Joab Winkler
Mahesan Niranjan
Neil Lawrence (Eds.)

LNAI 3635

Deterministic and Statistical Methods in Machine Learning

First International Workshop
Sheffield, UK, September 2004
Revised Lectures

Springer

Lecture Notes in Artificial Intelligence 3635

Edited by J. G. Carbonell and J. Siekmann

Subseries of Lecture Notes in Computer Science

Joab Winkler Mahesan Niranjan
Neil Lawrence (Eds.)

Deterministic and Statistical Methods in Machine Learning

First International Workshop
Sheffield, UK, September 7-10, 2004
Revised Lectures

Springer

Series Editors

Jaime G. Carbonell, Carnegie Mellon University, Pittsburgh, PA, USA
Jörg Siekmann, University of Saarland, Saarbrücken, Germany

Volume Editors

Joab Winkler
Mahesan Niranjan
Neil Lawrence
The University of Sheffield, Department of Computer Science
Regent Court, 211 Portobello Street, Sheffield S1 4DP, UK
E-mail: Winkler@dcs.shef.ac.uk
M.Niranjan@sheffield.ac.uk
neil@dcs.shef.ac.uk

Library of Congress Control Number: 2005933155

CR Subject Classification (1998): I.2, F.2.2, I.5, I.4, F.4.1, H.3

ISSN 0302-9743
ISBN-10 3-540-29073-7 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-29073-5 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11559887 06/3142 5 4 3 2 1 0

Preface

Machine learning is a rapidly maturing field that aims to provide practical methods for data discovery, categorization and modelling. The Sheffield Machine Learning Workshop, which was held 7–10 September 2004, brought together some of the leading international researchers in the field for a series of talks and posters that represented new developments in machine learning and numerical methods.

The workshop was sponsored by the Engineering and Physical Sciences Research Council (EPSRC) and the London Mathematical Society (LMS) through the MathFIT program, whose aim is the encouragement of new interdisciplinary research. Additional funding was provided by the PASCAL European Framework 6 Network of Excellence and the University of Sheffield. It was the commitment of these funding bodies that enabled the workshop to have a strong program of invited speakers, and the organizers wish to thank these funding bodies for their financial support. The particular focus for interactions at the workshop was *Advanced Research Methods in Machine Learning and Statistical Signal Processing.*

These proceedings contain work that was presented at the workshop, and ideas that were developed through, or inspired by, attendance at the workshop. The proceedings reflect this mixture and illustrate the diversity of applications and theoretical work in machine learning.

We would like to thank the presenters and attendees at the workshop for the excellent quality of presentation and discussion during the oral and poster sessions. We are also grateful to Gillian Callaghan for her support in the organization of the workshop, and finally we wish to thank the anonymous reviewers for their help in compiling the proceedings.

July 2005

Joab Winkler
Neil Lawrence
Mahesan Niranjan

Table of Contents

Object Recognition via Local Patch Labelling

Christopher M. Bishop[1] and Ilkay Ulusoy[2]

[1] Microsoft Research,
7 J J Thompson Avenue,
Cambridge, UK
http://research.microsoft.com/∼cmbishop
[2] METU, Computer Vision and Intelligent Systems Research Lab.,
06531 Ankara, Turkey
http://www.eee.metu.edu.tr/∼ilkay

Abstract. In recent years the problem of object recognition has received considerable attention from both the machine learning and computer vision communities. The key challenge of this problem is to be able to recognize any member of a category of objects in spite of wide variations in visual appearance due to variations in the form and colour of the object, occlusions, geometrical transformations (such as scaling and rotation), changes in illumination, and potentially non-rigid deformations of the object itself. In this paper we focus on the detection of objects within images by combining information from a large number of small regions, or 'patches', of the image. Since detailed hand-segmentation and labelling of images is very labour intensive, we make use of 'weakly labelled' data in which the training images are labelled only according to the presence or absence of each category of object. A major challenge presented by this problem is that the foreground object is accompanied by widely varying background clutter, and the system must learn to distinguish the foreground from the background without the aid of labelled data. In this paper we first show that patches which are highly relevant for the object discrimination problem can be selected automatically from a large dictionary of candidate patches during learning, and that this leads to improved classification compared to direct use of the full dictionary. We then explore alternative techniques which are able to provide labels for the individual patches, as well as for the image as a whole, so that each patch is identified as belonging to one of the object categories or to the background class. This provides a rough indication of the location of the object or objects within the image. Again these individual patch labels must be learned on the basis only of overall image class labels. We develop two such approaches, one discriminative and one generative, and compare their performance both in terms of patch labelling and image labelling. Our results show that good classification performance can be obtained on challenging data sets using only weak training labels, and they also highlight some of the relative merits of discriminative and generative approaches.

1 Introduction

The problem of object recognition has emerged as a 'grand challenge' for computer vision, with the longer term aim of being able to achieve near human levels of recognition for tens of thousands of object categories under a wide variety of conditions. Many of

J. Winkler, N. Lawrence, M. Niranjan (Eds.): Machine Learning Workshop, LNAI 3635, pp. 1–21, 2005.

the current approaches to this problem rely on the use of local features obtained from small patches of the image. The motivation for this is that the variability of small patches is much less than that of whole images and so there are much better prospects for generalization, in other words for recognizing that a patch from a test image is similar to patches in the training images. However, the patches must be sufficiently variable, and therefore sufficiently large, to be able to discriminate between the different object categories and also between objects and background clutter. A good way to balance these two conflicting requirements is to determine the object categories present in an image by fusing together partial ambiguous information from multiple patches. Probability theory provides a powerful framework for combining such uncertain information in a principled manner, and will form the basis for our research (the specific local features that we use in this paper are described in Section 2.) Also, the locations of those patches which provide strong evidence for an object also give an indication of the location and spatial extent of that object.

In common with a number of previous approaches, we do not attempt to model the spatial relationship between patches. Although such spatial information is certainly very relevant to the object recognition problem, and its inclusion would be expected to improved recognition performance for many object categories, its role is complementary to that of the texture-like evidence provided by local patches. Here we show that local information alone can already give good discriminatory results.

A key issue in object recognition is the need for predictions to be invariant to a wide variety of transformations of the input image due to translations and rotations of the object in 3D space, changes in viewing direction and distance, variations in the intensity and nature of the illumination, and non-rigid transformations of the object. Although the informative features used in [13] are shown to be superior to generic features when used with a simple classification method, they are not invariant to scale and orientation. By contrast, generic interest point operators such as saliency [6], DoG [7] and Harris-Laplace [9] detectors are repeatable in the sense that they are invariant to location, scale and orientation, and some are also affine invariant [7,9] to some extent. For the purposes of this paper we shall consider the use of invariant features obtained from local regions of the image centered on interest points.

Fergus et al. [5] learn jointly the appearances and relative locations of a small set of parts whose potential locations are determined by a saliency detector [6]. Since their algorithm is very complex, the number of parts has to be kept small and the type of detector they used is appropriate for this purpose. Csurka *et al*. [3] used Harris-Laplace interest point operators [9] with SIFT features [7] for the purpose of multi class object category recognition. Features are clustered using K-Means and each feature is labelled according to the closest cluster centre. Histograms of feature labels are then used as class-conditional densities. Since such interest point operators detect many points from the background as well as from the object itself, the features are used collectively to determine the object category, and no information on object localization is obtained. In [4], informative features were selected based on information criteria such as likelihood ratio and mutual information in which DoG and Harris-Laplace interest point detectors with SIFT descriptors were compared. However, in this supervised approach, hundreds of images were hand segmented in order to train support vector machine and Gaussian

mixture models (GMMs) for foreground/background classification. The two detectors gave similar results although DoG produces more features from the background. Finally, Xie and Perez [14] extended the GMM based approach of [4] to a semi-supervised case inspired from [5]. A multi-modal GMM was trained to model foreground and background features where some uncluttered images of foreground were used for the purpose of initialization.

In this paper we develop several new approaches to object recognition based on features extracted from local patches centered on interest points. We begin, in Section 3, by extending the model of [3] which constructs a large dictionary of candidate feature 'prototypes'. By using the technique of *automatic relevance determination*, our approach can learn which of these prototypes are particularly salient for the problem of discriminating object classes and can thereby give appropriately less emphasis to those which carry little discriminatory information (such as those associated with background clutter). This leads to a significant improvement in classification performance.

While this approach allows the system to focus on the foreground objects, it does not directly lead to a labelling of the individual patches. We therefore develop new probabilistic approaches to object recognition based on local patches in which the system learns not only to classify the overall image, but also to assign labels to patches themselves. In particular, we develop two complementary approaches one of which is discriminative (Section 4) and one of which is generative (Section 5).

To understand the distinction between discriminative and generative, consider a scenario in which an image described by a vector $\mathbf{X}$ (which might comprise raw pixel intensities, or some set of features extracted from the image) is to be assigned to one of K classes $k = 1, \ldots, K$. From basic decision theory [2] we know that the most complete characterization of the solution is expressed in terms of the set of posterior probabilities $p(k|\mathbf{X})$. Once we know these probabilities it is straightforward to assign the image $\mathbf{X}$ to a particular class to minimize the expected loss (for instance, if we wish to minimize the number of misclassifications we assign $\mathbf{X}$ to the class having the largest posterior probability).

In a discriminative approach we introduce a parametric model for the posterior probabilities, and infer the values of the parameters from a set of labelled training data. This may be done by making point estimates of the parameters using maximum likelihood, or by computing distributions over the parameters in a Bayesian setting (for example by using variational inference).

By contrast, in a generative approach we model the joint distribution $p(k, \mathbf{X})$ of images and labels. This can be done, for instance, by learning the class prior probabilities $p(k)$ and the class-conditional densities $p(\mathbf{X}|k)$ separately. The required posterior probabilities are then obtained using Bayes' theorem

$$p(k|\mathbf{X}) = \frac{p(\mathbf{X}|k)p(k)}{\sum_j p(\mathbf{X}|j)p(j)} \tag{1}$$

where the sum in the denominator is taken over all classes.

Comparative results from the various approaches are presented in Section 6. These show that the generative approach gives excellent classification performance both for individual patches and for the complete images, but that careful initialization of the

training procedure is required. By contrast the discriminative approach, which gives good results for image labelling but not for patch labelling, is significantly faster in processing test images. Ideas for future work, including techniques for combining the benefits of generative and discriminative approaches, are discussed briefly in Section 7.

2 Local Feature Extraction

Our goal in this paper is not to find optimal features and representations for solving a specific object recognition task, but rather to fix on a particular, widely used, feature set and use this as the basis to compare alternative learning methodologies. We shall also fix on a specific data set, chosen for the wide variability of the objects in order to present a non-trivial classification problem. In particular, we consider the task of detecting and distinguishing cows and sheep in natural images.

We therefore follow several recent approaches [7,9] and use an interest point detector to focus attention on a small number of local patches in each image. This is followed by invariant feature extraction from a neighbourhood around each interest point. Specifically we use DoG interest point detectors, and at each interest point we extract a 128 dimensional SIFT feature vector [7] from a patch whose scale is determined by the DoG detector. Following [1] we concatenate the SIFT features with additional colour features comprising average and standard deviation of (R, G, B), (L, a, b) and $(r = R/(R+G+B), g = G/(R+G+B))$, which gives an overall 144 dimensional feature vector. The result of applying the DoG operator to a cow image is shown in Figure 1.

In this paper we use $\mathbf{t}_n$ to denote the image label vector for image n with independent components $t_{nk} \in \{0, 1\}$ in which $k = 1, \ldots K$ labels the class. Each class can be present or absent independently in an image, and we make no distinction between foreground and background classes within the model itself. $\mathbf{X}_n$ denotes the observation for image n and this comprises as set of J_n patch vectors $\{\mathbf{x}_{nj}\}$ where $j = 1, \ldots, J_n$. Note that the number J_n of detected interest points will in general vary from image to image.

On a small-scale problem it is reasonable to segment and label the objects present in the training images. However, for large-scale object recognition involving thousands of categories this will not be feasible, and so instead it is necessary to employ training data which is at best 'weakly labelled'. Here we consider a training set in which each image is labelled only according to the presence or absence of each category of object (in our example each image contains either cows or sheep).

3 Patch Saliency Using Automatic Relevance Determination

We begin by considering a simple approach based on [3]. In this method the features extracted from all of the training images are clustered into C classes using the K-means algorithm, after which each patch in each image is assigned to the closest prototype. Each image n is therefore described by a fixed-length histogram feature vector $\mathbf{h}_n$ of length C in which element h_{nc} represents the number of patches in image n which are assigned to cluster c, where $c \in \{1, \ldots, C\}$ and $n \in \{1, \ldots, N\}$. These feature vectors are then used to construct a classifier which takes an image $\mathbf{X}_n$ as input, converts

Fig. 1. Difference of Gaussian interest points with their local regions, in which the squares are centered at the interest points and the size of the squares indicates the scale of the interest points. The SIFT descriptors and colour features are obtained from these square patches Note that interest points fall both on the objects of interest (the cows) and also on the background.

it to a feature vector $\mathbf{h}_n$ and then assigns this vector to an object category. Here the assumption is that each image belongs to one and only one of some number K of mutually exclusive classes. In [3] the classifier was based either on naive Bayes or on support vector machines.

Here we use a linear softmax model since this can be readily extended to determine feature saliency as discussed shortly. Thus the model computes a set of outputs given by

$$y_k(\mathbf{h}_n, \mathbf{w}) = \frac{\exp(\mathbf{w}_k^{\mathrm{T}}\mathbf{h}_n)}{\sum_l \exp(\mathbf{w}_l^{\mathrm{T}}\mathbf{h}_n)} \tag{2}$$

where $k \in \{1, \ldots, K\}$. Here the quantity $y_k(\mathbf{h}_n, \mathbf{w})$ which can be interpreted as the posterior probability that image vector $\mathbf{h}_n$ belongs to class k. The parameter vector $\mathbf{w} = \{\mathbf{w}_k\}$ is found by maximum likelihood using iterative re-weighted least squares [10]. We shall refer to this approach as VQ-S for vector quantized softmax. Results from this method will be presented in Section 6.

An obvious problem with this approach is that the patches which contribute to the feature vector come from both the foreground object(s) and also from the background. Changes to the background cause changes in the feature vector even if the foreground object is the same. Furthermore, some foreground patches might occur on objects from different classes, and are therefore provide relatively little discriminatory information compared to other patches which are more closely associated with particular object categories.

We can address this problem using the Bayesian technique of *automatic relevance determination* or *ARD* [8]. This involves the introduction of a prior distribution over the parameter vector $\mathbf{w}$ in which each input variable h_c has a separate hyperparameter α_c corresponding to the inverse variance (or precision) of the prior distribution of the weights $\mathbf{w}_c$ associated with that input, so that

$$p(\mathbf{w}|\boldsymbol{\alpha}) = \prod_{c=1}^{C} \mathcal{N}(\mathbf{w}_c|\mathbf{0}, \alpha_c^{-1}\mathbf{I}). \tag{3}$$

During learning the hyperparameters are updated by maximizing the marginal likelihood, i.e. the probability of the training labels D given $\boldsymbol{\alpha}$ in which $\mathbf{w}$ has been integrated out, given by

$$p(D|\alpha) = \int p(D|\mathbf{w})p(\mathbf{w})\, d\mathbf{w}. \tag{4}$$

This is known as the *evidence procedure* and the values of the hyperparameters found at convergence express the relative importance of the input variables in determining the image class label. Specifically, the hyperparameters represent the inverse variances of the weights, and so a large value of α_c implies that the corresponding parameter vector $\mathbf{w}_c$ has a distribution which is concentrated around zero and so the associated input variable h_c has little effect in determining the output values y_k. Such inputs have low relevance. By contrast a high value of α_c corresponds to an input h_c whose value plays an important role in determining the class label. The inclusion of ARD leads to an improvement in classification performance, as discussed in Section 6. We shall refer to this model as VQ-ARD.

With this approach we can rank the patch clusters according to their relevance. The logarithm of the inverse of the hyperparameter α_c is sorted and plotted in Figure 2.

Equivalently this can be plotted as a histogram of α_c values, as shown in Figure 3. It is interesting to note that in this problem the hyperparameter values form two groups in which one group can loosely be considered as relevant and the other as not relevant, so far as the discrimination task is concerned.

Figure 4 shows the properties of the most relevant cluster and of the least relevant cluster, as well as that of an intermediate cluster, according to the ARD analysis based on $C = 100$ cluster centers. Note that the images have been hand segmented in order to

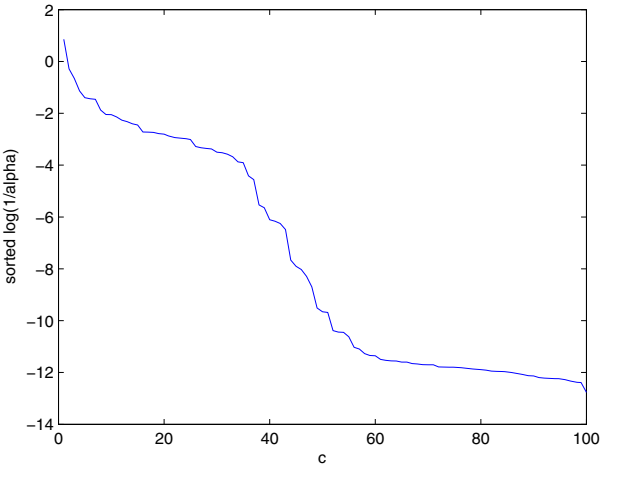


Fig. 2. The sorted values of the log variance (inverse of the hyperparameter α).

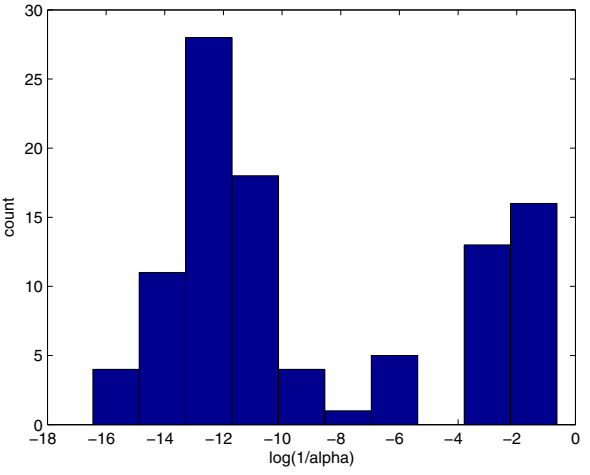


Fig. 3. The histogram of the log variances

identify the foreground region. This segmentation is used purely for test purposes and plays no role during training. The top row shows the features belonging to the worst cluster, i.e. ranked 100, on a sheep image and on a cow image. This feature exists in both classes and thus provides a little information to make a classification. The middle row shows the locations of patches assigned to the cluster which is ranked 27, in which we see that all of the patches belong to the background. Finally, the bottom row of the figure shows the features belonging to the most relevant cluster, ranked 1, on the same sheep and cow images. This feature is not observed on the sheep image but there are several patches assigned to this cluster on the cow image. Thus the detection of this feature is a good indicator of the presence of a cow.

It is also interesting to explore the behaviour of the two groups of clusters corresponding to the two modes in the distribution of hyper-parameter values shown in Figure 3. Figure 5 shows examples of cow and sheep images in each case showing the locations of the clusters associated with the two modes.

Although this approach is able to focuss attention on foreground regions, we have seen that not all foreground patches have high saliency, and so this approach cannot reliably identify regions occupied by the foreground objects. We therefore turn to the development of new models in which we explicitly consider the identity of individual patches and not simply their saliency for overall image classification. In particular the hard quantization of K-means is abandoned in favour of more probabilistic approaches. First we discuss a discriminative model and then we turn to a complementary generative model.

4 The Discriminative Model with Patch Labelling

Since our goal is to determine the class membership of individual patches, we associate with each patch j in an image n a binary label $\tau_{njk} \in \{0, 1\}$ denoting the class k of

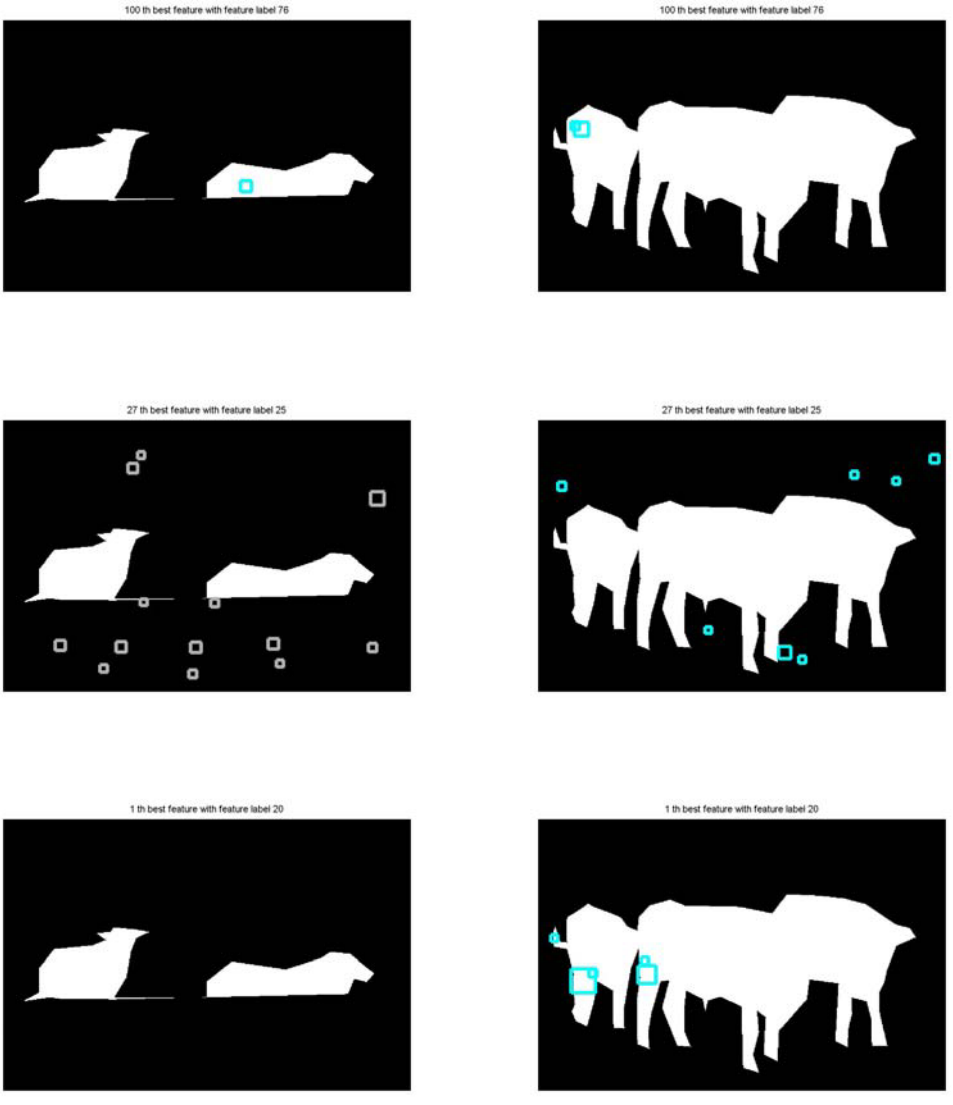


Fig. 4. The top row shows example cow and sheep images, with the foreground regions segmented, together with the locations of patches assigned to the least relevant (ranked 100) cluster center. Similarly the middle row analogous results for a cluster of intermediate relevance (ranked 27) and the bottom row shows the cluster assignments for the most relevant cluster (ranked 1). The centers of the squares are the locations of the patches from which the features are obtained and the size of the squares show the scale of the patches.

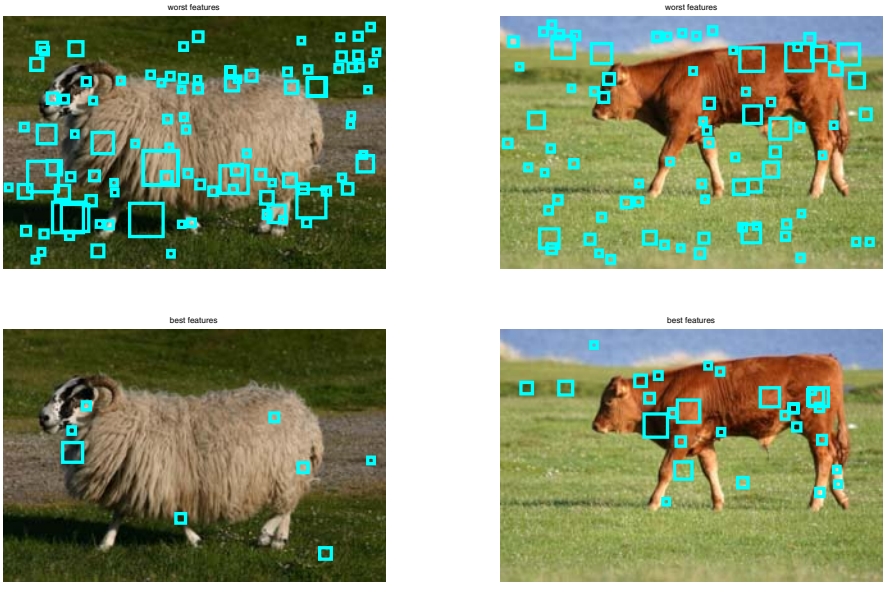


Fig. 5. Illustration of the behaviour of the two modes in the histogram of hyper-parameter values seen in Figure 5. The left column shows a typical example from the sheep class while the right column shows a typical example from the cow class. In the top row the squares denote the locations of interest points assigned to clusters in the left hand mode of the histogram corresponding to low relevance clusters, while the bottom row gives the analogous results to the high relevance model. The threshold between high and low was set by eye to $\ln(1/\alpha) = -5$. Note that the high relevance clusters are associated predominantly with the foreground, while the low relevance ones occur on both the foreground and the background.

the patch. For the models developed in this paper we shall consider these labels to be mutually exclusive, so that $\sum_{k=1}^{K} \tau_{njk} = 1$, in other words each patch is assumed to be either cow, sheep or background. Note that this assumption is not essential, and other formulations could also be considered. These components can be grouped together into vectors $\boldsymbol{\tau}_{nj}$. If the values of these labels were available during training (corresponding to strongly labelled images) then the development of recognition models would be greatly simplified. For weakly labelled data, however, the $\{\boldsymbol{\tau}_{nj}\}$ labels are hidden (latent) variables, which of course makes the training problem much harder.

We now introduce a discriminative model, which corresponds to the directed graph shown in Figure 6. Consider for a moment a particular image n (and omit the index n to keep the notation uncluttered). We build a parametric model $y_k(\mathbf{x}_j, \mathbf{w})$ for the probability that patch $\mathbf{x}_j$ belongs to class k. For example we might use a simple linear-softmax model with outputs

$$y_k(\mathbf{x}_j, \mathbf{w}) = \frac{\exp(\mathbf{w}_k^{\mathrm{T}} \mathbf{x}_j)}{\sum_l \exp(\mathbf{w}_l^{\mathrm{T}} \mathbf{x}_j)} \tag{5}$$

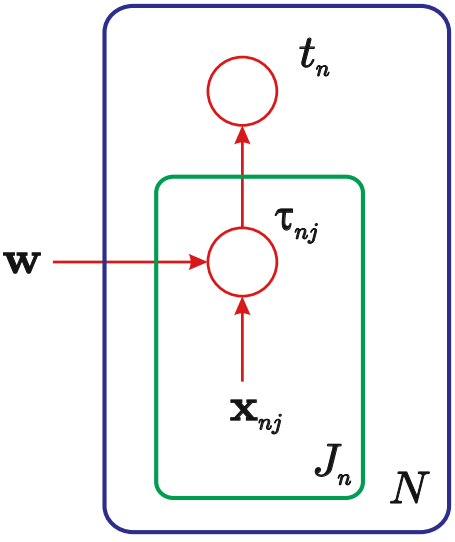


Fig. 6. Graphical representation of the discriminative model for object recognition.

which satisfy $0 \leqslant y_k \leqslant 1$ and $\sum_k y_k = 1$. More generally we can use a multi-layer neural network, a relevance vector machine, or any other parametric model that gives probabilistic outputs and which can be optimized using gradient-based methods. The probability of a patch label $\boldsymbol{\tau}_j$ is then given by

$$p(\boldsymbol{\tau}_j|\mathbf{x}_j) = \prod_{k=1}^{K} y_k(\mathbf{x}_j, \mathbf{w})^{\tau_{jk}} \tag{6}$$

where the binary exponent τ_{jk} simply pulls out the required term (since $y_k^0 = 1$ and $y_k^1 = y_k$).

Next we assume that if one, or more, of the patches carries the label for a particular class, then the whole image will. For instance, if there is at least one local patch in the image which is labelled 'cow' then the whole image will carry a 'cow' label (recall that an image can carry more than one class label at a time). Thus the conditional distribution of the image label, given the patch labels, is given by

$$p(\mathbf{t}|\boldsymbol{\tau}) = \prod_{k=1}^{K} \left[1 - \prod_{j=1}^{J}[1 - \tau_{jk}]\right]^{t_k} \left[\prod_{j=1}^{J}[1 - \tau_{jk}]\right]^{1-t_k}. \tag{7}$$

In order to obtain the conditional distribution $p(\mathbf{t}|\mathbf{X})$ we have to marginalize over the latent patch labels. Although there are exponentially many terms in this sum, it can be performed analytically for our model due to the factorization implied by the graph

in Figure 6 to give

$$p(\mathbf{t}|\mathbf{X}) = \sum_{\boldsymbol{\tau}} \left\{ p(\mathbf{t}|\boldsymbol{\tau}) \prod_{j=1}^{J} p(\boldsymbol{\tau}_j|\mathbf{x}_j) \right\}$$
$$= \prod_{k=1}^{K} \left[1 - \prod_{j=1}^{J} [1 - y_k(\mathbf{x}_j, \mathbf{w})] \right]^{t_k} \left[\prod_{j=1}^{J} [1 - y_k(\mathbf{x}_j, \mathbf{w})] \right]^{1-t_k} . \quad (8)$$

This can be viewed as a softened (probabilistic) version of the logical'OR' function [12].

Given a training set of N images, which are assumed to be independent, we can construct the likelihood function from the product of such distributions, one for each data point. Taking the negative logarithm then gives the following error function

$$E(\mathbf{w}) = -\sum_{n=1}^{N} \sum_{k=1}^{C} \{ t_{nk} \ln [1 - Z_{nk}] + (1 - t_{nk}) \ln Z_{nk} \} \quad (9)$$

where we have defined

$$Z_{nk} = \prod_{j=1}^{J_n} [1 - y_k(\mathbf{x}_{nj}, \mathbf{w})] . \quad (10)$$

The parameter vector $\mathbf{w}$ can be determined by minimizing this error (which corresponds to maximizing the likelihood function) using a standard optimization algorithm such as scaled conjugate gradients [2]. More generally the likelihood function could be used as the basis of a Bayesian treatment, although we do not consider this here.

Once the optimal value $\mathbf{w}_{\mathrm{ML}}$ is found, the corresponding functions $y_k(\mathbf{x}, \mathbf{w}_{\mathrm{ML}})$ for $k = 1, \ldots, K$ will give the posterior class probabilities for a new patch feature vector $\mathbf{x}$. Thus the model has learned to label the patches even though the training data contained only image labels. Note, however, that as a consequence of the noisy 'OR' assumption, the model only needs to label one foreground patch correctly in order to predict the image label. It will therefore learn to pick out a small number of highly discriminative foreground patches, and will classify the remaining foreground patches, as well as those falling on the background, as 'background' meaning non-discriminative for the foreground class. This will be illustrated in Section 6.

5 The Generative Model with Patch Labelling

Next we turn to a description of our generative model, whose graphical representation is shown in Figure 7. The structure of this model mirrors closely that of the discriminative model. In particular, the same class-label variables $\boldsymbol{\tau}_{nj}$ are associated with the patches in each image, and again these are unobserved and must be marginalized out in order to obtain maximum likelihood solutions.

In the discriminative model we represented the conditional distribution $p(\mathbf{t}|\mathbf{X})$ directly as a parametric model. By contrast in the generative approach we model $p(\mathbf{t}, \mathbf{X})$,

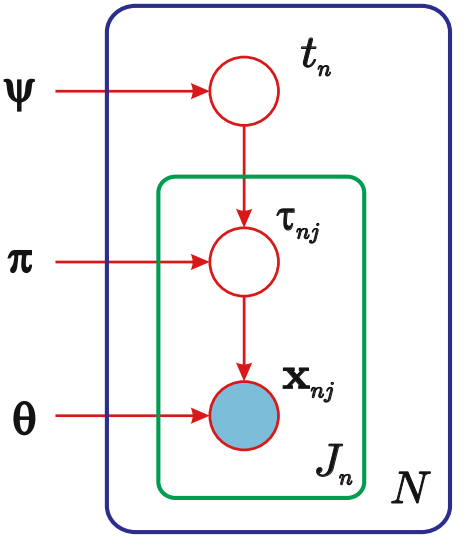


Fig. 7. Graphical representation of the generative model for object recognition.

which we decompose into $p(\mathbf{t}, \mathbf{X}) = p(\mathbf{X}|\mathbf{t})p(\mathbf{t})$ and then model the two factors separately. This decomposition would allow us, for instance, to employ large numbers of 'background' images (those containing no instances of the object classes) during training to determined $p(\mathbf{X}|\mathbf{t})$ without concluding that the prior probabilities $p(\mathbf{t})$ of objects is small.

Again, we begin by considering a single image n. The prior $p(\mathbf{t})$ is specified in terms of K parameters ψ_k where $0 \leqslant \psi_k \leqslant 1$ and $k = 1, \ldots, K$, so that

$$p(\mathbf{t}) = \prod_{k=1}^{K} \psi_k^{t_k} (1 - \psi_k)^{1-t_k}. \tag{11}$$

In general we do not need to learn these from the training data since the prior occurrences of different classes is more a property of the way the data was collected than of the real world frequencies. (Similarly in the discriminative model we will typically wish to correct for different priors between the training set and test data using Bayes' theorem.)

The remainder of the model is specified in terms of the conditional probabilities $p(\boldsymbol{\tau}|\mathbf{t})$ and $p(\mathbf{X}|\boldsymbol{\tau})$. The probability of generating a patch from a particular class is governed by a set of parameters π_k, one for each class, such that $\pi_k \geqslant 0$, constrained by the subset of classes actually present in the image. Thus

$$p(\boldsymbol{\tau}_j|\mathbf{t}) = \left(\sum_{l=1}^{K} t_l \pi_l \right)^{-1} \prod_{k=1}^{K} (t_k \pi_k)^{\tau_{jk}}. \tag{12}$$

Note that there is an overall undetermined scale to these parameters, which may be removed by fixing one of them, e.g. $\pi_1 = 1$.

For each class k, the distribution of the patch feature vector $\mathbf{x}$ is governed by a separate mixture of Gaussians which we denote by $\phi_k(\mathbf{x};\boldsymbol{\theta}_k)$, so that

$$p(\mathbf{x}_j|\boldsymbol{\tau}_j) = \prod_{k=1}^{K} \phi_k(\mathbf{x}_j;\boldsymbol{\theta}_k)^{\tau_{jk}} \tag{13}$$

where $\boldsymbol{\theta}_k$ denotes the set of parameters (means, covariances and mixing coefficients) associated with this mixture model, and again the binary exponent τ_{jk} simply picks out the required class.

If we assume N independent images, and for image n we have J_n patches drawn independently, then the joint distribution of all random variables is

$$\prod_{n=1}^{N} p(\mathbf{t}_n) \prod_{j=1}^{J_n} \left[p(\mathbf{x}_{nj}|\boldsymbol{\tau}_{nj})p(\boldsymbol{\tau}_{nj}|\mathbf{t}_n)\right]. \tag{14}$$

Since we wish to maximize likelihood in the presence of latent variables, namely the $\{\boldsymbol{\tau}_{nj}\}$, we use the EM algorithm. The expected complete-data log likelihood is given by

$$\sum_{n=1}^{N}\sum_{j=1}^{J_n}\left\{\sum_{k=1}^{K}\langle\tau_{njk}\rangle \ln\left[t_{nk}\pi_k\phi_k(\mathbf{x}_{nj})\right] - \ln\left(\sum_{l=1}^{K} t_{nl}\pi_l\right)\right\}. \tag{15}$$

In the E-step the expected values of τ_{nkj} are computed using

$$\langle\tau_{njk}\rangle = \sum_{\{\boldsymbol{\tau}_{nj}\}} \tau_{njk}\, p(\boldsymbol{\tau}_{nj}|\mathbf{x}_{nj},\mathbf{t}_n) = \frac{t_{nk}\pi_k\phi_k(\mathbf{x}_{nj})}{\sum_{l=1}^{K} t_{nl}\pi_l\phi_l(\mathbf{x}_{nj})}. \tag{16}$$

Notice that the first factor on the right hand side of (12) has cancelled in the evaluation of $\langle\tau_{njk}\rangle$.

For the M-step we first set the derivative with respect to one of the parameters π_k equal to zero (no Lagrange multiplier is required since there is no summation constraint on the $\{\pi_k\}$) and then re-arrange to give the following re-estimation equations

$$\pi_k = \left[\sum_{n=1}^{N} J_n t_{nk}\left(\sum_{l=1}^{K} t_{nl}\pi_l\right)^{-1}\right]^{-1} \sum_{n=1}^{N}\sum_{j=1}^{J_n}\langle\tau_{njk}\rangle. \tag{17}$$

Since these represent coupled equations we perform several (fast) iterations of these equations before proceeding with the next EM cycle (note that for this purpose the sums over j can be pre-computed since they do not depend on the $\{\pi_k\}$).

Now consider the optimization with respect to the parameters $\boldsymbol{\theta}_k$ governing the distribution $\phi_k(\mathbf{x};\boldsymbol{\theta}_k)$. The dependence of the expected complete-data log likelihood on $\boldsymbol{\theta}_k$ takes the form

$$\sum_{n=1}^{N}\sum_{j=1}^{J_n}\langle\tau_{njk}\rangle \ln\phi_k(\mathbf{x}_{nj};\boldsymbol{\theta}_k) + \text{const.} \tag{18}$$

This is easily maximized for each class k separately using the EM algorithm (in an inner loop), since (18) simply represents a log likelihood function for a weighted data set in which patch (n, j) is weighted with $\langle \tau_{njk} \rangle$. Specifically, we use a model in which $\phi_k(\mathbf{x}; \boldsymbol{\theta}_k)$ is given by a Gaussian mixture distribution of the form

$$\phi_k(\mathbf{x}; \boldsymbol{\theta}_k) = \sum_{m=1}^{M} \rho_{km} \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_{km}, \boldsymbol{\Sigma}_{km}). \tag{19}$$

The E-step is given by

$$\gamma_{njkm} = \frac{\rho_{km} \mathcal{N}(\mathbf{x}_{nj} | \boldsymbol{\mu}_{km}, \boldsymbol{\Sigma}_{km})}{\sum_{m'} \rho_{km'} \mathcal{N}(\mathbf{x}_{nj} | \boldsymbol{\mu}_{km'}, \boldsymbol{\Sigma}_{km'})} \tag{20}$$

while the M-step equations are weighted by the coefficients $\langle \tau_{njk} \rangle$ to give

$$\boldsymbol{\mu}_{km}^{\text{new}} = \frac{\sum_n \sum_j \langle \tau_{njk} \rangle \gamma_{njkm} \mathbf{x}_{nj}}{\sum_n \sum_j \langle \tau_{njk} \rangle \gamma_{njkm}}$$

$$\boldsymbol{\Sigma}_{km}^{\text{new}} = \frac{\sum_n \sum_j \langle \tau_{njk} \rangle \gamma_{njkm} (\mathbf{x}_{nj} - \boldsymbol{\mu}_{km}^{\text{new}})(\mathbf{x}_{nj} - \boldsymbol{\mu}_{km}^{\text{new}})^{\mathrm{T}}}{\sum_n \sum_j \langle \tau_{njk} \rangle \gamma_{njkm}}$$

$$\rho_{km}^{\text{new}} = \frac{\sum_n \sum_j \langle \tau_{njk} \rangle \gamma_{njkm}}{\sum_n \sum_j \langle \tau_{njk} \rangle}.$$

If one EM cycle is performed for each mixture model $\phi_k(\mathbf{x}; \boldsymbol{\theta}_k)$ this is equivalent to a global EM algorithm for the whole model. However, it is also possible to perform several EM cycle for each mixture model $\phi_k(\mathbf{x}; \boldsymbol{\theta}_k)$ within the outer EM algorithm. Such variants yield valid EM algorithms in which the likelihood never decreases.

The incomplete-data log likelihood can be evaluated after each iteration to ensure that it is correctly increasing. It is given by

$$\sum_{n=1}^{N} \sum_{j=1}^{J_n} \left\{ \ln \left(\sum_{k=1}^{K} t_{nk} \pi_k \phi_k(\mathbf{x}_{nj}) \right) - \ln \left(\sum_{l=1}^{K} t_{nl} \pi_l \right) \right\}.$$

Note that, for a data set in which all $t_{nk} = 1$, the model simply reduces to fitting a flat mixture to all observations, and the standard EM is recovered as a special case of the above equations.

This model can be viewed as a generalization of that presented in [14] in which a parameter is learned for each mixture component representing the probability of that component being foreground. This parameter is then used to select the most informative N components in a similar approach to [4] and [13] where the number N is chosen heuristically. In our case, however, the probability of each feature belonging to one of the K classes is learned directly.

Inference in the generative model is more complicated than in the discriminative model. Given all patches $\mathbf{X} = \{\mathbf{x}_j\}$ from an image, the posterior probability of the label $\boldsymbol{\tau}_j$ for patch j can be found by marginalizing out all other hidden variables

$$p(\boldsymbol{\tau}_j|\mathbf{X}) = \sum_{\mathbf{t}} \sum_{\boldsymbol{\tau}/\boldsymbol{\tau}_j} p(\boldsymbol{\tau}, \mathbf{X}, \mathbf{t})$$
$$= \sum_{\mathbf{t}} p(\mathbf{t}) \frac{1}{\left(\sum_{l=1}^{K} \pi_l t_l\right)^J} \prod_{k=1}^{K} \left(\pi_k t_k \phi_k(\mathbf{x}_j)\right)^{\tau_{jk}} \prod_{i \neq j} \left[\sum_{k=1}^{K} \pi_k t_k \phi_k(\mathbf{x}_i)\right] \quad (21)$$

where $\boldsymbol{\tau} = \{\boldsymbol{\tau}_j\}$ denotes the set of all patch labels, and $\boldsymbol{\tau}/\boldsymbol{\tau}_j$ denotes this set with $\boldsymbol{\tau}_j$ omitted. Note that the summation over all possible $\mathbf{t}$ values, which must be done explicitly, is computationally expensive.

For the inference of image label we require the posterior probability of image label $\mathbf{t}$, which can be computed using

$$p(\mathbf{t}|\mathbf{X}) \propto p(\mathbf{X}|\mathbf{t})\, p(\mathbf{t}) \quad (22)$$

in $p(\mathbf{t})$ is computed from the coefficients $\{\psi_k\}$ for each setting of $\mathbf{t}$ in turn, and $p(\mathbf{X}|\mathbf{t})$ is found by summing out patch labels

$$p(\mathbf{X}|\mathbf{t}) = \sum_{\boldsymbol{\tau}} \prod_{j=1}^{J} p(\mathbf{X}, \boldsymbol{\tau}_j|\mathbf{t}) = \prod_{j=1}^{J_n} \frac{\sum_{k=1}^{K} t_k \pi_k \phi_k(\mathbf{x}_j)}{\sum_{l=1}^{K} t_l \pi_l}. \quad (23)$$

6 Results

In this study, we have used a test bed of weakly labelled images each containing either cows or sheep, in which the animals vary widely in terms of number, pose, size, colour and texture. There are 167 images in each class, and 10-fold cross-validation is used to measure performance. For the discriminative model we used a linear network of the form (5) with 144 inputs, corresponding to the 144 features discussed in Section 2 and 3 outputs (cow, sheep, background). We also explore two-layer non-linear networks having 50 hidden units with '$\tanh$' activation functions, and a quadratic regularizer with hyper-parameter 0.2. For the generative model we used a separate Gaussian mixture for cow, sheep and background, each of which has 10 components with diagonal covariance matrices.

Initial results with the generative model showed that with random initialization of the mixture model parameters it is incapable of learning a satisfactory solution. We conjectured that this is due to the problem of multiple local maxima in the likelihood function (a similar effect was found by [14]). To test this we used some segmented images for initialization purposes (but not for optimization). 30 cow and 30 sheep images were hand-segmented, and features belonging to each class were clustered using the K-means algorithm and the component centers of a class mixture model were assigned to the cluster centers of the respective class. The mixing coefficients were set to the number of points in the corresponding cluster divided by the total number of points in that class. Similarly, covariance matrices were computed using the data points assigned to the respective center.

In the test phase of both discriminative and generative models, we input the patch features to the models and obtain the posterior probabilities of the patch labels as the

outputs using (5) for discriminative model and (21) for the generative model. The posterior probability of the image label is computed as in (8) for the discriminative model and (22) for the generative case. We can therefore investigate the ability of the two models both to predict the class labels of whole images and of their constituent patches. The latter is important for object localization.

The overall correct rates of object recognition, i.e. image labelling, is given in Table 1 for the VQ-S, VQ-ARD, linear discriminative (D-L), nonlinear discriminative (D-NL) and generative (G) models.

Table 1. Overall correct rates

VQ-S	VQ-ARD	D-L	D-NL	G
80%	92%	82.5%	87.2%	97%

Table 2. Patch labelling scores

Class	D-BG	D-FG	G-BG	G-FG
Cow	99%	17%	82%	68%
Sheep	99%	5%	52%	82%

It is also interesting to investigate the extent to which the discriminative and generative models correctly label the individual patches. In order to make a comparison in terms of patch labelling we used 30 hand segmented images for each class. In Table 2 patch labelling scores for foreground (FG) and background (BG) for discriminative and generative models are given. Various thresholds are used on patch label probabilities in order to produce ROC curves for the generative model and the non-linear network version of the discriminative model, as shown in Figure 8. We also plot the ROC curve for the generative model when random initialization is performed to show the importance of initialization for such models. As already noted, the discriminative model finds a small number of highly discriminative foreground patches, and labels all other patches as background, whereas the generative model must balance the accurate labelling of both foreground and background patches. Some examples of patch labelling for test images are given in Figure 9 for cow images and in Figure 10 for sheep images.

There is a huge difference between discriminative and generative models in terms of speed. The generative model is more than 20 times slower than the discriminative model in training and more than 200 times slower in testing. Typical values for the duration of a single cycle and the total duration of training and testing are given, for a Matlab implementation, in Table 3.

7 Discussion

In this paper we have introduced and compared a variety of local patch-based models for object recognition. We have shown that automatic relevance determination allows

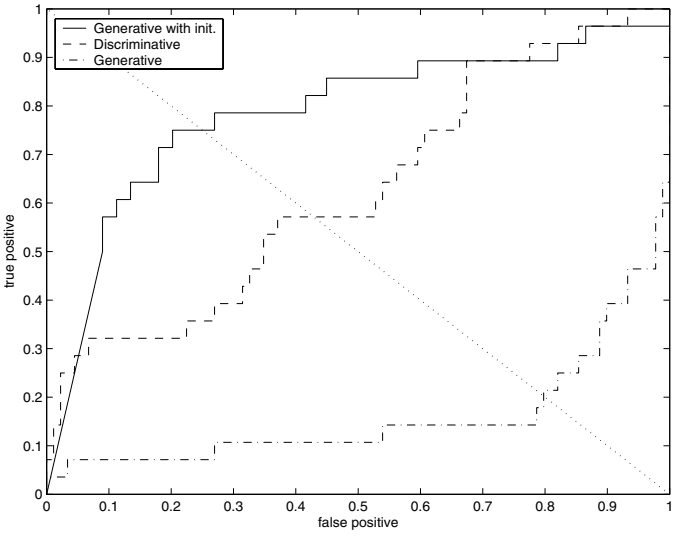


Fig. 8. ROC curves of patch labelling

Table 3. Typical values for speed (sec)

Model	Single train cycle	Total training	Testing
D-L	3	510	0.0015
D-NL	5	625	0.0033
G	386	15440	0.31

a system to learn which features are most salient in determining the present of an object. We have also introduced novel discriminative and generative models which have complementary strengths and limitations, and shown that the discriminative model is capable of fast inference, and is able to focus on highly informative features, while the generative model gives high classification accuracy, and also has some ability to localize the objects within the image. However, the generative model requires careful initialization in order to achieve good results.

One major potential benefit of the generative model is the ability to augment the labelled data with unlabelled data. Indeed, a combination of images which are unlabelled, weakly labelled (having image labels only) and strongly labelled (in which patch labels are also provided as well as the image labels) could be used, provided that all missing variables are 'missing at random'.

Another significant potential advantage of generative models is the relative ease with which invariances can be specified, particularly those arising from geometrical transformations. For instance, the effect of a translation is simply to shift the pixels. By contrast, in a discriminative model ensuring invariance to the resulting highly non-linear transformations of the input variables is non-trivial. However, inference in such a generative model can be very complex due to the need to determine values for the trans-

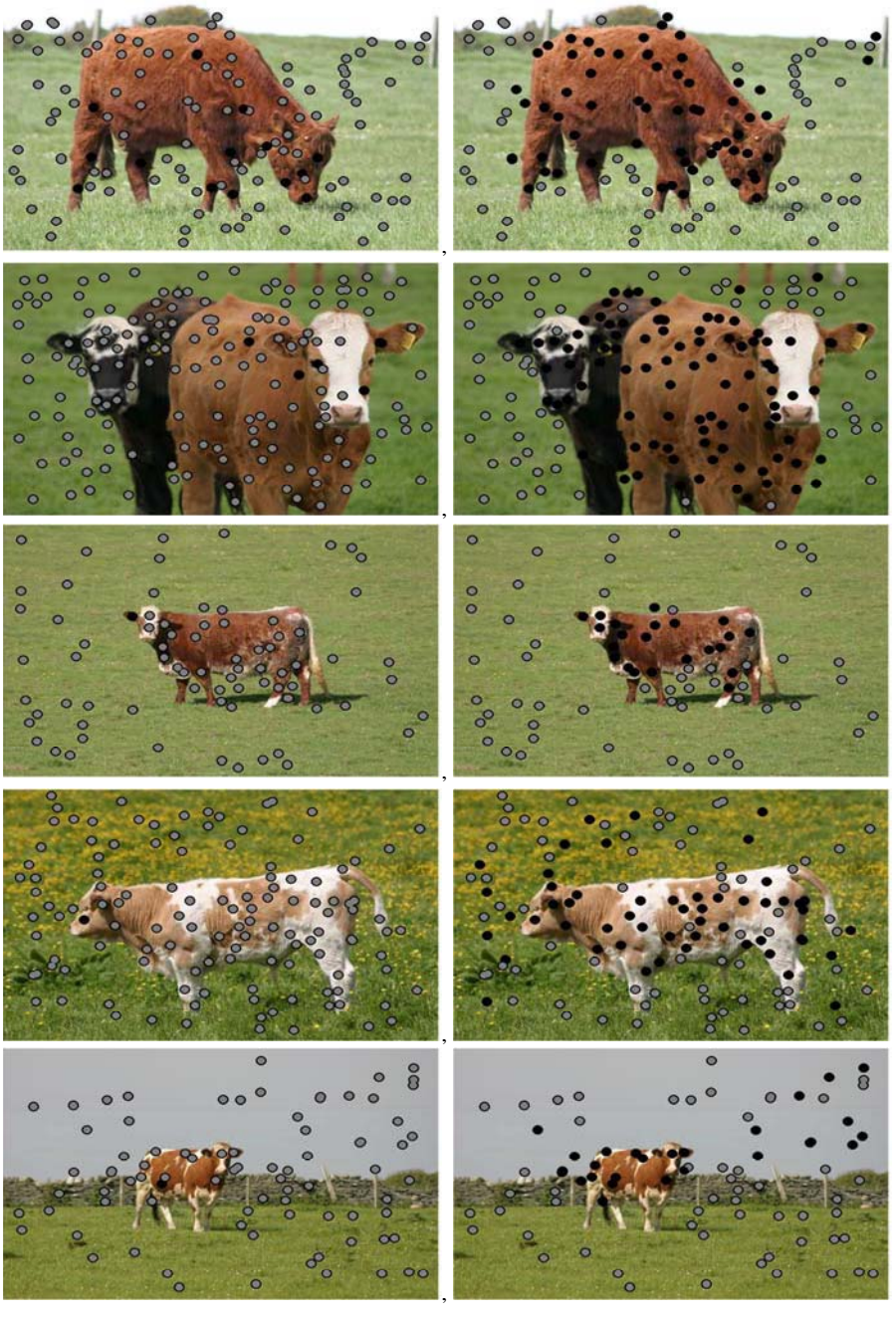


Fig. 9. Cow patch labelling examples for discriminative model (left column) and generative model (right column). Black, gray and white dots denote cow, background and sheep patches respectively (and are obtained by assigning each patch to the most probable class).

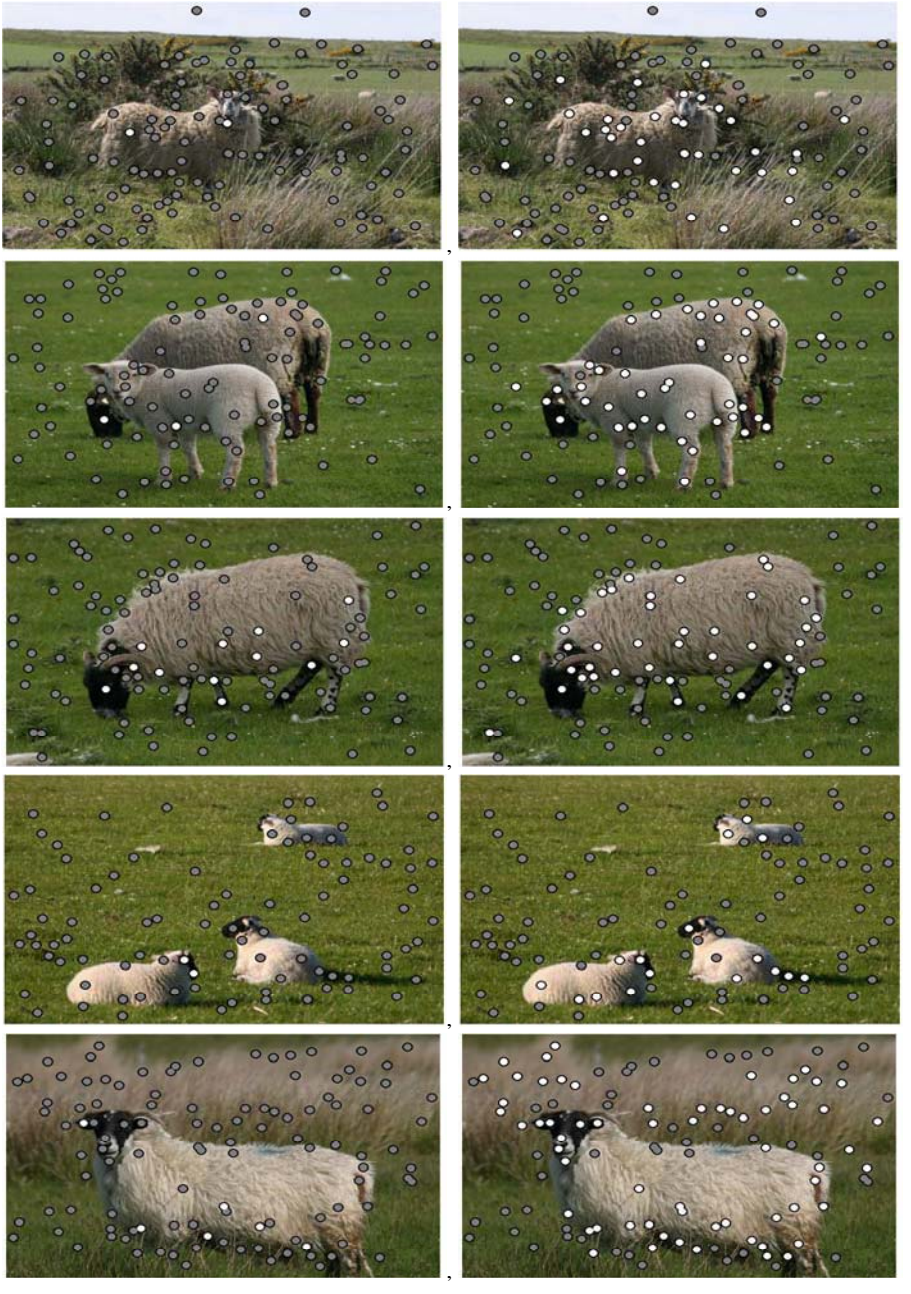


Fig. 10. Sheep patch labelling examples for discriminative model (left column) and generative model (right column). Black, gray and white dots denote cow, background and sheep patches respectively.

formation parameters which have high posterior probability, and this generally involves iteration. A discriminative model, on the other hand, is typically very fast once trained.

Our investigations suggest that the most fruitful approaches will involve some combination of generative and discriminative models. Indeed, this is already found to be the case in speech recognition where generative hidden Markov models are used to express invariance to non-linear time warping, and are then trained discriminatively by maximizing mutual information in order to achieve high predictive performance.

One promising avenue for investigation is to use a fast discriminative model to locate regions of high probability in the parameter space of a generative model, which can subsequently refine the inferences. Indeed, such coupled generative and discriminative models can mutually train each other, as has already been demonstrated in a simple context in [11].

One of the limitations of the techniques discussed here is the use of interest point detectors that are not tuned to the problem being solved (since they are hand-crafted rather than learned) and which are therefore unlikely in general to focus on the most discriminative regions of the image. Similarly, the invariant features used in our study were hand-selected. We expect that robust recognition of a large class of object categories will require that local features be learned from data.

Finally, for the purposes of this study we have ignored spatial information regarding the relative locations of feature patches in the image. However, most of our conclusions remain valid if a spatial model is combined with the local information provided by the patch features.

Acknowledgements

We would like to thank Antonio Criminisi, Geoffrey Hinton, Fei Fei Li, Tom Minka, Markus Svensen and John Winn for numerous discussions.

References

1. K. Barnard, P. Duygulu, D. Forsyth, N. Freitas, D. Blei, and M. I. Jordan. Matching words and pictures. *Journal of Machine Learning Research*, 3:1107–1135, 2003.
2. C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.
3. G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *Workshop on Statistical Learning in Computer Vision, ECCV*, 2004.
4. G. Dorko and C. Schmid. Selection of scale invariant parts for object class recognition. In *ICCV*, 2003.
5. R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale invariant learning. In *CVPR*, 2003.
6. T. Kadir and M. Brady. Scale, saliency and image description. *International Journal of Computer Vision*, 45(2):83–105, 2001.
7. D. Lowe. Distinctive image features from scale invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.
8. D. J. C. MacKay. Probable networks and plausible predictions – a review of practical Bayesian methods for supervised neural networks. 6(3):469–505, 1995.

9. K. Mikolajczyk and C. Schmid. Scale and affine invariant interest point detectors. *International Journal of Computer Vision*, 60:63–86, 2004.
10. I. T. Nabney. *Netlab Algorithms for Pattern Recognition*. Springer, 2004.
11. R. Neal P. Dayan, G. E. Hinton and R. S. Zemel. The helmholtz machine. *Neural Computation*, pages 1022–1037, 1995.
12. J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Net- works of Plausible Inference*. Morgan Kaufmann Publishers, 1998.
13. M. Vidal-Naquet and S. Ullman. Object recognition with informative features and linear classification. In *ICCV*, 2003.
14. L. Xie and P. Perez. Slightly supervised learning of part-based appearance models. In *IEEE Workshop on Learning in CVPR*, 2004.

Multi Channel Sequence Processing

Samy Bengio[1] and Hervé Bourlard[1,2]

[1] IDIAP Research Institute, CP 592, rue du Simplon 4, 1920 Martigny, Switzerland
[2] Swiss Federal Institute of Technology at Lausanne (EPFL), Switzerland
{bengio, bourlard}@idiap.ch

Abstract. This paper summarizes some of the current research challenges arising from multi-channel sequence processing. Indeed, multiple real life applications involve simultaneous recording and analysis of multiple information sources, which may be asynchronous, have different frame rates, exhibit different stationarity properties, and carry complementary (or correlated) information. Some of these problems can already be tackled by one of the many statistical approaches towards sequence modeling. However, several challenging research issues are still open, such as taking into account asynchrony and correlation between several feature streams, or handling the underlying growing complexity. In this framework, we discuss here two novel approaches, which recently started to be investigated with success in the context of large multimodal problems. These include the asynchronous HMM, providing a principled approach towards the processing of multiple feature streams, and the layered HMM approach, providing a good formalism for decomposing large and complex (multi-stream) problems into layered architectures. As briefly reported here, combination of these two approaches yielded successful results on several multi-channel tasks, ranging from audio-visual speech recognition to automatic meeting analysis.

1 Introduction

Given the proliferation of electronic recording devices (cameras, microphones, EEGs, etc) with ever cheaper, and ever increasing processing speed, storage, and bandwidth, together with the advances in automatically extracting and managing information recorded from these devices (such as speech recognition, face tracking, etc), it becomes more and more feasible to simultaneously capture a same event (or multiple events) with several devices, generating richer and more robust sets of feature-streams.

Modeling such data coming from multiple channels (thus resulting in multiple observation streams) is the goal of *multi-channel sequence processing*. Examples of practical applications of this field are numerous, such as audio-visual speech recognition, which can be more robust to ambient noise than only using an audio stream. While several statistical models were presented recently in the literature to cope with this growing amount of data accessible in parallel, several open research problems are still to be solved. The purpose of this paper is thus to discuss some of these solutions, and specifically addressing two important issues, i.e.,

J. Winkler, N. Lawrence, M. Niranjan (Eds.): Machine Learning Workshop, LNAI 3635, pp. 22–36, 2005.

asynchrony (when the feature streams are supposed to be piecewise stationary, but with different stationary properties) and *complexity* (when it is furthermore necessary to split the problem into several multi-stream sub-problems).

The outline of the paper is as follows. Section 2 justifies the need for multi-channel sequence processing by discussing some of the numerous applications that require such a framework. Section 3 reviews some of the current models used in the literature. Section 4 shows that despite all these models, there is still room for several improvements. Section 5 proposes a model to handle temporal asynchrony between channels, while Section 6 proposes a principled approach to control the complexity of multi-channel sequence processing through "optimal" hierarchical processing.

2 Some Applications

Several tasks that are currently handled with only one stream of information could in fact benefit from the addition of other parallel streams. Furthermore, like in speech recognition (as well as video processing), it becomes more and more usual to apply different feature extraction techniques to the same signal, resulting in multiple feature streams

For instance, in *audio-visual speech recognition*, the audio signal is typically complemented by the video recording of the face (and thus the lips) of the person. It has already been shown [1,2] that if the resulting audio and visual feature streams are properly modeled, such a multi-channel approach will significantly help in recognizing the speech utterances under noise conditions. Similar settings have also been used successfully for *audio-visual person authentication* [2]. In fact, even using only one raw source of information can yield better results in a multi-channel setting, e.g., using multiple sampling rates (*multi-rate*) or feature extraction (*multi-stream*) techniques, as already demonstrated for the task of speech recognition [3].

The field of *multimedia analysis*, which includes analysis of news, sports, home videos, meetings, etc, is very rich and these events are often recorded with at least two streams of information (audio and video) and sometimes more (as for the meeting scenario described later in this paper), and may contain complex human human interactions [4]. These multimedia documents also give rise to other applications such as *multimodal tracking of objects/humans* [5]. Furthermore, as the quantity of such archived documents grows, it becomes important to develop *multimedia document retrieval* systems [6,7] to find relevant documents based not only on their textual content but also on their joint visual and audio content.

Finally, numerous multi-channel sequence processing processing also appear in the context *wearable computers* [8], aiming at assisting people in various every-day activities (e.g., life saving, security, health monitoring, mobile web services) by using small devices such as cameras, microphones (e.g., recording all what you see and all what you hear), and multiple extra sensors (e.g., recording diverse physiological signals), etc.

In all the above applications, multi-channel processing presents several challenges. As already mentioned earlier, we first have to develop new sequence recognition strategies accommodating multiple frame rates, asynchrony, correlation between stream, etc. One solution to this problem, referred to as "Asynchronous HMM" (AHMM) will be discussed in the paper (Section 5). Furthermore, multi-channel processing may also impact differently the different levels of information that we aim at extracting from the observation streams. While AHMM can be well suited to classify sequential patterns into "low level" classes, they may not be appropriate, or easily tractable (because of training data and complexity issues), when one aims at extracting higher level information, such as semantic classes. In this case, it may be necessary to use a "hierarchical HMM" approach, where each "HMM layer" will use different types of multiple observation streams (possibly resulting of the previous HMM layer). This layered approach will be discussed in Section 6.

3 Notation and Models

Several models have already been proposed in the literature to handle multi-channel applications. We briefly discuss here some of the most successful approaches, using a unified notation. Let us denote an observation sequence $\mathbf{O}$ of T feature vectors as

$$\mathbf{O} = (\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_T), \tag{1}$$

where $\mathbf{o}_t$ is the vector of all multimodal features available at time t. In general, such a set of features can be broken down into multiple streams (associated with channels, modalities, or different pre-processing) m. We thus further define the feature vector

$$\mathbf{o}_t^m \in \mathbb{R}^{N_m}, \tag{2}$$

where N_m is the number of features for stream m, with $1 \leq m \leq M$ (the total number of observation streams). Each observation sequence is typically associated with a corresponding sequence of high level classes or "events". For instance, in speech or handwriting recognition, this would correspond to a sequence of words. The most successful types of model used to handle observation sequences are all based on a statistical framework. In this context, the general idea is to estimate, for each type of high level event $\mathbf{v}_j \in V$, the parameters θ_j of a distribution over corresponding observation sequences $p(\mathbf{O}|\theta_j)$, where $\mathbf{O}$ would correspond to the event $\mathbf{v}_j$. The most well-known solution to efficiently model such distributions is to use Hidden Markov Models (HMMs).

HMMs have been used with success for numerous sequence recognition tasks, including speech recognition [9], video segmentation [10], sports event recognition [11], and broadcast news segmentation [12]. HMMs introduce a state variable q_t and factor the joint distribution of the observation sequence and the underlying (unobserved) HMM state sequence into two simpler distributions, namely emission distributions $p(\mathbf{o}_t|q_t)$ and transition distributions $p(q_t|q_{t-1})$. Such factorization assumes an underlying piece-wise stationary process (each stationary

segment being associated with a specific HMM state), and yields efficient training algorithms such as the Expectation-Maximization (EM) algorithm [13] which can be used to select the set of parameters θ_j^* of the model corresponding to event $\mathbf{v}_j$ in order to maximize the likelihood of L observation sequences:

$$\theta_j^* = \arg\max_{\theta_j} \prod_{l=1}^{L} p(\mathbf{O}_l|\theta_j). \tag{3}$$

The success of HMMs applied to sequences of events is based on a careful design of sub-models (topologies and distributions) corresponding to lexical units (phonemes, words, letters, events), and possibly semantic units (like the meeting group actions discussed in Section 6.1). Given a training set of observation sequences for which we know the corresponding labeling in terms of high level events (but not necessarily the precise alignment), we create a new HMM for each sequence as the concatenation of sub-model HMMs corresponding to the sequence of high level events. This HMM can then be trained using EM, thus adapting each sub-model HMM accordingly.

During testing, when observing a new observation sequence, the objective is simply to find the optimal sequence of sub-model HMMs (representing high level events) that could have generated the given observation sequence. Multiple algorithms have been developed to efficiently solve this problem, even in large search spaces, including stack decoders [14], or different approximations based on the well-known Viterbi algorithm [15].

While HMMs can be used to model various kinds of observation sequences, several extensions have been proposed to handle simultaneously multiple streams of observations, all corresponding to the same sequence of events [3,1,16]. The first and simplest solution is to *merge* all observations related to all streams into a single stream (frame by frame), and to model it using a single HMM as explained above. This solution is often called *early integration*. Note that in some cases, when the streams represent information collected at different frame rates (such as audio and video streams for instance), up-sampling or down-sampling of the streams is first necessary in order to align the streams to a common frame rate.

A better solution may be to use the *multi-stream* approach [17]. In this case, each stream is modeled separately using its own HMM. For instance, if we consider the modalities as separate streams, we would create one model $\theta_{m,j}^*$ for each event $\mathbf{v}_j$ and stream m such that

$$\theta_{m,j}^* = \arg\max_{\theta_{m,j}} \prod_{l=1}^{L} p(\mathbf{O}_l^m|\theta_{m,j}), \tag{4}$$

where $\mathbf{O}_l^m$ is the l^{th} observation sequence of stream m. When a new sequence of events needs to be analyzed, a special HMM is then created, recombining all the single stream HMM likelihoods at various specific temporal ("anchor") points automatically determined during training and decoding. Depending on these

recombination points, various solutions appear. When the models are recombined after each state, the underlying system is equivalent to making the hypothesis that all streams are state-synchronous and independent of each other given a specific HMM state. This solution can be implemented efficiently and has shown robustness to various stream-dependent noises. The emission probability of the combined observations of M streams in a given state of the model corresponding to event $\mathbf{v}_j$ at time t is estimated as:

$$p(\mathbf{o}_t|q_t) = \prod_{m=1}^{M} p(\mathbf{o}_t^m|q_t, \theta_{m,j}). \tag{5}$$

One can see this solution as searching the best path into an HMM where each state i would be a combination of all states i of the single stream HMMs[1]. A more powerful recombination strategy enables some form of asynchrony between the states of each stream: one could consider an HMM in which states would include all possible combinations of the single stream HMM states. Unfortunately, the total number of states of this model would be exponential in the number of streams, hence quickly intractable. An intermediate solution, which we call *composite HMM*, considers all combinations of states in the same event only [18]. Hence, in this model, each event HMM j now contains all possible combinations of states of the corresponding event $\mathbf{v}_{m,j}$ of each stream HMM m. The total number of states remains exponential but is more tractable, when the number of states of each stream remains low as well as the number of streams. The underlying hypothesis of this intermediate solution is that all streams are now event-synchronous instead of state-synchronous.

Several other approaches to combine multiple streams of information have been proposed in the literature, but generally suffer from an underlying training or decoding algorithm complexity which is exponential in the number of streams. For instance, *Coupled Hidden Markov Models* (CHMMs) [19] can model two concurrent streams (such as one audio and one video stream) with two concurrent HMMs where the transition probability distribution of the state variable of each stream depends also on the value of the state variable of the other stream at the previous time step. More formally, let q and r be respectively the state variables of both streams, then CHMMs model transitions according to $p(q_t{=}i|q_{t-1}{=}j, r_{t-1}{=}k)$ and $p(r_t{=}i|r_{t-1}{=}j, q_{t-1}{=}k)$. While the exact training algorithm for such a model quickly becomes intractable when extended to more than 2 streams, an approximate algorithm which relaxes the requirement to visit every transition (termed the N-heads algorithm) was proposed in [19], and can be tractable for a small number of streams.

Two additional approaches have been proposed recently, and will be the focus of Sections 5 and 6. These are the *Asynchronous HMM* [20], that can handle asynchrony between streams, and the *Layered HMM* [21,22] than can help in constraining the model according to levels of prior knowledge.

[1] Note that this solution forces the topology of each single stream to be the same.

4 Challenges

While there are already several models proposed in the literature to cope with multi channel sequence processing, we believe that there are still several research challenges that have not been adequately addressed yet, including:

1. **How to handle more than two streams?** Most solutions that model the joint probability of the streams need in general exponential resources with respect to the number of streams, the number of states of each underlying Markov chain, or the size of each stream. This practically means that handling more than two streams is already a challenge. One possible alternative is to limit the search space through the use of reasonable heuristics, which should depend on *a priori* knowledge on the interdependencies of the streams.
2. **How to handle learning in high dimensional spaces?** The observation space (the total number of observed features per time step) grows naturally with the number of streams. Furthermore, it is often the case that the total number of parameters of the model grows linearly or more with the number of observations (for instance if the conditional observation distributions are modeled with Gaussian Mixture Models). Hence, one has to fight the well-known *curse of dimensionality* [23].
3. **How to handle long term temporal dependencies?** This problem deals with sequential data where one needs to relate information observed at time t with information observed at time $t + k$ where k is rather large. It has been shown [24] that this becomes exponentially difficult with k when no structural knowledge is built *a priori* in the model. Hence, in order for multi channel processing to be successful, an appropriate structure is necessary.
4. **Joint feature extraction and heterogeneity of sources.** In current systems involving multiple streams of information, features used to represent each stream are extracted independently. On the other hand, if one agrees that there may be some correlation between the streams, one should therefore devise joint feature extraction techniques, which should then yield more robust performance. However, what should we then do with streams of different nature (such as the slides of a presentation, together with the video of the person performing the same presentation)?
5. **How to handle different levels of** *a priori* **knowledge constraints?** It has been known for decades that in order to obtain good speech recognition performance, one has to constrain the recognition model with a good language model, that only permits valid and probable sequences of words to be recognized. The same idea should thus be applied to other domains, such as videos, which contain rich high level information that should be constrained somehow. Several levels of description should thus be used in such language model; for instance, a visual scene could be described by the pixels of the image, the persons present in the image, the action taking place, the body language, etc. For each of these levels, a probabilistic model of what is possible and what is not should therefore be trained. Furthermore, one should devise **multi channel language models** in order to take into account information coming from several streams at the same time.

6. **Asynchrony between streams.** Let us consider the simplest multi-channel case, with 2 streams, and let us assume that these 2 streams describe the same sequence of 3 "events" (classes) A, B and C. Furthermore, let us assume, as illustrated in Figure 1, that the best piecewise stationary alignment of each stream to the sequence A-B-C would not coincide temporally with each other (which we refer to "stream asynchrony"). In such a case (which is discussed in more details in Section 5), a naive solution to try to model the joint probability of the two streams (e.g., applying early integration) would need an exponential number of states (with respect to the number of streams), as depicted in the third line of Figure 1. A better solution, depicted in the fourth line of Figure 1, would *stretch* or *compress* the streams along a single HMM model with the goal to *re-align* them during training and decoding. Such a model is described in Section 5.

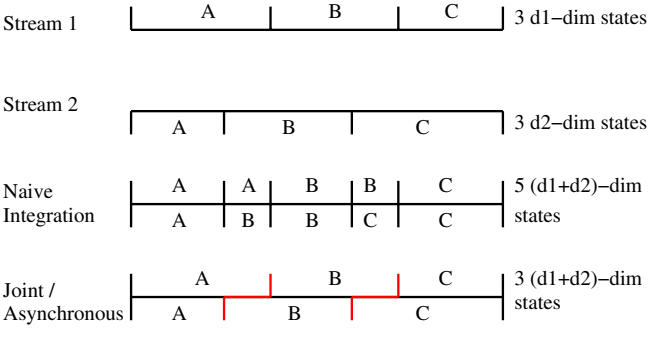


Fig. 1. Complexity issue with asynchronous streams

7. **Available benchmark datasets for evaluation**. One of the reasons of the steady progress of speech recognition has been the ever increasing availability of larger and larger realistic labeled datasets, and the yearly organization of international competitions. It is well known that this is a key point for progress in any scientific research field. However, to date, very little material has been recorded and properly annotated for multi channel sequence processing. Audio-visual speech recognition and person authentication are probably the fields where most available databases can be found. What about other scenarios, such as multimedia analysis, multimodal surveillance, etc? In Section 6, we describe a first initiative of such a benchmark database available for the meeting scenario.

5 Handling Asynchrony

Properly modeling asynchrony and correlation between multiple observation streams is thus a challenging problem. However, as a matter of fact, there are multiple evidences of real life applications involving several asynchronous

streams. For instance, audio-visual speech recognition usually exhibits asynchrony. Indeed, the lips of a person often start moving earlier than any sound is uttered, mainly because the person is preparing to utter the sound. Another example is the *speaking and pointing* scenario, where a person complement the speech signal with a pointing gesture (to a point of interest). In this case, of course, although the two streams are related to the same high-level event, the pointing event will usually never occur exactly at the same time as the vocal event. One last example of asynchrony: in a news video, there is almost always a variable delay between the moment when the newscaster says the name of a public personality and the moment when the personality's picture actually appears on the screen.

One can think of several other instances involving asynchrony between streams, and there is thus a need to model this phenomenon in a principled way. As described below, such a solution, referred to as *Asynchronous HMM* was recently proposed.

5.1 The Asynchronous HMM

Let us consider the case where one is interested in modeling the joint probability of two asynchronous streams, denoted here $\mathbf{O}^1$ of length T_1 and $\mathbf{O}^2$ of length T_2 with $T_2 \leq T_1$ without loss of generality[2]. We are thus interested in modeling $p(\mathbf{O}^1, \mathbf{O}^2)$. Following the ideas introduced for HMMs, we represent this distribution using a hidden variable Q which represents the (discrete) *state* of the generating system, which in our case is synchronized with the longest sequence $\mathbf{O}^1$.

Moreover, since we know that $\mathbf{O}^2$ is smaller than $\mathbf{O}^1$, let the system always emit $\mathbf{o}_t^1$ at time t but only sometimes emit $\mathbf{o}_s^2$ at time t, with $s \leq t$. Let us define $\tau_t{=}s$ as the fact that o_t^1 is emitted at the same time as o_s^2; τ can thus be seen as the alignment between $\mathbf{O}^1$ and $\mathbf{O}^2$. Hence, an Asynchronous HMM (AHMM) [20] models $p(\mathbf{O}^1, \mathbf{O}^2, Q, \tau)$.

Using these hidden variables, and using several reasonable independence assumptions, we can factor the joint likelihood of the data and the hidden variables into several simple conditional distributions:

- $P(q_t{=}i|q_{t-1}{=}j)$, the probability to go from state j to state i at time t,
- $p(\mathbf{o}_t^1, \mathbf{o}_s^2|q_t{=}i)$, the joint emission distribution of $\mathbf{o}_t^1$ and $\mathbf{o}_s^2$, while in state i at time t,
- $p(\mathbf{o}_t^1|q_t{=}i)$, the emission distribution of $\mathbf{o}_t^1$ only, while in state i at time t,
- $P(\tau_t{=}s|\tau_{t-1}{=}s{-}1, q_t{=}i, \mathbf{o}_{1:t}^1, \mathbf{o}_{1:s}^2)$, the probability to emit on both sequences while in state i at time t.

We showed in [20] that using these simple distributions, new algorithms could be developed to (1) estimate the *joint likelihood* of the two streams, (2) *train a*

[2] Since all the reasoning below can easily be generalized to sequences (even of the same length) where the warping (stretching and compressing) can occur at different instances in the different streams.

model to maximize the joint likelihood of pairs of streams, and (3) jointly estimate the *best sequence of states* Q and the best alignment between pairs of streams.

Furthermore, one can still constrain the model to consider only reasonable alignments, e.g., integrating some minimum and maximum asynchrony between the streams. Using this constraint and denoting N_q the number of states of the model, the training and decoding complexity become $\mathcal{O}(N_q^2 \cdot T_1 \cdot k)$, which is only k times the usual HMM complexity.

