of the hypotheses generated by C4.5Rule-PANE is much better.

Table 3 shows that the accuracy of the hypotheses generated by C4.5Rule-PANE is very comparable to the best result reported before. In fact, since the test data set is very small (only 34 cases), the reliability of a model is not guaranteed even though its test accuracy reaches 100%. Different to the black-box models used before [1][2][4][5][7][8][13][17][22], the hypotheses generated by C4.5Rule-PANE can be verified by human experts. Therefore, it can be anticipated that they are of greater value in prediction than black-box models. Moreover, it is worth noting that the value of these hypotheses are beyond pure prediction, because they are comprehensible and might help human experts to disclose the relationship between certain genes and diseases.

Table 3. Comparing the test accuracy of the hypotheses generated by C4.5Rule-PANE with the best accuracy reported by different researchers

Authors	Year	Method	Accuray
Golub et al. [8]	1999	Self-Organizing Map	85.3%
Ben-Dor et al. [2]	2000	AdaBoost	95.8%
Furey et al. [7]	2000	Support Vector Machine	94.1%
Li & Yang [13]	2001	Logistic regression	94.1%
Cho & Ryu [4]	2002	Ensemble of heterogeneous classifiers	100%
Dudoit et al. [5]	2002	BoostCART	95.0%
Li & Wong [12]	2002	Emerging Patterns	91.2%
Nguyen & Rocke [17]	2002	Logistic discriminant	97.1%
Albrecht et al. [1]	2003	Ensemble of perceptrons	100%
Tan & Gilbert [22]	2003	Ensemble of decision trees	91.2%
Yun & Keong [23]	2005	Discrete Function Learning	94.1%
this paper	now	C4.5Rule-PANE	97.1%

4 Conclusion

Machine learning techniques have been introduced into gene expression analysis recently. However, almost all the previous works emphasize on constructing models with high prediction accuracy, nevertheless the models are black-boxes. This paper claims that an important role for machine learning techniques to play in gene expression analysis is to help human experts grasp the laws behind biological facts, therefore comprehensible hypotheses might be more helpful than black-box models. Based on this recognition, this paper presents a general approach to generate comprehensible hypotheses from gene expression data, which utilizes a recent proposed machine learning technique, i.e. the C4.5Rule-PANE method. Case study show that this approach work well on human acute leukemias. It is evident that such an approach can be applied to generate comprehensible hypotheses from other gene expression data, which may help enlarge the benefit from microarray technology. In the future the authors expect to work with experts on genes and diseases, wishing that the power of the C4.5Rule-PANE method can be really utilized.

Acknowledgement

We want to thank the anonymous reviewers for their constructive comments. This work was supported by the National Science Foundation of China (60505013), the Jiangsu Science Foundation (BK2004001, BK2005412), and the Fok Ying Tung Education Foundation (91067).

References

1. Albrecht, A., Vinterbo, S.A., Ohno-Machado, L.: An epicurean learning approach to gene-expression data classification. Artificial Intelligence in Medicine **28** (2003) 75–87
2. Ben-Dor, A., Bruhn, L., Friedman, N., Nachman, I., Schummer, M., Yakhini, Z.: Tissue classification with gene expression profiles. Journal of Computational Biology **7** (2000) 559–584
3. Bishop, J.F.: Adult acute myeloid leukaemia: update on treatment. Medical Journal of Australia **170** (1999) 39–43
4. Cho, S.-B., Ryu, J.: Classifying gene expression data of cancer using classifier ensemble with mutually exclusive features. Proceedings of the IEEE **90** (2002) 1744–1753
5. Dudoit, S., Fridlyand, J., Speed, T.P.: Comparison of discrimination methods for the classification of tumors using gene expression data. Journal of the American Statistical Association **97** (2002) 77–87
6. Efron, B., Tibshirani, R.: An Introduction to the Bootstrap. Chapman & Hall, New York (1993)
7. Furey, T.S., Cristianini, N., Duffy, N., Bednarski, D.W., Schummer, M., Haussler, D.: Support vector machine classification and validation of cancer tissue samples using microarray expression data. Bioinformatics **16** (2000) 906–914

8. Golub, T.R., Slonim, D.K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J.P., Coller, H., Loh, M.L., Downing, J.R., Caligiuri, M.A., Bloomfield, C.D., Lander, E.S.: Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. Science **286** (1999) 531–537
9. Guyon, I., Weston, J., Barnhill, S., Vapnik, V.: Gene selection for cancer classification using support vector machines. Machine Learning **46** (2002) 389–422
10. Hayashi, Y., Setiono, R., Yoshida, K.: A comparison between two neural network rule extraction techniques for the diagnosis of hepatobiliary disorders. Artificial Intelligence in Medicine **20** (2000) 205–216
11. Khan, J., Wei, J.S., Ringner, M., Saal, L.H., Ladanyi, M., Westermann, F., Berthold, F., Schwab, M., Antonescu, C.R., Peterson, C., Meltzer, P.S.: Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. Nature Medicine **7** (2001) 673–679
12. Li, J., Wong, L.: Identifying good diagnostic gene groups from gene expression profiles using the concept of emerging patterns. Bioinformatics **18** (2002) 725–734
13. Li, W., Yang, Y.: How many genes are needed for a discriminant microarray data analysis. In: Lin, S.M., Johnson, K.F. (eds.): Methods of Microarray Data Analysis. Kluwer, Boston, MA (2001) 137–150
14. Maughan, N.J., Lewis, F.A., Smith, V.: An introduction to arrays. Journal of Pathology **195** (2001) 3–6
15. Mitchell, T.M.: Machine Learning. McGraw-Hill, New York (1997)
16. Mjolsness, E., DeCoste, D.: Machine learning for science: state of the art and future prospects. Science **293** (2001) 2051–2055
17. Nguyen, D.V., Rocke, D.M.: Tumor classification by partial least squares using microarray gene expression data. Bioinformatics **18** (2002) 39–50
18. Pui, C.H., Evans, W.E.: Acute lymphoblastic leukemia. New England Journal of Medicine **339** (1998) 605–615
19. Quackenbush, J.: Computational analysis of microarray data. Nature Reviews Genetics **2** (2001) 418–427
20. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo, CA (1993)
21. Setiono, R.: Generating concise and accurate classification rules for breast cancer diagnosis. Artificial Intelligence in Medicine **18** (2000) 205–219
22. Tan, A.C., Gilbert, D.: Ensemble machine learning on gene expression data for cancer classification. Applied Bioinformatics **2** (2003) S75–S83
23. Yun, Z., Keong, K.C.: Identifying simple discriminatory gene vectors with an information theory approach. In: Proceedings of the 4th IEEE Computational Systems Bioinformatics Conference. Stanford, CA (2005) 13–24
24. Zhou, Z.-H.: Rule extraction: using neural networks or for neural networks? Journal of Computer Science & Technology **19** (2004) 249–253
25. Zhou, Z.-H., Jiang, Y.: Medical diagnosis with C4.5 rule preceded by artificial neural network ensemble. IEEE Transactions on Information Technology in Biomedicine **7** (2003) 37–42
26. Zhou, Z.-H., Jiang, Y.: NeC4.5: neural ensemble based C4.5. IEEE Transactions on Knowledge and Data Engineering **16** (2004) 770–773
27. Zhou, Z.-H., Wu, J., Tang, W.: Ensembling neural networks: many could be better than all. Artificial Intelligence **137** (2002) 239–263

Classification of Brain Glioma by Using SVMs Bagging with Feature Selection

Guo-Zheng Li[1,2,⋆], Tian-Yu Liu[2], and Victor S. Cheng[3]

[1] State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China

[2] School of Computer Engineering and Science, Shanghai University, Shanghai 200072, China
gzli@staff.shu.edu.cn

[3] Institute of Biomedical Instrument, Shanghai Jiao Tong University, Shanghai 200030, China

Abstract. The degree of malignancy in brain glioma needs to be assessed by MRI findings and clinical data before operations. There have been previous attempts to solve this problem by using fuzzy max-min neural networks and support vector machines (SVMs), while in this paper, a novel algorithm named PRIFEB is proposed by combining bagging of SVMs with embedded feature selection for its individuals. PRIFEB is compared with the general case of bagging on UCI data sets, experimental results show PRIFEB can obtain better performance than the general case of bagging. Then, PRIFEB is used to predict the degree of malignancy in brain glioma, computation results show that PRIFEB obtains better accuracy than other several methods like bagging of SVMs and single SVMs does.

1 Introduction

The degree of malignancy in brain glioma [1] decides the treatment, because if it is grade I or II according to Kernohan, the success rate of operation is satisfactory; otherwise, if it is grade III or IV, there will be high surgical risk and poor life quality [2] after surgery which must be taken into account before any further decision. Now, the degree of malignancy is predicted mainly by Magnetic Resonance Imaging(MRI) findings [3] and clinical data [4] before operations. Some features obtained manually are fuzzy values, some features are redundant, even irrelevant, which makes the prediction of the degree of malignancy a hard task. Moreover, brain glioma is severe but infrequent and only a small number of neuroradiologists have the chances to accumulate enough experiences to make correct judgments. Therefore, it is worth helping the neuroradiologists to predict the degree of malignancy of tumors.

In the previous work, degree prediction of malignancy in brain glioma had been solved by Ye et al. [5], where a fuzzy max-min neural networks was used.

⋆ Corresponding author.

Furthermore, Li et al. [6] tried to employ support vector machines (SVMs) to improve the prediction accuracy, and obtained satisfactory results. Nowadays, ensemble learning is becoming a hot topic in the data mining community [7], which has been widely used to improve the generalization performance of single learning machine. Therefore, using the bagging [8] of SVMs to predict the degree of malignancy is an interesting issue. Yet, there are many irrelevant and redundant features in the medical data sets, so feature selection [9, 10] is important to improve the ensemble method.

In order to improve the accuracy of bagging of SVMs, a novel algorithm of PRIFEB is proposed to use the feature selection methods to improve the accuracy of individuals; this is motivated by the work of Valentini and Dietterich [11], in which they concluded that improving the accuracy of SVMs will improve the accuracy of their bagging. Fortunately, we just finished one work which used an embedded feature selection method to improve the accuracy of SVMs [12]. Therefore, combining them to improve the prediction accuracy of the degree of malignancy is studied in this paper.

The rest of this paper are arranged as follows: In Section 2, a data set of brain glioma is briefly described. In Section 3, PRIFEB, an embedded feature selection model with the prediction risk criteria for bagging, is described in details. In Section 4, PRIFEB is compared with the general bagging method on UCI data sets, and then, PRIFEB is employed to predict the degree of malignancy. At last, conclusions will be given in Section 5.

2 The Data Set for Classification of Brain Glioma

The brain glioma data set [5, 6] was gathered by neuroradiologists from Hua-Shan Hospital in Shanghai of China. There are more than 20 items in each case, including symptoms on different features, preoperative diagnosis made by some neuroradiologist and, without an exception, a clinical grade (the actual grade of glioma obtained from surgery). With the help of domain experts, we chose fifteen features, *Gender, Age, Shape, Contour, Capsule of Tumor, Edema, Mass Effect (Occupation), Post-Contrast Enhancement, Blood Supply, Necrosis/Cyst Degeneration, Calcification, Hemorrhage, Signal Intensity of the T1-weighted Image, Signal Intensity of the T2-weighted Image*, and *Clinical Grade* [4, 13]. In some cases, the value of *Post-Contrast enhancement* is unknown. In fact, *Location* and Size also help to make the diagnosis, but their complex descriptions can not be well modeled by our algorithms, so we didn't adopt it. Except for *Gender, Age* and *Clinical Grade*, the other items are obtained from MRI of the patient and described with uncertainty to various extents. In order to predict the degree of malignancy in brain glioma, descriptions of all features are converted into numerical values as in the work [6], of which the unknown value of *Post-Contrast enhancement* is defined as -1.