5.2 Audio-Visual Speech Recognition

The proposed AHMM model was applied to several tasks, including audio-visual speech recognition and speaker verification [2], as well multi-channel meeting analysis [21]. We report here results on the M2VTS database [25] for the task of audio-visual speech recognition, where the speech features where standard Mel-Frequency Cepstral Coefficients (MFCCs), while the visual features where shapes and intensities around the mouth region, obtained by lip tracking. In order to evaluate the robustness of audio-visual speech recognition, various levels of noise were injected into the audio stream during decoding, while training was always done using clean audio only. The noise was taken from the Noisex[3] database [26], and added to the speech signal injected to reach segmental signal-to-noise ratios (SNR) of 10dB, 5dB and 0dB.

Asynchronous HMMs were compared to classical HMMs using only the audio stream, only the video stream, or both streams combined using the *early integration* scheme. Figure 2 presents the results in terms of *Word Error Rate* (WER), a commonly used measure in the field of speech recognition, which takes into account the number of insertions, deletions and substitutions[4]. As observed from Figure 2, the AHMM consistently yielded lower WER as soon as the noise level was significant. Actually, it did not yield significantly lower performance (using a 95% confidence interval) than the video stream alone in case of very low (0dB) SNR, while performing as well as the audio stream alone in case of "clean" speech (10dB).

An interesting side effect of the model is to provide the "optimal" alignment between the audio and the video streams, as a by-product of the decoding process. This is illustrated in Figure 3 showing the audio-visual stream alignment resulting from the AHMM decoding of a specific digit sequence corrupted with 10dB Noisex noise. As it can be seen, the alignment is far from being linear. This shows that computing and maximizing the joint stream probability using AHMM appears more informative than using a naive alignment and a normal HMM.

6 A Layered Approach

6.1 The Meeting Scenario

Automatic analysis of meetings (including, e.g., automatic modeling of human interaction in meetings by modeling the joint behavior of participants through

[3] We took the *stationary speech noise*.

[4] Basically, the edit (Levenshtein) distance between the recognized and reference word sequences.

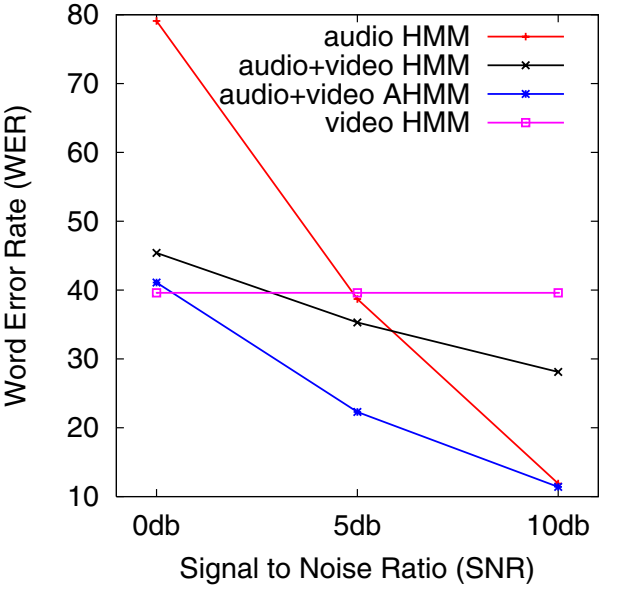


Fig. 2. Word Error Rates (in percent, the lower the better), of various systems under various noise conditions

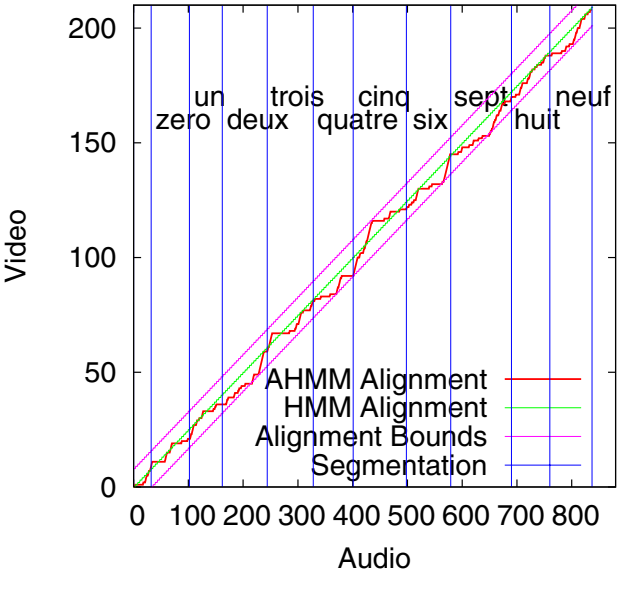


Fig. 3. Alignment obtained by the model between video and audio streams on a typical sequence corrupted with a 10dB Noisex noise. The vertical lines show the obtained segmentation between the words. The alignment bounds represent the maximum allowed stretch between the audio and the video streams.

multiple audio and visual features) is a particularly challenging application of multi-channel sequence processing. It is multimodal by nature (meetings can be recorded with several cameras and microphones, as well as with other devices capturing information coming from the white-board, the slide projector, etc) and is also a rich case study of human interaction.

In [4], a principled approach to the automatic analysis of meetings was proposed, defining meetings as continuous sequences of *group actions* chosen from a predefined dictionary of actions (including, for instance, monologue, discussion, white-board presentation, with or without note-taking, agreement/disagreement, etc). This made the problem well suited for supervised learning approaches. The group actions should be mutually exclusive, exhaustive, and as much as possible unambiguous to human observers. To this end, we have collected a corpus of 60 short meetings of about 5 minutes each (30 for training, and 30 for test purposes) in a room equipped with synchronized multi-channel audio and video recorders. The resulting corpus, including annotation, is now publicly available at `http://mmm.idiap.ch`[5]. Each meeting consisted of four participants seated at a table in a typical workplace setting. Three cameras captured the participants, the projector screen and white-board. Audio was recorded using one lapel microphone per participant and an eight-microphone array located in the center of the table. The overall goal was to minimize the *Action Error Rate* (AER), similarly to what is done in speech recognition with Word Error Rate (WER), but over sequences of high level group actions. To this end, several extensions of HMMs, including AHMMs, were tested and results are reported in [4].

More recently, we proposed a multi-layered solution [21,22] intended at simplifying the complexity of the task, based on an approach presented in [16].

6.2 A Two-Layer Approach

Let us define two sets of actions, whether they are specific to individual participants or to the group. While the overall goal is at the level of group actions, we believe that individual actions could act as a bridge between high level complex group actions and low level features, thus decomposing the problem into stages, or layers.

To this end, we defined the group action vocabulary set with the following 8 actions: *discussion*, *monologue*, *monologue+note-taking*, *note-taking*, *presentation*, *presentation+note-taking*, *white-board*, *white-board+note-taking*. Furthermore, we defined the individual action vocabulary with the following 3 actions: *speaking*, *writing*, *idle*.

Obviously, individual actions should be easier to annotate in the corpus (as being less ambiguous) and should also be easier to learn with some training data, as they are obviously more related to low level features that can be extracted from the raw multiple channels. Furthermore, knowing the sequence of individual actions of each participant, one should easily be able to infer the

[5] In the framework of the AMI European Integrated Project (`http://www.amiproject.org`) this corpus is now extended to about 100 hours of multimodal meeting data.

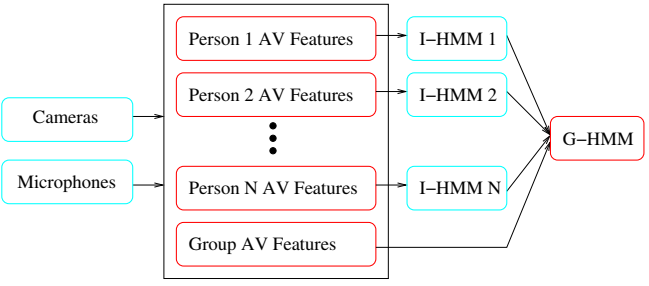


Fig. 4. A two-layer approach

underlying sequence of group actions. Thus considering every meeting participant as a "multi-stream generator", each of the participant's streams should be processed by a first layer of HMMs, and the resulting HMM's outputs (likelihoods/posteriors) will then be combined by a second HMM layer yielding, higher level, group actions.

Figure 4 illustrates the overall strategy. Audio-visual features are first extracted for each of the meeting participants [21], complemented by more general *group-level* features. An *individual HMM* (I-HMM) is then trained for each participant, using the individual action vocabulary. To have these I-HMMs as much "participant independent" as possible, all parameters are shared among all models, yielding up to 4 times more data to train the I-HMMs. Several models were compared, including early integration, multi-stream, and asynchronous HMMs (AHMM).

We then estimate for each participant i the posterior probability of each individual task $\mathbf{v}_{i,j}$ at each time step t given the individual observation sequence up to time t, $p(\mathbf{v}_{i,j}|\mathbf{o}^i_{1:t})$. These posterior probabilities, together with *group-level features*, are then used as observations for the second layer, the *group* HMM, (G-HMM), which are trained on the group action vocabulary. Again, this G-HMM was implemented in various flavors, including early integration, multi-stream and asynchronous HMMs. Section 6.3 below further discusses this aspect and shows how these (lower level) posterior probabilities can be estimated to guarantee some form of "optimality", while preserving maximum information (i.e., avoiding local decisions) across the different layers.

Table 1 reports the AER performance achieved by the different systems. It can be seen that (1) the two-layer approach always outperforms the single-layer one, and (2) the best I-HMM model is the Asynchronous HMM, which probably means that some asynchrony exists in this task, and is actually well captured by the model.

6.3 General Multi-layered (Hierarchical) HMM Approach

As illustrated from the above meeting scenario, the complexity resulting from the processing of multiple channels of information, in order to extract low-level

Table 1. Action error rates (AER) for various systems applied to the meeting scenario

Method		AER (%)
Single-layer	Visual only	48.20
	Audio only	36.70
	Early Integration	23.74
	Multi-Stream	23.13
	Asynchronous	22.20
Two-layer	Visual only	42.45
	Audio only	32.37
	Early Integration	16.55
	Multi-Stream	15.83
	Asynchronous	15.11

as well as high-level information (such as the analysis of multimodal meetings in terms of high level meeting actions), is often such that it will often be necessary to *break down* the problem in terms of multiple layers of sub-problems, probably using different constraints and prior knowledge information sources. The layered approach is one possible and principled solution to achieve this. Given a complex task, the goal is then to break it down into several hierarchically embedded sub-tasks, for which one can devise proper models (from enough training data), and use adequate (level specific) constraints.

We recently proposed such an approach for the task of speech recognition [22], where a general theoretical framework was proposed to compute low-level (e.g., phoneme) class posteriors, based on all the acoustic context, and to hierarchically combine those posteriors to yield higher-level (e.g., sentence) posteriors. In this approach, each layer is integrating its own prior constraints.

More precisely, a first layer, which could be an HMM or an AHMM, as in the meeting scenario, or any other model such as an Artificial Neural Network (ANN), is used to estimate posterior probabilities $p(q_t = i|\mathbf{O})$ of sub-classes i (such as phonemes, for the case of speech recognition) at each time step t given all the available information (for instance, all the acoustic sequence $\mathbf{O}$). In HMM, as well as in hybrid HMM/ANN systems, this posterior probability estimate is given by the so-called $\gamma(i,t) = p(q_t = i|\mathbf{O})$, which can be obtained by running and combining the so-called α and β recurrences through the appropriate HMM. Ideally, this HMM should embed all known lexical constraints about legal and probable sequences of phonemes. One should then use the resulting posterior probabilities (of every sub-class at every time step) as input to the next layer model, which would then estimate the posterior probabilities (again through new γ's) of higher level classes, such as words, constraining the underlying HMM model with all known language constraints that pertains to legal and probable sequences of words. In theory, this operation could be repeated up to the the level of sentences, and even to the level of summarization, always using posterior probabilities resulting from the previous layer as intermediate features.

Initial results on several speech tasks, as well as on the meeting task discussed previously, resulted in significant improvements. In [22], speech recogni-

tion results where presented on Numbers'95 (speaker independent recognition of free format numbers spoken over the telephone) and on a reduced vocabulary version (1,000 words) of the DARPA Conversational Telephone Speech-to-text (CTS) task, and both resulted in significant improvements.

7 Conclusion

This paper discussed several issues arising from the processing of complex multi-channel data, including large multimodal problems (meeting data). More specifically, this paper focused on two important issues, namely stream asynchrony and complexity of high-level decision processes. The proposed Asynchronous HMMs (AHMM) actually maximize the likelihood of the joint observation sequences through a single HMM, while also automatically allowing for stretching and/or compressing of the different streams. However, in the case of very complex problems, using AHMMs is often not enough, and the problem needs to be broken down into simpler processing blocks. A solution to this problem, referred to as "multi-layered/hierarchical HMMs" (and where each layer can integrate different levels of constraints and prior information) was also proposed and shown to be effective in modeling the joint behavior of participants in multimodal meetings. A full theoretical motivation of this approach is described in [22].

Acknowledgements

This work was partly funded (through OFES) by the European PASCAL Network of Excellence and the AMI Integrated Project. The results reported here also benefited from financial support from the Swiss National Science Foundation, through the National Center of Competence in Research IM2 (Interactive Multimodal Information Management).

References

1. Dupont, S., Luettin, J.: Audio-visual speech modeling for continuous speech recognition. IEEE Transactions on Multimedia **2** (2000) 141–151
2. Bengio, S.: Multimodal speech processing using asynchronous hidden markov models. Information Fusion **5** (2004) 81–89
3. Morris, A., Hagen, A., Glotin, H., Bourlard, H.: Multi-stream adaptive evidence combination for noise robust ASR. Speech Communication (2001)
4. McCowan, I., Gatica-Perez, D., Bengio, S., Lathoud, G., Barnard, M., Zhang, D.: Automatic analysis of multimodal group actions in meetings. IEEE Transactions on Pattern Analysis and Machine Intelligence **27** (2005) 305–317
5. Gatica-Perez, D., Lathoud, G., McCowan, I., Odobez, J.M.: A mixed-state i-particle filter for multi-camera speaker tracking. In: Proc. of WOMTEC. (2003)
6. Renals, S., Abberley, D., Kirby, D., Robinson, T.: Indexing and retrieval of broadcast news. Speech Communication **32** (2000) 5–20
7. Westerveld, T., de Vries, A.P., van Ballegooij, A., de Jong, F., Hiemstra, D.: A probabilistic multimedia retrieval model and its evaluation. EURASIP Journal on Applied Signal Processing **2** (2003)

8. Mann, S.: Smart clothing: The wearable computer and wearcam. Personal Technologies (1997) Volume 1, Issue 1.
9. Rabiner, L.R., Juang, B.H.: Fundamentals of Speech Recognition. Prentice-Hall (1993)
10. Boreczky, J.S., Wilcox, L.D.: A Hidden Markov Model framework for video segmentation using audio and image features. In: Proc. of ICASSP. Volume 6. (1998)
11. Xie, L., Chang, S.F., Divakaran, A., Sun, H.: Structure analysis of soccer video with Hidden Markov Models. In: ICASSP. (2002)
12. Eickeler, S., Müller, S.: Content-based video indexing of TV broadcast news using Hidden Markov Models. In: Proc. of ICASSP. (1999)
13. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum-likelihood from incomplete data via the EM algorithm. Journal of Royal Statistical Society B **39** (1977) 1–38
14. Jelinek, F.: A fast sequential decoding algorithm using a stack. IBM Journal of Research and Development **13** (1969) 675–685
15. Viterbi, A.: Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. IEEE Transactions on Information Theory (1967) 260–269
16. Oliver, N., Horvitz, E., Garg, A.: Layered representations for learning and inferring office activity from multiple sensory channels. In: Proc. of the Int. Conf. on Multimodal Interfaces. (2002)
17. Bourlard, H., Dupont, S.: Subband-based speech recognition. In: Proc. IEEE ICASSP. (1997)
18. Potamianos, G., Neti, C., Luettin, J., Matthews, I.: Audio-visual automatic speech recognition: An overview. In Bailly, G., Vatikiotis-Bateson, E., Perrier, P., eds.: Issues in Visual and Audio-Visual Speech Processing. MIT Press (2004)
19. Brand, M.: Coupled hidden markov models for modeling interacting processes. Technical Report 405, MIT Media Lab Vision and Modeling (1996)
20. Bengio, S.: An asynchronous hidden markov model for audio-visual speech recognition. In Becker, S., Thrun, S., Obermayer, K., eds.: Advances in Neural Information Processing Systems 15. (2003)
21. Zhang, D., Gatica-Perez, D., Bengio, S., McCowan, I., Lathoud, G.: Modeling individual and group actions in meetings: a two-layer hmm framework. In: IEEE Workshop on Event Mining at CVPR. (2004)
22. Bourlard, H., Bengio, S., Doss, M.M., Zhu, Q., Mesot, B., Morgan, N.: Towards using hierarchical posteriors for flexible automatic speech recognition systems. In: Proc. of DARPA EARS Rich Transcription Workshop. (2004)
23. Bishop, C.: Neural Networks for Pattern Recognition. Oxford University Press, London, UK (1995)
24. Bengio, Y., Simard, P., Frasconi, P.: Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks **5** (1994) 157–166
25. Pigeon, S., Vandendorpe, L.: The M2VTS multimodal face database (release 1.00). In: Proc. of the Conf. on AVBPA. (1997)
26. Varga, A., Steeneken, H., Tomlinson, M., Jones, D.: The noisex-92 study on the effect of additive noise on automatic speech recognition. Technical report, DRA Speech Research Unit (1992)

Bayesian Kernel Learning Methods for Parametric Accelerated Life Survival Analysis

Gavin C. Cawley[1], Nicola L.C. Talbot[1], Gareth J. Janacek[1], and Michael W. Peck[2]

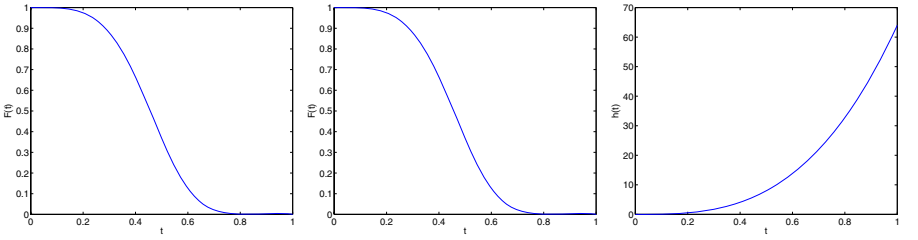
[1] School of Computing Sciences, University of East Anglia, Norwich NR4 7TJ, U.K
{gcc, nlct}@cmp.uea.ac.uk
[2] Institute of Food Research, Norwich Research Park, Norwich NR4 7UA, U.K
mike.peck@bbsrc.ac.uk

Abstract. Survival analysis is a branch of statistics concerned with the time elapsing before "failure", with diverse applications in medical statistics and the analysis of the reliability of electrical or mechanical components. In this paper we introduce a parametric accelerated life survival analysis model based on kernel learning methods that, at least in principal, is able to learn arbitrary dependencies between a vector of explanatory variables and the scale of the distribution of survival times. The proposed kernel survival analysis method is then used to model the growth domain of *Clostridium botulinum*, that is the food processing and storage conditions permitting the growth of this foodborne microbial pathogen, leading to the production of the neurotoxin responsible for botulism. A Bayesian training procedure, based on the evidence framework, is used for model selection and to provide a credible interval on model predictions. The kernel survival analysis models are found to be more accurate than models based on more traditional survival analysis techniques, but also suggest a risk assessment of the foodborne botulism hazard would benefit from the collection of additional data.

1 Introduction

Survival analysis is a field of classical statistics concerned with data recording the time that elapses before the occurrences of a set of point events, known as "failures". Many applications of survival analysis arise in medical studies, for example it might be of interest to model the length of time that patients survive following each of a range of competing treatments for a given illness. Survival analysis requires there to be a well-defined origin at which time $t = 0$, an appropriate scale for measuring time and a unambiguous definition of failure. In the case of our medical example, the origin is defined by the time of treatment, the time scale is measured in days following treatment and failure defined by the patient's death. The remainder of this section is devoted to a brief overview of parametric survival analysis methods, for a full introduction, see Cox and Oakes [1].

J. Winkler, N. Lawrence, M. Niranjan (Eds.): Machine Learning Workshop, LNAI 3635, pp. 37–55, 2005.



(a) (b) (c)

Fig. 1. Example of (a) the survivor function, $F(t)$, (b) the distribution of survival times, $f(t)$, and (c) hazard function, $h(t)$, for a Weibull distribution with parameters $\psi = 2$ and $\kappa = 4$

1.1 Parametric Survival Analysis

Parametric survival analysis aims to determine the optimal parameters of a fixed distribution describing time to failure, T,

$$F(t) = \Pr(T \geq t). \tag{1}$$

Note this differs from the usual statistical convention where cumulative distribution functions are defined in terms of right continuity, i.e. $F(t) = \Pr(T \leq t)$, and hence the probability density function is defined as,

$$f(t) = -F'(t). \tag{2}$$

Another function of interest in survival analysis is the *hazard* function, given by

$$h(t) = \frac{f(t)}{F(t)}, \tag{3}$$

which represents the instantaneous probability of failure at time t, given survival until time t. Examples of the survivor function, distribution of survival times and hazard function for a Weibull distribution are shown in Figure 1. Any parametric distribution over non-negative values of t may be used (in most applications it does not make sense to consider failure before time $t = 0$). Table 1 shows a number of statistical distributions commonly used in parametric survival analysis, where ψ represents the *scale* parameter of the distribution, and κ is used to represent a *shape* parameter. The optimal parameters of the survival distribution are traditionally found via a maximum likelihood approach. Given a dataset $\mathcal{D} = \{t_i\}_{i=1}^{\ell}$, recording the failure time for ℓ discrete events, then assuming that the data represents an independent and identically distributed (i.i.d.) sample from some underlying distribution, the *likelihood* of the data is given by the product of the density function over the observed data, i.e.

$$L = \prod_{i=1}^{n} f(t_i). \tag{4}$$

The optimal parameters are then determined by minimising the negative logarithm of the likelihood function.

Table 1. Statistical distributions commonly encountered in parametric survival analysis

Distribution	**Density Function** $f(t)$	**Survivor Function** $F(t)$
Exponential	$\psi \exp\{-\psi t\}$	$\exp\{-\psi t\}$
Weibull	$\kappa\psi(\psi t)^{\kappa-1}\exp\{-(\psi t)^{\kappa}\}$	$\exp\{-(\psi t)^{\kappa}\}$
Log-logistic	$\kappa\psi^{\kappa}t^{\kappa-1}[1+(\psi t)^{\kappa}]^{-2}$	$[1+(\psi t)^{\kappa}]^{-1}$
Gamma	$\frac{\psi(\psi t)^{\kappa-1}}{\Gamma(\kappa)}\exp\{-\psi t\}$	$\int_t^{\infty} f(w)dw$

1.2 Censoring of Data in Parametric Survival Analysis

In many applications of survival analysis it is not practical to observe every trial until failure occurs, and instead a fixed observation period is imposed. Trials where failure is not observed are said to have been "censored". Returning to our medical example, it is to be expected that not all of the patients will have died by the time the observation period has ended (a period of 5 years is commonly used in defining survival rates for medical procedures), other patients may have died from totally unrelated causes, for instance road traffic accidents, or simply may have moved away and are no longer in contact with the medical institution conducting the study. Clearly, even though the failure time is unknown, censored data should still be included in fitting the survival distribution, as they provide information on an interval of time where failure was *not* observed. It is a simple matter to incorporate censoring into the likelihood function. Uncensored data are handled as before, for censored data, however, all that is known is that the failure will occur at some time greater than the censoring time, so the likelihood for censored observations is given by the survivor function, $F(t)$. The likelihood function then becomes

$$L = \prod_{i\in\mathcal{U}} f(t_i) \prod_{i\in\mathcal{C}} F(t_i). \tag{5}$$

where $\mathcal{U}$ and $\mathcal{C}$ represent the index sets of uncensored and censored observations respectively. Note that the censoring time may vary for each observation, or may be constant for all trials.

A second type of censoring arises where trials are not continuously monitored for failure, but instead are inspected periodically. In this case the exact time of failure is again unknown, only the interval of time between inspections in which the failure occurred, a situation known as *interval censoring*. Let $T_0 = 0, T_1, T_2, \ldots, T_N, T_{N+1} = \infty$ represent the observation times, such that $T_i < T_{i+1},\ \forall\ i \in \{0, 1, 2, \ldots, N\}$, then the likelihood for interval censored data is given by,

$$L = \prod_{i=1}^{n} \left[F\left(T_{t_i - 1}\right) - F\left(T_{t_i}\right) \right], \tag{6}$$

where $t_i \in \{1, 2, \ldots, N+1\}$, now indexes the observation time at which failure was first observed in the i^{th} trial; if no failure is actually observed, then failure is assumed to occur in the interval $(T_N, \ +\infty)$, and so $t_i = N + 1$.

1.3 Accelerated Life Survival Analysis

In most applications of survival analysis, we seek to construct a model that captures some underlying relationship between the distribution of survival times and the values of a set of d explanatory variables, $\{\boldsymbol{x}_i \in \mathcal{X} \subset \mathbb{R}^d\}_{i=1}^{\ell}$. In our medical example, the explanatory variables might include information about dosages, medical treatments applied and information about the patient that may have a bearing on survival. The *accelerated life* model assumes that a baseline survivor function F_0 adequately describes the form of the distribution of survival times, but that a function, $\psi(\boldsymbol{x})$, is required to model variation in the *scale* of the distribution according to the values of the explanatory variables,

$$F(t; \boldsymbol{x}) = F_0(\psi(\boldsymbol{x})t)$$

such that the probability density becomes

$$f(t; \boldsymbol{x}) = \psi(\boldsymbol{x}) f_0(\psi(\boldsymbol{x})t)$$

and the hazard function becomes

$$h(t; \boldsymbol{x}) = \psi(\boldsymbol{x}) h_0(\psi(\boldsymbol{x})t),$$

where f_0 and h_0 represent the corresponding baseline probability density and hazard functions respectively. It is rarely appropriate for the function $\psi(\cdot)$ to adopt negative values, and so the form most commonly encountered in classical survival analysis is given by

$$\psi(\boldsymbol{x}; \boldsymbol{w}, b) = \exp\left\{\boldsymbol{w} \cdot \boldsymbol{x} + b\right\},$$

where the regression coefficients, $(\boldsymbol{w}, b)$ and any shape parameters of the baseline survivor function are optimised by minimisation of the negative logarithm of the likelihood function (4), (5) or (6). The accelerated life model is perhaps the simplest form of covariate modelling used in survival analysis, but is adequate in many applications.

1.4 Kernel Learning Methods in Parametric Survival Analysis

In this paper, we investigate the use of kernel learning methods to construct a more general form of accelerated life survival analysis, such that the function $\psi(\cdot)$ is able to model arbitrary, possibly non-linear dependencies between the

explanatory variables and the scale of the survivor function. The "kernel trick" allows a wide range of functional forms to be easily investigated, while the use of regularisation provides a simple means of controlling model complexity and so avoiding over-fitting. Section 2 describes a simple kernel method for covariate modelling in accelerated life survival analysis. This is extended in section 3 to support Bayesian selection of the usual kernel and regularisation parameters, based on the *evidence framework* of MacKay [2,3,4], which is also used to provide a credible interval on model predictions. Section 4 describes the use of conventional and kernel accelerated life survival analysis in modelling the growth domain of the foodborne microbial pathogen *Clostridium botulinum*, that is the food processing steps and incubation conditions that support growth. Accurate modelling of the growth domain of microbial pathogens is a vital step in ensuring the safety of food. Finally, the work is summarised in section 5.

2 Kernel Accelerated Life Survival Analysis

Kernel learning methods typically aim to construct a linear model $\psi(\boldsymbol{x};\boldsymbol{w},b) = s\{\boldsymbol{w}\cdot\boldsymbol{\phi}(\boldsymbol{x})+b\}$, where $s\{\cdot\}$ is a scalar function, in a feature space $\mathcal{F}$ given by a fixed transformation of the input space, $\boldsymbol{\phi}(\boldsymbol{x}):\mathcal{X}\rightarrow\mathcal{F}$. The optimal model parameters, $(\boldsymbol{w},b)$, are given by the minimum of a regularised [5] point-wise loss function,

$$L(\boldsymbol{w},b;\zeta) = \sum_{i=1}^{\ell} C\{t_i,\psi(\boldsymbol{x}_i;\boldsymbol{w},b)\} + \frac{\mu}{2}\|\boldsymbol{w}\|^2, \tag{7}$$

where μ is a regularisation parameter controlling the bias-variance trade-off [6] and $C\{\cdot,\cdot\}$ is a convex loss function. Rather than specifying the transformation $\boldsymbol{\phi}(\cdot)$ explicitly, a kernel function $\mathcal{K}:\mathcal{X}\times\mathcal{X}\rightarrow\mathbb{R}$ is used, which defines the inner product between vectors in the feature space $\mathcal{F}$, i.e. $\mathcal{K}(\boldsymbol{x},\boldsymbol{x}') = \boldsymbol{\phi}(\boldsymbol{x})\cdot\boldsymbol{\phi}(\boldsymbol{x}')$. $\mathcal{K}$ may be any kernel function, such that the Gram or kernel matrix, $\boldsymbol{K} = [k_{ij} = \mathcal{K}(\boldsymbol{x}_i,\boldsymbol{x_j})]_{i,j=1}^{\ell}$ is positive semi-definite [7,8]. For a more detailed introduction to kernel learning methods, see Cristianini and Shawe-Taylor [9] or Schölkopf and Smola [10]. In this study, we adopt the anisotropic Gaussian radial basis function (RBF) kernel

$$\mathcal{K}(\boldsymbol{x},\boldsymbol{x}') = \exp\left\{-\sum_{i=1}^{d}\eta_i|x_i-x_i'|^2\right\} \tag{8}$$

in which case $\mathcal{F}$ consists of one quadrant of an infinite-dimensional unit hypersphere, and the polynomial kernel

$$\mathcal{K}(\boldsymbol{x},\boldsymbol{x}') = (\boldsymbol{x}\cdot\boldsymbol{x}'+1)^p, \tag{9}$$

in which case the feature space is the space of all monomials of order p or less. Here, $\boldsymbol{\eta}$ and p are examples of *kernel parameters*, which must also be estimated

from the training data. The representer theorem [11,12] states that the minimiser of a loss function of the form of (7) can be written as

$$\boldsymbol{w} = \sum_{i=1}^{\ell} \alpha_i \boldsymbol{\phi}(\boldsymbol{x}_i),$$

such that the output of the model is given by an expansion over evaluations of the kernel function,

$$\psi(\boldsymbol{x}; \boldsymbol{\alpha}, b) = s\left\{\sum_{i=1}^{\ell} \alpha_i \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}) + b\right\}. \tag{10}$$

The point-wise regularised loss function (7) can then be written in terms of the *dual* parameters as

$$L(\boldsymbol{\alpha}, b; \zeta) = \sum_{i=1}^{\ell} C\{t_i, \psi(\boldsymbol{x}_i; \boldsymbol{\alpha}, b)\} + \frac{\mu}{2}\boldsymbol{\alpha}^T \boldsymbol{K}\boldsymbol{\alpha}. \tag{11}$$

As a concrete example, we will consider the case of accelerated life survival analysis using the exponential survivor function and assuming simple right censoring; the loss is then defined by the negative log-likelihood,

$$C\{t_i, \psi_i\} = \begin{cases} \psi_i t_i - \log\{\psi_i\} & i \in \mathcal{U} \\ \psi_i t_i & i \in \mathcal{C} \end{cases},$$

where $\psi_i = \psi(\boldsymbol{x}_i; \boldsymbol{\alpha}, b)$. In this case, $s(\cdot)$ is taken to be the exponential function, such that

$$\log\{\psi(\boldsymbol{x}; \boldsymbol{\alpha}, b)\} = \sum_{i=1}^{\ell} \alpha_i \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}) + b. \tag{12}$$

The optimal model parameters, $\boldsymbol{\omega} = (\boldsymbol{\alpha}, b)$, can then be found straight-forwardly using a simple second-order gradient descent optimisation procedure (Newton's method) [13]. Let $\boldsymbol{g}$ and $\boldsymbol{A}$ represent the gradient vector and Hessian matrix of L with respect to the model parameters respectively,

$$\boldsymbol{g} = \left(g_i = \frac{\partial L}{\partial \omega_i}\right)_{i=1}^{\ell} \quad \text{and} \quad \boldsymbol{A} = \left[a_{ij} = \frac{\partial^2 L}{\partial \omega_i \partial \omega_j}\right]_{i,j=1}^{\ell},$$

the model parameters are the iteratively optimised according to the following update rule,

$$\boldsymbol{\omega}^{\text{new}} = \boldsymbol{\omega}^{\text{old}} - \boldsymbol{A}^{-1}\boldsymbol{g}.$$

If a more complex survivor function is required, for instance a Weibull or log-logistic model, the additional shape parameter, κ, must also be estimated by minimising the negative log-likelihood.

2.1 Imposing Sparsity

The kernel accelerated life model (12) is normally fully dense in the sense that all of the coefficients, $\boldsymbol{\alpha}$, assume non-zero values. This is undesirable as the computational complexity of the training algorithm is $\mathcal{O}(\ell^3)$ operations and has memory requirements of order $\mathcal{O}(\ell^2)$. The Gram matrix, $\boldsymbol{K}$, for a radial basis function kernel is at least in principle positive definite and of full rank, assuming that $\boldsymbol{x}_i \neq \boldsymbol{x}_j,\ \forall\ i, j \in \{1, 2, \ldots, \ell\}$ [14]; however it is possible for $\boldsymbol{K}$ to be *numerically* rank-deficient. We therefore use the incomplete Cholesky factorisation with symmetric pivoting, due to Fine and Scheinberg [15], to find the Cholesky factor of $\hat{\boldsymbol{K}}$, a numerically full-rank symmetric sub-matrix of $\boldsymbol{K}$ [16]. Without loss of generality, we assume that only the first N columns of $\boldsymbol{K}$ contribute to forming $\hat{\boldsymbol{K}}$. The remaining columns of $\boldsymbol{K}$ are then linearly dependent, or very close to being linearly dependent, on columns $1, 2, \ldots, N$, and can be safely deleted prior to training without significantly affecting model performance [17]. We then obtain a sparse kernel model,

$$\log\{\psi(\boldsymbol{x}; \boldsymbol{\beta}, b)\} = \sum_{i=1}^{N} \beta_i \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}) + b. \tag{13}$$

Maximum likelihood estimates of the model parameters, $\boldsymbol{\omega} = (\boldsymbol{\beta}, b)$, are obtained by minimising a regularised loss function,

$$L(\boldsymbol{\beta}, b; \mu) = \sum_{i=1}^{\ell} C\{t_i, \psi(\boldsymbol{x}_i; \boldsymbol{\beta}, b)\} + \frac{\mu}{2} \boldsymbol{\beta}^T \hat{\boldsymbol{K}} \boldsymbol{\beta}, \tag{14}$$

again using Newton's method. The degree of sparsity achieved depends on the value(s) of the scale parameter(s) of the RBF kernel and on the number of input features. Sparsity is greatest for very smooth kernels, and for a small number of input features, as both of these factors tend to restrict the distribution of training points over the unit hyper-sphere formed in the feature space.

2.2 Model Selection

The task of selecting the optimal values for kernel and regularisation parameters, in this case $\boldsymbol{\eta}$ or p and μ respectively, is known as model selection. Rather than performing model selection via lengthy manual trial-and-error experimentation, we would prefer an automated model selection strategy that does not require significant intervention by a skilled operator. Cross-validation [18] provides a simple means of estimating the value of a given performance criterion on unseen data. Under k-fold cross-validation, the training data are partitioned into k disjoint sets. A model is trained using a combination of $k-1$ subsets and the test statistic evaluated on the unused partition. This process is repeated for all k combinations of $k-1$ subsets. The sum of the test statistics for each of the k trials is then taken as an estimate of generalisation performance. The most extreme form of cross-validation, where each partition contains only one pattern, i.e. $k = \ell$, is known

as leave-one-out cross-validation [19]. In this study, the kernel and regularisation parameters are selected by minimising a 10-fold cross-validation estimate of an appropriate negative log-likelihood statistic using the Nelder-Mead simplex optimisation algorithm [20]. The leave-one-out cross-validation estimate is then used for model comparison purposes.

3 Bayesian Kernel Survival Analysis

In this section, we apply the evidence framework of MacKay [2,3,4] to develop a hierarchical Bayesian formulation of the sparse kernel accelerated life survival analysis model. Minimising the regularised loss function (14) is equivalent to maximising the posterior distribution

$$\mathrm{p}(\boldsymbol{\beta}|\mathcal{D};\mu) = \frac{\mathrm{p}(\mathcal{D}|\boldsymbol{\beta})p(\boldsymbol{\beta};\mu)}{\mathrm{p}(\mathcal{D})}. \tag{15}$$

Here $\mathrm{p}(\mathcal{D}|\boldsymbol{\beta})$ represents the likelihood of the training data,

$$\mathrm{p}(\mathcal{D}|\boldsymbol{\beta}) = \frac{1}{Z_{\mathcal{D}}} \exp\left\{-E_{\mathcal{D}}\right\},$$

where $Z_{\mathcal{D}}$ is an appropriate normalising constant and $E_{\mathcal{D}}$ measures the "data misfit",

$$E_{\mathcal{D}} = \sum_{i=1}^{\ell} C\{t_i, \psi(\boldsymbol{x}_i;\boldsymbol{\beta},b)\}.$$

For kernel survival analysis, the data misfit is simply the negative logarithm of the likelihood given by (4), (5) or (6) depending on the censoring regime in operation. The prior over model parameters, $p(\boldsymbol{\beta};\mu)$, is then a multivariate Gaussian distribution,

$$\mathrm{p}(\boldsymbol{\beta};\mu) = \frac{1}{Z_{\mathcal{W}}} \exp\left\{-\frac{\mu}{2}E_{\mathcal{W}}\right\},$$

where again $Z_{\mathcal{W}}$ is an appropriate normalising constant and $E_{\mathcal{W}}$ is the regularisation term in the regularised loss function,

$$E_{\mathcal{W}} = \boldsymbol{\beta}^T \hat{\boldsymbol{K}} \boldsymbol{\beta}.$$

Note that, unlike the weight decay prior normally encountered in Bayesian treatments of multi-layer perceptron networks [21,2,3,4,22,23], the prior is non-spherical. The Taylor expansion of $L(\boldsymbol{\beta},b;\mu)$ around the most probable value, $\boldsymbol{\beta}^{\mathrm{MP}}$, gives rise to familiar Gaussian approximation to the posterior distribution over $\boldsymbol{\beta}$, known as the "Laplace approximation",

$$\mathrm{p}(\boldsymbol{\beta}|\mathcal{D}) \approx \frac{1}{Z^*} \exp\left\{-L\left(\boldsymbol{\beta}^{\mathrm{MP}}\right) - \frac{1}{2}\Delta\boldsymbol{\beta}^T \boldsymbol{A} \Delta\boldsymbol{\beta}\right\}, \tag{16}$$

where Z^* is an normalising constant, $\Delta\boldsymbol{\beta} = \boldsymbol{\beta} - \boldsymbol{\beta}^{\mathrm{MP}}$ and $\boldsymbol{A}$ is the Hessian of $L(\boldsymbol{\beta},b;\zeta)$ with respect to $\boldsymbol{\beta}$ evaluated at $\boldsymbol{\beta}^{\mathrm{MP}}$. For further details, see e.g. Bishop [22].

3.1 The Evidence Approximation for μ

The evidence approximation [2,3,4] assumes that the posterior distribution for the regularisation parameter, $p(\mu|\mathcal{D})$, is sharply peaked about its most probable value, μ^{MP}, suggesting the following approximation to the posterior distribution for $\boldsymbol{\beta}$,

$$\mathrm{p}(\boldsymbol{\beta}|\mathcal{D}) = \int \mathrm{p}(\boldsymbol{\beta}|\mu, \mathcal{D})\mathrm{p}(\mu|\mathcal{D})d\mu \approx \mathrm{p}(\boldsymbol{\beta}|\mu^{\mathrm{MP}}, \mathcal{D}).$$

Thus, rather than integrate out the regularisation parameter entirely (e.g. Buntine and Weigend [21]), we simply proceed with the analysis using the regularisation parameter fixed at its most likely value. For a discussion of the validity of this approach, see MacKay [24]. We seek therefore to maximise the posterior distribution,

$$\mathrm{p}(\mu|\mathcal{D}) = \frac{\mathrm{p}(\mathcal{D}|\mu)\mathrm{p}(\mu)}{\mathrm{p}(\mathcal{D})}.$$

If the prior, $\mathrm{p}(\mu)$ is relatively insensitive to the value μ, then maximising the posterior is approximately equivalent to maximising the likelihood term, $\mathrm{p}(\mathcal{D}|\mu)$, known as the *evidence* for μ. Adopting the Gaussian approximation to the the posterior for the model parameters, the log-evidence is given by

$$\log \mathrm{p}(\mathcal{D}|\mu) = -E_{\mathcal{D}}^{\mathrm{MP}} - \mu E_{\mathcal{W}}^{\mathrm{MP}} - \frac{1}{2}\log|\boldsymbol{A}| + \frac{N}{2}\log\mu. \tag{17}$$

3.2 Update Formula for the Regularisation Parameter

As a consequence of the non-spherical prior over model parameters, the Hessian matrix for a *sparse* kernel survival analysis model is of the form

$$\boldsymbol{A} = \boldsymbol{H} + \mu\hat{\boldsymbol{K}},$$

where $\boldsymbol{H}$ is the Hessian of $E_{\mathcal{D}}$ with respect to $\boldsymbol{\beta}$. Unfortunately this complicates the derivation of an efficient update formula used to select the best value for the regularisation parameter, μ. We therefore seek to re-parameterise the model such that the anisotropic Gaussian prior over the coefficients of the sparse kernel expansion is replaced by an isotropic Gaussian prior over the transformed parameters, i.e.

$$E_{\mathcal{W}} = \boldsymbol{\beta}^T\hat{\boldsymbol{K}}\boldsymbol{\beta} = \tilde{\boldsymbol{\beta}}^T\tilde{\boldsymbol{\beta}},$$

where $\tilde{\boldsymbol{\beta}}$ is the vector of transformed parameters. Let $\hat{\boldsymbol{G}}$ represent the upper triangular Cholesky factor [16] of the symmetric positive-definite matrix $\hat{\boldsymbol{K}}$, such that $\hat{\boldsymbol{K}} = \hat{\boldsymbol{G}}^T\hat{\boldsymbol{G}}$. By inspection, the desired parameterisation is given then by

$$\tilde{\boldsymbol{\beta}} = \hat{\boldsymbol{G}}\boldsymbol{\beta} \quad \Longrightarrow \quad \boldsymbol{\beta} = \hat{\boldsymbol{G}}^{-1}\tilde{\boldsymbol{\beta}}.$$

The optimisation criterion then becomes

$$L(\tilde{\boldsymbol{\beta}}, b; \mu) = \sum_{i=1}^{\ell} C\left\{t_i, \psi\left(\boldsymbol{x}_i; \hat{\boldsymbol{G}}^{-1}\tilde{\boldsymbol{\beta}}, b\right)\right\} + \frac{\mu}{2}\tilde{\boldsymbol{\beta}}^T\tilde{\boldsymbol{\beta}} \tag{18}$$

The Hessian of L with respect to $\tilde{\boldsymbol{\beta}}$ is then of the form

$$\tilde{\boldsymbol{A}} = \tilde{\boldsymbol{H}} + \mu \boldsymbol{I},$$

where $\tilde{\boldsymbol{H}}$ is the Hessian of $E_{\mathcal{D}}$ with respect to $\tilde{\boldsymbol{\beta}}$. If the eigenvalues of $\tilde{\boldsymbol{H}}$ are $\lambda_1, \lambda_2, \ldots, \lambda_W$, then the eigenvalues of $\tilde{\boldsymbol{A}}$ are $(\lambda_1 + \mu), (\lambda_2 + \mu), \ldots, (\lambda_W + \mu)$. The derivative of $\log |\tilde{\boldsymbol{A}}|$ with respect to μ (assuming that the eigenvalues of $\tilde{\boldsymbol{H}}$ are independent of μ) is then given by

$$\frac{d}{d\mu} \log |\tilde{\boldsymbol{A}}| = \frac{d}{d\mu} \log \left\{ \prod_{i=1}^{N} (\lambda_i + \mu) \right\} = \sum_{i=1}^{N} \frac{1}{\lambda_i + \mu}.$$

Setting the derivative of the log-evidence with respect to μ to zero, we have that

$$2\mu E_{\mathcal{W}}^{\mathrm{MP}} = N - \sum_{i=1}^{N} \frac{\mu}{\lambda_i + \mu} = \sum_{i=1}^{N} \frac{\lambda_i}{\lambda_i + \mu} = \gamma,$$

where γ is the number of well determined parameters in the model. This leads to a simple update formula for the regularisation parameter:

$$\mu^{\mathrm{new}} = \frac{\gamma}{2E_{\mathcal{W}}^{\mathrm{MP}}}. \tag{19}$$

The training procedure then alternates between updates of the primary model parameters using the Newton's method and updates of the regularisation parameter according to equation (19). In practise, rather than using the transformed parameters only in computing the number of well-determined parameters, the entire training procedure is most easily conducted in the transformed parameters, $\tilde{\boldsymbol{\beta}}$, and the original model parameters, $\boldsymbol{\beta}$, reclaimed afterwards. The values of kernel parameters can then be optimised by maximising the log-evidence for model, $\mathcal{H}_i$,

$$\log p(\mathcal{D}|\mathcal{H}_i) = \mu^{\mathrm{MP}} E_{\mathcal{W}}^{\mathrm{MP}} - E_{\mathcal{D}}^{\mathrm{MP}} - \frac{1}{2} \log |\boldsymbol{A}| + \frac{N}{2} \log \mu^{\mathrm{MP}} + \frac{1}{2} \log \left\{ \frac{2}{\gamma} \right\}, \tag{20}$$

where in this case $\mathcal{H}_i$ specifies the kernel function (see MacKay [2] or Bishop [22] for further details).

4 Modelling Growth Domain of Food Borne Microbial Pathogens

Clostridium botulinum is an anaerobic bacterium that produces one of the most powerful toxins known to science as a by-product of its growth processes. Ingestion of only 30ng of the toxin can result in severe illness and even death [25]. It is therefore vital that steps should be taken to ensure that the toxin is not present in food. As *C. botulinum* spores are ubiquitous in the raw ingredients,

food must be processed to ensure that all of these spores are destroyed, or so that the spores are prevented from germinating, leading to cell division and subsequent toxin production. Growth of *C. botulinum* is, in most cases, principally dependent on environmental factors such as temperature, pH, NaCl concentration and gas atmosphere. It is important then to be able to define the "growth domain", that is the conditions under which the spores are able to germinate, giving rise to toxin production. This is especially true in the case of minimally processed chilled foods, as non-proteolytic *C. botulinum* is capable of growth and toxin production at chill temperatures. The safety of these foods with respect to non-proteolytic *C. botulinum* is likely to rely on a combination of heat treatment and subsequent incubation at refrigeration temperatures (Lund and Peck [25], Peck [26]). In this section, we describe a growth domain model for *C. botulinum* based on Bayesian kernel accelerated life survival analysis.

4.1 The Dataset

The growth domain models for *C. botulinum* described here are based on the dataset described in Fernández and Peck [27]. Tubes containing a sterile meat-based medium (Peck *et al.* [28]), were inoculated with a suspension of the spores of eight strains of non-proteolytic *C. botulinum*, at a final concentration of 10^6 spores per tube, and subjected to a range of heat treatments, shown in table 2. The tubes were then cooled and incubated at temperatures of 5, 8, 12, 16 and 25°C for 90 days. The temperature of the heat treatment applied is represented by T_c and the duration by t_c, T_i and t are used to represent the incubation temperature and incubation time respectively. Five replicates were performed at each incubation temperature, for each heat treatment regime. The tubes were inspected daily for signs of growth, as indicated by obvious formation of gas, during the early stages of incubation and every 2–3 days during later stages. At the end of the experiment, samples from each heat treatment regime, showing growth at the lowest incubation temperature and for the highest incubation temperature that did not show growth, were tested for toxin (Peck *et al.* [28], Stringer *et al.* [29], Carlin and Peck [30]). This type of dataset is known as *time to growth* data, as the results are presented in terms of a table showing the number of days elapsing before growth is first observed in each tube. Full details of the experimental method are recorded in Fernández and Peck [27].

4.2 Objective Analysis of Model Performance

The first step in evaluating the kernel growth domain models is to compare survivor functions, censoring schemes, kernel functions and model selection strategies. In this study, we consider the exponential, Weibull and log-logistic survivor functions, with right censoring, where trials are censored only if no growth is observed within the 90 day period of the experiment, and interval censoring schemes. In this case, the tubes were not constantly monitored for growth, but were inspected at irregular intervals, and so an interval censoring scheme is the more intuitively appealing option as it incorporates the uncertainty relating to

the exact time of failure in each trial. Linear ($p = 1$), quadratic ($p = 2$) and cubic ($p = 3$) variants of the polynomial kernel were investigated, giving rise to models with similar complexity to the polynomial regression models typically used in classical survival analysis. The radial basis function kernel was also used to determine whether the use of a more powerful regression component of the model can be justified. Finally, three model selection strategies were evaluated:

- **Bayesian** - Both regularisation and kernel parameters are selected via minimisation of the negative logarithm of the Bayesian evidence, given by equations (17) and (20) respectively.
- **Cross-validation** - Both regularisation and kernel parameters selected via minimisation of a 10-fold cross-validation estimate of the negative log-likelihood, as described in section 2.2. Due to the strong correlation in time-to-growth for trials sharing a common heat treatment and incubation temperature, the five replicates for a given combination of heat treatment and incubation temperature are always assigned to the same partition.
- **Hybrid** - The regularisation parameter, for which an efficient update formula is available, is selected according to the Bayesian evidence (17), but the kernel parameters chosen according to a 10-fold cross-validation estimate of the negative log-likelihood.

As the experimental data resulted from a periodic inspection schedule, we adopt a cross-validation estimate of the negative logarithm of the likelihood assuming an interval censoring regime. As 10-fold cross-validation is used in model selection, and since the dataset is quite small, a leave-one-experiment-out cross-validation scheme is used for model comparison purposes, where at each iteration, the test partition consists of the five replicates for a single combination of heat treatment and incubation temperature. Table 3 shows the test statistic for

Table 2. Heat treatments applied to a meat-based medium containing spores of *Clostridium botulinum*

Temperature (°C)	Duration (min)
70°C	104.9 529.1 998.9 1596.3 2065.9 2544.5
75°C	284.6 463.1 734.2 1071.5 1376.5 1793.0
80°C	11.4 69.7 98.0 127.9 183.8 229.6 294.9 362.7
85°C	23.3 35.7 52.0 57.8 83.8
90°C	10.3 10.9 15.3 23.5 33.5

a range of kernel functions, survival distributions and model selection strategies. The results seem to justify use of a quadratic or cubic polynomial or radial basis function kernel, rather than a simple linear model. The use of a two-parameter Weibull or log-logistic survivor function, rather than the exponential distribution, is also justified although the difference in performance between two-parameter distributions is slight. An interval censoring scheme is marginally better than simple right censoring in most cases. The Bayesian model selection scheme generally favours somewhat overly-regularised models and so performs rather worse than a 10-fold cross-validation scheme. This is possibly because the level of Bayesian inference used to select the kernel parameters is based on several simplifying approximations, which may significantly degrade performance. It is argued in the next section that it is likely that insufficient data is available to generate a truly reliable statistical model, and so there is also insufficient data to unambiguously differentiate between relatively similar models based on an objective performance criterion.

Table 3. Leave-one-experiment-out estimates of the negative log-likelihood of parametric accelerated life survival analysis models of the growth domain of *C. botulinum*, based on kernel learning methods incorporating a range of survival distributions, kernel functions and model selection procedures. Figures displayed in bold font represent the best value in each column, underlined figures represent the best value in each row.

Kernel	Model	Exponential		Weibull		Log-logistic	
	Selection	Right	Interval	Right	Interval	Right	Interval
Linear	Bayesian	1581.8	1585.5	1570.6	1572.6	1553.7	1586.2
	Cross-Val	1581.6	1583.9	1569.8	1570.6	1550.1	1550.9
Quadratic	Bayesian	1240.9	1239.1	1186.5	1190.0	1177.0	1178.0
	Cross-Val	1240.1	1238.4	1185.2	1187.9	1175.5	1175.0
Cubic	Bayesian	1234.6	1229.1	1190.3	1182.6	1182.2	1155.9
	Cross-Val	1239.7	1223.3	1195.1	1171.3	1157.4	1134.9
RBF	Bayesian	1207.5	1208.3	1191.6	1197.3	1260.6	1134.6
	Hybrid	1201.0	1202.9	1213.1	1425.2	1130.7	1111.6
	Cross-Val	**1198.0**	**1193.1**	**1147.4**	**1154.2**	**1116.4**	**1096.2**

For comparison, table 4 shows the negative log-likelihood for the set of "optimal" accelerated life survival analysis models, where a single shape parameter (if required) is estimated for the entire dataset, but a separate scale parameter for each set of five replicates sharing a common heat treatment and incubation temperature. This gives the irreducible component of the negative log-likelihood statistic, which depends on the inherent variability of the data, and so provides

Table 4. Minimum possible negative log-likelihood for the *C. botulinum* growth-domain dataset, assuming interval censoring

Distribution	$-\log \mathcal{L}$
exponential	1084.9
Weibull	826.5
log-logistic	817.1

a lower bound on the negative log-likelihood. Again the results provide good evidence for the use of a two-parameter survivor function, but only marginal evidence for the log-logistic over the Weibull distribution.

4.3 Subjective Analysis of Model Performance

Figure 2 shows contour plots of the probability of growth as a function of incubation temperature, T_i, and incubation time, t, for two heat treatment regimes, generated by a kernel accelerated life survival analysis model, with Bayesian model selection, radial basis function kernel and the log-logistic survivor function with interval censoring. Figure 2 (a) shows an example of a heat treatment regime for which the kernel survival analysis performs well; the experimental data lie principally within the contours depicting a probability of growth between 0.1 and 0.9. The key advantage of a parametric survival analysis is that it provides a reliable means of defining the growth domain at an arbitrary probability level, in this case shown by the contour at a probability level of 0.001. The only extrapolation here involves only the survivor function, not the regression part of the model, and so the properties of the growth domain are well understood.

Figure 2 (b) shows an example of a heat treatment regime where the model performs relatively poorly, with a significant proportion of the experimental data lying below the contour denoting a probability of growth of 0.1. Figure 3 (a) shows the cumulative probability of growth for this heat treatment regime, for an incubation temperature of 25°C; it can easily be seen that the experimental data lie within the tail of the survival distribution. Figure 3 (b) shows the corresponding cumulative probability plot for the "optimal" accelerated life model, importantly this shows that an accelerated life model is capable of modelling the data accurately. The failure of the kernel survival analysis model for this heat treatment regime must therefore be due to the regression part of the model. In this study, we have used a radial basis function kernel, which in principle is able to form highly complex non-linear models, however both Bayesian and cross-validation based model selection schemes have been employed to ensure that the kernel survival analysis model is no more complex than is justified by the available data. The best generalisation performance is obtained using a relatively smooth regression model, giving rise to a rather broad survival distribution for the majority of heat treatment regimes and incubation temperatures. Our interpretation of this result is that, with only 154 combinations of heat treatment

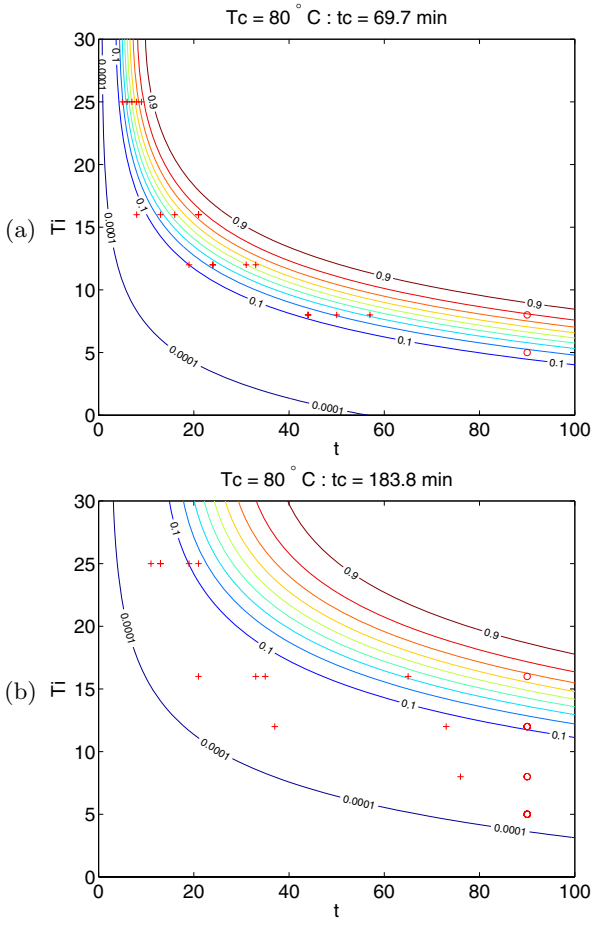


Fig. 2. Contour plots of the probability of growth as a function of the incubation temperature, T_i and incubation time, t, for two heat treatment regimes. The growth domain can be defined at an arbitrary probability level, in this case $Pr(T < t) = 0.001$. Experimental data where growth was observed are shown as crosses, censored data, where growth was not observed by the 90^{th} day are shown as circles.

and incubation temperature, there are simply not enough experimental data to adequately describe the influence of heat treatment and incubation conditions on the parameters of the survivor function. Obviously, this may have significant implications on statistical risk assessment of the hazards associated with foodborne microbial pathogens.

4.4 Credible Interval on Model Predictions

Figure 4 shows a plot of the cumulative probability of growth as a function of incubation time, t, for a set of experimental conditions where the model performs

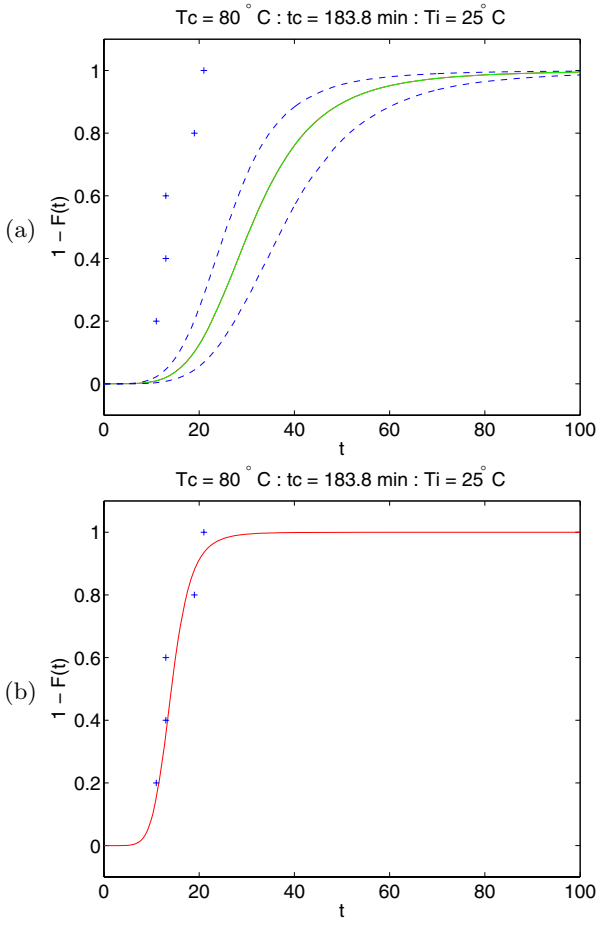


Fig. 3. Plots of the cumulative probability of growth as a function of incubation time, t, for a heat treatment regime and incubation temperature where the kernel survival analysis model performs relatively poorly (a) and equivalent results for the "optimal" accelerated life model (b). Experimental data where growth was observed during the trial period are depicted by crosses, the dashed lines in (a) delimit a credible interval on model predictions.

reasonably well. The figure also demonstrates a Bayesian credible interval around the most probable cumulative probability function. One thousand samples were drawn from the posterior distribution over all model parameters, $(\boldsymbol{\beta}, b, \kappa)$. The credible interval is formed from the 5^{th} and 95^{th} centiles of the cumulative distribution functions given by the output of the models corresponding to each of these samples. Note that the upper and lower limits of the credible interval are not symmetric. The Bayesian credible interval on model prediction provides an indication of the uncertainty inherent in determination of the model parameters. The upper limit of the credible interval provides a "conservative" model that

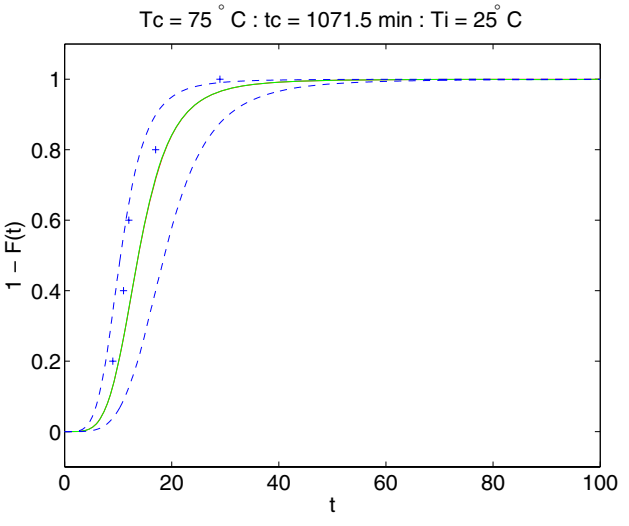


Fig. 4. Plot of the cumulative probability of growth as a function of incubation time, t, for a particular heat treatment regime and incubation temperature, experimental data where growth was observed during the trial period are depicted by crosses. Note the Bayesian credible interval on model predictions, shown by the dashed lines.

could be used in safety-critical applications. In this case, the contour defining the growth domain could be loosely interpreted as indicating that there is a 95% probability that the probability of growth is less than 0.1%. This may be a useful feature for public communication of confidence in the prediction of the model.

5 Summary

In this paper, we have proposed an accelerated life survival analysis model based on Bayesian kernel learning methods, providing a statistically sound means of modelling survival data, allowing for an arbitrary relationship between the explanatory variables and the scale parameter of the survivor function. Bayesian and cross-validation model selection strategies provide reliable means of selecting the regularisation and kernel parameters such that the model is no more complex than is justified by the training data. The Bayesian interpretation also provides a useful credible interval on model predictions. The kernel survival analysis procedure was then used to model the growth domain of the foodborne microbial pathogen *Clostridium botulinum*. It was found that the use of a more powerful non-linear method for analysis of the data than is normally applied to this type of dataset reveals that the experimental data is probably insufficient to form a statistical model that both generalises well and provides an adequate subjective fit to the data. This study therefore supports the collection of more substantial datasets in this field for more reliable risk analysis of the hazards associated with foodborne microbial pathogens. As the collection of data is a time-consuming

and expensive procedure, we aim to use active learning to concentrate further experimental work in areas where the uncertainty of the model is greatest.

Acknowledgements

This work was supported by the Biotechnology and Biological Sciences Research Council (grant number 83/D17534 to GCC and competitive strategic grant funding to MWP).

References

1. Cox, D.R., Oakes, D.: Analysis of Survival Data. Volume 21 of Monographs on Statistics and Applied Probability. Chapman and Hall (1984)
2. MacKay, D.J.C.: Bayesian interpolation. Neural Computation **4** (1992) 415–447
3. MacKay, D.J.C.: A practical Bayesian framework for backprop networks. Neural Computation **4** (1992) 448–472
4. MacKay, D.J.C.: The evidence framework applied to classification networks. Neural Computation **4** (1992) 720–736
5. Tikhonov, A.N., Arsenin, V.Y.: Solutions of ill-posed problems. John Wiley, New York (1977)
6. Geman, S., Bienenstock, E., Doursat, R.: Neural networks and the bias/variance dilemma. Neural Computation **4** (1992) 1–58
7. Mercer, J.: Functions of positive and negative type and their connection with the theory of integral equations. Philosophical Transactions of the Royal Society of London, A **209** (1909) 415–446
8. Aronszajn, N.: Theory of reproducing kernels. Transactions of the American Mathematical Society **68** (1950) 337–404
9. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines (and other kernel-based learning methods). Cambridge University Press, Cambridge, U.K. (2000)
10. Schölkopf, B., Smola, A.J.: Learning with kernels - support vector machines, regularization, optimization and beyond. MIT Press, Cambridge, MA (2002)
11. Kimeldorf, G.S., Wahba, G.: Some results on Tchebycheffian spline functions. J. Math. Anal. Applic. **33** (1971) 82–95
12. Schölkopf, B., Herbrich, R., Smola, A.J.: A generalised representer theorem. In: Proceedings of the Fourteenth International Conference on Computational Learning Theory, Amsterdam, the Netherlands (2001) 416–426
13. Fletcher, R.: Practical Methods of Optimization. second edn. John Wiley and Sons (2000)
14. Micchelli, C.A.: Interpolation of scattered data: Distance matrices and conditionally positive definite functions. Constructive Approximation **2** (1986) 11–22
15. Fine, S., Scheinberg, K.: Efficient SVM training using low-rank kernel representations. Journal of Machine Learning Research **2** (2001) 243–264
16. Golub, G.H., Van Loan, C.F.: Matrix Computations. third edition edn. The Johns Hopkins University Press, Baltimore (1996)
17. Baudat, G., Anouar, F.: Kernel-based methods and function approximation. In: Proc. IJCNN, Washington, DC (2001) 1244–1249

18. Stone, M.: Cross-validatory choice and assessment of statistical predictions. Journal of the Royal Statistical Society **B 36** (1974) 111–147
19. Luntz, A., Brailovsky, V.: On estimation of characters obtained in statistical procedure of recognition (in Russian). Techicheskaya Kibernetica **3** (1969)
20. Nelder, J.A., Mead, R.: A simplex method for function minimization. Computer Journal **7** (1965) 308–313
21. Buntine, W.L., Weigend, A.S.: Bayesian back-propagation. Complex Systems **5** (1991) 603–643
22. Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press (1995)
23. Neal, R.M.: Bayesian learning for neural networks. Lecture Notes in Statistics. Springer (1996)
24. MacKay, D.J.C.: Hyperparameters : optimise or integrate out? In Heidbreder, G., ed.: Maximum Entropy and Bayesian Methods. Kluwer (1994)
25. Lund, B.M., Peck, M.W.: Clostridium botulinum. In Lund, B.M., Baird-Parker, A.C., Gould, G.W., eds.: The Microbiological Safety and Quality of Food. Aspen, Gaithersburg, USA (2000) 1057–1109
26. Peck, M.W.: *Clostridium botulinum* and the safety of refrigerated processed foods of extended durability. Trends in Food Science and Technology **8** (1997) 186–192
27. Fernández, P.S., Peck, M.W.: A predictive model that describes the effect of prolonged heating at $70 - 80^{\circ}$ C and incubation at refrigeration temperatures on growth and toxigenesis by non-proteolytic *Clostridium botulinum*. Journal of Food Protection **60** (1997) 1064–1071
28. Peck, M.W., Lund, B.M., Fairbairn, D.A., Kassperson, A.S., Undeland, P.C.: Effect of heat treatment on survival of, and growth from, spores of non-proteolytic *Clostridium botulinum* at refrigeration temperatures. Applied and Environmental Microbiology **61** (1995) 1780–1785
29. Stringer, S.C., Haque, N., Peck, M.W.: Growth from spores of non-proteolytic *Clostridium botulinum* in heat treated vegetable juice. Applied and Environmental Microbiology **65** (1999) 2136–2142
30. Carlin, F., Peck, M.W.: Growth and toxin production by non-proteolytic and proteolytic *Clostridium botulinum* in cooked vegetables. Letters in Applied Microbiology **20** (1995) 152–156

Extensions of the Informative Vector Machine

Neil D. Lawrence[1], John C. Platt[2], and Michael I. Jordan[3]

[1] Department of Computer Science, University of Sheffield, Regent Court, 211 Portobello Street, Sheffield S1 4DP, U.K.
neil@dcs.shef.ac.uk
[2] Microsoft Research, Microsoft Corporation, One Microsoft Way, Redmond, WA 98052, U.S.A.
jplatt@microsoft.com
[3] Computer Science and Statistics, University of California, Berkeley, CA 94720, U.S.A.
jordan@cs.berkeley.edu

Abstract. The informative vector machine (IVM) is a practical method for Gaussian process regression and classification. The IVM produces a sparse approximation to a Gaussian process by combining assumed density filtering with a heuristic for choosing points based on minimizing posterior entropy. This paper extends IVM in several ways. First, we propose a novel noise model that allows the IVM to be applied to a mixture of labeled and unlabeled data. Second, we use IVM on a block-diagonal covariance matrix, for "learning to learn" from related tasks. Third, we modify the IVM to incorporate prior knowledge from known invariances. All of these extensions are tested on artificial and real data.

1 Introduction

Kernel-based methods have become increasingly popular in the machine learning field. Their close relationships to convex optimization and numerical linear algebra—and their corresponding amenability to theoretical analysis—have helped to fuel their fast growth relative to older non-convex regression and classification methods such as neural networks. Kernel-based methods reduce the data to a "kernel matrix," the entries of which are evaluations of a "kernel function" or "covariance function." Computationally efficient algorithms are available to optimize various statistical functionals of interest for given values of this matrix.

As with most statistical procedures, one can take either a Bayesian or a frequentist perspective on kernel-based methods. The Bayesian point of view involves using a Gaussian process framework to place prior distributions on families of regression or discriminant functions [28]. A Gaussian process is characterized by a mean function and a covariance function, and it is this latter function that yields the "kernel matrix" (when evaluated at the observed data). The frequentist point of view focuses on the optimization of loss functions defined on an inner product space; the choice of inner product defines the kernel function and thus the kernel matrix [19].

J. Winkler, N. Lawrence, M. Niranjan (Eds.): Machine Learning Workshop, LNAI 3635, pp. 56–87, 2005.

By focusing on point estimates, the frequentist approach can yield a stripped-down computational problem that has the appeal of tractability in the context of large-scale problems. For example, the support vector machine reduces to a relatively simple quadratic program [6]. This appeal, however, is somewhat diminished in practice by two factors: (1) many kernel-based procedures involve hyperparameters, and setting hyperparameters generally involves a computationally-intensive outer loops involving cross-validation; and (2) fuller frequentist inference requires calculating error bars for point estimates. This too often requires computationally-intensive extensions of the basic parameter-fitting algorithm (e.g., the bootstrap).

The Bayesian approach to kernel-based methods, on the other hand, focuses on the computation or approximation of posterior probability distributions under the Gaussian process prior. This is generally a more ambitious goal than computing a point estimate, but it also goes a significant distance towards solving the other practical problems associated with using kernel-based methods. Indeed, (1) standard procedures involving the marginal likelihood can be used to set hyperparameters; and (2) the posterior distribution naturally provides an assessment of uncertainty.

Another virtue of the Bayesian approach is the relative ease with which one can consider extensions of a basic model, and it is this virtue that is our focus in the current paper. In particular, we consider two elaborations of the basic classification setup: *semi-supervised learning* and *multi-task learning*. While the frequentist paradigm both permits and requires substantial creativity in deriving methods to attack problems such as these, the Bayesian approach provides standard tools with which to proceed. In particular, in this paper we show how standard Bayesian modeling tools—latent variable modeling and hierarchical modeling—can be used to address nonparametric semi-supervised and multi-task learning within the Gaussian process framework.

Kernel matrices are $N \times N$ matrices (where N is the number of data points), and in both the Bayesian approach and the frequentist approach to kernel-based methods it is important to develop procedures that take advantage of putative low-rank structure in these matrices. One generally useful idea involves *sparsity*—one seeks functions that have expansions in which most of the coefficients are zero. Sparsity can be imposed in the loss function and/or imposed as part of a computational approximation. For example, in the support vector machine the use of a hinge loss implies that only a subset of the training data have non-zero coefficients—these are the *support vectors*. Support vectors embody computational efficiencies and also permit the design of useful extensions that are themselves computationally efficient. For example, *virtual support vectors* allow invariances to be incorporated into support vector machine training [18]. In the current paper, we show how a similar notion can be developed within the Bayesian paradigm—a notion that we refer to as *virtual informative vectors*.

In summary, we view sparse Gaussian process methods as providing a flexible, computationally-efficient approach to nonparametric regression and classification; one that is as viable in practice as its frequentist cousins, and one

that is particularly easy to extend. In this paper we focus on a specific sparse Gaussian process methodology — the *informative vector machine* (IVM). We show how the basic IVM can be extended to accommodate our proposals for semi-supervised learning and multi-task learning. These extensions are not restricted to the IVM methodology: our approach to multi-task learning falls within the broader class of hierarchical Bayesian methods and our noise model for semi-supervised learning can be used in combination with a wide range of models. However the approach we take to incorporating invariances exploits a particular characteristic of the IVM. It relies on the fact that the IVM can be viewed as a *compression scheme*. By this we mean that the decision boundary can be inferred by considering only the 'active set'. This is a characteristic that the IVM shares with the support vector machine. Previously proposed sparse Gaussian process methods seek sparse representations for mean and covariance functions but rely on the entire data set to formulate them.

The paper is organized as follows. After an overview of the general Gaussian process methodology in Section 2, we review the the informative vector machine in Section 3. In Section 4, we present a latent variable approach to semi-supervised learning within the IVM framework. Section 5 develops the IVM-based approach to multi-task learning. Finally, in Section 6 we show how the notion of sparsity provided by the IVM may be exploited in the incorporation of prior invariances in the model.

2 Gaussian Processes

Consider a simple latent variable model in which the output observations, $\mathbf{y} = [y_1 \dots y_N]^{\mathrm{T}}$, are assumed independent from input data, $\mathbf{X} = [\mathbf{x}_1 \dots \mathbf{x}_N]^{\mathrm{T}}$, given a set of latent variables, $\mathbf{f} = [f_1 \dots f_N]^{\mathrm{T}}$. The prior distribution for these latent variables is given by a Gaussian process [1],

$$p\left(\mathbf{f}|\mathbf{X}, \boldsymbol{\theta}\right) = N\left(\mathbf{f}|\mathbf{0}, \mathbf{K}\right),$$

with a covariance function, or 'kernel', $\mathbf{K}$, that is parameterised by the vector $\boldsymbol{\theta}$ and evaluated at the points given in $\mathbf{X}$. This relationship is shown graphically in Figure 1.

The joint likelihood can be written as

$$p\left(\mathbf{y}, \mathbf{f}|\mathbf{X}, \boldsymbol{\theta}\right) = p\left(\mathbf{f}|\mathbf{X}, \boldsymbol{\theta}\right) \prod_{n=1}^{N} p\left(y_n|f_n\right) \tag{1}$$

where $p\left(y_n|f_n\right)$ gives the relationship between the latent variable and our observations and is sometimes referred to as the *noise model*. For the regression case the noise model can also take the form of a Gaussian.

[1] We use $N\left(\mathbf{x}|\boldsymbol{\mu}, \Sigma\right)$ to denote a Gaussian distribution over $\mathbf{x}$ with a mean vector $\boldsymbol{\mu}$ and covariance matrix Σ. When dealing with one dimensional Gaussians the vectors and matrices are replaced by scalars.

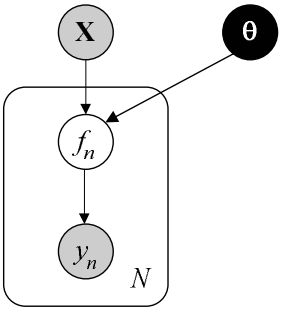


Fig. 1. The Gaussian Process model drawn graphically. We have made use of 'plate' notation to indicate the independence relationship between $\mathbf{f}$ and $\mathbf{y}$.

$$p\left(y_n|f_n\right) = N\left(y_n|f_n, \sigma^2\right). \tag{2}$$

Then the marginalised likelihood can be obtained by integrating over $\mathbf{f}$ in (1). This marginalisation is straightforward—it can be seen to be a special case of the more general result,

$$\begin{aligned} p\left(\mathbf{y}\right) &= \int N\left(\mathbf{f}|\mathbf{0}, \mathbf{K}\right) \prod_{n=1}^{N} N\left(y_n|f_n, \beta_n^{-1}\right) d\mathbf{f} \\ &= N\left(\mathbf{y}|\mathbf{0}, \mathbf{K} + \mathbf{B}^{-1}\right), \end{aligned} \tag{3}$$

where $\mathbf{B}$ is a diagonal matrix whose nth diagonal element is given by β_n. To recover the special case associated with the spherical Gaussian noise model from (2) we simply substitute each β_n with $\frac{1}{\sigma^2}$ and each m_n with y_n.

2.1 Optimising Kernel Parameters

A key advantage of the Gaussian process framework is that once the marginal likelihood is computed then it can be maximised with respect to parameters of the kernel, $\boldsymbol{\theta}$, yielding a so-called *empirical Bayes* method for fitting the model. We have seen that for Gaussian noise models the computation of this marginal likelihood is straightforward. Unfortunately for non-Gaussian noise models the required marginalisation is intractable and we must turn to approximations. One such approximation is known as *assumed density filtering* (ADF). As we shall see, the ADF approximation proceeds by incorporating the data a single point at a time and using Gaussian approximations to the non-Gaussian posterior distributions that arise to maintain the tractability of the model.

2.2 The ADF Approximation

Assumed density filtering has its origins in on-line learning: it proceeds by absorbing one data point at a time, computing the modified posterior and replacing it with an approximation that maintains the tractability of the algorithm. A

thorough treatment of this approach is given by [16,7]. Here we review the main points of relevance for the IVM algorithm.