Originally, four grades are used to mark the degree of malignancy; we merge grade I and II into low-grade and grade III and IV to high-grade. According to the grade, all 280 cases of brain glioma are divided into two classes: low-grade and high-grade, in which 169 are of low-grade glioma and 111 are of high-grade ones. There are 126 cases containing missing values on *Post-Contrast enhancement*, and in the other subset of 154 complete cases, 85 cases are of low-grade gliomas and 69 are of high-grade.

More details about this data set can also be found in [6].

3 Computational Methods

The classification problem of brain glioma has been solved using the fuzzy max-min neural networks (FMMNN) method, [5], yet, there are so many parameters to be predefined that FMMNN is hard to operate. Then, support vector machines (SVMs) were used to predict the degree of malignancy in brain glioma and obtained satisfactory results [6]. Nowadays, ensemble learning is a state-of-the-art technique, which can effectively improve the prediction accuracy of single learning machine, therefore, we try to employ ensemble learning techniques to improve the prediction accuracy. Yet, there are irrelevant and/or redundant features in the brain glioma data sets which will hurt the performance of learning machine. Motivated by this, we want to use feature selection to remove irrelevant features for ensemble learning techniques and then apply the improved ensemble learning method to the degree prediction of malignancy.

Feature selection for the individuals can help to improve the accuracy of bagging and is based on the conclusion of Valentini and Dietterich [11] where they concluded that reducing the error of support vector machines (SVMs) will reduce the error of bagging of SVMs. At the same time, we used embedded feature selection to reduce the error of SVMs effectively [12]. Enlightened by these works, we propose a novel algorithm PRIFEB (Prediction Risk based Feature sElection for Bagging) which uses the embedded feature selection method with the prediction risk criteria for bagging of SVMs to test if feature selection can effectively improve the accuracy of bagging methods and furthermore improve the degree prediction of malignancy in brain glioma.

In PRIFEB, the prediction risk criteria [14] is used to rank the features, which evaluates one feature through estimating prediction error of the data sets when the values of all examples of this feature are replaced by their mean value.

$$S_i = \mathrm{ERR}(\overline{x}^i) - \mathrm{ERR} \tag{1}$$

where ERR is the training error, and $\mathrm{ERR}(\overline{x}^i)$ is the test error on the training data set with the mean value of i^{th} feature and defined as

$$\mathrm{ERR}(\overline{x}^i) = \frac{1}{\ell}\sum_{j=1}^{\ell}(\widetilde{y}(x_j^1, \cdots, \overline{x}^i, \cdots, x_j^D) \neq y_j)$$

where ℓ is the number of examples and D is the number of features, $\overline{x}^i$ is the mean value of the i^{th} feature. $\widetilde{y}()$ is the prediction value of the j^{th} example after the value of the i^{th} feature is replaced by its mean value. Finally, the feature corresponding with the smallest will be deleted, because this feature causes the smallest error and is the least important one. Since the search strategy is heuristic, the selected feature subset is only near optimal. This criteria had been compared with Optimal Brain Damage (OBD) [15] by using multi-class support vector machines, and obtained better results than OBD did [12].

The basic steps of PRIFEB are described as follows.

Algorithm PRIFEB
Suppose $T_r(x^1, x^2, \cdots, x^D, C)$ is the training set and p is the number of individuals of ensemble.

T_r and p are input into the procedure and ensemble model L is the output.

Step 1. Generate a training subset T_{rk} from T_r by using Bootstrap sampling algorithm [8], the size of T_{rk} is three quarters of the size of T_r.
Step 2. Train an individual model L_k on the training subset T_{rk} by using support vector machines algorithm and calculate the training error ERR.
Step 3. Compute the prediction risk value S_i using Equation (1). If S_i is greater than 0, the i^{th} feature is selected as one of optimal features.
Step 4. Repeat Step 3 until all the features in T_{rk} are computed.
Step 5. Generate the optimal training subset $T_{rk-optimal}$ from T_{rk} according to the optimal features obtained in Step 3.
Step 6. Re-train the individual model L_k on the optimal training subset $T_{rk-optimal}$ by using support vector machines.
Step 7. Repeat from Step 2 to Step 6 until p models are set up, p is 20 in this paper.
Step 8. Ensemble the obtained models L by the way of majority voting method for classification problems.

4 Computational Results

In this section, the novel algorithm of PRIFEB will be validated on the UCI data set, and then applied to the degree prediction of malignancy in brain glioma.

4.1 Experiments on UCI Data Sets

Eight data sets are selected from the UCI machine learning repository [16] and listed in Table 1, in which the number of cases ranges from hundreds to thousands and the number of features ranges from 9 to 35. To make them suitable for our algorithms, the nominal values are changed to be numerical in all data sets. Then, all the attributes are transformed into the interval of [-1, 1] by an affine function.

The hold out method is used to validate the results. Experiments will be repeated fifty times on each data set. The same pair of parameters for SVMs, C =100, σ =10, is used and the number of individuals for bagging is 20.

Experimental results of the accuracy obtained by the different bagging methods are shown in Table 2, from which we can see that the mean accuracy obtained by the bagging methods with feature selection are improved in various degree on different data sets which ranges from 1% to 6%, the mean value improved is 3.17 percent for the PRIFEB method. At the same time, the standard deviation are also reduced in some degree.

Table 1. The properties of the used UCI data sets

Data set	Number of classes	Number of features	Number of cases
all-bp	3	29	3772
backup	19	35	683
breast-cancer-W	2	9	699
glass	6	9	214
proc-C	5	13	303
proc-H	2	13	294
soybean-l	19	34	307
statlog-h	2	13	270

Table 2. Statistical prediction accuracy by the bagging methods with feature selection and/or without feature selection on the UCI data sets

Data set	PRIFEB	Bagging
all-bp	97.07 ± 0.45	95.95 ± 0.13
backup	91.99 ± 1.41	89.90 ± 2.06
breast-cancer-W	94.38 ± 1.02	91.23 ± 1.71
glass	64.95 ± 3.74	61.75 ± 5.12
proc-C	53.71 ± 3.67	49.93 ± 3.65
proc-H	79.84 ± 2.55	73.89 ± 3.52
soybean-l	85.54 ± 3.64	83.25 ± 3.57
statlog-h	78.62 ± 3.30	74.88 ± 3.87
Average	80.76 ± 2.47	77.60 ± 2.59

4.2 The Brain Glioma Case

To improve the prediction accuracy, we employ the bagging of SVMs and a novel algorithm PRIFEB to the classification of brain glioma. To compare the classification ability of the methods of PRIFEB with that of bagging of SVMs and single SVMs, the 10-fold cross validation technique is used in this computation. The SVMs used in this experiment are with linear kernel and the parameter of the trade off between the complexity and the error is $C = 100$, and the number of individuals of bagging is 20. These parameters are not optimal, but they can

Table 3. Results of accuracy obtained by 10-fold cross validation method

	Data set	R_{acc} (%)	Std. dev.(%)	Highest(%)	Lowest(%)
PRIFEB	D280	87.29	7.91	97.68	75.00
	D154	86.57	8.23	100.00	73.33
Bagging of SVMs	D280	86.36	7.22	96.43	75.00
	D154	86.43	8.17	100.00	73.33
SVMs	D280	85.70	6.52	96.43	71.43
	D154	84.96	9.97	100.00	73.33
FMMNN-FRE	D280	83.21	5.31	89.29	75.00
	D154	86.37	8.49	100.00	73.33

make the learning methods obtain satisfying results according to the experience [6]. Computational results are shown in Table 3, where D280 is the total data set of 280 cases with missing values in *Post-Contrast enhancement*, while D154 is the the data set of 154 complete cases. The results of the previous studies are also reprinted in Table 3, where results of SVMs are collected from [6] and those of FMMNN-FRE are collected from [5].

From Table 3, it can be seen that 1) The results of Bagging are better than that of single SVMs, which show that Bagging of SVMs can really improve the prediction accuracy of single ones. 2) The results of PRIFEB, the novel algorithm, are better than that of the general case of bagging of SVMs.

5 Conclusions

To improve the degree prediction accuracy of malignancy in brain glioma, a novel algorithm of PRIFEB (Prediction RIsk based Feature sElection for Bagging) is proposed in this paper. The experimental results on UCI data sets and the tumor data set show that PRIFEB obtained better results than bagging of SVMs and single SVMs did. From the view of the classification of brain glioma, the computation results imply that there is a close relation between the degree of malignancy in brain glioma with the MRI findings and clinical data, and this relationship can be modeled by SVMs based method, of which PRIFEB is the best method on prediction accuracy.

From the view of learning methods, the computation results imply that there are redundant features in the brain glioma and other real world data sets, and feature selection can really improve the accuracy of bagging of SVMs. For the success of PRIFEB, we think that feature selection can reduce the irrelevant features and even redundant features to improve the accuracy of single individuals, which has also been proven by the previous work. At the same time, feature selection reduces different features for different individuals and help to increase the diversity among the individuals of bagging. According to the theory [7], improving the accuracy of each individual and increasing the diversity among individuals will effectively improve the accuracy of an ensemble method. The above two aspects caused by feature selection will make it true that feature selection can improve the accuracy of bagging of SVMs.

This work improves the classification accuracy of brain glioma and shows the effectiveness of feature selection for bagging. There is still much work to be done, such as further improving the feature search method to make feature selection efficient and effective.

Acknowledgement

This work was supported by NSFC (20503015) and SHMEC NSF (05AZ67). The authors would like to thank the anonymous reviewers for their valuable advices.

References

1. Bredel, M., Pollack, L.F.: The P21-Ras Signal Transduction Pathway and Growth Regulation in Human High-Grade Gliomas. Brain Research Reviews **29** (1999) 232–249
2. Wang, C., Zhang, J., Liu, A., Sun, B., Zhao, Y.: Surgical Treatment Of Primary Midbrain Gliomas. Surg Neurol **53** (2000) 41–51
3. Lopez Gonzalez, M.A., Sotelo, J.: Brain Tumors in Mexico: Characteristics and Prognosis of Glioblastoma. Surg Neurol **53** (2000) 157–162
4. Chow, L.K., Gobin, Y.P., Cloughesy, T.F., Sayre, J.W., Villablanca, J.P., Vinuela, F.: Prognostic Factors in Recurrent Glioblastoma Multiforme and Anaplastic Astrocytoma Treated with Selective Intra-Arteral Chemotherapy. AJNR Am J Neuroradiol **21** (2000) 471–478
5. Ye, C.Z., Yang, J., Geng, D.Y., Zhou, Y., Chen, N.Y.: Fuzzy Rules to Predict Degree of Malignancy in Brain Glioma. Medical and Biological Engineering and Computing **40** (2002) 145–152
6. Li, G.Z., Yang, J., Ye, C.Z., Geng, D.: Degree Prediction of Malignancy in Brain Glioma Using Support Vector Machines. Computers in Biology and Medicine **36** (2006) in press
7. Dietterich, T.: Machine-Learning Research: Four Current Directions. The AI Magazine **18** (1998) 97–136
8. Breiman, L.: Bagging Predictors. Machine Learning **24** (1996) 123–140
9. Kohavi, R., George, J.H.: Wrappers for Feature Subset Selection. Artificial Intelligence **97** (1997) 273–324
10. Guyon, I., Elisseeff, A.: An Introduction to Variable and Feature Selection. Journal of machine learning research **3** (2003) 1157–1182
11. Valentini, G., Dietterich, T.: Bias-Variance Analysis Of Support Vector Machines for The Development Of SVM-Based Ensemble Methods. Journal of Machine Learning Research **5** (2004) 725–775
12. Li, G.Z., Yang, J., Liu, G.P., Xue, L.: Feature Selection for Multi-Class Problems Using Support Vector Machines. In: Lecture Notes on Artificial Intelligence 3173, Auckland, New Zealand, Springer (2004) 292–300
13. Arle, J.E., Morriss, C., Wang, Z., Zimmerman, R.A., Phillips, P.G., Sutton, L.N.: Prediction of Posterior Fossa Tumor Type in Children by Means of Magnetic Resonance Image Properties, Spectroscopy, and Neural Networks. Journal of Nonsurgical **86** (1997) 755–761
14. Moody, J., Utans, J.: Principled Architecture Selection for Neural Networks: Application to Corporate Bond Rating Prediction. In Moody, J.E., Hanson, S.J., Lippmann, R.P., eds.: Advances in Neural Information Processing Systems. Volume 4., Morgan Kaufmann Publishers, Inc. (1992) 683–690
15. Guyon, I., Weston, J., Barnhill, S., Vapnik, V.: Gene Selection for Cancer Classification Using Support Vector Machines. Machine Learning **46** (2002) 389–422
16. Blake, C., Keogh, E., Merz, C.J.: UCI Repository of Machine Learning Databases. Technical report, Department of Information and Computer Science, University of California, Irvine, CA (1998) http://www.ics.uci.edu/ mlearn/MLRepository.htm.