Given a noise model, $p(y_n|f_n)$, we note that the joint distribution over the data and the latent variables factorises as follows

$$p(\mathbf{y}, \mathbf{f}) = N(\mathbf{f}|\mathbf{0}, \mathbf{K}) \prod_{n=1}^{N} p(y_n|f_n)$$

which may be written

$$p(\mathbf{y}, \mathbf{f}) = \prod_{n=0}^{N} t_n(\mathbf{f})$$

where we have defined $t_n(\mathbf{f}) = p(y_n|f_n)$ and in a slight abuse of notation we have taken $t_0(\mathbf{f}) = N(\mathbf{f}|\mathbf{0}, \mathbf{K})$. ADF takes advantage of this factorised form to build up an approximation, $q(\mathbf{f})$, to the true process posterior, $p(\mathbf{f}|\mathbf{y})$. The factorisation of the joint posterior is exploited by building up this approximation in a sequential way so that after i points are included we have an approximation $q_i(\mathbf{f})$. The starting point is to match the approximation to the prior process, *i.e.* $q_0(\mathbf{f}) = t_0(\mathbf{f}) = N(\mathbf{f}|\mathbf{0}, \mathbf{K})$. The approximation is then constrained to always have this functional form. As the data point inclusion process is sequential two index sets can be maintained. The first, I, is referred to as the *active set* and represents those data points that have been included in our approximation. The second, J, is referred to as the *inactive set* and represents those data points that have not yet been incorporated in our approximation. Initially I is empty and $J = \{1 \ldots N\}$. The approximation to the true posterior is built up by selecting a point, n_1, from J. This point is included in the active set leading to an updated posterior distribution of the form,

$$\hat{p}_1(\mathbf{f}) \propto q_0(\mathbf{f})\, t_{n_1}(\mathbf{f}),$$

our new approximation, $q_1(\mathbf{f})$, is then found by minimising the Kullback Leibler (KL) divergence between the two distributions.

$$\mathrm{KL}(\hat{p}_1||q_1) = -\int \hat{p}_1(\mathbf{f}) \log \frac{q_1(\mathbf{f})}{\hat{p}_1(\mathbf{f})} d\mathbf{f}.$$

More generally, for the inclusion of the ith data point, we can write

$$\hat{p}_i(\mathbf{f}) = \frac{q_{i-1}(\mathbf{f})\, t_{n_i}(\mathbf{f})}{Z_i} \tag{4}$$

where the normalization constant is

$$Z_i = \int t_{n_i}(\mathbf{f})\, q_{i-1}(\mathbf{f})\, d\mathbf{f}. \tag{5}$$

ADF minimises $\mathrm{KL}(\hat{p}_i||q_i)$ to obtain the new approximation. This KL divergence can be minimised by 'moment matching' equations of the form

$$q_i(\mathbf{f}) = N(\mathbf{f}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i).$$

where

$$\boldsymbol{\mu}_i = \langle \mathbf{f} \rangle_{\hat{p}_i(\mathbf{f})} \tag{6}$$

$$\boldsymbol{\Sigma}_i = \left\langle \mathbf{f}\mathbf{f}^{\mathrm{T}} \right\rangle_{\hat{p}_i(\mathbf{f})} - \langle \mathbf{f} \rangle_{\hat{p}_i(\mathbf{f})} \langle \mathbf{f} \rangle_{\hat{p}_i(\mathbf{f})}{}^{\mathrm{T}} \tag{7}$$

and where $\langle \cdot \rangle_{p(\cdot)}$ denotes an expectation under the distribution $p(\cdot)$. It turns out that our ability to compute (6) and (7) depends on the tractability of the normalisation constant in (4). An update equation for the posterior mean (see Appendix A) is given by

$$\boldsymbol{\mu}_i = \boldsymbol{\mu}_{i-1} + \Sigma_{i-1}\mathbf{g}_i,$$

where the dependence on Z_i is through

$$\mathbf{g}_i = \frac{\partial \log Z_i}{\partial \boldsymbol{\mu}_i}.$$

The corresponding update for Σ_i is given by

$$\Sigma_i = \Sigma_{i-1} - \Sigma_{i-1}\left(\mathbf{g}_i\mathbf{g}_i^{\mathrm{T}} - 2\Gamma_i\right)\Sigma_{i-1}$$

where

$$\Gamma_i = \frac{\partial \log Z_i}{\partial \Sigma_{i-1}}.$$

The ADF approximation can therefore be easily be applied for any noise model for which the expectation of the noise model in (5) is tractable. In the next sections we shall review two of the most commonly used noise models within the context of the ADF algorithm. The Gaussian noise model is commonly used in regression applications and the probit noise model is applicable for classification problems. Later, in Section 4, we describe a noise model that is suitable for semi-supervised learning.

2.3 Gaussian Noise Model

One of the most widely used approaches for interpolation between data points is model fitting through least squares. From the probabilistic point of view this is equivalent to a Gaussian noise model. If each data point, y_n, has a different associated variance, β_n^{-1}, then the noise model is given by

$$p(y_n|f_n) = \sqrt{\frac{\beta_n}{2\pi}} \exp\left(-\frac{\beta_n (y_n - f_n)^2}{2}\right).$$

As we described in the previous section, the updates for the mean and covariance functions depend on Z_i, the expectation of the noise model under the current estimate of the posterior process, $q_{i-1}(\mathbf{f})$. Since the noise model only depends on f_n, the nth element of $\mathbf{f}$, we may rewrite this expectation in the form

$$Z_i = \int p(y_n|f_n)\, q(f_n)\, df_n, \tag{8}$$

where we have exploited the fact that the marginal, $q(f_n)$, of the full multivariate Gaussian, $q(\mathbf{f})$, is given by

$$q(f_n) = N(f_n|\mu_{i-1,n}, \varsigma_{i-1,n})$$

where $\mu_{i-1,n}$ is the nth element of $\boldsymbol{\mu}_{i-1}$ and $\varsigma_{i-1,n}$ is the nth element from the diagonal of Σ_{i-1}. As a result the expectation in (8) is easily computed as

$$Z_i = N\left(y_n|\mu_{i-1,n}, \left(\varsigma_{i-1,n} + \beta_n^{-1}\right)\right)$$

The log partition function,

$$\log Z_i = -\frac{1}{2}\log 2\pi - \frac{1}{2}\log\left(\varsigma_{i-1,n} + \beta_n^{-1}\right) - \frac{\left(y_n - \mu_{i-1,n}\right)^2}{2\left(\beta_n^{-1} + \varsigma_{i-1,n}\right)},$$

can then be differentiated with respect to $\boldsymbol{\mu}_{i-1}$ to give

$$g_{in} = \frac{y_n - \mu_{i-1,n}}{\beta_n^{-1} + \varsigma_{i-1,n}}, \tag{9}$$

where g_{in} is the nth element of $\mathbf{g}_i$ and all other elements are zero. Similarly we can differentiate the log partition with respect to Σ_{i-1} to find the nth diagonal element of Γ_i

$$\gamma_{in} = -\frac{1}{2\left(\varsigma_{i-1} + \beta_n^{-1}\right)} + \frac{1}{2}g_{in}^2$$

all other elements of Γ_i are zero.

For the update of the covariance we need to consider the matrix $\mathbf{g}_i\mathbf{g}_i^{\mathrm{T}} - 2\Gamma_i$. However, for the Gaussian noise model this matrix can be written in diagonal form, $\mathrm{diag}(\boldsymbol{\nu}_i)$, where the nth element of $\boldsymbol{\nu}_i$ is

$$\begin{aligned}\nu_{in} &= -2\gamma_{in} + g_{in}^2 \\ &= \frac{1}{\left(\varsigma_{i-1} + \beta_n^{-1}\right)}\end{aligned} \tag{10}$$

and all other elements are zero. This leads to a set of simplified update equations of the form

$$\boldsymbol{\mu}_i = \boldsymbol{\mu}_{i-1} + g_{in}\mathbf{s}_{i-1,n}, \tag{11}$$

$$\Sigma_i = \Sigma_{i-1} - \nu_{in}\mathbf{s}_{i-1,n}\mathbf{s}_{i-1,n}^{\mathrm{T}}, \tag{12}$$

where $\mathbf{s}_{i-1,n}$ is the nth column from Σ_{i-1}.

Implicit in these update equations is the minimisation of a KL divergence between the true posterior and the approximating Gaussian process posterior, $q(\mathbf{f})$. Of course, for the Gaussian noise model, the true posterior process is Gaussian so the approximation is exact. In other words, for the special case of

the Gaussian noise, these updates are simply a scheme for on-line learning of Gaussian processes without any approximations involved.

In the next section, we consider the probit classification noise model. This noise model is an example from the more general case where the true posterior distribution is non-Gaussian and every update of the mean and posterior implicitly involves an approximation to this true posterior through moment matching.

2.4 Probit Noise Model

We consider binary classification where $y_n \in \{-1, 1\}$. A convenient representation for the probability of class membership given the latent variable f_n is given by the *probit* noise model for classification

$$p\left(y_n | f_n\right) = \phi\left(y_n \lambda \left(f_n + b\right)\right)$$

where $\phi\left(\cdot\right)$ is the cumulative Gaussian given by

$$\phi\left(z\right) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} \exp\left(-\frac{t^2}{2}\right) dt,$$

the slope of which is controlled by λ. The use of the probit, rather than the more commonly encounter logit, is convenient as it leads to a tractable integral for the partition function:

$$Z_i = \frac{1}{\left(2\pi\right)^{\frac{N}{2}} \varsigma_{i-1,n}^{\frac{1}{2}}} \int \phi\left(y_n \lambda \left(f_n + b\right)\right) \exp\left(-\frac{\left(f_n - \boldsymbol{\mu}_{i-1}\right)^2}{2\varsigma_{i-1,n}}\right) df_n$$

which may be integrated to obtain

$$Z_i = \phi\left(u_{i-1,n}\right).$$

where

$$u_{i-1,n} = c_{i-1,n}\left(\mu_{i-1,n} + b\right)$$

$$c_{i-1,n} = \frac{y_n}{\sqrt{\lambda^{-2} + \varsigma_{i-1,n}}}. \tag{13}$$

Once again, the partition function is only dependent on one element of $\boldsymbol{\mu}_{i-1}$ and one element of Σ_{i-1}. Performing the necessary derivatives to obtain g_{in} and ν_{in} we have[2]

$$g_{in} = c_{i-1,n} \mathcal{N}\left(u_{i-1,n}\right) \left[\phi\left(u_{i-1,n}\right)\right]^{-1}, \tag{14}$$

[2] In implementation, care must be taken in computing g_{in}: when $u_{i-1,n}$ has large magnitude both $\phi\left(u_{i-1,n}\right)$and $N\left(u_{i-1,n}\right)$ become very small and numerical precision issues arise. This can be avoided by performing the computations should be done in the log domain.

where

$$\mathcal{N}\left(u_{i-1,n}\right) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{u_{i-1,n}^2}{2}\right)$$

and

$$\gamma_{in} = -\frac{1}{2} g_{in} u_{i-1} c_{i-1,n}$$

which implies

$$\nu_{in} = g_{in}\left(g_{in} + u_{i-1} c_{i-1,n}\right). \tag{15}$$

Updates for $\boldsymbol{\mu}_{i-1} \rightarrow \boldsymbol{\mu}_i$ and $\Sigma_{i-1} \rightarrow \Sigma_i$, the parameters of $q\left(\mathbf{f}\right)$, are then identical to those given in (11) and (12). Note from (13) that we can consider the slope, λ, of the noise model to be infinite and develop an equivalent representation by adding a matrix $\lambda^{-2}\mathbf{I}$ to the kernel matrix thereby causing $\varsigma_{i,n}$ to increase by λ^{-2}. In our experiments, this approach is preferred because the noise model slope can then be optimised in tandem with the kernel parameters.

2.5 Kernel Parameter Updates

So far we have discussed how the posterior's mean and covariance functions can be updated in an on-line way given a fixed kernel. We suggested in the introduction that one of the main reasons we may wish to keep track of a representation of the posterior covariance is so that we may learn the kernel parameters via an empirical Bayes approach.

With reference to the graphical representation of our model in Figure 1 our objective is to marginalise the latent variable $\mathbf{f}$ and optimise the parameters of the kernel by maximising the resulting likelihood. This marginalisation can only be achieved exactly if the noise model is Gaussian. When dealing with non-Gaussian noise models we must again consider approximations.

One perspective on the ADF approximation is that we are taking non-Gaussian noise models and approximating them with Gaussian noise models. While in practise we are approximating a non-Gaussian posterior distribution with a Gaussian distribution, $q\left(\mathbf{f}\right)$, this approximation can also be viewed as replacing the true noise model with a 'apparent Gaussian noise model' that also induces the posterior distribution $q\left(\mathbf{f}\right)$. Note that this point of view is only reasonable if the noise model is log concave, otherwise the implied Gaussian distribution can have a negative variance.

If we accept this point of view then we consider

$$p\left(\mathbf{y}\right) \approx N\left(\mathbf{m}|\mathbf{0}, \mathbf{K} + \mathbf{B}^{-1}\right), \tag{16}$$

where $\mathbf{m}$ and $\boldsymbol{\beta}$ are the means and precisions associated with the implied Gaussians, to be a valid approximation to the marginal likelihood. The individual site mean, m_n, and precision, β_n, associated with each 'apparent Gaussian noise model' can be computed given that noise model's values for g_{in} and ν_{in}. By rearranging (9) and (10) and replacing y_n with m_n we have

$$g_i = \frac{m_n - \mu_{i-1,n}}{\beta_n^{-1} + \varsigma_{i-1,n}}$$
$$g_i = \frac{m_n - \mu_{i-1,n}}{\nu_{in}}$$

$$m_n = \frac{g_{in}}{\nu_{in}} + \mu_{i-1,n} \tag{17}$$

and

$$\nu_{in} = \frac{1}{\varsigma_{i-1,n} + \beta_n^{-1}}$$
$$\beta_n^{-1} = \frac{1}{\nu_{in}} - \varsigma_{i-1,n}$$

$$\beta_n = \frac{\nu_{in}}{1 - \nu_{in}\varsigma_{i-1,n}}. \tag{18}$$

which give the site parameters for a given noise model. Note that if $\nu_{in}\varsigma_{i-1,n} > 1$ the site precision, β_n, becomes negative. This can only occur if the noise model is not log concave—we shall encounter such a noise model in Section 4.

3 The Informative Vector Machine

One advantage of the Gaussian process perspective referred to in the introduction is automatic selection of the kernel parameters through likelihood optimisation. In practise though gradients with respect to the parameters must be computed. Computation of these gradients involves an inverse of the kernel matrix, $\mathbf{K}$, at each step. This matrix inverse gives each gradient step $O\left(N^3\right)$ complexity. Even if the kernel parameters are given including N data points through ADF still leads to an $O\left(N^3\right)$ complexity. The IVM algorithm seeks to resolve these problems by seeking a sparse representation for the data set. The inspiration for this approach is the support vector machine, a kernel algorithm for which the solution is typically naturally sparse. This natural sparsity can arise because the SVM algorithm ignores the process variances, however these process variances must be accounted for if optimisation of kernel parameters via empirical Bayes is undertaken. Csató and Opper suggest obtaining a sparse representation through minimisation of KL divergences between the approximate posterior and a sparse representation [8,7]. The IVM takes a simpler approach of using a selection heuristic to incorporate only the most informative points and stopping after d inclusions [14,21]. By making only d inclusions the resulting model is forced to be sparse. As a result we can reduce the computational requirements of the algorithm from $O\left(N^3\right)$ to $O\left(d^2N\right)$ and the storage requirements form $O\left(N^2\right)$ to $O\left(dN\right)$. In the IVM this 'sparsification' of the original Gaussian process is imposed by only selecting a subset of the data. The ADF algorithm allows us to select this subset in an on-line way, given a data point selection criterion. The IVM uses an information theoretic selection heuristic to select which data point

to include. Specifically, the point that gives the largest reduction in posterior process entropy is included. In this way the entropy of the posterior process is greedily minimised.

3.1 Data Point Selection

The simple greedy selection criterion on which the IVM is based is inspired by information theory. The change in entropy of the posterior process after including a point can be seen as a measure of reduction in the level of uncertainty. This entropy reduction can only be evaluated because we are propagating the posterior covariance function. Loosely speaking, the entropy of the posterior process is analogous to the version space in the SVM. Greedily minimising the entropy can be seen as slicing the largest possible section away from version space and tracking the posterior covariance can be seen as maintaining a representation of the size and shape of version space while the model is being constructed.

The entropy change associated with including the nth point at the ith inclusion is given by

$$\begin{aligned}\Delta H_{in} &= -\frac{1}{2}\log|\Sigma_{i,n}| + \frac{1}{2}\log|\Sigma_{i-1}| \\ &= -\frac{1}{2}\log|\mathbf{I} - \Sigma_{i-1}\mathrm{diag}(\boldsymbol{\nu}_i)| \\ &= -\frac{1}{2}\log\left(1 - \nu_{in}\varsigma_{i-1,n}\right). \end{aligned} \tag{19}$$

This entropy change can be evaluated for every point in the inactive set, J, and the point associated with the largest entropy decrease can then be included. This process is repeated until d inclusions have been made. Other criteria (such as information gain) are also straightforward to compute. Such greedy selection criteria are inspired by information theory and have also been applied in the context of active learning [15,22], however in active learning the label of the data point, y_n, is assumed to be unknown before selection: here the label is known and can strongly influence whether a particular point is selected.

We note from (19) that in order to score each point we need to keep track of the diagonal terms from Σ_{i-1} and the values ν_{in}. If these terms can be stored and updated efficiently then the computational complexity of the algorithm can be controlled.

Maintaining the whole of Σ_{i-1} in memory would require $O\left(N^2\right)$ storage, which is undesirable, so our first task is to seek an efficient representation of the posterior covariance.

From (12) it is clear the posterior covariance Σ_i has a particular structure that arises from the addition of successive outer products to the original prior covariance matrix $\Sigma_0 = \mathbf{K}$. This can be re-written as

$$\Sigma_i = \mathbf{K} - \mathbf{M}_i^{\mathrm{T}}\mathbf{M}_i \tag{20}$$

where the kth row of $\mathbf{M}_i \in \Re^{i\times N}$ is given by $\sqrt{\nu_{k,n_k}}\mathbf{s}_{k-1,n_k}$ and n_k represents kth included data point. Recalling that $\mathbf{s}_{i-1,n_i}$ is the n_ith column of Σ_{i-1} we

note that, if we are not storing Σ_{i-1}, we are not able to represent (for instance) $\mathbf{s}_{i-1,n_i}$ directly. However, this column can be recomputed from $\mathbf{M}_{i-1}$ and $\mathbf{K}$ by

$$\mathbf{s}_{i-1,n_i} = \mathbf{k}_{n_i} - \mathbf{a}_{i-1,n_i}^{\mathrm{T}} \mathbf{M}_{i-1} \tag{21}$$

where $\mathbf{k}_{n_i} = \mathbf{s}_{0,n_i}$ is the n_ith column of $\mathbf{K} = \Sigma_0$ and $\mathbf{a}_{i-1,n_i}$ is the n_ith column of $\mathbf{M}_{i-1}$. This recomputation requires $O\left((i-1)\,N\right)$ operations.

3.2 The Matrix $\mathbf{M}_i$

Let us consider more closely what the matrix $\mathbf{M}_i$ represents. It is straightforward to show that a Gaussian process which has included an active set I with i elements has a posterior covariance of the form

$$\Sigma_i = \mathbf{K} - \mathbf{K}_{:,I} \left(\mathbf{B}_I^{-1} + \mathbf{K}_I\right)^{-1} \mathbf{K}_{I,:} \tag{22}$$

where $\mathbf{K}_{:,I}$ is a matrix containing the columns of $\mathbf{K}$ that are in the active set, $\mathbf{K}_I$ is an $i \times i$ symmetric matrix containing the rows and columns from $\mathbf{K}$ that are in I and $\mathbf{B}_I$ is a diagonal matrix of the 'site precisions' for the active set. Equating with (20) we note that $\mathbf{M}_i^{\mathrm{T}} \mathbf{M}_i = \mathbf{K}_{:,I} \left(\mathbf{B}_I^{-1} + \mathbf{K}_I\right)^{-1} \mathbf{K}_{I,:}$ or

$$\mathbf{L}_i^{\mathrm{T}} \mathbf{M}_i = \mathbf{K}_{I,:}$$

where $\mathbf{L}_i \mathbf{L}_i^{\mathrm{T}} = \left(\mathbf{B}_I^{-1} + \mathbf{K}_I\right)^{-1}$.

From the way that the updates in (12) accumulate it can be shown that a valid factorisation of the matrix $\mathbf{L}_i \mathbf{L}_i^{\mathrm{T}}$ is given by the lower triangular matrix

$$\mathbf{L}_i = \begin{bmatrix} \mathbf{L}_{i-1} & \mathbf{0} \\ \mathbf{a}_{i-1,n_i}^{\mathrm{T}} & \nu_{i,n_i}^{-\frac{1}{2}} \end{bmatrix}, \tag{23}$$

where $\mathbf{L}_1 = \nu_{i,n_1}^{-\frac{1}{2}}$. This lower triangular matrix is recognised as a Cholesky factor of $\left(\mathbf{B}_I^{-1} + \mathbf{K}_I\right)^{-1}$. We may gain some reassurance from the implicit presence of a Cholesky decomposition within the algorithm as they are typically considered to be a numerically stable. Note however that in [21,14] a slightly different representation is suggested. Instead of keeping track of the Cholesky decomposition of $\left(\mathbf{B}_I^{-1} + \mathbf{K}_I\right)$, the decomposition of $\left(\mathbf{I} + \mathbf{B}_I^{-\frac{1}{2}} \mathbf{K}_I \mathbf{B}_I^{-\frac{1}{2}}\right)$ is used to give greater numerical stability when $\mathbf{K}_I$ is not full rank and when some values of $\mathbf{B}_I^{-1}$ are close to zero. Here we prefer our representation as it arises naturally and keeps the update equations clear.

Note that the matrix $\mathbf{L_i}$ is only a valid lower Cholesky factor if ν_{i,n_i} is non-negative. This can be shown to hold true if the noise model is log-concave. Complications arise if the noise model is not log-concave; a simple strategy for avoiding this problem is suggested in Section 4.

We have already stated that storing the entire posterior covariance matrix is impractical because of memory requirements. However, maintaining an estimate

Algorithm 1. The IVM point selection algorithm.

Require: Require d = number of active points. Start with $\mathbf{m} = \mathbf{0}$ and $\boldsymbol{\beta} = \mathbf{0}$ for classification. For regression substitute appropriate target values. Take $\boldsymbol{\varsigma}_0 = \text{diag}(\mathbf{K})$, $\boldsymbol{\mu} = \mathbf{0}$, $J = \{1, \ldots, N\}$, $I = \{\cdot\}$, $\mathbf{S}_0$ is an empty matrix.
  for $k = 1$ to d **do**
    for all $n \in J$ **do**
      compute g_{kn} according to (14) (not required for Gaussian).
      compute ν_{kn} according to (15) or (10).
      compute ΔH_{kn} according to (19).
    end for
    $n_k = \text{argmax}_{n \in J} \Delta H_{kn}$.
    Update m_{n_k} and β_{n_k} using (17) and (18).
    Compute $\boldsymbol{\varsigma}_k$ and $\boldsymbol{\mu}_k$ using (21), (24) and (25).
    Append $\sqrt{\nu_{kn_k}}\mathbf{s}_{k-1,n_k}^{\mathrm{T}}$ to $\mathbf{M}_{k-1}$ using (21) to form $\mathbf{M}_k$.
    Update $\mathbf{L}_k$ from $\mathbf{L}_{k-1}$ using (23).
    Add n_k to I and remove n_k from J.
  end for

of the process variance associated with each data point is of interest. These variances are the diagonal elements from the posterior covariance matrix, $\boldsymbol{\varsigma}_i = \text{diag}(\Sigma_i)$. A vector storing these variances can be updated easily using (11), (12) and (21) to give,

$$\boldsymbol{\varsigma}_i = \boldsymbol{\varsigma}_{i-1} - \nu_{i,n_i} \text{diag}\left(\mathbf{s}_{i-1,n_i}\mathbf{s}_{i-1,n_i}^{\mathrm{T}}\right) \tag{24}$$

which is computed in $O(N)$ operations. The posterior mean output vector should also be stored and updated using

$$\boldsymbol{\mu}_i = \boldsymbol{\mu}_{i-1} + g_{in_i}\mathbf{s}_{i-1,n_j} \tag{25}$$

which also requires $O(N)$ operations.

An example of how these updates may be combined efficiently in practise is given in Algorithm 1.

3.3 IVM Kernel Parameter Updates

The ADF-based IVM algorithm described in Algorithm 1 leads to sparse vectors $\mathbf{m}$ and $\boldsymbol{\beta}$ each with d non-zero elements. In Section 2.5, we described how kernel parameters could be updated given non-sparse vectors of these site parameters. To maximise kernel parameters for the IVM we need to express the likelihood in terms of sparse versions of these vectors. In [21] this is achieved by expressing the likelihood in terms of both the active and inactive sets. Here we take a much simpler approach of maximising the likelihood of the active points only,

$$p(\mathbf{y}_I|\boldsymbol{\theta}) \approx N\left(\mathbf{m}_I|\mathbf{0}, \mathbf{K}_I + \mathbf{B}_I^{-1}\right), \tag{26}$$

where $\mathbf{y}_I$ is a vector containing only those elements of $\mathbf{y}$ that are in the active set. The dependence of the likelihood on the parameters, $\boldsymbol{\theta}$, is through the active portion of the kernel matrix $\mathbf{K}_I$.

Given the active set, I, and the site parameters, $\mathbf{m}$ and $\boldsymbol{\beta}$, we can optimise our approximation with respect to the kernel parameters by using a non-linear optimiser such as scaled conjugate gradients.

Note that in practise, the quality of the active set depends on the kernel parameter selection as does the optimal site parameters. We can certainly imagine more complex schema for optimising the kernel parameters that take account of these dependencies in a better way, some examples of which are given in [21], but for our experiments we simply iterated between active set selection and kernel parameter optimisation.

3.4 Noise Parameter Updates

As well as updating the parameters of the kernel, it may be helpful to update the parameters of the noise function. However, the likelihood approximation (26) is only indirectly dependent on those parameter values so cannot be used as an objective function. One way forward would be to optimise a variational lower bound,

$$\sum_{n=1}^{N} \int q_d(f_n) \log p(y_n|f_n, \boldsymbol{\theta})\, p(f_n)\, df_n - \sum_{n=1}^{N} \int q_d(f_n) \log q(f_n)\, df_n,$$

on the likelihood, where $\boldsymbol{\theta}$ are the parameters of the noise model that we wish to optimise. The relevant term in this bound is

$$\sum_{n=1}^{N} \int q_d(f_n) \log p(y_n|f_n, \boldsymbol{\theta}). \tag{27}$$

This lower bound can be upper bounded by

$$\begin{aligned} \mathcal{L}(\theta) &= \sum_{n=1}^{N} \log \int q_d(f_n)\, p(y_n|f_n, \boldsymbol{\theta}) \\ &= \sum_{n=1}^{N} \log Z_n. \end{aligned} \tag{28}$$

For many models it is straightforward to compute (27), however for all noise models it is possible to compute (28). We found that, for an ordered categorical noise model (where there are an atypically large number of noise model parameters), optimisation of (28) was sufficient.

3.5 Point Removal and Expectation Propagation

The sequential nature of the ADF algorithm leads to a weakness in the approximation for which a fairly straightforward fix has been suggested [16]. When the Gaussian approximation to the likelihood after i inclusions is computed, its

Algorithm 2. The IVM algorithm for parameter optimisation.

Require: Require d active points. T iterations.
  for $i = 1$ to T **do**
    Select points using Algorithm 1.
    Maximise the approximation to the likelihood (26) using a scaled conjugate gradient optimiser [17].
    if noise parameter updates are required. **then**
      Select points using Algorithm 1.
      Maximise the sum of the log partition functions (28) using a scaled conjugate gradient optimiser.
    end if
  end for

associated site parameters, β_{n_i} and m_{n_i} are based on only the $i-1$ points that are already in the active set. However as more points are included and the posterior approximation evolves it is likely that the quality of this approximation becomes worse. The approach suggested by [16] is to revisit the approximation and improve it. This refinement of the ADF algorithm is known as *expectation propagation* (EP). Conceptually one can view these later updates as removing a point from the active set (by resetting the associated site parameters to zero) and then computing a new Gaussian approximation in the light of the current (and usually more accurate) representation of the posterior covariance. The key issue for the algorithm is, therefore, to be able remove a point in an efficient manner.

Even if the expectation propagation algorithm is not being used, it may be desirable to remove points within the IVM algorithm as it is likely to be the case that points that were included in the early stages of the algorithm, when the estimate of the posterior's covariance and mean functions were poor, are less useful than they originally appeared. Inclusion of such points is a natural consequence of greedily selecting points to reduce entropy. Some approaches to the implementation of point removal and expectation propagation with the IVM are further discussed in [21].

3.6 IVM Implementation

In the experimental sections that follow we make use of the IVM algorithm for learning. For all these experiments we used the kernel parameter initialisations and the types of kernels detailed in Appendix B. The algorithm we used for point selection is specified in Algorithm 1 and when optimisation was required we typically made use of 8 iterations of Algorithm 2.

4 Semi-supervised Learning

In a traditional classification problem, data are presented in the form of a set of input vectors $\mathbf{X} = [\mathbf{x}_1 \dots \mathbf{x}_N]^{\mathrm{T}}$ and associated labels $\mathbf{y} = [y_1 \dots y_N]^{\mathrm{T}}$, $y_n \in$

$\{-1, 1\}$. When performing classification the IVM algorithm seeks a process mapping between the inputs and the labels that yields high predictive accuracy. This is known as *supervised learning*, as each input data point, $\mathbf{x}_n$, has an associated label, y_n. In practice, labelling a data set can be a time consuming process; very often it is straightforward to obtain input data points, $\mathbf{x}_n$, but expensive to obtain an associated label y_n. It is natural then to consider whether unlabelled data points can be used to improve predictive performance. Fitting models using such partially labelled data sets is known as 'semi-supervised learning'.

Most probabilistic classification algorithms can be categorised as either discriminative or generative methods. Discriminative methods, such as the IVM, estimate the probability of class membership, $p(y_n|\mathbf{x}_n)$ directly. Generative methods typically model the class-conditional densities, $p(\mathbf{x}_n|y_n)$ and use them in combination with an estimate of the class priors to infer $p(y_n|\mathbf{x}_n)$ through Bayes' theorem. In the former case, if we fail to make any assumptions about the underlying distribution of input data, the unlabelled data does not affect our predictions. Thus, most attempts to make use of unlabelled data within a probabilistic framework focus on incorporating a model of $p(\mathbf{x}_n)$; for example, by treating it as a mixture, $\sum_{y_n} p(\mathbf{x}_n|y_n)\, p(y_n)$, and inferring $p(y_n|\mathbf{x}_n)$ (e.g., [13]), or by building kernels based on $p(\mathbf{x}_n)$ (e.g., [20]). These approaches can be unwieldy, however, in that the complexities of the input distribution are typically of little interest when performing classification, so that much of the effort spent modelling $p(\mathbf{x}_n)$ may be wasted.

An alternative is to make weaker assumptions regarding $p(\mathbf{x}_n)$ that are of particular relevance to classification. In particular, the *cluster assumption* asserts that the data density should be reduced in the vicinity of a decision boundary (e.g., [5]). Such a qualitative assumption is readily implemented within the context of non-probabilistic kernel-based classifiers. Here we discuss how the same assumption can be incorporated within the IVM algorithm [11].

Our approach involves a notion of a 'null category region', a region that acts to exclude unlabelled data points. Such a region is analogous to the traditional notion of a 'margin' and indeed our approach is similar in spirit to the transductive SVM [26], which seeks to maximise the margin by allocating labels to the unlabelled data. A major difference, however, is that through our use of the IVM algorithm our approach maintains and updates the process variance. As we will see, this variance turns out to interact in a significant way with the null category concept.

4.1 Null Category Noise Model

Before considering the null category noise model (NCNM) we first briefly review ordered categorical models that underpin the NCNM. In the specific context of binary classification, we consider an ordered categorical model containing three categories[3].

[3] See also [23] that makes use of a similar noise model in a discussion of Bayesian interpretations of the SVM.

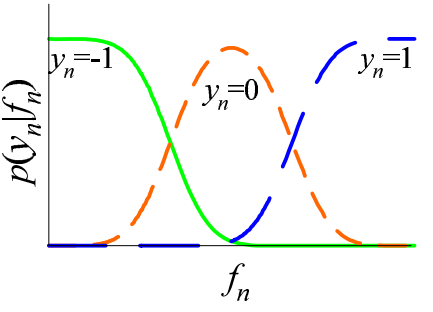


Fig. 2. The ordered categorical noise model. The plot shows $p(y_n|f_n)$ for different values of y_n. Here we have assumed three categories.

$$p(y_n|f_n) = \begin{cases} \phi\left(-\left(f_n + \frac{w}{2}\right)\right) & \text{for } y_n = -1 \\ \phi\left(f_n + \frac{w}{2}\right) - \phi\left(f_n - \frac{w}{2}\right) & \text{for } y_n = 0 \\ \phi\left(f_n - \frac{w}{2}\right) & \text{for } y_n = 1 \end{cases},$$

where $\phi(x) = \int_{-\infty}^{x} N(z|0,1)\, dz$ is the cumulative Gaussian distribution function and w is a parameter giving the width of category $y_n = 0$ (see Figure 2).

We can also express this model in an equivalent and simpler form by replacing the cumulative Gaussian distribution by a Heaviside step function $H(\cdot)$ and adding independent Gaussian noise to the process model:

$$p(y_n|f_n) = \begin{cases} H\left(-\left(f_n + \frac{1}{2}\right)\right) & \text{for } y_n = -1 \\ H\left(f_n + \frac{1}{2}\right) - H\left(f_n - \frac{1}{2}\right) & \text{for } y_n = 0 \\ H\left(f_n - \frac{1}{2}\right) & \text{for } y_n = 1 \end{cases},$$

where we have standardised the width parameter to 1, by assuming that the overall scale is also handled by the process model.

To use this model in an unlabelled setting, we introduce a further variable, z_n, that is one if a data point is unlabelled and zero otherwise. We first impose

$$p(z_n = 1|y_n = 0) = 0; \tag{29}$$

in other words, a data point can not be from the category $y_n = 0$ and be unlabelled. We assign probabilities of missing labels to the other classes

$$p(z_n = 1|y_n = 1) = \gamma_+ \quad \text{and}$$
$$p(z_n = 1|y_n = -1) = \gamma_-.$$

We see from the graphical representation in Figure 3 that z_n is d-separated from $\mathbf{x}_n$. Thus when y_n is observed, the posterior process is updated by using $p(y_n|f_n)$. On the other hand, when the data point is unlabelled the posterior process must be updated by $p(z_n|f_n)$ which is easily computed as:

$$p(z_n = 1|f_n) = \sum_{y_n} p(y_n|f_n)\, p(z_n = 1|y_n).$$

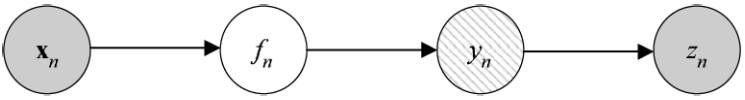


Fig. 3. Graphical representation of the null category model. The fully-shaded nodes are always observed, whereas the lightly-shaded node is observed when $z_n = 0$.

The "effective likelihood function" for a single data point, $L(f_n)$, therefore takes one of three forms:

$$L(f_n) = \begin{cases} H\left(-\left(f_n + \frac{1}{2}\right)\right) & \text{for } y_n = -1,\ z_n = 0 \\ \gamma_- H\left(-\left(f_n + \frac{1}{2}\right)\right) + \gamma_+ H\left(f_n - \frac{1}{2}\right) & \text{for } \quad z_n = 1 \\ H\left(f_n - \frac{1}{2}\right) & \text{for } \quad y_n = 1\ z_n = 0 \end{cases}.$$

The constraint imposed by (29) implies that an unlabelled data point never comes from the class $y_n = 0$. Since $y_n = 0$ lies between the labelled classes this is equivalent to a hard assumption that no data comes from the region around the decision boundary. We can also soften this hard assumption if so desired by injection of noise into the process model. If we also assume that our labelled data only comes from the classes $y_n = 1$ and $y_n = -1$ we never obtain any evidence for data with $y_n = 0$; for this reason we refer to this category as the *null category* and the overall model as a *null category noise model* (NCNM).

4.2 Process Model and Effect of the Null Category

To work within the Gaussian process framework we take

$$p(f_n|\mathbf{x}_n) = N(f_n|\mu(\mathbf{x}_n), \varsigma(\mathbf{x}_n)),$$

where the mean $\mu(\mathbf{x}_n)$ and the variance $\varsigma(\mathbf{x}_n)$ are functions of the input vector. A natural consideration in this setting is the effect of our likelihood function on the distribution over f_n from incorporating a new data point. First we note that if $y_n \in \{-1, 1\}$ the effect of the likelihood is similar to that incurred in binary classification, in that the posterior is a convolution of the step function and a Gaussian distribution. This is comforting as when a data point is labelled the model acts in a similar manner to a standard binary classification model. Consider now the case when the data point is unlabelled. The effect of the data point depends on the mean and variance of $p(f_n|\mathbf{x}_n)$. If this Gaussian has little mass in the null category region, the posterior is similar to the prior. However, if the Gaussian has significant mass in the null category region, the outcome may be loosely described in two ways:

1. If $p(f_n|\mathbf{x}_n)$ 'spans the likelihood', Figure 4 (Left), then the mass of the posterior can be apportioned to either side of the null category region, leading to a bimodal posterior. The variance of the posterior is greater than the variance of the prior, a consequence of the fact that the effective likelihood function is not log-concave (as can be easily verified).

2. If $p(f_n|\mathbf{x}_n)$ is 'rectified by the likelihood', Figure 4 (Right), then the mass of the posterior is pushed in to one side of the null category and the variance of the posterior is smaller than the variance of the prior.

Note that for all situations when a portion of the mass of the prior distribution falls within the null category region it is pushed out to one side or both sides. The intuition behind the two situations is that in case 1, it is not clear what label the data point has, however it is clear that it shouldn't be where it currently is (in the null category). The result is that the process variance increases. In case 2, the data point is being assigned a label and the decision boundary is pushed to one side of the point so that it is classified according to the assigned label.

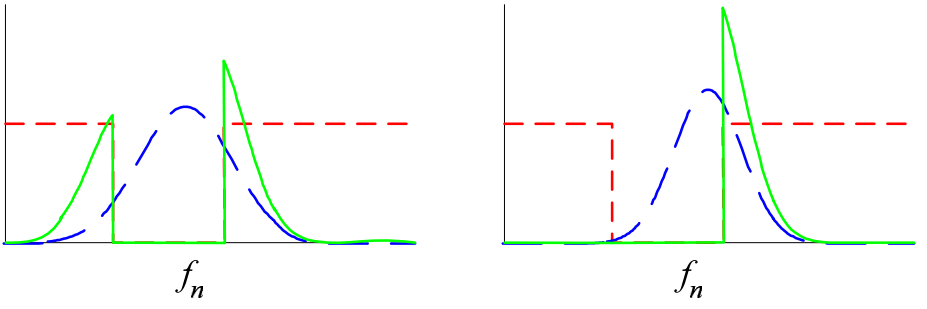


Fig. 4. Two situations of interest. Diagrams show the prior distribution over f_n (long dashes) the effective likelihood function from the noise model when $z_n = 1$ (short dashes) and a schematic of the resulting posterior over f_n (solid line). *Left*: The posterior is bimodal and has a larger variance than the prior. *Right*: The posterior has one dominant mode and a lower variance than the prior. In both cases the process is pushed away from the null category.

4.3 Posterior Inference and Prediction

Recall from Section 4.2 that the noise model is not log-concave. When the variance of the process increases the best Gaussian approximation to our noise model can have negative variance. This has an important implication: the site precision can be computed as a negative value (see Section 2.5). This situation is discussed in [16], where various suggestions are given to cope with the issue. For the IVM we followed the simplest suggestion: we set a negative variance to zero. As we have discussed, one important advantage of the Gaussian process framework is that hyperparameters in the covariance function (i.e., the kernel function), can be fit by optimising the marginal likelihood. In practise, however, if the process variance is maximised in an unconstrained manner the effective width of the null category can be driven to zero, yielding a model that is equivalent to a standard binary classification noise model[4]. To prevent this from happening we regularise with an L_1 penalty on the process variances (this is equivalent to placing an exponential prior on those parameters).

[4] Recall, as discussed in Section 2, that we fix the width of the null category to unity: changes in the scale of the process model are equivalent to changing this width.

4.4 Semi-supervised Learning: Toy Problem

In all our experiments we used an RBF kernel with a white noise term (see Appendix B).

We made use of Algorithm 2 without the optional noise parameter update. To ensure that the width of the null category region wasn't collapsing with repeated iteration of the algorithm we used 15 iterations (typically the algorithm found a good solution after only 4). The parameters of the noise model, $\{\gamma_+, \gamma_-\}$ could also be optimised, but note that if we constrain $\gamma_+ = \gamma_- = \gamma$ then the likelihood is maximised by setting γ to the proportion of the training set that is unlabelled.

We first considered an illustrative toy problem to demonstrate the capabilities of our model. We generated two-dimensional data in which two class-conditional densities interlock. There were 400 points in the original data set. Each point was labelled with probability 0.1, leading to 37 labelled points. First a standard IVM classifier was trained on the labelled data only (Figure 5, Left). We then used the null category approach to train a classifier that incorporates the unlabelled data. As shown in Figure 5 (Right), the resulting decision boundary finds a region of low data density and more accurately reflects the underlying data distribution.

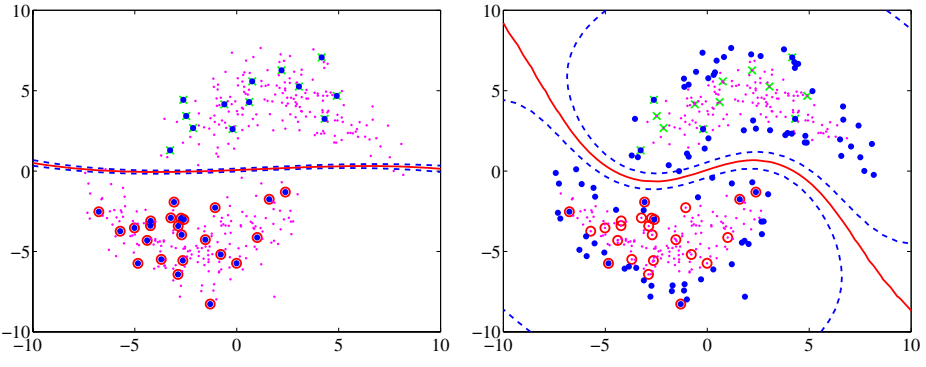


Fig. 5. Results from the toy problem. There are 400 points that are labelled with probability 0.1. Labelled data points are shown as circles and crosses. data points in the active set are shown as large dots. All other data points are shown as small dots. *Left*: Learning on the labelled data only with the IVM algorithm. All labelled points are used in the active set. *Right*: Learning on the labelled and unlabelled data with the NCNM. There are 100 points in the active set. In both plots, decision boundaries are shown as a solid line; dotted lines represent contours within 0.5 of the decision boundary (for the NCNM this is the edge of the null category).

4.5 Semi-supervised Learning: USPS Digits

We next considered the null category noise model for learning of USPS handwritten digit data set. This data set is fully labelled, but we can ignore a proportion of the labels and treat the data set as a semi-supervised task. In the experiments

that followed we used an RBF kernel with a linear component. We ran each experiment ten times, randomly selecting the data points that were labelled. The fraction of labelled points, r, was varied between 0.01 and 0.25. Each digit was treated as a separate 'one against the others' binary classification class. We also summarised these binary classification tasks in the standard way with an overall error rate. In the first of our experiments, we attempted to learn the parameters of the kernel using 8 iterations of Algorithm 2 to learn these parameters. The results are summarised in Table 1.

Table 1. Table of results for semi-supervised learning on the USPS digit data. For these results the model learned the kernel parameters. We give the results for the individual binary classification tasks and the overall error computed from the combined classifiers. Each result is summarised by the mean and standard deviation of the percent classification error across ten runs with different random seeds.

r	**0**	**1**	**2**	**3**	**4**	
0.010	17.9 ± 0.00	7.99 ± 6.48	9.87 ± 0.00	8.27 ± 0.00	9.97 ± 0.00	
0.025	11.4 ± 8.81	0.977 ± 0.10	9.87 ± 0.00	6.51 ± 2.43	9.97 ± 0.00	
0.050	1.72 ± 0.21	1.01 ± 0.10	3.66 ± 0.40	5.35 ± 2.67	7.41 ± 3.50	
0.10	1.73 ± 0.14	0.947 ± 0.14	3.17 ± 0.23	3.24 ± 0.30	3.33 ± 0.30	
0.25	1.63 ± 0.16	0.967 ± 0.09	2.50 ± 0.18	2.90 ± 0.20	2.77 ± 0.08	
r	**5**	**6**	**7**	**8**	**9**	**Overall**
0.010	7.97 ± 0.00	8.47 ± 0.00	7.32 ± 0.00	8.27 ± 0.00	8.82 ± 0.00	83.3 ± 7.30
0.025	7.97 ± 0.00	8.47 ± 0.00	7.32 ± 0.00	8.27 ± 0.00	8.82 ± 0.00	64.2 ± 5.08
0.05	7.11 ± 1.94	1.69 ± 0.15	7.32 ± 0.00	7.42 ± 1.89	7.60 ± 2.72	33.3 ± 7.15
0.1	3.02 ± 0.27	1.48 ± 0.05	1.32 ± 0.08	3.40 ± 0.15	1.95 ± 0.25	7.67 ± 0.19
0.25	2.38 ± 0.19	1.25 ± 0.17	1.19 ± 0.06	2.61 ± 0.25	1.59 ± 0.15	6.44 ± 0.22

It is immediately apparent that for values of r below 0.1 a sensible decision boundary is not found for many of the binary classification tasks. At first sight, this reflects badly on the approach: many semi-supervised learning algorithms give excellent performance even when the proportion of unlabelled data is so low. However here it must be bourne in mind that the algorithm has the additional burden of learning the kernel parameters. Most other approaches do not have this capability and therefore results are typically reported for a given set of kernel parameters. For this reason we also undertook experiments using kernel parameters that were precomputed by an IVM trained on the fully labelled data set. These results are summarised in Table 2. As expected these results are much better in the range where $r < 0.1$. With the exception of the digit 2 at $r = 0.01$ a sensible decision boundary was learned for at least one of the runs even when $r = 0.01$.

We would certainly expect the results to be better in this second experiment as we are providing more information (in terms of precomputed kernel parameters) to the model. However it is very encouraging that for $r > 0.1$ the results of both experiments are similar.

Table 2. Table of results for semi-supervised learning on the USPS digit data. For these results the model was given the kernel parameters learnt by the IVM on the standard fully labelled data. We give the results for the individual binary classification tasks and the overall error computed from the combined classifiers.

r	**0**	**1**	**2**	**3**	**4**	
0.010	3.17 ± 5.17	13.3 ± 14.18	9.87 ± 0.00	3.09 ± 0.22	8.32 ± 2.63	
0.025	1.50 ± 0.18	1.52 ± 0.87	5.18 ± 1.95	2.90 ± 0.19	4.36 ± 2.08	
0.050	1.50 ± 0.15	1.24 ± 0.22	3.37 ± 0.35	2.85 ± 0.14	3.27 ± 0.19	
0.10	1.46 ± 0.09	1.24 ± 0.13	2.82 ± 0.16	2.75 ± 0.19	3.07 ± 0.22	
0.25	1.40 ± 0.15	1.31 ± 0.16	2.44 ± 0.21	2.61 ± 0.19	2.80 ± 0.16	
r	**5**	**6**	**7**	**8**	**9**	**Overall**
0.010	7.50 ± 0.99	7.70 ± 8.48	12.3 ± 16.91	7.48 ± 1.23	35.2 ± 22.90	42.3 ± 10.05
0.025	5.03 ± 1.30	1.60 ± 0.15	1.93 ± 1.90	4.32 ± 0.45	9.93 ± 8.51	140 ± 6.06
0.050	3.63 ± 0.56	1.49 ± 0.11	1.28 ± 0.09	4.07 ± 0.43	2.59 ± 1.33	8.43 ± 0.66
0.10	2.75 ± 0.22	1.30 ± 0.10	1.30 ± 0.14	3.48 ± 0.26	1.97 ± 0.24	7.24 ± 0.51
0.25	2.32 ± 0.19	1.15 ± 0.10	1.15 ± 0.05	2.74 ± 0.19	1.62 ± 0.16	6.10 ± 0.41

5 Multi-task Learning

In this section, we show how the IVM approach can be extended to handle the situation where we have multiple independent tasks (see also [12]).

The idea behind multi-task learning is that the tasks are related and that an algorithm that makes use of these relationships may allow us to adapt to an additional task with very little training data [1,24,4].

From a Bayesian perspective the multi-task learning problem can be approached as an instance of the general hierarchical Bayesian approach to sharing strength among related statistical models [9,10]. The component models ("tasks") are assumed to be independent given an underlying latent variable; marginalising across this variable couples these components. Inferentially, learning about one task updates the posterior distribution for the latent variable, which provides a sharpened prior distribution for learning a subsequent task.

In our setting, one natural hierarchical model to consider is a model in which there are M Gaussian process models, $p(\mathbf{y}_m|\mathbf{X}_m, \boldsymbol{\theta})$, for $m = 1, \ldots, M$, and in which the kernel hyperparameter $\boldsymbol{\theta}$ is shared among the component models. Conditional on $\boldsymbol{\theta}$ the component models are assumed independent. While in a full Bayesian approach we would place a prior distribution on $\boldsymbol{\theta}$ and integrate over its posterior, in the current section we again make the empirical Bayesian approximation and find a single best-fitting value of $\boldsymbol{\theta}$ by maximising the marginal likelihood.

This setup is depicted as a graphical model in Figure 6. We model the output data for each task, $\mathbf{y}_m$, as a GP so that the probability distribution for the matrix $\mathbf{Y}$, whose columns are $\mathbf{y}_m$, is

$$p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\theta}) = \prod_{m=1}^{M} p(\mathbf{y}_m|\mathbf{X}_m, \boldsymbol{\theta})$$

where each $p(\mathbf{y}_m|\mathbf{X}_m, \boldsymbol{\theta})$ is a Gaussian process.

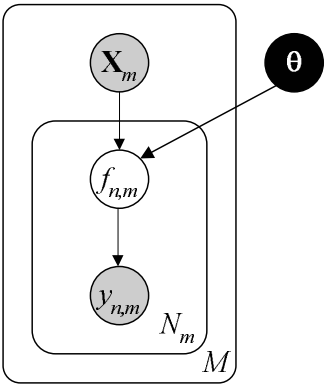


Fig. 6. A graphical model that represents a multi-task GP, compare with the single task GP in Figure 1.

The overall likelihood can be considered to be a Gaussian process over a vector $\mathbf{y}$ that is formed by stacking columns of $\mathbf{Y}$, $\mathbf{y} = \left[\mathbf{y}_1^{\mathrm{T}} \dots \mathbf{y}_M^{\mathrm{T}}\right]^{\mathrm{T}}$. The covariance matrix is then

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_1 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_2 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{K}_M \end{bmatrix}$$

and we write

$$p\left(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}\right) = N\left(\mathbf{0}, \mathbf{K}\right). \tag{30}$$

This establishes the equivalence of the multi-task GP to a standard GP. Once again we obtain an estimate, $\hat{\boldsymbol{\theta}}$, of the parameters by optimising the log-likelihood with respect to the kernel parameters. These gradients require the inverse of $\mathbf{K}$ and, while we can take advantage of its block diagonal structure to compute this inverse, we are still faced with inverting $N_m \times N_m$ matrices, where N_m is the number of data points associated with task m: however we can look to the IVM algorithm to sparsify the GP specified by (30).

It is straightforward to show that the new posterior covariance structure after k inclusions, $q_k\left(\mathbf{f}\right)$, also is a Gaussian with a block-diagonal covariance matrix,

$$q_k\left(\mathbf{f}\right) = \prod_{m=1}^{M} N\left(\mathbf{f}_m|\boldsymbol{\mu}_i^{(m)}, \Sigma_i^{(m)}\right). \tag{31}$$

Note that k inclusions in total does not mean k inclusions for each task. Each block of the posterior covariance is

$$\Sigma_i^{(m)} = \mathbf{K}_m - \mathbf{M}_i^{(m)\,\mathrm{T}} \mathbf{M}_i^{(m)},$$

where the rows of $\mathbf{M}_i^{(m)}$ are given by $\sqrt{\nu_{in_i}^{(m)}}\mathbf{s}_{i-1,n_i}^{(m)}$. The means associated with each task are given by

$$\boldsymbol{\mu}_i^{(m)} = \boldsymbol{\mu}_{i-1}^{(m)} + g_{in_i}^{(m)}\mathbf{s}_{i-1,n_i}^{(m)} \tag{32}$$

and updates of $\boldsymbol{\varsigma}_i^{(m)}$ can still be achieved through

$$\boldsymbol{\varsigma}_i^{(m)} = \boldsymbol{\varsigma}_{i-1}^{(m)} - \nu_{i,n_i}^{(m)}\mathrm{diag}\left(\mathbf{s}_{i-1,n_i}^{(m)}\ \mathbf{s}_{i-1,n_i}^{(m)\ \mathrm{T}}\right). \tag{33}$$

From (32) and (33) it is obvious that the updates of $q_i^{(m)}(\mathbf{f}_m)$ are performed independently for each of the M models. Point selection, however, should be performed across models, allowing the algorithm to select the most informative point both within and across the different tasks.

$$\Delta H_{in}^{(m)} = -\frac{1}{2}\log\left(1 - \nu_{in}^{(m)}\varsigma_{i-1,n}^{(m)}\right).$$

We also need to maintain an active, $I^{(m)}$, and an inactive, $J^{(m)}$, set for each task. The details of the algorithm are given in Algorithm 3.

The effect of selecting across tasks, rather than selecting independently within tasks is shown by a simple experiment in Figure 7. Here there are three tasks, each contains 30 data points sampled from sine waves with frequency $\frac{\pi}{5}$ and differing offsets. The tasks used different distributions for the input data: in the first it was sampled from a strongly bimodal distribution; in the second it was sampled from a zero mean Gaussian with standard deviation of 2 and in the third task data was sampled uniformly from the range $[-15, 15]$. An MT-IVM with $d = 15$ and an RBF kernel of width 1 was trained on the data. The data points that the MT-IVM used are circled. Note that all but six of the points came from the third task. The first and second task contain less information because the input data is less widely distributed, thus the MT-IVM algorithm relies most heavily on the third task. This toy example illustrates the importance of selecting the data points from across the different tasks.

To determine the kernel parameters, we again maximise the approximation to the likelihood,

$$p(\mathbf{Y}) \approx \prod_{m=1}^{M} p\left(\mathbf{z}_m|\mathbf{0}, \mathbf{K}_m + \mathbf{B}_m^{-1}\right).$$

5.1 Multi-task Learning of Speech

We considered a speech data set from the UCI repository [3]. The data consist of 15 different speakers saying 11 different phonemes 6 times each (giving 66 training points for each speaker). By considering the each speaker as a separate task we sought kernel parameters that are appropriate for classifying the different phonemes, *i.e.* we considered that each speaker is independent given the kernel parameters associated with the phoneme. We used 14 of the speakers to

Algorithm 3. The multi-task IVM algorithm.

Require: d the number of active points.
  for $m = 1$ to M **do**
    For classification $\mathbf{z}^{(m)} = \mathbf{0}$ and $\boldsymbol{\beta}^{(m)} = \mathbf{0}$. For regression substitute appropriate target values. Take $\boldsymbol{\varsigma}_0^{(m)} = \text{diag}(\mathbf{K}_m)$, $\boldsymbol{\mu}^{(m)} = \mathbf{0}$, $J^{(m)} = \{1, \ldots, N_m\}$, $\mathbf{M}_0^{(m)}$ is an empty matrix.
  end for
  for $k = 1$ to d **do**
    for $m = 1$ to M **do**
      for all $n \in J^{(m)}$ **do**
        compute $g_{kn}^{(m)}$ according to (14) (not required for Gaussian).
        compute $\nu_{kn}^{(m)}$ according to (15) or (10).
        compute $\Delta H_{kn}^{(m)}$ according to (19).
      end for
      {Comment: Select largest reduction in entropy for each task.}
      $\Delta H_k^{(m)} = \max_n \Delta H_{kn}^{(m)}$
      $n_k^{(m)} = \text{argmax}_{n \in J} \Delta H_{kn}$.
    end for
    {Comment: Select the task with the largest entropy reduction.}
    $m_k = \text{argmax}_{m \in J} \Delta H_k^{(m)}$, $n_i = n_k^{(m_k)}$
    Update m_{n_i} and $\beta_{n_i}^{(m_k)}$ using (17) and (18).
    Compute $\boldsymbol{\varsigma}_i^{(m_k)}$ and $\boldsymbol{\mu}_i^{(m_k)}$ using (21), (24) and (25).
    Append $\sqrt{\nu_{in_i}^{(m_k)}}\, \mathbf{s}_{i-1,n_i}^{(m_k)\,\mathrm{T}}$ to $\mathbf{M}_{i-1}^{(m_k)}$ using (21) to form $\mathbf{M}_i$.
    Add n_i to $I^{(m_k)}$ and remove n_i from $J^{(m)}$.
  end for

learn the kernel parameters for each phoneme giving 14 tasks. Model evaluation was then done by taking one example of each phoneme from the remaining speaker (11 points) and using this data to construct a new Gaussian process model based on those kernel parameters. Then we evaluated this model's performance on the remaining 55 points for that speaker. This mechanism was used for both an MT-IVM model and an ADF trained GP where points were randomly sub-sampled.

To demonstrate the utility of the multi-task framework we also built a IVM based Gaussian process model on the 14 speakers ignoring which speaker was associated with each data point (924 training points). The kernel parameters were optimised for this model and then the model was evaluated as above.

For enough information to be stored by the kernel about the phonemes it needs to be 'parameter rich'. We therefore used used an ARD RBF kernel in combination with an ARD linear kernel, a white noise and constant component. This leads to a total of 15 parameters for the kernel.

The results on the speech data are shown in Figure 8. The convergence of MT-IVM to $\approx 10\%$ error is roughly 10 times faster than the IVM. The MT-IVM takes advantage of the assumed independence to train much faster than the regular IVM. While this data set is relatively small, the structure of this

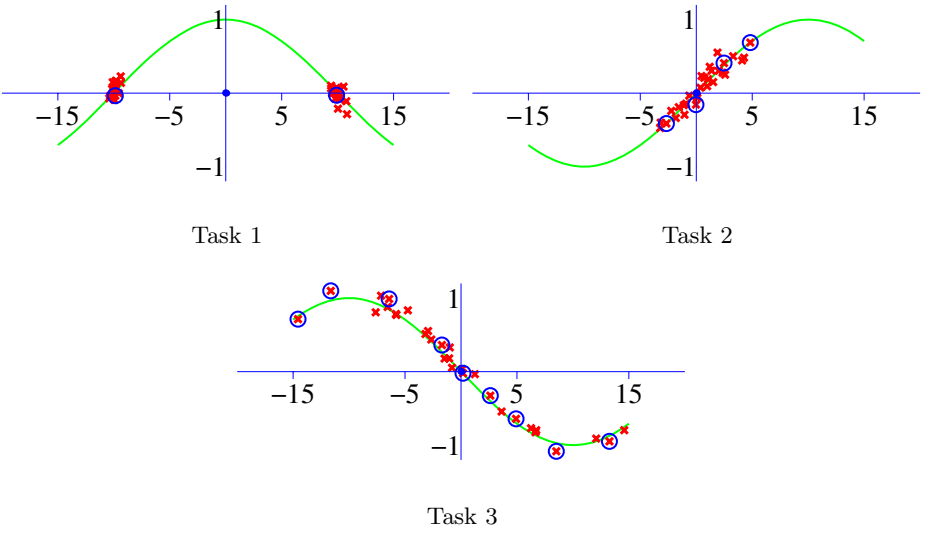


Fig. 7. Three different learning tasks sampled from sine waves. The input distribution for each task is different. Points used by the MT-IVM are circled. Note that more points are taken from tasks that give more information about the function.

experiment is important. One reason for the popularity of the HMM/mixture of Gaussians in speech recognition is the ease with which these generative models can be modified to take account of an individual speakers—this is known as speaker-dependent recognition. Up until now it has not been clear how to achieve this with discriminative models. The approach we are suggesting may be applicable to large vocabulary word recognisers and used in speaker-dependent recognition.

6 Incorporating Invariances

A learning algorithm can often have its performance improved if invariances present within a data set can be handled. Invariances occur very commonly, for example, in image based data sets. Images can often be rotated or translated without affecting the class of object they contain. The brute force approach to handling such invariances is simply to augment the original data set with data points that have undergone the transformation to which the data is considered invariance. Naturally, applying this technique leads to a rapid expansion in the size of the data set. A solution to this problem that has been suggested in the context of the SVM is to augment only the support vectors (those data points that are included in the expansion of the final solution) [18]. If these augmented points are then included in the final solution they are known as 'virtual support vectors'. The resulting algorithm yields state-of-the-art performance on the USPS data set. Since the IVM also maintains a sparse representation of the data set it seems natural to use the same idea in the IVM context.

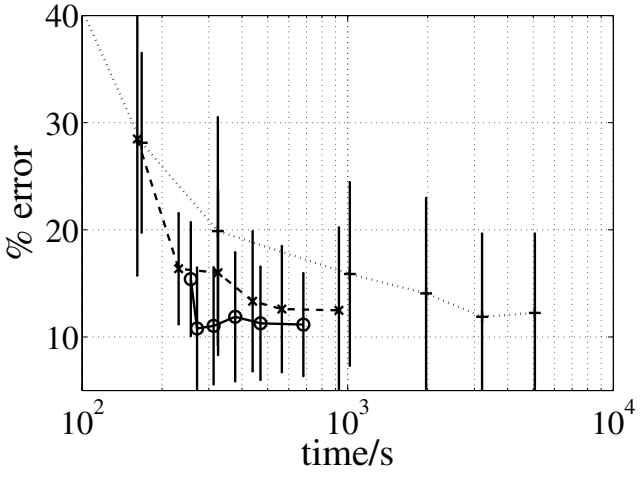


Fig. 8. Average average time vs error rate for a MT-IVM (solid line with circles) and sub-sampled ADF-GP (dashed line with crosses) whose kernel parameters are optimised by considering each speaker to be an independent task and an IVM optimised by considering all points to belong to the same task (dotted line with pluses).

6.1 USPS with Virtual Informative Vectors

In [18], it was found that the biggest improvement in performance was obtained when using translation invariances. We therefore only considered translation invariances in our experiments. We first applied the standard IVM classification algorithm to the data set using an RBF kernel combined with a linear term. We used 8 iterations of Algorithm 2 for learning the kernel parameters. We then took the resulting active set from these experiments and used it to construct an augmented data set. Each augmented data set had the original active set examples plus four translations of these examples each by one pixel in the up, down, left and right directions as prescribed in [18]. This results in an augmented active set of 2500 points. We then reselected an active set of size $d = 1000$ from this augmented active set using the standard IVM algorithm without further learning of the kernel parameters. The results are shown in Table 3. The resulting performance of the IVM with 'virtual informative vectors' is very similar to that found through the use of 'virtual support vectors'. However with the IVM all the model selection was performed completely automatically.

7 Discussion

In this paper, we have reviewed and extended the IVM algorithm from a standard classification and regression technique to an approach that can perform semi-supervised learning; multi-task learning; and incorporate invariances into the model. The IVM algorithm is as computationally efficient as the popular SVM

Table 3. Table of results for when using virtual informative vectors on the USPS digit data. The figures show the results for the individual binary classification tasks and the overall error computed from the combined classifiers. The experiments are summarised by the mean and variance of the % classification error across ten runs with different random seeds.

0	**1**	**2**	**3**	**4**	
0.648 ± 0.00	0.389 ± 0.03	0.967 ± 0.06	0.683 ± 0.05	1.06 ± 0.02	
5	**6**	**7**	**8**	**9**	**Overall**
0.747 ± 0.06	0.523 ± 0.03	0.399 ± 0.00	0.638 ± 0.04	0.523 ± 0.04	3.30 ± 0.03

and typically as performant [14]. However the IVM algorithm is an approximation to a Gaussian process and as such it sits within the wider framework of Bayesian models. This firstly allowed us to use the hierarchical Bayesian machinery to extend the IVM to so that it may perform multi-task learning and secondly it allowed us to easily incorporate a simple noise model designed to accommodate the cluster hypothesis for semi-supervised learning.

Acknowledgements

NL acknowledges the support of the EPSRC grant 'Learning Classifiers from Sloppily Labelled Data'. MJ acknowledges the support of NSF grant 0412995.

References

1. J. Baxter. Learning internal representations. In *Proc. COLT*, volume 8, pages 311–320. Morgan Kaufmann Publishers, 1995.
2. S. Becker, S. Thrun, and K. Obermayer, editors. *Advances in Neural Information Processing Systems*, volume 15, Cambridge, MA, 2003. MIT Press.
3. C. L. Blake and C. J. Merz. UCI repository of machine learning databases, 1998.
4. R. Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.
5. O. Chapelle, J. Weston, and B. Schölkopf. Cluster kernels for semi-supervised learning. In Becker et al. [2].
6. C. Cortes and V. N. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.
7. L. Csató. *Gaussian Processes — Iterative Sparse Approximations*. PhD thesis, Aston University, 2002.
8. L. Csató and M. Opper. Sparse representation for Gaussian process models. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13, pages 444–450, Cambridge, MA, 2001. MIT Press.
9. A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian Data Analysis*. Chapman and Hall, 1995.
10. R. E. Kass and D. Steffey. Approximate Bayesian inference in conditionally independent hierarchical models (parametric empirical Bayes models). *Journal of the American Statistical Association*, 84:717–726, 1989.
11. N. D. Lawrence and M. I. Jordan. Semi-supervised learning via Gaussian processes. In *Advances in Neural Information Processing Systems*, volume 17, Cambridge, MA, 2005. MIT Press. To appear.

12. N. D. Lawrence and J. C. Platt. Learning to learn with the informative vector machine. In R. Greiner and D. Schuurmans, editors, *Proceedings of the International Conference in Machine Learning*, volume 21, pages 512–519, San Francisco, CA, 2004. Morgan Kauffman.
13. N. D. Lawrence and B. Schölkopf. Estimating a kernel Fisher discriminant in the presence of label noise. In C. Brodley and A. P. Danyluk, editors, *Proceedings of the International Conference in Machine Learning*, volume 18, San Francisco, CA, 2001. Morgan Kauffman.
14. N. D. Lawrence, M. Seeger, and R. Herbrich. Fast sparse Gaussian process methods: The informative vector machine. In Becker et al. [2], pages 625–632.
15. D. J. C. MacKay. *Bayesian Methods for Adaptive Models*. PhD thesis, California Institute of Technology, 1991.
16. T. P. Minka. *A family of algorithms for approximate Bayesian inference*. PhD thesis, Massachusetts Institute of Technology, 2001.
17. I. T. Nabney. *Netlab: Algorithms for Pattern Recognition*. Advances in Pattern Recognition. Springer, Berlin, 2001. Code available from `http://www.ncrg.aston.ac.uk/netlab/`.
18. B. Schölkopf, C. J. C. Burges, and V. N. Vapnik. Incorporating invariances in support vector learning machines. In *Artificial Neural Networks — ICANN'96*, volume 1112, pages 47–52, 1996.
19. B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, 2001.
20. M. Seeger. Covariance kernels from Bayesian generative models. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14, pages 905–912, Cambridge, MA, 2002. MIT Press.
21. M. Seeger. *Bayesian Gaussian Process Models: PAC-Bayesian Generalisation Error Bounds and Sparse Approximations*. PhD thesis, The University of Edinburgh, 2004.
22. H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Conference on Computational Learning Theory 10*, pages 287–294. Morgan Kauffman, 1992.
23. P. Sollich. Probabilistic interpretation and Bayesian methods for support vector machines. In *Proceedings 1999 International Conference on Artificial Neural Networks, ICANN'99*, pages 91–96, London, U.K., 1999. The Institution of Electrical Engineers.
24. S. Thrun. Is learning the n-th thing any easier than learning the first? In Touretzky et al. [25], pages 640–646.
25. D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors. *Advances in Neural Information Processing Systems*, volume 8, Cambridge, MA, 1996. MIT Press.
26. V. N. Vapnik. *Statistical Learning Theory*. John Wiley and Sons, New York, 1998.
27. C. K. I. Williams. Computing with infinite networks. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, Cambridge, MA, 1997. MIT Press.
28. C. K. I. Williams and C. E. Rasmussen. Gaussian processes for regression. In Touretzky et al. [25], pages 514–520.

A ADF Mean and Covariance Updates

In Section 2.2, we stated that the updates to the mean function and the covariance function for the ADF algorithm could be expressed in terms of the gradients of

$$Z_i = \int t_{n_i}(\mathbf{f})\, q_{i-1}(\mathbf{f})\, d\mathbf{f}.$$

In this appendix, we derive these results, see also [16,7,21]. First we consider that the mean under $q_i(\mathbf{f})$ is given by

$$\langle \mathbf{f} \rangle = Z_i^{-1} \int \mathbf{f} t_{n_i}(\mathbf{f})\, q_{i-1}(\mathbf{f})\, d\mathbf{f}.$$

At this point we could substitute in the relevant noise model for $t_{n_i}(\mathbf{f})$ and compute the expectation, however it turns out that we can express this expectation, and the second moment, in terms of gradients of the log partition function. This is convenient, as it means we can rapidly compute the update formula for novel noise models. To see how this is done we first note that

$$\nabla_{\boldsymbol{\mu}_{i-1}} q_{i-1}(\mathbf{f}) = \Sigma_{i-1}^{-1}\left(\mathbf{f} - \boldsymbol{\mu}_{i-1}\right) q_{i-1}(\mathbf{f})$$

which can be re-expressed in terms of $\mathbf{f} q_{i-1}(\mathbf{f})$,

$$\mathbf{f} q_{i-1}(\mathbf{f}) = \Sigma_{i-1} \nabla_{\boldsymbol{\mu}_{i-1}} q_{i-1}(\mathbf{f}) + \boldsymbol{\mu}_{i-1} q_{i-1}(\mathbf{f}),$$

now multiplying both sides by $Z_i^{-1} t_{n_i}(\mathbf{f})$ and integrating over $\mathbf{f}$ gives

$$\begin{aligned}\langle \mathbf{f} \rangle_i &= \boldsymbol{\mu}_{i-1} + Z_i^{-1} \Sigma_{i-1} \nabla_{\boldsymbol{\mu}_{i-1}} \int t_{n_i}(\mathbf{f})\, q_{i-1}(\mathbf{f})\, d\mathbf{f} \\ &= \boldsymbol{\mu}_{i-1} + Z_i^{-1} \Sigma_{i-1} \nabla_{\boldsymbol{\mu}_{i-1}} Z_i\end{aligned}$$

$$\langle \mathbf{f} \rangle_i = \boldsymbol{\mu}_{i-1} + \Sigma_{i-1} \mathbf{g}_i \tag{34}$$

where we have defined $\mathbf{g}_i = \nabla_{\boldsymbol{\mu}_{i-1}} \log Z_i$. The second moment matrix is given by

$$\left\langle \mathbf{f}\mathbf{f}^{\mathrm{T}} \right\rangle_i = Z_i^{-1} \int \mathbf{f}\mathbf{f}^{\mathrm{T}} t_{n_i}(\mathbf{f})\, q_{i-1}(\mathbf{f})\, d\mathbf{f}$$

Once again we take gradients of q_{i-1}, but this time with respect to the covariance matrix Σ_{i-1}.

$$\nabla_{\Sigma_{i-1}} q_{i-1}(\mathbf{f}) = \left(-\frac{1}{2}\Sigma_{i-1}^{-1} + \frac{1}{2}\Sigma_{i-1}^{-1}\left(\mathbf{f} - \boldsymbol{\mu}_{i-1}\right)\left(\mathbf{f} - \boldsymbol{\mu}_{i-1}\right)^{\mathrm{T}} \Sigma_{i-1}^{-1} \right) q_{i-1}(\mathbf{f})$$

which can be re-arranged, as we did for $\langle \mathbf{f} \rangle_i$ to obtain

$$\begin{aligned}\left\langle \mathbf{f}\mathbf{f}^{\mathrm{T}} \right\rangle_i = \Sigma_{i-1} + 2\Sigma_{i-1} \Gamma^{(i)} \Sigma_{i-1} + \langle \mathbf{f} \rangle_i \boldsymbol{\mu}_{i-1}^{\mathrm{T}} \\ + \boldsymbol{\mu}_{i-1} \langle \mathbf{f} \rangle_i^{\mathrm{T}} - \boldsymbol{\mu}_{i-1}\boldsymbol{\mu}_{i-1}^{\mathrm{T}}\end{aligned}$$

where

$$\Gamma^{(i)} = \nabla_{\Sigma_{i-1}} \log Z_i.$$

The update (7) for the covariance requires

$$\left\langle \mathbf{f}\mathbf{f}^{\mathrm{T}} \right\rangle_i - \langle \mathbf{f} \rangle_i \left\langle \mathbf{f}^{\mathrm{T}} \right\rangle_i = \Sigma_{i-1} - \Sigma_{i-1} \left(\mathbf{g}_i \mathbf{g}_i^{\mathrm{T}} - 2\Gamma^{(i)} \right) \Sigma_{i-1}. \tag{35}$$

Substituting (34) and (35) into (6) and (7) we obtain the required updates for the mean and covariance in a form that applies regardless of our noise model.

$$\boldsymbol{\mu}_i = \boldsymbol{\mu}_{i-1} + \Sigma_{i-1} \mathbf{g}_i \tag{36}$$

$$\Sigma_i = \Sigma_{i-1} - \Sigma_{i-1} \left(\mathbf{g}_i \mathbf{g}_i^{\mathrm{T}} - 2\Gamma^{(i)} \right) \Sigma_{i-1} \tag{37}$$

B Kernels Used in Experiments

Throughout the experiments discussed in the main text we construct and learn the parameters of a range of different covariance functions. In this appendix, we give an overview of all the kernels used.

A covariance function can be developed from any positive definite kernel. A new kernel function can also be formed by adding kernels together. In the experiments we present we principally made use of the following three kernel matrices.

The inner product kernel constrains the process prior to be over linear functions,

$$k_{\mathrm{lin}}(\mathbf{x}_i, \mathbf{x}_j) = \theta_{\mathrm{lin}} \mathbf{x}_i^{\mathrm{T}} \mathbf{x}_j,$$

where θ is the process variance and controls the scale of the output functions. Non linear functions may be obtained through the RBF kernel,

$$k_{\mathrm{rbf}}(\mathbf{x}_i, \mathbf{x}_j) = \theta_{\mathrm{rbf}} \exp\left(-\frac{\gamma}{2} (\mathbf{x}_i - \mathbf{x}_j)^{\mathrm{T}} (\mathbf{x}_i - \mathbf{x}_j) \right),$$

where γ is the inverse width parameter, which leads to infinitely differentiable functions. Finally we considered the MLP kernel [27] which is derived by considering a multi-layer perceptron in the limit of an infinite number of hidden units,

$$k_{\mathrm{mlp}}(\mathbf{x}_i, \mathbf{x}_j) = \theta_{\mathrm{mlp}} \sin^{-1}\left(\frac{w \mathbf{x}_i^{\mathrm{T}} \mathbf{x}_j + b}{\sqrt{\left(w \mathbf{x}_i^{\mathrm{T}} \mathbf{x}_i + b + 1\right)\left(w \mathbf{x}_j^{\mathrm{T}} \mathbf{x}_j + b + 1\right)}} \right)$$

where we call w the weight variance and b the bias variance (they have interpretations as the variances of prior distributions in the neural network model).

In combination with these kernels we often make use of a white noise process

$$k_{\mathrm{white}}(\mathbf{x}_i, \mathbf{x}_j) = \theta_{\mathrm{white}} \delta_{ij}$$

where δ_{ij} is the Kronecker delta [5]. It is possible to represent uncertainty in the bias by adding a constant term to the kernel matrix,

$$k_{\text{bias}}(\mathbf{x}_i, \mathbf{x}_j) = \theta_{\text{bias}}$$

where we have denoted the variance, θ_{bias}.

All the parameters we have introduced in these kernels need to be constrained to be positive. In our experiments, this constraint was implemented by re-parameterising,

$$\theta = \log(1 + \exp(\theta')).$$

Note that as our transformed parameter $\theta' \to -\infty$ the parameter $\theta \to 0$ and as $\theta' \to \infty$ we see that $\theta \to \theta'$.

Finally we can also consider automatic relevance determination (ARD) versions of each of these covariance functions. These process priors are formed by replacing any inner product, $\mathbf{x}_i^{\mathrm{T}}\mathbf{x}_j$, with a matrix inner product, $\mathbf{x}_i^{\mathrm{T}}\mathbf{A}\mathbf{x}_j$. If $\mathbf{A}$ is positive (semi-)definite then each kernel is still valid. The ARD kernels are the specific case where $\mathbf{A}$ is a diagonal matrix, $\mathbf{A} = \text{diag}(\boldsymbol{\alpha})$ and the ith diagonal element, α_i, provides a scaling on the ith input variable, these input scales are constrained to be between 0 and 1 by re-parameterising with a sigmoid function,

$$\alpha_i = \frac{1}{1 + \exp(-\alpha')}.$$

Unless otherwise stated the kernel parameters were initialised with $\theta_{\text{lin}} = 1$, $\theta_{\text{rbf}} = 1$, $\gamma = 1$, $\theta_{\text{mlp}} = 1$, $w = 10$, $b = 10$ and $\boldsymbol{\alpha} = [0.999, \ldots, 0.999]^{\mathrm{T}}$.

[5] The Kronecker delta δ_{ij} is zero unless $i = j$ when it takes the value 1.

Efficient Communication by Breathing

Tom H. Shorrock, David J.C. MacKay, and Chris J. Ball

Cavendish Laboratory, Cambridge, CB3 0HE, United Kingdom

Abstract. The arithmetic-coding-based communication system, Dasher, can be driven by a one-dimensional continuous signal. A belt-mounted breath-mouse, delivering a signal related to lung volume, enables a user to communicate by breath alone. With practice, an expert user can write English at 15 words per minute.

Dasher is a communication system based on a beautiful idea from information theory called arithmetic coding (Witten et al., 1987; MacKay, 2003, Chapter 6). Arithmetic coding is an optimal method for text-*compression* using a language model. By turning arithmetic coding on its head, we obtain an optimal method for text-*generation*.

We view a person's gestures as a source of information, and the sentences they wish to communicate as the sink of information. Good interface design maximizes the number of bits per second that are conveyed from the user into text. Poor interfaces waste the user's time either by failing to extract all the bits that the user could easily generate, or by diverting the user's bits into redundant activity.

The Dasher approach to interface design decouples the issues of efficient bit-generation and efficient language-generation. Unlike in most interfaces, a Dasher-user's gestures have no relationship to particular symbols in the language. Instead, they control *navigation* in a continuous space whose contents are laid out using a language model. For demonstrations, or to try Dasher for yourself – it's free – please visit `www.inference.phy.cam.ac.uk/dasher/`.

The objective of this paper is to offer a new method for helping a disabled person to communicate by breath alone. In contrast to widely used switch-scanning systems, our method makes use of fine breathing control. Of course, not everyone has fine breath control, but to those who have, we would like to offer the chance to make use of that information, rather than discard it.