Missing Value Imputation Framework for Microarray Significant Gene Selection and Class Prediction

Muhammad Shoaib B. Sehgal, Iqbal Gondal, and Laurence Dooley

Faculty of IT, Monash University, Churchill VIC 3842, Australia
{Shoaib.Sehgal,Iqbal.Gondal,
Laurence.Dooley}@infotech.monash.edu.au

Abstract. Microarray data is used in a large number of applications ranging from diagnosis through to drug discovery. Such data however, often contains multiple missing genetic expressions which are generally ignored thus degrading the reliability of inferred results. This paper presents an innovative and robust imputation framework that more accurately estimates missing values leading subsequently to better gene selection and class prediction. To prove this premise, several missing value techniques including the *Collateral Missing Values Estimation* (CMVE), *Bayesian Principal Component Analysis* (BPCA), *Least Square Impute* (LSImpute), *k-Nearest Neighbour* (KNN) and *ZeroImpute* are analysed. A combination of univariate and multiple gene selection methods, namely, *Between Group to within Group Sum of Squares* and *Weighted Partial Least Squares* is then performed before applying class prediction using the *Ridge Partial Least Square* method. Overall, CMVE imputation consistently provided superior missing values estimation accuracy compared with the other algorithms examined, by virtue of exploiting local and global as well as positive and negative correlations between genes, with all empirical results being corroborated by the two-sided Wilcoxon Rank sum statistical significance test.

1 Introduction

Microarray gene expression data is already extensively utilised in providing a valuable insight in many different areas of both biological and medical science. Many researchers have considered such data for class prediction applications including, Sehgal et al [1] for breast and ovarian cancer classification, Golub et al [2] for acute leukemia and Bhattacharjee et al [3] in the area of human lung carcinomas. Despite its wide usage, microarray data frequently contains missing values with up to 90% of genes affected [4]. The reasons for these missing values are many and varied, ranging from spotting problems, slide scratches, chip blemishes, hybridization errors and image corruption through to simply dust on the slide [5]. Whatever the reason, missing values have the potential to make a significant impact upon subsequent inferences made from the microarray data, in applications such as gene selection and class prediction, which involve techniques like *Between Group to within Group Sum of Squares* (BSS/WSS), *Neural Networks* (NN), *Support Vector Machines* (SVM), *Principal Component Analysis* (PCA) and *Singular Value Decomposition* (SVD). There are many well established approaches to address the problem of missing values, from repeating the experiment to simply ignoring those samples containing missing values. Both these solutions however, are unacceptable due to the cost and limited availability of samples, so a better strategy is to attempt to accurately estimate the missing values by exploiting the correlation between data so the missing value prediction error is significantly reduced [4].

Another feature of microarray data is that in general, the number of samples is relatively small compared to the number of genes in each sample (usually thousands), with the result that most classical class prediction methods perform poorly due to over-fitting of the training data [6]. In such circumstances, feature selection techniques are a preferred choice, though since these involve dimension reduction i.e. PCA and SVD, they do not consider class discrimination when converting data to Eigen space with the corollary that lower class prediction accuracy is achieved. Univariate algorithms provide an alternative solution, with examples including the *t-test*, *signal to noise ratio*, BSS/WSS and *Significance Analysis of Microarray* [7], though each of these is designed either for a binary class response or considers each relevant gene individually in selecting the most correlated genes, thus introducing redundancy. These problems can be avoided if multivariate gene selection is applied simultaneously to consider multiple genes and class information [8] to reduce covariate genes, while retaining class discrimination, though the drawback of this strategy is that if a multivariate method is combined with a class prediction technique, it becomes highly dependent on the learning method employed.

This paper proposes an innovative framework (Fig. 1) to address the aforementioned problems by applying the *Collateral Missing Value Estimation* (CMVE) algorithm [1], which not only generates a lower prediction error compared with other existing methods including, *Bayesian Principal Component Analysis* (BPCA), *Least Square Impute* (LSImpute), *k-Nearest Neighbour* (KNN) and *ZeroImpute,* but has also increases the classification accuracy for a range of missing values from 1% to 5% for multi-class Breast cancer datasets [9]. The paper emphasizes the core tenet that better estimation of missing values leads to more accurate gene selection and subsequent class prediction results. For gene selection, a cascaded approach is adopted employing both univariate and multivariate techniques. Firstly, *p* discriminant genes are selected using BSS/WSS to exploit the model independence property of univariate methods before redundant genes are removed using a *Weighted Partial Least Square* (WPLS) algorithm (Fig. 1). Applying BSS/WSS gene selection prior to WPLS affords the advantage of reducing the computational time by constraining the WPLS search space. For class prediction, several approaches can be employed including *Genetic Algorithms (GA)*, NN, SVM [10] and *Diagonal Linear Discriminant Analysis (DLDA)*, however, the motivation for using the *Ridge Partial Least Squares* (RPLS) technique in the new framework is its improved performance compared to other classification algorithms due to being able to handle numerical degeneracy of multicollinearity in gene expression data using *Penalized Logistic Regression* [7]. The classifier is applied to a test dataset to predict classes reliably in multi-class microarray data by regressing the significant genes with a ridge penalty [7]. Finally, a *two-sided Wilcoxon rank sum significance* test is applied to validate the hypothesis that lower missing value estimation errors lead to better subsequent microarray data analysis.

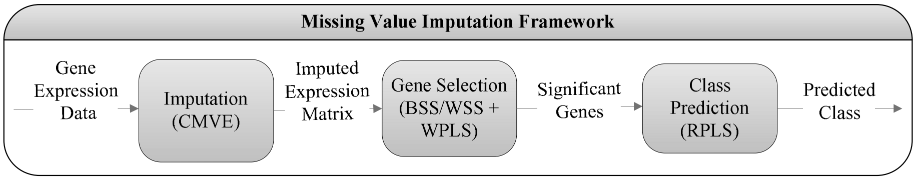


Fig. 1. Missing Value Imputation Framework for Significant Gene Selection and Class Prediction

The remainder of this paper is organized as follows. Section 2 outlines the various algorithms used for missing value imputation, gene selection and classification respectively, while Section 3 presents a rigorous analysis of their respective empirical performance. Some conclusions are provided in Section 4.

2 Review of Missing Value, Gene Selection and Classification Algorithms

This section respectively outlines the various missing value imputation, gene selection and class prediction algorithms (Fig. 1) that are applied and compared in this paper.

2.1 Missing Value Estimation Algorithms

A brief review is now presented of the key features of the estimation techniques used to compare the imputation performance of the CMVE algorithm. For clarity, the following convention is adopted: Y is assumed to be the gene expression matrix $Y \in \mathbb{R}^{m \times n}$, where m and n are the number of genes and samples respectively, g_I is gene expression vector of gene I and $g_I(J)$ is the expression value of g_I in sample J.

2.1.1 Least Square Impute (LSImpute) [11]

This is a regression-based estimation method that exploits the correlation between genes. To estimate the missing value Ξ of g_I of gene expression matrix Y, the k-most correlated genes are firstly selected whose expression vectors are similar to gene I from Y in all samples except J, containing non-missing values for gene I. By having the flexibility to adjust the number of predictor genes k in the regression, LSImpute performs best whenever the data possesses a strong localized correlation structure, though there is no formal strategy for automatically determining the best k value.

2.1.2 Bayesian Principal Component Analysis (BPCA) [5]

This approach estimates missing values Ξ in data matrix Y by using Y_{obs}, which have genes with no missing values. The Bayesian estimation algorithm is executed for both model parameters computed using Bayes estimates θ and Ξ (similar to the *Expectation Maximization* repetitive algorithm) and calculates the posterior distributions for θ and Y_{miss}, $q(\theta)$ and $q(\Xi)$ [5], before the missing values in Y are imputed using:

$$\hat{Y} = \int \Xi \, q(\Xi) \, d\Xi \tag{1}$$

$$q(\Xi) = p(\Xi \mid Y^{obs}, \theta_{true}) \tag{2}$$

where θ_{true} is the posterior distribution of the missing value. BPCA however only considers the global correlation structure of the data, so this algorithm is not well suited for data which has a strong local correlation structure.

2.1.3 k-Nearest Neighbour (KNN) [12]

This algorithm estimates the missing value Ξ for *$g_I(J)$,* by selecting k genes whose expression vectors are similar to *$g_I(J)$* [12]. The k genes are selected by computing the distance of g_I from all the other genes in $Y \in \mathbb{R}^{m \times n}$ for sample J. The missing value is then estimated as the weighted average of the corresponding entries in the selected k expression. The Euclidean distance used by KNN is sensitive to outlier values which may be present in microarray data; although log-transforming the data significantly reduces their effects on gene similarity determination [12]. The main drawbacks of KNN are that it does not consider negative correlations between data which can lead to higher estimation errors [4] and secondly, as with LSImpute, there is no deterministic method to resolve the best value of k.

2.1.4 Collateral Missing Value Estimation (CMVE) [1]

This new algorithm, which is detailed in Fig. 2, is based on generating multiple parallel missing value estimations, which are subsequently combined to construct the final imputation value. For example, if value *$g_I(J)$*, of gene I and sample J is missing, then firstly the diagonal covariance of I is computed together with the other gene expressions using (3), where m is the number of genes, I is the gene number with missing value for sample J and G_i is the gene in Y, other than I. Rows are then sorted according to their covariance, with the first k-ranked covariate genes R_k being selected.

$$C = \frac{1}{(n-1)} \sum_{i=1}^{m} (I_i - \bar{I})(G_i - \bar{G}) \tag{3}$$

Covariance is used in CMVE instead of a distance function because it considers both negative and positive correlation values, in contrast to the Euclidean distance used by KNN, which only considers positive correlations [13]. Another option would have been to use Pearson Correlation, though the overall effect is exactly same for normally distributed data i.e. z-scored [14], so therefore covariance is chosen due to its lower computation complexity. The missing values are then estimated by fusing together multiple estimates Φ_1, Φ_2 and Φ_3 (see (9)), with the various steps involved in the CMVE imputation algorithm now discussed.