1 How Dasher Works

Imagine writing a piece of text by going into the library that contains *all possible books*, and finding the book that contains exactly that text. In this way, writing can be turned into a navigational task. What is written is determined by where the user goes. In Dasher's idealized library, the 'books' are arranged alphabetically on one enormous shelf. When the user points at a part of the shelf, the view zooms in continuously on that part of the shelf. To write a message that

J. Winkler, N. Lawrence, M. Niranjan (Eds.): Machine Learning Workshop, LNAI 3635, pp. 88–97, 2005.

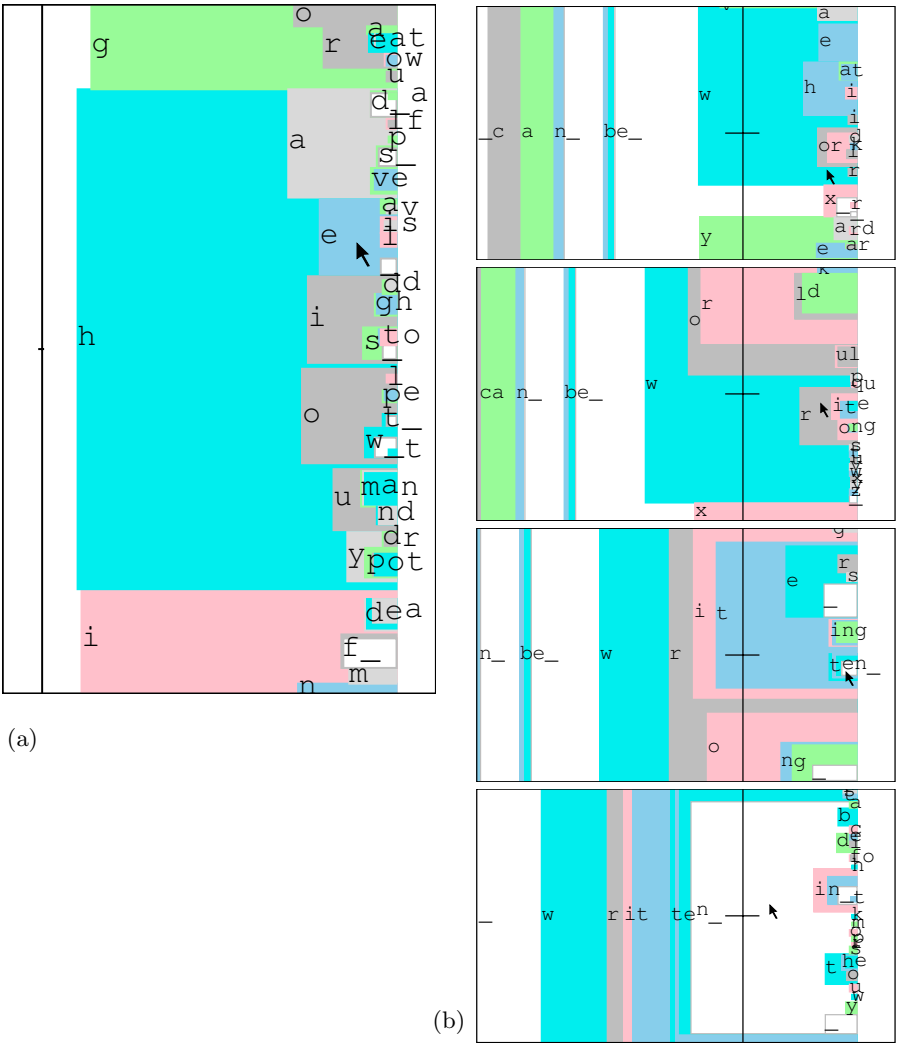


Fig. 1. A Screenshot of Dasher when the user starts writing `hello`. The shelf of the alphabetical 'library' is displayed vertically. The space character, '_', is included in the alphabet after `z`. In panel (a), the user has zoomed in on the portion of the shelf containing messages beginning with `g`, `h`, and `i`. Following the letter `h`, the language model makes the letters `a`, `e`, `i`, `o`, `u`, and `y` easier to write by giving them more V space. Common words such as `had` and `have` are visible. The pointer's vertical coordinate controls the point that is zoomed in on, and its horizontal coordinate controls the rate of zooming; pointing to the left makes the view zoom out, allowing the correction of recent errors.
Panel (b) shows screenshots while the user writes '`any sentence can be written`'.

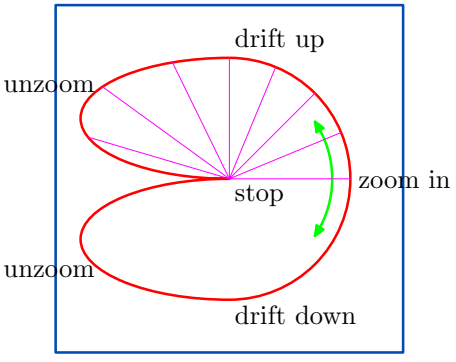


Fig. 2. Dasher's one-dimensional mode. The curved line shows the sequence of two-dimensional control positions created throughout the range of one-dimensional control positions. The central point of the display corresponds to no motion; the two ends of the one-dimensional scale both map to this point. The centre of the one-dimensional scale is mapped to the three-o-clock position, zooming in at the maximum rate. The horizontal coordinate on the curve determines the rate of zooming in or out. The radial lines indicate the direction of motion produced for some positions along the upper half of the range.

begins '`hello`', one first steers towards the section of the shelf marked `h`, where all the books beginning with `h` are found. Within this section are sections for books beginning `ha`, `hb`, `hc`, etc.; one enters the `he` section, then the `hel` section within it, and so forth.

To make the writing process efficient we use a language model, which predicts the probability of each letter in a given context, to allocate the shelf-space for each letter of the alphabet, as illustrated in figure 1. The shelf is recursively chopped up in such a way that the amount of shelf-space devoted to a string is proportional to its probability. The user's gestures are turned into steering commands, controlling the portion of the display zoomed into. If the user can generate information at a rate of, say, 5 bits per second, then our aim is to feed these bits to Dasher in such a way that, each second, the display zooms in by a factor of $2^5 = 32$ on the region containing the text required by the user. When the language model's predictions are accurate, many successive characters can be selected by a single gesture. The language model we use, PPMD5 (Cleary and Witten, 1984; Teahan, 1995), generates text at an exchange rate of about two bits per character. Thus the user will be able to write at 5/2 characters per second, or 30 words per minute. We could only beat this writing speed by enhancing the rate at which the user generates bits, or improving the predictions of the language model.

Dasher was first developed to be driven by continuous *two-dimensional* gestures, delivered via a mouse, touch screen, or gazetracker. Our experiments showed that, with Dasher, it is easy to spell correctly and hard to make spelling mistakes. Using an ordinary mouse, typical novice users reach a writing speed of

25 words per minute after 60 minutes of practice, and expert users can write at 35 words per minute (Ward et al, 2002). Results using Dasher with a gazetracker were record-breaking: after 60 minutes' practice, novice users can drive Dasher using a gazetracker at a speed of about 15 words per minute; expert users can write at 25 words per minute, and make almost no spelling mistakes (Ward and MacKay, 2002). We know of no faster method for communication by gaze.

In this paper, we discuss how Dasher can be driven by *one-dimensional* gestures.

2 Dasher's One-Dimensional Mode

In normal two-dimensional Dasher, the information content concerning the text desired by the user is conveyed entirely through the vertical dimension of the pointer. The horizontal dimension controls only the speed of text entry. Expert users of Dasher tend to write at a fairly constant zooming rate such as five bits per second. Thus the horizontal dimension is scarcely used: an expert uses it only if he makes a mistake or wishes to slow down, pause or unzoom.

In Dasher's one-dimensional mode, we select a simple one-dimensional curve from regular Dasher's two-dimensional navigation space; the single dimension conveyed by the user selects the steering direction from this curve (figure 2). The middle of the curve offers normal forward motion at a fixed zooming rate, with the one-dimensional coordinate determining the direction of forward motion. The extreme ends of the curve offer unzooming. As the control is moved from one end to the other, unzooming blends smoothly into drifting up without zooming, zooming up, zooming straight forward, zooming down, drifting down, and unzooming again. (The curve is composed of three half-ellipses.)

We can include *control nodes* in the Dasher alphabet so that the user can access special functions such as pausing and stopping by the same zooming process as is used for writing (much as an escape key can be used to access special modes in a keyboard-based editor). (Such control nodes were not used in the experiments described in the present paper.)

3 Experiments with a Breath Mouse

We obtained a continuous one-dimensional breathing signal using a breath mouse (figure 3).

3.1 Experiments on Novices

Eight volunteers from the Cavendish Laboratory staff with very little or no experience with Dasher used breath-Dasher for a total of one hour. Of the volunteers, four were women. Six had English as their first language; one, German; and one, Italian.

Our protocol was similar to that of Ward and MacKay (2002). We gave users dictation from Jane Austen's *Emma* in five minute periods. Dasher's language

Fig. 3. Our first breath mouse, made from an optical mouse, a belt, and a piece of elastic. The mouse is fixed to a piece of wood, to which a belt is also attached. Two inches of the belt are replaced by elastic, so that changes in the waist circumference produce motion of the belt underneath the eye of the mouse. This sensor measures breathing if the user breathes using their diaphragm (rather than their rib cage). We oriented the mouse so that breathing in moves the on-screen mouse up and rotates the pointer anti-clockwise along the curve; and breathing out moves the on-screen mouse down and rotates the pointer clockwise. The sensor also responds to clenching of the stomach muscles, but we encourage the user to navigate by breathing normally.

model was trained on *Emma*, excluding the dictated passages. We used a 54-letter alphabet (the twenty six letters in both upper and lower case, the space character and the full stop). Dasher was started and stopped manually at the beginning and end of each dictation period.

Each subject's twelve five-minute dictation trials were spaced out over several days. Two trials could be taken one after one another in a single session, with a few minutes' break between. The volunteers were allowed up to two sessions each day, with a maximum of three days between two consecutive trials. In one case, three sessions were conducted on a single day, with at least three hours separating successive sessions.

Before dictation all volunteers were allowed to read a paper copy of the text that they were expected to write, to try to reduce the frequency of writing-errors *not* associated with using Dasher.

After each dictation trial, the subject was offered the chance to adjust the overall speed of the interface by 5 or 10%.

The writing speeds and error rates for all 8 novices and one expert are shown in figure 4a. Figure 5 shows the speed settings chosen by the users.

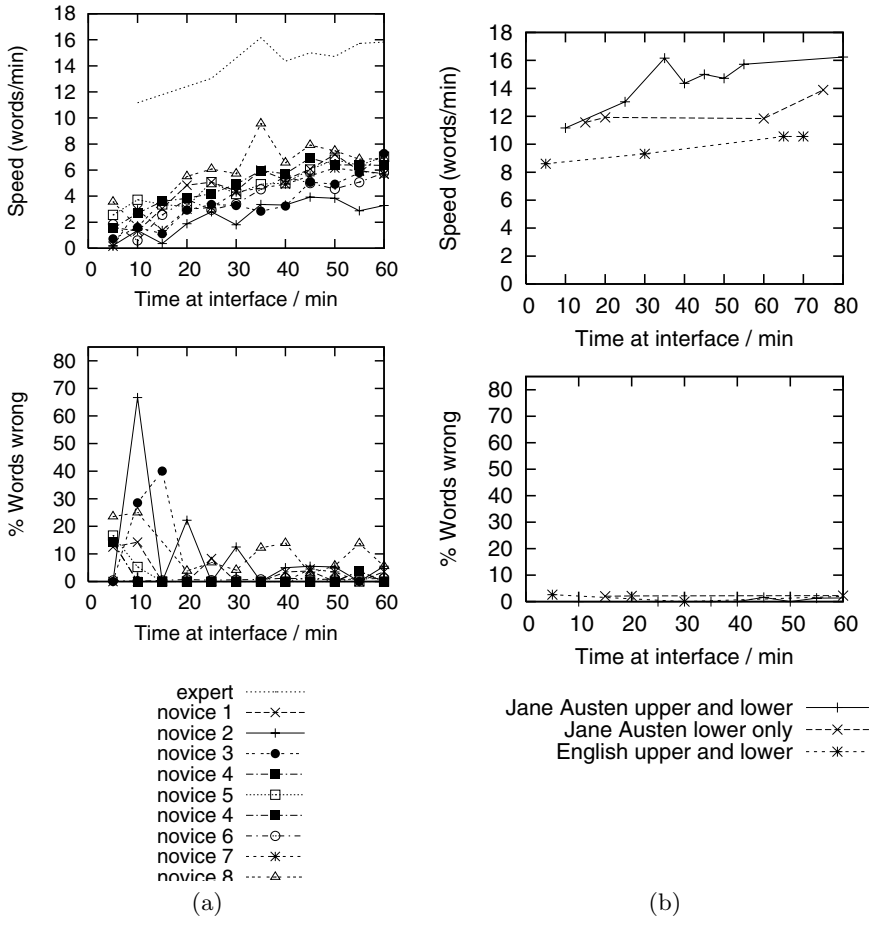


Fig. 4. (a) Breath-Dasher results for 8 Dasher novices and 1 Dasher expert. Upper graph shows writing speed in words per minute. Lower graph shows the percentage of words containing errors. (b) Expert user: results for different training texts and alphabets.

Observations. Two volunteers (novices 2 and novice 3) had difficulty controlling the breath mouse. We believe they sometimes clenched their stomach muscles instead of breathing naturally.

Most novices had difficulty finding the full stops. Users had relatively little practice in using them since in early trials a single sentence was often not completed. This inexperience was compounded by the large probability of a space character following the full stop, causing the users to not notice the full stop. In early experiments the location of the full stop often had to be pointed out. Although this problem reduced with experience, of all the letters this was the most persistently troublesome. Users generally dealt with capital letters well.

Some users had difficulties at low speeds because while the low speed was necessary for them to find their place, once the correct direction had been deter-

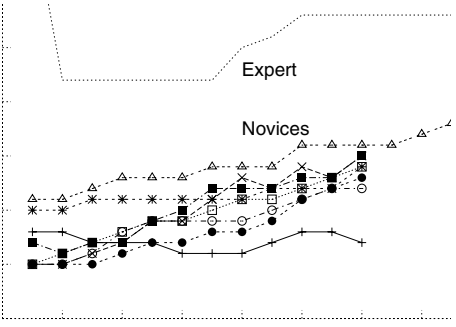


Fig. 5. Dasher speed settings chosen by each user at the start of each trial. Dasher's maximum speed is specified in bits per second.

mined they found the wait for the interface to zoom uncomfortable. Some users made use of the feature that the interface could be stopped by breathing right in or out, to give themselves time to find their place.

When the speed control was set at a low speed, zooming out at extremes of breath intake was found uncomfortably slow. With experience, this problem diminished, firstly because users noticed their mistakes earlier and so did not need to unzoom so much, and secondly because their speed was increased so they did not have to hold their breath for so long.

3.2 Expert Trials

An expert who was very familiar with Dasher (with perhaps 50 hours of use) and had considerable experience of the breath mouse (about two hours of practice before the experiment started) was also tested. We measured his performance using three different combinations of alphabet and training text, so as to quantify the effects of (1) including upper and lower case characters; (2) choosing a training text that is well matched to the dictation text.

Figure 4b shows the results.

Alphabet Choice. The top, solid line in figure 4b shows the results where the training text and alphabet were identical to those used by the novices. The second line (with crosses) shows results where the alphabet was lower-case only; the training text was the same *Emma* corpus. It is striking that *increasing* the number of characters from single-case to mixed-case, which doubles the number of letters available, actually *increases* the rate of writing.

The explanation for this result is that the mixed-case language is easier for our language model to predict. Even though the number of possible characters is twice as great, the entropy of the mixed-case language is slightly smaller. The cost of selecting occasional upper-case characters is offset by the increased predictive power of mixed-case contexts.

Training Text. The expert also took dictation of *Emma* using a Dasher system that had been trained on generic English text (the default 300 kilobyte training file of assorted English sentences from the Dasher website).

The lowest line in figure 4b shows that the writing speed drops by about 33% when a generic training text is used. Users can therefore expect a 50% increase in speed if they pre-train Dasher with texts similar to what they intend to write.

3.3 Comparison with Sip-and-Puff

Beginner users of Dasher wrote at 6.0 ± 1.3 words per minute after an hour's training, with on average 2.0% of words misspelled. An expert user can write at over 16 words per minute.

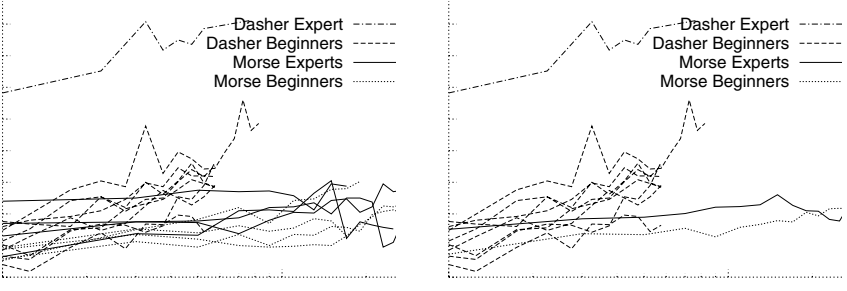


Fig. 6. Breath-Dasher writing speeds compared with writing speeds achieved by 8 sip-and-puff Morse code users. Horizontal axis shows time using the interface in minutes, *on a logarithmic scale*. Vertical axis is writing speed in words per minute. In the righthand panel, the results for the Morse users have been averaged. Morse data kindly provided by Denis Anson (Anson et al, 2003).

For comparison, one method for writing by breath is 'sip-and-puff', with sips and puffs being mapped to the dots and dashes of Morse code. One experienced sip-and-puff user reports that he can write at 17 words per minute when using a combination of Morse code and word-completion software.[1] Data on learning curves for this method were kindly provided by Denis Anson. The study by Anson et al (2003) involved 8 subjects, four of whom had no prior experience with Morse, and four of whom were radio hams with Morse experience. All subjects wrote for more than 180 minutes in 20-minute trials. The sip-and-puff with Morse writing method required no visual feedback, but did use auditory feedback: users could hear the dots and dashes they entered. The learning curves for sip-and-puff Morse are compared with those for breath-Dasher in figure 6. The plateau writing speeds reached by Morse code novices were 4.9, 2.2, 4.1, and 5.7 words per minute, with error rates of 5%, 4%, 0%, and 4%, respectively. The plateau values of Morse code experts were only a little better: 4.9, 5.2, 5.3, and 6.4 words per minute, with error rates of 3%, 4%, 5%, and 2% respectively.

[1] `http://www.makoa.org/jlubin/ahfeat4.htm`

Another widely-used method for communication by sip-and-puff is a scanning system that offers the users sequences of discrete menus to select from. Vanderheiden (1985) reported that users of scanning systems wrote at six or less words per minute; we know experienced users who can write at 12 words per minute by scanning, but have not been able to find full learning curves for this method.

We conclude that Dasher has a better learning curve than sip-and-puff with Morse. Dasher is a promising writing method for a sip-and-puff user who could convey a continuous signal with their breath. An alternative approach that would use standard sip-and-puff hardware would be to use one of the two-button modes of 'button-Dasher' (MacKay et al, 2004).

3.4 Development Ideas

In the light of users' complaints that they occasionally ran short of breath when using breath Dasher, we propose to include the option for breath-Dasher's to add a 0.1 Hz periodic signal to the one-dimensional coordinate. To steer Dasher as before, the user will have to breathe in and out to cancel the effect of this added signal.

We hope the one-dimensional mode of Dasher will also be useful for hand-held computers with tilt sensors.

Dasher is free software, distributed under the GNU General Public License, and available from `www.inference.phy.cam.ac.uk/dasher/`.

Acknowledgments

We thank Caroline Gray, David Colven, Alan Blackwell, and Matthew Garrett for helpful discussions.

We gratefully acknowledge the support of the Gatsby Charitable Foundation, Martin King, and the Nine Tuna Foundation.

References

Witten et al.(1987). Witten, I. H., Neal, R. M., and Cleary, J. G., Arithmetic coding for data compression, *Communications of the ACM*, **30**, 520–540 (1987).

MacKay(2003). MacKay, D. J. C., *Information Theory, Inference, and Learning Algorithms*, Cambridge University Press, 2003, available from `www.inference.phy.cam.ac.uk/mackay/itila/`.

Cleary and Witten(1984). Cleary, J. G., and Witten, I. H., Data compression using adaptive coding and partial string matching, *IEEE Trans. on Communications*, **32**, 396–402 (1984).

Teahan(1995). Teahan, W. J., Probability estimation for PPM (1995), Probability estimation for PPM. In *Proc. of the N.Z. Comp. Sci. Research Students' Conf.,* available from `citeseer.nj.nec.com/teahan95probability.html` (1995).

Ward et al.(2002). Ward, D. J., Blackwell, A. F., and MacKay, D. J. C., Dasher – A data entry interface using continuous gestures and language models, *Human-Computer Interaction*, **17**, 199–228 (2002).

Ward and MacKay(2002). Ward, D. J., and MacKay, D. J. C., Fast hands-free writing by gaze direction, *Nature*, **418**, 838 (2002).

Anson et al.(2003). Anson, D. K., Glodek, M., Peiffer, R. M., Rubino, C. G., and Schwartz, P. T., Long-term speed and accuracy of Morse code vs. head-pointer interface for text generation (2003), presented at RESNA 2004 Annual Conference, Orlando, FL. Available from: `atri.misericordia.edu/Papers/MorseVrsOnScreen.php`.

Vanderheiden(1985). Vanderheiden, P. J., "Writing aids," in *Electronic aids for rehabilitation*, edited by J. Webster, A. Cook, W. Tompkins, and G. Vanderheiden, Chapman and Hall, London, 1985, pp. 262–282.

MacKay et al.(2004). MacKay, D. J. C., Ball, C. J., and Donegan, M., "Efficient communication with one or two buttons," in *Proceedings of* Maximum Entropy and Bayesian Methods, edited by R. Fischer, R. Preuss, and U. von Toussaint, American Institute of Physics, Melville, New York, 2004, vol. 735 of *AIP Conference Proceedings*, pp. 207–218.

Guiding Local Regression Using Visualisation

Dharmesh M. Maniyar and Ian T. Nabney

Neural Computing Research Group, Aston University, Birmingham B4 7ET, UK
maniyard@aston.ac.uk,
http://www.ncrg.aston.ac.uk/

Abstract. Solving many scientific problems requires effective regression and/or classification models for large high-dimensional datasets. Experts from these problem domains (*e.g.* biologists, chemists, financial analysts) have insights into the domain which can be helpful in developing powerful models but they need a modelling framework that helps them to use these insights. Data visualisation is an effective technique for presenting data and requiring feedback from the experts. A single global regression model can rarely capture the full behavioural variability of a huge multi-dimensional dataset. Instead, local regression models, each focused on a separate area of input space, often work better since the behaviour of different areas may vary. Classical local models such as Mixture of Experts segment the input space automatically, which is not always effective and it also lacks involvement of the domain experts to guide a meaningful segmentation of the input space. In this paper we addresses this issue by allowing domain experts to interactively segment the input space using data visualisation. The segmentation output obtained is then further used to develop effective local regression models.

1 Introduction

The work presented here was motivated by a problem in the Chemoinformatics domain where there is a need for a computational model that relates physico-chemical properties of compounds with their biological activity. A reliable regression model would allow a screening scientist to predict the biological activity of compounds and then decide which compounds are worth physically testing.

There are many regression techniques available from the statistical and neural computing domains. Broadly they can be divided into global and local regression models. Global models use a single model for the problem which covers the entire input space. Local regression models use a combination of models, each of which applies to a smaller part of the input space.

Because of the quantity and diversity of data points (e.g. a huge chemical compound library), trying to develop a single model to make prediction for all data points (e.g. chemical compounds in a library) is unlikely to succeed. What is more likely to be effective is a group of local models, each of which working on a set of similar data points, in other words, in different regions of the input space. In this paper, we present a *guided* local regression approach which first, with the help of principled visualisation techniques, allows domain experts to create

J. Winkler, N. Lawrence, M. Niranjan (Eds.): Machine Learning Workshop, LNAI 3635, pp. 98–109, 2005.

an *informed segmentation* of the input space. Then, we use that segmentation output to develop local regression models. We compare our results with the results from classical global and local regression models.

The next section briefly describes the Mixture of Experts (ME) model since it is related to the guided regression models we introduce here. Section 3 gives a brief introduction to the Hierarchical Generative Topographic Map (HGTM) which we use for visualisation and segmentation. In Section 4 we present the guided local regression models. The experimental results are reported in Section 5. Finally, the paper ends with a discussion in Section 6.

2 Mixture of Experts (ME)

Jacobs et *al.* introduced the mixture of experts model, which determines a decomposition of the data as a part of the learning process [1]. In this model, all of the expert networks, as well as a gating network, are trained together. The goal of the training procedure is to have the gating network learn an appropriate decomposition of the input space into different regions, while each expert network learns to generate the outputs for input vectors falling within a specific region. The gating network outputs $g_i(\mathbf{x})$ can be regarded as the probability that input $\mathbf{x}$ is attributed to expert i. This probabilistic interpretation is ensured because of the choice of output for the gating network is the softmax activation function:

$$g_i = \frac{\exp(\gamma_i)}{\sum_{j=1}^{M} \exp(\gamma_i)}, \tag{1}$$

where the $\gamma_i (i = 1, 2, ..., M)$ are the outputs of the gating network and M is the number of experts.

The error function for the complete model is given by the negative logarithm of the likelihood with respect to a probability distribution given by a mixture of M Gaussians of the form

$$E = -\sum_{n} \ln \left\{ \sum_{i=1}^{M} g_i(\mathbf{x}^n) \phi_i(\mathbf{t}^n \mid \mathbf{x}^n) \right\}, \tag{2}$$

where $\mathbf{t}$ is the output vector and the $\phi_i(\mathbf{t} \mid \mathbf{x})$ are regression models with Gaussian noise.

When the trained network is used to make predictions, the input vector is presented to the gating network and all of the expert networks. The output vector of a ME is the weighted mean (with weighting given by the gating network outputs) of the expert outputs:

$$\mathbf{y}(\mathbf{x}) = \sum_{i=1}^{M} g_i(\mathbf{x}) \phi_i(\mathbf{x}). \tag{3}$$

The mixture of experts network is trained by minimising the error function (2) simultaneously with respect to the weights in all of the expert networks and in

the gating network. The standard choices for gating and expert networks are generalised linear models (GLM) and multi-layer perceptrons (MLP).

3 Hierarchical Generative Topographic Map (HGTM)

The HGTM [2] is a probabilistic model that provides a hierarchical visualisation of data. It arranges a set of GTMs [3] and their corresponding plots in a tree structure $\mathcal{T}$. The GTM models a probability distribution in the high-dimensional data space by means of a low-dimensional (usually 2-dimensional) latent space.

- In GTM, the non-linear transformation, $f : \mathcal{H} \Rightarrow \mathcal{D}$, from the latent space to the data space is defined using a Radial Basis Function (RBF) network with weights $\mathbf{W}$. The density in the latent space is defined as a sum of delta functions centred on nodes $\mathbf{k}_i$. The unconditional probability of a data point $\mathbf{x}$ is given by a mixture

$$p(\mathbf{x} \mid \mathbf{W}, \beta) = \frac{1}{M} \sum_{i=1}^{M} p(\mathbf{x} \mid \mathbf{k}_i, \mathbf{W}, \beta), \tag{4}$$

  where the ith component density is a Gaussian distribution whose mean is the image of $\mathbf{k}_i$ under f with inverse variance β.
- Bayes' theorem is used to invert the transformation f. The posterior probability $R_{i,n}$ (responsibility) that the ith Gaussian generated the point $\mathbf{x}_n$, is given by

$$R_{i,n} = \frac{P(\mathbf{x}_n \mid \mathbf{k}_i, \mathbf{W}, \beta)}{\sum_{j=1}^{C} P(\mathbf{x}_n \mid \mathbf{x}_j, \mathbf{W}, \beta)} \tag{5}$$

  In order to visualise a whole dataset in a single plot, the latent space representation of the point $\mathbf{x}_n$ is taken to be the mean, $\sum_{i=1}^{C} R_{i,n}\mathbf{k}_i$, of the posterior distribution on $\mathcal{H}$ where C is total number of latent space centres.

An example HGTM structure is shown in the Figure 1. In this section we give a general formulation of hierarchical GTM, more details can be found in [2].

The *Root* of the hierarchy is at level 1, i.e. *Level(Root)* $= 1$. Children of a model $\mathcal{N}$ with *Level*$(\mathcal{N}) = \ell$ are at level $\ell + 1$, i.e. *Level*$(\mathcal{M}) = \ell + 1$, for all $\mathcal{M} \in$ *Children*$(\mathcal{N})$. Each model $\mathcal{M}$ in the hierarchy, except for *Root*, has an associated non-negative parent-conditional mixture coefficient, or prior $\pi(\mathcal{M} \mid$ *Parent*$(\mathcal{M}))$. The priors satisfy the consistency condition: $\sum_{\mathcal{M} \in Children(\mathcal{N})} \pi(\mathcal{M} \mid \mathcal{N}) = 1$. Unconditional priors for the models are recursively calculated as: $\pi($*Root*$) = 1$, and for all other models

$$\pi(\mathcal{M}) = \prod_{i=2}^{Level(\mathcal{M})} \pi(Path(\mathcal{M})_i \mid Path(\mathcal{M})_{i-1}), \tag{6}$$

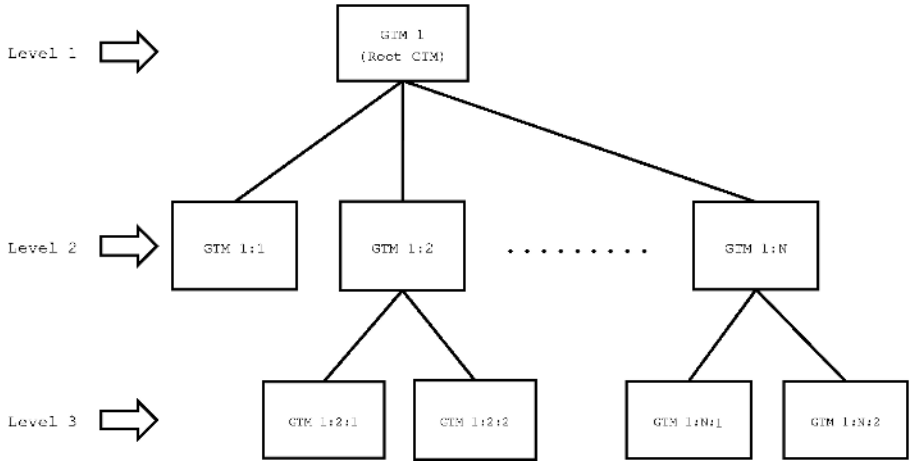


Fig. 1. Example plot structure for HGTM. Each model corresponds to a visualisation.

where $Path(\mathcal{M}) = (Root, ..., \mathcal{M})$ is the N-tuple ($N = Level(\mathcal{M})$) of nodes defining the path in $\mathcal{T}$ from $Root$ to $\mathcal{M}$.

The distribution given by the hierarchical model is a mixture of leaf models of $\mathcal{T}$,

$$P(\mathbf{x} \mid \mathcal{T}) = \sum_{\mathcal{M} \in Leaves(\mathcal{T})} \pi(\mathcal{M}) P(\mathbf{x} \mid \mathcal{M}). \tag{7}$$

Non-leaf models not only play a role in the process of creating the hierarchical model, but in the context of data visualization can be useful for determining the relationship between related subplots in the hierarchy.

The hierarchical GTM is trained using the EM algorithm to maximize its likelihood with respect to the data sample $\varsigma = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$. Training of a hierarchy of GTMs proceeds in a recursive fashion. First, a base ($Root$) GTM is trained and used to visualise the data. Then the user identifies interesting regions on the visualization plot that they would like to model in greater detail. In particular, the user chooses a collection of points, $c_i \in \mathcal{H}$, by clicking on the plot. These points are used to initialise the next level of GTMs. Voronoi compartments [4] are defined in the data space by the mapped points $f_{Root}(c_i) \in \mathcal{D}$, where f_{Root} is the map of the $Root$ GTM. The child GTMs are initialised by local PCA in the corresponding Voronoi compartments. After training the child GTMs and seeing the lower level visualization plots, the user may decide to proceed further and model in greater detail some portions of the lower level plots, etc. At each stage of the construction of an hierarchical GTM, the EM algorithm alternates between the E- and M-steps until convergence is satisfactory (typically after 10-20 iterations).

We can calculate magnification factors using the Jacobian of the GTM map f [5]. Magnification factor plots are used to observe the amount of stretching

in a GTM manifold on different parts of the latent space which helps in outlier detection and cluster separation. Tiňo *et. al.* [6] derived a closed-form formula for directional curvature of the GTM projection manifold. Directional curvature plots allow the user to observe folding in the GTM manifold. Magnification factors and directional curvatures help the user to decide where to place submodels.

We have developed an interactive software tool which allows a user to see the magnification factor and directional curvature plots with the actual HGTM visualisation. The software also provides a *parallel coordinate* facility to let the user explore patterns of a few neighbouring points (determined using Euclidean distance) from the point selected by the user in the latent space. This is useful for understanding different regions of the latent space as the user can observe the corresponding data space patterns. The tool can be used by domain experts to understand and segment vast data.

4 Guided Local Regression Models

The divide-and-conquer approach used in ME discussed in Section 2 can particularly prove useful in modeling diversities in the input-output mapping. One of the most important issues in applying a divide-and-conquer strategy is to find the different regions to divide the input space. Doing it automatically as in ME might not be effective for a complex dataset.

One of the main differences between the mixture of experts and the guided regression models presented in this section, is the way of segmenting the input space. In ME, the gating network learns a decomposition of the input space into different regions with the training of expert models, while in the guided local regression models, we let the domain experts interactively decide the decomposition of the input space using a visualisation algorithm and other visualisation aids, such as magnification factors, directional curvature and parallel coordinates. Thus the segmentation process here is not *automatic* as in ME but it is *guided* by the domain experts.

In this paper we only use a 2-level HGTM tree structure for simplicity, but the results can be extended to an HGTM of any depth. Consider an HGTM tree structure, $\mathcal{T}$, as in Figure 1.

Model responsibilities, $\mathbf{R}$, corresponding to all the models, $\mathcal{M}_i$, $i = 1, \ldots, M$, in the HGTM tree structure, $\mathcal{T}$, are calculated as follows:

$$R_{i,n} = P(\mathcal{M}_i \mid \mathit{Parent}(\mathcal{M}_i), \mathbf{x}_n) = \frac{\pi(\mathcal{M}_i \mid \mathit{Parent}(\mathcal{M}_i))P(\mathbf{x}_n \mid \mathcal{M}_i)}{\sum_{\mathcal{N} \in [\mathcal{M}_i]} \pi(\mathcal{N} \mid \mathit{Parent}(\mathcal{M}_i))P(\mathbf{x}_n \mid \mathcal{N})}, \tag{8}$$

where $[\mathcal{M}_i] = \mathit{Children}(\mathit{Parent}(\mathcal{M}_i))$.

Imposing $P(\mathit{Root} \mid \mathbf{x}_n) = 1$, the unconditional (on parent) model responsibilities are recursively determined by the formula:

$$P(\mathcal{M} \mid \mathbf{x}_n) = P(\mathcal{M} \mid \mathit{Parent}(\mathcal{M}), \mathbf{x}_n)P(\mathit{Parent}(\mathcal{M}) \mid \mathbf{x}_n). \tag{9}$$

The model responsibility matrix, $\mathbf{R}$, has the property

$$\sum_{i=1}^{M} R_{i,n} = 1 \qquad \forall\ n. \tag{10}$$

Equation (10) confirms the *soft* segmentation of the input space we obtain from the HGTM model. It is similar to the segmentation derived from the softmax function in the trained gating network in the ME (eq. 1). The soft segmentation obtained using HGTM is non-linear, so the segmentation regions can have an arbitrary shape. The individual experts can arbitrarily be linear or non-linear regression models. The trained HGTM model is then used to train local regression model, which we name as Guided Mixture of Experts (GME), as specified in Procedure 1. Notice that in step 2, for the training of a local expert, using the model responsibility obtained by the trained HGTM model, we select only those data points which belong to a particular local region. It means that only those data points which lie in a particular local region are used to train the expert responsible for modelling that region. In the work presented here, during the training of a local expert, we do not weight data points with their corresponding model responsibility. One of our future extensions will be to use responsibilities for weighting during the training. We have already implemented a weighted Generalised Linear Model.

Procedure 1 (Training)

1. *Using a previously trained HGTM visualisation model, calculate the model responsibility matrix,* $\mathbf{R}$*, for all the training points for all leaves (eq. 8).*
2. *Train an expert regression model corresponding to each leaf node. Each expert,* $\phi_i(\mathbf{t} \mid \mathbf{x})$*, is trained individually on all the training points,* $\mathbf{x}_n$*, for which* $R_{i,n}$ *is greater than a threshold. Different thresholds can be tried and validated.*
3. *During the training of each expert,* $\phi_i(\mathbf{t} \mid \mathbf{x})$*, possible best architecture is selected through validation on the local points it is responsible for.*

While making predictions for new inputs, in ME, we present inputs to all of the experts and the gating network. The outputs of the experts are weighted by the output of the gating network and summed (eq. 3). In the GME, the inputs are first presented to a trained HGTM visualisation model and responsibilities for each expert are calculated using (eq. 8). Then the output of each expert is weighted by the corresponding responsibility and finally summed as shown in Figure 2. The prediction (testing) procedure, using a trained GME model, is given below:

Procedure 2 (Testing)

1. *Calculate the model responsibility matrix,* $\mathbf{R}$*, for all the testing points using the trained HGTM model stored with the trained GME model.*

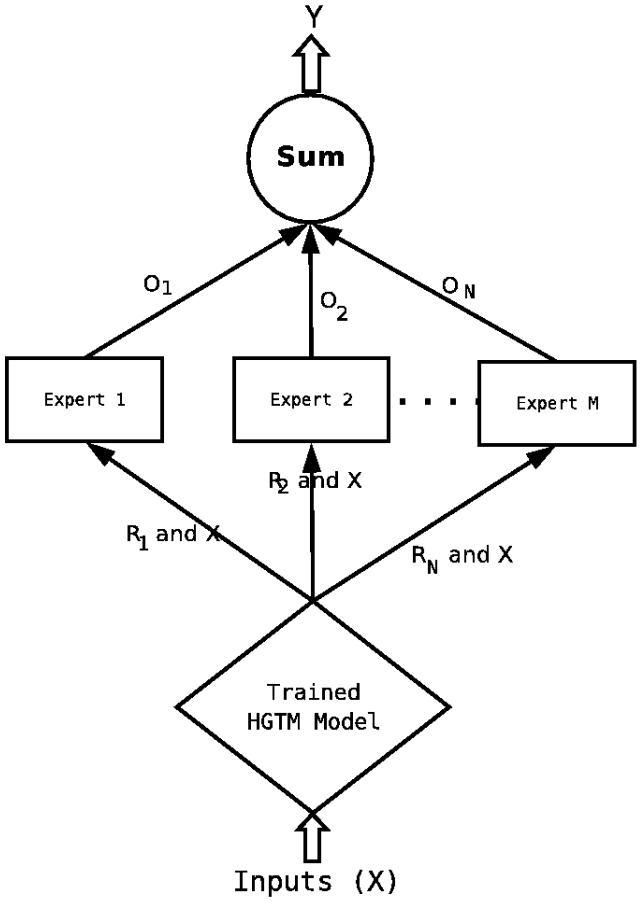


Fig. 2. Architecture of Guided Mixture of Experts (GME)

2. *Each trained expert is presented with all the inputs (see Figure 2). All experts produce the outputs for the input point,* $\mathbf{x}_n$*, which are then weighted by the corresponding model responsibilities and summed to get the final output for that particular input.*

$$\mathbf{y}^n = \sum_{i=1}^{M} R_{i,n} \phi_i(\mathbf{t}^n \mid \mathbf{x}^n), \tag{11}$$

where $\phi_i(\mathbf{t}^n \mid \mathbf{x}^n)$ *is the output from the trained expert* i*.*

5 Results

Two experiments were carried out: one with a synthetic dataset and one with Chemoinformatics data.

5.1 Synthetic Dataset

The data set consisted of around 2900 points, $\mathbf{x} = (x_1, x_2, x_3)^T$ lying on a two-dimensional manifold in the three-dimensional Euclidean space. The manifold is shown in Figure 3 and is described by the equation

$$x_3 = 2 \sum_{c_1, c_2 \in \{-2,2\}} \exp\{-(x_1 - c_1)^2 - (x_2 - c_2)^2\}, \quad (x_1, x_2) \in [-4, 4]^2. \tag{12}$$

To have a different mapping in each "hump", we define the following functions:

$$\begin{aligned}
y &= x_1 - x_2^2 - x_3 \quad \forall x_1, x_2, x_3 \quad 0 < x_1 < 4,\ 0 < x_2 < 4, \text{ and } -2 < x_3 < 0,\\
y &= x_1^2 + x_2 + x_3 \quad \forall x_1, x_2, x_3 \quad -4 < x_1 < 0,\ 0 < x_2 < 4, \text{ and } 0 < x_3 < 2,\\
y &= x_1 + x_2 - x_3^2 \quad \forall x_1, x_2, x_3 \quad -4 < x_1 < 0,\ -4 < x_2 < 0, \text{ and } -2 < x_3 < 0,\\
y &= x_1 - x_2^2 + x_3^2 \quad \forall x_1, x_2, x_3 \quad 0 < x_1 < 4,\ -4 < x_2 < 0, \text{ and } 0 < x_3 < 2.
\end{aligned}$$

From the total dataset of around 2900 data points, 80% of the points were used as the training set and rest were kept aside for testing. 20% of the training set was used for validation to choose the model architecture. Figure 4 shows a trained HGTM output on the testing set of the synthetic dataset.

We trained models with different complexities for MLP, ME and GME. The validation set error was calculated for all the models and, for each architecture, the model with the minimum validation set error was selected. The selected model from a given class was then trained on the whole training set (including the validation set).

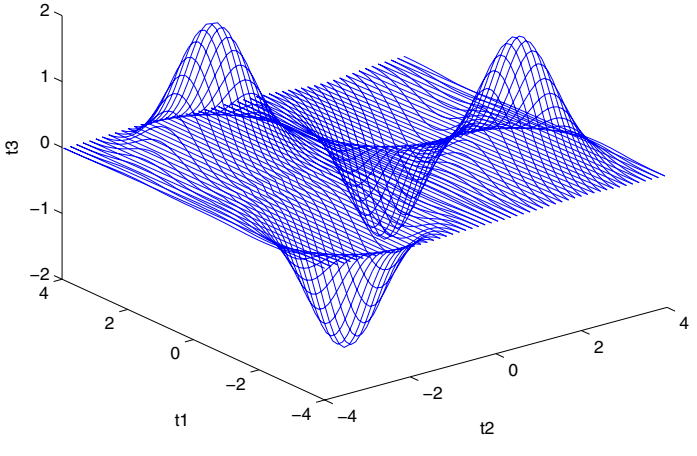


Fig. 3. A two-dimensional manifold in three-dimensional Euclidean space

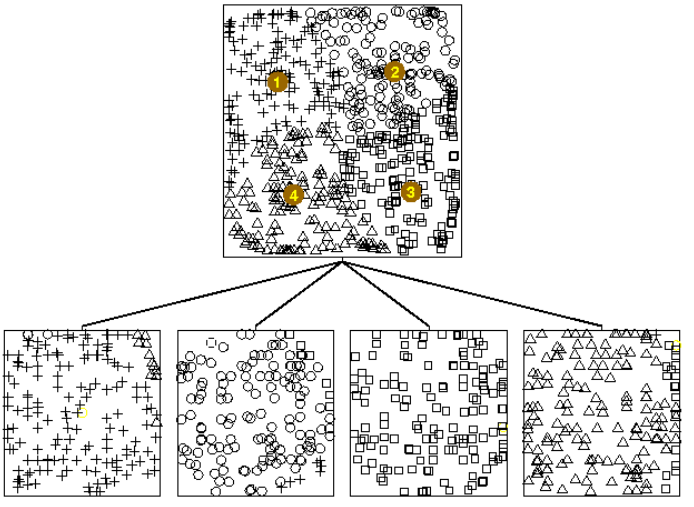


Fig. 4. HGTM visualisation output for the testing set of synthetic data

To analyse the properties of the input-space segmentation obtained from ME and GME, we measure its average entropy [7]. The average entropy was calculated as below:

$$H = -\frac{1}{M}\sum_{m=1}^{M}\frac{1}{N}\sum_{n=1}^{N} P_m(\mathbf{x}_n)\log P_m(\mathbf{x}_n), \tag{13}$$

where $P_m(\mathbf{x}_n)\log P_m(\mathbf{x}_n)$ is defined as 0 if $P_m(\mathbf{x}_n) = 0$. For ME, $P_m(\mathbf{x}_n)$ is the output of the gating network for the mth expert, and input point x_n, while for GME, $P_m(\mathbf{x}_n)$ is the model responsibility, $R_{m,n}$. For the selected architectures, the average entropy values for ME and GME were obtained as 0.0621 and 0.0058 respectively. These values reveal that the ME gives a comparatively soft segmentation with more overlaps, while the GME provides a harder segmentation which separates the input space in to distinct different regions with little overlap which is also easier for the domain experts to interpret.

Table 1 presents the normalised mean squared error (NMSE) [8] we obtained for the training and the test sets. The 4th column in Table 1 displays the t-test significance value compared with the result of the GME. The t-test assesses

Table 1. Regression results for the synthetic dataset

Model	Training NMSE	Testing NMSE	P-value	Architecture
MLP	0.1009	0.0968	7.5816e-48	$N_{hid} = 21$
ME	0.0433	0.0466	0.0021	$N_{experts} = 11$
GME	0.0234	0.0227	-	$N_{experts} = 4$

whether the means of two groups are statistically different from each other [9]. The smaller the value, the more significant the difference between the means. Information about which model architecture was selected, using the validation set, is given in the last column. We note that the GME result is significantly better than the MLP and ME.

5.2 Chemoinformatics Data

The second experiment was carried out on a real life problem in the Chemoinformatics domain where we need to predict the biological activity of chemical compounds, for a particular target, from 11 physicochemical properties of the compounds. The dataset (of around 20700 chemical compounds selected randomly from around 1000000 compounds) was divided equally into training and testing sets. 20% of the training set was kept aside for validation to choose the model architecture. Figure 5 presents the HGTM visualisation output for a subset (random 600 compounds, 300 active and 300 inactive) of testing set.

Table 2. Regression results for biological activity prediction

Model	Training NMSE	Testing NMSE	P-value	Architecture
MLP	0.8439	0.8458	0.0129	$N_{hid} = 25$
ME	0.8370	0.8405	0.0200	$N_{experts} = 12$
GME	0.8104	0.8214	-	$N_{experts} = 7$

The results are presented in Table 2. For the selected architectures, the average entropy values for ME and GME were obtained as 0.1953 and 0.0298 respectively which demonstrates better segmentation obtained by GME. The GME result is better than the two models, though only at a level of 2%.

6 Discussion

Our approach of using visualisation output to develop guided local regression models has given better results than the classical ME. That is in line with our assumption that the segmentation obtained from principled visualisation algorithms, such as HGTM, can be sensibly used for the development of new local regression models.

The advantage of the approach is that the informed segmentation is obtained with the help of domain experts who have some understanding of the data in this way, domain experts are more involved in the model development process. The disadvantage of GME is that, as a 2 stage process, it requires user interactions and thus it takes comparatively more time to develop a new model.

Overall, for the synthetic dataset, the local regression models gave better results than the global regression model. We believe that the principal local

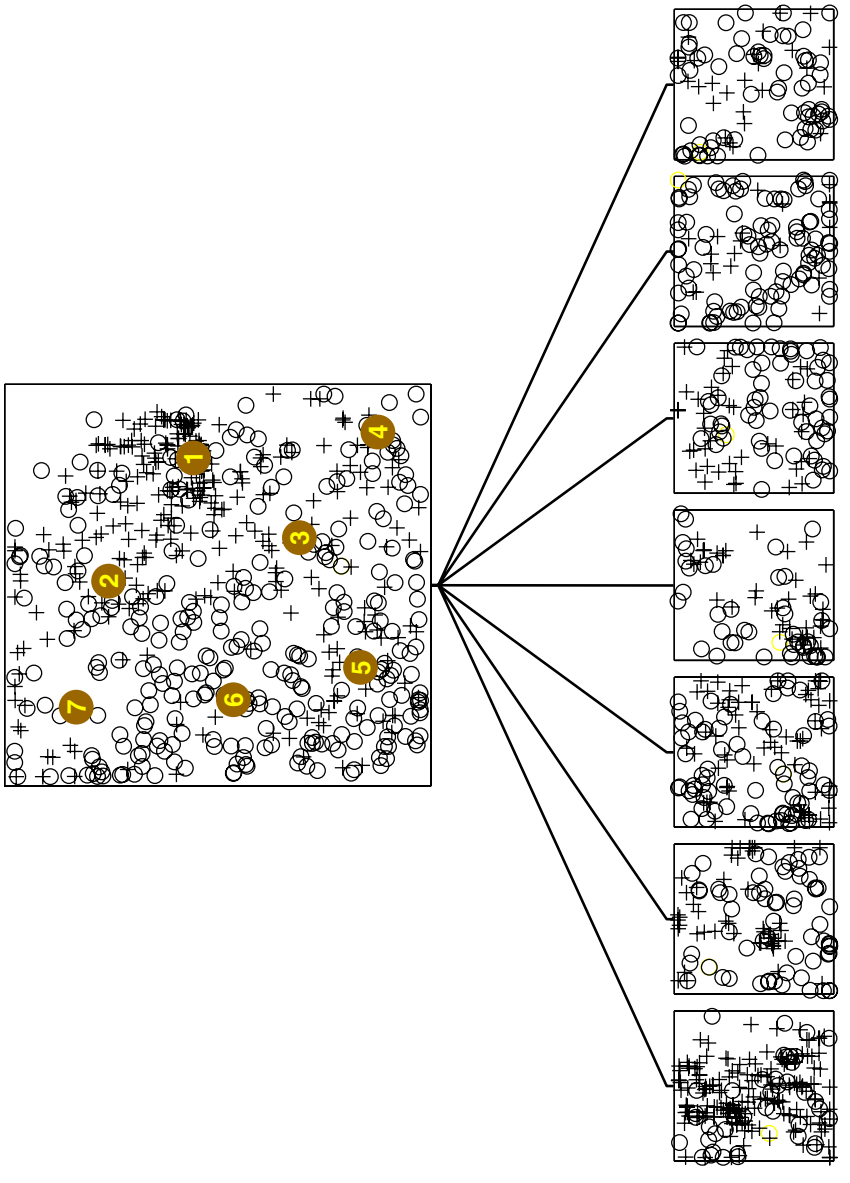


Fig. 5. HGTM output for the subset of the testing set of chemoinformatics data

regression models, such as ME and GME, will perform well for high dimensional diverse datasets generally found in drug discovery and bioinformatics domains.

However, the experiments on chemical compounds gave NMSE of more than 0.8 which is not satisfactory for practical use. The relatively high value of NMSE indicates that the models are close to predicting "in the mean" [8]. After dis-

cussion with screening scientists, it was realised that the descriptors (11 physicochemical properties) used to predict the biological activities do not contain enough information to make a robust prediction. Using structure information of chemical compounds should help in improving regression models performance since the pharmacophore[1] of compounds plays an important role in making a compound active for a target [10].

Acknowledgements

DM is grateful to Pfizer Central Research for their financial support. We thank Bruce S. Williams and Andreas Sewing for useful discussions on the results with the Chemioinformatics data.

References

1. Jacobs, R.A., Jordan, M.I., Nowlan, S.J., Hinton, G.E.: Adaptive mixture of local experts. Neural Computation **3** (1991) 79–87
2. Tiňo, P., Nabney, I.T.: Constructing localized non-linear projection manifolds in a principled way: hierarchical generative topographic mapping. IEEE T. Pattern Analysis and Machine Intelligence **24** (2002) 639–656
3. Bishop, C.M., Svensén, M., Williams, C.K.I.: GTM: The generative topographic mapping. Neural Computation **10** (1998) 215–234
4. Aurenhammer, F.: Voronoi diagrams - survey of a fundamental geometric data structure". ACM Computing Surveys **3** (1991) 345–405
5. Bishop, C.M., Svensén, M., Williams, C.K.I.: Magnification factors for the GTM algorithm. Proceedings IEE Fifth International Conference on Artificial Neural Networks (1997) 64–69
6. Tiňo, P., Nabney, I.T., Sun, Y.: Using directional curvatures to visualize folding patterns of the GTM projection manifolds. Artificial Neural Networks - ICANN (eds) G. Dorffner, H. Bischof and K. Hornik (2001) 421–428
7. Ellis, R.: Entropy, Large Deviations, and Statistical Mechanics. Springer-Verlag, New York (1985)
8. Bishop, C.M.: Neural Networks for Pattern Recognition. 1st edn. Oxford University Press (1995)
9. Weiss, N.: Elementary Statistics. 3rd edn. Addison Wesley (1996)
10. Good, A.C., Krystek, S.R., Mason, J.S.: High-throughput and virtual screening: core lead discovery techologies move towards integration. Drug Discovery Today **5** (2000) S61–S69

[1] A specific arrangement of chemical groups in a compound that are essential for recognition by a target.

Transformations of Gaussian Process Priors

Roderick Murray-Smith[12] and Barak A. Pearlmutter[2]

[1] Department of Computing Science, University of Glasgow, Scotland
[2] Hamilton Institute, NUI Maynooth, Co. Kildare, Ireland
rod@dcs.gla.ac.uk
barak@cs.nuim.ie

Abstract. Gaussian process prior systems generally consist of noisy measurements of samples of the putatively Gaussian process of interest, where the samples serve to constrain the posterior estimate. Here we consider the case where the measurements are instead *noisy weighted sums* of samples. This framework incorporates measurements of derivative information and of filtered versions of the process, thereby allowing GPs to perform sensor fusion and tomography; allows certain group invariances (ie symmetries) to be weakly enforced; and under certain conditions suitable application allows the dataset to be dramatically reduced in size. The method is applied to a sparsely sampled image, where each sample is taken using a broad and non-monotonic point spread function. It is also applied to nonlinear dynamic system identification applications where a nonlinear function is followed by a known linear dynamic system, and where observed data can be a mixture of irregularly sampled higher derivatives of the signal of interest.

1 Introduction

Gaussian process priors are increasingly used as a flexible nonparametric model in a range of application areas (e.g. O'Hagan, 1978; Rasmussen, 1996; Williams, 1998b; Murray-Smith and Sbarbaro, 2002). In (Solak et al., 2003) we used the fact that the derivative of a Gaussian process is itself a Gaussian process to integrate function and derivative observations. This is particularly useful when modeling nonlinear dynamic systems. Here we generalise the results to arbitrary transformations of a Gaussian process, which in discrete form can be summarised by a linear transformation. Like the 'generalised observations' obtained from bounded linear functionals introduced in (Wahba, 1990). We show four major practical advantages this can offer:

1. We can fuse information from multiple sensors, where the (potentially nonlinear) transformation associated with the sensor can be approximated by a linear weighting on discretisation. GP inference can then solve ill-posed inverse problems.
2. We can add 'artificial' data points which introduce prior knowledge by enforcing certain chosen linear constraints, such as symmetry, or higher-order derivative operators.
3. We can choose $n \times N$ linear transformations, where N is the number of points in the original training set, which reduce the computational complexity to $O(n^3) + O(N^2)$. For $n \ll N$ this can lead to a significant improvement in speed. We show that such mappings can be derived from smooths of less refined models.

J. Winkler, N. Lawrence, M. Niranjan (Eds.): Machine Learning Workshop, LNAI 3635, pp. 110–123, 2005.

4. In many applications we can choose a series of linear transformations which compress the training set, as above, and correspond to multi-scale learning.

2 Transformations of Gaussian Process Priors

Consider N observations of inputs X and outputs Y, where we assume the Y are drawn from an N-dimensional normal distribution,

$$Y \sim \mathcal{N}(0, \Sigma),$$

where Σ is the $N \times N$ covariance matrix, the elements of which are functions of inputs X, an $N \times d$ matrix. The covariance function is of the form

$$\mathrm{cov}(x_i, x_j) = v_0 \exp\Big(-\sum_k w_k (x_{i,k} - x_{j,k})^2\Big) + \sigma_y^2,$$

and reflects prior beliefs that the target function is smooth, so penalising high-frequency components. The parameter w_k reflects the length-scale of changes in input dimension k.

We will now assume that instead of observing y's directly, we observe a transformation m of the latent variables y. In the continuous case

$$\begin{aligned} \textit{output} &= \int_\Omega \textit{system} \times \textit{input}\, d\Omega, \\ m(t) &= \int K(t,x)\, y(x)\, dx \end{aligned} \tag{1}$$

which in discrete form is

$$m_k = \sum_{j=1}^{N} K_{kj}\, Y_j, \tag{2}$$

In other words, for the vector of latents Y we observe outputs $M = KY$, where K is known. This could, for example, correspond to an inverse problem such as image restoration, where the observable is the image, the system is the lens, and the scenery is the input. Note that although the discretised form K is a linear transformation, the original kernel $K(t,x)$ could represent a nonlinear mapping.

The vector M is therefore drawn from an n-dimensional normal distribution:

$$M \sim \mathcal{N}(0, K\Sigma K^T + \Sigma_M),$$

where Σ_M is the diagonal matrix of observation variances. If we wish to predict some M_2 given X_1, M_1, K_1, and X_2, K_2 then the conditional mean and variance are

$$\begin{aligned} \mu_{2|1} &= K_2 \Sigma_{12} K_1^T (K_1 \Sigma K_1^T)^{-1} M_1 \\ \Sigma_{2|1} &= \Sigma_2 - K_2 \Sigma_{12} K_1^T (K_1 \Sigma K_1^T)^{-1} K_1 \Sigma_{21} K_2 \end{aligned} \tag{3}$$

By selecting the transformation K_2, associated with the mapping from the latent space y to the outputs at the test points x_2, we can perform inference to any of the variables chosen. If $K_2 = I$, then we are inferring y directly from observations of M_1, and implicitly solving the inverse problem of finding the conditional mean and variance of the latent variable y.

2.1 Learning the Covariance Function Parameters

The log-likelihood, given the training data M_1 is

$$L = -\tfrac{1}{2}\log|K_1\Sigma_1K_1^T| - \tfrac{1}{2}M_1^T(K_1\Sigma_1K_1^T)^{-1}M_1 - \tfrac{1}{2}N_1\log 2\pi.$$

If we wish to maximise the likelihood, we use the derivative with respect to the hyperparameters θ,

$$\begin{aligned}\frac{\partial L}{\partial\theta} &= -\tfrac{1}{2}\operatorname{tr}\Big((K_1\Sigma_1K_1^T)^{-1}\frac{\partial(K_1\Sigma_1K_1^T)}{\partial\theta}\Big)\\ &\quad + \tfrac{1}{2}M_1^T(K_1\Sigma_1K_1^T)^{-1}\frac{\partial(K_1\Sigma_1K_1^T)}{\partial\theta}(K_1\Sigma_1K_1^T)^{-1}M_1 \qquad (4)\\ &= -\tfrac{1}{2}\operatorname{tr}\Big(K_1\frac{\partial\Sigma_1}{\partial\theta}\Big) + \tfrac{1}{2}M_1^T(K_1\Sigma_1K_1^T)^{-1}\frac{\partial(K_1\Sigma_1K_1^T)}{\partial\theta}(K_1\Sigma_1K_1^T)^{-1}M_1\end{aligned}$$

and optimise the hyperparameters using an appropriate routine—we used a conjugate gradient approach, or use a Markov-Chain Monte Carlo algorithm to implement a numerical integration.

The ability to adapt the parameters of the covariance function means that the regularising effect is automatically estimated from the data, reducing the w_k of uninformative input dimensions (see discussion in Williams, 1998b)—this is important in learning in general, but especially interesting for the inverse problem aspects of this paper.

If K is uncertain, then we can take a parametric model $K(t, x; \theta)$, and identify θ, or potentially use a second Gaussian process as a prior for the mapping $K(t, x)$. The covariance function and mean function can be chosen appropriately, depending on knowledge of the mapping from x, y to m.

2.2 Examples of Transformations

The linear transformation K can be used to perform a number of roles:

Filtering the Data. The K can represent filters applied to the latent variables before observation, reflecting sensor characteristics or intervening transformation of the states by other means. As noted above, the sensor characteristics described in $K(t, x)$ could be nonlinear, changing with state x, while retaining a linear transformation K on discretisation. Explicitly building the sensor characteristics into the model will tend to be better conditioned than simply pre-filtering the data with an inverse model.

Enforcing Constraints. We can add new data points which enforce constraints, such that a weighted sum of outputs equals some constant. For example, symmetry can be achieved using matrices of the form

$$K_{\text{even}} = \begin{bmatrix} 1 & & & -1 \\ & 1 & -1 & \\ & 1 & -1 & \end{bmatrix} \qquad K_{\text{odd}} = \begin{bmatrix} 1 & & & 1 \\ & 1 & 1 & \\ & 1 & 1 & \end{bmatrix}$$

for

$$X = \begin{bmatrix} x_1 & x_2 & x_3 & -x_3 & -x_2 & -x_1 \end{bmatrix}^T \qquad M = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T$$

which will produce an even or odd function depending on the matrix chosen. Examples of inference with Gaussian process priors incorporating such symmetry constraints are shown in Figure 1.

An alternative approach to enforce symmetry would be by appropriate design of the covariance function, which would be more appropriate for fully symmetric functions. The use of individual data points as constraints does have potential advantages where prior knowledge of symmetry is restricted to localised regions.

Differentiation. An example of enforcing weighted constraints is to represent derivatives. These can be approximated by finite differences, e.g. for first and second derivatives,

$$K' = \frac{1}{\Delta x}\begin{bmatrix} 1 & -1 & & & \\ & 1 & -1 & & \\ & & \ddots & \ddots & \\ & & & 1 & -1 \end{bmatrix} \qquad K'' = \frac{1}{\Delta x}\begin{bmatrix} 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \end{bmatrix}$$

where Δx indicates the distance between points in x. We can continue in this manner to be able to add arbitrary linear combinations of higher-order derivatives, i.e. differential forms. We can therefore add prior knowledge of combinations of derivatives of any order, by including fictive pairs of data points (x_1, x_2), and their known derivative m, or include information from different sensors which measure different derivatives of y.

3 Fusion of Multiple Transformations of Latent Variables

In the case of an observation vector M composed of a number of vectors $M_i = K_i Y$, we have

$$M = \begin{bmatrix} M_1 \\ M_2 \\ \vdots \\ M_k \end{bmatrix} = \begin{bmatrix} K_1 \\ K_2 \\ \vdots \\ K_k \end{bmatrix} Y = KY.$$

We can now integrate multiple observations which might be a mixture of readings from different physical sensors, artificial data points in the form of constraints on the function, or differential operators applied to the data, to derive a model based on a latent variable y which is compatible with all of them. Such consistent integration of multiple observations, constraints and derivatives is far from trivial, as can be observed in the theoretical and practical problems associated with design and verification of gain scheduled and fuzzy controllers (Leith and Leithead, 1999).

3.1 Relevance for Solving Inverse Problems

If the filters K_i are derived from the physics of the sensing mechanisms, does this approach give us any advantages for solving inverse problems? Standard approaches to inversion of ill-posed problems use regularisation where solution components corresponding to small singular values are filtered out. A common approach would use

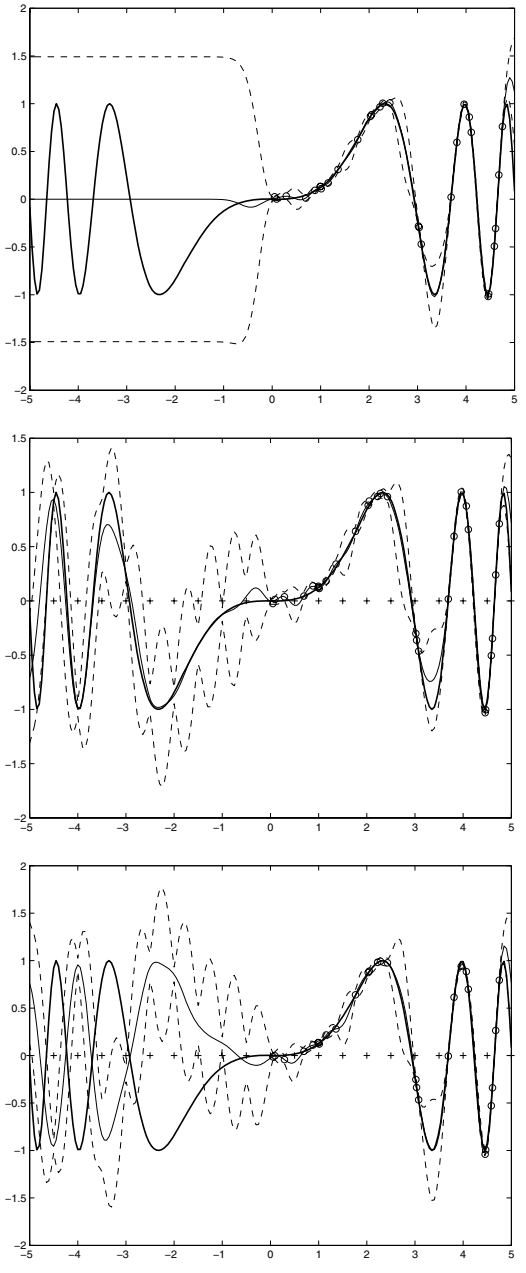


Fig. 1. Artificial data points used to locally enforce symmetry. Top: no symmetry constraints. Centre: odd symmetry constraints. Bottom: even symmetry constraint. Circles are normal observed outputs, crosses are points on x-axis where symmetry constraint has been added. Plots show model mean $\pm$ 2σ as thin solid line and dashed contours. Note that due to the sparse enforcement of symmetry, the error region about the inferred symmetric portion of the curve is looser than on the side with the data.

$Y = K^+M = (K^TK)^{-1}K^TM$, where the inversion would be based around SVD or the Generalised SVD approach, including a filter matrix L would filter the singular values of K. Specific examples of this include Tikhonov regularisation, where a regularisation operator $\Omega(Y)$, is added—minimising $\|KY - M\| + \Omega(Y)$. See Hansen (1997) for a review.

In the GP case presented in this paper, the smoothness constraint is provided by the covariance function. As shown in equation (3), $Y = \Sigma_{12}K^T(K\Sigma K^T)^{-1}M$. Numerically, the inversion of $K\Sigma K^T$ should be better conditioned. Via the covariance function we effectively include estimated or prior knowledge about noise in Y and M, and correlation among elements of Y, which improve the condition number of the matrix $K\Sigma K^T$ and have a regularising effect on the solution.

3.2 Example: Reconstruction of Images from Ganglion Cell Signals

Tipping and Bishop (2002) presented a Gaussian process approach to super-resolution in images, with uniform sampling from a series of low-resolution images. Here we consider a $k \times k$ pixel image measured using noisy sensors, then linearly transformed by a suite $m \ll k^2$ of on-center off-surround receptive fields prior to transmission through a noisy channel. Given the values received, along with a noise model of the channel and knowledge of the receptive fields, we wish to estimate the original image. This reconstruction problem, intended to be reminiscent of interpretation of signals sent through the optic nerve, is shown in Figure 2, with varying levels of sparsity, $k = 41, m = 625$ and 1009 pixels in image (60% of the original pixels) available.

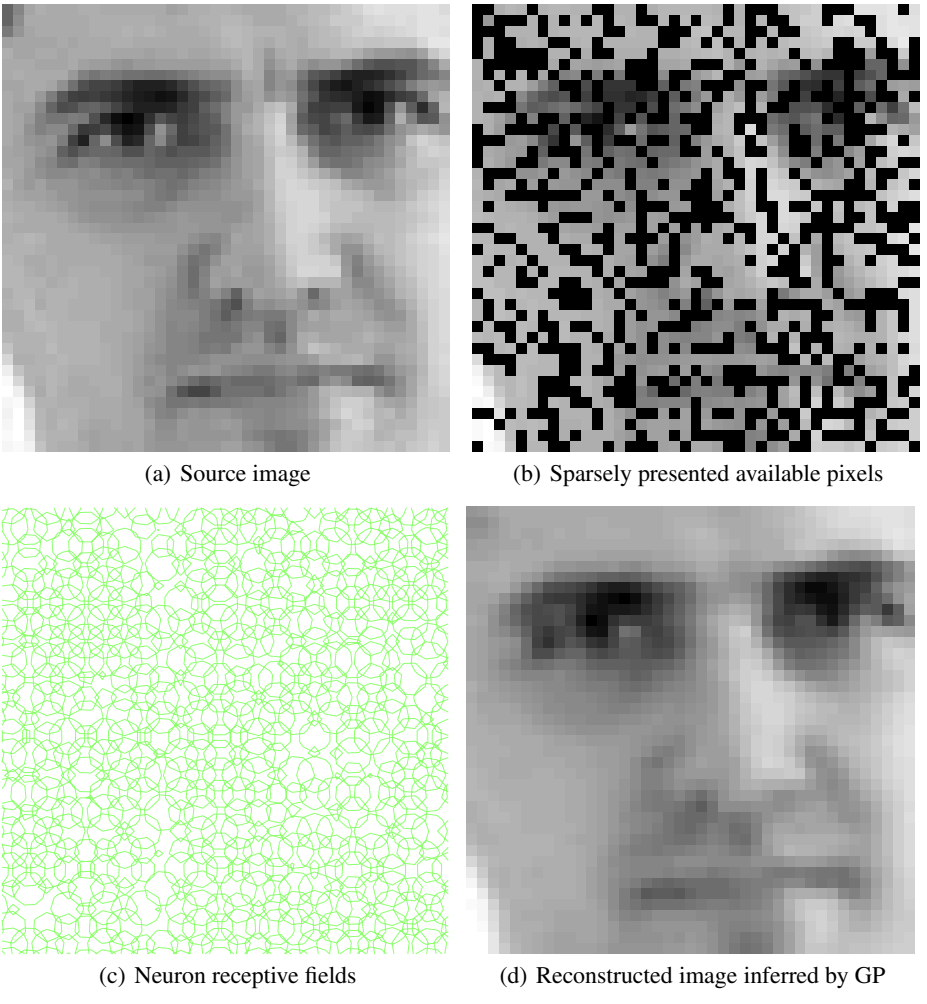
Inspection of natural images such as that shown in Figure 2 suggests that the use of a stationary covariance function is inappropriate. Instead we use a nonstationary one,

$$\mathrm{cov}(x_i, x_j) = v_0 \sin^{-1} \frac{\mathbf{x}_i^T \Sigma \mathbf{x}_j}{\sqrt{(1 + 2\mathbf{x}_i^T \Sigma \mathbf{x}_i)(1 + \mathbf{x}_j^T \Sigma \mathbf{x}_j)}} + \delta_{i,j}\sigma_y^2 \tag{5}$$

as described in Williams (1998a), using Rasmussen's MATLAB implementation.[1] This covariance function corresponds to that of a GP describing a single hidden-layer neural network of sigmoidal neurons with infinite neurons (Williams, 1998a). The Σ is a diagonal matrix with positive entries, weighting each input (and an additional constant one acting as a bias term).

The idea can be extended to colour images. The 'ganglion' cells are now made to be receptive to one colour only, with the allocation of cells to colours done randomly. A further area of interest is that we can use the GP to learn covariances between colours in a natural image, such that if we interpolate between observations we are less likely to generate spurious artifacts. We added an identifier to the inputs, indicating whether the pixel was red (1), green (0) or blue (-1). This was compared to the result of training three independent Gaussian process prior models on the red, green and blue components of the image, independently. Informal inspection of a number of test images showed more frequent colour artefacts in the independent GPs model, than in the dependent one.

[1] `http://www.kyb.tuebingen.mpg.de/bs/people/carl/code/gp/`



(a) Source image (b) Sparsely presented available pixels

(c) Neuron receptive fields (d) Reconstructed image inferred by GP

Fig. 2. Inverse problem solved using a GP. Source image (top left) is sparsely presented, with additive noise (top right) to neurons, and responses on output 'neurons' measured (bottom left). Inference in GP model to training data gives inferred reconstructed image (bottom right).

4 Dynamic Systems Applications

In many applications we will have a learning task which involves identifying a nonlinear subsystem $f(x)$, where we do not have direct access to the outputs y of that subsystem, but to a transformation of them through another dynamic system $m = g(y, z)$, where z is the internal state of $g()$. This transformation may be another subsystem or it could represent the sensor dynamics. We assume that the dynamics of $g(y, z)$ are known and investigate identification of $f(x)$ from observed m and x.

As an example we simulate a system with

$$\begin{aligned} y(t) &= f(x(t)) = 0.3\,x(t)^3 + \sin(5\,x(t)) + \mathcal{N}(0, 0.05) \\ m(t) &= g(y, z) \end{aligned} \tag{6}$$

where there is an unknown initial condition $z(0) = z_0$. The simple discrete-time state-space system $g(y, z)$ has an output matrix C.

$$\begin{aligned} z(t+1) &= Az(t) + By(t) \\ m(t) &= Cz(t) \end{aligned} \tag{7}$$

$$M = C \begin{bmatrix} B & & & & & \\ AB & B & & & & \\ A^2B & AB & B & & & \\ A^3B & A^2B & AB & B & & \\ & & & & \ddots & \\ A^{N-1}B & A^{N-2}B & \cdots & A^2B & AB & B \end{bmatrix} Y + \begin{bmatrix} A \\ A^2 \\ A^3 \\ A^4 \\ \vdots \\ A^N \end{bmatrix} z_0, \tag{8}$$

where $M = \begin{bmatrix} m(1) \cdots m(N) \end{bmatrix}^T$, $Y = \begin{bmatrix} y(1) \cdots y(N) \end{bmatrix}^T$. If we subtract the contribution of the initial condition z_0 from M, we have an effective filter matrix K_d

$$K = C \begin{bmatrix} B & & & & & \\ AB & B & & & & \\ A^2B & AB & B & & & \\ A^3B & A^2B & AB & B & & \\ & & & & \ddots & \\ A^{N-1}B & A^{N-2}B & \cdots & A^2B & AB & B \end{bmatrix} \tag{9}$$

and we can apply the filtered Gaussian process approach in a straightforward manner.

4.1 Observing Multiple Derivatives of Time-Series

In many practical engineering applications observations are made with multiple sensors which measure, e.g. position, velocity and acceleration of state variables of interest. These sensors will often have different noise characteristics and different sampling rates. We would like to be able to combine these variables in a consistent manner. In Solak et al. (2003) we presented methods which perform inference on derivatives with respect to the inputs by analytically differentiating the covariance function. The approach described in this paper allows us to take derivatives with respect to variables not used by the covariance function.

As a simple example, we can infer the distribution of position from observed higher derivatives, and occasional noisy observations of position. Using the filters described in section 2.2,

$$K_v = \frac{1}{\Delta t_v} \begin{bmatrix} -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \end{bmatrix}, K_a = \frac{1}{\Delta t_a} \begin{bmatrix} 1 & -2 & 1 & & \\ & 1 & -2 & 1 & \\ & & \ddots & \ddots & \ddots \\ & & & 1 & -2\ 1 \end{bmatrix}, \tag{10}$$

where Δt_v and Δt_a are the sampling times of the velocity and acceleration signals. Similarly using the transposes K_v^T or K_a^T would allow us to infer a Gaussian process model of acceleration or velocity signals from position observations. This approach also guarantees an internally consistent set of higher derivatives, and could therefore be used as pre-processing technique for system identification and analysis tasks.

This can be combined with other filters associated with the system dynamics as described in equation (9),

$$K = K_d \begin{bmatrix} I_p & K_v & K_a \end{bmatrix} \tag{11}$$

where I_p is the identify matrix of size p for p position observations. General differential operators could be composed of weighted combinations of the basic matrices, $K = w_1 K_v + w_2 K_a$. This lets us use Gaussian processes as an alternative to the Functional Data Analysis methods suggested by Ramsay and Silverman (1997) for inferring higher derivatives of a function without numerical problems.

4.2 Simulation of Dynamic System

Here we generate a time-series of $N = 1000$ points from a second order, discrete-time, linear system, following a nonlinear transformation of the inputs x, as described in equations (6) and (7), with $A = \begin{bmatrix} 1 & 1-\exp(-T_s) \\ 0 & \exp(-T_s) \end{bmatrix}$, $B = \begin{bmatrix} T_s - 1 + \exp(-T_s) \\ 1 - \exp(-T_s) \end{bmatrix}$, $z_0 = 0$, and $T_s = 0.1\,\mathrm{s}$ is the sample time. Furthermore, in the simulation we include a velocity 'sensor', i.e. the observations also include observations of dz/dt. The observations were subsampled to every 20th position observation, and every fourth velocity observation.

In Figure 3 we show the training data inputs x, the unobserved outputs y, and the observed filtered outputs m. The observed outputs are corrupted by white noise $\mathcal{N}(0, 0.001)$.

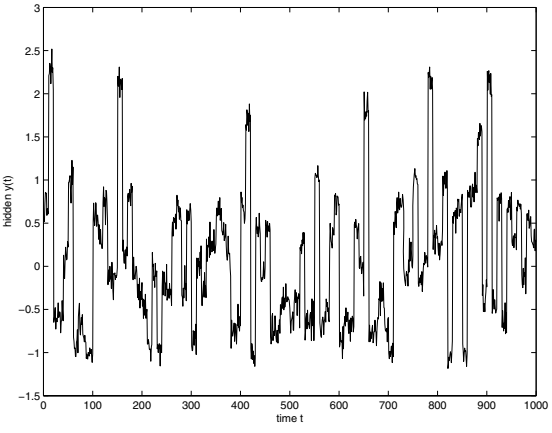
In Figure 4(b) we show the estimate of $f(x)$ over the range of interest, along with the true function, and in Figure 4(a) the estimate of the hidden states y, compared to the actual values. To show the accuracy of the velocity predictions, Figure 5 compares the inferred, observed and true velocities, and Figure 4(a) shows the estimate of the hidden states y, compared to the actual values. To show the accuracy of the velocity predictions, Figure 5 compares the inferred, observed, and true velocities.

5 Discussion

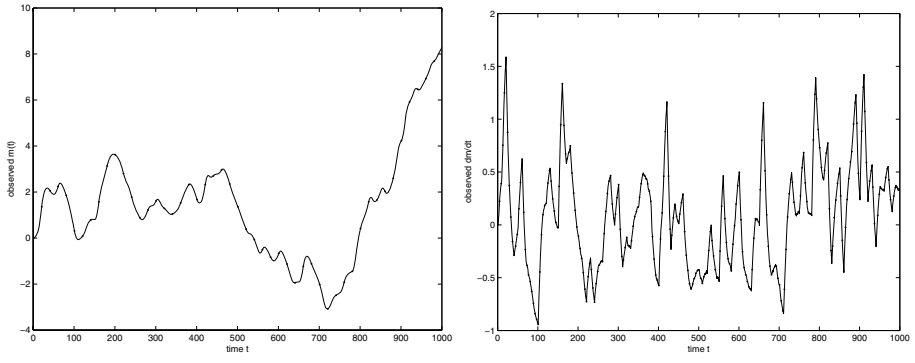
5.1 Learning with Large Data-Sets

A major limiting factor in the acceptance of GP-prior approaches in practice is the computational effort associated with large training sets, as the complexity grows at $O(N^3)$ for a training set with N points. Attempts to overcome this include the use of the Nyström method (Williams and Seeger, 2001), selection mechanisms (Seeger et al., 2003), mixtures of GPs (Shi et al., 2002), and Bayesian committee machine (Tresp, 2000).

A key feature of the filtering approach is that the K_i need not be square matrices. In fact in many applications the filter can represent a significant reduction in the



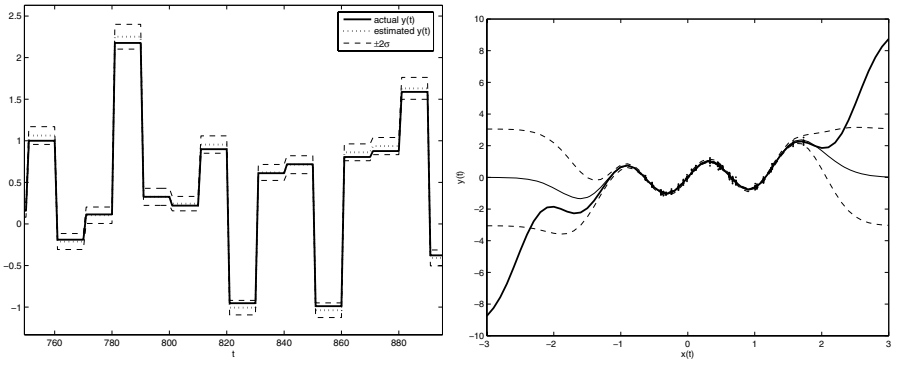
(a) Time-series of the hidden y time-series associated with the training data.