Φ_1 is the estimate of *$g_I(J)$* (Step 4a) using the *LSImpute* linear regression method (Section 2.1.1), while Step 4b estimates two other sets of missing values Φ_2 and Φ_3, the former is estimated using:

$$\Phi_2 = \sum_{i=1}^{k} \phi + \eta - \sum_{i=1}^{k} \xi^2 \tag{4}$$

```
Pre Condition: Gene expression matrix $Y$ with $m$ number of genes, $n$ samples, $I$ missing values, $index=1$
Post Condition: $Y$ without any missing values.
Algorithm:
Step 1 Compute absolute covariance $C$ using (3)
Step 2 Rank genes (rows) based on $C$
Step 3 Select the $k$ most effective rows $R_k$
Step 4 Use values of $R_k$ to
    Step 4a Estimate value $\Phi_1$ using Least Square Regression
    Step 4b Compute $\Phi_2$ and $\Phi_3$ using (4) and (5)
Step 5 Compute missing value of $I[index]$ using (9) and reuse in future predictions
Step 6 Increment $index$ and Repeat Steps 1–5 until all missing values of G are estimated
```

Fig. 2. Collateral Missing Value Estimation (CMVE) Algorithm

While the value of Φ_3 is computed using:

$$\Phi_3 = \frac{\sum_{i=1}^{k} (\phi^T \times I)}{k} + \eta \tag{5}$$

where η and ϕ in (4) and (5) are obtained using the *Non Negative Least Square* (NNLS) method [4]. The aim is to find a linear combination of models, that best fit R_k and I. The objective function in NNLS is used to minimise the prediction error ξ_0 as:

$$\phi, \eta = \min(\xi_0) \tag{6}$$

Linear programming is used to compute the coefficients ϕ that generate the minimum prediction error and residual η. The value of ξ_0 in (6) is calculated using:

$$\xi_0 = \max(SV(R_k.\phi - I)) \tag{7}$$

where SV are the singular values of the difference vector between product R_k and prediction coefficients ϕ with the gene expression row I containing missing values. The tolerance used by the linear programming method to compute vector ϕ is:

$$Tol = k \times N \times \max(SV(R_k)) \times N_f \tag{8}$$

where k = number of predictor genes R_k and N_f is the number of predictor gene samples. Finally, value χ for $g_I(J)$ is computed (Fig. 2. Step 5) using:

$$\chi = \alpha.\Phi_1 + \beta.\Phi_2 + \gamma.\Phi_3 \tag{9}$$

where α, β and γ are set to 0.33 to ensure an equal weighting to the respective estimates Φ_1, Φ_2 and Φ_3. The rationale for this choice is that as each estimate is highly data dependent, it avoids any bias towards one particular estimate.

After imputing the missing values, significant genes are selected using BSS/WSS. The following section provides a short overview of this technique, together with the motivation for using this gene selection algorithm.

2.2 Feature Selection Method: Between Group to Within Group Sum of Squares (BSS/WSS)

This gene selection method identifies those genes which concomitantly have large inter-class variations and small intra-class variations. For any gene *I* in $Y \in \mathbb{R}^{m \times n}$ BSS/WSS is calculated as follows:

$$BSS(I)/WSS(I) = \frac{\sum_{t=1}^{T}\sum_{q=1}^{Q} F(L_t = q)(\bar{Y}_{qI} - \bar{Y}_I)^2}{\sum_{t=1}^{T}\sum_{q=1}^{Q} F(L_t = q)(Y_{It} - \bar{Y}_{qI})^2}, \tag{10}$$

where *T* is the training sample size, *Q* is the number of classes and *F(•)* is a Boolean function which is 1 if the condition is true, and zero otherwise. $\bar{Y}_J$ denotes the average expression level of gene *I* across all samples and $\bar{Y}_{qI}$ is the average expression level of gene *I* across all samples belonging to class *q*. Genes are then ranked by BSS/WSS ratios, from the highest to the lowest to form a significant gene expression matrix ϑ, with the first *p* genes selected for subsequent class prediction. This particular gene selection method is preferred because of its wide usage, model independence and ability to increase class separability [15]. To eliminate all the correlated genes from *p*, a *Weighted Partial Least Square* (WPLS) algorithm is applied. The attraction of combining WPLS with BSS/WSS in the gene selection process is that BSS/WSS does not simultaneously select multiple genes and so accounts for gene interdependency. Also, it ignores model uncertainty by firstly predicting the set of relevant genes and then the relevant class [16], while WPLS takes cognisance of model uncertainty by considering class prediction accuracy. In circumstances where only WPLS is applied, the selected genes will be highly dependent on the prediction model [8]. BSS/WSS also provides a smaller gene-to-sample ratio which results in shorter convergence times for the WPLS algorithm.

2.3 Class Prediction: Ridge Partial Least Squares (RPLS)

For class prediction, several different approaches have been proposed including GA, NN, SVM [10] and DLDA, however the *Ridge Partial Least Squares* (RPLS) method has demonstrated better performance for gene expression data classification due to its ability to handle numerical degeneracy of multi-collinearity in gene expression data by penalizing the likelihood [7]. RPLS uses *Partial Least Squares* (PLS) with *Penalized Logistic Regression* (PLR) for class prediction and comprises three major steps:

1- Replace class labels with the continuous pseudo-response variable Z^{∞} and then estimate Z^{∞} and the weighted matrix W^{∞} using *Iterative Re-Weighted Least Square with Ridge Penalty* (RIRLS):

$$(Z^{\infty}, W^{\infty}) = RIRLS(L, Y, \lambda), \tag{11}$$

where λ is a positive real constant derived by minimizing the *Bayesian Information Criterion* (BIC) [17] and *L* is a set containing discrete class labels.

2- Matrices Z^{∞} and W^{∞} are used to compute $\hat{\alpha} \in \mathbb{R}^{p+1}$ using the WPLS method:

$$\hat{\alpha}^{PLS,\kappa} = WPLS(Z^{\infty}, Y, W^{\infty}, \kappa), \tag{12}$$

where Y is the input matrix and κ is a positive integer which determines the number of iterations.

3- Finally, the class response is determined using *Linear Logistic Discrimination* (LLD), where the conditional class probability of response L for a given data Y is:

$$P(L = 1 | Y = y;\ \hat{\alpha}\ ), \tag{13}$$

where parameter $\hat{\alpha} \in \mathbb{R}^{p+1}$ is estimated using (12) and p is number of significant genes determined from BSS/WSS (Section 2.2). The probability P in (13) is given by:

$$P(L = 1 | Y = y;\ \hat{\alpha}\ ) = h([1\ \ y]\hat{\alpha}), \tag{14}$$

where $h(\eta) = 1/[1+exp(-\eta)]$ and the quantity $h([1\ \ y]\hat{\alpha})$ is a linear predictor. The log-likelihood of the observations of parameter $\hat{\alpha}$ is given by:

$$l(\hat{\alpha}) = \sum_{i=1}^{n} \{L_i \upsilon_i(\hat{\alpha}) - \ln[1 + \exp(\upsilon_i(\hat{\alpha}))]\}, \tag{15}$$

which for all $1 \le i \le n, \upsilon_i(\hat{\alpha}) = (Z\hat{\alpha})_i$ and Z = $[\Upsilon_n\ \ Y]$ of size $n \times (p+1)$ and Υ_n is the column matrix of size n. The class label $L = 1$ if $\wp > 1\text{-}\wp$ and zero otherwise where:

$$\wp = h([1\ \ y]\hat{\alpha}). \tag{16}$$

The following Section provides a detailed analysis of the empirical results confirming the improved performance of the proposed imputation framework (Fig. 1) compared to existing approaches.

3 Results Analysis

To quantitatively evaluate the performance of the different imputation methodologies to improve gene selection and classification, the established breast cancer microarray dataset [9] was used in all the experiments. The dataset contains 7, 7 and 8 samples of BRCA1, BRCA2 and Sporadic mutations (neither BRCA1 nor BRCA2) respectively, with each data sample containing logarithmic microarray data of 3226 genes. To assess the affect of missing values on gene selection, a set of significant genes Ъ was selected using the BSS/WSS from the original data with no missing values, to serve as a bench mark. Then, between 1% and 5% of expression values were randomly removed from the actual dataset. This was followed by imputation using ZeroImpute, BPCA, LSImpute, KNN and CMVE respectively. Significant genes were selected from these imputed matrices iteratively and compared with the Ъ to compute *True Positive* (TP) rate. Fig.3. demonstrates high TP rate achieved by CMVE, compared to

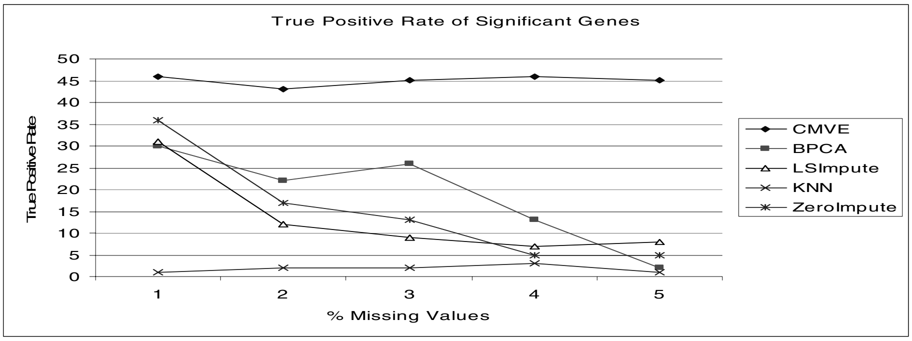


Fig. 3. True Positive Rate of 50 Most Significant Genes

other imputation techniques due to its better estimation ability, for a set of 50 significant genes. Also, the same improved performance was observed for higher number of selected genes.

To investigate the impact of estimation on class prediction, again, between 1% and 5% of data values were randomly removed and the most significant genes were selected using the BSS/WSS and WPLS combination, followed by classification using the RPLS algorithm, with missing values being estimated using ZeroImpute, BPCA, LSImpute, KNN and CMVE respectively.

The validation results were generated using *k*-fold (leave one out) cross validation. The motivation to use this technique over the *classical hold out* or *random resampling* methods was that it uses data sets evenly both for training and testing, thereby giving better estimation of the classification rates. To remove the bias in the gene selection step, significant genes were selected by using the training folds only while selecting the corresponding genes in the validation fold.

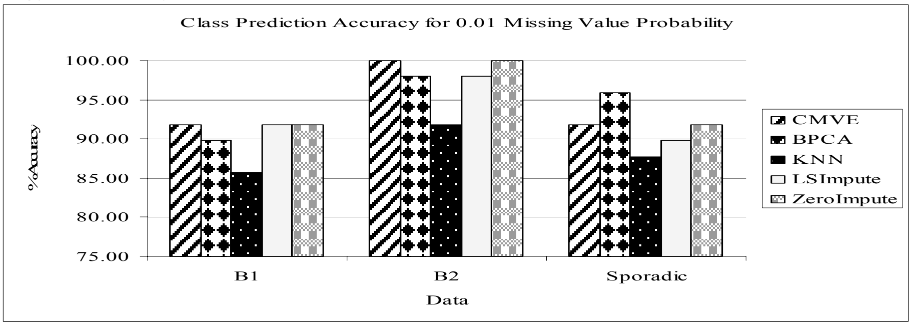


Fig. 4. Class Prediction Accuracy of 1% Missing Valued Gene Expression Data

Figs. 4-7 show various individual classification accuracies for the range of missing values from 1% to 5%, while Fig. 8 plots the overall classification performance for the range of missing values, which in particular confirms CMVE consistently outperformed BPCA, LSImpute, KNN and ZeroImpute. In particular, CMVE performed far better than BPCA especially for higher number of missing values because BPCA only

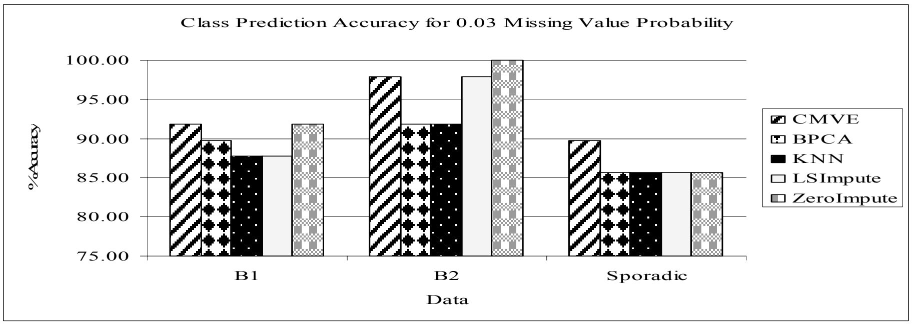


Fig. 5. Class Prediction Accuracy of 3% Missing Valued Gene Expression Data

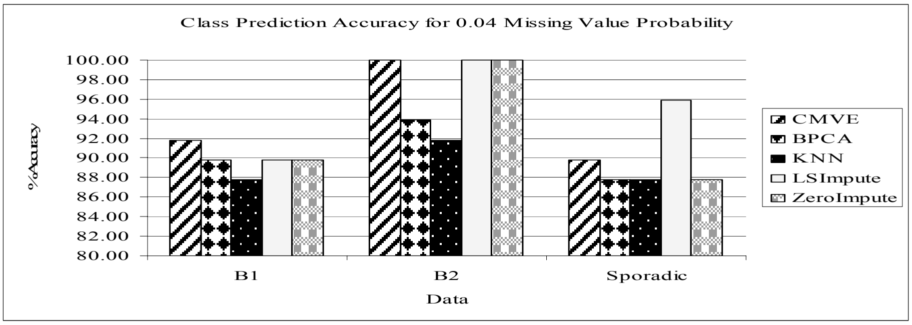


Fig. 6. Class Prediction Accuracy of 4% Missing Valued Gene Expression Data

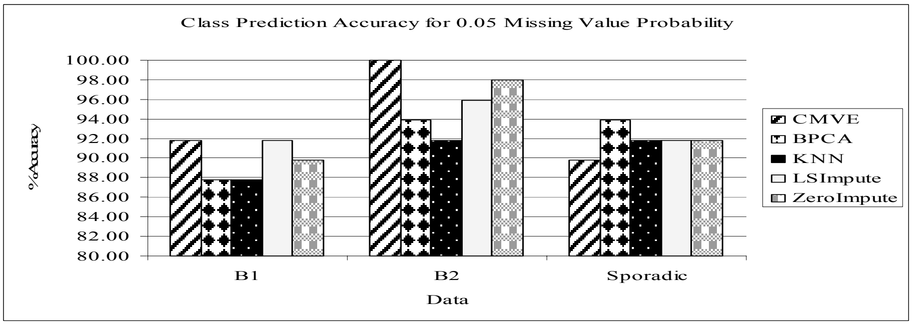


Fig. 7. Class Prediction Accuracy of 5% Missing Valued Gene Expression Data

considers global correlations and also does not consider estimated values for future estimates which leads to higher estimation errors [1].

The individual results interestingly reveal that LSImpute provided a slightly improved prediction accuracy compared with CMVE at 4% missing values (Fig. 8) in the Sporadic dataset, due to the higher classification accuracy achieved for this dataset (Fig. 6), though for all other datasets, CMVE performed either equally well or better than LSimpute at 4% missing values (Fig. 6 BRCA1). The reason for this apparent

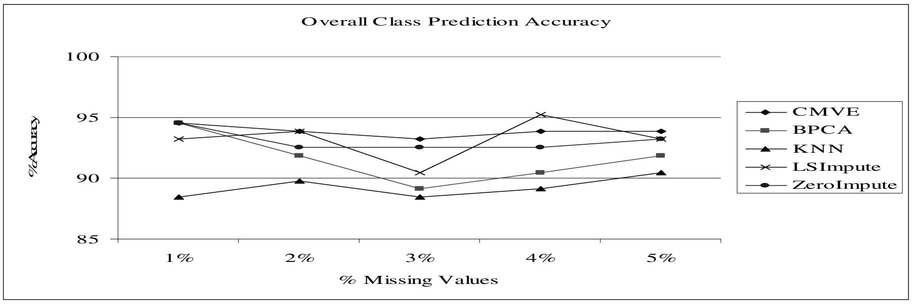


Fig. 8. Overall Class Prediction Accuracy

anomaly is that in computing the final missing value in (9), CMVE gives equal weighting to all three individual estimates in order to provide an unbiased estimate, a strategy which may not be optimal in certain cases.

Also, noteworthy is the fact ZeroImpute performed better than the other estimation techniques, including CMVE, for BRCA2 class when data had 3% missing values (Fig. 5, BRCA2) because the data between classes were more separable and thus easier to classify, i.e., zero values actually improved separability. These occurrences however, are very reliant on serendipity as to when better accuracy is achieved and thereby superior separation in terms of gene selection and class prediction. In practice, it is highly improbable as for the vast majority of datasets, zero imputation does not improve separability because for instance if a particular gene has missing values, for both classes to be classified, ZeroImpute produces the same value, namely zero [18]. This means the gene has the same value for both classes despite some genes being more significant than others, so it is always better to attempt to accurately estimate missing values rather than to just impute zero values.

To evaluate the estimation performance of all the imputation algorithms and also corroborate the superior performance of CMVE, the *two-sided Wilcoxon Rank sum statistical significance* test was applied. The motivation for using this particular test is that compared to some other parametric significance tests [19], it does not mandate data of equal variance, which is vital given that the variance of data can be disturbed due to erroneous estimation, especially for ZeroImpute. To test the hypothesis $H_{0,}\ Y \neq Y_{est}$ where Y and Y_{est} are the actual and estimated matrices respectively, the P-value of the hypothesis is calculated using:

$$H_{0,}\ P\text{-}Value = 2P_r(R \leq y_r) \qquad (17)$$

where y_r is the sum of the ranks of observations for Y and R is the corresponding random variable.

Fig. 9 plots the average P value of H_o for the range of estimations for between 1% and 5% missing values for each of the aforementioned imputation techniques. The results show CMVE provides the best performance, as a low P value of H_0 rejects the null hypothesis [19], while not surprisingly ZeroImpute exhibits the largest disparity with the original data because it does not consider any underlying correlation structure latent in the data.

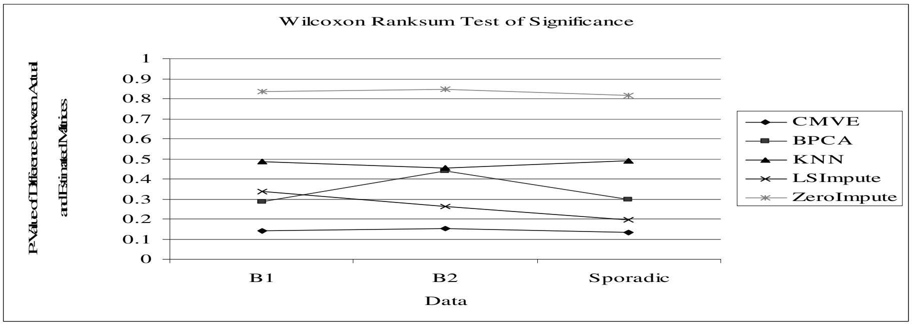


Fig. 9. Two-Sided Wilcoxon Rank Sum Significance Test Results

4 Conclusions

This paper has presented a new framework based on the *Collateral Missing Value Estimation* (CMVE) algorithm for accurate missing value estimation for gene selection and class prediction. CMVE has demonstrated superior imputation performance compared to the *Bayesian Principal Component Analysis* (BPCA), *Least Square Impute* (LSImpute), *k-Nearest Neighbour* (KNN) algorithm and *ZeroImpute* method, for estimating randomly missing values over a probability range from 0.01 to 0.05 in the BRCA1, BRCA2 and Sporadic genetic mutation samples present in breast cancer. Experimental results vindicate that CMVE consistently outperformed comparative methods in terms of their classification accuracies by exploiting positive and negative, as well as local and global data correlations. The *Between Group to within Group Sum of Squares* (BSS/WSS) and *Weighted Partial Least Square* (WPLS) combination which was used as the feature selection method, together with the *Ridge Partial Least Squares* (RPLS) classifier collectively provided consistently improved classification performance for all experiments on the test microarray dataset, when combined with CMVE. A significance test also confirmed the fundamental hypothesis that accurate missing value estimation is essential to ensure subsequent superior class prediction and gene selection.

References

[1] M. S. B. Sehgal, I. Gondal, and L. Dooley, "Collateral Missing Value Imputation: a new robust missing value estimation algorithm for microarray data," *Bioinformatics*, vol. 21(10), pp. 2417-2423, 2005.

[2] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasen-beek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Down-ing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lan-der, "Molecular classification of cancer: class discovery and class prediction by gene expression monitoring," *Science*, pp. 286(5439):531-537, 1999.

[3] A. Bhattacharjee, W. G. Richards, J. Staunton, C. Li, S. Monti, P. Vasa, C. Ladd, J. Beheshti, R. Bueno, M. Gillette, M. Loda, G. Weber, E. F. Mark, E. S. Lander, W. Wong, B. E. Johnson, T. R. Golub, D. J. Sugarbaker, and M. Meyerson, "Classification of human lung carcinomas by mRNA expression profiling reveals distinct adenocarcinoma subclasses," *Proc. Natl. Acad. Sci*, pp. 13790-13795, 2001.

[4] M. S. B. Sehgal, I. Gondal, and L. Dooley, "A Collateral Missing Value Estimation Algorithm for DNA Microarrays," *2005 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), USA*, pp. 377-380, 2005.

[5] S. Oba, M. A. Sato, I. Takemasa, M. Monden, K. Matsubara, and S. Ishii, "A Bayesian Missing Value Estimation Method for Gene Expression Profile Data," *Bioinformatics*, vol. 19, pp. 2088-2096, 2003.

[6] M. S. B. Sehgal, I. Gondal, and L. Dooley, "Support Vector Machine and Generalized Regression Neural Network Based Classification Fusion Models for Cancer Diagnosis," *IEEE Hybrid Intelligent Systems (HIS)'04, Japan*, pp. 49–54, 2004.

[7] G. Fort and S. Lambert-Lacroix, "Classification using partial least squares with penalized logistic regression," *Bioinformatics*, vol. 21, pp. 1104-1111, 2005.

[8] X. Liu, A. Krishnan, and A. Mondry, "An Entropy-based gene selection method for cancer classification using microarray data," *BMC Bioinformatics*, vol. 6:76, 2005.

[9] I. Hedenfalk, D. Duggan, Y. Chen, M. Radmacher, M. Bittner, R. Simon, P. Meltzer, B. Gusterson, M. Esteller, O. P. Kallioniemi, B. Wilfond, A. Borg, and J. Trent, "Gene-expression profiles in hereditary breast cancer," *N. Engl. J. Med*, pp. 22; 344(8):539-548, 2001.

[10] M. S. B. Sehgal, I. Gondal, and L. Dooley, "Statistical Neural Networks and Support Vector Machine for the Classification of Genetic Mutations in Ovarian Cancer," *IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)'04, USA.*, pp. 140-146, 2004.

[11] T. H. Bø, B. Dysvik, and I. Jonassen, "LSimpute: Accurate estimation of missing values in microarray data with least squares methods," *Nucleic Acids Res.*, pp. 32(3):e34, 2004.