(b) Time-series of observations used by the GP to infer the nonlinear function associated with the hidden y time-series in (a).

Fig. 3. Time-series of observed and hidden states from the simulation used to generate the training data

number of data points, so K will be $n \times N$ where $n \ll N$. Note that in the equations for the inference and likelihood calculations we needed to invert $K\Sigma K^T$, which is the major computational hurdle for this method, scaling as $O(N^3)$. For nonsquare K we now need only invert an $n \times n$ matrix, as opposed to an $N \times N$. We still need to calculate the covariance values of Σ for all N points, but this is $O(N^2)$. To further increase the efficiency of the method we can eliminate points from the calculation of the covariance matrix Σ which correspond to a column of entries in $K_{i,j}\, y_j$ which are below some threshold ϵ. In such cases, the original observation y_j associated with this column has little impact on the model's predictions at the chosen test points. In (Shi et al., 2005) we used a Karhunen-Loeve expansion to choose a subset of points. In (Solak et al., 2003) we compressed large amounts of observed data close to equilibria into local linear models, and used these very few parameters as estimated derivative observations.



(a) Comparison of hidden and inferred y as time-series

(b) Learned & true nonlinearities. (Thick line is true function, thin line mean prediction, and dashed lines ± 2 std. deviation contours.)

Fig. 4. Learning the unknown nonlinearity. The identified nonlinear function, compared to the ideal function $y(t) = 0.3\,x(t)^3 + \sin(5\,x(t))$, with $\pm 2\sigma$ contours. The actual value of y at training data is indicated by the points plotted in Figure 4(b), but the GP did not have access to this information

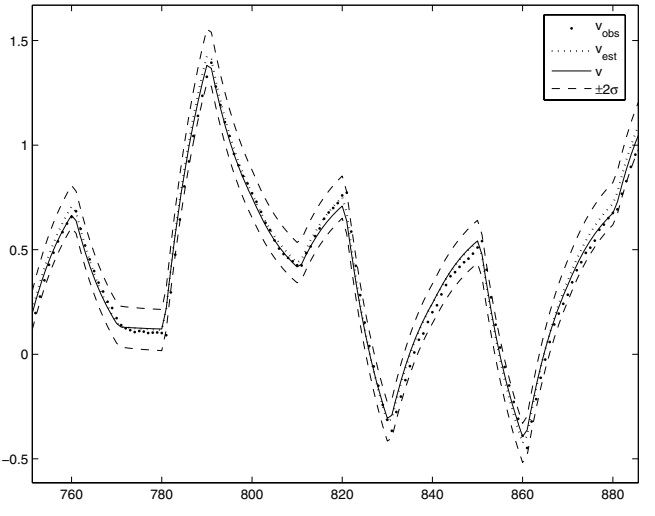


Fig. 5. The GP-inferred, observed (including effects of additive noise on the y's, and true (noise-free) velocities for a segment of the training time-series

To summarize, when $n \ll N$ this method results in substantially decreased computational burden because

$$\textit{complexity} = \overbrace{O(n^3)}^{\text{invert}} + \overbrace{O(N^2)}^{\text{covar}} \ll \overbrace{O(N^3)}^{\text{naive}}$$

Bias and Variance. In most machine learning approaches to data-dependent control of model complexity, the size of the model is reduced in order to trade off model bias against model variance. For instance, it is common to fit a large neural network to a large corpus of data, and then prune the network gradually removing weights. As weights are removed, the computational burden is reduced; the variance decreases; and the bias increases. This results in a tradeoff between bias and variance, meaning there is a point of best generalisation after some amount of pruning. With less than this amount of pruning, it is possible to both improve generalisation performance and reduce computational burden by further pruning. Below it, there is a tradeoff between generalisation and efficiency.

Here we have a superficially similar situation, in which exemplars can be removed from the data. As they are gradually removed, the computational burden is reduced, but the bias and variance *both* increase. As a result, the tradeoff is solely between computational burden and generalisation performance, with no need to find the point of best generalisation. Furthermore, since we would expect the exemplars removed first to contribute least to generalisation, we might expect dramatic computational savings at a very modest cost in generalisation during the initial phases.

Reusing *Effective Kernels* from Earlier Models. A practical approach for finding a suitable K, with $n \ll N$, is the use of prior knowledge of the problem to determine appropriate filters. An alternative is to base the filter on existing approximate models, which might be less computationally expensive to estimate. We now generalise this idea to a broader class of model—we take an existing nonlinear representation of the input-output relationship from any linear-in-the-parameters nonlinear empirical model, and at any input point of interest, we can calculate the effective kernel of the model. For any basis function model, such as an RBF network, spline model etc, with basis functions $\phi_i(x)$, and weighting parameters θ_i, the estimated output $\hat{y}^*$ for a test input x^* is $\hat{y}^* = \sum_i \phi_i(x^*)\hat{\theta}_i = \Phi(x^*)\hat{\theta}$, where the parameters are identified using standard approaches, e.g. $\hat{\theta} = \Phi(X)^+ Y$. We can now reinterpret the basis function model as smoothing the training outputs, $\hat{y}^* = \Phi(x^*)\Phi(X)^+ Y$, where the vector $k_* = \Phi(x^*)\Phi(X)^+$ is the *effective kernel*, a weighting of the y's in the training set for the model prediction at test point x^*. Repeating this at all points in the training set gives us the smoothing matrix $S = \Phi(X)\Phi(X)^+$. The larger the value of the entries $S_{i,j}$, the more leverage observation y_j has on the prediction of $\hat{y}_i$. We can use this effective kernel as a way of generating rows of the linear transformation matrix K to create new, filtered training data. The filter will be well-suited to the specific modelling task, and its application creates 'high-value' data points from weighted combinations of the observed data. This might provide a useful way to bring Gaussian Processes to the attention of a broader user base, as modellers who currently use a linear-in-the-parameters model could 'wrap' a Gaussian process around their current best model, getting potentially better lower model bias, and the benefit of conditional variance estimates.

5.2 Differential Forms and Ease of Implementation

The above derivations assume that measurements correspond to the outputs of known linear filters applied to the underlying function, and that these linear filters are sim-

ply weighted sums. The derivation is reasonably straightforward to extend to a broader class of linear filters corresponding to weighted sums of not just the function itself but also its derivatives, including potentially higher-order derivatives. This would combine the derivation above with that of Solak et al. (2003) yielding a theory that treats both as special cases. However in practice, as we have seen above, it is simple to approximate derivatives simple weighted sums of nearby points, which fits naturally into the framework here. The main advantage of this arguably less elegant approach to handling derivatives is three-fold. First, it avoids the cumbersome notational complexity required for referring to derivatives as well as the corresponding additional matrices. Second, it allows standard GP tools to be easily pressed into service for data of this type, making the approach more accessible to the casual practitioner. And thirdly, it makes it easy for the exploratory practitioner to constrain their models *locally* by introducing virtual data points representing some constraint.

6 Conclusions

We have demonstrated how transformations of Gaussian process priors can, for known transformations, allow us to use GPs to consistently fuse information from multiple sensors, which is of immediate practical importance in many engineering applications. We also demonstrate the use of GPs to solve ill-posed inverse problems. The amount of noise on both latent variables and observed variables, and the amount of regularisation required in the inversion process are automatically optimised during adaptation of the model covariance hyperparameters. More detailed comparison of these benefits with the algorithms currently used in the inverse-problems community is required.

The incorporation of 'artificial' data points is a novel way to introduce prior knowledge by enforcing certain chosen linear constraints, such as symmetry, or higher-order derivative operators, which is easy to use, and has application in a range of areas. The reduction in the computational complexity to $O(n^3) + O(N^2)$ for GP's may also be significant in broadening the application base of GP inference, and there is great scope for extension of the methods to create interesting multi-scale learning algorithms, and for stepwise optimisation or integration of covariance hyperparameters.

The application of the method to a toy dynamic system indicates the promise this approach has to real-world system identification tasks where a number of different sensors, each with its own dynamics and noise level, need to be integrated.

Acknowledgements

The authors gratefully acknowledge the support of the *Multi-Agent Control* Research Training Network by EC TMR grant HPRN-CT-1999-00107, support from EPSRC grant *Modern statistical approaches to off-equilibrium modelling for nonlinear system control* GR/M76379/01, support from EPSRC grant GR/R15863/01, and Science Foundation Ireland grant 00/PI.1/C067.

References

Hansen, P. C. (1997). *Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion*. SIAM. SIAM Monographs on Mathematical Modeling and Computation 4.

Leith, D. and Leithead, W. (1999). Analytic framework for blended multiple model systems using linear local models. *International Journal of Control*, 72(7/8):605–619.

Murray-Smith, R. and Sbarbaro, D. (2002). Nonlinear adaptive control using non-parametric Gaussian process prior models. In *15th IFAC World Congress on Automatic Control, Barcelona*.

O'Hagan, A. (1978). On curve fitting and optimal design for regression (with discussion). *Journal of the Royal Statistical Society B*, 40:1–42.

Ramsay, J. O. and Silverman, B. W. (1997). *Functional Data Analysis*. Springer-Verlag.

Rasmussen, C. E. (1996). *Evaluation of Gaussian Processes and other Methods for Non-Linear Regression*. PhD thesis, Graduate department of Computer Science, University of Toronto.

Seeger, M., Williams, C. K. I., and Lawrence, N. D. (2003). Fast forward selection to speed up sparse Gaussian process regression. In Bishop, C. M. and Frey, B. J., editors, *Proceedings of the Ninth International Workshop on AI and Statistics*.

Shi, J., Murray-Smith, R., Titterington, D., and Pearlmutter, B. (2005). Learning with large data sets using filtered gaussian process priors. In Murray-Smith, R. and Shorten, R., editors, *Proceedings of the Hamilton Summer School on Switching and Learning in Feedback systems*, volume 3355 of *Lecture Notes in Computing Science*, pages 128–139. Springer-Verlag.

Shi, J. Q., Murray-Smith, R., and Titterington, D. M. (2002). Hierarchical Gaussian process mixtures for regression. Technical Report TR-2002-107, University of Glasgow, Scotland, UK.

Solak, E., Murray-Smith, R., Leithead, W. E., Leith, D. J., and Rasmussen, C. E. (2003). Derivative observations in Gaussian process models of dynamic systems. In S. Becker, S. T. and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*, pages 1033–1040. MIT Press, Cambridge, MA.

Tipping, M. and Bishop, C. M. (2002). Bayesian image super-resolution. In Becker, S., Thrun, S., and Obermeyer, K., editors, *Neural Information Processing Systems*, volume 12, pages 1303–1310.

Tresp, V. (2000). A Bayesian committee machine. *Neural Computation*, 12:2719–2741.

Wahba, G. (1990). Spline models for observation data. In *Regional Conference Series in Applied Mathematics*, Philadelphia, PA. SIAM.

Williams, C. K. I. (1998a). Computation with infinite neural networks. *Neural Computation*, 10:1203–1216.

Williams, C. K. I. (1998b). Prediction with Gaussian processes: From linear regression to linear prediction and beyond. In Jordan, M. I., editor, *Learning and Inference in Graphical Models*, pages 599–621. Kluwer.

Williams, C. K. I. and Seeger, M. (2001). Using the Nyström method to speed up kernel machines. In T. K. Leen, T. G. Diettrich, V. T., editor, *Advances in Neural Information Processing Systems 13, MIT Press*.

Kernel Based Learning Methods: Regularization Networks and RBF Networks

Petra Kudová and Roman Neruda

Institute of Computer Science,
Academy of Sciences of the Czech Republic,
18207 Prague, Czech Republic
{petra, roman}@cs.cas.cz

Abstract. We discuss two kernel based learning methods, namely the Regularization Networks (RN) and the Radial Basis Function (RBF) Networks. The RNs are derived from the regularization theory, they had been studied thoroughly from a function approximation point of view, and they posses a sound theoretical background. The RBF networks represent a model of artificial neural networks with both neuro-physiological and mathematical motivation. In addition they may be treated as a generalized form of Regularization Networks. We demonstrate the performance of both approaches on experiments, including both benchmark and real-life learning tasks. We claim that RN and RBF networks are comparable in terms of generalization error, but they differ with respect to their model complexity. The RN approach usually leads to solutions with higher number of base units, thus, the RBF networks can be used as a 'cheaper' alternative. This allows to utilize the RBF networks in modeling tasks with large amounts of data, such as time series prediction or semantic web classification.

1 Introduction

The problem of *learning from examples* (also called *supervised learning*) is a subject of great interest. Systems with the ability to autonomously learn a given task, would be very useful in many real life applications, namely those involving prediction, classification, control, etc.

The problem can be formulated as follows. We are given a set of examples (pairs) $\{(\boldsymbol{x}_i, y_i) \in R^d \times R\}_{i=1}^N$ that was obtained by random sampling of some real function f, generally in the presence of noise. To this set we refer as *a training set*. Our goal is to recover the function f from data, or find the best estimate of it. It is not necessary that the function exactly interpolates all the given data points, but we need a function with good *generalization*, that is a function that gives relevant outputs also for the data not included in the training set.

The problem of learning from examples is studied as a function approximation problem. Given the data set, we are looking for the function that approximate the unknown function f. It can be done by the *Empirical Risk Minimization*, i.e. minimizing the functional $H[f] = \frac{1}{N}\sum_{i=1}^N (f(\boldsymbol{x}_i) - y_i)^2$ over a chosen *hypothesis space*. In section 2 we will study the problem of learning from examples as a function approximation problem and show how a regularization network (RN) is derived from regularization theory. In section 3 we will discuss a learning algorithm for RNs.

J. Winkler, N. Lawrence, M. Niranjan (Eds.): Machine Learning Workshop, LNAI 3635, pp. 124–136, 2005.

Alternatively, the problem of learning from examples can be also handled by artificial neural networks (ANNs). There is a good supply of network architectures and corresponding supervised learning algorithms (see [1] for example). In this case the model — a particular type of neural network — is chosen in advance, and its parameters are tuned during learning so as to fit the given data. In terms of function approximation, the Empirical Risk is minimized over the hypothesis space defined by the chosen type of ANN, i.e. the space of functions representable by this type of ANN. In section 4 we will describe one type of neural networks — an RBF network — which is closely related to RN.

In section 5 the performances of RBF network and RN are compared on experiments, including both benchmark and real learning tasks.

2 Regularization Networks

In this section we will study the problem of learning from examples by means of regularization theory. We are given a set of examples $\{(\boldsymbol{x}_i, y_i) \in R^d \times R\}_{i=1}^N$ obtained by random sampling of some real function f, and we would like to find this function.

Since this problem is ill-posed, we have to consider some a priori knowledge about the function. It is usually assumed that the function is *smooth*, in the sense that two similar inputs corresponds to two similar outputs and the function does not oscillate much. This is the main idea of the regularization theory, where the solution is found by minimizing the functional (1) containing both the data term and the smoothness information.

$$H[f] = \frac{1}{N}\sum_{i=1}^{N}(f(\boldsymbol{x}_i) - y_i)^2 + \gamma\Phi[f], \tag{1}$$

where Φ is called a *stabilizer* and $\gamma > 0$ is *the regularization parameter* controlling the trade off between the closeness to data and the smoothness of the solution. The regularization scheme (1) was first introduced by Tikhonov [2] and therefore it is often called a Tikhonov regularization.

Poggio, Girosi and Jones in [3] proposed a form of a smoothness functional based on Fourier transform:

$$\Phi[f] = \int_{R^d} d\boldsymbol{s}\frac{|\tilde{f}(\boldsymbol{s})|^2}{\tilde{G}(\boldsymbol{s})}, \tag{2}$$

where $\tilde{f}$ indicates the Fourier transform of f , $\tilde{G}$ is some positive function that goes to zero for $||s|| \to \infty$ (i.e. $1/\tilde{G}$ is a high-pass filter). The stabilizer (2) measures the energy in the high frequency and so penalizes the functions with high oscillations.

It was shown that for a wide class of stabilizers in form of (2) the solution has a form of feed-forward neural network with one hidden layer, called *Regularization Network*, and that different types of stabilizers lead to different types of Regularization Networks [3,4].

Poggio and Smale in [4] studied the Regularization Networks derived using a Reproducing Kernel Hilbert Space (RKHS) as the hypothesis space. Let $\mathcal{H}_K$ be an RKHS defined by a symmetric, positive-definite kernel function $K_{\boldsymbol{x}}(\boldsymbol{x}') = K(\boldsymbol{x}, \boldsymbol{x}')$. Then if we define the stabilizer by means of norm in $\mathcal{H}_K$ and minimize the functional

$$s \min_{f \in \mathcal{H}_K} H[f], \text{where } H[f] = \frac{1}{N} \sum_{i=1}^{N} (y_i - f(\boldsymbol{x}_i))^2 + \gamma ||f||_K^2 \tag{3}$$

over the hypothesis space $\mathcal{H}_K$, the solution of minimization (3) is unique and has the form

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} c_i K_{\boldsymbol{x}_i}(\boldsymbol{x}), \qquad (N\gamma I + K)\boldsymbol{c} = \boldsymbol{y}, \tag{4}$$

where I is the identity matrix, K is the matrix $K_{i,j} = K(\boldsymbol{x_i}, \boldsymbol{x_j})$, and $\boldsymbol{y} = (y_1, \ldots, y_N)$. Girosi in [5] showed that for positive definite functions of the form $K(\boldsymbol{x} - \boldsymbol{y})$ (such as Gaussian function) the norm in RKHS defined by K is equivalent to stabilizer (2):

$$||f||_K^2 = \int_{R^d} d\boldsymbol{s} \frac{|\tilde{f}(\boldsymbol{s})|^2}{\tilde{G}(\boldsymbol{s})}. \tag{5}$$

This means, that using such a norm as a regularization term (as in (3)) indeed penalizes highly oscillating functions f.

3 Learning with Regularization Networks

The form of Regularization Network in (4) leads to the learning algorithm (3.1). The power of this algorithm is in its simplicity and effectiveness, the drawback is that the size of the model (a number of kernel functions) is equal to the size of the training set, and so the tasks with large data set lead to solutions of implausible size. It is also supposed that the type of kernel function and the regularization parameter γ are chosen in advance.

Input: Data set $\{\boldsymbol{x}_i, y_i\}_{i=1}^N \subseteq X \times Y$ **Output:** Function f.

1. Choose a symmetric, positive-definite function $K_{\boldsymbol{x}}(\boldsymbol{x}')$, continuous on $X \times X$.
2. Create $f : X \to Y$ as $f(\boldsymbol{x}) = \sum_{i=1}^{N} c_i K_{\boldsymbol{x}_i}(\boldsymbol{x})$ and compute $\boldsymbol{c} = (c_1, \ldots, c_N)$ by solving

$$(N\gamma I + K)\boldsymbol{c} = \boldsymbol{y}, \tag{6}$$

where I is the identity matrix, K is the matrix $K_{i,j} = K(\boldsymbol{x_i}, \boldsymbol{x_j})$, and $\boldsymbol{y} = (y_1, \ldots, y_N)$, $\gamma > 0$ is real number.

Algorithm 3.1

Let us discuss more closely the case of Gaussian kernel $K(\boldsymbol{x}, \boldsymbol{x}') = e^{-\left(\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|}{b}\right)^2}$, which is widely used. Once the width b and the regularization parameter γ are fixed, the algorithm reduces to the problem of solving linear system of equations (6).

Since the system has N variables, N equations, K is positive-definite and $(N\gamma I + K)$ is strictly positive, it is well-posed, i.e. a unique solution exists. We would also like it to be well-conditioned, i.e. insensitive to small perturbations of the data. In other words, we would like the condition number of the matrix $(N\gamma I + K)$ to be small (see [6] to learn more about ill-posed problems). In our case, the condition number can be make small by setting $N\gamma$ large (see Fig. 1 for example on a particular data set). Unfortunately, the γ parameter cannot be chosen arbitrarily, because with large γ we lose the close fit to the data.

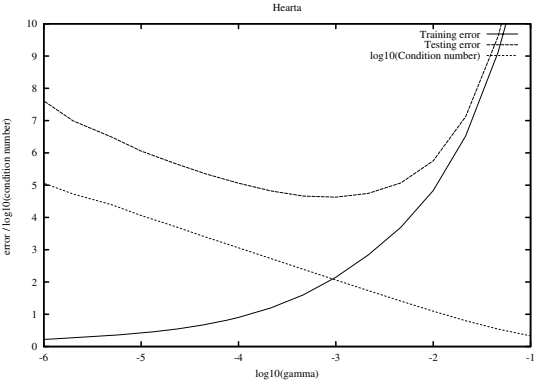


Fig. 1. Dependency of errors (on training and testing data sets) and the condition number of the linear system 6 on the regularization parameter γ

Fig. 1 shows an example of relation between error on training and testing data sets (by testing data we understand a part of data that were not used during learning, so it is an estimate of a real performance of the network), condition number and parameter γ. One can see that for the low values of γ the error on the training set is low, but the performance on the testing set is poor, i.e. the network have poor generalization ability. Nevertheless, if γ is set to be too large, the data term in (1) is suppressed and the result doesn't fit the data neither on training nor testing set properly.

The parameter b determines the width of the Gaussians, and should reflect the density of data points. Suppose that the distances between the data points are high, or the widths are small, then the matrix K has 1s on diagonal and small numbers everywhere else, and therefore it is well-conditioned. But if the widths are too small the matrix goes to identity and contains almost no information. On the other hand, if the widths are too large, all elements of the matrix K are close to 1 and its condition number tends to be high.

The real performance of the algorithm depends significantly on the choice of parameters γ and b (see Fig 2). Optimal choice of these parameters in turn depends on a particular data set.

We estimate both parameters by adaptive grid search and k-fold cross-validation. Adaptive grid search starts with a coarse grid of pairs (γ, b) defined by user and for each pair computes the cross-validation error. Then finer grid is evaluated only in the smaller region containing the pair with the lowest cross-validation error. The process is

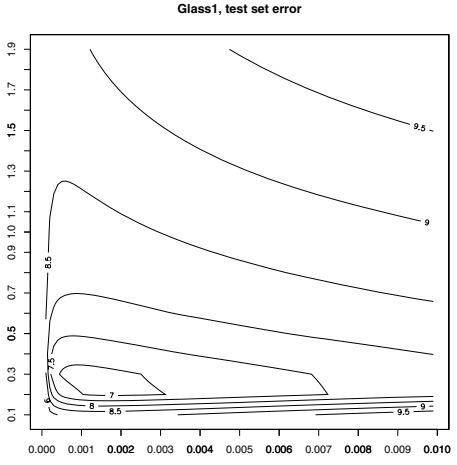


Fig. 2. Dependency of error on testing data set on the choice of regularization parameter γ and width b

repeated until the cross-validation error stops decreasing. Then the parameters with the lowest cross-validation error are picked up and used for evaluation of the algorithm on the whole training set.

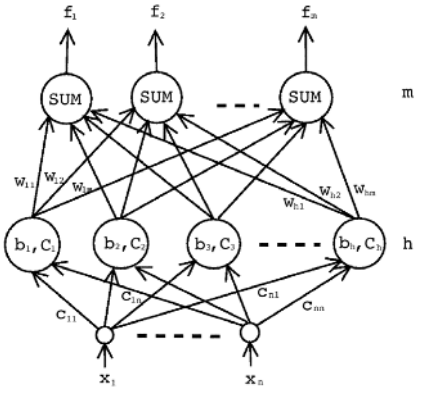
4 RBF Neural Networks

An RBF neural network represents a relatively new neural network architecture. In contrast with the more classical models (such as multilayer perceptron) the RBF network contains local units, which was motivated by the presence of many local response units in human brain. Other motivation came from numerical mathematics, radial basis functions were first introduced as a solution of real multivariate interpolation problems [7].

An RBF network is a feed-forward neural network with one hidden layer of RBF units and a linear output layer (see Fig. 3). By an RBF unit we mean a neuron with n real inputs and one real output, realizing a radial basis function (7), such as Gaussian. Instead of the most commonly used Euclidean norm we use the *weighted norm* $\|\cdot\|_C$, where $\|\boldsymbol{x}\|_C^2 = (C\boldsymbol{x})^T(C\boldsymbol{x}) = \boldsymbol{x}^T C^T C\boldsymbol{x}$.

From the regularization framework point of view, RBF networks belong to the family of generalized regularization networks. Generalized regularization networks are RN with lower number of kernels than data points and also it is not necessary that the kernels are uniform (thus, for example the network with Gaussian kernels may use kernels with different widths).

The goal of RBF network learning is to find the parameters (i.e. centers $\boldsymbol{c}$, widths b, norm matrices C and weights w) so as the network function approximates the function given by the training set $\{(\boldsymbol{x}_i, \boldsymbol{y}_i) \in R^n \times R^m\}_{i=1}^N$. There is a variety of algorithms for RBF network learning, in our previous work we studied their behavior and possibilities of their combinations [8,9]. The two most significant algorithms,*Three step learning* and *Gradient learning*, are sketched in Algorithm 4.1 and Algorithm 4.2. See [8] for details.



$$y(\boldsymbol{x}) = \varphi\left(\frac{\| \boldsymbol{x} - \boldsymbol{c} \|_C}{b}\right) \quad (7)$$

$$f_s(\boldsymbol{x}) = \sum_{j=1}^{h} w_{js}\varphi\left(\frac{\| \boldsymbol{x} - \boldsymbol{c}_j \|_{C_j}}{b_j}\right) \quad (8)$$

Fig. 3. a) RBF network architecture b) RBF network function

Input: Data set $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^{N}$ **Output:** $\{\boldsymbol{c}_i, b_i, C_i, w_{ij}\}_{i=1..h}^{j=1..m}$

1. Set the centers $\boldsymbol{c}_i$ by a k-means clustering.
2. Set the widths b_i and matrices C_i based on relative position of $\boldsymbol{c}_i$.
3. Set the weights w_{ij} by solving $\Phi W = D$.

$$D_{ij} = \sum_{t=1}^{N} y_{tj} e^{-\left(\frac{\|\boldsymbol{x}_t - \boldsymbol{c}_i\|_{C_i}}{b_i}\right)^2}, \Phi_{qr} = \sum_{t=1}^{N} e^{-\left(\frac{\|\boldsymbol{x}_t - \boldsymbol{c}_q\|_{C_q}}{b_q}\right)^2} e^{-\left(\frac{\|\boldsymbol{x}_t - \boldsymbol{c}_r\|_{C_r}}{b_r}\right)^2}$$

Algorithm 4.1

Input: Data set $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^{N}$ **Output:** $\{\boldsymbol{c}_i, b_i, C_i, w_{ij}\}_{i=1..h}^{j=1..m}$

1. Put the small part of data aside as an evaluation set ES, keep the rest as a training set TS .
2. $\forall j$ $\boldsymbol{c}_j(i) \leftarrow$ random sample from TS_1, $\forall j$ $b_j(i), \Sigma_j^{-1}(i) \leftarrow$ small random value, $i \leftarrow 0$
3. $\forall j, p(i)$ in $\boldsymbol{c}_j(i), b_j(i), \Sigma_j^{-1}(i)$:
   $\Delta p(i) \leftarrow -\epsilon\frac{\delta E_1}{\delta p} + \alpha\Delta p(i-1), \qquad p(i) \leftarrow p(i) + \Delta p(i)$
4. $E_1 \leftarrow \sum_{\boldsymbol{x}\in TS_1}(f(\boldsymbol{x}) - y_i)^2$, $E_2 \leftarrow \sum_{\boldsymbol{x}\in TS_2}(f(\boldsymbol{x}) - y_i)^2$
5. If E_1 and E_2 are decreasing, $i \leftarrow i+1$, go to 3, else STOP. If E_2 started to increase, STOP.

Algorithm 4.2

5 Experiments

The aim of the experimental part of this work was to to assess the relative performance of RN and RBF networks, and to relate the results to the performance of multilayer perceptrons, wherever possible. We have used two different collections of data to perform the comparisons.

The first collection is the Proben1 benchmark database [10] which consists of ten tasks (either classification or approximation) with several hundred data instances and a dozen or several dozens of inputs (features). Detailed description of the Proben1 data is given in Tab. 1. The methodology of the experiments follows the procedure proposed by original author of the database, there are always three different divisions of the data set into training (2/3) and testing (1/3) data (thus, e.g. from the cancer data we obtain cancer1, cancer2 and cancer3 sets of training and testing data).

In all results, a normalized error (9) is reported, thus it can be compared across the tasks. On the other hand, the running times indicated in the following tables should be considered only a rough indicator of algorithms time complexity. They refer to production runs on IBM Blade server with 2GHz Xeon processor and 2GB RAM.

$$E_{ts} = 100\frac{1}{N}\sum_{i=1}^{N} ||\boldsymbol{y}_i - f(\boldsymbol{x}_i)||^2 \qquad \begin{array}{ll} N & \text{number of examples in } \{(\boldsymbol{x}_i, \boldsymbol{y}_i)_{i=1}^N\} \\ f & \text{network output} \end{array} \tag{9}$$

Regularization networks have been trained according to Algorithm 2.1 with cross validation technique (described in section 3) for setting parameters γ and b. RBF networks have been trained by the Gradient algorithm 4.2, the statistics are always computed from 10 repetitions of the runs. A very simple procedure has been applied to determine the best architecture for RBF networks: a few reasonable sizes of hidden layer (10, 15, 20, 30 units) have been tried and the best one selected for comparison. It is possible that a cross-validation might further improve these settings, nevertheless, current results are already competitive.

Table 1. Overview of Proben1 tasks. Number of inputs, number of outputs, number of samples in training and testing sets. Type of task: approximation or classification.

Task name	Inputs	Outputs	Train. set	Test. set	Type
cancer	9	2	525	174	class
card	51	2	518	172	class
flare	24	3	800	266	approx
glass	9	6	161	53	class
heartac	35	1	228	75	approx
hearta	35	1	690	230	approx
heartc	35	2	228	75	class
heart	35	2	690	230	class
horse	58	3	273	91	class
soybean	82	19	513	170	class

Table 2. Overview of results obtained by Regularization Network. Error on the training set E_{train}, error on the testing set E_{test}, winning regularization parameter γ, winning width b and time needed for the computation.

Task	E_{train}	E_{test}	γ	b	time
cancer1	2.29	1.76	0.2690×10^{-3}	1.63	4:5:49
cancer2	1.82	3.01	0.2642×10^{-3}	1.46	3:30:13
cancer3	2.12	2.80	0.4958×10^{-3}	1.58	4:22:27
card1	8.80	10.00	1.5963×10^{-3}	4.46	3:36:37
card2	7.63	12.53	1.2864×10^{-3}	4.31	3:8:30
card3	6.58	12.32	0.3078×10^{-3}	4.43	4:10:19
diabetes1	13.94	16.04	1.4590×10^{-3}	1.00	5:29:3
diabetes2	13.85	16.81	1.9810×10^{-3}	0.97	5:24:10
diabetes3	13.75	15.93	0.2943×10^{-3}	1.42	4:42:47
flare1	0.36	0.54	3.6517×10^{-3}	5.70	6:19:53
flare2	0.43	0.27	3.6517×10^{-3}	4.07	7:26:6
flare3	0.41	0.34	2.5483×10^{-3}	4.85	9:2:17
glass1	3.26	6.95	2.4472×10^{-3}	0.30	0:31:18
glass2	4.26	7.91	2.1480×10^{-3}	0.51	0:24:30
glass3	4.06	7.33	2.3607×10^{-3}	0.42	0:26:42
heartac1	4.19	2.78	1.6144×10^{-3}	6.51	1:12:13
heartac2	3.47	3.86	0.8467×10^{-3}	6.00	0:56:2
heartac3	3.32	5.01	1.0413×10^{-3}	6.50	0:55:14
hearta1	3.49	4.40	0.2618×10^{-3}	5.74	7:30:12
hearta2	3.59	4.05	0.2996×10^{-3}	5.72	8:43:32
hearta3	3.47	4.43	0.3398×10^{-3}	5.48	6:11:4
heartc1	9.90	16.02	1.9832×10^{-3}	6.51	1:31:35
heartc2	12.48	6.10	1.1665×10^{-3}	6.51	1:29:34
heartc3	8.88	12.66	1.9810×10^{-3}	3.37	0:47:22
heart1	9.57	13.65	1.5679×10^{-3}	2.89	6:37:13
heart2	9.37	13.80	1.3824×10^{-3}	3.09	7:30:9
heart3	9.27	15.99	0.9647×10^{-3}	3.90	7:29:45
horse1	7.55	11.90	3.7855×10^{-3}	3.40	1:10:59
horse2	7.84	15.18	3.7855×10^{-3}	3.87	1:6:2
horse3	4.81	13.58	2.4144×10^{-3}	2.94	1:27:51
soybean1	0.12	0.66	0.1075×10^{-3}	3.04	3:18:12
soybean2	0.23	0.49	0.1433×10^{-3}	3.60	3:17:22
soybean3	0.24	0.58	0.1334×10^{-3}	3.88	2:4:27

Results of the experiments for RN and RBF networks are shown in Tables 2 and 3 and Figure 4. One can see that while the training error and testing error values are quite comparable, the running times of the algorithms differ significantly. While typically the RN needs hours to achieve a particular error (on the above mentioned hardware), the RBF network needs about 5–10 times less. Finally, Table 4 compares our results with the ones obtained in [10] for multilayer perceptron networks.

The second batch of data consists of a time series of daily rainfall and runoff values measured on a small Czech river Ploučnice basin. There is 4 years worth of daily data divided again into roughly 2/3 of training and 1/3 testing examples. We have tested

Table 3. Overview of results obtained by RBF network

	# units	E_{train}		E_{test}		average time
		mean	std	mean	std	
cancer1	15	2.31	0.72	2.11	0.01	0:49:18
cancer2	15	1.91	0.26	3.12	0.07	1:1:3
cancer3	15	1.66	0.36	3.19	0.13	0:58:8
card1	10	8.12	0.75	10.16	0.56	0:23:23
card2	10	8.05	0.10	12.81	0.01	0:2:6
card3	10	6.77	0.09	12.09	0.00	0:55:12
flare1	10	0.37	0.01	0.37	0.00	1:12:33
flare2	10	0.41	0.00	0.31	0.00	0:39:3
flare3	10	0.37	0.00	0.38	0.00	0:51:34
glass1	20	5.10	0.14	6.76	0.02	0:4:31
glass2	20	4.93	0.06	7.96	0.00	0:4:51
glass3	20	5.80	0.98	8.06	0.97	0:3:24
heartac1	10	2.26	0.28	3.69	0.07	0:28:27
heartac2	10	1.78	0.19	4.98	0.03	0:28:20
heartac3	10	1.66	0.06	5.81	0.00	0:29:31
hearta1	15	3.08	0.08	4.36	0.00	0:25:12
hearta2	10	3.36	0.07	4.05	0.00	0:20:41
hearta3	10	3.19	0.04	4.29	0.00	0:36:2
heartc1	10	6.07	0.25	16.17	0.06	0:12:24
heartc2	10	7.99	0.19	6.49	0.03	0:21:34
heartc3	10	7.13	0.60	14.35	0.37	0:3:57
heart1	10	9.96	0.39	14.05	0.15	0:20:45
heart2	20	6.36	5.87	11.67	0.46	0:35:8
heart3	15	6.95	6.04	12.02	0.50	0:27:46
horse1	10	10.57	0.21	11.96	0.04	0:10:51
horse2	10	10.04	0.31	16.80	0.10	0:12:19
horse3	10	9.88	0.26	14.56	0.07	0:14:16
soybean1	30	0.28	0.06	0.73	0.00	0:48:32
soybean2	30	0.38	0.04	0.60	0.14	0:20:23
soybean3	30	0.31	0.09	0.72	0.01	0:40:52

series with one or two day history of rainfall and runoff as inputs, predicting next day runoff (output). Furthermore, two cases — with or without the next day rainfall value among inputs — have been considered. Thus, in total four different data sets have been created: 1- or 2-day history, either with or without the current rainfall value.

Both RBF networks and RN have been tested on these data. Results are summarized in the Table 6. Figure 5 shows the resulting prediction of flow rate by both RN and RBFN. Again, it can be seen that the performance of both architectures is comparable. On this example, it seems that RN tend to obtain better results for training error, but their generalization error is worse.

6 Conclusion

In this paper, two relative kernel-based network architectures have been described, and their capabilities compared. Several experiments have been performed to test the prac-

Table 4. Comparison of E_{test} of RN, RBF and MLP

	RN		**RBF**			**MLP**		
	E_{test}	# units	E_{test}		# units	E_{test}		arch.
			mean	std		mean	std	
cancer1	**1.76**	525	2.11	0.01	15	1.60	0.41	4+2
cancer2	**3.01**	525	3.12	0.07	15	3.40	0.33	8+4
cancer3	**2.80**	525	3.19	0.13	15	2.57	0.24	16+8
card1	**10.00**	518	10.16	0.56	10	10.53	0.57	32+0
card2	**12.53**	518	12.81	0.01	10	15.47	0.75	24+0
card3	12.32	518	**12.09**	0.00	10	13.03	0.50	16+8
flare1	0.54	800	**0.37**	0.00	10	0.74	0.80	32+0
flare2	**0.27**	800	0.31	0.00	10	0.41	0.47	32+0
flare3	**0.34**	800	0.38	0.00	10	0.37	0.01	24+0
glass1	6.95	161	**6.76**	0.02	20	9.75	0.41	16+8
glass2	**7.91**	161	7.96	0.00	20	10.27	0.40	16+8
glass3	**7.33**	161	8.06	0.97	20	10.91	0.48	16+8
heartac1	**2.78**	228	3.69	0.07	10	2.82	0.22	2+0
heartac2	**3.86**	228	4.98	0.03	10	4.54	0.87	8+4
heartac3	**5.01**	228	5.81	0.00	10	5.37	0.56	16+8
hearta1	4.40	690	**4.36**	0.00	15	4.76	1.14	32+0
hearta2	**4.05**	690	**4.05**	0.00	10	4.52	1.10	16+0
hearta3	4.43	690	**4.29**	0.00	10	4.81	0.87	32+0
heartc1	**16.02**	228	16.17	0.06	10	17.18	0.79	16+8
heartc2	**6.10**	228	6.49	0.03	10	6.47	2.86	8+8
heartc3	**12.66**	228	14.35	0.37	10	14.57	2.82	32+0
heart1	**13.65**	690	14.05	0.15	10	14.33	1.26	32+0
heart2	13.80	690	**11.67**	0.46	20	14.43	3.29	32+0
heart3	15.99	690	**12.02**	0.50	15	16.58	0.39	32+0
horse1	**11.90**	273	11.96	0.04	10	13.95	0.60	16+8
horse2	**15.18**	273	16.80	0.10	10	18.99	1.21	16+8
horse3	**13.58**	273	14.56	0.07	10	17.79	2.45	32+0
soybean1	**0.66**	513	0.73	0.00	30	1.03	0.05	16+8
soybean2	**0.49**	513	0.60	0.14	30	0.90	0.08	32+0
soybean3	**0.58**	513	0.72	0.01	30	1.05	0.09	16+0

tical properties and behavior of these networks both on benchmark and real-world data. It has been demonstrated that both types of networks represent a vital alternative to the most common network architecture, the multilayer perceptron, delivering comparable models in terms of training and generalization error.

Regularization networks have a strong mathematical background, yet in practical tests they may suffer from longer training times, mostly because of the cross-validation needed to find suitable values of free parameters (namely the regularization parameter γ). Another drawback can be a large number of their units — this can in turn sometimes worsen the generalization capabilities of the network trained. Practical experiments have also shown that finding the right value of the regularization parameter is crucial, since too large γ causes significantly worse error values.

Table 5. Correlation between various data characteristics and optimal width (the one that wins in grid search). Characteristics: minimal, maximal and mean distance between two data points, mean distance of 3, 5, and 10 nearest neighbors of each data point. Computed over Proben1 tasks.

Characteristic	Correlation coefficient
min	0.158
max	0.421
mean	0.552
3 nearest neighbors	0.357
5 nearest neighbors	0.360
10 nearest neighbors	0.290

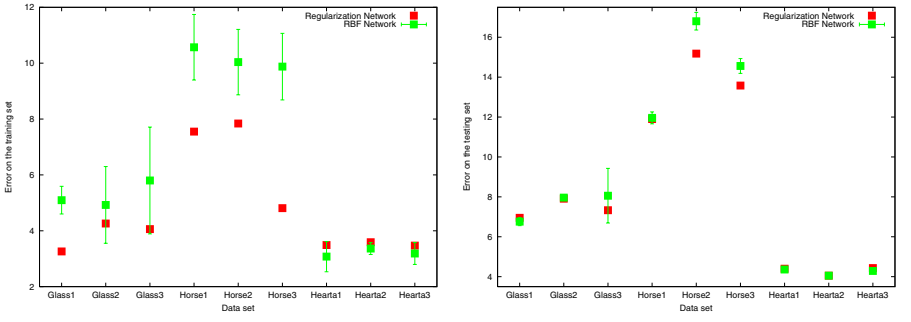


Fig. 4. Comparison of error values of RBF vs. RN results on training and testing data

Table 6. Results of RN and RBF on Ploucnice data sets. The plo1 and plo2 are data sets for 1- and 2-day history, the plo1r and plo2r are datasets for 1- or 2-day history with the current rainfall value.

	RN		RBF	
	E_{train}	E_{test}	E_{train}	E_{test}
plo1	0.057	0.048	0.059	0.049
plo1r	0.0257	0.0891	0.061	0.051
plo2	0.062	0.182	0.088	0.062
plo2r	0.0611	0.167	0.099	0.092

RBF networks may be trained by a variety of methods, out of which the gradient and three-step learning usually give the best results in reasonable times. When comparing time complexity of gradient learning of RBF networks with RN, one usually achieves 5–10 times better values for RBFN in order to achieve similar error. One can think of using an exhaustive search to find an optimal number of RBF units, which would take more time. But it is not a common practice to include search for the architecture size into a training time. It can be concluded that the RBF and Regulation networks both provide similar approximation and generalization results. Moreover, the RBFN usually

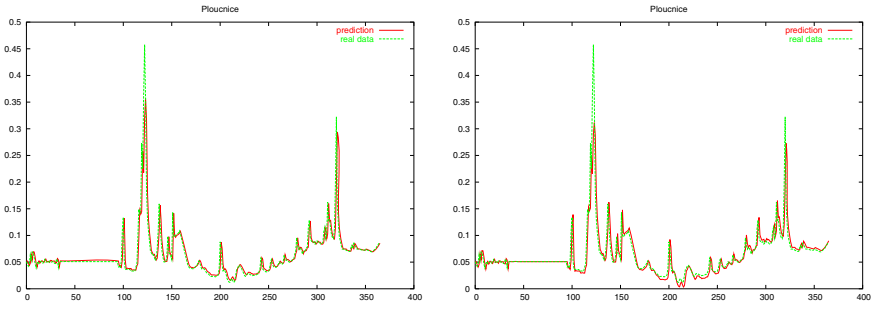


Fig. 5. Prediction of flow rate by a) RN b) RBF

have less parameters and take less time to train, thus it seems that at present they are more suitable for practical tasks, as a 'cheaper' substitute for RN.

Nevertheless, this work still leaves several problems open for both architectures. In the case of Regularization networks, one should search for a way to overcome the main drawback — a long lasting search for γ and b. To us, nothing is known about the dependency of those parameters, and the answer (both positive and negative) could be easily used to make the search more efficient. Preliminary investigations shows (cf. Tab. 5) that there probably is not a linear dependency of the width parameter on various characteristics of the data, thus a non-trivial algorithm to finding these is really necessary. Maybe employing a simple evolutionary algorithm would also help.

For the RBF networks, the area of learning algorithms seems to be more explored. Not much attention has been paid so far to RBF networks with different sets of basis functions. These are the topics we plan to address in the future research.

Acknowledgments

This work has been supported by the project no. 1ET100300419 *"Intelligent Models, Algorithms, Methods and Tools for the Semantic Web Realization"* of the Information Society Program (within the Thematic Program II of the National Research Program of the Czech Republic).

References

1. Haykin, S.: Neural Networks: a comprehensive foundation. 2nd edn. Tom Robins (1999)
2. Tikhonov, A., Arsenin, V.: Solutions of Ill-posed Problems. W.H. Winston, Washington, D.C (1977)
3. Girosi, F., Jones, M., Poggio, T.: Regularization theory and Neural Networks architectures. Neural Computation **7** (1995) 219–269
4. Poggio, T., Smale, S.: The mathematics of learning: Dealing with data. Notices of the AMS **50** (2003) 537–544
5. Girosi, F.: An equivalence between sparse approximation and support vector machines. Technical report (1997) A.I. Memo No. 1606.

6. Hansen, P.C.: Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion. SIAM, Philadelphia (1998)
7. Powel, M.: Radial basis functions for multivariable interpolation: A review. In: IMA Conference on Algorithms for the Approximation of Functions and Data, RMCS, Shrivenham, England (1985) 143–167
8. Neruda, R., Kudová, P.: Hybrid learning of RBF networks. Neural Networks World **12** (2002) 573–585
9. Neruda, R., Kudová, P.: Learning methods for RBF neural networks. Future Generation of Computer Systems (2005) (in print).
10. Prechelt, L.: Proben1 – a set of benchmarks and benchmarking rules for neural network training algorithms. Technical Report 21/94, Universitaet Karlsruhe (1994)

Redundant Bit Vectors for Quickly Searching High-Dimensional Regions

Jonathan Goldstein, John C. Platt, and Christopher J.C. Burges

Microsoft Research 1 Microsoft Way Redmond, WA 98052 USA

Abstract. Applications such as audio fingerprinting require search in high dimensions: find an item in a database that is similar to a query. An important property of this search task is that negative answers are very frequent: much of the time, a query does not correspond to any database item.

We propose *Redundant Bit Vectors* (RBVs): a novel method for quickly solving this search problem. RBVs rely on three key ideas: 1) approximate the high-dimensional regions/distributions as tightened hyperrectangles, 2) partition the query space to store each item redundantly in an index and 3) use bit vectors to store and search the index efficiently.

We show that our method is the preferred method for very large databases or when the queries are often not in the database. Our method is 109 times faster than linear scan, and 48 times faster than locality-sensitive hashing on a data set of 239369 audio fingerprints.

1 Introduction

Consider the abstract search problem: given a database of items (described in some way), and a stream of queries, how can we quickly find an item in the database that is similar to a query?

This search problem can be solved by mapping the items into geometric regions (e.g., hyperspheres) or probability distributions in a high dimensional space. Queries map into points in the same high-dimensional space. To determine whether a query is similar to an item, we determine whether the query point lies within the item's geometric region, or equivalently, whether the query point was likely to be generated by the item's probability distribution.

A simple method for executing the search is to perform a linear scan: for each item, determine whether the item includes the query. For a large number of items, this linear scan becomes very slow. Therefore, we use *indexing*: a set of precomputations whose results are stored in addition to the items. The index should assist in finding overlapping items more quickly than linear scan, but without using an excessive amount of extra memory.

This paper introduces Redundant Bit Vectors (RBVs): an indexing technique that allows us to quickly perform high-dimensional search. We show that RBVs excel at applications where many of the queries do not correspond to database items.

J. Winkler, N. Lawrence, M. Niranjan (Eds.): Machine Learning Workshop, LNAI 3635, pp. 137–158, 2005.

1.1 High-Dimensional Search in Audio Fingerprinting

We created RBVs for an important application: audio fingerprinting. A streaming audio fingerprinting system recognizes audio clips in an audio stream [1]. For example, such a system could recognize songs playing on the radio or in a bar. Audio recognition must be robust to distortions and noise in the audio stream: for example, radio stations compress broadcast songs in order to fit in more advertisements.

Audio fingerprinting can be mapped into high-dimensional search by converting a segment of audio into a vector of features that form the high-dimensional space. A database of known audio clips then get mapped into points in that high-dimensional space. For each clip, there is a sphere of acceptable distortions around each point: common audio distortions will perturb the point by a known amount. When a stream is being recognized, the sliding windows of the audio are continually being converted into high-dimensional queries. This stream of queries is compared to the database. When a query approximately matches a clip, the stream is identified.

Note that the vast majority of queries to the audio clip database are negative: there is no match in the database. That is because the database only stores a few samples from each song, to save space in the database. The stream away from those samples do not match anything in the database, and hence are negative queries.

It is important that the audio fingerprinting database be efficient in both time and space: it must store millions of songs in a database, and recognize queries for thousands of simultaneous users. Therefore, any indexing of the database must be smaller than the original database, while simultaneously speeding up search dramatically.

RBVs are applicable to audio fingerprinting, but are a generic technique. When machine learning is applied to signals, such as audio, images, or video, large fractions of the input are not relevant. When these irrelevant inputs are compared to a database (for detection or recognition), they should be rejected. We thus believe that high-dimensional search with negative queries should be common in applications of machine learning to signals. Fragment-based recognition of objects in images is another example of an application of such high-dimensional search [2].

1.2 Structure of Paper

Section 2 starts with a definition of the high-dimensional search problem, and Section 2.1 transforms that problem into a more tractable problem. Section 3 describes our RBV algorithm, including pseudo-code described in Section 3.3. Section 4 puts RBVs in the context of previous work. Section 5 compares RBVs to the best of the previous algorithms, on artificial data (Section 5.1) and real audio fingerprinting data (Section 5.2). We discuss extensions and future work in Section 6 and conclude in Section 7.

2 Definition of Problem

High-dimensional geometric search problems can be generated in several different ways. Raw data items can be extremely high-dimensional or not even lie in a vector space: they typically need to be pre-processed before they can be stored in a database. This pre-processing can be carried out by random projection [3][4], by principal components analysis, or by oriented principal components analysis [5]. If the data does not lie in a vector space, it can still get mapped to a vector space by kernel PCA [6] or multi-dimensional scaling [7].

Once the input space is chosen, the search problem can be formalized. Let i identify the items in the database ($i \in \{1..N\}$). Let $\boldsymbol{x} \in R^d$ be the location in the input space of a point with unknown label. Call this point with unknown label the *query*. Let $\boldsymbol{y}_i \in R^d$ be the location of the ith stored item. Let L_i be the label of the ith stored item. We wish to label $\boldsymbol{x}$ with a single label L_i for some i, or label it as "junk" (not in the database). Assume we have models for $P(\boldsymbol{x}|L_i)$ and $P(\boldsymbol{x}|\text{junk})$, with prior probabilities of label L_i and junk being $P(L_i)$ and $P(\text{junk})$. Then, simple decision theory leads us to the following search problem:

Problem 1. **Decision-theoretic search:** Find at least one i such that

$$P(\boldsymbol{x}|\text{item } i)P(i) > P(\boldsymbol{x}|\text{junk})P(\text{junk}). \tag{1}$$

Label $\boldsymbol{x}$ with L_i if it exists, or else label it junk.

Strictly, decision theory would have us find the L_i with the highest posterior. However, in practice, it is rare that a query has two labels that are more likely than junk. Therefore, for efficiency reasons, we allow the search to find one item that is more likely than junk, then allow it to stop.

Very often, we do not have a good model for "junk". We often assume that it is globally constant, or is constant in the region of high density for each item. We can use this assumption to create our first geometric search problem:

Problem 2. **Implicit surface search:** Find at least one i such that

$$f_i(\boldsymbol{x}) < c_i. \tag{2}$$

Label $\boldsymbol{x}$ with L_i if it exists, or else label it junk.

where f_i is a monotonic decreasing function of the likelihood, e.g.,$-\log(P(\boldsymbol{x}|L_i))$.

This implicit surface search is now in the realm of geometry. Each function f_i specifies a blob in the input space. We want to find whether any blob overlaps the query point.

Computing a general implicit surface (e.g., for a mixture of Gaussians) could be very computationally expensive. Also, we frequently do not have enough data to fit a complex model for every item. Instead, let us use a single spherical Gaussian per item. Combining with the negative log function yields our primary geometric search problem:

Problem 3. **Sphere overlaps point:** Find at least one i such that

$$||\boldsymbol{x} - \boldsymbol{y}_i||_2^2 < R_i.$$

Label $\boldsymbol{x}$ with L_i if it exists, or else label it junk.

2.1 Transforming Regions into Hyperrectangles

Problems 1, 2, and 3 are difficult to index, because all of the dimensions are coupled. That is, for points near the decision boundary, an index needs to know all of the coordinates to precisely determine which side of the boundary the point lies on. In high dimensions, most of the volume (or probability mass) of an item lies near the boundary of the region. Therefore, it is difficult to use lower-dimensional coordinates to determine whether a point is inside or outside of a high-dimensional region.

Our idea is to approximate the determination of the decision boundary surface. All problems stated so far are inexact: they allow a non-zero false negative rate — some items that truly match to item i will be excluded, even with perfect indexing. Therefore, let us use a method that may introduce more false negatives, but with a tunable parameter.

The idea is to approximate problems 1, 2, and 3 by the following problem:

Problem 4. **Hyperrectangle overlaps point:** Find at least one i such that

$$\max_n \left| \frac{x_n - y_{in}}{\epsilon_{in}} \right| < 1$$

where x_n is the nth component of $\mathbf{x}$. Label $\boldsymbol{x}$ with L_i if it exists, or else label it junk.

where y_{in} is the coordinate of the ith item in the nth dimension.

For spherically symmetric distributions or regions (such as Problem 3), we can reduce the number of parameters per item to 1, with

Problem 5. **Hypercube overlaps point:** Find at least one i such that

$$\max_n |x_n - y_{in}| < \epsilon_i$$

Label $\boldsymbol{x}$ with L_i if it exists, or else label it junk.

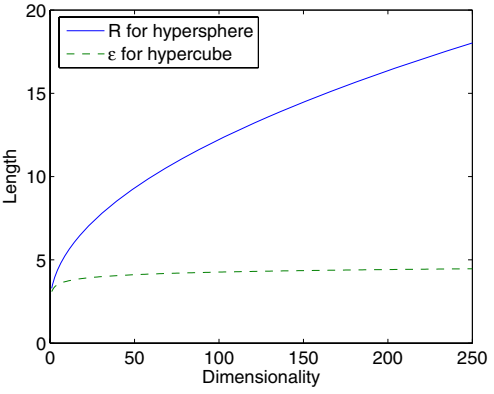


Fig. 1. R for the hypersphere and ϵ for the hypercube that encloses 99.9% of the probability mass of a unit spherical Gaussian

In general, we select the parameters ϵ_{in} or ϵ_i to be "too tight." That is, we can dramatically increase the indexability of the data set by approximating the sphere, regions, etc. with hypercubes that do not enclose the entire region defined in Problems 2 or 3.

If we have access to the original probability density for item i (in Problem 1), then we can use Monte Carlo: generate points from the distribution $P(\boldsymbol{x}|L_i)$ and find a hypercube or hyperrectangle that encloses $1-\delta$ of the generated points, if the desired false negative rate due to indexing is δ.

Alternatively, if the original probability distribution is unknown and we are solving Problems 2 or 3, we can generate Monte Carlo samples from a uniform distribution over each region, and then choose a hypercube or hyperrectangle that encloses $1-\delta$ of those samples.

Using these tight hyperrectangles yields suprisingly small regions to index. This can be seen in the following two examples. First, consider Problem 1, where $P(\boldsymbol{x}|\text{item } i)$ is a Gaussian with unit variance. If we convert this problem to use hyperspheres (Problem 3), the radius of the hypersphere that encloses $1-\delta$ of the probability mass is simply

$$R = \sqrt{(\chi^2_d)^{-1}(1-\delta)}, \tag{3}$$

where $(\chi^2_d)^{-1}(1-\delta)$ is the inverse of the cumulative chi-square density function with d degrees of freedom evaluated at $1-\delta$, $d =$ the dimensionality of the input space. In contrast, the ϵ for a hypercube that encloses $1-\delta$ of the probability mass is simply

$$\epsilon = \mathcal{N}^{-1}\left(1-\frac{\delta}{d}, 0, 1\right) \tag{4}$$

where $\mathcal{N}^{-1}$ is the inverse of the cumulative Gaussian density function.

The R of a hypersphere and the ϵ of the hypercube are plotted in Figure 1, for $\delta = 0.001$. As can be seen, the diameter of the sphere than mostly encloses

the Gaussian grows as $\sqrt{d}$, while the sidelength of the hypercube grows very slowly, for $d > 30$.

This surprising result indicates that the vast majority of a Gaussian distribution has very low L_∞ distance to the mean. Another way of thinking about this result is that samples from a Gaussian are very likely to be have distance from the origin near $\sqrt{d}$, for large d. It is very unlikely for a sample to have one dimension near $\sqrt{d}$ while all other dimensions are near zero. Most of the probability mass of a Gaussian is in the corners, away from the coordinate axes, because there are many more corners than axes in high-dimensional space.

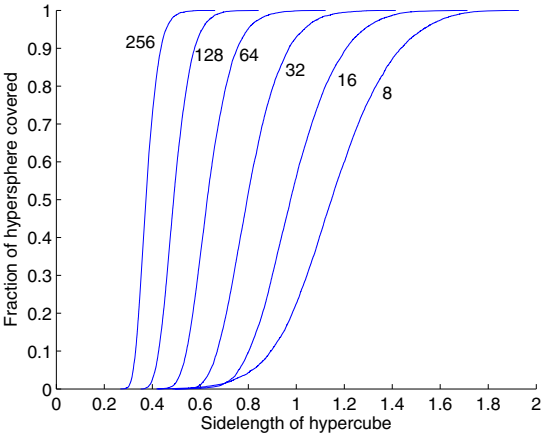


Fig. 2. Fraction of unit hypersphere covered as a function of the sidelength of the hypercube

To show that this result is not specific to Gaussians, we present Figure 2. This Figure is computed for Problem 3, where we are trying to fit a hypercube that encloses a large fraction of the volume of a hypersphere.[1]

The Figure shows that the size of the hypercube that encloses 99.9% of a hypersphere is substantially smaller than the size of the circumscribing hypercube (which would have sidelength=2), which would have a false negative rate of 0. Notice the dramatic savings by using a tight hypercube: in 256 dimensions, we can save a factor of 10^{134} in volume by using a tight hypercube instead of a circumscribing hypercube. This will both reduce the false positive rate and dramatically reduce the number of hypercubes needed to be searched (as will be seen in section 5).

[1] We generated Figure 2 by drawing uniform random samples from a hypersphere [8], then sorting them by their L_∞ distance to the origin. The x-axis of Figure 2 is then the L_∞ distance and the y-axis is the order in the list (scaled from 0 to 1).

3 Redundant Bit Vectors

The most straightforward way to solve Problems 4 or 5 is by a brute force method called "linear scan". In a linear scan, the bounds of each data item's hyperrectangle are checked to determine if the query point $\boldsymbol{x}$ lies within the hyperrectangle. Linear scan potentially involves performing $2N$ comparisons.

As an alternative, this paper introduces a technique that, while still linear in the number of operations, performs the search much faster than a linear scan. This is accomplished through the careful use of redundancy and precomputation.

3.1 Partition the Query Space

Non-redundant techniques for searching high dimensional data fail because they uniquely assign data entries to buckets/pages. Since, for truly high dimensional data, locality properties of data are weak [9], there is no one way of grouping like data entries together in a way that is frequently helpful during searching. Recent strategies to cope with this situation involve using redundancy to group data entries together in multiple ways such that a useful grouping frequently exists when the structure is queried [10,4].

In this tradition, we propose a way of generating redundancy that is particularly suited to high dimensional point queries over hyperrectangles. More specifically, we propose a redundancy strategy based on partitioning the query space rather than the data space.

In each dimension, we can compare the coordinate of the query point to intervals spanned by each of the stored item hyperrectangles. Per dimension, only a subset of the hyperrectangles overlap the query point. Now, notice that if we move the query point by a small amount, the hyperrectangles that overlap it stay largely constant: perhaps a few are added or dropped.

We can exploit this spatial structure by creating m bins (or intervals) of possible query points per dimension and pre-compute which hyperrectangles can overlap each query bin. These precomputed hyperrectangles are then stored in a bucket associated with the interval. Our index consists of the set of dm stored bins and buckets. For any query that lands in a bin, the set of hyperrectangles stored in the bucket is a superset of the hyperrectangles that overlap the query. We will use the buckets to systematically exclude possible items from our search. Thus, the grouping of queries into bins does not introduce false negatives.

Creating an index out of precomputed overlaps is a method for creating redundancy. To understand why, consider a single item and a single dimension. The hyperrectangle for that item appears redundantly in many buckets for that dimension. A query only accesses one bucket, so the algorithm does not need to backtrack to find all the hyperrectangles that may overlap it.

For example, in Figure 3, the dimension shown is partitioned into 4 bins. There are six hyperrectangles in the dataset labeled $R_1 - R_6$. Each bin is associated with a set of hyperrectangles that represent all possible answers to queries that land in the bin. Therefore, the bucket associated with the second bin, which

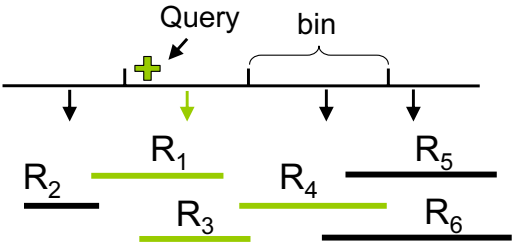


Fig. 3. This figure shows one dimension. In each dimension, the location of the query (the cross) gets assigned to one bin. Knowing the query lands in this partition eliminates certain rectangles: R_2, R_5, R_6.

corresponds to the highlighted arrow in the drawing, contains the three rectangles R_1, R_3, and R_4. Note that the first bin also contains R_1, which means that R_1 is represented twice (once in each bucket). Of course in this figure, the partition boundaries are given. In general, we will choose these partition boundaries using a fast heuristic described in Section 3.3.

Now that we have our index, when we run a query, for each dimension, we find the bin that contains the projected query point (e.g. the highlighted bin in Figure 3). This can be done with either a linear or binary search of bin boundaries. We then select the associated bucket of hyperrectangles, which is guaranteed to contain a superset of the correct hyperrectangle answers. Then we perform set intersection on all the selected hyperrectangle buckets, producing a smaller superset of the correct answer. Note that the result of the set intersection is all possible hyperrectangles that overlap a hyperrectangle in query space.

Even though each dimension may not be very selective, the intersection of the selected dimensions is quite small, since the selectivity of the final intersection is close to the product of all the individual dimensions' selectivities. For instance in our audio fingerprinting application, individual dimension selectivities ranged from $50-90\%$ (see Figure 4) while the selectivity after the intersection was much less than 1%.

For each query, this procedure leaves us with a superset of the correct answer. Since the superset is small compared to the original dataset, we can compute, for each candidate, whether the query point is actually contained in each candidate hyperrectangle without incurring the cost of a linear scan.

3.2 Indexing with Redundant Bit Vectors

While the above strategy for using the index to generate and prune candidate answer sets may sound reasonable at first glance, careful examination yields some problems. For instance, assume the buckets of hyperrectangles are represented using arrays of 4 byte data IDs. Given that most buckets found will contain most of the data entries, the amount of data that must be sorted and combined for set intersection is close to the amount of data in the dataset. Since sorting

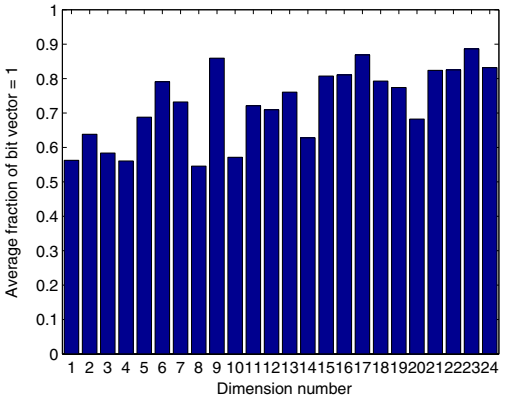


Fig. 4. For the audio fingerprinting task, average fraction of "1" bits in each bit vector, per dimension, weighted by bin selection frequency.

is a superlinear operation, this procedure will almost certainly be slower than a linear scan of the data.

In order to deal more effectively with these buckets, we must use a representation for the sets that is both more compact, and allows for linear time set intersection. Fortunately, bit vector indices [11] provide us with such a representation.

Instead of representing each bucket using an array of IDs, each bucket will be represented using a string of N bits. Each bit represents the presence of an item in the bucket. More precisely, the kth bit of the string is a 1 iff the item with ID k is in the set.

Figure 5 shows how bit indices are used in our example from Figure 3 to represent the buckets associated with the given bins. Note that, as mentioned earlier, R_1 is in both the first and second bucket. As a result, the first bit in the associated bit indices are both set to 1.

A nice consequence of using bit indices to represent the hyperrectangle buckets is that set intersection now becomes a linear bitwise intersection of the associated bit strings (e.g. logically AND all the bit strings together). Note that these linear bitwise operations are very efficient since given a CPU with 4 byte registers, set intersection between 2 sets for 32 data items is performed with 1 CPU operation. A 64-bit processor can process set intersections twice as quickly. Also, given the ordered nature of performing set intersection, excellent memory subsystem performance is achieved. Memory accesses become linear in nature and cache misses are relatively rare, leading to very high bandwidth from the memory subsystem. Finally, for very dense sets, the individual sets use approximately 1/32 of the space of the ID array approach. All these properties cumulatively work to produce an algorithm for set intersection that, while linear in the number of data entries, is extremely efficient.

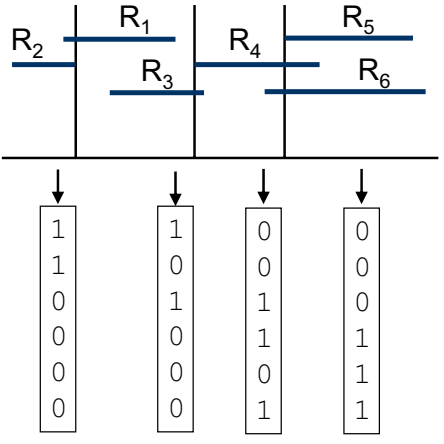


Fig. 5. For each bin, store whether each rectangle overlaps that bin in a bit vector: the ith bit represents whether the hyperrectangle associated with the ith item in the database overlaps this query bin.

The final algorithm for performing a query using our index is shown in Algorithm 1.. The first *for* loop loads the appropriate bit index from the first dimension into C. Note that the arrays lo and hi will be explained in more detail later: the algorithm functions as if $lo = 1$ and $hi = N$. The second loop iterates over each dimension beyond the first. For each dimension, the appropriate bit index is selected and intersected with C. The final loop iterates over C, which contains the result of all the index intersections, and performs a final test for each remaining item.

The bit vector representation highlights the redundancy that arises from query partitioning. If we examine the bits for a single item across all bins for one dimension, they look like

$$\underbrace{0000000}_{\text{item above query}}\underbrace{1111111111111}_{\text{item overlaps query}}\underbrace{000000}_{\text{item below query}}$$

Thus, the span of the item hyperrectangle (relative to the bin boundaries) is stored in a redundant binary code. Only one bit of this binary code need be accessed for each query.

3.3 Details and Notes

We have not yet described how the partition boundaries are determined. We employ a heuristic that is designed for a particular dimension, to evenly spread the selectivity amongst all indices for that dimension. This uniform selectivity

Algorithm 1. Querying the Redundant Bit Vector index

Require: Database of N items/distributions, Bit vector index B, Bin edge array E, hi and lo arrays, number of indexed dimensions I, query vector x
  j = smallest index such that $x_1 < E_{1j}$
  for i =lo[j] to hi[j] **do**
    $C_i = B_{1ji}$
  end for
  for $k = 2$ to I **do**
    j = smallest index such that $x_k < E_{kj}$
    for i =lo[j] to hi[j] **do**
      $C_i = C_i \& B_{kji}$ (once every machine word)
    end for
  end for
  for all i such that $C_i = 1$ **do**
    if $x \in$ item$_i$ **then**
      Finished
    end if
  end for
  If no item found, return empty

is accomplished by keeping the Hamming distance between adjacent bit indices roughly constant. More precisely, our heuristic first sorts the interval boundaries of the projected data, resulting in a sorted list of $2N$ interval boundaries. Given a user defined number of partitions Q per dimension, we divide the sorted list into Q maximally equal sized pieces and choose as the partition boundaries the last elements of the first $Q - 1$ pieces.

We are now ready to present the complete index-building algorithm shown in Algorithm 2.. The first *for* loop calculates and stores the partition edges as described previously. The space for the bit vectors is then allocated to ensure that the individual bit vectors are packed linearly in memory. The next *for* loop then calculates the actual bit vectors. Finally, the last loop sets *lo* and *hi*.

lo and *hi* are used to reduce the portion of the bit indices that have to be "*AND*"ed during query time. If, while querying, the first bit vector selected begins with 800 "off" bits (100 bytes of 0 values), there is no point in using the first hundred bytes of any of the indices for this query, since the first index will ensure that the first 100 bytes of the result is all zeroes. Therefore, for the first dimension, for each bit index, we keep track of the leading and trailing number of "off" bits and use this to reduce the work that must be done during querying. This technique is made most effective by always choosing the most selective dimension for the first dimension, and by sorting the dataset by the first dimension. The result is that the "on" bits tend to cluster together somewhere in the bit index. For instance, the first bit index of the first dimension has many trailing zeroes while the last bit index of the first dimension has many leading zeroes.

Algorithm 2. Building the Redundant Bit Vector index

Require: Database of N $d-$dimensional vectors x_{ij}, hypercube half-sidelength ϵ_i for each vector, number of bins per dimension Q, number of indexed dimensions I.
Sort database by increasing value of most selective dimension
Initialize temp array $t[2N]$
for $k = 1$ to I **do**
  for $j = 1$ to N **do**
    Append $x_{jk} \pm \epsilon_j$ to t
  end for
  Sort t
  for $j = 1$ to $Q - 1$ **do**
    Bin edge array $E_{kj} = t_{\lceil 2jN/Q \rceil}$
  end for
end for
Initialize bitvector $B[I, Q, N]$ (last dimension packed into machine words)
for $k = 1$ to I **do**
  for $j = 1$ to N **do**
    Initialize B_{k*j} to all 1
    for all i such that $x_{jk} + \epsilon_j \leq E_{ki}$ **do**
      $B_{k,i+1,j} = 0$
    end for
    for all i such that $x_{jk} - \epsilon_j \geq E_{ki}$ **do**
      $B_{kij} = 0$
    end for
  end for
end for
for $j = 1$ to Q **do**
  lo[j] = index of first "on" bit in B_{1j} (rounded down to word boundary)
  hi[j] = index of last "on" bit in B_{1j} (rounded up to word boundary)
end for

4 Previous Work

There is an extensive literature in speeding up high-dimensional search. In order to understand the related work, we must transform Problem 3 into one of two related problems. First, Problem 3 is equivalent to the following problem, if all of the spheres' radii are the same:

Problem 6. **ε-range search** Find at least one i such that

$$||\boldsymbol{x} - \boldsymbol{y}_i||_2^2 < R$$

Label $\boldsymbol{x}$ with L_i if it exists, or else label it junk.

There is an extensive literature in approximately solving Problem 6 (e.g., [12]). This Problem has been more extensively studied than Problem 3, because it permits the database to store $\boldsymbol{y}_i$ as points, and consider the query as a sphere around the point $\boldsymbol{x}$. Thus, only points need to be indexed, not regions.

The research into Problem 6 has culminated in Locality-Sensitive Hashing (LSH) [4], which has sublinear time performance. We compare RBV to LSH in Section 5, below.

LSH operates by randomly projecting an input space into a number of dimensions. This random projection can be accomplished by taking a dot product with a vector drawn from a spherical Gaussian distribution with unit variance. Each projected dimension is divided into randomly-offset bins, whose width is proportional to the radius of the sphere in the original space. The bin number is then a hash function. A number of these hash functions, k, are then grouped together to form a key in a hash table: the k integers hashed together with an additional hash function is the key. The index of all items that match that hash table key are then associated with the key.

LSH maintains a redundant data structure by constructing l different hash tables in this way, each with their own random projections and random bin offsets. All l of these hashtables are checked, until either one item that solves Problem 6 is found, or else all hashtables are searched.

4.1 Nearest Neighbor Search

If we start to solve Problem 6 for small r, and then gradually increase r if no points are found, then we are solving:

Problem 7. **Nearest neighbor search**

$$c = \arg\min_i ||\boldsymbol{x} - \boldsymbol{y}_i||_2^2$$

Label $\boldsymbol{x}$ with L_c.

Again, there is an extensive literature in exactly or approximately solving nearest neighbor search in logarithmic time [13][14][15][16][17]. However, these techniques always force the query to map to one item, even if the item does not match well. Thus, the methods can spend a lot of index time and space finding the nearest neighbor, even when the query is clearly junk. Also, these methods often have exponential dependence on dimensionality. Therefore, we do not test these methods in this paper.

5 Speed and Memory Comparisons

In order to test the effectiveness of RBVs, we tested them versus LSH and linear scan. We perform two major tests: first, we test on an artificial dataset, which can be parameterized to explore dependency on the number of database items, dimensionality, and the radius of the sphere; second, we test on real audio fingerprinting data, to check the effectiveness on a problem that we care about.

Both RBV and LSH have a tuning parameter that performs a memory vs time trade-off. For RBVs, the number of indexed dimensions can affect the speed,

while for LSH, the number of hashtables can affect the speed. For both RBV and LSH, the parameter was tuned to optimize speed and the optimal speed was reported.

All experiments were performed on an unloaded Xeon 2.4 GHz processor running Windows Server 2003 with 1.5 GB of physical memory. All real numbers were stored as 4-byte floating point. Because the database of items must be loaded in RAM, we limited the size of any index to be 800MB. In the figures below, you can see the LSH memory consumption curves saturating at that level, which causes upward kinks in the LSH time curves.

In order to ensure a fair comparison, we optimized an LSH implementation. The most important implementation detail was to first build the hash tables for LSH using chaining, then copy the hash tables into a single array using linear probing, where the length of each chain is stored in a separate array. Since the index is only built once, this saves a large amount of memory: each entry uses only 16 bytes of memory (in contrast to [4], which used 60 bytes of memory per entry). As suggested by [4], we used a second hash function to disambiguate collisions in the main hash table (rather than checking for equality on the separate components of the first hash function). Following [18], we set the bin size on the output of each projection to be 4 times the radius of the sphere in the input space. Finally, as in the RBV linear scan, we halt the search through the hashtables when a match is found.

5.1 Artificial Data

To test indexing under various conditions, we generated an artificial training set, which consists of random deviates drawn from a spherical Gaussian distribution of unit variance. Each point becomes the center of a sphere. All spheres have equal radius.

We created two kinds of tests: "positive queries" and "negative queries." Positive queries are those which are very likely to have a match in the database, while negative queries are very unlikely to have a match.

We generated positive queries (following [4]) by adding Gaussian random noise to 1000 randomly selected points in the database. The variance of the Gaussian noise was selected to introduce a false negative rate of 10^{-3} (given a previously chosen radius).

We generated negative queries by generating 1000 randomly selected points from the unit spherical Gaussian distribution. The radius of the spheres was chosen to produce a false positive rate of 10^{-10}. Given this low false positive rate, it is very unlikely that new draws from the same distribution will end up inside any of the spheres.

The false positive and false negative rates were chosen to be realistic for a fingerprinting application: the false positive rate (per database item) must be very low, because there can be tens of millions of items in a fingerprinting database.

We also tuned the RBV and LSH algorithms to produce a false negative rate of 10^{-3}, to be compatible with the underlying false negative rate of the problem.

The false negative rate of the RBV method is tuned by choosing the size of the hypercubes. This is done by examining Figure 1 and scaling the unit Gaussian of the Figure to match the variance of the Gaussian used to generate the noise. For LSH, the false negative rate is tuned by the number of hashtables [19] via the formula

$$l = \lceil \frac{\log(\delta)}{\log(1 - p^k)} \rceil, \tag{5}$$

where l is the number of hashtables, δ is the desired false negative rate, k is the number of hash functions that are used to generate a key, and p is the probability that two input vectors that are the sphere radius R apart will collide under the same hash function. Since we are using Gaussian random projections in LSH, $p = 0.8$.

Vary Database Size, Fix Dimensionality, Positive Queries. For the first experiment, we fix the dimensionality of the Gaussian to be 64, and the radius of all spheres to be $R = 5.6239$, to yield a false positive rate of 10^{-10}. For the RBV, we used a hypercube sidelength of 4.5767. We added Gaussian noise of variance 0.3020 to items in the database to generate queries. As we add items to the database, we measure how much time and memory each of the methods consume.

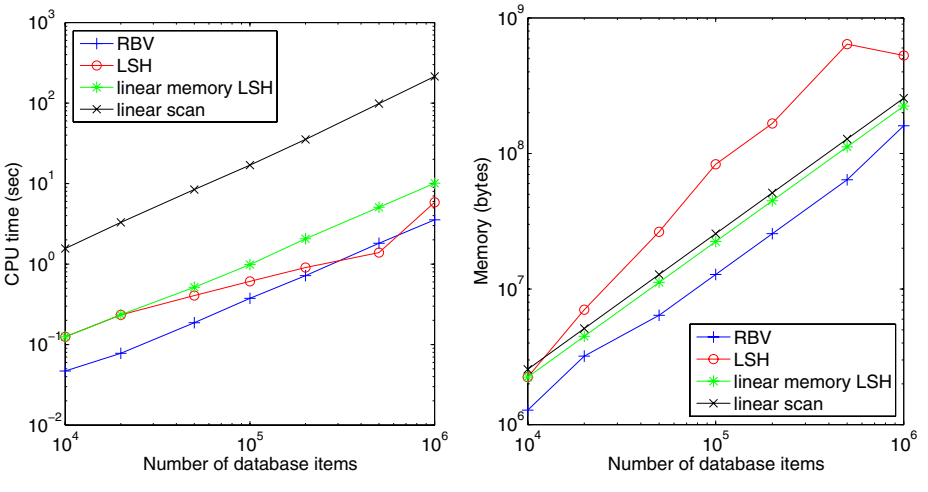


Fig. 6. Positive queries: Log-log plot of CPU time for 1000 queries (left) and memory (right) vs. database size, for RBVs, LSH, LSH limited in size to database size, and linear scan. In the memory plot, all schemes show the size of the index, except for linear scan, where the size of the database is plotted.

Our first experimental results are for positive queries, shown in Figure 6. Both RBV and LSH beat linear scan by a substantial margin. RBV grows linearly in time and space, being approximately 46 times faster than linear scan and requiring an average of 53% of the size of the database items to store the bit

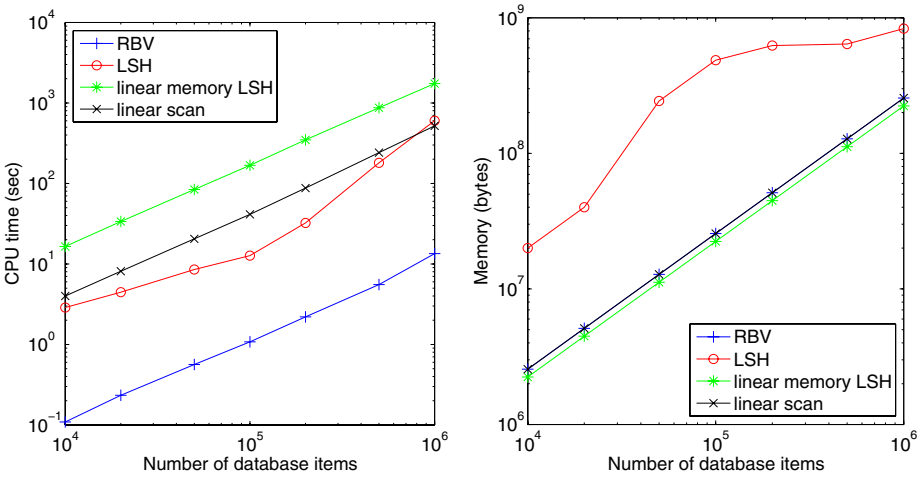


Fig. 7. Negative queries: Log-log plot of CPU time for 1000 queries (left) and memory (right) vs. database size, for RBVs, LSH, LSH limited in size to database size, and linear scan. Note that RBV and linear scan consumes the same amount of memory in this experiment.

vector indices. RBVs reduced the number of items for final linear scanning by approximately a factor of 700 (compared to the total size of the database).