[12] Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. Altman, "Missing Value Estimation Methods for DNA Microarrays," *Bioinformatics*, vol. 17, pp. 520-525, 2001.

[13] M. S. B. Sehgal, I. Gondal, and L. Dooley, "Collateral Missing Value Estimation: Robust missing value estimation for consequent microarray data processing," *Lecture Notes in Artificial Intelligence (LNAI), Springer-Verlag*, pp. 274-283, 2005.

[14] P. Y. Chen and P. M. Popovich, *Correlation: Parametric and Nonparametric Measures*, 1st edition: SAGE Publications, 2002.

[15] A.-L. Boulesteix, "PLS Dimension Reduction for Classification with Microarray Data," *Statistical Applications in Genetics and Molecular Biology*, vol. 3, 2003.

[16] K. Y. Yeung, R. E. Bumgarner, and A. E. Raftery, "Bayesian Model Averaging: development of an improved multi-class, gene selection and classification tool for microarray data," *Bioinformatics*, vol. 21 no.10, pp. 2394-2402, 2005.

[17] X. Zhou, X. Wang, and E. R. Dougherty, "Gene Selection Using Logistic Regressions Based on AIC, BIC and MDL Criteria," *New Mathematics and Natural Computation*, vol. 1, pp. 129-145, 2005.

[18] M. S. B. Sehgal, I. Gondal, and L. Dooley, "Missing Values Imputation for DNA Microarray Data using Ranked Covariance Vectors," *The International Journal of Hybrid Intelligent Systems (IJHIS)*, vol. ISSN 1448-5869, 2005.

[19] Z. Sidak, P. K. Sen, and J. Hajek, *Theory of Rank Tests (Probability and Mathematical Statistics)*: Academic Press, 1999.

Informative MicroRNA Expression Patterns for Cancer Classification

Yun Zheng and Chee Keong Kwoh

Bioinformatics Research Center, School of Computer Engineering,
Nanyang Technology Univerity, Singapore 639798
pg04325488@ntu.edu.sg, asckkwoh@ntu.edu.sg

Abstract. Some non-coding small RNAs, known as microRNAs (miRNAs), have been shown to play important roles in gene regulation and various biological processes. The abnormal expression of some specific miRNA genes often results in the development of cancer. In this paper, we find discriminatory miRNA patterns for cancer classification from miRNA expression profiles. The experimental results show that the expression patterns from a small set of miRNAs are very accurate in prediction. Further, the experimental results also suggest that the expression patterns of these informative miRNAs are conserved in different vertebrates during the evolution process.

1 Introduction

More and more evidences show that miRNAs play important roles in gene regulation and various biological processes [1, 2, 3]. Some recent work has reported that the abnormal expression of some specific miRNA genes often results in the development of cancer [3, 4, 5].

Lu *et al.* [6] described a new bead-based flow cytometric technique to obtain miRNA expression profiles, which is used to capture the concentration levels of miRNAs in different tissues. The miRNAs show globally lower expression in cancer tissues than in normal tissues [6]. However, it is still not clear which miRNAs contribute more information to the normal/cancer distinction, since complex prediction models, like k-Nearest-Neighbors (kNN) [7] and Probabilistic Neural Networks (PNN) [8], are used in [6]. These models are black-boxes and very hard to understand.

In this paper, we aim at finding informative expression patterns of a small subset of miRNAs for cancer classification. We apply the Discrete Function Learning (DFL) algorithm [9] to the miRNA expression profiles in [6] to find the subset of miRNAs that shows strong distinction of expression levels in normal and tumor tissues.

In this study, we find that a small subset of miRNAs, common for different cancer types, is highly informative and discriminatory. The classifier, built from 75 human normal/tumor tissue samples, only contains these features and successfully obtains 100% accuracy when predicting 12 independent samples of mouse. This result suggests that the expression patterns of these informative miRNAs are conserved in different vertebrates.

We also compare the performances of the DFL algorithm to those from six other classification algorithms. The DFL algorithm obtains better or comparable prediction accuracies to those from the compared algorithms and to the result reported in the literature. However, it is worth mentioning that most of the compared methods, such as the Naive Bayes [10] and k-Nearest-Neighbors algorithm [7], use complex models that suffer the risk of overfitting the training data set.

The rest of the paper are organized as follows. In Section 2, we briefly review the DFL algorithm. In Section 3, we introduce the data sets and display the results. In Section 4, we summarize this paper.

2 Methods

In this section, we briefly review the our method. The details of the DFL algorithm are shown in supplementary Figure S1 to S3 [1] and in [9].

2.1 The Discrete Function Learning Algorithm

We will first introduce our notation. We use capital letters to represent discrete random variables, such as X and Y; lower case letters to represent an instance of the random variables, such as x and y; bold capital letters, like $\mathbf{X}$, to represent a vector; and lower case bold letters, like $\mathbf{x}$, to represent an instance of $\mathbf{X}$. The cardinality of $\mathbf{X}$ is represented with $|\mathbf{X}|$. In the remainder parts of this paper, we denote the attributes except the class attribute as a set of discrete random variables $\mathbf{V} = \{X_1, \ldots, X_n\}$, the class attribute as variable Y. In miRNA profiles, X_is stand for the expression levels of the miRNA genes.

The entropy of a discrete random variable or vector $\mathbf{X}$ is defined in terms of probability of observing a particular value $\mathbf{x}$ of $\mathbf{X}$ as [11]:

$$H(\mathbf{X}) = -\sum_{\mathbf{x}} P(\mathbf{X} = \mathbf{x}) log P(\mathbf{X} = \mathbf{x}). \tag{1}$$

The entropy is used to describe the diversity of $\mathbf{X}$. The more diverse a variable or vector is, the larger entropy it will have. Generally, vectors are more diverse than individual variables, hence have larger entropy. Hereafter, for the purpose of simplicity, we represent $P(\mathbf{X} = \mathbf{x})$ with $p(\mathbf{x})$, $P(Y = y)$ with $p(y)$, and so on. The mutual information between a vector $\mathbf{X}$ and Y is defined as [11]:

$$\begin{aligned} I(\mathbf{X}; Y) &= H(Y) - H(Y|\mathbf{X}) = H(\mathbf{X}) - H(\mathbf{X}|Y) \\ &= H(\mathbf{X}) + H(Y) - H(\mathbf{X}, Y) = \sum_{\mathbf{x}} \sum_{y} p(\mathbf{x}, y) log \frac{p(\mathbf{x}, y)}{p(\mathbf{x})p(y)}. \end{aligned} \tag{2}$$

[1] The supplements of this paper are available at http://www.ntu.edu.sg/home5/pg04325488/miRNA05.htm.

Mutual information is always non-negative and can be used to measure the relation between two variable, a variable and a vector (Equation 2), or two vectors. Basically, the stronger the relation between two variables, the larger the mutual information they will have. Zero mutual information means the two variables are independent or have no relation.

We restate a theorem about the relationship between the mutual information $I(\mathbf{X};Y)$ and the number of attributes in $\mathbf{X}$.

Theorem 1 (McEliece, 1977 [12], p. 26). *$I(\{\mathbf{X},Z\};Y) \geq I(\mathbf{X};Y)$, with equality if and only if $p(y|\mathbf{x}) = p(y|\mathbf{x},z)$ for all $(\mathbf{x},y,z)$ with $p(\mathbf{x},y,z) > 0$.*

In Theorem 1, it can be seen that $\{\mathbf{X},Z\}$ will contain more or equal information about Y as $\mathbf{X}$ does. To put it another way, the more variables, the more information is provided about another variable.

In [13], it is proved that if $H(Y|X) = 0$, then Y is a function of X. Since $I(X;Y) = H(X) - H(Y|X)$, it is immediate to obtain Theorem 2.

Theorem 2. *If the mutual information between $\mathbf{X}$ and Y is equal to the entropy of Y, i.e., $I(\mathbf{X};Y) = H(Y)$, then Y is a function of $\mathbf{X}$.*

The entropy $H(Y)$ represents the diversity of the variable Y. The mutual information $I(\mathbf{X};Y)$ represents the relation between vector $\mathbf{X}$ and Y. From this point of view, Theorem 2 actually says that the relation between vector $\mathbf{X}$ and Y is very strong, such that there is no more diversity for Y if $\mathbf{X}$ has been known, as shown in Figure 1 (a). In other words, the value of $\mathbf{X}$ can fully determine the value of Y. We call the subset of features that satisfy the criterion of Theorem 2 as the Essential Attributes (EAs), since they essentially determine the class attribute.

A classification problem is trying to learn or approximate a function, which takes the values of attributes (except the class attribute) in a new sample as input and output a categorical value indicating its class, from a given training data set. The goal of the training process is to obtain a function which makes the output value of this function be the class value of the new sample as accurately

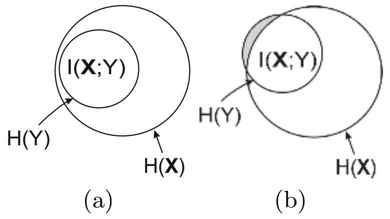


Fig. 1. The Venn diagram of $H(\mathbf{X})$,$H(Y)$ and $I(\mathbf{X},Y)$, when $Y = f(\mathbf{X})$. (a) The noiseless case, where the mutual information between $\mathbf{X}$ and Y is the entropy of Y. (b) The noisy case, where the entropy of Y is not equal to the mutual information between $\mathbf{X}$ and Y strictly. The shaded region is resulted from the noise. The ϵ value method means that if the area of the shaded region is smaller than or equal to $\epsilon \times H(Y)$, then the DFL algorithm will stop searching process, and build the function for Y with $\mathbf{X}$.

as possible. From Theorem 2, the problem is converted to finding a subset of attributes $\mathbf{U} \subseteq \mathbf{V}$ whose mutual information with Y is equal to the entropy of Y, where the $\mathbf{U}$ is the EAs which we are trying to find from the training data sets.

For n discrete variables, there are totally 2^n subsets. Clearly, it is NP-hard to examine all possible subsets exhaustively. It is often the case that there are some irrelevant and redundant features in the domain $\mathbf{V}$. Therefore, it is reasonable to reduce the searching space by considering those subsets with limited number of features. In the DFL algorithm, we introduce a parameter, called the expected cardinality of the EAs K, to restrict the searching space.

To efficiently find the EAs, the DFL algorithm performs a greedy search in the first round of its searching procedures (see supplementary Figure S1 to S3). $I(\mathbf{X};Y)$ is evaluated with respect to $H(Y)$ in the DFL algorithm, as shown in Equation 3. The DFL algorithm will first choose the feature $X_{(1)} = argmax_i I(X_i;Y)$. Next, suppose that $\mathbf{U}_{s-1}$ is the already selected feature subset, then the DFL algorithm will add a new feature $Z \in \mathbf{V} \setminus \mathbf{U}_{s-1}$ to $\mathbf{U}$ with

$$X_{(s)} = argmax_Z I(\{\mathbf{U}_{s-1}, Z\};Y), \tag{3}$$

where $\forall s$, $1 < s \leq k$, $\mathbf{U}_1 = \{X_{(1)}\}$, and $\mathbf{U}_s = \mathbf{U}_{s-1} \cup \{X_{(s)}\}$. From The chain rule of mutual information [13], we have $I(\{\mathbf{U}_{s-1}, Z\};Y) = I(\mathbf{U}_{s-1};Y) + I(Z;Y|\mathbf{U}_{s-1})$. Since $I(\mathbf{U}_{s-1};Y)$ does not change when trying different Z, the maximization of $I(\{\mathbf{U}_{s-1}, Z\};Y)$ in the DFL algorithm is actually maximizing $I(Z;Y|\mathbf{U}_{s-1})$, the conditional mutual information of Z and Y given the already selected features $\mathbf{U}_{s-1}$, i.e., the information of Y not captured by $\mathbf{U}_{s-1}$ but carried by Z. Hence, the redundancy of features carried by the new feature is minimized [9].

After the EAs $\mathbf{U}$ are found, the DFL algorithm will construct classifiers with $\mathbf{U}$. Firstly, the irrelevant features are deleted from training data set since they are non-essential attributes. Then, the duplicate instances of $\{\mathbf{U} \to Y\}$, i.e., $\{\mathbf{u} \to y\}$, are removed from the training data set to obtain the final classifier in form of truth table. In the meantime, the counts of different instances of $\{\mathbf{U} \to Y\}$ are also stored in the classifier and will be used in the prediction process.

After the DFL algorithm obtains the classifiers as function tables of the pairs $\{\mathbf{u} \to y\}$, or called as rules, the most reasonable way to use such function tables is to match the input values $\mathbf{u}$, then find the corresponding output values y. Therefore, we perform predictions in the *EA space*, with the 1-Nearest-Neighbor algorithm [7]. If a new sample has the same distance to more than one rule, then the rule with largest count value is chosen to predict the new sample.

2.2 The ϵ Value Method

Next, we reintroduce the ϵ value method. In Theorem 2, the exact functional relation demands the strict equality between $H(Y)$ and $I(\mathbf{X};Y)$. However, this equality is often ruined by the noisy data, like microarray gene expression data. In these cases, we have to relax the requirement to obtain a best estimated result. As shown in Fig. 1 (b), in the ϵ value method, if the difference between $I(\mathbf{X};Y)$

and $H(Y)$ is less than $\epsilon \times H(Y)$, then the DFL algorithm will stop the searching process, and will build the classifier for Y with $\mathbf{X}$ at the significant level ϵ.

2.3 The Selection of the Parameters

We discuss how to select the two parameters, the expected cardinality of the EAs K and the ϵ value, of the DFL algorithm in this section.

Selection of the Expected Cardinality K. Generally, if a data set has a large number of features, like several thousands, then K can be assigned to a small constant, like 20, since the models with large number of features will be very difficult to understand. If the number of features is small, then the K can be directly specified to the number of features n.

Another usage of K is to control model complexity. If the number of features is more important than accuracy, then a predefined K can be set. Thus, the learned model will have less than or equal to K features.

The expected cardinality K can also be used to incorporate the prior knowledge about the number of relevant features. If we have the prior knowledge about the number of relevant features, then the K can be specified as the predetermined value.

Selection of the ϵ Value. For a given noisy data set, the missing part of $H(Y)$, as demonstrated in Figure 1, is determined, i.e., there exists a specific minimum ϵ value, ϵ_m, with which the DFL algorithm can find the original model. If the ϵ value is smaller than the ϵ_m, the DFL algorithm will not find the original model. Here, we will introduce two methods to efficiently find the ϵ_m.

First, the ϵ_m can automatically be found by a restricted learning process. To efficiently find the ϵ_m, we restrict the maximum number of the subsets to be checked to $\sum_{i=0}^{K-1}(n-i) \sim K \times n$. The searching scope of ϵ is specified in prior. If the DFL algorithm can not find a model for a noisy data set with the specified minimum ϵ value, then the ϵ will be increased with a step of 0.01. The restricted learning will be performed, until the DFL algorithm finds a model with a threshold value of ϵ, i.e., the ϵ_m. Since only $K \times n$ subsets are checked in the restricted learning process, the time to find ϵ_m will be $O(K \cdot n)$.

The restricted learning process can also be used to find optimal model. To get optimal model, we change the ϵ value from 0 to the upper limit of the searching scope, like 0.8, with a step of 0.01. For each ϵ value, we validate the performances of the DFL algorithm with cross validation on the training data set. According to the principle of Occam's razor, simpler models are preferable to complex ones if they can produce the same or comparable prediction performances [14]. Hence, if the DFL algorithm reaches the same prediction accuracies with different ϵ values, then the larger ϵ value is chosen, since the model obtained with larger ϵ value contains fewer features based on Theorem 1.

Second, the ϵ_m can also be found with a manual binary search method (see supplementary Figure S4). Since $\epsilon \in [0, 1)$, ϵ is specified to 0.5 in the first try. If the DFL algorithm finds a model with ϵ value of 0.5, then ϵ is specified to 0.25 in the second try. Otherwise, if the DFL algorithm can not find a model with a long time, like 10 minutes, then the DFL algorithm can be stopped and ϵ is specified to 0.75 in the second try. The selection process is carried out until the ϵ_m value is found so that the DFL algorithm can find a model with it but cannot when $\epsilon = \epsilon_m - 0.01$. This selection process is also efficient. Since $\epsilon \in [0, 1)$, only 5 to 6 tries are needed to find the ϵ_m on the average.

3 Results

In this section, we will first introduce the miRNA data sets. Then, we show the experimental results. We implement the DFL algorithm with the Java language version 1.4.1. All experiments are performed on an HP *AlphaServer* SC computer, with one EV68 1GHz CPU and 1GB memory, running the *Tru64* Unix operating system. All the data sets and the software of the DFL algorithm are available at the supplementary website of this paper.

3.1 Data Sets

In [6], 75 miRNA expression profiles from human are used to build a normal/cancer classifier. Within these 75 sample, 32 sample are normal samples from 6 different tissues: colon, kidney, prostate, uterus, lung and breast. The remaining 43 samples are tumor samples from the same 6 different tissues.

We first randomly split this data set to a training:testing ratio of 50:25, which is the D1 data set (the D1 later) in Table 1. Then, in the data set D2 (the D2 later), we use the 75 samples as a training data set to build a classifier and to predict 12 testing samples from the mouse tissues. The 12 testing samples are from lung of mouse, and with the normal:tumor ratio of 5:7. In the data set D3 (the D3 later), we use 23 samples from 4 different tumor types to build a classifier and to predict an independent testing sample of 17 poorly differentiated tumors, the histological appearance of which was non-diagnostic, but for which clinical diagnosis was established by anatomical context, either directly (for example, a primary tumor arising in the colon) or indirectly (a metastasis of a previously identified primary tumor) [6].

Because the DFL algorithm is not designed for continuous data sets, we use a discretization algorithm from [15] to discretize the miRNA data sets. This method has been implemented by the *Weka*[2] software [16].

The discretization is carried out in such a way that the training data set is first discretized. Then the testing data set is discretized according to the cutting points of features (genes) determined in the training data set. After the discretization process, a substantial number of features, which are not contributing to the class distinction, are assigned with only one expression state and can be removed from the data sets. Meanwhile, the remaining discriminatory features are assigned with limited expression intervals, with the number shown in the column "D. Ftr. No." of Table 1.

[2] The *Weka* software, available at http://www.cs.waikato.ac.nz/~ml/weka/, is written with the Java language and is an open source software issued under the GNU General Public License.

Table 1. The miRNA data sets used in our experiments. The features are the expression level of different miRNAs. The columns Ftr. No., Cls. No., Train No., Test No., D. Ftr. No., K, ϵ, k and r are the number of features, the number of classes, the training sample size, testing sample size, and the number of features after applying the discretization pre-processing method [15], the expected cardinality of the DFL algorithm, the optimal ϵ value, the number of features chosen by the DFL algorithm and the number of rules in the DFL classifiers respectively.

Data	Ftr.	Cls.	Train	Test	D. Ftr.	Settings		Classifiers	
Set	No.*	No.	No.	No.	No.*	K	ϵ	k	r
D1	217	2	50	25	132	20	0.36	1	4
D2	217	2	75	12	148	20	0.14	2	5
D3	217	4	23	17	42	20	0.08	3	10

* The number does not include the class attribute.

3.2 Results

In all data sets, the expected cardinality of the DFL algorithm is set to 20. To obtain the optimal ϵ value, we perform the leave-one-out cross validation (LOOCV) on the training data sets for different ϵ values, from 0 to 0.8 with a step of 0.01. Then, we choose the optimal ϵ values, ϵ_{op}, with which the DFL obtains best prediction performances in the LOOCV processes, as shown in Figure 2. For the D1 and D2, we also perform the training testing validation for various ϵ values, we find that the DFL algorithm reaches its best performances for the testing data sets with the ϵ_{op} chosen in the LOOCV processes. In our implementation, this searching process to find optimal ϵ value can be done automatically. For the D3, we only perform training testing validation, since the training sample size is too small. The optimal settings and brief information of the obtained classifiers of the DFL algorithm are shown in Table 1.

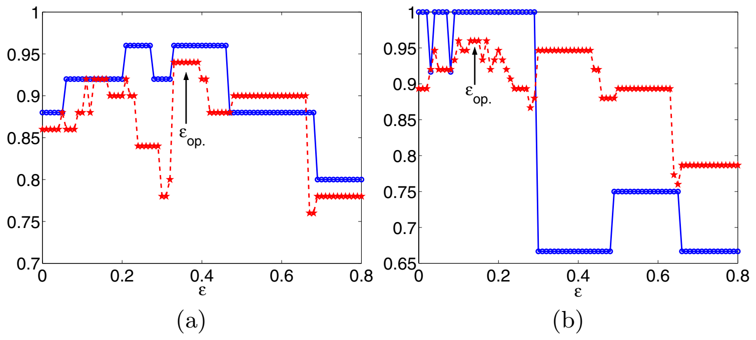


Fig. 2. The ϵ vs accuracies of the DFL algorithm. The curves marked with circles and pentagrams are for the training/testing and the LOOCV respectively. In both data sets, the expected cardinality of the DFL algorithm is set to 20. The $\epsilon_{op.}$ pointed by an arrow is the optimal value that we choose. (a) For the data set D1. (b) For the data set D2.

Table 2. The miRNAs chosen as EAs by the DFL algorithm, with the settings listed in Table 1

Data Set	miRNAs	Sequences
D1	*hsa-miR-16*	UAGCAGCACGUAAAUAUUGGCG
D2	*hsa-miR-26a*	UUCAAGUAAUCCAGGAUAGGCU
	hsa-miR-16	UAGCAGCACGUAAAUAUUGGCG
D3	*hsa-miR-7*	UGGAAGACUAGUGAUUUUGUU
	hsa-miR-130a	CAGUGCAAUGUUAAAAGGGC
	hsa-miR-135b	UAUGGCUUUUCAUUCCUAUGUG

Table 3. The summary of prediction errors made by different algorithms

Data Set	DFL	C4.5	NB	1NN	*k*NN*	SVM	RIP
D1 (Discrete)	1	2	3	3	3	2	2
D1 (Continuous)	-	2	3	0	2	2	1
D2 (Discrete)	0	0	2	1	0	1	0
D2 (Continuous)	-	4	3	1	0	1	0
D3 (Discrete)	7	10	5	4	4	6	10
D3 (Continuous)	-	8	11	9	7	7	11

* The k of the kNN algorithm is set to 5.

As shown in Figure 2, the DFL algorithm finds the optimal model with ϵ value of 0.36 and 0.14 for the data set D1 and D2 respectively. The DFL algorithm makes 1, 0 and 7 prediction errors (details available at supplementary Table S2) for the D1, D2 and D3 data set respectively, as shown in Table 3. With these settings, the DFL algorithm chooses the miRNAs listed in Table 2 as EAs.