These positive queries are similar to the experiments performed in [4], which show sublinear time for LSH. This can be seen in the slope of the LSH line performance, which corresponds to a scaling of $O(N^{0.60})$. The speed of LSH surpasses BVs at approximately 300,000 items. Notice that, at 1,000,000 items, the memory required by LSH reaches the 800 MB limit, which causes LSH to be unable to use the optimal number of hash functions and tables, and therefore slow down.

Unlike RBV, the memory required by LSH grows superlinearly with database size. Fitting a line to the points in Figure 6 yields a memory requirement that scales as $O(N^{1.43})$. This can become an issue when scaling up to large databases: a 40 million item database would require 350 GB of RAM to store the hashtables, while the RBV method would require only 5 GB of RAM.

In this experiment, we test LSH in another way: we chose the number of hash functions k (hence the number of hashtables l) so that LSH takes up as much memory as the items in the database, then tested LSH's speed. Interestingly, in this experiment, forcing LSH to take a linear amount of memory caused it to have linear time performance: RBVs are an average of 2.8 times faster than linear-memory LSH.

Thus, for situations where memory is constrained (as in very large databases), RBV is preferable to both LSH and linear scan.

Vary Database Size, Fix Dimensionality, Negative Queries. Our next experiment is shown in Figure 7. Here, we present the more realistic scenario for

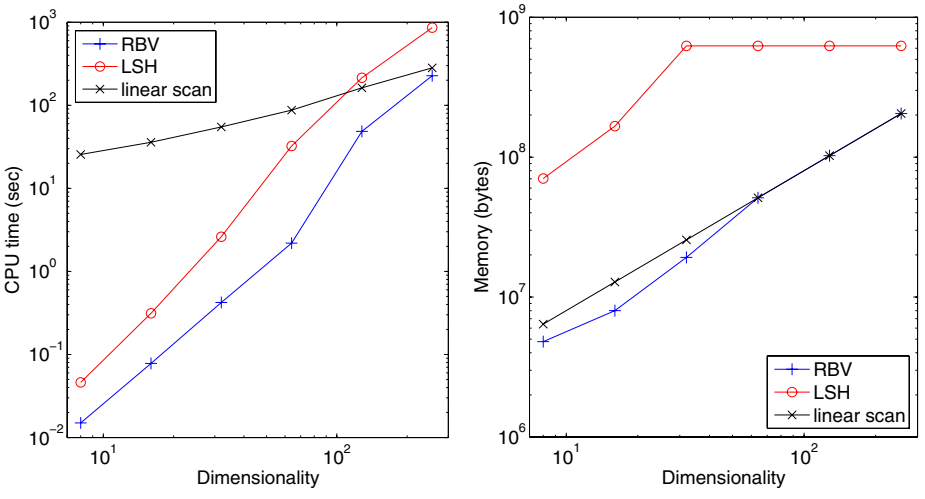


Fig. 8. Log-log plot of CPU time for 1000 queries (left) and memory (right) vs. dimensionality, for RBVs, LSH, and linear scan. Results shown for negative queries. False positive and false negative rates were held constant across dimensionality.

fingerprinting-like search, where the vast majority of queries do not match in the database. For negative queries, RBV still maintains its superiority over linear scan. Here, the index requires as much memory as storing all the database items, and is 38 times faster than linear scan. RBVs reduced the number of items for final linear scanning by approximately a factor of 200.

For negative queries, LSH becomes much slower. Part of the reason why LSH is fast on positive queries is because it stops searching the hashtables when it finds a match. This is not possible on negative queries, so LSH must slow down. Performing a line fit to the first 4 points in Figure 7, we see that the time for LSH is still sublinear in time ($O(N^{0.65})$), and is still superlinear in memory ($O(N^{1.5})$), but with a much worse constant than RBV. Performing a possibly inaccurate extrapolation, we would expect LSH to be as fast as RBV for a database of 72 million items, but require 8 terabytes of RAM, while RBV would require 18 gigabytes. Clearly, LSH is not feasible in this situation. Thus, RBV is clearly the method of choice when negative queries make up even a small fraction of the input.

As in the positive query case, if we force LSH to have linear growth in memory, it has linear growth in time: RBV is 148 times faster than LSH with linear memory.

Fix Database Size, Vary Dimensionality. V To test the behavior of the algorithms in high dimension, we varied the dimensionality of the Gaussian that generated the items in the database while keeping the number of database items at 200,000. There are multiple ways to set the sphere radii as the dimensionality rises. We chose to keep the false positive rate of the system constant at 10^{-10},

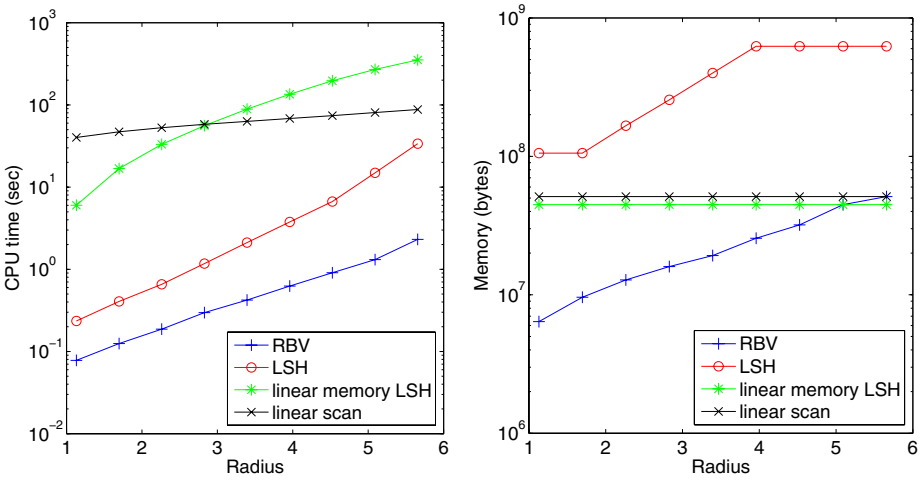


Fig. 9. Log-log plot of CPU time for 1000 queries (left) and memory (right) vs. radius of spheres, for RBVs, LSH, and linear scan. Results shown for negative queries.

Table 1. Parameters used in dimensionality experiments

Dimensionality	Sphere Radius	Hypercube Sidelength
8	0.1674	0.2399
16	0.9313	1.1405
32	2.6768	2.7111
64	5.6239	4.5767
128	10.0834	6.4372
256	16.5662	8.1340

and the false negative rate constant at 10^{-3}. These choices yielded parameters as shown in Table 1.

This choice forces the sphere radius to grow as a fraction of the mean interpoint distance, because the distribution of the interpoint distances are approaching a delta function. This growing radius makes the projections of the hypercubes ever larger, which interferes with efficient indexing, as can be seen in Figure 8. Here, linear scan is as efficient as LSH for $d = 128$, and as efficient as RBV for $d = 256$.

Under these severe conditions, all indexing techniques break down in high enough dimension. However, real high-dimensional problems may have more benign conditions than those in table 1: the techniques need to be tested to see if they are effective on a particular data set.

Fix Database Size, Vary Sphere Radius. As a final experiment on artificial data, we fixed the database size to be 200,000 items and the dimensionality at 64. We shrink the radius of the hyperspheres to see if RBV maintains its superiority

over LSH, even for easy cases (at very low false positive rates). We maintain the indexing false negative rates at 10^{-3}, with a RBV sidelength of 0.5033 the sphere diameter.

The results of this experiment are shown in Figure 9, which shows that, for negative queries, all indexing techniques get faster when the false positive rate tends to zero. RBV maintains its superiority over both variants of LSH, although LSH starts to approach the RBV performance for small sphere radii: RBV is 14 times faster than LSH at radius 5.659, but only 3 times faster at radius 1.1314.

5.2 Real Audio Fingerprinting Data

As a true test of the RBV method, we applied RBV to the audio fingerprinting task described in [1] and in Section 1.1. In this task, 6 seconds of audio are represented as a 64-dimensional vector. The database consists of 239369 items, each item taken from a unique song. An item is compared to a query with Euclidean distance.

For this test, we used realistic query data: 1000 queries taken from sequential frames of a single song.

Audio fingerprinting adds additional complexity to indexing methods: the radii of the spheres are not identical. According to [1], each radius is chosen to be 0.374 of the average distance of the item to the closest vector taken from 100 randomly chosen songs. This ensures that the false positive rate is more uniform across the database of items: the radii vary by a factor of 3. The false positive rate reported in [1] is roughly 10^{-6}.

RBVs can handle variable sphere radii in Algorithms 2. and 1.. The LSH algorithm needs a fixed radius to compute the hash functions: we use the largest radius in the database for that radius.

As in the artificial data case, we set the false negative rate for 10^{-3} for both RBV and LSH. For RBV, this yields hypercube sidelength S_i:

$$S_i = 0.5033 D_i \tag{6}$$

where D_i is the diameter of the ith hypersphere.

Figure 10 shows the performance of RBVs, linear scan, and LSH on the real audio fingerprinting database. Because the audio fingerprinting data is not parameterized, we show the performance of RBVs and LSH as a function of their speed/memory tradeoff parameters: sweeping the number of indexed dimensions for RBV, and the number of hash functions used to construct a key in LSH.

For the audio fingerprinting task, RBV is much faster than other techniques. RBV is 109 times faster than linear scan and requires 3/8 of the memory to store the database items. RBV is 48 times faster and requires 34 times less memory than LSH.

We have previously tested SS-trees [15], and R-trees for this task [20], and found them to be worse than linear scan: we do not present those results here.

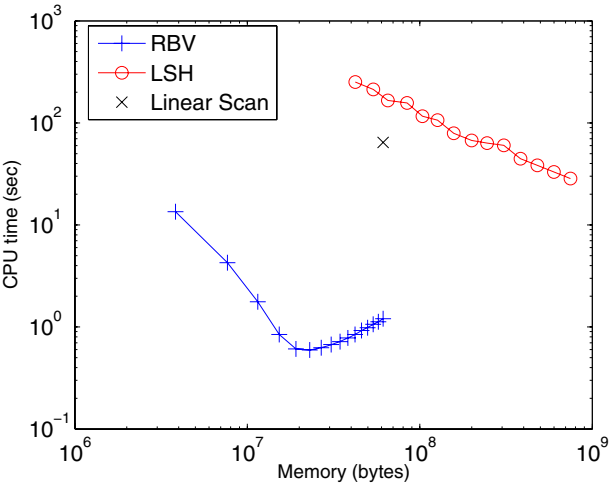


Fig. 10. Log-log plot of CPU time for 1000 queries vs. memory for RBVs, LSH, and linear scan (one point). Curves are generated by changing the number of indexed dimensions (for RBVs) and the number of hashtables (for LSH).

6 Extensions and Future Work

So far, this paper has assumed that the query is a point and that we want to find all data regions that contain the query point. We may also want to test which data regions spatially overlap a given query region. The query region can be mapped to a hyperrectangle in the manner described previously for mapping items. We now are testing overlap between a query hyperrectangle and all item hyperrectangles.

This test can be accomplished by building two sets of redundant bit vector indices. The first set divides the query space on the lower bound of a query hyper-rectangle while the second divides the query space on the upper bound. In other words, the answer sets associated with the lower bound index contain all data hyper-rectangles that may spatially overlap a query rectangle whose lower bound falls into the partition associated with that set. Similarly, the answer sets associated with the upper bound index contain all data hyper-rectangles that may spatially overlap a query rectangle whose upper bound falls into the partition associated with that set. When a query is performed, 2 indices are picked per dimension; one for the lower bound of the query hyper-rectangle and one for the upper bound. As with previous strategies, all bit indices are "AND"ed together to generate a candidate list.

Possible future work also includes comparing redundant bit vector indices and LSH to R-trees and approximate kd-trees. A particularly interesting question is in the cases where LSH is competetive or better than redundant bit vector indices (for positive queries), do other techniques like R-Trees or kd-trees outperform both? This may be possible, since they are also sublinear techniques when applied to easier problems.

7 Conclusions

This paper has introduced Redundant Bit Vectors (RBVs), a method guaranteed to have linear time and memory complexity.

RBVs are built on three key ideas:

1. *High-dimensional geometry* — We use the fact that we can approximate hyperspheres (or more general regions) with hypercubes that are substantially tighter. This improves the selectivity of the indexing.
2. *Partition the queries, not the data* — We avoid the difficult task of partitioning the data by partitioning the queries into bins per dimension. These bins become "coarse queries," which can be precomputed and stored. These precomputations form a redundant binary code for the location and size of the objects. This redundant code means we do not need to backtrack a data structure to find objects.
3. *Bit vectors* — We store the precomputed indicies in bit vectors, which gives two advantages: first, we can become much faster than linear scan by exploiting the innate 32-way or 64-way bit-wise parallelism of modern CPUs. Second, our index is extremely compact and takes up less memory than the database of objects.

Realistic applications, such as audio fingerprinting, can have a large fraction of negative queries. RBVs are the first high-dimensional indexing scheme that is much faster than linear scan with a large fraction of negative queries. We showed that the best previous known algorithm (LSH) is slower than RBVs and requires an unrealistic amount of memory in this situation.

Acknowledgements

We would like to acknowledge Shimon Ullman, for suggesting that RBVs could be applied to his fragment-based object recognition task.

References

1. Burges, C.J., Platt, J.C., Jana, S.: Distortion discriminant analysis for audio fingerprinting. IEEE Transactions on Speech and Audio Processing **11** (2003) 165–174
2. Ullman, S., Sali, E., Vidal-Naquet, M.: A fragment-based approach to object representation and classification. In: Proc. of the 4th Intl. Workshop on Visual Form. Volume 2059 of Springer-Verlag Lecture Notes In Computer Science. (2001) 85–102
3. Achlioptas, D.: Database-friendly random projections. In: Proc. of the 20th Ann. Symp. on Principles of Database Systems. (2001) 274–281
4. Datar, M., Immorlica, N., Indyk, P., Mirrokni, V.S.: Locality-sensitive hashing scheme based on p-stable distributions. In: Proc. of the 20th Ann. Symp. on Computational Geometry. (2004) 253–262
5. Diamantaras, K., Kung, S.: Principal Components Neural Networks. John Wiley (1996)

6. Schölkopf, B., Smola, A.: Learning with Kernels. MIT Press (2002)
7. Cox, T.F., Cox, M.A.A.: Multidimensional Scaling. Chapman and Hall (1994)
8. Tax, D.M., Duin, R.P.: Uniform object generation for optimizing one-class classifiers. Journal of Machine Learning Research **2** (2001) 155–173
9. Beyer, K.S., Goldstein, J., Ramakrishnan, R., Shaft, U.: When is "nearest neighbor" meaningful? In: Proc. of the 7th Intl. Conf. on Database Theory. Volume 1540 of Springer-Verlag Lecture Notes in Computer Science. (1999) 217–235
10. Goldstein, J., Ramakrishnan, R.: Contrast plots and P-Sphere trees: Space vs. time in nearest neighbour searches. In: Proc. of the 26th Intl. Conf. on Very Large Databases. (2000) 429–440
11. O'Neil, P., Quass, D.: Improved query performance with variant indexes. In: Proc. of the 1997 ACM SIGMOD Intl. Conf. (1997) 38–49
12. Arya, S., Mount, D.M.: Approximate range searching. In: Proc. of the 11th Ann. Symp. on Computational Geometry. (1995) 172–181
13. Bentley, J.L.: Multidimensional binary search trees used for associative searching. Communications of the ACM **11** (1979) 397–409
14. Guttman, A.: R-trees: a dynamic index structure for spatial searching. In: Proc. of the 1984 ACM SIGMOD Conf. (1984) 47–57
15. White, D.A., Jain, R.: Similarity indexing with the SS-tree. In: Proc. of the 12th Intl. Conf. on Data Engineering. (1996) 516–523
16. Berchtold, S., Keim, D., Kriegel, H.P.: The X-tree: an index structure for high-dimensional data. In: Proc. of the 22nd Intl. Conf. on Very Large Databases. (1996) 28–39
17. Pagel, B.U., Korn, F., Faltusos, C.: Deflating the dimensionality curse using multiple fractal dimensions. In: Proc. of the 16th Intl. Conf. on Data Engineering. (2000) 589–598
18. Andoni, A., Indyk, P.: E2LSH 0.1 user manual. Technical report, Massachusetts Institute of Technology (2004) http://web.mit.edu/andoni/www/LSH/manual.pdf.
19. Cohen, E., Datar, M., Fujiwara, S., Gionis, A., Indyk, P., Motwani, R., Ullman, J.D., Yang, C.: Finding interesting associations without support pruning. IEEE Transactions on Knowledge and Data Engineering **13** (2001) 64–78
20. Goldstein, J., Platt, J.C., Burges, C.J.: Indexing high-dimensional rectangles for fast multimedia identification. Technical Report MSR-TR-2003-38, Microsoft Research (2003)

Bayesian Independent Component Analysis with Prior Constraints: An Application in Biosignal Analysis

Stephen Roberts and Rizwan Choudrey

Robotics Research Group, University of Oxford, UK

Abstract. In many data-driven machine learning problems it is useful to consider the data as generated from a set of unknown (latent) generators or sources. The observations we make are then taken to be related to these sources through some unknown functionaility. Furthermore, the (unknown) number of underlying latent sources may be different to the number of observations and hence issues of model complexity plague the analysis. Recent developments in Independent Component Analysis (ICA) have shown that, in the case where the unknown function linking sources to observations is linear, data decomposition may be achieved in a mathematically elegant manner. In this paper we extend the general ICA paradigm to include a very flexible source model and prior constraints and argue that for particular biomedical signal processing problems (we consider EEG analysis) we require the constraint of *positivity* in the mixing process.

Keywords: Independent component analysis, biosignal analysis, variational Bayes, prior constraints.

1 Introduction

Independent Component Analysis (ICA) has been widely used in data analysis and decomposition in recent years (see, for example, [1,2] for an overview). ICA, typically, aims to solve the *blind source separation* problem in which a set of unknown sources are mixed in some way to form the data that are available. The classic example being that of multiple speakers in a room with several microphones. Several ICA algorithms exploit the fact that the probability density functions (pdfs) of mixtures of sources are more Gaussian (normally distributed) than the source pdfs themselves and hence forming source estimates whose pdfs are maximally non-Gaussian may achieve the goal of ICA [3,4]. Alternatively, ICA may be seen as a *generative model* in a probabilistic framework [5,6,7,8,9]. This has the advantage that, in principle, fully Bayesian models may be formulated in which *priors* exist over all the model parameters. This enables, for example, model selection (to determine e.g. how many sources there are) and the principled handling of uncertainty and noise. Recently the use of fully Bayesian, very flexible ICA models has been realized [10,11,12]. The application of ICA techniques to biomedical signals has received considerable attention in recent years, in part due to the linear mixing assumptions of 'traditional' ICA being approximately correct in many biomedical problems [13,14,15,16,17].

J. Winkler, N. Lawrence, M. Niranjan (Eds.): Machine Learning Workshop, LNAI 3635, pp. 159–179, 2005.

In this paper we consider the case in which constraints exist in our beliefs regarding the mixing process and the sources. In particular we investigate the issue of enforcing *positivity* onto the ICA model. As the models are evaluated in a fully Bayesian manner model selection may be applied to infer the complexity of data representation in the solution space.

This paper is organised as follows. We first introduce the basic concepts of ICA theory and our data decomposition goals. The issue of why a constrained model may be more appropriate are then discussed. Details of the variational Bayes paradigm under which model inference is performed are then presented. Representative results are given in the next section followed by conclusions and discussion.

We have decided to concentrate on concepts and example results in this paper. Whilst the mathematical details of the theory are of importance, they are fully covered elsewhere, for example in [10,11,12,18]. The problems associated with ICA are an intriguing case study in model complexity selection and inference in a scenario where very little information regarding the 'ground truth' is available. As such, then, this paper offers evidence that probabilistic modelling may be useful in overcoming many of these problems in a principled manner. It must be remembered that these are generic issues which plague many data-driven machine learning problems.

2 Generic ICA Theory

In general, we have a set of observations vectors, $\mathbf{x} \in \mathbb{R}^N$ which are believed to be caused by a set of underlying *latent* sources, $\mathbf{s} \in \mathbb{R}^M$, which we cannot directly access.

We start by reviewing the simplest ICA model, that of noiseless mixing; in part to introduce nomenclature but also to consider the more general issue of decompositional models. We model the observations vector (at some time instant, t), $\mathbf{x}[t]$, as a linear mixing via the (time-invariant) *mixing matrix*, $\mathbf{A}$, of a source vector (at the same time instant – we do not here consider *convolutive* or *time-delay models*), $\mathbf{s}[t]$.

$$\mathbf{x}[t] = \mathbf{A}\mathbf{s}[t]. \tag{1}$$

The above vector equation may be re-written in matrix form as:

$$\mathbf{X} = \mathbf{A}\mathbf{S} \tag{2}$$

in which the observations form a matrix, $\mathbf{X}$, which is modeled as a *product* of two other matrices. It is instructive to see this as a *basis model* in which $\mathbf{A}$ are the mixings of a matrix of basis responses, $\mathbf{S}$, which we may interpret as the set of unknown (i.e. latent, or hidden) sources.

Without constraint, Equation 2 is ill-posed. ICA can be hence seen as a member of a family of approaches which impose constraint into the solution space to all evaluation of the matrix-product decomposition.

Principal Component Analysis [19] avoids the problem of an ill-posed solution space by making constraints of orthogonality. ICA, on the other hand, allows solutions to Equation 2 by forcing independence between the components of the basis. If we regard the unknown sources as random variables, whose joint density is $p(\mathbf{s})$ then

this independence means that the joint density over the basis sources factorizes such that:

$$p(\mathbf{s}[t]) = \prod_{i=1}^{M} p_i(s_i[t]) \tag{3}$$

It is worth noting that independence does not imply orthogonality, indeed the latter is a considerably weaker constraint. Forcing independence is sufficient to make the problem well-posed up to an arbitrary permutation and scaling of the basis. This means that sources may be recovered using ICA but their scaling is arbitrary (and many implementations will simply normalise to a constant power) and there is, unlike PCA, no 'natural' ordering. These drawbacks are not significant in practice, however. Seeking a factorization of the source densities is equivalent to the minimization of the *mutual information* between the inferred sources. This mutual information (MI) measure may be seen as the canonical objective of all ICA approaches [1]. Direct estimation of the MI, however, is computationally impractical for most problems and ICA algorithms typically exploit either approximate measures (such as higher order statistics of the sources) or make parametric models for the sources from which MI may be obtained analytically [6,8,9]. It is this latter approach which we take in this paper.

In most cases involving data analysis, we may not be confident that the observations we make are in the noiseless limit. Incorporating an explicit noise term into the model has the advantage that it allows the model to be probabilistic (in that a generative likelihood may be formulated). In all the examples presented in this paper we utilize a noisy ICA model of the form

$$\mathbf{x}[t] = \mathbf{A}\mathbf{s}[t] + \mathbf{n}[t]. \tag{4}$$

The statistics of the noise term are detailed in section 5.3 below.

3 Brain Activity

The activity of the brain may be monitored using small scalp electrodes. The resultant signals are known as electroencephalograms, or EEGs. The clinical use of EEG is widespread; from the analysis of sleep disorders, the detection and treatment of epilepsy to analysis of cognitive function. Typical EEG recordings may be made from a large number (64 or more) electrodes, but it is more common for smaller sets to be used. Our present knowledge of neurophysiology, though, indicates that the EEG contains information from a considerably smaller set of underlying sources of activity. The form of the transfer from brain (really the cortex - the 'top' layer of brain) to scalp is a linear propagation (at least for frequencies in the EEG range, below about 40Hz) [20].

We consider the EEG propagation process as the detection, at the surface of the scalp, of cortical potentials (sub-cortical potentials are considerably weaker and contribute insignificantly to the surface EEG [20]). To further complicate matters, the effect of the layer of cerebro-spinal fluid, in particular, is to 'smear' potentials generated on the cortex (via a volume conduction effect) before observation at the scalp surface. Hence a hypothetical delta function, δ, generated on the cortex will be observed at the surface of the scalp as some smeared function x, as depicted schematically in Figure 1(a).

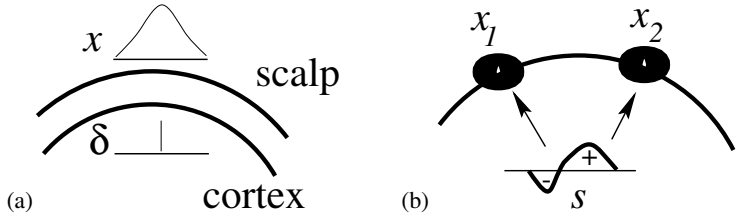


Fig. 1. (a) Volume conduction: The effect of volume conduction is to 'smear' a delta function, δ, on the cortex to some spread function x upon observation on the scalp. **(b) Simple EEG propagation model:** In this simple model, an EEG generator (a patch of cortex, for example) generates a source signal, s. This is detected at the surface of the scalp by two electrodes, $x_{1,2}$.

The form of the smearing function (i.e. specifying the functional form of x) has been discussed at considerable length in the literature relating to this topic and the interested reader is referred to [20]-[26] for more information. What concerns us in this paper is that the theoretical and experimental form of this potential is that of a spatial low-pass filter kernel (a summation of Legendre polynomials) and is *strictly non-negative*. A positive going potential from the cortex will hence be observed at the scalp as a positive going potential as well (albeit smeared), and a negative going potential will be observed similarly as negative going on the scalp, as depicted schematically in Figure 1(b). When we consider the issue of multiple scalp electrodes, then, we may form a simple model for the observed potentials. Consider a single source, $s_j[t]$, say on the cortex. Each observed EEG signal, $x_i[t]$, has a contribution from $s_j[t]$ which we define as $A_{ij}s_j[t]$ where the coupling coefficient, A_{ij}, is related to the volume conduction effect and the distance from source to electrode etc. The major point is that all these coefficients are *non-negative*.

Given a set of multiple sources and additive noise processes, $n_i[t]$, we may write a model for each observed EEG signal as

$$x_i[t] = \sum_j A_{ij}s_j[t] + n_i[t] \tag{5}$$

or, in vector-matrix notation as

$$\mathbf{x}[t] = \mathbf{A}\mathbf{s}[t] + \mathbf{n}[t] \tag{6}$$

This is, of course, the canonical form of the ICA source mixing process with one crucial constraint, the elements of $\mathbf{A}$ are *non-negative*. In the following section we review and detail our approach to inference in the ICA model and consider the impact of positivity constraints on the mixing process.

It is important at this point to note that we do not consider the 'true' underlying sources of EEG to be fully 'independent'. The name of Independent Component Analysis is somewhat misleading in this sense. Whenever we mix any set of source signals, independent or not, the resultant mixtures are *less independent* than the originals. The role of ICA (in the framework which we consider) is then to unmix by making the putative sources as independent *as possible*. This subtle, but often overlooked, differ-

ence is important as there is no guarantee that the EEG components we seek are indeed independent in the strict sense but we still can regard ICA as an appropriate model.

One further point to remember is that by constraining $\mathbf{A}$ to be positive, yet allowing the sources, $\mathbf{s}[t]$, to be unconstrained, we are constraining a *sign consistency* on the observed EEGs. In other words, for a common-reference set of electrodes, a positive (e.g.) inflection in one channel (due to an inflection in an underlying source) will be observed (albeit at differing amplitude) as a positive inflection in other channels which have significant coupling to the same source.

4 Implementation - Bayesian ICA

In this section we give an overview of the Bayesian ICA methodology used in this paper. In common with ICA in the literature, we choose a generative model to work with. This is defined via the standard noisy ICA model of Equation (4).

The noise, $\mathbf{n}[t]$, is assumed to be Gaussian[1], with zero mean and diagonal precision[2] matrix $\mathbf{R}$. The probability of observing data vector $\mathbf{x}[t]$ is then given by

$$p(\mathbf{x}[t]|\mathbf{s}[t], \mathbf{A}, \mathbf{R}) = \left|\det\left(\frac{\mathbf{R}}{2\pi}\right)\right|^{\frac{1}{2}} \exp[-E_D] \tag{7}$$

where

$$E_D = \frac{1}{2}(\mathbf{x}[t] - \mathbf{A}\mathbf{s}[t])^{\mathsf{T}}\mathbf{R}(\mathbf{x}[t] - \mathbf{A}\mathbf{s}[t]) \tag{8}$$

Since the sources $\mathbf{s}[t] = \{s_1[t], .., s_i[t], .., s_M[t]\}$ are mutually independent in the model, the distribution over $\mathbf{s}[t]$ for the t-th data point can be written as

$$p(\mathbf{s}[t]) = \prod_{i=1}^{M} p(s_i[t]) \tag{9}$$

where the product runs over the M sources.

In ICA, one attempts to uncover the hidden sources that give rise to a set of observations. In principle, this is achieved by calculating the posterior over the latent variables (sources) given the observed variables and the model, $\mathcal{M}$

$$p(\mathbf{s}[t]|\mathbf{x}[t], \mathcal{M}) = \frac{p(\mathbf{x}[t]|\mathbf{s}[t], \mathcal{M})p(\mathbf{s}[t]|\mathcal{M})}{p(\mathbf{x}[t]|\mathcal{M})} \tag{10}$$

where $p(\mathbf{s}[t]|\mathcal{M})$ is the source model and $p(\mathbf{x}[t]|\mathcal{M})$ is a normalising factor often called the marginal likelihood, or *evidence* for model $\mathcal{M}$.

[1] This could be relaxed, but we run the risk of non-Gaussian noise being interpreted as an extra source in the model.

[2] Precision is inverse (co)variance.

4.1 Source Model

The choice of a flexible and mathematically attractive (tractable) source model is crucial if a wide variety of source distributions are to be modeled; in particular, the source model should be capable of encompassing both super- and sub-Gaussian distributions (distributions with positive and negative kurtosis respectively) and complex multi-modal distributions.

One such distribution is a mixture of Gaussians (MoG). Mixture of Gaussian source models are attractive, as they offer excellent source density modelling capabilities. It is worth reviewing briefly the issue of source densities. Many ICA models, such as the 'InfoMax' models described by Bell & Sejnowski [5] have a fixed non-linearity, corresponding to a fixed source density model. Typically this take the form of a $\tanh$ non-linearity (for detailed reasoning behind this choice, the reader is referred to [5]) which gives rise to an equivalent reciprocal cosh source density as $p(s_i) = 1/\pi \cosh(s_i)$. It remains testimony to the power of the ICA methodology that such fixed source models work so well. The major problem lies in the fact that they are only able to separate sources which are heavier in the tails than a Gaussian (i.e. platykurtic sources). To enable the separation of sources of arbitary densities, flexible models are required. One such approach was developed in [9] in which the use of Generalised Exponential (GenExp) sources was formulated in which $p(s_i) \propto \exp(-\beta_i |s_i|^{r_i})$ and β_i, r_i are inferred parameters. Although the GenExp solution does give better performance compared to standard fixed source models, it is still uni-modal and less well-suited to a variety of real-world applications then a MoG model [27]. Figure 2(a) shows the histogram of values from a complex multi-modal source (an image in this example). Plot (b) shows the learned GenExp source density model and plot (c) the MoG model. We simply note that, although computationally more intensive, the MoG model offers considerably better density models and, in principle therefore, better ICA source estimates. In all the examples in this paper we use the MoG source density model. The i-th source density is hence given as

$$p(s_i) = \sum_{c=1}^{C_i} \pi_{i,c} \mathcal{N}(s_i | \mu_{i,c}, \beta_{i,c})$$

where c indexes the components in the mixture model (in Figure 2(c) there are three), $\pi_{i,c}$ are the mixing fractions and $\mu_{i,c}, \beta_{i,c}$ are the mean and precision of the c-th Gaussian distribution in the model for the i-th source.

5 Bayesian Inference and Variational Learning

The parameters of the model *could* be learnt through a maximum likelihood approach such as the Expectation-Maximisation (EM) algorithm [28,29] (see also [30] for a comprehensive derivation of the EM algorithm with regard to ICA). The resultant values can then be used to reconstruct the sources via the MoG source model. We take, arguably, a more comprehensive approach in which a full Bayesian paradigm is employed. This is detailed in this next section.

The maximum likelihood approach to learning the parameters of the model is well documented (see [30], [31], [32] for an introduction), as are the pitfalls. We choose to

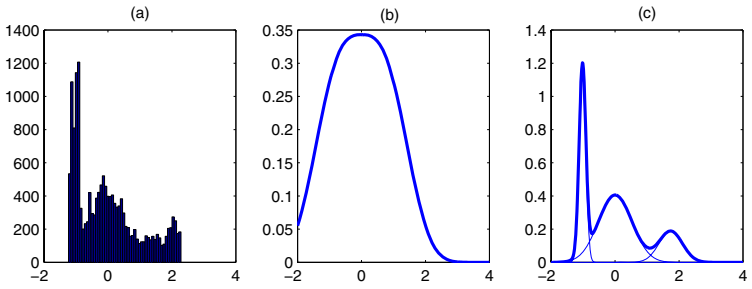


Fig. 2. Source models: (a) source histogram, (b) GenExp source model and (c) MoG source model. The latter offers a better representation of the source density.

take a fully Bayesian approach and infer the posterior distributions over all parameters and associated hyper-parameters.[3] First, we will briefly discuss the prior distributions over the model parameters and then go on to offer a brief overview of variational learning.

5.1 Prior Distributions

In Bayesian inference plausible priors must be placed over the variables of interest to reflect our degree of knowledge, or lack or it, regarding the parameters. We briefly detail the priors over the source model, the noise model and the mixing matrix in this section. More mathematical detail is available in [12,27].

Source model: Because of the source independence intrinsic to generic ICA models, it follows that the distribution over our source densities factorises. The source distribution is hence specified by the set of parameters $\boldsymbol{\theta}_i \stackrel{\text{def}}{=} \{\boldsymbol{\pi}_i, \boldsymbol{\mu}_i, \boldsymbol{\beta}_i\}$ where the index i ranges over all source models and the $\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\beta}$ are, respectively, the mixings, means and precisions of the Mixture of Gaussian (MoG) model components. The prior over the i-th source model parameters is thus taken as a product of priors over $\boldsymbol{\pi}_i, \boldsymbol{\mu}_i, \boldsymbol{\beta}_i$.

$$p(\boldsymbol{\theta}_i) = p(\boldsymbol{\pi}_i)p(\boldsymbol{\mu}_i)p(\boldsymbol{\beta}_i) \tag{11}$$

The prior over the mixing proportions, $\boldsymbol{\pi}_i$, for the i^{th} source is a symmetric Dirichlet. The prior over each mixture component mean is itself a Gaussian and the priors over the associated precisions, $\boldsymbol{\beta}_i$, are Gamma distributions.

Noise model: The prior over the observation noise precision, $\mathbf{R}$, is a product of Gamma distributions for each diagonal element, R_j.

[3] Hyper-parameters define the distributions over parameters of the model. For example, if a parameter has a normal distribution, the associated hyper-parameters are the mean and precision of that distribution.

Mixing matrix: The prior over each element of the mixing matrix, $\mathbf{A}$ is a *rectified* Gaussian[4] with precision α_i for each column. By using a rectified Gaussian we force non-negative solutions on the mixing process [11]. The resultant prior over $\mathbf{A}$ is then

$$p(\mathbf{A}) = \prod_{i=1}^{M} \prod_{j=1}^{N} \mathcal{N}^r(A_{ji}|\alpha_i). \tag{12}$$

By monitoring the evolution of the precision hyperparameters $\boldsymbol{\alpha}$, the relevance of each source may be determined (this is referred to as *Automatic Relevance Determination* and is discussed in the next section of this paper). If α_i is large, column i of $\mathbf{A}$ will be close to zero, indicating source i is irrelevant. Finally, the prior over each α_i is a Gamma distribution $p(\alpha_i) = \mathcal{G}(\alpha_i|b_{\alpha_i}, c_{\alpha_i})$. Hence,

$$p(\boldsymbol{\alpha}) = \prod_{i=1}^{M} \mathcal{G}(\alpha_i|b_{\alpha_i}, c_{\alpha_i}) \tag{13}$$

5.2 Model Complexity

One of the most important issues in ICA lies in the determination of the appropriate model complexity, i.e. the number of hypothesised sources. Why is this so important? Unlike Principle Component Analysis (PCA), ICA does *not* form an orthogonal basis set of Independent Components. The profound impact of this is that the addition of an extra source (a basis) is to change all other sources. This is in sharp contrast to the PCA case in which the addition of components does not affect the others. This means that if we have the 'wrong' number of ICs in our model the sources inferred will be erroneous. This is a serious issue, which has had scant coverage in the ICA literature.

As a trivial illustration of this effect we consider a set of six observations, created by mixing two true sources. Figure 3 shows the results for 4, 3 and 2 hypothesised sources from a set of six observation time series. In this simple case, we can probably just see 'by eye' that there are two sources, but we also note that the 'spurious' extra sources are 'sensible' and that the detailed nature of each reconstruction is dependent upon the number of sources. This problem may be dealt with in a mathematically principled manner by using 'weight-decay' priors on the rows of the mixing matrix.

Automatic Relevance Determination: (ARD) For the simplest discussion of this concept, we consider a single parameter, A say, in a model with error E. In a Bayesian learning scheme, this error is defined via two terms, the error associated with the data fit (actually the negative logarithm if the generative model) and the prior over A (again, the negative logarithm of the prior probability). Hence,

$$E \propto E_{\text{data}} - \log p(A).$$

If we take $p(A) \propto \exp(-\frac{\alpha}{2}A^2)$ i.e. a Gaussian with zero mean and precision α then the error becomes

$$E \propto E_{\text{data}} + \frac{\alpha}{2}A^2.$$

[4] A rectified Gaussian is defined as $\mathcal{N}^r(y|\alpha) = 2\mathcal{N}(y|0, \alpha)$ for $y \geq 0$ and zero for $y < 0$. The hyperparameter α is the precision, or inverse variance.

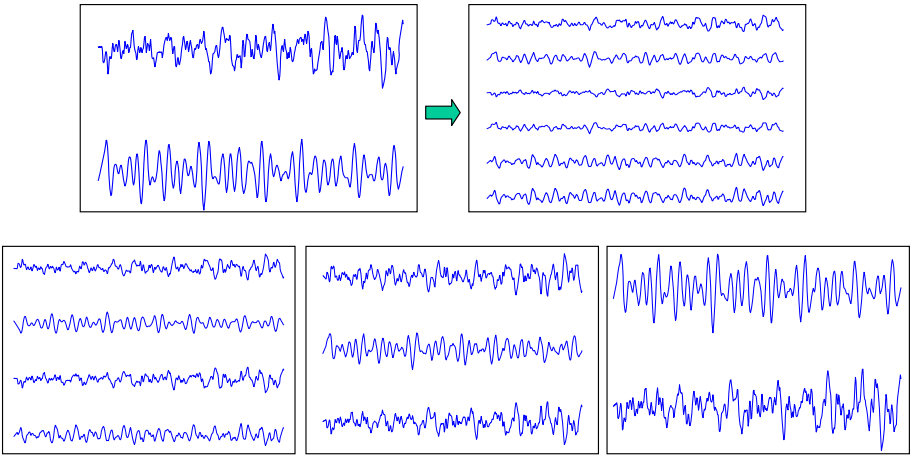


Fig. 3. Number of sources?: In this simple example two sources are mixed to a set of six observations (top sub-figures) and subsequently decomposed with ICA to 2,3 and 4 sources (lower sub-figures). Without care, it is difficult to guess the correct number of ICs.

The importance of this form of prior becomes clear when we consider the gradient

$$\frac{dE}{dA} = \frac{d}{dA}E_{\text{data}} + \alpha A.$$

Consider the case in which changes to the parameter A did not lead to significant changes in the model's performance in modelling the data. In this situation the term $\frac{d}{dA}E_{\text{data}}$ is close to zero and changes with iteration, t, to the parameter take the form,

$$\frac{dA}{dt} \propto -\frac{dE}{dA} \approx -\alpha A$$

which leads to an exponential decay of A to zero. This *weight-decay* prior is well-known in statistics and machine learning [33,34]. The key effect is that, in the case of parameters which do not significantly contribute to an explanation of the observed data via the model, the distrubution over the parameter shrinks and resultant values are close to zero. This has the very desirable effect in generalised linear models, such as ICA, of effectively removing basis components if the scheme is applied to the mixing coefficients. In the ICA case, the latter are the rows of the mixing matrix. It is worth noting that the elements of the 'unused' components will not be *exactly* zero and hence ARD offers a 'soft removal' of components. In all examples discussed in this paper, we observe that the 'relevant' components of the mixing matrix have magnitudes some fifteen orders of magnitude more than the 'irrelevant' ones making confirmation of correct model selection, where possible, straighforward.

5.3 Variational Bayesian Learning

Bayesian inference in complex models is often computationally intensive and intractable. An important and efficient tool in approximating posterior distributions is

the *variational method* (see [35] for an excellent tutorial). In particular, we take the *variational Bayes* approach detailed in [36].

The central quantity of interest in Bayesian learning is the posterior distribution $p(\boldsymbol{\Theta}|\mathbf{X})$ which fully describes our knowledge regarding all the parameters of the model, $\boldsymbol{\Theta}$, when we have observed the data $\mathbf{X}$. In non-linear or non-Gaussian models, however, the posterior is often difficult to estimate as although one may be able to provide values for the posterior for a particular $\boldsymbol{\Theta}$, the partition function, or normalization term, may involve an intractable integral. To circumvent this problem two approaches have been developed: the sampling framework and the parametric framework. In the sampling framework integration is performed via a stochastic sampling procedure such as Markov-Chain Monte-Carlo (MCMC). The latter, however, can be computationally intensive and assessment of convergence is often problematic. Alternatively, the posterior can be assumed to be of a particular parametric form. In the Laplace approximation, employed for example in the 'Evidence framework', the posterior is assumed to be Gaussian [37]. This procedure is quick but is often inaccurate. Recently an alternative parametric method has been proposed: Variational Bayes (VB) or 'Ensemble Learning'. In what follows we briefly describe the key features of VB learning.

Given a probabilistic model with parameters $\boldsymbol{\Theta}$ of the observations data $\mathbf{X}$ the log 'evidence' or log 'marginal likelihood' may be given by:

$$\log p(\mathbf{X}) = \int q(\boldsymbol{\Theta}|\mathbf{X}) \log p(\mathbf{X}) d\boldsymbol{\Theta} \tag{14}$$

where $q(\boldsymbol{\Theta}|\mathbf{X})$ is a hypothesized, or *proposal*, posterior density. This may be introduced in the above equation as we note that $q(\boldsymbol{\Theta}|\mathbf{X})$ is a density function (i.e. it integrates to unity). Hence,

$$\begin{aligned}
\log p(\mathbf{X}) &= \int q(\boldsymbol{\Theta}|\mathbf{X}) \log p(\mathbf{X}) d\boldsymbol{\Theta} \\
&= \int q(\boldsymbol{\Theta}|\mathbf{X}) \log \left(p(\mathbf{X}) \frac{p(\mathbf{X}, \boldsymbol{\Theta})}{p(\mathbf{X}, \boldsymbol{\Theta})} \right) d\boldsymbol{\Theta} \\
&= \int q(\boldsymbol{\Theta}|\mathbf{X}) \log \frac{p(\mathbf{X}, \boldsymbol{\Theta})}{p(\boldsymbol{\Theta}|\mathbf{X})} d\boldsymbol{\Theta} \\
&= \int q(\boldsymbol{\Theta}|\mathbf{X}) \left(\log p(\mathbf{X}, \boldsymbol{\Theta}) - \log p(\boldsymbol{\Theta}|\mathbf{X}) + \log q(\boldsymbol{\Theta}|\mathbf{X}) - \log q(\boldsymbol{\Theta}|\mathbf{X}) \right) d\boldsymbol{\Theta} \\
&= \int q(\boldsymbol{\Theta}|\mathbf{X}) \log \frac{p(\mathbf{X}, \boldsymbol{\Theta})}{q(\boldsymbol{\Theta}|\mathbf{X})} d\boldsymbol{\Theta} + \int q(\boldsymbol{\Theta}|\mathbf{X}) \log \frac{q(\boldsymbol{\Theta}|\mathbf{X})}{p(\boldsymbol{\Theta}|\mathbf{X})} d\boldsymbol{\Theta}
\end{aligned} \tag{15}$$

We may write the latter equation as

$$\log p(\mathbf{X}) = F(p, q) + KL_{post}(q, p) \tag{16}$$

where

$$F(p, q) \stackrel{\text{def}}{=} \int q(\boldsymbol{\Theta}|\mathbf{X}) \log \frac{p(\mathbf{X}, \boldsymbol{\Theta})}{q(\boldsymbol{\Theta}|\mathbf{X})} d\boldsymbol{\Theta} \tag{17}$$

is known as the negative variational free energy and

$$KL_{post}(q, p) \stackrel{\text{def}}{=} \int q(\boldsymbol{\Theta}|\mathbf{X}) \log \frac{q(\boldsymbol{\Theta}|\mathbf{X})}{p(\boldsymbol{\Theta}|\mathbf{X})} d\boldsymbol{\Theta} \tag{18}$$

is the KL-divergence [38] between the proposal posterior and the true posterior.

Equation 16 is the fundamental equation of the VB-framework. Importantly, because the KL-divergence is always positive [38], $F(p,q)$ provides a strict *lower bound* on the model evidence. Moreover, because the KL-divergence is zero when the two densities are the same, $F(p,q)$ will become equal to the model evidence when the approximating posterior is equal to the true posterior ie. if $q(\boldsymbol{\Theta}|\mathbf{X}) = p(\boldsymbol{\Theta}|\mathbf{X})$.

The aim of VB-learning is therefore to maximise $F(p,q)$ and so make the approximate posterior as close as possible to the true posterior. To obtain a practical learning algorithm we must also ensure that the integrals in $F(p,q)$ are tractable. One generic procedure for attaining this goal is to assume that the approximating density factorizes over groups of parameters (in physics this is known as the mean field approximation). Thus, following [39], we consider:

$$q(\boldsymbol{\Theta}|\mathbf{X}) = \prod_i q(\boldsymbol{\Theta}_i|\mathbf{X}) \tag{19}$$

where $\boldsymbol{\Theta}_i$ is the ith group of parameters (from the ith source, for example). The distributions which maximise the negative free energy can then be shown to be of the following form (see [27]), here shown for parameter group $\boldsymbol{\Theta}_i$,

$$q(\boldsymbol{\Theta}_i|\mathbf{X}) = \frac{\exp[I(\boldsymbol{\Theta}_i)]}{\int \exp[I(\boldsymbol{\Theta}_i)] d\boldsymbol{\Theta}_i}. \tag{20}$$

where

$$I(\boldsymbol{\Theta}_i) \stackrel{\text{def}}{=} \int q(\boldsymbol{\Theta}^{\setminus i}|\mathbf{X}) \log p(\mathbf{X}|\boldsymbol{\Theta}) p(\boldsymbol{\Theta}) d\boldsymbol{\Theta}^{\setminus i} \tag{21}$$

and $\boldsymbol{\Theta}^{\setminus i}$ denotes all the parameters *not* in the ith group. For models having suitable priors, the above equations are available in closed analytic form. This leads to a set of coupled update rules. Iterated application of these leads to the desired maximization. Full details of this scheme applied to the ICA models considered in this paper may be found in [10,12,18]. [5]

All the derived posteriors require solving a set of coupled hyper-parameter update equations. In practice, this is best achieved by cycling through groups of parameters until convergence. The grouping we applied in this paper is detailed in [18] and is briefly presented below:

- Mixture of Gaussians source model: means, variances and priors (mixing fractions) are all updated along with the associated hyper-parameters.
- Mixing matrix: The elements of the mixing matrix,$\mathbf{A}$ are modelled as rectified normals. Updates are made to the components of $\mathbf{A}$, the precisions, α and associated hyper-parameters.
- Noise component: The noise process, $\mathbf{n}$ is modelled as a zero-mean Gaussian with covariance $\mathbf{R}$. Updates are made to $\mathbf{R}$ and associated hyper-parameters.

Once trained, the model can be used to reconstruct hidden source signals (to within a scaling and permutation) given a data-set by calculating $\langle s_i \rangle$ under their respective posteriors.

[5] Code for all the implementations of ICA detailed in these references and in this paper may be found via www.robots.ox.ac.uk/∼parg.

6 Results

6.1 Initialisation

We set vague distributions for all prior distributions for all datasets presented here. Three component mixture of Gaussians were used for all source models. Model learning was terminated when the relative change in free energy dropped below 10^{-5}. This took between 10 and 50 iterations of the model for the examples shown in this paper. Fifty iterations took just under 1 minute of CPU time running under Matlab on a 1.4GHz processor. It is worth noting that, as a full Bayesian learning paradigm is taken, we avoid

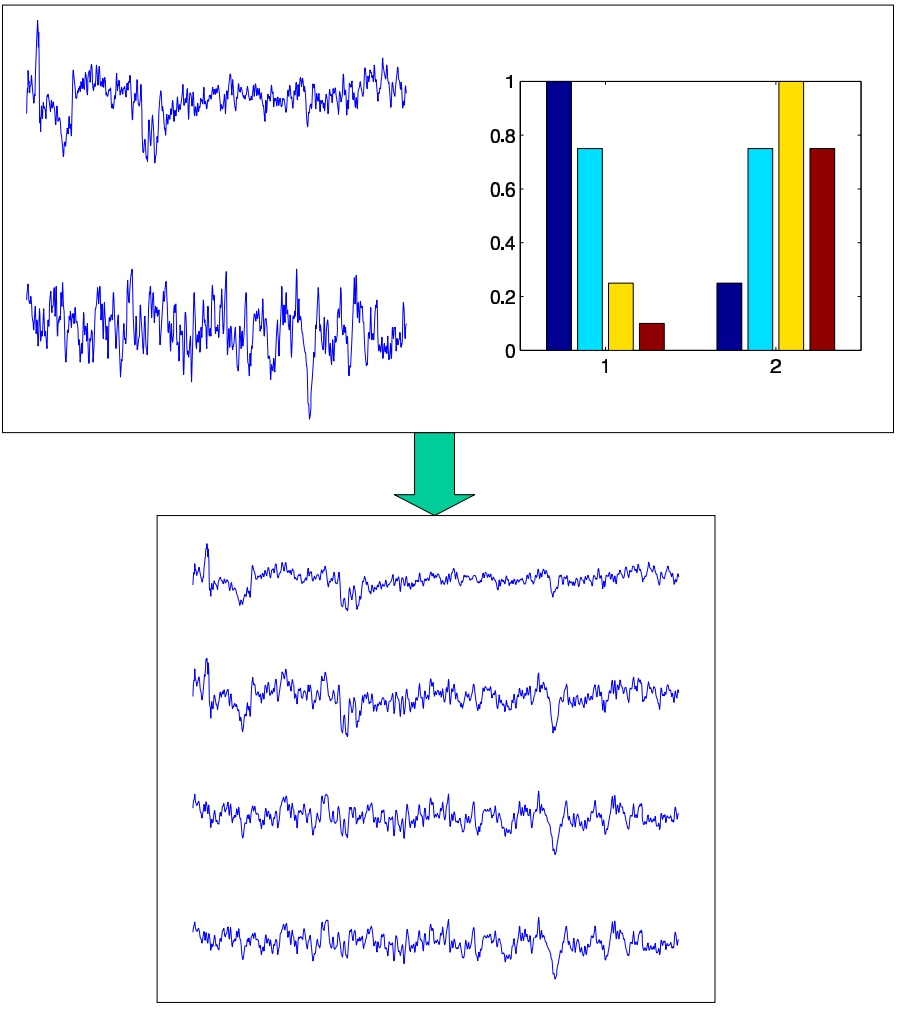


Fig. 4. Synthetic data: Two sources of 'activity' are mixed via a positive process to four observed 'EEGs'

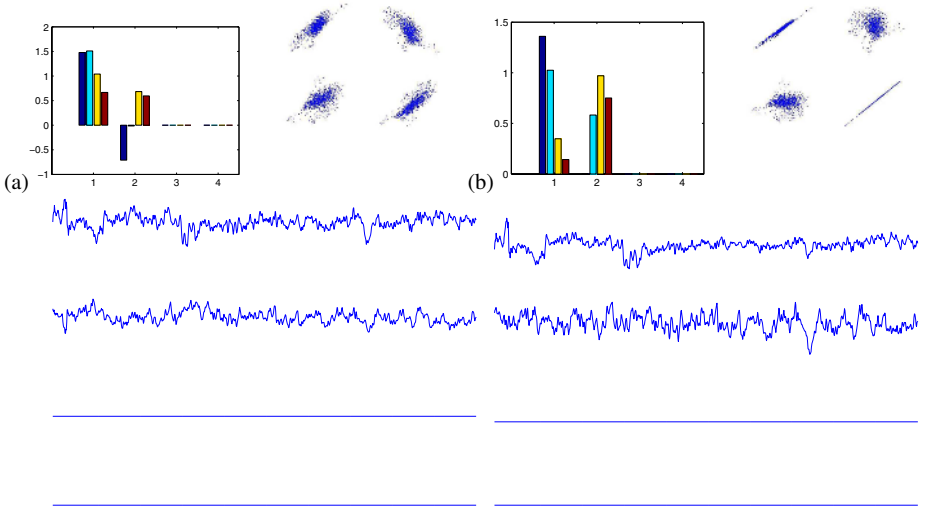


Fig. 5. Synthetic data: (a) Unconstrained ICA solution showing resultant estimated mixing matrix (top left), scatter plot of original sources against reconstructed (top right), and reconstructed sources (lowermost). Plot (b) shows the same but for positive-constrained ICA. The flat lines in the reconstructed sources are those components which have extremely low magnitude due to the ARD process.

the need for user-specified parameters in the model. The choice of hyper-parameters in the priors simply reflects our lack of knowledge about the problems ahead of time (hence 'vague') and the algorithm is not critically dependent upon the precise values of these.

6.2 Synthetic Data

As a first example, we use a set of data synthesised from a positive mixing. Figure 4 shows two synthetic 'source activities', a mixing process (to model the mixing from cortex to a set of four scalp electrodes) and the resultant 'EEGs'. Note, as discussed previously, the constraint of positivity is imposed to ensure that there is *sign consistency* assumed in the observed EEGs.

Figure 5 shows the results from (a) unconstrained ICA and (b) constrained ICA. The negative free energies for the two models strongly favour the constrained model. Note that in the latter case, not only are the sources better recovered (as evident in the scatter-plot) but also the mixing matrix is better estimated than in the unconstrained case. Note the near-zero elements in the estimated mixing matrices in both cases due to the action of ARD. In both cases the resultant mixing matrix elements were of order 10^{-15}. This gives rise to the 'flat' lines in the reconstructed source set. These can, of course, be easily removed if required but are presented here for completeness. Note that the reconstructed sources are all shown on the same scale.

One of the other key aspects of a probabilistic model for ICA lies in the inference of a *distribution* over the sources rather than just point estimates. This means, for example,

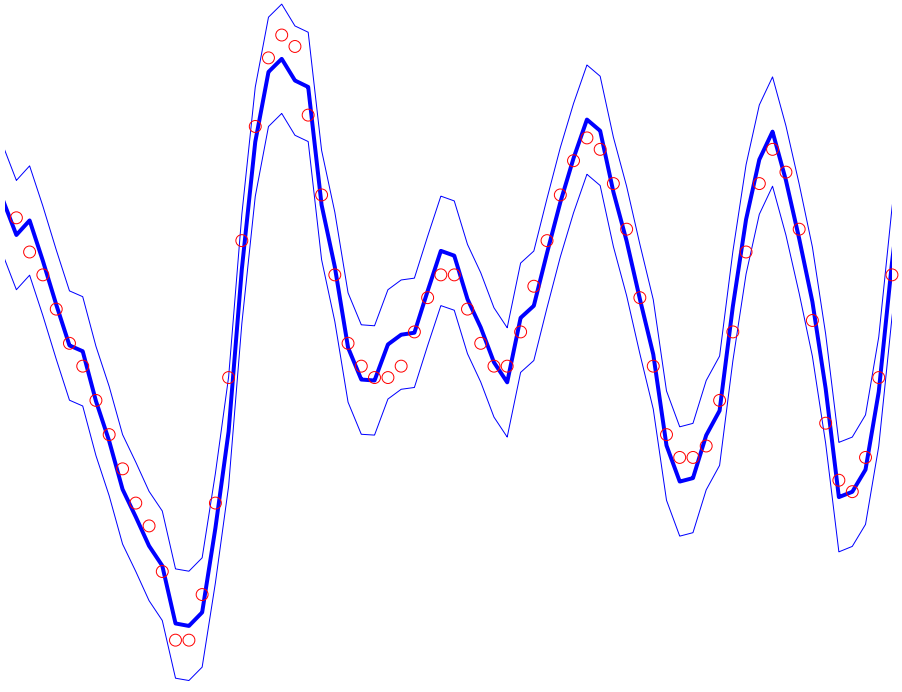


Fig. 6. Predictive distribution: Circles are the 'ground truth' source data, the thick line $\langle s[t] \rangle$ and the thin lines at $\pm\sigma$

that error bars may be obtained on a point-by-point basis with ease. Figure **??** shows a small section from the above synthetic data along with the predicted source distribution. The circles are the 'ground truth' source data, the thick line the most-probable estimated source and the thin lines at one standard deviation.

6.3 Brain-Computer Interface Data

We detail in this section results on a data set taken as part of a Brain-Computer Interface (BCI) experiment [40,41]. A total of nine, 30-minute long, EEG channels were used in this example, arranged 1cm apart in a 3×3 configuration over the left motor cortex. The signals were sampled at 128Hz to 12-bit accuracy with an anti-aliasing filter with a cut-off at 40Hz. A 7-second sample from the EEG recording is shown in Figure 7. Note the consistency of deflection in the channels, i.e. positive going events are positive going in all channels, albeit at different amplitudes.

Figure 8 shows the reconstructed source estimates (over the 30-second example section of data) for the case of no constraint on $\mathbf{A}$ (a) and with a positivity constraint (b). Note that, firstly, the recovered indepedent components are different and secondly that the effect of ARD has been different in the two cases, with four sources inferred in (a) whilst only three are inferred in (b). The negative free energies in the two cases are -6.18×10^4 for no constraints and -2.95×10^4 for the positive constraint case. This

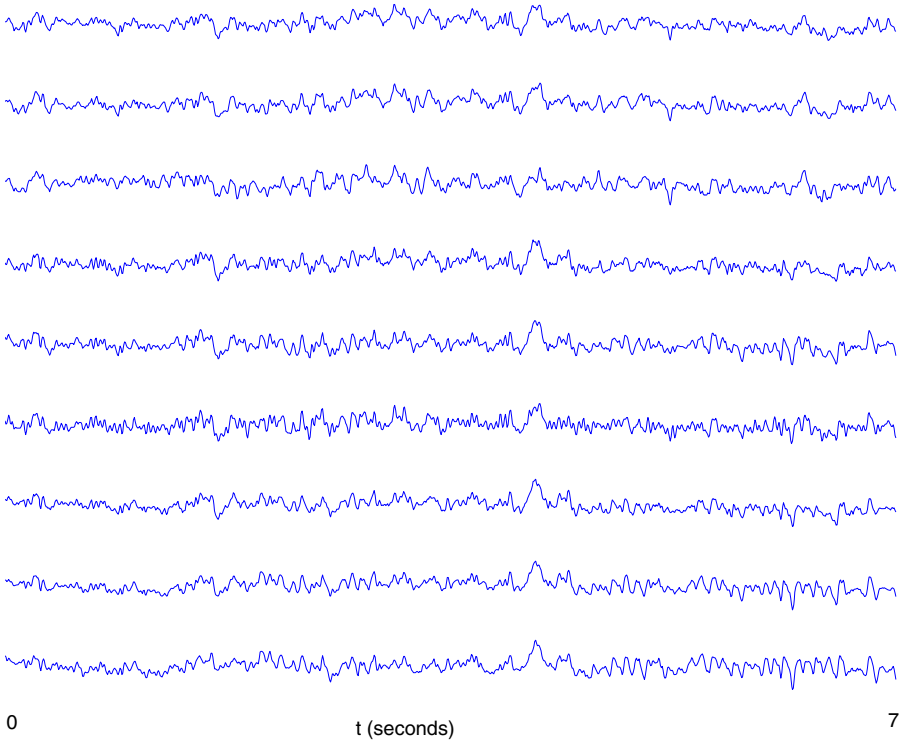


Fig. 7. BCI data: Nine channels of EEG from a square grid of electrodes over the left motor cortex. Note the consistency of events across all channels.

represents a clear hypothesis choice for the case of positive mixing especially when we consider that the free energies are on a logarithmic scale.

For completeness we show the inferred values of the mixing matrices in the two cases in Figure 9. As above, (a) shows the inferred elements of $\mathbf{A}$ for no constraints and (b) for the case of positivity. The effect of ARD in the suppression of redundant sources is clear from the rows of zero elements in $\mathbf{A}$ in both plots.

6.4 Epilepsy EEG

In this section we show results from the application of the ICA techniques discussed to eight-channel data from an epilepsy sufferer. A total of four hours of data was used, sampled as per the BCI data. Figure 10 shows a 30-second section of this data, chosen so as to include a representative 'spike and wave' epileptic seizure event.

Figure 11 shows the inferred sources from (a) un-constrained mixing model and (b) positivity-constrained mixing model. We once again note that the number of unsuppressed (non-zero) sources is very different with six sources inferred in case (a) and only three in case (b). Once again positivity has found a considerably more compact

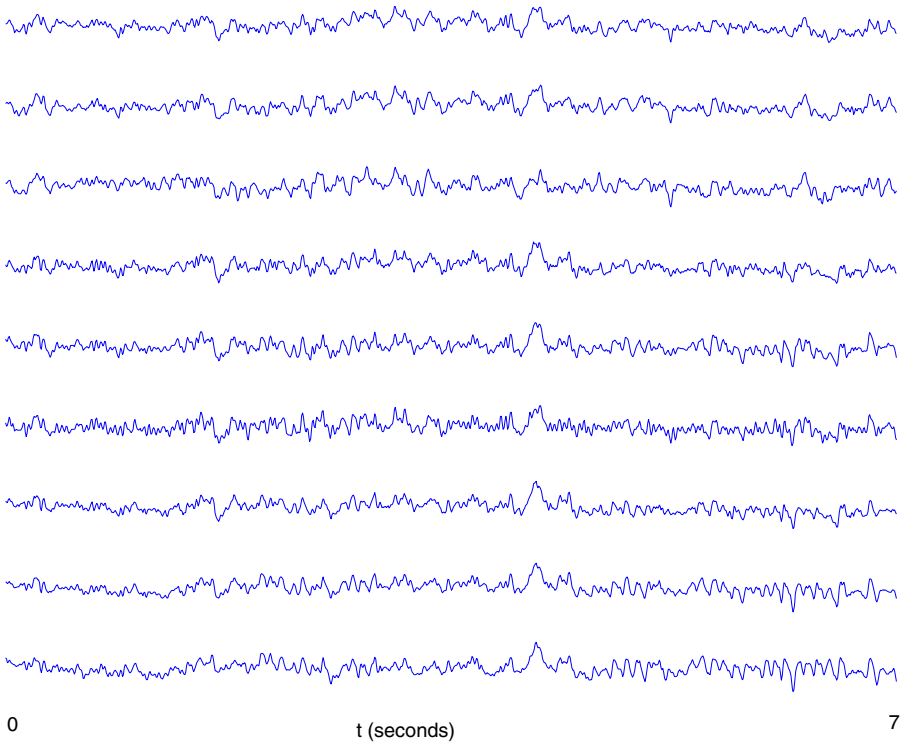


Fig. 8. BCI data, reconstructed source estimates: Plot (a) shows the reconstructed source estimates for the case when no constraint is placed on the elements of the mixing matrix. Plot (b) is for the case of positive constraints. Note, in particular, that the effect of ARD has been different in the two cases, with fewer sources being inferred in the latter case.

representation of the data and this results in an improvement in negative free energy (log data likelihood under the model) from -5.6×10^4 to -3.2×10^4.

As in the case of the BCI data, the elements of the inferred mixing matrix, $\mathbf{A}$ show a considerable difference in the un-constrained (a) and constrained (b) cases as shown in Figure 12.

6.5 Sampled EEG Data

In this section we apply the two ICA models considered in this paper, namely with and without positivity mixing constraints, to a large database of EEG. A total of 36 hours of eight-channel EEG data (sampled as per the other EEG data in this paper), was analysed. A randomly selected five-second block of data was analysed using unconstrained and constrained algorithms and the resultant negative free energies from the models stored. This process was repeated 1000 times (with replacement). The negative free energies

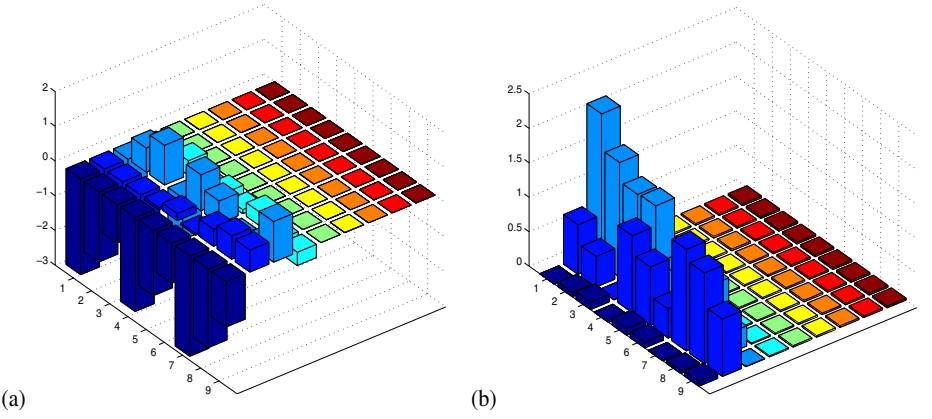


Fig. 9. BCI data, bar plots of mixing matrix elements: plot (a) shows the elements of **A** for the case of no constraints and plot (b) with positivity. Note the clear effect of ARD in both plots, to suppress redundant sources by inference of (near) zero elements in the mixing matrix.

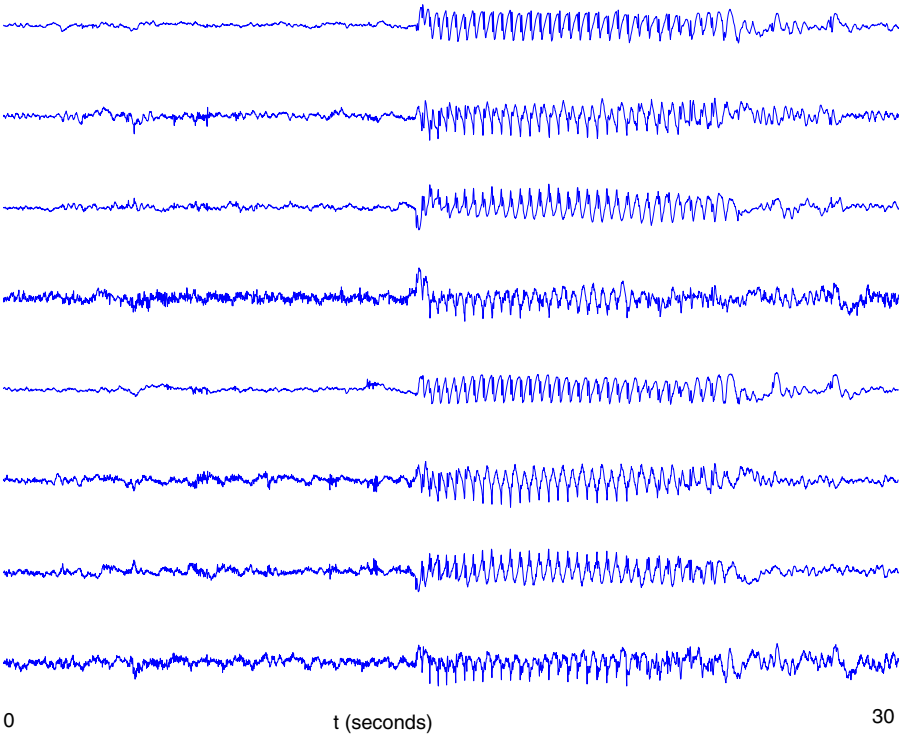


Fig. 10. Epilepsy data: A 30-second section of the eight channels of EEG in this data set

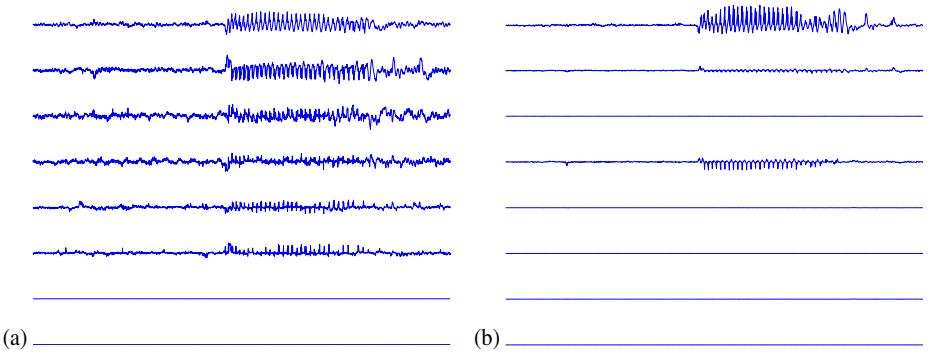


Fig. 11. Epilepsy data, reconstructed source estimates: Plot (a) is the un-constrained mixing case and plot (b) the case with positivity constraints. Note the considerable change in the number of significant sources

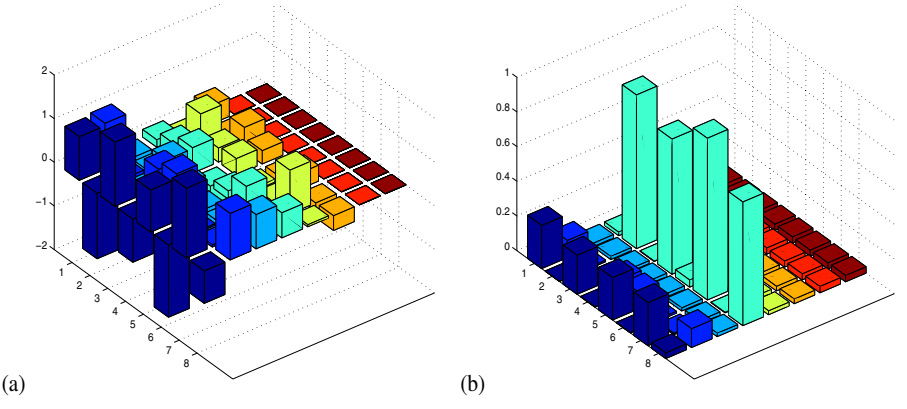


Fig. 12. Epilepsy data, bar plots of mixing matrix elements: (a) un-constrained mixing model, (b) positivity constrained mixing. Note that the number of non-zero rows has reduced from six in plot (a) to only three in plot (b).

(data log likelihoods) from this experiment are presented as a boxplot[6] in Figure 13. We note that the free energies of the two models, evaluated over a large number of randomly selected five-second sections of EEG, are highly significantly different and clearly favour a positivity constraint on the mixing model.

7 Discussion and Conclusions

We have shown in this paper that constrained ICA models may be formed under the framework of generative Bayesian modelling. The unknown parameters in the model

[6] A boxplot shows the range of the data (upper and lower extensions from the box) along with the median (line in box), upper and lower quartiles (the upper and lower edges of the box).

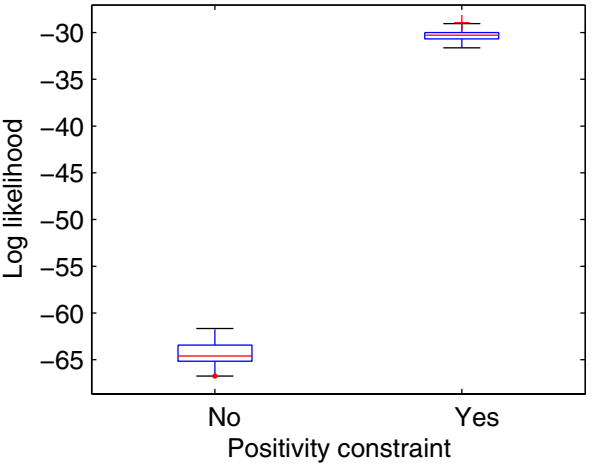


Fig. 13. Boxplot of negative free energy: The profound effect of imposing a positivity prior is shown in this Figure. The left-hand box plot shows negative free energy (model log-likelihood) without positivity constraints and the right-hand e with the constraints.

are inferred using the methodology of variational Bayes learning. The model, hence, allows for priors over all parameters. This has the benefit that, not only is 'correct' Bayesian inference performed but that the priors allow components of the model which are not supported by the data (i.e. do not explain the data) to collapse thus giving a principled manner in which to infer the number of underlying latent data 'sources'. The use of priors also allows for constraints to the model space, and in this paper we explore the use of positivity priors on the mixing process from sources to observations. We show that such positivity constraints are expected from simple considerations of the EEG propagation process and show that in practice considerably sparser models (with much higher likelihoods) are indeed found by imposing this constraint on the mixing model in ICA.

Acknowledgements

R.C. was funded by the UK Engineering and Physical Sciences Research Council for whose support we are most grateful. The authors would like to thank Will Penny, Iead Rezek, Richard Everson and Evangelos Roussos for many thought-provoking discussions. S.R. would like to thank Clare Waterstone for her enthusiastic support.

References

1. S. Roberts and R. Everson. *Independent Component Analysis: principles and practice*. Cambridge University Press, 2001.
2. *Independent Component Analysis*. John Wiley & Sons, 2001.
3. P. Comon. Independent component analysis, a new concept? *Signal Processing*, 36:287–314, 1994.

4. A Hyvärinen. Fast and Robust Fixed-Point Algorithms for Independent Component Analysis. *IEEE Transactions on Neural Networks*, 10(3):626–634, 1999.
5. A.J. Bell and T.J. Sejnowski. An information maximisation approach to blind separation and blind deconvolution. *Neural Computation*, 7(6):1129–1159, 1995.
6. D.J.C. MacKay. Maximum Likelihood and Covariant Algorithms for Independent Component Analysis. Technical report, University of Cambridge, December 1996. Available from `http://wol.ra.phy.cam.ac.uk/mackay/`.
7. J-F. Cardoso. Infomax and Maximum Likelihood for Blind Separation. *IEEE Signal Processing Letters*, 4(4):112–114, 1997.
8. S.J. Roberts. Independent Component Analysis: Source Assessment and Separation, a Bayesian Approach. *IEE Proceedings, Vision, Image and Signal Processing*, 145(3):149–154, 1998.
9. R. Everson and S. Roberts. Independent Component Analysis: A flexible non-linearity and decorrelating manifold approach. *Neural Computation*, 11(8):1957–1984, 1999.
10. R. Choudrey, W. Penny, and S. Roberts. An ensemble learning approach to Independent Component Analysis. In *Proceedings of Neural Networks for Signal Processing*, Sydney, Australia, December 2000.
11. J. Miskin and D. MacKay. *Ensemble learning for blind source separation*, chapter 8 in *Independent Component Analysis: Principles and Practice*, S. Roberts & R. Everson (Eds.). Cambridge University Press, 2001.
12. R. Choudrey and S. Roberts. Variational Mixture of Bayesian Independent Component Analysers. *Neural Computation*, 15(1):213–252, 2003.
13. S. Makeig, A.J. Bell, T.-P. Jung, and T.J. Sejnowski. Independent component analysis of electroencephalographic data. In *Advances in Neural Information Processing Systems*, volume 8, pages 145–151. MIT Press, 1996.
14. S. Makeig, T.-P. Jung, A.J. Bell, D. Ghahremani, and T.J. Sejnowski. Blind separation of auditory event-related brain responses into independent components. *Proceedings of the National Academy of Sciences*, 94:10979–84, 1997.
15. T.-P. Jung, S. Makeig, M. Westerfield, J. Townsend, E. Courchesne, and T.J. Sejnowski. Independent component analysis of single-trial event-related potentials. In *Proc. First International Conference on Independent Component Analysis and Blind Source Separation ICA'99*, pages 173–178, Aussois, France, 1999.
16. G. Wübbeler, A. Ziehe, B.-M. Mackert, K.-R. Müller, L. Trahms, and G. Curio. Independent component analysis of non-invasively recorded cortical magnetic DC-fields in humans. *IEEE Trans Biomed Eng.*, 47(5), 2000.
17. W. Penny, R. Everson, and S. Roberts. *Hidden Markov Independent Components Analysis*, chapter in Advances in Independent Components Analysis, M. Girolami (Ed). Springer, 2000.
18. S. Roberts, E. Roussos, and R. Choudrey. Hierarchy, Priors and Wavelets: Structure and Signal Modelling using ICA. *Signal Processing*, 84:283–297, 2004.
19. I.T. Jolliffe. *Principal Component Analysis*. Springer, 1986.
20. P.L. Nunez. *Electric Fields of the Brain*. Oxford University Press, Oxford, New York, 1981.
21. C.D. Gielser and G.L. Gerstein. The Surface EEG in Relation to its Sources. *Electroenceph. Clin. Neur.*, pages 927–934, 1961.
22. P. Gloor. *Neuronal Generators and the Problem of Localization in Electroencephalography : Application of Volume Conductor Theory to Electroencephalography*. *Journ. Clin. Neur.*, 2(4):327–354, 1985.
23. R.S. Hosek, A. Sances, R.W. Jodat, and S.J. Larson. Contributions of Intracerebral Currents to the EEG and Evoked Potentials. *IEEE Transactions on Biomed. Eng.*, 25(5):405–413.
24. C. Nicholson. Theoretic Analysis of Field Potentials in Anisotropic Ensembles of Neuronal Elements. *IEEE Trans. Biomed. Eng.*, 20(4):279–288, 1973.