We use the *Weka* software (version 3.4) to evaluate the performance of other classification methods. Specifically, we compare the DFL algorithm to the C4.5 algorithm by Quinlan [17], the Naive Bayes (NB) algorithm described by Langley *et al.* [10], the 1NN and k-Nearest-Neighbors (kNN) algorithm by Aha *et al.* [7], the Support Vector Machines (SVM) algorithm by Platt [18] and the Ripper algorithm (RIP) by Cohen [19]. All these methods are implemented in the *Weka* software. The prediction errors of all compared algorithms are listed in Table 3, where the values for all algorithms are the average of 10 runs.

As shown in Table 3, the DFL algorithm makes only one prediction error in the D1, which is better than most other compared algorithms. In the D2, the DFL algorithm correctly classifies the 12 testing samples. In the D3, the DFL algorithm makes a slightly more prediction errors than the NB, 1NN, kNN and SVM algorithm in the discretized data sets. But the DFL algorithm is still one of the best if compared to other algorithms for the continuous D3 data set.

Accuracy is only one aspect of the quality of the classification models. To comprehensively compare classifiers, the comparisons of model complexity and training time are also important. Next, we compare the models from different

Table 4. The comparison of models from the DFL, C4.5 and RIP algorithms. The k and r are the number of features and the number of rules in the classifiers.

Data Set	DFL		C4.5		RIP	
	k	r	k	r^*	k	r
D1 (Discrete)	1	4	2	5	2	2
D1 (Continuous)	-	-	2	5	2	2
D2 (Discrete)	2	5	2	5	2	2
D2 (Continuous)	-	-	2	5	2	2
D3 (Discrete)	3	10	3	7	3	4
D3 (Continuous)	-	-	3	7	3	4

* The value shown is the number of nodes in the C4.5 tree.

Table 5. The comparison of training time for different classification algorithms. The values are the training times for the discretized data sets and shown in second.

	DFL	C4.5	NB	1NN	kNN	SVM	RIP
D1	0.09	0.09	0.02	0.05	0.05	0.16	0.12
D2	0.10	0.09	0.02	0.07	0.07	0.18	0.13
D3	0.09	0.09	0.02	0.03	0.03	0.32	0.02
average	0.09	0.09	0.02	0.05	0.05	0.22	0.09

algorithms in Table 4. We only compare the models of the DFL, C4.5 and RIP algorithms. The NB, 1NN, kNN and SVM algorithms build very complex models, using all features of the data sets. The complex models from these algorithms make it difficult for the users to understand which set of miRNAs is really important in contributing to the class distinctions between samples. As shown in Table 4, the complexity of models from the DFL, C4.5 and RIP algorithms are comparable. But, the DFL obtains better prediction performances than the C4.5 and RIP algorithms, as shown in Table 3.

It is interesting to point out that for the D2, the RIP algorithm chooses the same features as those chosen by the DFL algorithm. These two algorithms both correctly classify all the samples in the testing data set, which suggests that the two features, *hsa-miR-26a* (EAM263) and *hsa-miR-16* (EAM115), are critical for differentiating the normal and cancer tissues.

Finally, we compare the training time of different classification algorithm in Table 5. Since all compared algorithms are implemented with the Java language and all experiments are performed on the same computer, the comparisons of their efficiency are meaningful. As shown in Table 5, the DFL algorithm is less efficient than the NB, 1NN and kNN algorithms, but more efficient than the SVM algorithm, and uses almost the same time the C4.5 and RIP algorithms. The training processes of the 1NN and kNN algorithms are efficient, but these two algorithms use much more time in prediction.

In summary, the DFL algorithm obtains comparable prediction performances to other compared methods, but uses more compact models.

We also compare the results of the DFL algorithm to those reported in the literature. Lu *et al.* [6] performed a top-ranking feature selection based on *t*-statistic, and chose 187 features. Because the top-ranking feature selection methods include many redundant features which carry similar information about the class attribute [9], the *k*NN classifier in [6] contained more features than the model learned by the DFL algorithm. The *k*NN classifier in [6] also correctly predicted the 12 testing samples in the D2. In summary, the prediction performance of the DFL algorithm is comparable to the method in [6], but the DFL classifier is more concise than the model in [6].

3.3 Detailed Analysis

In the D2, we find that two miRNAs, *hsa-miR-26a* (EAM263) and *hsa-miR-16* (EAM115), are very informative. In the D2, The mutual information between these two genes and the class attribute is 0.7 bits, contributing to 71% of the

Table 6. The DFL classifier for the data set D2

hsa-miR-16	*hsa-miR-26a*	Class	Count
(11.5171-∞)	(11.49575-∞)	NORMAL	31
(-∞-11.5171]	(-∞-11.49575]	NORMAL	1
(-∞-11.5171]	(-∞-11.49575]	TUMOR	37
(-∞-11.5171]	(11.49575-∞)	TUMOR	3
(11.5171-∞)	(-∞-11.49575]	TUMOR	3

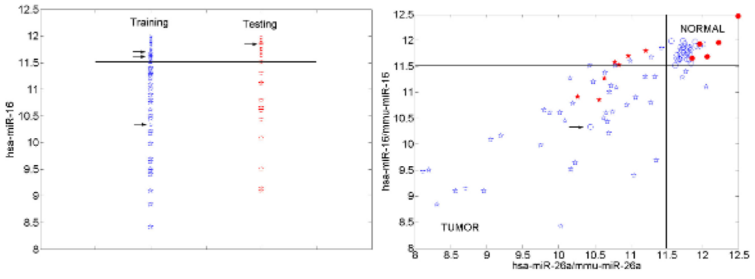


Fig. 3. The expression values of the miRNAs chosen by the DFL algorithm in the D1 and D2. Normal and tumor samples are represented with circles, and pentagrams respectively. In part (b), hollow and solid samples are from training and testing data sets respectively. The black solid lines are the cutting points of the genes introduced in the discretization preprocessing. (a) The expression values of *hsa-miR-16* in the D1. The samples pointed by arrows are incorrect predictions, where the left side is for the LOOCV in the training data set and the right side is for the training/testing validation. (b) The expression values of *hsa-miR-26a* and *hsa-miR-16* in the data set D2, whose homologue genes of mouse are *mmu-miR-26a* and *mmu-miR-16*. The sample pointed by an arrow is an incorrect prediction in the LOOCV.

diversity/entropy of the class attribute, which is 0.98 bits. When considered as a vector, they capture 91% of the total diversity of the class attribute. The classifier built with the DFL algorithm for the D2 is shown in Table 6. For instance, the first rule in Table 6 means that if the expression level of *hsa-miR-16* is larger than 11.5171 and the expression level of *hsa-miR-26a* is larger than 11.49575 in a sample, then the corresponding class value of this sample is NORMAL. And this rule has happened for 31 times out of the 75 samples in the training data set.

In Figure 3, we further investigate the expression patterns of the miRNAs chosen by the DFL algorithm for the D1 and D2. From Figure 3 (a), it is shown that the DFL algorithm makes 3 and 1 prediction errors in the LOOCV and training/testing validation of the D1 respectively. In Figure 3 (b), it is shown that the classifier in Table 6 correctly predicts the 12 samples from mouse tissues. As shown in Figure 3 (b), *hsa-miR-26a* and *hsa-miR-16* have higher expression levels in normal tissues than in tumor tissues, since most normal samples are located in the upper-right region of the space defined by the two miRNAs, *hsa-miR-26a* and *hsa-miR-16*. There is one outlier sample, pointed by an arrow, which is an incorrect prediction during the LOOCV. On the other hand, most tumor samples are located in the lower-left region of the space defined by the two miRNAs, *hsa-miR-26a* and *hsa-miR-16*, which means that the two miRNAs generally have lower expression levels in tumor tissues than in normal tissues.

4 Discussion

We have demonstrated that there exist simple and informative miRNA expression patterns, which can be used to classify normal against tumor tissues. These patterns obtain better or comparable prediction performances to the models learned from other classification methods compared in this paper and results reported in the literature.

The miRNAs in the patterns learned with the DFL algorithm generally show higher and lower expression levels in the normal and tumor tissues respectively, which is consistent with the results in the literature [6, 20].

It has been reported that the many miRNAs are phylogenetically conserved [1]. In this research, we find that the expression patterns of the two miRNAs, *hsa-miR-26a* and *hsa-miR-16*, built with human miRNA expression profiles can accurately predict the miRNA normal/tumor samples from mouse. This suggests that the expression patterns of these miRNAs are also highly conserved in vertebrates during the evolution process.

References

1. Ambros, V.: The functions of animal microRNAs. Nature **431** (2004) 350–5
2. Bartel, D.P.: MicroRNAs: Genomics, biogenesis, mechanism, and function. Cell **116** (2004) 281–297
3. Alvarez-Garcia, I., Miska, E.A.: MicroRNA functions in animal development and human disease. Development **132** (2005) 4653–62

4. Gregory, R.I., Shiekhattar, R.: MicroRNA Biogenesis and Cancer. Cancer Res **65**(9) (2005) 3509–3512
5. He, L., Thomson, J., Hemann, M., Hernando-Monge, E., Mu, D., Goodson, S., Powers, S., Cordon-Cardo, C., Lowe, S., Hannon, G., Hammond, S.: A MicroRNA polycistron as a potential human oncogene. Nature **435** (2005) 828–33
6. Lu, J., Getz, G., Miska, E.A., Alvarez-Saavedra, E., Lamb, J., Peck, D., Sweet-Cordero, A., Ebert, B.L., Mak, R.H., Ferrando, A.A., Downing, J.R., Jacks, T., Horvitz, H.R., Golub, T.R.: MicroRNA expression profiles classify human cancers. Nature **435** (2005) 834–8
7. Aha, D., Kibler, D., Albert, M.: Instance-based learning algorithms. Machine Learning **6** (1991) 37–66
8. Specht, D.F.: Probabilistic neural networks. Neural Networks **3** (1990) 109–118
9. Zheng, Y., Kwoh, C.K.: Identifying simple discriminatory gene vectors with an information theory approach. In: Proceedings of the 4th Computational Systems Bioinformatics Conference, CSB 2005, IEEE Computer Society Press (2005) 12–23
10. Langley, P., Iba, W., Thompson, K.: An analysis of bayesian classifiers. In: National Conference on Artificial Intelligence. (1992) 223–228
11. Shannon, C., Weaver, W.: The Mathematical Theory of Communication. University of Illinois Press, Urbana, IL. (1963)
12. McEliece, R.: The Theory of Information and Coding: A Mathematical Framework for Communication. Volume 3 of Encyclopedia of Mathematics and Its Applications. Addison-Wesley Publishing Company, Reading, MA. (1977)
13. Cover, T., Thomas, J.: Elements of Information Theory. John Wiley & Sons, Inc. (1991)
14. Cherkassky, V., Mulier, F.: Learning from Data: Concepts, Theory, and Methods. John Wiley & Sons, Inc., New York, NY (1998)
15. Fayyad, U., Irani, K.: Multi-interval discretization of continuous-valued attributes for classification learning. In: Proceedings of the 13th International Joint Conference on Artificial Intelligence, IJCAI-93, Chambery, France (1993) 1022–1027
16. Frank, E., Hall, M., Trigg, L., Holmes, G., Witten, I.: Data mining in bioinformatics using Weka. Bioinformatics **20**(15) (2004) 2479–2481
17. Quinlan, J.: C4.5: Programs for machine learning. Morgan Kaufmann (1993)
18. Platt, J.: Fast training of support vector machines using sequential minimal optimization. In: Advances in kernel methods: support vector learning. MIT Press (1999) 185–208
19. Cohen, W.W.: Fast effective rule induction. In: Proc. 12th International Conference on Machine Learning, Morgan Kaufmann (1995) 115–123
20. Meltzer, P.S.: Cancer genomics: Small RNAs with big impacts. Nature **435** (2005) 745–6

Author Index