25. M.J. Peters. On the Magnetic Field and the Electric Potential generated by Bioelectric Sources in an Anisotropic Volume Conductor. *Med. & Biol. Eng. & Comput.*, 26:617–623, 1988.
26. S. Rush and D.A. Drsicoll. Current Distribution in the Brain from Surface Electrodes. *Anesth. Analg.*, 47:717–723, 1968.
27. R. Choudrey and S. Roberts. Flexible Bayesian Independent Component Analysis for Blind Source Separation. In *Proceedings of ICA-2001*, San Diego, December 2001.
28. A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *J. Roy. Stat. Soc.*, 39(1):1–38, 1977.
29. R. M. Neal and G. E. Hinton. A view of the EM algorithm that justifies incremental, sparse and other variants. In Jordan [42], pages 355–368.
30. H. Attias. Independent Factor Analysis. *Neural Computation*, 11:803–851, 1999.
31. B. Pearlmutter and L. Parra. A Context-Sensitive Generalization of ICA. In *1996 International Conference on Neural Information Processing.*, 1996.
32. J.-F. Cardoso. Blind signal separation: statistical principles. *IEEE Transactions on Signal Processing*, 9(10):2009–2025, 1998.
33. D. Husmeier, W.D. Penny, and S.J. Roberts. An empirical evaluation of Bayesian sampling with hybrid Monte Carlo for training neural network classifiers. *Neural Networks*, 12:677–705, 1999.
34. R.M. Neal. *Bayesian learning for neural networks.* Lecture notes in statistics. Springer, Berlin, 1996.
35. M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. In Jordan [42].
36. T. S. Jaakkola and M. I. Jordan. Bayesian parameter estimation via variational methods. *Statistics and Computing*, 10:25–37, 2000.
37. J.J.K. O' Ruanaidth and W.J. Fitzgerald. *Numerical Bayesian Methods Applied to Signal Processing.* Springer, 1996.
38. *Information Theory*. John Wiley, 1991.
39. H. Attias. Inferring Parameters and Structure of Latent Variable Models by Variational Bayes. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, 1999.
40. Z.A. Keirn and J.I. Aunon. A new mode of communication between man and his surroundings. *IEEE Transactions of Biomedical Engineering*, 37(12):1209–1214, 1990.
41. S.J. Roberts and W.D. Penny. Real-time Brain Computer Interfacing: a preliminary study using Bayesian learning. *Medical and Biological Engineering & Computing*, 38(1):56–61, 2000.
42. M. I. Jordan, editor. *Learning in Graphical Models*. MIT Press, Cambridge, MA, 1999.

Ensemble Algorithms for Feature Selection

Jeremy D. Rogers and Steve R. Gunn

Image, Speech and Intelligent Systems Research Group
School of Electronics and Computer Science
University of Southampton, U.K.

Abstract. Many feature selection algorithms are limited in that they attempt to identify relevant feature subsets by examining the features individually. This paper introduces a technique for determining feature relevance using the average information gain achieved during the construction of decision tree ensembles. The technique introduces a node complexity measure and a statistical method for updating the feature sampling distribution based upon confidence intervals to control the rate of convergence. A feature selection threshold is also derived, using the expected performance of an irrelevant feature. Experiments demonstrate the potential of these methods and illustrate the need for both feature weighting and selection.

1 Introduction

Ensemble algorithms have achieved success in machine learning by combining multiple weak learners to form one strong learner. The Adaboost algorithm, [1] and the Bagging algorithm [2] are two examples of this. Much research has been conducted into understanding the mechanics of these methods and of finding ways to improve them. The explanations centre around the idea of diversity in the base learners, which enable good exploration of possible hypotheses. A good generalisation ability of an ensemble can be obtained by constructing accurate base learners that make their mistakes in different parts of the training data. Many improvements to ensemble algorithms exploit this idea by attempting to increase the diversity. The Random Forest technique, [3], is one such algorithm, which adopts the randomisation principle of [4] to achieve an increase in diversity. The base learners in this algorithm are CART based trees [5]. These trees usually perform a search through a large number of possible binary splits for every feature in order to find the optimal split for each node. The criterion for each split is the measure of information gain, which is the reduction in entropy that results from the split. The Random Forest algorithm uses Bagging to generate a training set for each tree. Diversity is injected into the ensemble by choosing a feature randomly at each node in the tree construction and optimising the split over a set of possible split values along that feature. Due to the random exploration of features, Random Forest lends itself to feature selection well.

Traditional approaches to feature selection have typically taken two forms, the Filter method which attempts to select the optimal feature subset by

J. Winkler, N. Lawrence, M. Niranjan (Eds.): Machine Learning Workshop, LNAI 3635, pp. 180–198, 2005.

analysing the structure of the data and the Wrapper method which performs a search through possible feature subsets and uses the learning algorithm to test the suitability of each. Both of these methods attempt to eliminate irrelevant features and reduce the probability of discovering false relationships in the data. Also, by reducing the dimensionality of the data, the computational requirement imposed upon the learning algorithm is reduced. The ability of each feature subset is partially dependent upon the learning algorithm used. The Wrapper method uses the learning algorithm to evaluate each feature subset and consequently, has the advantage of incorporating this bias. However, when using high dimensional data, the process of searching through possible feature subsets can be computationally expensive.

The selection of features is not the only application that is available once the relevance of each feature is known. Feature weighting algorithms are also used, where all of the features are included in the learning process. In this case each feature is relied upon to a different extent, which is determined by its level of importance. Although this approach can improve generalisation, it does not create a reduction in dimensionality. In random forest the feature weighting concept can be realised by applying these levels of feature importance to the feature sampling distribution from which the features are randomly chosen.

When feature selection is applied to ensemble learning, the criterion for selection is somewhat different. The identification of relevant features is still an important issue, but the selected features also need to promote diversity in the constructed base learners. [6] proposed an algorithm which employed the random subspace method, [7] to generate the learners which were evaluated in terms of their accuracy and diversity.

The measure of feature importance adopted here is the average information gain achieved during tree construction and a node complexity measure is introduced to improve the accuracy of this measure. These levels of importance are applied to the feature sampling distribution in a parallel scheme where the rate at which the feature sampling distribution is updated is controlled using a confidence interval method. This is also compared to a fast two stage method where the feature sampling distribution is set before forest construction.

Feature weighting is shown to be successful here, but if the data contains a significant number of irrelevant features, a selection scheme will improve performance. An approximate threshold for feature selection is derived here, which attempts to predict the expected average information gain achieved by an irrelevant feature. This is compared to a correlation based feature selection algorithm, CFS [8] and it is shown that both feature weighting and selection should be exploited to optimise the generalisation.

Section Two introduces the concept of feature relevance and the measure adopted in this paper. The node complexity measure is also introduced here. Section Three examines the methods of updating the feature sampling distribution and introduces the parallel algorithm. The feature selection threshold is described in section Four and the experimental results are shown in section Five. Section Six discusses the results and gives some directions for further work.

2 Identifying Feature Relevance

2.1 Defining Feature Relevance

In order to identify relevant features it is important to first consider a suitable definition for feature relevance. The goal is to identify the features that carry as much information concerning the target as possible and eliminate information that is repeated in multiple features. As discussed by [9], an obvious definition is that a feature X_i is relevant if there exist values x_i and y assigned to X_i and the target Y respectively, such that,

$$P\left(Y = y | X_i = x_i\right) \neq P\left(Y = y\right) \tag{1}$$

Intuitively this makes sense, as knowledge about the value of a relevant feature should affect the prediction of the target. However, it is not always the case that a relevant feature taken by itself provides valuable information about the target. This is illustrated using the XOR example.

Example 1. If the target, Y is given by the exclusive OR of the binary features, X_1 and X_2, then Y is fully described by the features and they are both relevant.

$$Y = X_1 \oplus X_2$$

However, if each feature assumes the values of 1 or 0 with equal probability, then the above definition shows them both to be irrelevant. This is because knowing the value of one of the features gives no information about the target without knowing the value of the other feature.

It is claimed by [9] that two different types of relevance are required for successful feature selection, strong relevance and weak relevance. Strong relevance is used to describe a feature, which carries information about the target that is not repeated in any other feature and is defined in the following manner, If S_i is the subset of all of the features apart from X_i and s_i is a value assignment to those features, then X_i is strongly relevant if there exists some x_i, y and s_i such that,

$$P\left(Y = y | X_i = x_i, S_i = s_i\right) \neq P\left(Y = y | S_i = s_i\right) \tag{2}$$

Removing a strongly relevant feature from the set will result in a loss of information about the target. Weak relevance is used to describe features that carry information about the target, but which is repeated in other features. Unlike strongly relevant features, if a weakly relevant feature is removed, no information about the target is lost. X_i is weakly relevant if it is not strongly relevant and there exists some subset, S_i' of S_i and values y, x_i and s_i' such that,

$$P\left(Y = y | X_i = x_i, S_i' = s_i'\right) \neq P\left(Y = y | S_i' = s_i'\right) \tag{3}$$

Some feature selection schemes examine the correlation between features in an attempt to discover this weak relevance such as [10], [11] and the CFS algorithm

of [8]. This type of method highlights information that is shared between features but does not discriminate between information that is useful for describing the target and random correlations in the data.

The definition of weak relevance can also be viewed as a definition of conditional independence, where the target Y is conditionally independent of the feature X_i given a subset S'_i, if there exists no value assignments to these variables which satisfy the inequality. This idea can be extended to the concept of identifying Markov Blankets [12]. If S'_i is some subset of the features, such that $X_i \notin S'_i$, and S_i is the subset of remaining features, that excludes S'_i and X_i, then S'_i is a Markov Blanket for X_i if there are no value assignments to the variables such that,

$$P\left(S_i = s_i, Y = y | X_i = x_i, S'_i = s'_i\right) \neq P\left(S_i = s_i, Y = y | S'_i = s'_i\right) \tag{4}$$

Therefore, S'_i is a Markov Blanket for X_i if it subsumes all of the predictive information about the target and the remaining features that is contained within X_i. One of the important properties of Markov Blankets is that features can be removed recursively. Koller and Sahami [12], showed that if features are only removed when a corresponding Markov Blanket is discovered, a feature that has been eliminated will not become relevant again as more features are removed.

The estimation of Markov Blankets can be difficult and algorithms, such as [12] and [11] use measures of correlation to find approximations of Markov Blankets. The measure of correlation varies between the standard linear correlation, which is limited to only identifying linear correlations in the data, and measures from information theory such as information gain, conditional entropy and symmetrical uncertainty. Roobaert et al. [10], use information gain as a measure of feature importance by calculating the reduction in entropy of the target caused by separating the data with the given feature. The information gain on the target Y, caused by the feature X_i is given by.

$$IG\left(Y|X_i\right) = H\left(Y\right) - H\left(Y|X_i\right), \tag{5}$$

where $H\left(Y\right)$ is the entropy of the class and $H\left(Y|X\right)$ is the conditional entropy. Care must be taken with this approach as information gain will favour features with more partitions .

This method evaluates the importance of a feature solely by examining the correlation to the target and consequently is equivalent to the definition of feature relevance given by Equation 1, which has already been shown to give incorrect results in certain situations.

The definitions of strongly and weakly relevant features are sufficient to describe the usefulness of the features concerned because they are based on the analysis of feature subsets, rather than individual features or pairs of features. There is always a possibility that redundancy and interaction can occur amongst larger feature subsets. This is one of the reasons for the accuracy of Wrapper methods but is also implemented by methods using random feature subset combination such as the Parcel algorithm of [13]. This algorithm regards classifiers

utilising diverse feature subsets as useful if they extend the convex hull over the ROC space.

When employing decision trees such as CART or Random Forest, estimates for the correlations between the features and the target are generated as part of the construction process in the form of information gain values. [14] uses these measures of feature importance to increase performance of the learning algorithm. Although these measures appear to be simpler forms of information gain, there are some benefits to using this method over standard information gain. Each terminal node in a decision tree can be viewed as a learner that has been trained on the features that were used in the path from the root. Consequently, the information gain values are not simply measures of the individual feature performance but measures of the ability of the feature in a variety of possible feature subsets. It is clear from the XOR example that if the data was split using one of the features and then split again using the other feature, that the second split would reveal the relevance of the feature. Therefore, there is some allowance for relationships between the features with this method. The advantage of this, as a technique for feature selection, over the random subspace method [7], is that multiple feature subsets can be evaluated within an individual learner, thus yielding a more efficient subset exploration. Also, because the decision trees are used for both the feature selection and learning processes, the bias of the learning algorithm is incorporated.

The problem with using the average information gain achieved by each feature, is that some nodes in the tree are easier to split than others. The number of ways a node can be split is determined by its composition and when smaller nodes are split, the measure of information gain is clearly more unreliable. The following introduces a measure to weight each value of information gain according to its reliability.

2.2 A Node Complexity Measure

This measure attempts to assign a value of reliability to a node by examining its composition and calculating the information associated with the splitting of such a node. For every split, the data is projected along a single feature. The assumptions made here are that once the data is projected into the one dimensional space, no data points lie on top of one another and that during the split optimisation procedure, all possible splits are found. Figure 1 shows the possible split positions for one node once the data is projected into the one dimensional space.

The problem is now a matter of considering how many possible arrangements of the data are possible. For binary classification problems, n is the number of examples contained in the node and i is the number of positive examples. The number of possible arrangements is given by the combinatorial function,

$$C_i^n = \frac{n!}{i!\,(n-i)!} \tag{6}$$

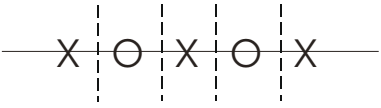


Fig. 1. Illustration of possible splits in one dimensional space of a node consisting of 3 data points from one class and 2 from the other

However, some of these arrangements are merely reflections of each other and will, therefore, result in the same optimised information gain. For example, a node containing only two examples, one of each class, will have two possible arrangements. The optimal split value is the same in both cases and this node can yield only one information gain value. Using the assumptions stated above, this example would be split perfectly by all features and would result in a maximum information gain value. Therefore, this illustrates the need for effective weighting of the nodes.

Not all of the arrangements have a reflected twin because some arrangements are symmetrical about their centre. These symmetrical arrangements shall be referred to as unique and their frequency designated by A_{u}. Their counterparts shall be referred to as non-unique and their frequency can be written $C_i^n - A_{\mathrm{u}}$. A_{u} can be calculated by considering the arrangements of half of the data and then taking the reflection to form the other half. This technique is dependant upon whether the values of n and i are odd or even and the corresponding functions are given in Table 1.

Table 1. Number of unique arrangements for node

n	i	A_{u}
EVEN	**EVEN**	$C_{\frac{i}{2}}^{\frac{n}{2}}$
ODD	**ODD**	$C_{\frac{i-1}{2}}^{\frac{n-1}{2}}$
ODD	**EVEN**	$C_{\frac{i}{2}}^{\frac{n-1}{2}}$
EVEN	**ODD**	0

The probability of a random occurrence of a particular unique arrangement, U_{x} is simply the probability of any particular arrangement,

$$P\left(U_{\mathrm{x}}\right) = \frac{1}{C_i^n} \tag{7}$$

As non-unique arrangements have two possible configurations, their corresponding probability, N_{x} is,

$$P\left(N_{\mathrm{x}}\right) = \frac{2}{C_i^n} \tag{8}$$

Assuming that the arrangements are random, the node complexity measure, NC, which is the information associated with the split of node l is,

$$NC\left(l\right) = -\frac{A_{\mathrm{u}}}{C_i^n}\log_2 P\left(U_{\mathrm{x}}\right) - \frac{C_i^n - A_{\mathrm{u}}}{C_i^n}\log_2 P\left(N_{\mathrm{x}}\right), \tag{9}$$

which can be simplified to,

$$NC\left(l\right) = \log_2 C_i^n - \left(1 - \frac{A_{\mathrm{u}}}{C_i^n}\right) \tag{10}$$

This is a suitable weight for calculating the average information gain because it represents the node complexity and therefore, how useful it is in identifying the predictive power of the feature.

3 The Feature Sampling Distribution

The measures of feature importance can be used to select the most relevant features but another application is to include all of the features in the learning process but weight their relevance to the problem according to their measure of importance. Random Forest can be adapted quite easily to achieve this. The standard Random Forest method chooses a feature at each split randomly from the set of all possible features. This feature sampling distribution is typically uniform but can be altered to incorporate the learned feature importance. By applying this technique, features that are deemed to be more important are chosen with a greater probability. CART trees consider all features at each stage of construction and choose the feature that provides the highest information gain. Altering the feature sampling distribution can be viewed as increasing the similarity of the randomly created trees to the ideal CART tree. If a feature selection algorithm is applied to Random Forest, then the class of possible trees that can be built is restricted and the diversity is reduced. By altering the feature sampling distribution, a trade off is introduced between increasing the strength of the base learners and maintaining the diversity of the ensemble. The goal is then to maximise the generalisation performance by optimising the feature sampling distribution in terms of these factors.

The alteration of the feature sampling distribution can be achieved in two ways. A two-stage method can be adopted, where an evaluation of the feature importance is conducted first and then applied to the construction of a Random Forest. Another approach is to combine the evaluation stage and the construction stage in a parallel scheme. As each tree in the forest is constructed, it can be used to evaluate the features and update the feature sampling distribution accordingly. The two-stage approach has the advantage of developing a reliable and accurate estimate of the ideal feature sampling distribution from which to build the forest. The parallel approach would be faster but has the problem of instability during the initial stages of the algorithm. When the forest is still small, there is very little information about the features from which to update

the sampling distribution. Initial overweighting of some features may create a sampling distribution that is far from ideal and the algorithm may not be able to recover from this as more trees are added. An implementation of the parallel method by [14] uses the measure of information gain as the feature importance metric. The weights of the features are updated according to,

$$w(X_i, m) = C \cdot I(X_i, 0) + \sum_{j=1}^{m} w(X_i, j), \tag{11}$$

where $w(X_i, m)$ is the weight assigned to feature i after construction of the m^{th} tree. $I(X_i, 0)$ is taken as the impurity of the whole data and C is a parameter which is used to control the rate at which the feature sampling distribution changes. By increasing the value of C the rate is decreased and the problem of initial overweighting is overcome. However, if C is too high, the sampling distribution will not change significantly and a forest very close to a standard Random Forest will be produced. Therefore, there is a need for tuning of the C parameter, which can typically be achieved using cross validation on the training data but the advantage of the small computational requirement is lost.

3.1 A Stable Parallel Method Using Confidence Intervals

A method of avoiding the cross validation stage would certainly be beneficial to the performance of the algorithm but a way of estimating the optimal convergence rate of the sampling distribution is required. The method introduced here, is to calculate a confidence interval for the estimate of expected information gain for each feature. Effectively, by observing the information gain values one is sampling from a distribution, which is assumed here to be normal. What is then required is the ability to approximate the probable distance between the mean of this normal distribution and the observed average information gain. Although the mean and variance of the true distribution are unknown, this can be accomplished by using the pivotal quantity method.

Given the sample mean (observed average information gain), $\overline{IG}$, the sample variance, S^2, the sample size, m and the true mean of the distribution μ. The pivotal quantity is,

$$\frac{\overline{IG} - \mu}{S/\sqrt{m}}, \tag{12}$$

and has a Student's t distribution with $m - 1$ degrees of freedom. A confidence interval can then be constructed within the distribution of the pivotal quantity.

$$P\left[q_1 < \frac{\overline{IG} - \mu}{S/\sqrt{m}} < q_2\right] = \gamma, \tag{13}$$

which gives the bound,

$$\overline{IG} - \frac{q_2 S}{\sqrt{m}} < \mu < \overline{IG} - \frac{q_1 S}{\sqrt{m}} \tag{14}$$

The process then consists of taking the observed information gains for each feature, calculating the sample mean and sample variance and deciding what level of confidence to use. A value of 0.95 for γ is typical. As the Student's t distribution is symmetrical, the optimal boundary values will be when $q_1 = -q_2$. These can be calculated from the value of γ by using an inverse Student's t distribution and then used to give the confidence interval around the sample mean.

If the sample mean is calculated using the weighted method then the sample variance must be weighted accordingly. Also, the sample size m must be re-examined, as a definition is required for a unit observation. A sensible value should be close to the information associated with the split of a node, averaged over all nodes in the tree. However, this is not known before the construction of the forest and must remain constant throughout.

These confidence intervals can then be used to update the feature sampling distribution by choosing values for each feature that lie within each confidence interval that yield the most uniform distribution. Here, the average information gain for each feature is viewed as assuming a value within a range of possible values, which are determined by the corresponding confidence interval. These average information gains can then be normalised and applied directly to set the feature sampling distribution, but their values must first be chosen such that they remain similar to each other and within their respective ranges. One simple method for achieving this, is to find the midpoint between the maximum lower bound and minimum upper bound of all of the confidence intervals. The value for each feature is then chosen to be as close to this value as possible without falling outside of the corresponding confidence interval.

This method will only update the feature sampling distribution when it has a confidence equal to γ. As more trees are added to the forest, the confidence in each estimate increases and the confidence intervals become smaller. Consequently, the feature sampling distribution becomes less uniform and closer to the ideal.

The confidence interval construction requires the calculation of an inverse Student's t distribution and the mean and variance of information gain for each feature. The experiments in this paper update the confidence intervals after the construction of every tree, in order to utilise the information concerning the features as soon as it is available. However, the computational load can be reduced, if desired, by updating the confidence intervals after a larger number of trees have been constructed. The cost of this is that some of the trees will be constructed using a feature sampling distribution that has not been created from all of the information that is available.

4 A Feature Selection Threshold

It is conjectured that the average information gain during the construction of decision trees is a measure of feature relevance. As previously discussed, it tests the feature on different areas of the input space and consequently accounts for the

different relationships between features. If these measures of feature importance are applied to learning algorithms which are based on decision trees, it also contains the bias of the learning algorithm. However, the worst performance of any given feature for the splitting of any given node, is that no reduction in entropy is possible and an information gain of zero is achieved. This means that the average information gain for any feature is the mean of a non-negative sample. The problem that arises from this, is that a feature which is completely irrelevant will produce some reductions in entropy purely by chance. Therefore, a non-zero feature importance value will be produced. This problem is particularly detrimental to performance when there are a relatively large number of irrelevant features and these values are used to update the feature sampling distribution. This is because, the probability of sampling any of the irrelevant features is the sum of all of their individual probabilities and although these may be small, the total can easily become significant if there are many. To overcome this problem, a feature selection threshold is introduced here, which approximates the expected information gain that is achieved by an irrelevant feature, given the size of the node being split.

Assuming that the task is binary classification and the data is projected onto a single feature, a node of size n, containing i positive examples has C_i^n possible arrangements. If the feature is irrelevant then these arrangements occur with equal probability. By constructing all of the possible arrangements and finding the maximum information gain, the expected value is calculated for various node constitutions and the outcome is shown in Figure 2.

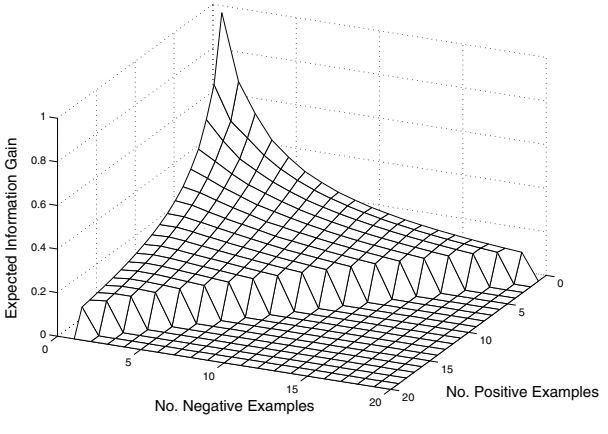


Fig. 2. Expected information gain for nodes containing various numbers of positive and negative examples

Due to the huge computational cost of evaluating the expected information gain in this manner, it is not a feasible method for feature selection, however, it can easily be approximated. For a fixed node size, the maximum expected

information gain appears to be when there are equal numbers of positive and negative examples and the minimum occurs when the ratio is most unbalanced. Therefore, the minimum expected information gain for a node of fixed size n, occurs when it contains only one example of one class, $i = 1$ or $i = n - 1$. This can be calculated in the following manner.

The information gain is the difference between the parent entropy and the combined child entropy and as the parent entropy for any given composition is fixed, only the combined child entropy needs to be considered. The case used here is that there is only one positive example, $i = 1$, and the optimal split leaves this example in the left node of size n_{l}. The right node then contains only negative examples and will have an entropy of zero. The combined child entropy CE is then,

$$CE = -\frac{n_{\text{l}}}{n}\left[\frac{1}{n_{\text{l}}}\log_2\frac{1}{n_{\text{l}}} + \frac{n_{\text{l}}-1}{n_{\text{l}}}\log_2\frac{n_{\text{l}}-1}{n_{\text{l}}}\right] \tag{15}$$

$$= -\frac{1}{n}\left[(n_{\text{l}}-1)\log_2(n_{\text{l}}-1) - n_{\text{l}}\log_2 n_{\text{l}}\right] \tag{16}$$

Differentiating by n_{l} then gives,

$$\frac{\partial}{\partial n_{\text{l}}}[CE] = \frac{1}{n}\left[\log_2 n_{\text{l}} - \log_2(n_{\text{l}}-1)\right] \tag{17}$$

For the case when n_{l} is not equal to 1 and consequently, must be a positive value of a least 2, the entropy is always increasing with n_{l}. Therefore, the optimal split is obtained when n_{l} is minimal. For a parent node of size n, the single positive example can assume only one of the possible n positions. If n is taken to be even, then by symmetry only $\frac{n}{2}$ of the arrangements need to be considered. The expected child entropy can then be written,

$$E[CE] = -\frac{2}{n}\sum_{n_{\text{l}}=1}^{\frac{n}{2}}\frac{n_{\text{l}}}{n}\left[\frac{1}{n_{\text{l}}}\log_2\frac{1}{n_{\text{l}}} + \frac{n_{\text{l}}-1}{n_{\text{l}}}\log_2\frac{n_{\text{l}}-1}{n_{\text{l}}}\right] \tag{18}$$

$$= -\frac{2}{n^2}\log_2\left[\prod_{n_{\text{l}}=1}^{n/2}\frac{(n_{\text{l}}-1)^{n_{\text{l}}-1}}{n_{\text{l}}^{n_{\text{l}}}}\right] \tag{19}$$

$$= \frac{1}{n}\log_2 n - \frac{1}{n} \tag{20}$$

Re-introducing the parent entropy, the expected information gain for a node of size n with only one example of one class, $E(IG_{\text{L}})$ can be written,

$$E[IG_{\text{L}}] = \frac{1}{n} - \frac{n-1}{n}\log_2\frac{n-1}{n} \tag{21}$$

The expected information gain for the case when there are equal numbers of each class cannot be calculated as easily. By examining the data that was generated for the construction of Figure 2 and plotting the expected information

gain for the case when the classes are equal on a logarithmic scale, it is seen that this quantity can be approximated by the following expression,

$$E\left[IG_{\mathrm{U}}\right] = \left(\frac{n}{2}\right)^{-0.82} \tag{22}$$

These two quantities can be viewed as upper and lower bounds on the expected information gain that is achieved by splitting a node of size n and are shown in Figure 3. The mid-point between these two bounds represents a reasonable estimate of the expected information gain of an irrelevant feature and can be applied as a feature selection threshold.

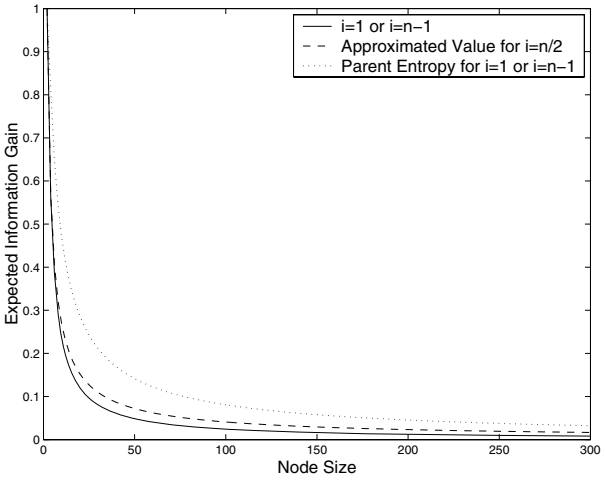


Fig. 3. Bounds on the expected information gain for varying node size. The parent entropy is also plotted for the case when $i = 1$ or $i = n - 1$ as this represents the maximum achievable information gain for this case.

5 Experiments

5.1 Datasets

The propeties of the data sets used in these experiments are shown in Table 2. The Wisconsin Breast Cancer (WBC), Pima Diabetes, Sonar, Ionosphere and Votes are available from the UCI Repository [15].

Simple is an artificial dataset consisting of 9 features and 300 examples. The output is generated according to the function,

$$Y = X_1^2 + 2X_2 \tag{23}$$

The remaining seven features are redundant and consequently this data set should benefit significantly from feature selection algorithms. It is important

Table 2. Data Set Properties

Data Set	No. Examples	No. Features	No. Relevant Features
WBC	683	9	?
Pima	768	8	?
Sonar	208	60	?
Ionosphere	351	34	?
Votes	435	16	?
Friedman	200	10	5
Simple	300	9	2

to note that as the input values to the function generator are randomly chosen between 0 and 1, feature 2 carries more predictive information than feature 1.

The Friedman dataset, [16] is another artificial dataset data set that is designed for testing feature selection algorithms. It is generated according to the following formula and has been thresholded in order to convert it into a binary classification problem. A threshold value of 14 was chosen to yield a reasonably balanced data set.

$$Y = 10\sin(\pi X_1 X_2) + 20\left(X_3 - \frac{1}{2}\right)^2 + 10X_4 + 5X_5 + N(0, 1.0) \tag{24}$$

5.2 The Node Complexity Measure

The Simple dataset is used to generate 5000 trees and the information gain values for all of the features are recorded. The values are discretised into intervals of size 0.01 so that each feature has a set of bins. As each node is split, the algorithm increments the bin corresponding to the feature being used and the information gain value obtained. As a comparison, one method increments the bins by a single unit, the other method increments by the measure of node complexity, $I(l)$. Incrementing by this value shows the effect of weighting the information gains in this manner. The results for three of the features are shown in Figure 4. The middle example is feature two, which carries the most information about the target. The left example is the next most important feature and the example on the right is a redundant feature.

It is particularly interesting to note the spikes that occur when unit weighting is used. At first glance, they appear to simply be noise but closer inspection reveals that they occur in the same places for all three features. The extreme right hand spike is the information gain that is achieved by the perfect split of a node made up from half of each class. This can occur when a node containing two examples, one from each class, is split. The spike immediately to the left of the maximum one can occur when a node containing two examples of one class and one of the other is split perfectly. These smaller nodes are much easier to split and can be split perfectly by features which carry very little information about the target. This is illustrated by the eradication of the spikes in the lower plots, where the sampling of the information gains is weighted according to

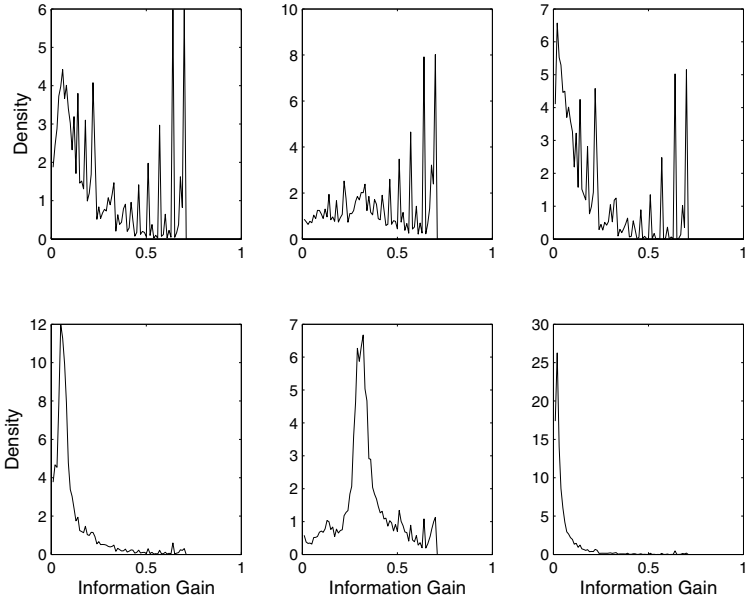


Fig. 4. Observed density functions of information gain for three features from the Simple dataset. Observed density using unit weighting (top) and observed density using node complexity weighting (bottom)

.

the node complexity. This result clearly demonstrates the ability of the node complexity measure to improve the estimate of feature importance by analysing the reliability of each sample.

5.3 Updating the Feature Sampling Distribution

The parallel method is employed to form confidence intervals on the estimates of feature importance and the feature sampling distribution is updated after every tree. The assumption that the information gain values are normally distributed is an approximation as the values are bounded on [0,1]. However, the shape of the distribution approximates normal if the node complexity weighting is used.

The initial feature sampling distribution is uniform and the algorithm keeps the most uniform distribution that is within the confidence interval of every feature. As the measures of information gain are weighted, a value for a unit of weight is required. This value should represent the information of the average split in a tree built on the dataset concerned. This value is approximated using a fraction of the node complexity of the entire data.

100 trials are conducted, using 90% of the data for training and 10% for testing. On each of the trials the rate of decrease of average confidence interval size is recorded and the result averaged over all of the trials. This represents

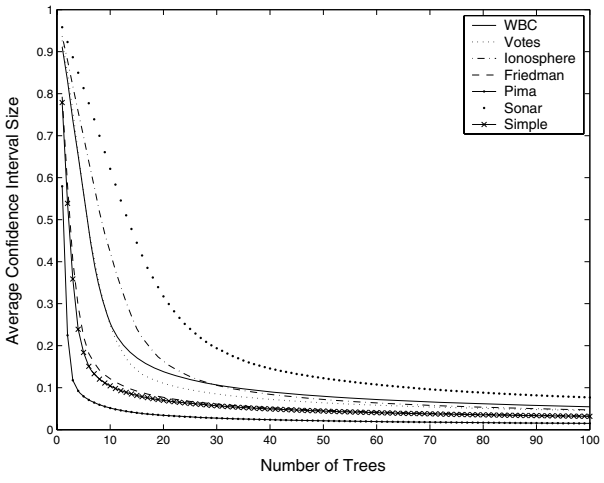


Fig. 5. Convergence rates for the feature sampling distribution. This shows how the average confidence interval size becomes smaller as more trees are added to the forest.

the rate of convergence towards the final feature sampling distribution and the results are shown in Figure 5.

The convergence rates vary with the size of the trees constructed and with the dimensionality of the data. The Pima data set has few features and produces large trees so the features are picked more often within each tree. In contrast, the Sonar data set has 60 features and produces smaller trees. Therefore, the convergence rates can still vary significantly, despite the incorporation of the prior knowledge.

A two-stage method is also applied where a single CART tree is built on the training data before construction of the forest to produce estimates of the feature importance. The average information gain is calculated using the measure of node complexity as before. However, by using a CART tree, each split supplies an information gain value for every feature, regardless of whether or not it is used. The forest is then constructed using the resultant fixed feature sampling distribution.

These methods are also compared to the CFS algorithm of [8], which selects a subset of features that have high correlation with the class and low correlation with each other. The selected features are then used to construct a Random Forest using a uniform feature sampling distribution.

Table 3 shows the error rates for standard Random Forest (RF), the CFS algorithm (CFS), weighted sampling using confidence interval method (CI WS RF) and the two-stage method using CART for evaluation (Two-stage CART).

Both methods of updating the feature sampling distribution improve the accuracy for some data sets. This improvement is most noticeable for the artificial data sets, which contain a number of irrelevant features. The confidence interval method does not significantly reduce the accuracy for any data set tested here,

Table 3. Test errors showing the improvement that three feature relevance identification techniques give to Random Forest construction. The values in brackets are the corresponding variances of test error over the 100 trials.

Data Set	RF	CFS	CI WS RF	Two-stage CART
WBC	0.0226(0.0003)	0.0235(0.0002)	0.0259(0.0003)	0.0226(0.0003)
Sonar	0.1657(0.0079)	0.2271(0.0066)	0.1462(0.0061)	0.1710(0.0069)
Votes	0.0650(0.0014)	0.0398(0.0007)	0.0493(0.0014)	0.0432(0.0008)
Pima	0.2343(0.0019)	0.2523(0.0024)	0.2394(0.0020)	0.2474(0.0018)
Ionosphere	0.0725(0.0017)	0.0650(0.0014)	0.0681(0.0018)	0.0661(0.0011)
Friedman	0.1865(0.0060)	0.1685(0.0055)	0.1690(0.0051)	0.1490(0.0052)
Simple	0.0937(0.0028)	0.1653(0.0044)	0.0450(0.0011)	0.0270(0.0009)

suggesting that the problem of initial over weighting of the features has been avoided. The two-stage CART method is shown to work well here, although there is no control over the reliability of this method and may encounter problems with certain data sets. Both methods compare favourably to the CFS algorithm, as CFS eliminates relevant features for some of the data sets, and consequently degrades the accuracy significantly.

5.4 Feature Selection Thresholding

To view the suitability of the measures of expected information gain as feature selection thresholds, 100 trees are constructed on the Simple dataset and the average information gain for each feature was recorded. The measures of expected information gain for irrelevant features are also calculated. Figure 6 shows that the seven irrelevant features are within the bounds that an irrelevant feature is expected to be in and the two relevant features are shown to be more important.

To approximate the expected information gain for a node of size n with any given composition, the mid-point between the two measures is used. Again, the data is split into 90% for training and 10% for testing. 100 trees are constructed on the training data for feature evaluation. During this, the average information gain is recorded and the approximate expected information gain is calculated. A further 100 trees are then constructed for classification of the test data. This process is repeated for 100 trials and the results averaged. This experiment is performed three times, the first experiment simply applies the recorded average information gain values to the feature sampling distribution (WS) and the second uses the expected information gain to select the relevant features but leaves the sampling distribution uniform (FS). The third experiment combines both methods by selecting the relevant features and altering the sampling distribution of the remaining features (WS & FS). Again, these results are compared to the CFS algorithm. The error rates for these experiments are shown in Table 4.

The results for the artificial datasets show an improvement when using feature selection, which is not surprising as it is known that they contain a significant number of irrelevant features. The Votes dataset shows the benefits of

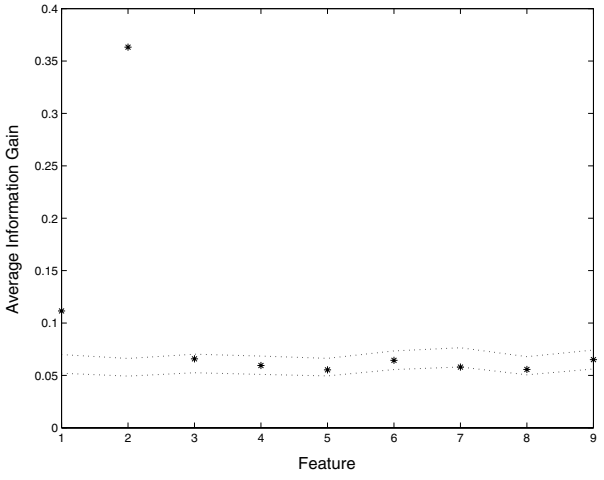


Fig. 6. The measures of feature importance for each feature in the Simple dataset and the approximate bounds for the expected values of irrelevant features. Features 1 and 2 are relevant, the remaining features are irrelevant.

Table 4. Comparison of CFS to three methods of applying the observed feature importances

Data Set	CFS	WS	FS	WS & FS
WBC	0.0235(0.0002)	0.0249(0.0003)	0.0245(0.0003)	0.0249(0.0003)
Sonar	0.2271(0.0066)	0.1757(0.0091)	0.1629(0.0060)	0.1643(0.0060)
Votes	0.0398(0.0007)	0.0464(0.0007)	0.0650(0.0013)	0.0439(0.0007)
Pima	0.2523(0.0024)	0.2312(0.0026)	0.2492(0.0021)	0.2486(0.0021)
Ionosphere	0.0650(0.0014)	0.0683(0.0017)	0.0747(0.0018)	0.0653(0.0016)
Friedman	0.1685(0.0055)	0.1555(0.0050)	0.1420(0.0060)	0.1370(0.0050)
Simple	0.1653(0.0044)	0.0393(0.0014)	0.0283(0.0009)	0.0303(0.0011)

feature weighting over the FS algorithm, as the error that is obtained from using feature selection only, is noticeably greater than when weighting is used. A problem with our feature selection approach is clearly encountered with the Pima dataset. This is most probably a consequence of the inaccuracy in the expected information gain value and the possible variance in the recorded information gain. The Pima dataset contains many features which are relevant, but whose relevance is very small. As a result of this, the recorded information gains for these relevant features are very close to the threshold and can easily fall below it. CFS also removes relevant features with this data set and consequently performs poorly. Using this threshold for average information gain as a feature selection algorithm, performs significantly better than CFS for some data.

6 Discussion

Random Forest is an ensemble algorithm, which improves the generalisation ability of weak learners by aggregation. The algorithm works well if the base learners from which the ensemble is comprised have a good generalisation ability and are diverse. The performance can be improved using ensemble feature selection, which chooses features that are not only predictive of the target and uncorrelated to each other but also promote diversity between the constructed hypotheses.

The average information gain achieved by each feature is shown to be a very reliable measure of feature importance if treated correctly. It is more than simply a measure of correlation as it tests the feature within different feature subsets and to an extent, accounts for interactions between the features. It also has a very small computational requirement, as the calculation is a product of forest construction. A method of weighting this average using a measure of node complexity is introduced and is shown to improve the accuracy considerably. By examining the distribution of the information gain, it appears reasonably normal.

Including all of the features and altering their sampling probabilities allows exploration of the trade-off between improving the accuracy of the base learners and maintaining the diversity within the ensemble. This can be performed by either using a parallel method or a two stage method. Parallel methods can suffer if the sampling distribution is updated too quickly. It is shown that by constructing confidence intervals on the estimates of feature importance, the rate of convergence can be controlled and the stability maintained. However, the rate of convergence is still dependent upon the dimensionality of the data and the resultant tree sizes.

A fast two stage method was introduced using a single CART tree to set the feature sampling distribution prior to forest construction. This method achieved surprisingly good results, although there are no bounds on the reliability of such a method. A logical next step would be to combine this method with a parallel one, where the CART tree could be used to initialise the feature sampling distribution. It could also provide prior knowledge concerning the average tree size, node complexity and number of features used. This would be useful for constructing the confidence intervals and controlling the convergence rates.

The convergence rate could also be controlled by altering the level of confidence used. Further work is required to find ways of identifying the optimal convergence rate in terms of the size of the constructed forest.

A threshold for feature selection was introduced here that approximates the performance of an irrelevant feature and performed well. It is clear that there are benefits to both feature selection and feature weighting algorithms. Irrelevant features degrade performance and need to be completely removed. The accuracy can be improved further by weighting the relevant features in order to reflect their relative importance. It has been shown here that both methods can be exploited to improve generalisation. A way of improving the measure of feature importance by identifying redundancies in the data should be investigated.

References

1. Freund, Y., Schapire, R.: A short introduction to boosting. Journal of Japanese Society for Artificial Intelligence **14** (1999) 771–780
2. Breiman, L.: Bagging predictors. Machine Learning **24** (1996) 123–140
3. Breiman, L.: Random forests. Machine Learning **45** (2001) 5–32
4. Dietterich, T.: An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. Machine Learning **40** (2000) 139–157
5. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification And Regression Trees. Wadsworth (1984)
6. Opitz, D.: Feature selection for ensembles. In: 16th National Conference on Artificial Intelligence, AAAI (1999) 379–384
7. Ho, T.: Nearest neighbours in random subspaces. In: Advances in Pattern Recognition. Volume 1451 of Lecture Notes in Computer Science., Springer (1998) 640–648
8. Hall, M.: Correlation-based feature selection for discrete and numeric class machine learning. In: 17th International Conference on Machine Learning. (2000) 359–366
9. John, G., Kohavi, R., Pfleger, K.: Irrelevant features and the subset selection problem. In Cohen, W., Hirsh, H., eds.: Machine Learning, Morgan Kaufmann (1994) 121–129
10. Roobaert, D., Karakoulas, G., Chawla, N.: Information gain, correlation and support vector machines. In Guyon, I., Gunn, S., Nikravesh, M., Zadeh, L., eds.: Feature Extraction, Foundations and Applications, Springer (2005) In Press.
11. Yu, L., Liu, H.: Feature selection for high-dimensional data: A fast correlation-based filter solution. In: Machine Learning, AAAI (2003) 856–863
12. Koller, D., Sahami, M.: Toward optimal feature selection. In: International Conference on Machine Learning. (1996) 284–292
13. Scott, M., Niranjan, M., Prager, R.: Parcel: feature subset selection in variable cost domains. Technical report, Cambridge University Engineering Department (1998)
14. Borisov, A., Eruhimov, V., Tuv, E.: Tree-based ensembles with dynamic soft feature selection. In Guyon, I., Gunn, S., Nikravesh, M., Zadeh, L., eds.: Feature Extraction, Foundations and Applications, Springer (2005) In Press.
15. Blake, C., Merz, C.: UCI repository of machine learning databases (1998)
16. Friedman, J.: Multivariate adaptive regression splines. The Annals of Statistics **19** (1991) 1–141

Can Gaussian Process Regression Be Made Robust Against Model Mismatch?

Peter Sollich

Department of Mathematics, King's College London,
Strand, London WC2R 2LS, U.K
peter.sollich@kcl.ac.uk

Abstract. Learning curves for Gaussian process (GP) regression can be strongly affected by a mismatch between the 'student' model and the 'teacher' (true data generation process), exhibiting e.g. multiple overfitting maxima and logarithmically slow learning. I investigate whether GPs can be made robust against such effects by adapting student model hyperparameters to maximize the evidence (data likelihood). An approximation for the average evidence is derived and used to predict the optimal hyperparameter values and the resulting generalization error. For large input space dimension, where the approximation becomes exact, Bayes-optimal performance is obtained at the evidence maximum, but the actual hyperparameters (e.g. the noise level) do not necessarily reflect the properties of the teacher. Also, the theoretically achievable evidence maximum cannot always be reached with the chosen set of hyperparameters, and maximizing the evidence in such cases can actually make generalization performance worse rather than better. In lower-dimensional learning scenarios, the theory predicts—in excellent qualitative and good quantitative accord with simulations—that evidence maximization eliminates logarithmically slow learning and recovers the optimal scaling of the decrease of generalization error with training set size.

1 Introduction

Gaussian processes (GPs) are by now a popular alternative to feedforward networks for regression, see e.g. [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]. They make prior assumptions about the problem to be learned very transparent, and—even though they are non-parametric models—inference is straightforward. Much work has been done to understand the learning behaviour of GPs as encoded in the learning curve, i.e. the average generalization performance for a given number of training examples [5, 7, 8, 9, 10, 12, 13]. This has mostly focused on the case where the 'student' model exactly matches the true 'teacher' generating the data. In practice, such a match is unlikely. In [11] I showed that much richer behaviour then results, with learning curves that can exhibit multiple overfitting maxima, or decay logarithmically slowly if the teacher is less smooth than the student assumes. An intriguing open question was whether these adverse effects of model mismatch can be avoided by adapting the student model during learning. This

J. Winkler, N. Lawrence, M. Niranjan (Eds.): Machine Learning Workshop, LNAI 3635, pp. 199–210, 2005.

is the issue I address in the present paper, focusing on the adaptation of model (hyper-)parameters by maximization of the data likelihood or evidence.

In its simplest form, the regression problem is this: We are trying to learn a function θ_* which maps inputs x (real-valued vectors) to (real-valued scalar) outputs $\theta_*(x)$. A set of training data D consists of n input-output pairs (x^l, y^l); the training outputs y^l may differ from the 'clean' teacher outputs $\theta_*(x^l)$ due to corruption by noise. Given a test input x, we are then asked to come up with a prediction $\hat{\theta}(x)$, plus error bar, for the corresponding output $\theta(x)$. In a Bayesian setting, one does this by specifying a prior $P(\theta)$ over hypothesis functions and a likelihood $P(D|\theta)$ with which each θ could have generated the training data; from these the posterior distribution $P(\theta|D) \propto P(D|\theta)P(\theta)$ can be deduced. For a GP, the prior is defined directly over input-output functions θ. Any θ is uniquely determined by its output values $\theta(x)$ for all x from the input domain, and for a GP, these are assumed to have a joint Gaussian distribution (hence the name). The means are usually set to zero so that the distribution is fully specified by the *covariance function* $\langle \theta(x)\theta(x') \rangle = C(x, x')$. The latter transparently encodes prior assumptions about the function to be learned. Smoothness, for example, is controlled by the behaviour of $C(x, x')$ for $x' \to x$: The Ornstein-Uhlenbeck (OU) covariance function $C(x, x') = a \exp(-|x - x'|/l)$ produces very rough (non-differentiable) functions, while functions sampled from the radial basis function (RBF) prior with $C(x, x') = a \exp[-|x - x'|^2/(2l^2)]$ are infinitely often differentiable. Here l is a length scale parameter, corresponding directly to the distance in input space over which significant variation in the function values is expected, while a determines the prior variance.

A summary of inference with GPs is as follows (for details see e.g. [14, 15]). The student assumes that outputs y are generated from the 'clean' values of a hypothesis function $\theta(x)$ by adding Gaussian noise of x-independent variance σ^2. The joint distribution of a set of training outputs $\{y^l\}$ and the function values $\theta(x)$ is then also Gaussian, with covariances given (under the student model) by

$$\langle y^l y^m \rangle = C(x^l, x^m) + \sigma^2 \delta_{lm} = (\mathbf{K})_{lm}, \qquad \langle y^l \theta(x) \rangle = C(x^l, x) = (\mathbf{k}(x))_l \quad (1)$$

Here I have defined an $n \times n$ matrix $\mathbf{K}$ and an x-dependent n-component vector $\mathbf{k}(x)$. The posterior distribution $P(\theta|D)$ is obtained by conditioning on the $\{y^l\}$; it is again Gaussian and has mean and variance

$$\langle \theta(x) \rangle_{\theta|D} \equiv \hat{\theta}(x) = \mathbf{k}(x)^{\mathrm{T}} \mathbf{K}^{-1} \mathbf{y} \quad (2)$$

$$\langle [\theta(x) - \hat{\theta}(x)]^2 \rangle_{\theta|D} = C(x, x) - \mathbf{k}(x)^{\mathrm{T}} \mathbf{K}^{-1} \mathbf{k}(x) \quad (3)$$

From the student's point of view, this solves the inference problem: the best prediction for $\theta(x)$ on the basis of the data D is $\hat{\theta}(x)$, with a (squared) error bar given by (3).

The squared deviation between the prediction and the teacher is $[\hat{\theta}(x) - \theta_*(x)]^2$; the average generalization error (which, as a function of n, defines the learning curve) is obtained by averaging this over the posterior distribution of teachers, all datasets, and the test input x:

$$\epsilon = \langle \langle \langle [\hat{\theta}(x) - \theta_*(x)]^2 \rangle_{\theta_*|D} \rangle_D \rangle_x \quad (4)$$

Of course the student does not know the true posterior of the teacher; to estimate ϵ, she must assume that it is identical to the student posterior, giving

$$\hat{\epsilon} = \langle\langle\langle[\hat{\theta}(x) - \theta(x)]^2\rangle_{\theta|D}\rangle_D\rangle_x \tag{5}$$

This generalization error estimate $\hat{\epsilon}$ coincides with the true error ϵ if the student model matches the true teacher model and then gives the Bayes error, i.e. the best achievable average generalization performance for the given teacher.

The *evidence* or data likelihood is $P(D) = \int d\theta P(D|\theta)P(\theta)$, i.e. the average of the likelihood $P(D|\theta) = \prod_{l=1}^{n}(2\pi\sigma^2)^{-1/2}\exp[-(y^l - \theta(x^l))^2/(2\sigma^2)]$ over the prior. Since the prior over the $\theta(x^l)$ is a zero mean Gaussian with covariance matrix $C(x^l, x^m)$, the integral can be done analytically and one finds

$$E \equiv \frac{1}{n}\ln P(D) = -\frac{1}{2}\ln(2\pi) - \frac{1}{2n}\mathbf{y}^{\mathrm{T}}\mathbf{K}^{-1}\mathbf{y} - \frac{1}{2n}\ln|\mathbf{K}| \tag{6}$$

The (normalized log-) evidence E depends, through $\mathbf{K}$, on all student model hyperparameters, i.e. σ^2 and any parameters specifying the covariance function. I will analyse the model selection algorithm which chooses these parameters, for each data set D, by maximizing E. For one particular hyperparameter the maximum can in fact be found analytically: if we write $C(x, x') = a\tilde{C}(x, x')$ and $\sigma^2 = a\tilde{\sigma}^2$, then the second term in (6) scales as $1/a$ and the third one gives the a-dependent contribution $(1/2)\ln a$; maximizing over a gives $a = n^{-1}\mathbf{y}^{\mathrm{T}}\tilde{\mathbf{K}}^{-1}\mathbf{y}$ and

$$\max_a E = -\frac{1}{2}\ln(2\pi/n) - \frac{1}{2} - \frac{1}{2}\ln(\mathbf{y}^{\mathrm{T}}\tilde{\mathbf{K}}^{-1}\mathbf{y}) - \frac{1}{2n}\ln|\tilde{\mathbf{K}}|$$

Note that the value of a does not affect the student's predictions (2), but only scales the error bars (3).

2 Calculating the Evidence

A theoretical analysis of the average generalization performance obtained by maximizing the evidence for each data set D is difficult because the optimal hyperparameter values fluctuate with D. However, a good approximation—at least for not too small n—can be obtained by neglecting these fluctuations, and considering the hyperparameter values that maximize the average $\bar{E}$ of the evidence over all data sets D of given size n produced by the teacher. To perform the average, I assume in what follows that the teacher is also a GP, but with a possibly different covariance function $C_*(x, x')$ and noise level σ_*^2. For fixed training inputs, the average of $y^l y^m$ is then $(\mathbf{K}_*)_{lm} = C_*(x^l, x^m) + \sigma_*^2\delta_{lm}$, and inserting into (6) gives

$$\bar{E} = -\frac{1}{2}\ln(2\pi\sigma^2) - \frac{1}{2n}\langle\mathrm{tr}\,\mathbf{K}_*\mathbf{K}^{-1}\rangle - \frac{1}{2n}\langle\ln|\sigma^{-2}\mathbf{K}|\rangle \tag{7}$$

where the remaining averages are over the distribution of all possible sets of training inputs. To tackle these, it is convenient to decompose (using Mercer's

theorem) the covariance function into its eigenfunctions $\phi_i(x)$ and eigenvalues Λ_i, defined w.r.t. the input distribution so that $\langle C(x,x')\phi_i(x')\rangle_{x'} = \Lambda_i\phi_i(x)$ with the corresponding normalization $\langle\phi_i(x)\phi_j(x)\rangle_x = \delta_{ij}$. Then

$$C(x,x') = \sum_{i=1}^{\infty} \Lambda_i\phi_i(x)\phi_i(x'), \quad \text{and similarly } C_*(x,x') = \sum_{i=1}^{\infty} \Lambda_i^*\phi_i(x)\phi_i(x') \quad (8)$$

For simplicity I assume here that the student and teacher covariance functions have the *same* eigenfunctions (but different eigenvalues). This is not as restrictive as it may seem; several examples are given below.

Introducing the diagonal eigenvalue matrix $(\mathbf{\Lambda})_{ij} = \Lambda_i\delta_{ij}$ and the 'design matrix' $(\mathbf{\Phi})_{li} = \phi_i(x^l)$, one now has $\mathbf{K} = \sigma^2 + \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^{\mathrm{T}}$, and similarly for $\mathbf{K}_*$. In the second term of (7) we need $\mathrm{tr}\,\mathbf{K}_*\mathbf{K}^{-1}$; the Woodbury formula gives the required inverse as $\mathbf{K}^{-1} = \sigma^{-2}[\mathbf{I} - \sigma^{-2}\mathbf{\Phi}\mathcal{G}\mathbf{\Phi}^{\mathrm{T}}]$, where $\mathcal{G} = (\mathbf{\Lambda}^{-1} + \sigma^{-2}\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi})^{-1}$. A little algebra then yields

$$\mathrm{tr}\,\mathbf{K}_*\mathbf{K}^{-1} = -\sigma_*^2\sigma^{-2}\mathrm{tr}\,(\mathbf{I} - \mathbf{\Lambda}^{-1}\mathcal{G}) + \mathrm{tr}\,\mathbf{\Lambda}_*\mathbf{\Lambda}^{-1}(\mathbf{I} - \mathbf{\Lambda}^{-1}\mathcal{G}) + n\sigma_*^2\sigma^{-2} \quad (9)$$

and the training inputs appear only via the matrix $\mathcal{G}$. A similar reduction is possible for the third term of (7). The eigenvalues of the matrix $\sigma^{-2}\mathbf{K} = \mathbf{I} + \sigma^{-2}\mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^{\mathrm{T}}$ are easily seen to be the same as the nontrivial ($\neq 1$) ones of $\mathbf{I} + \sigma^{-2}\mathbf{\Lambda}\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}$, so that $\ln|\sigma^{-2}\mathbf{K}| = \ln|\mathbf{I}+\sigma^{-2}\mathbf{\Lambda}\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}|$. If we generalize the definition of $\mathcal{G}$ to $\mathcal{G} = (\mathbf{\Lambda}^{-1} + v\mathbf{I} + \sigma^{-2}\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi})^{-1}$ and also define $T(v) = \ln|(\mathbf{\Lambda}^{-1} + v)\mathcal{G}|$, then $T(\infty) = 0$ and so

$$\ln|\sigma^{-2}\mathbf{K}| = \ln|\mathbf{I} + \sigma^{-2}\mathbf{\Lambda}\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}| = T(0) - T(\infty) = \int_0^{\infty} dv\ [\mathrm{tr}\,(\mathbf{\Lambda}^{-1} + v)^{-1} - \mathrm{tr}\,\mathcal{G}] \quad (10)$$

Eqs. (9,10) show that all the averages required in (7) are of the form $\langle\mathrm{tr}\,\mathbf{M}\mathcal{G}\rangle$ with some matrix $\mathbf{M}$. We derived an accurate approximation for such averages in [5, 10, 11], with the result $\langle\mathrm{tr}\,\mathbf{M}\mathcal{G}\rangle = \mathrm{tr}\,\mathbf{M}\mathbf{G}$ where

$$\mathbf{G}^{-1} = \mathbf{\Lambda}^{-1} + \left(v + \frac{n}{\sigma^2 + g(n,v)}\right)\mathbf{I} \quad (11)$$

and the function $g(n,v)$ is determined by the self-consistency equation $g = \mathrm{tr}\,\mathbf{G}$. Using this approximation and (9,10) in (7) gives, after a few rearrangements,

$$\bar{E} = -\frac{1}{2}\ln(2\pi\sigma^2) - \frac{1}{2}\frac{\sigma_*^2 + \mathrm{tr}\,\mathbf{\Lambda}_*\mathbf{\Lambda}^{-1}\mathbf{G}}{\sigma^2 + g} - \frac{1}{2n}\int_0^{\infty} dv\ [g(0,v) - g(n,v)] \quad (12)$$

where in the second term $\mathbf{G}$ and g are evaluated at $v = 0$. This approximation for the average (normalized log-) evidence is the main result of this paper. The true $\bar{E}$ is known to achieve its maximum value when the student and teacher model are exactly matched, since the deviation from this maximum is essentially the KL-divergence between the student and teacher distributions over data sets. Remarkably, this property is preserved by the approximation (12): a rather lengthy calculation shows that it has a stationary point w.r.t. variation of $\mathbf{\Lambda}$ and σ^2 (which numerically always turns out to be maximum) at $\mathbf{\Lambda} = \mathbf{\Lambda}_*$ and $\sigma^2 = \sigma_*^2$.

3 Examples

If the eigenvalue spectra $\mathbf{\Lambda}$ and $\mathbf{\Lambda}_*$ are known, the approximation (12) for the average evidence can easily be evaluated numerically and maximized over the hyperparameters. As in the case of the unaveraged evidence (6), the maximization over the overall amplitude factor a can be carried out analytically. The resulting generalization performance can then be predicted using the results of [11].

As a first example scenario, consider inputs x which are binary vectors[1] with d components $x_a \in \{-1, 1\}$, and assume that the input distribution is uniform. I consider covariance functions for student and teacher that depend on the product $x \cdot x'$ only; this includes the standard choices (e.g. OU and RBF) which are functions of the Euclidean distance $|x-x'|$, since $|x-x'|^2 = 2d-2x\cdot x'$. All these covariance functions have the same eigenfunctions [17], so our above assumption is satisfied. The eigenfunctions are indexed by subsets ρ of $\{1, 2 \dots d\}$ and given explicitly by $\phi_\rho(x) = \prod_{a\in\rho} x_a$. The corresponding eigenvalues depend only on the size $s = |\rho|$ of the subsets and are therefore $\binom{d}{s}$-fold degenerate; letting $e = (1, 1 \dots 1)$ be the 'all ones' input vector, they can be written as $\Lambda_s = \langle C(x, e)\phi_\rho(x)\rangle_x$. From this the eigenvalues can easily be found numerically for any d, but here I focus on the limit of large d where all results can be obtained in closed form. If we write $C(x, x') = f(x \cdot x'/d)$, the eigenvalues become, for $d \to \infty$, $\Lambda_s = d^{-s} f^{(s)}(0)$ where $f^{(s)}(z) \equiv (d/dz)^s f(z)$. The contribution to $C(x, x) = f(1)$ from the s-th eigenvalue block is then $\lambda_s \equiv \binom{d}{s}\Lambda_s \to f^{(s)}(0)/s!$, consistent with $f(1) = \sum_{s=0}^{\infty} f^{(s)}(0)/s!$. Because of their scaling with d, the Λ_s become infinitely separated for $d \to \infty$. For training sets of size $n = \mathcal{O}(d^L)$, one then sees in (11) that eigenvalues with $s > L$ contribute as if $n = 0$, since $\Lambda_s \gg n/(\sigma^2 + g)$; these correspond to components of the teacher that have effectively not yet been learned [11]. On the other hand, eigenvalues with $s < L$ are completely suppressed and have been learnt perfectly. A hierarchical learning process thus results, where different scalings of n with d—as defined by L—correspond to different 'learning stages'. Formally, one can analyse the stages separately by letting $d \to \infty$ at a constant ratio $\alpha = n/\binom{d}{L}$ of the number of examples to the number of parameters to be learned at stage L; note that $\binom{d}{L} = \mathcal{O}(d^L)$ for large d. A replica calculation along the lines of Ref. [16] shows that the approximation (12) for the average evidence actually becomes *exact* in this limit. Fluctuations in E across different data sets also tend to zero so that considering $\bar{E}$ rather than E introduces no error.

Intriguingly, the resulting exact expression for the evidence at stage L turns out to depend only on two functions of the student hyperparameters. Setting $f_L = \sum_{s\geq L} \lambda_s$ (so that $f_0 = f(1)$), they are $f_{L+1} + \sigma^2$ and λ_L. The learning curve analysis in [11] showed that these correspond, respectively, to the student's

[1] This assumption simplifies the determination of the eigenfunctions and eigenvalues. For large d, one expects distributions with continuously varying x and the same first- and second-order statistics to give similar results [16]. A case where this can be shown explicitly is that of a uniform distribution over input vectors x of fixed length, which gives spherical harmonics as eigenfunctions.

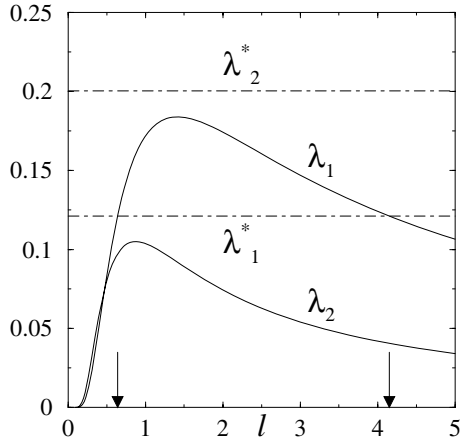


Fig. 1. Illustration of choice of optimal length scale in a scenario with large input space dimension d, for an OU student learning an RBF teacher with length scale $l_* = 0.55$. Evidence maximization gives the optimality criterion $\lambda_L = \lambda_L^*$ for learning stage L. At stage $L = 1$, this has two solutions for the student length scale l, marked by the arrows, while at stage $L = 2$ no solutions exist.

assumed values for the effective level of noise and for the signal to be learnt in the current stage. Independently of the number of training examples α, the evidence as calculated above can be shown to be maximal when these two parameters match the true values for the teacher, and it follows from the results of [11] that the resulting generalization error is then optimal, i.e. equal to the Bayes error. This implies in particular that overfitting maxima cannot occur.

A first implication of the above analysis is that even though evidence maximization can ensure optimal generalization performance, the resulting hyperparameter values are not meaningful as estimates of the underlying 'true' values of the teacher. Consider e.g. the case where the student assumes an OU covariance function, i.e. $C(x, x') = \exp[-|x - x'|/(ld^{1/2})]$ and therefore $f(z) = \exp[-\sqrt{2 - 2z}/l]$, but the teacher has an RBF covariance function, for which $C_*(x, x') = \exp[-|x - x'|^2/(2l^2 d)]$ and $f_*(z) = \exp[-(1 - z)/l^2]$. The length scales have been scaled by $d^{1/2}$ here to get sensible behaviour for $d \to \infty$. Then one has, for example,

$$\lambda_0 = e^{-\sqrt{2}/l}, \quad \lambda_1 = \lambda_0/(\sqrt{2}l), \qquad \lambda_0^* = e^{-1/l_*^2}, \quad \lambda_1^* = \lambda_0^*/l_*^2$$

For a given teacher length scale l_*, the optimal value of the student length scale l determined from the criterion $\lambda_L = \lambda_L^*$ will therefore generally differ from l_*, and actually depend on the learning stage L. Similarly, the optimal student noise level will not be identical to the true teacher noise level. At stage $L = 1$, for example, the optimal choice of length scale implies $\lambda_1 = \lambda_1^*$; but then $f_2 = 1 - \lambda_0 - \lambda_1$ will differ from f_2^* and the optimality condition $f_2 + \sigma^2 = f_2^* + \sigma_*^2$ tells us that $\sigma^2 \neq \sigma_*^2$.

A second interesting feature is that, since the λ_L and f_L depend in a complicated way on the hyperparameters, the optimality conditions $\lambda_L = \lambda_L^*$ and $f_{L+1} + \sigma^2 = f_{L+1}^* + \sigma_*^2$ may have more than one solution, or none at all, depending on the situation. An example is shown in Fig. 1. For a noise free ($\sigma_*^2 = 0$) RBF teacher with $l_* = 0.55$, one has $\lambda_1^* = 0.121$ and at learning stage $L = 1$ there are two very different optimal assignments for the student length scale, $l = 0.639$ and $l = 4.15$ (marked by arrows in Fig. 1) which achieve $\lambda_1(l) = \lambda_1^*$. The corresponding optimal noise levels are also very different at $\sigma^2 = 0.0730$ and $\sigma^2 = 0.674$, respectively. At stage $L = 2$, on the other hand, $\lambda_2^* = 0.2004$ and there is no value of the student length scale l for which $\lambda_2(l) = \lambda_2^*$. One finds that the evidence is in this case maximized by choosing l as large as possible. With l large, all λ_i for $i > 0$ are very small, and the student's assumed effective noise-to-signal ratio $(f_3 + \sigma^2)/\lambda_2$ becomes large. The results of [11] imply that the generalization error will decay extremely slowly in this case, and in fact not at all in the strict limit $l \to \infty$. Here we therefore have a case where strongly suboptimal performance results from evidence maximization, for the reason that the 'ideal' evidence maximum cannot be reached by tuning the chosen hyperparameters. In fact evidence maximization performs worse than learning with *any* fixed set of hyperparameters! Including a tunable overall amplitude factor a for the student's covariance function and noise level would, for the example values used above, solve this problem, and in fact produce a one-parameter family of optimal assignments of a, l and σ^2. One might expect this to be the generic situation but even here there are counter-examples: the optimality conditions demand equality of the student's effective noise-to-signal ratio, $\kappa_L = (f_{L+1} + \sigma^2)/\lambda_L$ with that of the teacher. But κ_L is independent of the amplitude factor a and $\geq f_{L+1}/\lambda_L$, and the latter ratio may be bounded above zero, e.g. $f_3/\lambda_2 \geq 3$ for any l for an OU student. For sufficiently low κ_L^* there is then no choice of l for which $\kappa_L = \kappa_L^*$.

In the second example scenario, I consider continuous-valued input vectors, uniformly distributed over the unit interval $[0, 1]$; generalization to d dimensions ($x \in [0, 1]^d$) is straightforward. For covariance functions which are stationary, i.e. dependent on x and x' only through $x - x'$, and assuming periodic boundary conditions (see [10] for details), one then again has covariance function-independent eigenfunctions. They are indexed by integers[2] q, with $\phi_q(x) = e^{2\pi i q x}$; the corresponding eigenvalues are $\Lambda_q = \int dx\, C(0, x) e^{-2\pi i q x}$. For the ('periodified' version of the) RBF covariance function $C(x, x') = a \exp[-(x - x')^2/(2l^2)]$, for example, one has $\Lambda_q \propto \exp(-\tilde{q}^2/2)$, where $\tilde{q} = 2\pi l q$. The OU case $C(x, x') = a \exp(-|x - x'|/l)$, on the other hand, gives $\Lambda_q \propto (1 + \tilde{q}^2)^{-1}$, thus $\Lambda_q \propto q^{-2}$ for large q. I also consider below covariance functions which interpolate in smoothness between the OU and RBF limits. E.g. the MB2 (modified Bessel or Matern class) covariance $C(x, x') = e^{-b}(1 + b)$, with $b = |x - x'|/l$, yields functions which are once differentiable [13, 15]; its eigenvalues $\Lambda_q \propto (1 + \tilde{q}^2)^{-2}$ show a faster asymptotic power law decay, $\Lambda_q \propto q^{-4}$, than those of the OU covari-

[2] Since $\Lambda_q = \Lambda_{-q}$, one can assume $q \geq 0$ if all Λ_q for $q > 0$ are taken as doubly degenerate.

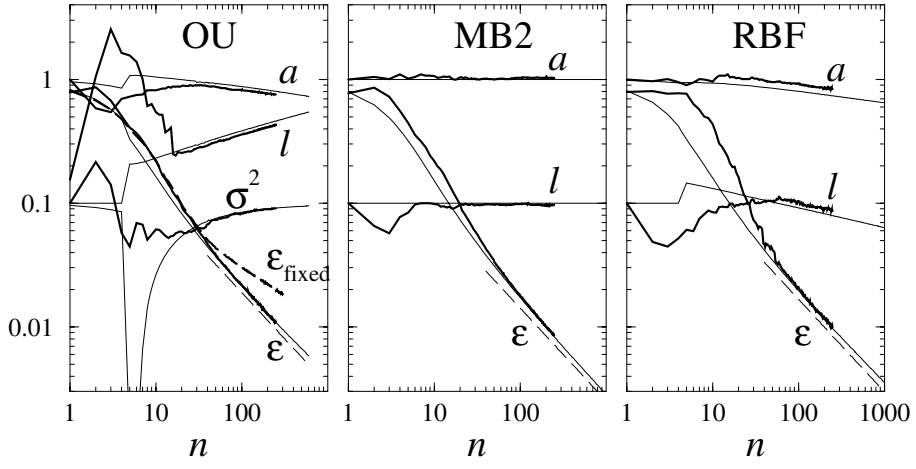


Fig. 2. Evidence maximization for a teacher with MB2 covariance function, $l_* = \sigma_*^2 = 0.1$, and inputs x uniformly distributed over $[0, 1]$. Bold lines: simulation averages over 20 to 50 independent sequences of training examples, thin lines: theory. Left: Hyperparameters a, l, σ^2 and generalization error ϵ vs. number of training examples n for OU student; the more slowly decaying generalization error ϵ_{fixed} for a fixed student model with $l = \sigma^2 = 0.1$, $a = 1$ is also shown. For the MB2 (middle) and RBF (right) students, σ^2 is close to constant at $\sigma^2 = \sigma_*^2$ in both theory and simulation and not shown. Dashed lines indicate the Bayes-optimal scaling of the asymptotic generalization error, $\epsilon \sim n^{-3/4}$, which with evidence maximization is obtained even in the cases with model mismatch (OU and RBF).

ance function. Writing the asymptotic behaviour of the eigenvalues generally as $\Lambda_q \propto q^{-r}$, and similarly $\Lambda_q^* \propto q^{-r_*}$, one has $r = 2$ for OU, $r = 4$ for MB2 and, due to the faster-than-power law decay of its eigenvalues, effectively $r = \infty$ for RBF. For the case of a fixed student model [11], the generalization error ϵ then generically decays as a power law with n for large n. If the student assumes a rougher function than the teacher provides ($r < r_*$), the asymptotic power law exponent $\epsilon \propto n^{-(r-1)/r}$ is determined by the student alone. In the converse case, the asymptotic decay is $\epsilon \propto n^{-(r_*-1)/r}$ and can be very slow, actually becoming logarithmic for an RBF student ($r \to \infty$). For $r = r_*$, the fastest decay for given r_* is obtained, as expected from the properties of the Bayes error.

The predictions for the effect of evidence maximization, based on (12), are shown in Fig. 2 for the case of an MB2 teacher ($r_* = 4$) being learned by a student with OU ($r = 2$), MB2 ($r = 4$) and RBF ($r = \infty$) covariance functions. Simulation results, obtained by averaging over 20 to 50 data sets for each n, are also shown. The most striking feature is that the theory predicts that in *all* three cases the generalization error now decays with the optimal power law scaling $\epsilon \sim n^{-(r_*-1)/r_*} = n^{-3/4}$; the simulations are consistent with this. In particular, for the RBF student the logarithmically slow learning has been eliminated. For the case of the MB2 student, the theory predicts that the optimal values of the student hyperparameters are constant and identical to those of the teacher; this

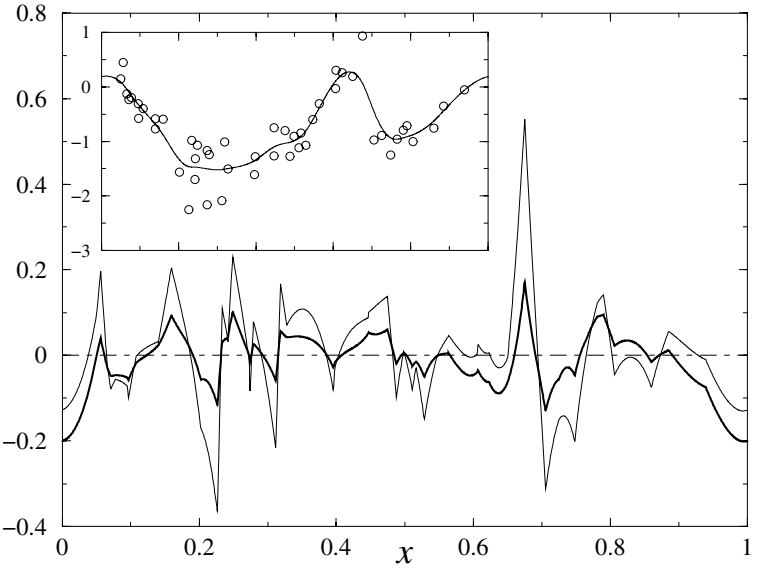


Fig. 3. Effects of evidence maximization for an OU student learning an RBF teacher, for input space dimension $d = 1$. Shown is a sample from one of the runs for $n = 50$ summarized in Fig. 2. Inset: Training data (circles) and Bayes optimal prediction function. Main graph: Difference between the prediction of the OU student and the Bayes-optimal prediction, for hyperparameters set equal to those of the teacher ($\sigma^2 = l = 0.1$, $a = 1$, thin line). Evidence maximization gives a larger l and thus a smoother student prediction that differs less from the Bayes-optimal prediction (bold line).

is as expected since then the models match exactly. The simulations again agree, though for small n the effects of our approximation of averaging the evidence over data sets and only then maximizing it become apparent.

For the OU student, inspection of the simulation results shows that the evidence maximum can, for some data sets, result in either one of two extreme hyperparameter assignments: $\sigma^2 = 0$, in which case the rough OU covariance function takes all noise on the teacher's underlying smooth target function as genuine signal, or l very large so that the covariance function is essentially constant and the student interprets the data as a constant function plus noise. Instances of the first type reduce the average of the optimal σ^2-values, a trend which the theory correctly predicts, but have a much stronger effect on the average optimal l through the rare occurrence of large values; our theory based on neglecting fluctuations cannot account for this. For larger n, where theory and simulation agree well, the optimal length scale l increases with n. This makes intuitive sense, since it effectively reduces the excessive roughness in the functions from the student's OU prior to produce a better match to the smoother teacher MB2 covariance function. An example of this effect is shown in Fig. 3. For the RBF student, the opposite trend in the variation of the optimal length scale l is seen: as n increases, l must be reduced to prevent the student from over-smoothing features of the rougher teacher.

4 Conclusion

In summary, the theory presented above shows that evidence maximization goes a long way towards making GP regression robust against model mismatch. The exact results for input spaces of large dimension $d \to \infty$ show that evidence maximization yields the (Bayes-)optimal generalization performance, as long as the true evidence maximum is achievable with the chosen hyperparameters. The optimal hyperparameter values are not, however, meaningful as estimates of the corresponding teacher parameters. The analysis also shows that evidence maximization has its risks, and does not always improve generalization performance: in cases where the ideal evidence maximum cannot be reached by tuning the available hyperparameters, evidence maximization can perform *worse* than learning with any fixed set of hyperparameters.

In the low-dimensional scenarios analysed, the theory predicts correctly that the optimal decay of the generalization error with training set size is obtained even for mismatched models, mainly by appropriate adaptation of the covariance function length scale. Our approximation of optimizing the evidence on average rather than for each specific data set performs worse for small data set sizes here, but predicts simulation results for larger n with surprising quantitative accuracy.

As an issue for further work, it would be interesting to derive the asymptotic decay of the generalization error analytically from (12). One puzzling issue is the increase of the length scale seen for an OU student in Fig. 2. One might argue naively that this increase cannot continue indefinitely because eventually the student covariance function would degenerate into a constant; the length scale should level off for sufficiently large n to prevent this. On the other hand, small deviations from a truly constant covariance function will be amplified by the presence of a large amount of data confirming that the target function is *not* a constant, and this suggests that a true divergence of the optimal length scale with n could occur.

A closer analysis of the effect of increasing d would also be worthwhile. For example, the fact that for $d \to \infty$ continuous ranges of optimal hyperparameter assignments can occur suggests that large fluctuations in the optimal values should be seen if scenarios with large but finite d are considered.

Finally, it will be interesting to understand how the above results for GP regression relate to work on *density estimation* using GP priors. There it has also been suggested that the decay of the estimation error with the number of available data points can be made largely independent of model mismatch by optimal hyperparameter tuning [18, 19, 20, 21, 22, 23, 24, 25].

Acknowledgment

This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the author's views.

References

[1] C K I Williams and C E Rasmussen. Gaussian processes for regression. In D S Touretzky, M C Mozer, and M E Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, pages 514–520, Cambridge, MA, 1996. MIT Press.

[2] C K I Williams. Computing with infinite networks. In M C Mozer, M I Jordan, and T Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 295–301, Cambridge, MA, 1997. MIT Press.

[3] D Barber and C K I Williams. Gaussian processes for Bayesian classification via hybrid Monte Carlo. In M C Mozer, M I Jordan, and T Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 340–346, Cambridge, MA, 1997. MIT Press.

[4] P W Goldberg, C K I Williams, and C M Bishop. Regression with input-dependent noise: A Gaussian process treatment. In M I Jordan, M J Kearns, and S A Solla, editors, *Advances in Neural Information Processing Systems 10*, pages 493–499, Cambridge, MA, 1998. MIT Press.

[5] P Sollich. Learning curves for Gaussian processes. In M S Kearns, S A Solla, and D A Cohn, editors, *Advances in Neural Information Processing Systems 11*, pages 344–350, Cambridge, MA, 1999. MIT Press.

[6] Lehel Csató, Ernest Fokoué, Manfred Opper, Bernhard Schottky, and Ole Winther. Efficient approaches to gaussian process classification. In S.A. Solla, T.K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 251–257, Cambridge, MA, 2000. MIT Press.

[7] D Malzahn and M Opper. Learning curves for Gaussian processes regression: A framework for good approximations. In T K Leen, T G Dietterich, and V Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 273–279, Cambridge, MA, 2001. MIT Press.

[8] D Malzahn and M Opper. Learning curves for Gaussian processes models: fluctuations and universality. *Lect. Notes Comp. Sci.*, 2130:271–276, 2001.

[9] D Malzahn and M Opper. A variational approach to learning curves. In T G Dietterich, S Becker, and Z Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 463–469, Cambridge, MA, 2002. MIT Press.

[10] P Sollich and A Halees. Learning curves for Gaussian process regression: approximations and bounds. *Neural Comput.*, 14(6):1393–1428, 2002.

[11] P Sollich. Gaussian process regression with mismatched models. In T G Dietterich, S Becker, and Z Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 519–526, Cambridge, MA, 2002. MIT Press.

[12] C A Michelli and G Wahba. Design problems for optimal surface interpolation. In Z Ziegler, editor, *Approximation theory and applications*, pages 329–348. Academic Press, 1981.

[13] C K I Williams and F Vivarelli. Upper and lower bounds on the learning curve for Gaussian processes. *Mach. Learn.*, 40(1):77–102, 2000.

[14] C K I Williams. Prediction with Gaussian processes: From linear regression to linear prediction and beyond. In M I Jordan, editor, *Learning and Inference in Graphical Models*, pages 599–621. Kluwer Academic, 1998.

[15] M Seeger. Gaussian processes for machine learning. *International Journal of Neural Systems*, 14(2):69–106, 2004.

[16] M Opper and R Urbanczik. Universal learning curves of Support Vector Machines. *Phys. Rev. Lett.*, 86(19):4410–4413, 2001.

[17] R Dietrich, M Opper, and H Sompolinsky. Statistical mechanics of Support Vector Networks. *Phys. Rev. Lett.*, 82(14):2975–2978, 1999.
[18] W Bialek, C G Callan, and S P Strong. Field theories for learning probability distributions. *Phys. Rev. Lett.*, 77(23):4693–4697, 1996.
[19] T E Holy. Analysis of data from continuous probability distributions. *Phys. Rev. Lett.*, 79(19):3545–3548, 1997.
[20] V Periwal. Reparametrization invariant statistical inference and gravity. *Phys. Rev. Lett.*, 78(25):4671–4674, 1997.
[21] V Periwal. Geometric statistical inference. *Nucl. Phys. B*, 554(3):719–730, 1999.
[22] T Aida. Field theoretical analysis of on-line learning of probability distributions. *Phys. Rev. Lett.*, 83(17):3554–3557, 1999.
[23] D M Schmidt. Continuous probability distributions from finite data. *Phys. Rev. E*, 61(2):1052–1055, 2000.
[24] T Aida. Reparametrization-covariant theory for on-line learning of probability distributions. *Phys. Rev. E*, 64:056128, 2001.
[25] I Nemenman and W Bialek. Occam factors and model independent Bayesian learning of continuous distributions. *Phys. Rev. E*, 65:026137, 2002.

Understanding Gaussian Process Regression Using the Equivalent Kernel

Peter Sollich[1] and Christopher K.I. Williams[2]

[1] Dept of Mathematics, King's College London,
Strand, London WC2R 2LS, U.K.
peter.sollich@kcl.ac.uk
[2] School of Informatics, University of Edinburgh,
5 Forrest Hill, Edinburgh EH1 2QL, U.K.
c.k.i.williams@ed.ac.uk

Abstract. The equivalent kernel [1] is a way of understanding how Gaussian process regression works for large sample sizes based on a continuum limit. In this paper we show how to approximate the equivalent kernel of the widely-used squared exponential (or Gaussian) kernel and related kernels. This is easiest for uniform input densities, but we also discuss the generalization to the non-uniform case. We show further that the equivalent kernel can be used to understand the learning curves for Gaussian processes, and investigate how kernel smoothing using the equivalent kernel compares to full Gaussian process regression.

1 Introduction

Consider the supervised regression problem for a dataset $\mathcal{D}$ with entries $(\mathbf{x}_i, y_i)$ for $i = 1, \ldots, n$. Under Gaussian Process (GP) assumptions the predictive mean at a test point $\mathbf{x}_*$ is given by

$$\bar{f}(\mathbf{x}_*) = \mathbf{k}^\top(\mathbf{x}_*)(K + \sigma^2 I)^{-1}\mathbf{y}, \tag{1}$$

where K denotes the $n \times n$ matrix of covariances between the training points with entries $k(\mathbf{x}_i, \mathbf{x}_j)$, $\mathbf{k}(\mathbf{x}_*)$ is the vector of covariances $k(\mathbf{x}_i, \mathbf{x}_*)$, σ^2 is the noise variance on the observations and $\mathbf{y}$ is a $n \times 1$ vector holding the training targets. See e.g. [2] for further details.

We can define a vector of functions $\mathbf{h}(\mathbf{x}_*) = (K + \sigma^2 I)^{-1}\mathbf{k}(\mathbf{x}_*)$. Thus we have $\bar{f}(\mathbf{x}_*) = \mathbf{h}^\top(\mathbf{x}_*)\mathbf{y}$, making it clear that the mean prediction at a point $\mathbf{x}_*$ is a linear combination of the target values $\mathbf{y}$. Gaussian process regression is thus a *linear smoother*, see [3, section 2.8] for further details. For a fixed test point $\mathbf{x}_*$, $\mathbf{h}(\mathbf{x}_*)$ gives the vector of weights applied to targets $\mathbf{y}$. Silverman [1] called $\mathbf{h}^\top(\mathbf{x}_*)$ the *weight function*.

Understanding the form of the weight function is made complicated by the matrix inversion of $K+\sigma^2 I$ and the fact that K depends on the specific locations of the n datapoints. Idealizing the situation one can consider the observations to be "smeared out" in $\mathbf{x}$-space at some constant density of observations. In this

J. Winkler, N. Lawrence, M. Niranjan (Eds.): Machine Learning Workshop, LNAI 3635, pp. 211–228, 2005.

case analytic tools can be brought to bear on the problem, as shown below. By analogy to kernel smoothing Silverman [1] called the idealized weight function the *equivalent kernel* (EK).

The structure of the remainder of the paper is as follows: In section 2 we describe how to derive the equivalent kernel in Fourier space. Section 3 derives approximations for the EK for the squared exponential and other kernels. In section 4 we show how use the EK approach to estimate learning curves for GP regression, and compare GP regression to kernel regression using the EK. A summary of our key results can be found in the short proceedings paper [4].

2 Gaussian Process Regression and the Equivalent Kernel

It is well known (see e.g. [5]) that the posterior mean for GP regression can be obtained as the function which minimizes the functional

$$J[f] = \frac{1}{2}\|f\|_{\mathcal{H}}^2 + \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - f(\mathbf{x}_i))^2, \tag{2}$$

where $\|f\|_{\mathcal{H}}$ is the RKHS norm corresponding to kernel k. (However, note that the GP framework gives much more than just this mean prediction, for example the predictive variance and the marginal likelihood $p(\mathbf{y})$ of the data under the model.)

Let $\eta(\mathbf{x}) = \mathbb{E}[y|\mathbf{x}]$ be the target function for our regression problem and write $\mathbb{E}[(y - f(\mathbf{x}))^2] = \mathbb{E}[(y - \eta(\mathbf{x}))^2] + (\eta(\mathbf{x}) - f(\mathbf{x}))^2$. Using the fact that the first term on the RHS is independent of f motivates considering a smoothed version of equation (2),

$$J_\rho[f] = \frac{\rho}{2\sigma^2}\int(\eta(\mathbf{x}) - f(\mathbf{x}))^2 d\mathbf{x} + \frac{1}{2}\|f\|_{\mathcal{H}}^2,$$

where ρ has dimensions of the number of observations per unit of $\mathbf{x}$-space (length/area/volume etc. as appropriate). If we consider kernels that are stationary, $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x} - \mathbf{x}')$, the natural basis in which to analyse equation (2) is the Fourier basis of complex sinusoids so that $f(\mathbf{x})$ is represented as $\int \tilde{f}(\mathbf{s})e^{2\pi i \mathbf{s}\cdot\mathbf{x}}d\mathbf{s}$ and similarly for $\eta(\mathbf{x})$. Thus we obtain

$$J_\rho[f] = \frac{1}{2}\int\left(\frac{\rho}{\sigma^2}|\tilde{f}(\mathbf{s}) - \tilde{\eta}(\mathbf{s})|^2 + \frac{|\tilde{f}(\mathbf{s})|^2}{S(\mathbf{s})}\right)d\mathbf{s}, \tag{3}$$

as $\|f\|_{\mathcal{H}}^2 = \int |\tilde{f}(\mathbf{s})|^2/S(\mathbf{s})d\mathbf{s}$ where $S(\mathbf{s})$ is the power spectrum of the kernel k, $S(\mathbf{s}) = \int k(\mathbf{x})e^{-2\pi i \mathbf{s}\cdot\mathbf{x}}d\mathbf{x}$. $J_\rho[f]$ can be minimized using calculus of variations to obtain $\tilde{f}(\mathbf{s}) = S(\mathbf{s})\eta(\mathbf{s})/(\sigma^2/\rho + S(\mathbf{s}))$ which is recognized as the convolution

$$f(\mathbf{x}_*) = \int h(\mathbf{x}_* - \mathbf{x})\eta(\mathbf{x})d\mathbf{x} \tag{4}$$

Here the Fourier transform of the equivalent kernel $h(\mathbf{x})$ is

$$\tilde{h}(\mathbf{s}) = \frac{S(\mathbf{s})}{S(\mathbf{s}) + \sigma^2/\rho} = \frac{1}{1 + \sigma^2/(\rho S(\mathbf{s}))}. \tag{5}$$

The term σ^2/ρ in the first expression for $\tilde{h}(\mathbf{s})$ corresponds to the power spectrum of a white noise process, whose delta-function covariance function becomes a constant in the Fourier domain. This analysis is known as Wiener filtering; see, e.g. [6, §14-1]. Notice that as $\rho \to \infty$, $h(\mathbf{x})$ tends to the delta function.

To see the relation between the EK and the weights $\mathbf{h}(\mathbf{x}_*)$ for prediction from a finite data set, one notes that the integral in equation (4) can be approximated by the discrete sum $(1/\rho)\sum_i h(\mathbf{x}_* - \mathbf{x}_i)y_i$; the factor $1/\rho$ represents the average volume element associated with each of the discrete training inputs. The EK $h(\mathbf{x}_* - \mathbf{x}_i)$ should therefore approximate the scaled weights $\rho\mathbf{h}(\mathbf{x}_*)$. We will see this confirmed below.

3 The EK for the Squared Exponential and Related Kernels

For certain kernels/covariance functions the EK $h(\mathbf{x})$ can be computed exactly by Fourier inversion. Examples include the Ornstein-Uhlenbeck process in $D = 1$ with covariance $k(x) = e^{-\alpha|x|}$ (see [6, p. 326]), splines in $D = 1$ corresponding to the regularizer $\|Pf\|^2 = \int (f^{(m)})^2 dx$ [1, 7], and the regularizer $\|Pf\|^2 = \int (\nabla^2 f)^2 d\mathbf{x}$ in two dimensions, where the EK is given in terms of the Kelvin function kei [8].

We now consider the commonly used squared exponential (SE) kernel $k(r) = \exp(-r^2/2\ell^2)$, where $r^2 = ||\mathbf{x} - \mathbf{x}'||^2$. (This is sometimes called the Gaussian or radial basis function kernel.) Its Fourier transform is given by

$$S(\mathbf{s}) = (2\pi\ell^2)^{D/2} \exp(-2\pi^2\ell^2|\mathbf{s}|^2)$$

where D denotes the dimensionality of $\mathbf{x}$ (and $\mathbf{s}$) space.

From equation (5) we obtain

$$\tilde{h}_{\mathrm{SE}}(\mathbf{s}) = \frac{1}{1 + b\exp(2\pi^2\ell^2|\mathbf{s}|^2)},$$

where $b = \sigma^2/\rho(2\pi\ell^2)^{D/2}$. We are unaware of an exact result in this case, but the following initial approximation is simple but effective. For large ρ, b will be small. Thus for small $s = |\mathbf{s}|$ we have that $\tilde{h}_{\mathrm{SE}} \simeq 1$, but for large s it is approximately 0. The change takes place around the point s_c where $b\exp(2\pi^2\ell^2 s_c^2) = 1$, i.e. $s_c^2 = \log(1/b)/2\pi^2\ell^2$. As $\exp(2\pi^2\ell^2 s^2)$ grows quickly with s, the transition of $\tilde{h}_{\mathrm{SE}}$ between 1 and 0 can be expected to be rapid, and thus be well-approximated by a step function.

Proposition 1. *The approximate form of the equivalent kernel for the squared-exponential kernel in D-dimensions is given by*

$$h_{\mathrm{SE}}(r) = \left(\frac{s_c}{r}\right)^{D/2} J_{D/2}(2\pi s_c r).$$

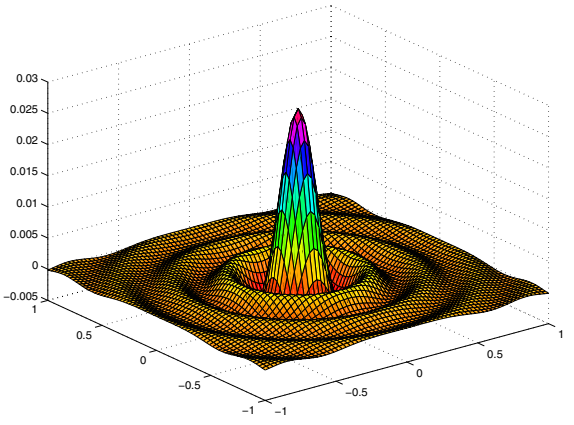


Fig. 1. Plot of the asymptotic form of the EK $(s_c/r)J_1(2\pi s_c r)$ for $D = 2$ and $\rho = 1225$

Proof: $\tilde{h}_{\mathrm{SE}}(\mathbf{s})$ is a function of $s = |\mathbf{s}|$ only, and for $D > 1$ the Fourier integral can be simplified by changing to spherical polar coordinates and integrating out the angular variables to give

$$h_{\mathrm{SE}}(r) = 2\pi r \int_0^\infty \left(\frac{s}{r}\right)^{\nu+1} J_\nu(2\pi rs)\tilde{h}_{\mathrm{SE}}(s)\, ds \tag{6}$$
$$\simeq 2\pi r \int_0^{s_c} \left(\frac{s}{r}\right)^{\nu+1} J_\nu(2\pi rs)\, ds = \left(\frac{s_c}{r}\right)^{D/2} J_{D/2}(2\pi s_c r).$$

where $\nu = D/2 - 1$, $J_\nu(z)$ is a Bessel function of the first kind and we have used the identity $z^{\nu+1}J_\nu(z) = (d/dz)[z^{\nu+1}J_{\nu+1}(z)]$. □

Note that in $D = 1$ by computing the Fourier transform of the boxcar function we obtain $h_{\mathrm{SE}}(x) = 2s_c\mathrm{sinc}(2\pi s_c x)$ where $\mathrm{sinc}(z) = \sin(z)/z$. This is consistent with Proposition 1 and $J_{1/2}(z) = (2/\pi z)^{1/2}\sin(z)$. The asymptotic form of the EK in $D = 2$ is shown in Figure 1.

Notice that s_c scales as $(\log(\rho))^{1/2}$ so that the width of the EK (which is proportional to $1/s_c$) will decay very slowly as ρ increases. In contrast for a spline of order m (with power spectrum $\propto |\mathbf{s}|^{-2m}$) the width of the EK scales as $\rho^{-1/2m}$ [1].

If instead of $\mathbb{R}^D$ we consider the input set to be the unit circle, a stationary kernel can be periodized by the construction $k_p(x, x') = \sum_{n\in\mathbb{Z}} k(x - x' + 2n\pi)$. This kernel will be represented as a Fourier series (rather than with a Fourier transform) because of the periodicity. In this case the step function in Fourier space approximation would give rise to a Dirichlet kernel as the EK (see [9, section 4.4.3] for further details on the Dirichlet kernel).

We now show that the result of Proposition 1 is asymptotically exact for $\rho \to \infty$, and calculate the leading corrections for finite ρ. The scaling of the width of the EK as $1/s_c$ suggests writing $h_{\mathrm{SE}}(r) = (2\pi s_c)^D g(2\pi s_c r)$. Then from equation (6) and using the definition of s_c

$$g(z) = \frac{z}{s_c(2\pi s_c)^D}\int_0^\infty \left(\frac{2\pi s_c s}{z}\right)^{\nu+1} \frac{J_\nu(zs/s_c)}{1+\exp[2\pi^2\ell^2(s^2-s_c^2)]}\,ds$$
$$= z\int_0^\infty \left(\frac{u}{2\pi z}\right)^{\nu+1} \frac{J_\nu(zu)}{1+\exp[2\pi^2\ell^2 s_c^2(u^2-1)]}\,du \qquad (7)$$

where we have rescaled $s = s_c u$ in the second step. The value of s_c, and hence ρ, now enters only in the exponential via $a = 2\pi^2\ell^2 s_c^2$. For $a \to \infty$, the exponential tends to zero for $u < 1$ and to infinity for $u > 1$. The factor $1/[1+\exp(\ldots)]$ is therefore a step function $\Theta(1-u)$ in the limit and Proposition 1 becomes exact, with $g_\infty(z) \equiv \lim_{a\to\infty} g(z) = (2\pi z)^{-D/2} J_{D/2}(z)$. To calculate corrections to this, one uses that for large but finite a the difference $\Delta(u) = \{1+\exp[a(u^2-1)]\}^{-1} - \Theta(1-u)$ is non-negligible only in a range of order $1/a$ around $u = 1$. The other factors in the integrand of equation (7) can thus be Taylor-expanded around that point to give

$$g(z) = g_\infty(z) + z\sum_{k=0}^\infty \frac{I_k}{k!}\frac{d^k}{du^k}\left[\left(\frac{u}{2\pi z}\right)^{\nu+1} J_\nu(zu)\right]\Big|_{u=1}, \quad I_k = \int_0^\infty \Delta(u)(u-1)^k\,du$$

The problem is thus reduced to calculating the integrals I_k. Setting $u = 1 + v/a$ one has

$$a^{k+1} I_k = \int_{-a}^0 \left[\frac{1}{1+\exp(v^2/a+2v)} - 1\right] v^k\,dv + \int_0^\infty \frac{v^k}{1+\exp(v^2/a+2v)}\,dv$$
$$= \int_0^a \frac{(-1)^{k+1}v^k}{1+\exp(-v^2/a+2v)}\,dv + \int_0^\infty \frac{v^k}{1+\exp(v^2/a+2v)}\,dv$$

In the first integral, extending the upper limit to ∞ gives an error that is exponentially small in a. Expanding the remaining $1/a$-dependence of the integrand one then gets, to leading order in $1/a$, $I_0 = c_0/a^2$, $I_1 = c_1/a^2$ while all I_k with $k \geq 2$ are smaller by at least $1/a^2$. The numerical constants are $-c_0 = c_1 = \pi^2/24$. This gives, using that $(d/dz)[z^{\nu+1}J_\nu(z)] = z^\nu J_\nu(z) + z^{\nu+1}J_{\nu-1}(z) = (2\nu+1)z^\nu J_\nu(z) - z^{\nu+1}J_{\nu+1}(z)$:

Proposition 2. *The equivalent kernel for the squared-exponential kernel is given for large ρ by $h_{\mathrm{SE}}(r) = (2\pi s_c)^D g(2\pi s_c r)$ with*

$$g(z) = \frac{1}{(2\pi z)^{\frac{D}{2}}}\left\{J_{D/2}(z) + \frac{z}{a^2}\left[(c_0 + c_1(D-1))J_{D/2-1}(z) - c_1 z J_{D/2}(z)\right]\right\}$$

within an expansion in $1/a$; the neglected terms are $O(1/a^4)$.

For e.g. $D = 1$ this becomes $g(z) = \pi^{-1}\{\sin(z)/z - \pi^2/(24a^2)[\cos(z) + z\sin(z)]\}$. Here and in general, by comparing the second part of the $1/a^2$ correction with the leading order term, one estimates that the correction is of relative size z^2/a^2. It will therefore provide a useful improvement as long as $z = 2\pi s_c r < a$; for larger z the expansion in powers of $1/a$ becomes a poor approximation because the correction terms (of all orders in $1/a$) are comparable to the leading order.

3.1 Accuracy of the Approximation

To evaluate the accuracy of the approximation we can compute the EK numerically as follows: Consider a dense grid of points in $\mathbb{R}^D$ with a sampling density ρ_{grid}. For making predictions at the grid points we obtain the smoother matrix $K(K+\sigma^2_{\text{grid}}I)^{-1}$, where[3] $\sigma^2_{\text{grid}} = \sigma^2\rho_{\text{grid}}/\rho$, as per equation (1). Each row of this matrix is (up to a factor ρ_{grid}) an approximation to the EK at the appropriate location, as this is the response to a $\mathbf{y}$ vector which is zero at all points except one. Note that in theory one should use a grid over the whole of $\mathbb{R}^D$ but in practice one can obtain an excellent approximation to the EK by only considering a grid around the point of interest as the EK typically decays with distance. Also, by only considering a finite grid one can understand how the EK is affected by edge effects.

Figure 2 shows plots of the weight function for $\rho = 100$, the EK computed on the grid as described above and the analytical sinc approximation. These are computed for parameter values of $\ell^2 = 0.004$ and $\sigma^2 = 0.1$, with $\rho_{\text{grid}}/\rho = 5/3$. To reduce edge effects, the interval $[-3/2, 3/2]$ was used for computations, although only the centre of this is shown in the figure. There is quite good agreement between the numerical computation and the analytical approximation, although the sidelobes decay more rapidly for the numerically computed EK. This is not surprising because the absence of a truly hard cutoff in Fourier space means one should expect less "ringing" than the analytical approximation predicts. The figure also shows good agreement between the weight function (based on the finite sample) and the numerically computed EK. The insets show the approximation of Proposition 2 to $g(z)$ for $\rho = 100$ ($a = 5.67$, left) and $\rho = 10^4$ ($a = 9.67$, right). As expected, the addition of the $1/a^2$-correction gives better agreement with the numerical result for $z < a$. Numerical experiments also show that the mean squared error between the numerically computed EK and the sinc approximation decreases like $1/\log(\rho)$. The is larger than the naïve estimate $(1/a^2)^2 \sim 1/(\log(\rho))^4$ based on the first correction term from Proposition 2, because the dominant part of the error comes from the region $z > a$ where the $1/a$ expansion breaks down.

3.2 Other Kernels

Our analysis is not in fact restricted to the SE kernel. First of all, it trivially extends to automatic-relevance determination kernels, which are obtained from the SE kernel by allowing separate lengthscales for each input dimension, i.e.

[3] To understand this scaling of σ^2_{grid} consider the case where $\rho_{\text{grid}} > \rho$ which means that the effective variance at each of the ρ_{grid} points per unit $\mathbf{x}$-space is larger, but as there are correspondingly more points this effect cancels out. This can be understood by imagining the situation where there are ρ_{grid}/ρ independent Gaussian observations with variance σ^2_{grid} at a single $\mathbf{x}$-point; this would be equivalent to one Gaussian observation with variance σ^2. In effect the ρ observations per unit $\mathbf{x}$-space have been smoothed out uniformly.

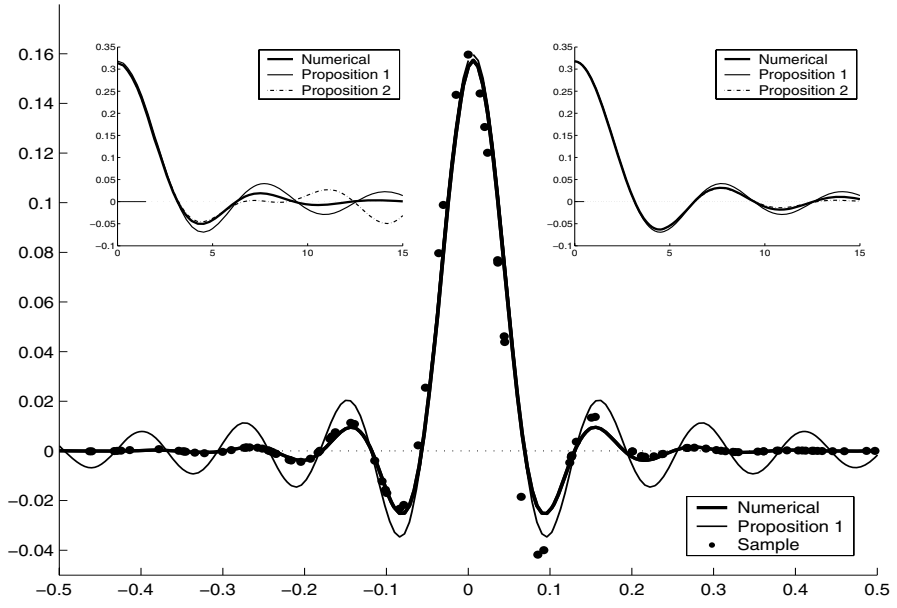


Fig. 2. Main figure: plot of the weight function $\rho \mathbf{h}(x_*)$ corresponding to $\rho = 100$ training points per unit length, plus the numerically computed equivalent kernel at $x_* = 0$ and the sinc approximation from Proposition 1. Insets: numerically evaluated $g(z)$ together with sinc and Proposition 2 approximations for $\rho = 100$ (left) and $\rho = 10^4$ (right).

$$k(\mathbf{x}) = \exp\left(-\frac{x_1^2}{2\ell_1^2} - \cdots - \frac{x_D^2}{2\ell_D^2}\right)$$

One can write this as $k(\mathbf{x}) = \exp(-||\mathbf{L}^{-1}\mathbf{x}||^2/2)$ with $\mathbf{L}$ a diagonal matrix containing the lengthscales $\ell_1, \ldots \ell_D$. This could be further extended to arbitrary linear transformations on input space, where $\mathbf{L}$ also has nonzero off-diagonal elements. Making the replacement $\tilde{\mathbf{x}} = \mathbf{L}^{-1}\mathbf{x}$ in the Fourier transform defining the power spectrum $S(\mathbf{s})$, and following through how this affects the EK as defined in equation (5), one easily finds that

$$h(\mathbf{x}) = |\mathbf{L}|^{-1}\, h_1\big(\mathbf{L}^{-1}\mathbf{x}, \rho|\mathbf{L}|\big) \tag{8}$$

where $h_1(\mathbf{x}, \rho)$ is the EK for the isotropic case with $\ell = 1$ and data point density ρ. Equation (8) tells us that the EK is stretched or squashed by exactly the same transformation matrix $\mathbf{L}$ as the underlying covariance kernel. The scaling of the density ρ by the determinant $|\mathbf{L}|$ is also reasonable: what is relevant is the number of data points per "correlation volume". The latter can be defined e.g. as the size of the region in $\mathbf{x}$-space where $k(\mathbf{x})$ is above some threshold value, and is proportional to $|\mathbf{L}|$. In the isotropic case one has $|\mathbf{L}| = \ell^D$, and the general result in equation (8) is consistent with the fact that, in our earlier results, ρ always appeared in the combination $\rho\ell^D$.

The identity (8) holds for linear transformations applied to any isotropic kernel. In the following we therefore only consider the latter; the power spectrum $S(\mathbf{s})$ then depends on $s = |\mathbf{s}|$ only. We can again define from equation (5) an effective cutoff s_c on the range of s in the EK via $\sigma^2/\rho = S(s_c)$, so that $\tilde{h}(s) = [1 + S(s_c)/S(s)]^{-1}$. The EK will then have the limiting form given in Proposition 1 if $\tilde{h}(s)$ approaches a step function $\Theta(s_c - s)$, i.e. if it becomes infinitely "steep" around the point $s = s_c$ for $s_c \to \infty$. A quantitative criterion for this is that the slope $|\tilde{h}'(s_c)|$ should become much larger than $1/s_c$, the inverse of the range of the step function. Since $\tilde{h}'(s) = S'(s)S(s_c)S^{-2}(s)[1 + S(s_c)/S(s)]^{-2}$, this is equivalent to requiring that $-s_c S'(s_c)/4S(s_c) \propto -d\log S(s_c)/d\log s_c$ must diverge for $s_c \to \infty$. The result of Proposition 1 therefore applies to *any* kernel whose power spectrum $S(s)$ decays more rapidly than any positive power of $1/s$.

A trivial example of a kernel obeying this condition would be a superposition of finitely many SE kernels with different lengthscales ℓ^2; the asymptotic behaviour of s_c is then governed by the smallest ℓ. A less obvious case is the "rational quadratic" $k(r) = [1 + (r/l)^2]^{-(D+1)/2}$ which has an exponentially decaying power spectrum $S(s) \propto \exp(-2\pi\ell s)$. (This relationship is often used in the reverse direction, to obtain the power spectrum of the Ornstein-Uhlenbeck (OU) kernel $\exp(-r/\ell)$.) Proposition 1 then applies, with the width of the EK now scaling as $1/s_c \propto 1/\log(\rho)$.

The previous example is a special case of kernels which can be written as superpositions of SE kernels with a distribution $p(\ell)$ of lengthscales ℓ, $k(r) = \int \exp(-r^2/2\ell^2)p(\ell)\,d\ell$. This is in fact the most general representation for an isotropic kernel which defines a valid covariance function in any dimension D, see [10, §2.10]. Such a kernel has power spectrum

$$S(s) = (2\pi)^{D/2} \int_0^\infty \ell^D \exp(-2\pi^2\ell^2 s^2) p(\ell)\,d\ell \tag{9}$$

and one easily verifies that the rational quadratic kernel, which has $S(s) \propto \exp(-2\pi\ell_0 s)$, is obtained for $p(\ell) \propto \ell^{-D-2}\exp(-\ell_0^2/2\ell^2)$. More generally, because the exponential factor in equation (9) acts like a cutoff for $\ell > 1/s$, one estimates $S(s) \sim \int_0^{1/s} \ell^D p(\ell)\,d\ell$ for large s. This will decay more strongly than any power of $1/s$ for $s \to \infty$ if $p(\ell)$ itself decreases more strongly than any power of ℓ for $\ell \to 0$. Any such choice of $p(\ell)$ will therefore yield a kernel to which Proposition 1 applies.

3.3 Non-uniform Input Densities

We next discuss how the above results generalize to the case where the input density $p(\mathbf{x})$ is not uniform. The smoothed version of the functional $J[f]$ in equation (2) is now

$$J_\rho[f] = \frac{n}{2\sigma^2} \int (\eta(\mathbf{x}) - f(\mathbf{x}))^2 p(\mathbf{x}) d\mathbf{x} + \frac{1}{2}\|f\|_{\tilde{\mathcal{H}}}^2, \tag{10}$$

To minimize this over f one decomposes the latter into eigenfunctions of the covariance kernel, rather than Fourier modes as before. The eigenfunctions are defined by the property

$$\int k(\mathbf{x},\mathbf{x}')\phi_s(\mathbf{x}')p(\mathbf{x}')d\mathbf{x}' = \lambda_s\phi_s(\mathbf{x})$$

where the λ_s are the associated eigenvalues. We index both by a subscript s to emphasize the similarity with the Fourier wavevector $\mathbf{s}$ we had so far; in particular, λ_s is the analogue of $S(\mathbf{s})$ above[4]. The eigenfunctions ϕ_s can always be chosen as normalized and orthogonal with respect to the input density, so that $\int \phi_s(\mathbf{x})\phi_{s'}(\mathbf{x})p(\mathbf{x})d\mathbf{x} = \delta_{ss'}$ and the covariance function has the decomposition $k(\mathbf{x},\mathbf{x}') = \sum_s \lambda_s\phi_s(\mathbf{x})\phi_s(\mathbf{x}')$. In terms of the components of f and η along the eigenfunctions, $\tilde{f}_s = \int f(\mathbf{x})\phi_s(\mathbf{x})p(\mathbf{x})d\mathbf{x}$ and similarly for $\tilde{\eta}_s$, the smoothed functional (10) can then be written as

$$J_\rho[f] = \frac{n}{2\sigma^2}\sum_s(\tilde{\eta}_s - \tilde{f}_s)^2 + \frac{1}{2}\sum\frac{\tilde{f}_s^2}{\lambda_s}\ .$$

Minimization over the $\tilde{f}_s$ then gives $\tilde{f}_s = \tilde{\eta}_s/[1+\sigma^2/(n\lambda_s)]$ or, after reassembling $f(\mathbf{x}) = \sum_s \tilde{f}_s\phi_s(\mathbf{x})$,

$$f(\mathbf{x}_*) = \sum_s \frac{\tilde{\eta}_s\phi_s(\mathbf{x}_*)}{1+\sigma^2/(n\lambda_s)} = \int h(\mathbf{x}_*,\mathbf{x})p(\mathbf{x})\eta(\mathbf{x})d\mathbf{x}' \tag{11}$$

where the equivalent kernel is now defined as

$$h(\mathbf{x}_*,\mathbf{x}) = \sum_s \frac{1}{1+\sigma^2/(n\lambda_s)}\,\phi_s(\mathbf{x}_*)\phi_s(\mathbf{x})\ . \tag{12}$$

One sees from this that, in general, the EK $h(\mathbf{x}_*,\mathbf{x})$ for non-uniform input densities is a function of its two arguments separately rather than just of their difference $\mathbf{x}_* - \mathbf{x}$ as in the uniform case. Also, the eigenfunctions $\phi_s(\mathbf{x})$ depend in a nontrivial manner on the input density and will not be known a priori.

To gain more insight into the behaviour of the EK for non-uniform input densities we now consider the specific case of one-dimensional inputs x with Gaussian density $p(x) = (2\pi)^{-1/2}\exp(-x^2/2)$ and a SE kernel. In that case the eigenfunctions are known to be [11]

$$\phi_s(x) = c^{1/4}(2^{s-1}s!)^{-1/2}e^{-(c-1/4)x^2}H_s(\sqrt{2c}x) \qquad s = 0,\ 1,\ldots,$$

[4] More precisely, if the input density $p(\mathbf{x})$ is uniform ($= 1/V$) over a large cubic box of size $V = L^D$, then the eigenfunctions $\phi_{\mathbf{s}}(\mathbf{x}) = \exp(2\pi i\mathbf{s}\cdot\mathbf{x})$ are indexed by Fourier wavevectors $\mathbf{s}$ whose components are multiples of $1/L$, and the eigenvalues are related to the power spectrum by $\lambda_{\mathbf{s}} = V^{-1}S(\mathbf{s})$. The resulting EK (12) reproduces the one derived earlier in (5) once it is multiplied by $p(\mathbf{x}) = 1/V$ and the limit $V \to \infty$ is taken; see also the discussion later in the text.

where $c = [1/16 + 1/(4l^2)]^{1/2}$ and $H_s(\cdot)$ is the s-th Hermite polynomial. The associated eigenvalues decay exponentially, $\lambda_s = l[2l(c - 1/4)]^{2k+1}$. The fraction in equation (12) again drops from $\simeq 1$ to $\simeq 0$, around the eigenfunction index s_c where $\lambda_{s_c} \simeq \sigma^2/n$. One can show that this drop becomes increasingly steep as $n\lambda_0/\sigma^2$ grows large, and imposing a hard cutoff at s_c according to $h(\mathbf{x}, \mathbf{x}') \approx \sum_{s=0}^{s_c} \phi_s(\mathbf{x})\phi_s(\mathbf{x}')$ should then give a good approximation to the EK. This approximation can in fact be evaluated in closed form because the eigenfunctions are, apart from s-independent factors, normalized orthogonal polynomials. The Christoffel-Darboux formula [12, eq. 22.12.1] then gives for the hard cutoff approximation to the EK

$$h(x, x') \approx \sqrt{\frac{s_c + 1}{4c}} \, \frac{\phi_{s_c+1}(x)\phi_{s_c}(x') - \phi_{s_c}(x)\phi_{s_c+1}(x')}{x - x'} \; . \tag{13}$$

In the results below, we have also calculated the unapproximated EK from equation (12), using the exact eigenfunctions and eigenvalues. We have compared this with a grid-based calculation: if K is the covariance matrix on a grid of density ρ_{grid}, one can show that the appropriate estimate for the product $h(\mathbf{x}, \mathbf{x}')p(\mathbf{x}')$ at the grid points is $\rho_{\text{grid}} K[K + \sigma^2 \rho_{\text{grid}}(nP)^{-1}]^{-1}$ where P is the diagonal matrix containing $p(\mathbf{x})$ at the grid points. This is directly analogous to the grid estimator we used for uniform input density, except for the replacement of the data point density ρ by nP. Numerically the grid estimate appears more robust, in particular for small ℓ where a larger number of eigenfunctions contribute to the EK.

A new issue in the case of non-uniform input densities is that both the EK $h(\mathbf{x}_*, \mathbf{x})$ and the combination $h(\mathbf{x}_*, \mathbf{x})p(\mathbf{x})$ are meaningful. The former should approximate the weights $\mathbf{h}(\mathbf{x}_*)$ one obtains for predicting at $\mathbf{x}$ for a given training sample of n discrete points. Indeed, the integral on the right-hand side of equation (11) is approximated by the discrete sum $(1/n)\sum_i h(\mathbf{x}_*, \mathbf{x}_i)y_i$ so that $h(\mathbf{x}_*, \mathbf{x}_i)$ approximates (n times) the weight given to training output y_i. Figure 3 shows that this correspondence holds rather well.

The fact that the combination of the EK with the input density, $h(\mathbf{x}_*, \mathbf{x})p(\mathbf{x})$, could be relevant is suggested by comparing the EK prediction (4) for the uniform case with its non-uniform analogue (11). This shows that $h(\mathbf{x}_*, \mathbf{x})p(\mathbf{x})$ plays the role of an effective smoothing kernel applied to the target function $\eta(\mathbf{x})$. We plot this combination in figure 4 and compare it to the result of the hard cutoff approximation (13), for sample size $n = 100$ and noise level $\sigma^2 = 0.1$. On the left, where the covariance function lengthscale is $\ell = 0.5$, the approximation is reasonable; as in the case of uniform input density, making the cutoff hard produces more ringing. In the plot on the right, where $\ell = 0.2$, this effect is rather more pronounced. This is consistent with the fact that the hard cutoff approximation should improve as $n\lambda_0/\sigma^2$ becomes large: this ratio is ≈ 390 for the situation on the left but ≈ 181 on the right.

Finally, one suspects that the EK for non-uniform input densities should reduce to the one for uniform densities if it is sufficiently peaked. More precisely, if the combination $h(\mathbf{x}_*, \mathbf{x})p(\mathbf{x})$ is concentrated in a region $\mathbf{x} \approx \mathbf{x}_*$ that is sufficiently small for $p(\mathbf{x})$ to be regarded as constant there, then it should coincide

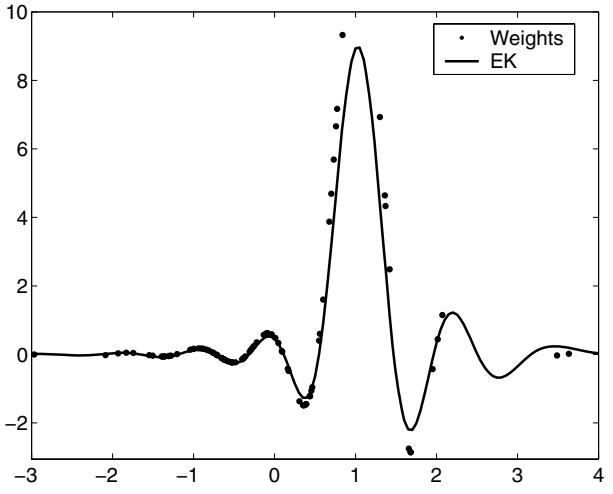


Fig. 3. The EK for Gaussian input density $p(x)$ and an SE covariance function with $\ell = 0.5$, at noise level $\sigma^2 = 0.1$ and for $n = 100$ data points. Solid line: numerically calculated EK $h(x_*, x)$ for $x_* = 1$. Markers: weights $\mathbf{h}(x_*)$ for prediction at the same point, for a random sample of $n = 100$ training inputs. The weights are multiplied by a factor of n to show the correspondence with the EK.

with the corresponding EK $h(\mathbf{x}_* - \mathbf{x})$ for a *uniform* input density of $\rho = np(\mathbf{x}_*)$. If ρ is large enough, one should be able to approximate further by using the asymptotic form of the EK from Proposition 1.

We test this intuition in figure 5, for a covariance function with lengthscale $\ell = 0.2$. For prediction at $x_* = 0, 1$ and 2 we observe that the EK calculated for uniform ρ provides a good approximation to the EK for the actual non-uniform $p(x)$, and the quality of asymptotic form from Proposition 1 is similar to the uniform case. As $|x_*|$ increases, the approximation of uniform density becomes worse. This arises from two trends. On the one hand, the width of the EK itself increases. On the other, the lengthscale over which $p(x)$ varies – which is $\sim 1/x$ for $x > 1$ – decreases. Eventually these two lengths therefore become comparable, and the variation of $p(x)$ can then no longer be neglected.

4 Understanding GP Learning Using the Equivalent Kernel

We now turn to using EK analysis to get a handle on average case learning curves for Gaussian processes. Here the setup is that a function η is drawn from a Gaussian process, and we obtain ρ noisy observations of η per unit $\mathbf{x}$-space at random $\mathbf{x}$ locations; note that for simplicity we revert to the case of uniform input density in this section. We are concerned with the mean squared error (MSE) between the GP prediction $\overline{f}$ and η. Averaging over the noise process,

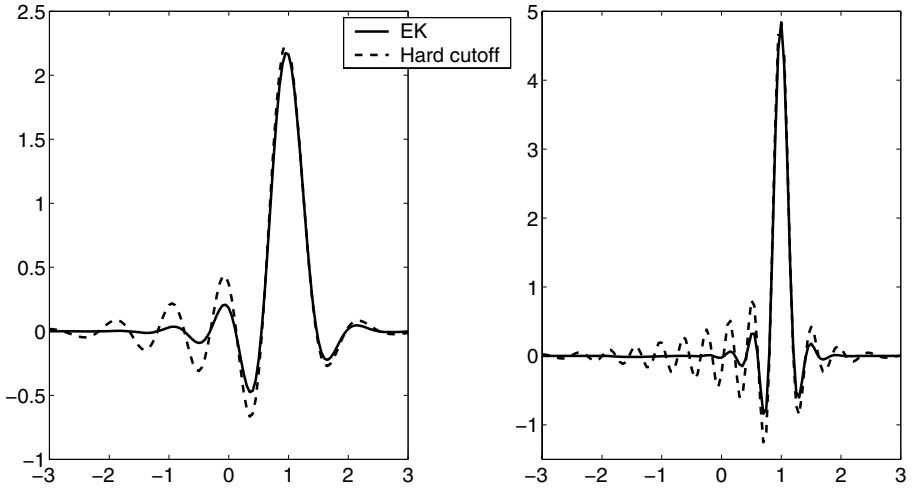


Fig. 4. EK times input density, $h(x_*, x)p(x)$, for Gaussian $p(x)$, $n = 100$, $\sigma^2 = 0.1$, and prediction point $x_* = 1$. The lengthscale of the covariance function is $\ell = 0.5$ on the left, and $\ell = 0.2$ on the right. The dashed lines show the corresponding results for the hard cutoff approximation (13).

the $\mathbf{x}$-locations of the training data and the prior over η we obtain the average MSE $\overline{\epsilon}$ as a function of ρ. See e.g. [13] and [14] for an overview of earlier work on GP learning curves.

To understand the asymptotic behaviour of $\overline{\epsilon}$ for large ρ, we now approximate the true GP predictions with the EK predictions from noisy data, given by $f_{\mathrm{EK}}(\mathbf{x}) = \int h(\mathbf{x} - \mathbf{x}')y(\mathbf{x}')d\mathbf{x}'$ in the continuum limit of "smoothed out" input locations. We assume as before that $y =$ target + noise, i.e. $y(\mathbf{x}) = \eta(\mathbf{x}) + \nu(\mathbf{x})$ where $\mathbb{E}[\nu(\mathbf{x})\nu(\mathbf{x}')] = (\sigma_*^2/\rho)\delta(\mathbf{x} - \mathbf{x}')$. Here σ_*^2 denotes the true noise variance, as opposed to the noise variance assumed in the EK; the scaling of σ_*^2 with ρ is explained in footnote 1. For a fixed target η, the MSE is $\epsilon = (\int d\mathbf{x})^{-1} \int [\eta(\mathbf{x}) - f_{\mathrm{EK}}(\mathbf{x})]^2 d\mathbf{x}$. Averaging over the noise process ν and target function η gives in Fourier space

$$\begin{aligned} \overline{\epsilon} &= \int \left\{ S_\eta(\mathbf{s})[1 - \tilde{h}(\mathbf{s})]^2 + (\sigma_*^2/\rho)\tilde{h}^2(\mathbf{s}) \right\} d\mathbf{s} \\ &= \frac{\sigma^2}{\rho} \int \frac{(\sigma^2/\rho)S_\eta(\mathbf{s})/S^2(\mathbf{s}) + \sigma_*^2/\sigma^2}{[1 + \sigma^2/(\rho S(\mathbf{s}))]^2} \, d\mathbf{s} \end{aligned} \tag{14}$$

where $S_\eta(\mathbf{s})$ is the power spectrum of the prior over target functions. In the case $S(\mathbf{s}) = S_\eta(\mathbf{s})$ and $\sigma^2 = \sigma_*^2$ where the kernel is exactly matched to the structure of the target, equation (14) gives the Bayes error $\overline{\epsilon}_{\mathrm{B}}$ and simplifies to

$$\overline{\epsilon}_{\mathrm{B}} = (\sigma^2/\rho) \int [1 + \sigma^2/(\rho S(\mathbf{s}))]^{-1} d\mathbf{s} \tag{15}$$

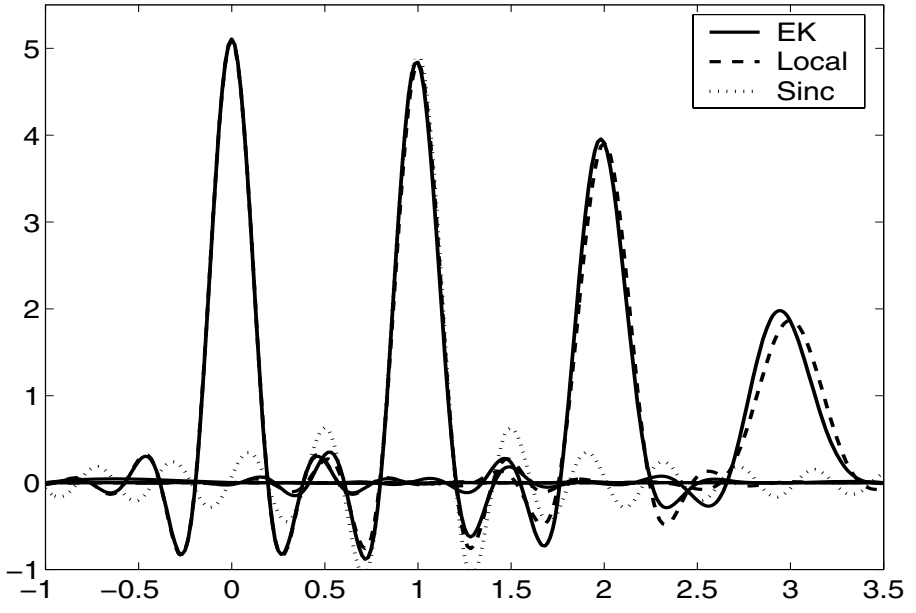


Fig. 5. EK times input density, $h(x_*, x)p(x)$ for Gaussian $p(x)$, $\ell = 0.2$, $n = 100$, $\sigma^2 = 0.1$, and $x_* = 0, 1, 2, 3$. Shown are the numerically calculated EK (solid lines), the EK calculated for a constant data point density $\rho = np(x_*)$ equal to the local density at x_* (dashed), and the approximation from Proposition 1 evaluated for the same ρ (dotted, shown for $x_* = 1$ only)

(see also [6, eq. 14-16]). Interestingly, this is just the analogue (for a continuous power spectrum of the kernel rather than a discrete set of eigenvalues) of the lower bound of [14] on the MSE of standard GP prediction from finite datasets. In experiments this bound provides a good approximation to the actual average MSE for large dataset size n [13]. This supports our approach of using the EK to understand the learning behaviour of GP regression.

Treating the denominator in the expression (3) for $\overline{\epsilon}_{\mathrm{B}}$ again as a hard cutoff at $s = s_c$, which is justified for large ρ, one obtains for an SE target and learner $\overline{\epsilon} = \overline{\epsilon}_{\mathrm{B}} \approx \sigma^2 s_c/\rho \propto (\log(\rho))^{D/2}/\rho$. Note that the Bayes error $\overline{\epsilon}_{\mathrm{B}}$ also indicates the mean-squared size of the errorbar of our predictions, i.e. the predictive variance, whether or not kernel and noise level match the target. This can be seen from the terms quadratic in f in the functional $J_\rho[f]$, equation (3).

To get analogous predictions of the MSE for the mismatched case, one can write equation (14) as

$$\overline{\epsilon} = \frac{\sigma_*^2}{\rho} \int \frac{[1 + \sigma^2/(\rho S(\mathbf{s}))] - \sigma^2/(\rho S(\mathbf{s}))}{[1 + \sigma^2/(\rho S(\mathbf{s}))]^2}\, d\mathbf{s} + \int \frac{S_\eta(\mathbf{s})}{[S(\mathbf{s})\rho/\sigma^2 + 1]^2}\, d\mathbf{s}.$$

The first integral is smaller than $(\sigma_*^2/\sigma^2)\overline{\epsilon}_{\mathrm{B}}$ and can be neglected as long as $\overline{\epsilon} \gg \overline{\epsilon}_{\mathrm{B}}$. In the second integral we can again make the cutoff approximation—though

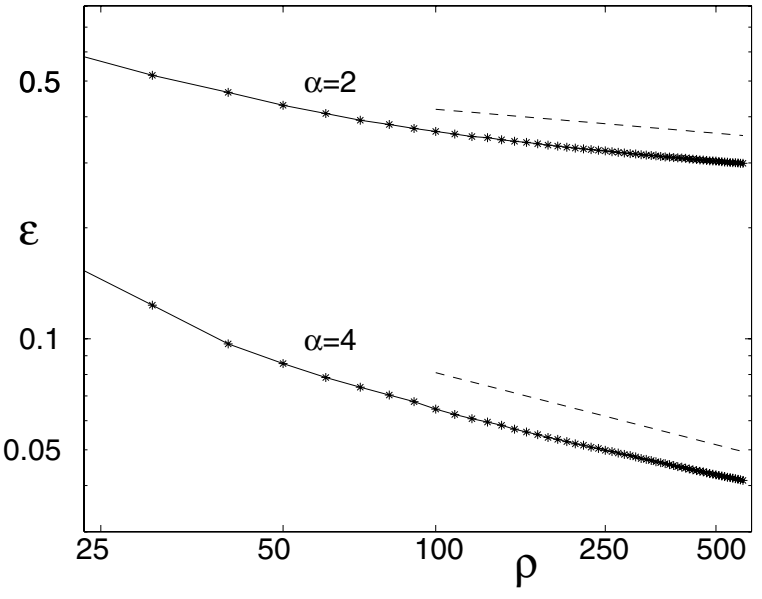


Fig. 6. Log-log plot of $\overline{\epsilon}$ against $\log(\rho)$ for the OU and Matern-class processes ($\alpha = 2,\ 4$ respectively). The dashed lines have gradients of $-1/2$ and $-3/2$ which are the predicted rates.

now with s having to be *above* s_c – to get the scaling $\overline{\epsilon} \propto \int_{s_c}^{\infty} s^{D-1} S_\eta(s)\, ds$. For target functions with a power-law decay $S_\eta(s) \propto s^{-\alpha}$ of the power spectrum at large s this predicts $\overline{\epsilon} \propto s_c^{D-\alpha} \propto (\log(\rho))^{(D-\alpha)/2}$. So we generically get slow logarithmic learning, consistent with the observations in [15]. For $D = 1$ and an OU target ($\alpha = 2$) we obtain $\overline{\epsilon} \sim (\log(\rho))^{-1/2}$, and for the Matern-class covariance function $k(r) = (1 + r/\ell)\exp(-r/\ell)$ (which has power spectrum $\propto (3/\ell^2 + 4\pi^2 s^2)^{-2}$, so $\alpha = 4$) we get $\overline{\epsilon} \sim (\log(\rho))^{-3/2}$. These predictions were tested experimentally using a GP learner with SE covariance function ($\ell = 0.1$ and assumed noise level $\sigma^2 = 0.1$) against targets from the OU and Matern-class priors (with $\ell = 0.05$) and with noise level $\sigma_*^2 = 0.01$, averaging over 100 replications for each value of ρ. To demonstrate the predicted power-law dependence of $\overline{\epsilon}$ on $\log(\rho)$, in Figure 6 we make a log-log plot of $\overline{\epsilon}$ against $\log(\rho)$. The dashed lines show the gradients of $-1/2$ and $-3/2$ and we observe good agreement between experimental and theoretical results for large ρ. We note that the predictive variance $\overline{\epsilon}_{\mathrm{B}} \sim 1/\rho$ (up to log-factors, see above) decays much faster than the true MSE $\overline{\epsilon}$, illustrating the possibility of overconfident predictions in mismatched scenarios.

5 Using the Equivalent Kernel in Kernel Regression

Above we have used the EK to understand how standard GP regression works. One could alternatively envisage using the EK to perform kernel regression, on

given finite data sets, producing a prediction $\rho^{-1}\sum_i h(\mathbf{x}_* - \mathbf{x}_i)y_i$ at $\mathbf{x}_*$. Intuitively this seems appealing as a cheap alternative to full GP regression, particularly for kernels such as the SE where the EK can be calculated analytically, at least to a good approximation. We now analyse how such an EK predictor would perform compared to standard GP prediction.

Letting $\langle\cdot\rangle$ denote averaging over noise, training input points and the test point and setting $f_\eta(\mathbf{x}_*) = \int h(\mathbf{x}, \mathbf{x}_*)\eta(\mathbf{x})d\mathbf{x}$, the average MSE of the EK predictor is

$$\begin{aligned}
\overline{\epsilon}_{\text{pred}} &= \left\langle \left[\eta(\mathbf{x}) - \frac{1}{\rho}\sum_i h(\mathbf{x}, \mathbf{x}_i)y_i \right]^2 \right\rangle \\
&= \left\langle [\eta(\mathbf{x}) - f_\eta(\mathbf{x})]^2 + \frac{\sigma_*^2}{\rho}\int h^2(\mathbf{x}, \mathbf{x}')d\mathbf{x}' \right\rangle \\
&\quad + \frac{1}{\rho}\left\langle \int h^2(\mathbf{x}, \mathbf{x}')\eta^2(\mathbf{x}')d\mathbf{x}' \right\rangle - \frac{1}{\rho}\left\langle f_\eta^2(\mathbf{x}) \right\rangle \\
&= \frac{\sigma^2}{\rho}\int \frac{(\sigma^2/\rho)S_\eta(\mathbf{s})/S^2(\mathbf{s}) + \sigma_*^2/\sigma^2}{[1 + \sigma^2/(\rho S(\mathbf{s}))]^2}\, d\mathbf{s} + \frac{\langle\eta^2\rangle}{\rho}\int \frac{d\mathbf{s}}{[1 + \sigma^2/(\rho S(\mathbf{s}))]^2}
\end{aligned}$$

Here we have set $\langle\eta^2\rangle = (\int d\mathbf{x})^{-1}\int \eta^2(\mathbf{x})\, d\mathbf{x} = \int S_\eta(\mathbf{s})\, d\mathbf{s}$ for the spatial average of the squared target amplitude. Taking the matched case, ($S_\eta(\mathbf{s}) = S(\mathbf{s})$ and $\sigma_*^2 = \sigma^2$) as an example, the first term in $\overline{\epsilon}_{\text{pred}}$ (which is the one we get for the prediction from "smoothed out" training inputs, see equation (14)) is of order $\sigma^2 s_c^D/\rho$, while the second one is $\sim \langle\eta^2\rangle s_c^D/\rho$. Thus both terms scale in the same way, but the ratio of the second term to the first is the signal-to-noise ratio $\langle\eta^2\rangle/\sigma^2$, which in practice is often large. The EK predictor will then perform significantly worse than standard GP prediction, by a roughly constant factor, and we have confirmed this prediction numerically. This result is somewhat surprising given the good agreement between the weight function $\mathbf{h}(\mathbf{x}_*)$ and the EK that we saw in figure 2, leading to the conclusion that the detailed structure of the weight function is important for optimal prediction from finite data sets.

One suspects intuitively that the suboptimal performance of the EK smoother must be related to the underlying assumption of uniformly distributed inputs. We therefore show in figure 7 a situation with the same parameters as in figure 2, but now for prediction at $x_* = 0.036$. In the training set for this figure there are two training points very near to this location, at $x = 0.0359$ and $x = 0.0362$. One sees that the effect of this is to depress the weights of these and nearby points significantly, causing deviations from the weights estimated from the EK. This phenomenon is easiest to understand in the limiting case where two training inputs essentially coincide, and there is no output noise. The true GP predictor then halves the weights these points would have if they were far apart: effectively, only one of the two points is used for prediction because the second one contributes no further information about the target function. The EK smoother, on the other hand, produces weights which are not sensitive to the locations of the training points in this way, and effectively overcounts the signal provided by the two nearby inputs. This problem would not be present

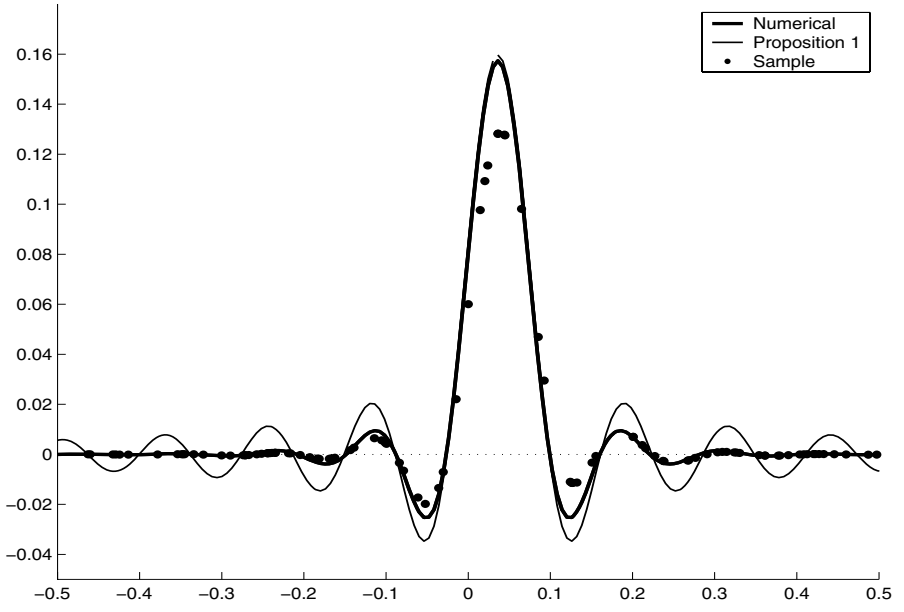


Fig. 7. Plot of the weight function corresponding to $\rho = 100$ training points/unit length (dots), plus the numerically computed equivalent kernel at $x_* = 0.036$ (thick solid line) and the sinc approximation from Proposition 1 (thin line)

when inputs are located on a regular grid, and indeed we find numerically that then the EK performs very similarly to the full GP predictor.

If the above picture is correct, then the EK smoother should not only generalize relatively poorly, but also provide a suboptimal fit to the training data. To check this, we worked out the average training error of the EK smoother,

$$\overline{\epsilon}_{\text{train}} = \frac{1}{n} \sum_i \left\langle \left[y_i - \frac{1}{\rho} \sum_j h(\mathbf{x}, \mathbf{x}_j) y_j \right]^2 \right\rangle$$

with the average taken as before over the noise process, the locations of the training inputs and the prior over the underlying target function η. The averages can be performed as in the calculation of the prediction error, but the number of terms is larger because the case where $i = j$ need to be treated separately. With the abbreviation $H = h(0)/\rho = \rho^{-1} \int [1 + \sigma^2/(\rho S(\mathbf{s}))]^{-1} d\mathbf{s}$ the result can be written as

$$\overline{\epsilon}_{\text{train}} - (\overline{\epsilon}_{\text{pred}} + \sigma_*^2) = H^2(\sigma_*^2 + \langle \eta^2 \rangle) - 2H \left\{ \sigma_*^2 + \int S_\eta(\mathbf{s}) \left[1 - \frac{1}{1 + \sigma^2/(\rho S(\mathbf{s}))} \right] d\mathbf{s} \right\} \tag{16}$$

The difference on the left-hand side is that between the training error and the noisy prediction error. One expects both of these quantities to tend to σ_*^2 for

large datasets; the noisy prediction error $\overline{\epsilon}_{\text{pred}} + \sigma_*^2$ clearly approaches the limit from above, while the training error $\overline{\epsilon}_{\text{train}}$ normally does so from below. For the EK smoother one finds, by estimating the dominant term on the right-hand side of equation (16) for the matched case (and for kernels where the approximation of the integrals in terms of a hard cutoff on s works),

$$\frac{\overline{\epsilon}_{\text{train}} - \sigma^2}{(\overline{\epsilon}_{\text{pred}} + \sigma^2) - \sigma^2} \to \frac{\langle \eta^2 \rangle - \sigma^2}{\langle \eta^2 \rangle + \sigma^2} \qquad \text{for} \qquad \rho \to \infty$$

For small noise, $\sigma^2 \ll \langle \eta^2 \rangle$, the training error of the EK smoother thus actually *decreases* towards its limiting value, in the same manner as the noisy prediction error. This is consistent with our expectation that the EK smoother provides a suboptimal fit to the training data. Only for large noise, $\sigma^2 > \langle \eta^2 \rangle$, do we recover the conventional behaviour whereby the training error approaches its limit value from below. This makes sense: in this regime even the full GP predictor will not ignore the second of two outputs corresponding to nearby input points, because significant output noise needs to be averaged out before the target function is learned.

6 Summary and Conclusion

In summary, we have derived accurate approximations for the equivalent kernel (EK) of GP regression with the widely used squared exponential kernel, and have shown that the same analysis in fact extends to a whole class of kernels. We discussed how our results generalize to cases with non-uniform input densities, and saw for the example of a Gaussian density that the resulting EK can often be approximated in a reasonable manner by taking the data point density to be uniform at its local value.

We have also demonstrated that EKs provide a simple means of understanding the learning behaviour of GP regression, even in cases where the learner's covariance function is not well matched to the structure of the target function. In future work, it will be interesting to explore in more detail the use of the EK in kernel smoothing. This is suboptimal compared to standard GP regression as we saw. However, it does remain feasible even for very large datasets, and may then be competitive with sparse methods for approximating GP regression. From the theoretical point of view, the average error of the EK predictor which we calculated may also provide the basis for useful upper bounds on GP learning curves.

Acknowledgments

This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views. We are grateful to Manfred Opper for pointing out the Christoffel-Darboux formula used to derive equation (13), and to two anonymous referees for helpful comments.

References

[1] B. W. Silverman. *Annals of Statistics*, 12:898–916, 1984.
[2] C. K. I. Williams. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 599–621. Kluwer Academic, 1998.
[3] T. J. Hastie and R. J. Tibshirani. *Generalized Additive Models*. Chapman and Hall, 1990.
[4] P. Sollich and C. K. I. Williams. In *NIPS 17*, 2005, to appear.
[5] F. Girosi, M. Jones, and T. Poggio. *Neural Computation*, 7(2):219–269, 1995.
[6] A. Papoulis. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, New York, 1991. Third Edition.
[7] C. Thomas-Agnan. *Numerical Algorithms*, 13:21–32, 1996.
[8] T. Poggio, H. Voorhees, and A. Yuille. Tech. Report AI Memo 833, MIT AI Laboratory, 1985.
[9] B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, 2002.
[10] M. L. Stein. *Interpolation of Spatial Data*. Springer-Verlag, New York, 1999.
[11] H. Zhu, C. K. I. Williams, R. J. Rohwer and M. Morciniec. In C. M. Bishop, editor, *Neural Networks and Machine Learning*. Springer, 1998.
[12] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions*. Dover, New York, 1965.
[13] P. Sollich and A. Halees. *Neural Computation*, 14:1393–1428, 2002.
[14] M. Opper and F. Vivarelli. In *NIPS 11*, pages 302–308, 1999.
[15] P. Sollich. In *NIPS 14*, pages 519–526, 2002.

Integrating Binding Site Predictions Using Non-linear Classification Methods

Yi Sun, Mark Robinson, Rod Adams, Paul Kaye,
Alistair Rust*, and Neil Davey

Science and technology research school,
University of Hertfordshire, United Kingdom
{comrys, m.robinson, r.g.adams, p.h.kaye, n.davey}@herts.ac.uk
http://homepages.feis.herts.ac.uk/∼nngroup
*Institute of Systems Biology,
1441 North 34th Street,
Seattle, WA 98103, USA
arust@systemsbiology.org

Abstract. Currently the best algorithms for transcription factor binding site prediction are severely limited in accuracy. There is good reason to believe that predictions from these different classes of algorithms could be used in conjunction to improve the quality of predictions. In this paper, we apply single layer networks, rules sets and support vector machines on predictions from 12 key algorithms. Furthermore, we use a 'window' of consecutive results in the input vector in order to contextualise the neighbouring results. Moreover, we improve the classification result with the aid of under- and over- sampling techniques. We find that support vector machines outperform each of the original individual algorithms and other classifiers employed in this work with both type of inputs, in that they maintain a better tradeoff between recall and precision.

1 Introduction

In this paper, we address the problem of identifying transcription factor binding sites on sequences of DNA. There are many different algorithms in current use to search for binding sites [1,2,3,4]. However, most of them produce a high rate of false positive predictions. The problem addressed here is to reduce these false positive predictions by means of classification techniques taken from the field of machine learning.

To do this we first integrate the results from 12 different algorithms for identifying binding sites, using non-linear classification techniques. To further improve classification results, we employ windowed inputs, where a fixed number of consecutive results are used as an input vector, so as to contextualise the neighbouring results. The data has two classes labeled as either binding sites or non-binding sites, with about 93% used being non-binding sites. We make use of sampling techniques, working with a traditional neural network: single

J. Winkler, N. Lawrence, M. Niranjan (Eds.): Machine Learning Workshop, LNAI 3635, pp. 229–241, 2005.

layer networks (SLN), rules sets (C4.5-Rules) and a contemporary classification algorithm: support vector machines (SVM).

We expound the problem domain in the next section. In Section 3, we introduce the datasets used in this paper. We explain how we apply under- and over- sampling techniques in Section 4. A set of common metrics and receiver operating characteristics graphs for assessing classifier performance are covered in Section 5. Section 6 briefly introduces our experiments and gives all the experimental results. The paper ends in Section 8 with conclusions.

2 Problem Domain

One of the most exciting and active areas of research in biology currently, is understanding how the exquisitely fine resolution of gene expression is achieved at the molecular level. It is clear that this is a highly non-trivial problem. While the mapping between the coding region of a gene and its protein product is straightforward and relatively well understood, the mapping between a gene's expression profile and the information contained in its non-coding region is neither so simple, nor well understood at present. It is estimated that as much as 50% of the human genome is cis-regulatory DNA [5] , undeciphered for the most part and tantalisingly full of regulatory instructions. Cis-regulatory elements form the nodes connecting the genes in the regulatory networks, controlling many important biological phenomena, and as such are an essential focus of research in this field [6]. Lines of research likely to directly benefit from more effective means of elucidating the cis-regulatory logic of genes include embryology, cancer and the pharmaceutical industry.

It is known that many of the mechanisms of gene regulation act directly at the transcriptional or sequence level, for example in those genes known to play integral roles during embryogenesis [6]. One set of regulatory interactions are those between a class of DNA-binding proteins known as transcription factors and short sequences of DNA which are bound by the proteins by virtue of their three dimensional conformation. Transcription factors will bind to a number of different but related sequences. A base substitution in a cis-regulatory element will commonly simply modify the intensity of the protein-DNA interaction rather than abolish it. This flexibility ensures that cis-regulatory elements, and the networks in which they form the connecting nodes, are fairly robust to various mutations. Unfortunately, it complicates the problem of predicting the cis-regulatory elements from out of the random background of the non-coding DNA sequences.

The current state of the art algorithms for transcription factor binding site prediction are, in spite of recent advances, still severely limited in accuracy. We show that in a large sample of annotated yeast promoter sequences, a selection of 12 key algorithms were unable to reduce the false positive predictions below 80%, with between 20% and 65% of annotated binding sites recovered. These algorithms represent a wide variety of approaches to the problem of transcription factor binding site prediction, such as the use of regular expression searches,

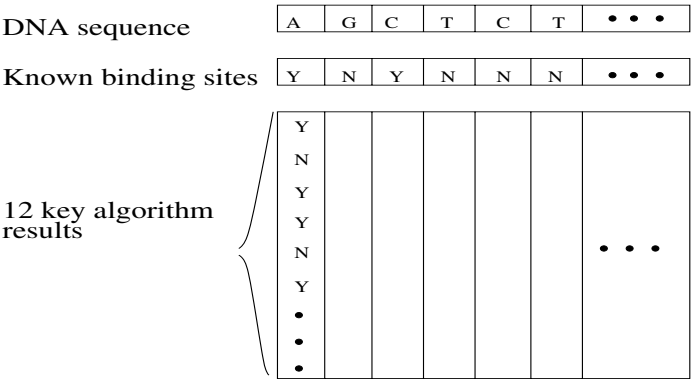


Fig. 1. The dataset has 68910 columns, each with a possible binding prediction (Y or N). The 12 algorithms give their own prediction for each sequence position and one such column is shown.

PWM scanning, statistical analysis, co-regulation and evolutionary comparisons. There is however good reason to believe that the predictions from these different classes of algorithms are complementary and could be integrated to improve the quality of predictions.

In the work described here we take the results from the 12 aforemention algorithms and combine them into 2 different feature vectors, as shown in the next section. We then investigate whether the integrated classification results of the algorithms can produce better classifications than any one algorithm alone.

3 Description of the Data

The data has 68910 possible binding positions and a prediction result for each of the 12 algorithms, see Figure 1. The 12 algorithms can be categorised as Single sequence algorithms (7) [1,7,8,9]; Coregulatory algorithms (3) [2,10]; A Comparative algorithm (1) [3]; An Evolutionary algorithm (1) [4].

The label information contains the best information we have been able to gather for the location of known binding sites in the sequences. Throughout we have used the following notation: 0 denotes the prediction that there is no binding site at this location; 1 the predictions that there is a binding site at this location, while 0.5 indicates that this algorithm is not able to analyse this location. The data therefore consists of 68910 12-ary ternary vectors each with an associated binary valued label.

In this work, we divide our dataset into a training set and a test set: the first 2/3 is the training set and the last 1/3 is the test set. Amongst the data there are many repeated vectors, some with the same label (repeated items) and some with different labels (inconsistent items). It is obviously unhelpful to have these repeated or inconsistent items in the training set, so they are removed. We call

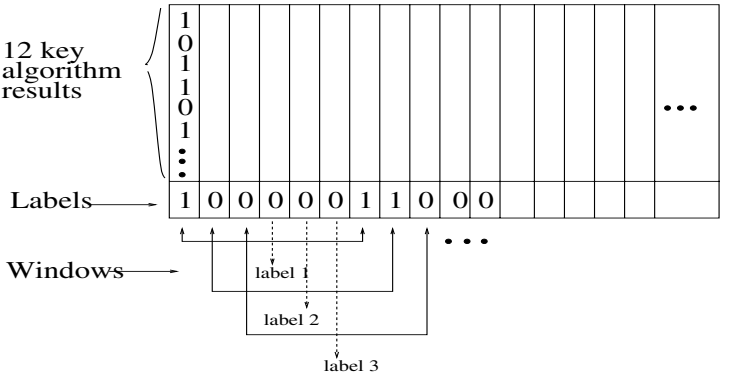


Fig. 2. The window size is set to 7 in this study. The middle label of 7 continuous prediction sites is the label for a new windowed inputs. The length of each windowed input now is 12×7.

the resulting data the *consistent* training set. However in the case of the test set we consider both the full set of data and the subset consisting of only the consistent test items. As can be seen in Table 1, the removal of repeated and inconsistent data dramatically reduces the number of data items: roughly 95% of data is lost.

As the data is drawn from a sequence of DNA nucleotides the label of other near locations is relevant to the label of a particular location. We therefore contextualise the training and test data by windowing the vectors as shown in Figure 2. We use the locations up to three either side, giving a window size of 7, and a consequent input vector size of 84. This has the considerable additional benefit of eliminating most of the repeated and inconsistent data: as can be seen in Table 1 now less than 25% of the data is lost.

Table 1 gives the sizes of all the different data sets used in this paper. The training set consists of either single vectors or windowed vectors. In both cases only consistent, non-repeating data is used. The test data consists of either single

Table 1. Description of the datasets used in this work

type			size
training	consistent	single	1862
		windowed	35577
test	consistent	single	1087
		restricted windowed	1087
		windowed	17045
	full	single	22967
		windowed	22967

vectors or windowed vectors as appropriate. Either the full test set or the relevant consistent subset is used. There is however, a special case, namely when we want to compare the windowed model with the single input version. Here we want to evaluate the windowed model on the locations represented in the consistent test set of the single vector model. We therefore construct a test set for the windowed model consisting of only those vectors corresponding to the 7 locations around each of the data points in the single consistent test set.

4 Sampling Techniques for Imbalanced Dataset Learning

In our dataset, there are less than 10% binding positions amongst all the vectors, so this is an *imbalanced* dataset [11]. Since the dataset is imbalanced, the supervised classification algorithms will be expected to over predict the majority class, namely the non-binding site category. There are various methods of dealing with *imbalanced* data [12]. In this work, we concentrate on the data-based method [13]: using under-sampling of the majority class (negative examples) and over-sampling of the minority class (positive examples). We combine both over-sampling and under-sampling methods in our experiments.

For under-sampling, we randomly selected a subset of data points from the majority class. The over-sampling case is more complex. In [11], the author addresses an important issue that the class imbalance problem is only a problem when the minority class contains very small subclusters. This indicates that simply over sampling with replacements may not significantly improve minority class recognition. To overcome this problem, we apply a synthetic minority oversampling technique as proposed in [13]. For each pattern in the minority class, we search for its $K-$nearest neighbours in the minority class using Hamming distance. A new pattern belonging to the minority class can then be generated by employing the majority voting principle to each element of the $K-$nearest neighbours in the feature vector space. We take 5 nearest neighbours, and double the number of items in the minority class. The actual ratio of minority to majority class is determined by the under-sampling rate of the majority class. We investigate final ratios of a half, one and two. The cross validation process (as described later) identified that a ratio of one half worked best for all the classifiers used.

5 Classifier Performance

It is apparent that for a problem domain with an imbalanced dataset, classification accuracy rate is not sufficient as a standard performance measure. To evaluate the classifiers used in this work, we apply *Receiver Operating Characteristics* (ROC) analysis [18], and several other common performance metrics, such as *recall*, *precision* and *F-score* [16,17], which are calculated to understand the performance of the classification algorithm on the minority class. Prior to introducing ROC curves, we give definitions of several common performance metrics.

5.1 Performance Metrics

Based on the confusion matrix computed from the test results (see Table 2, where TN is the number of true negative samples; FP is false positive samples; FN is false negative samples; FP is true positive samples), several common performance metrics can be defined as follows:

$$\text{Recall} = \text{TP} \;/\; (\text{TP} + \text{FN}), \tag{1}$$

$$\text{Precision} = \text{TP} \;/\; (\text{TP} + \text{FP}), \tag{2}$$

$$\text{F-score} = \frac{2 \cdot \text{Recall} \cdot \text{Precision}}{\text{Recall+Precision}}, \tag{3}$$

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+FN+TN+FP}}, \tag{4}$$

$$\text{fp\_rate} = \frac{\text{FP}}{\text{FP+TN}}. \tag{5}$$

Table 2. A confusion matrix

TN	FP
FN	TP

5.2 ROC Curves

ROC analysis has been used in the field of signal detection for a long time. Recently, it has also been employed in the machine learning and data mining domains. Here we follow [18] to give a basic idea of ROC curves.

ROC Curves. In a ROC diagram, the *true positive rate* (also called recall, see eq. (1)) is plotted on the Y axis and the *false positive rate* (fp_rate, see eq. (5)) is plotted on the X axis. Points in the top left of the diagram therefore have a high TP rate and a low FP rate and so represent good classifiers. The classifiers used here all produce a real valued output, that can be considered as a class membership probability. It is normal, when using a ROC diagram, to compare classifiers, to generate a set of points in ROC space by varying the threshold used to determine class membership. In this way a ROC curve corresponding to the performance of a single classifier but with a varying threshold is produced. One classifier is clearly better only when it dominates another over the entire performance space [18]. One attractive property of ROC curves is that they are insensitive to changes in class distribution, which makes them useful for analysing performance of classifiers using imbalanced datasets.

As noted for a ROC curve to be generated a real valued classifier is needed. The original SVM is a binary classifier. As described in [22] it is possible for the SVM to generate probabilistic outputs. For majority voting and weighted majority voting, we adopt methods proposed in [21]. The score assigned to each

pattern is the fraction of votes won by the majority in majority voting; while in weighted majority voting, each base algorithm votes with its *confidence*, which is measured by the probability that the given pattern (**I**) is positive (**P**), i.e.,

$$p(\mathbf{P}|\mathbf{I}) \approx \text{TP} \ / \ (\text{TP} + \text{FP}). \tag{6}$$

The class with the highest summed confidence wins, and the score is the average confidence. For the neural network classifiers a real valued output is automatically generated.

Often to measure a classifier performance, it is convenient to use a single value and the area under a ROC curve (AUC) can be used for this purpose. Its value ranges from 0 to 1. An effective classifier should have an AUC more than 0.5.

6 Experiments

The classification techniques we used in this work are single layer network (SLN) [14], support vector machine (SVM) [15], rule sets (C4.5-Rules) [20], majority voting (MV), and weighted majority voting (WMV).

The SVM experiments were completed using LIBSVM, which is available from the URL
`http://www.csie.ntu.edu.tw/~cjlin/libsvm`. The C4.5-Rules experiments were done using C4.5 software from [20]. C4.5-Rules is a companion program to C4.5. It creates rules sets by post-processing decision trees generated using the C4.5 algorithm first. The others were implemented using the NETLAB toolbox, which is available from the URL
`http://www.ncrg.aston.ac.uk/netlab/`.

6.1 Parameter Settings

All the user-chosen parameters are obtained using cross-validation. There are two training sets (single or windowed), and for each of these sets, and each classifier, the following cross validation procedure is carried out. The training set is divided into 5 equal subsets, one of which is to be a validation set, and there are therefore 5 possible such sets. For each classifier a range of reasonable parameter settings are selected. Each parameter setting is validated on each of the five validation sets having previously been trained on the other 4/5 of the training data. The mean performance as measured by the AUC metric over these 5 validations is taken as the overall performance of the classifier with this parameter setting. The parameter setting with best performance is then used with this classifier and the corresponding data set (single or windowed) in the subsequent experiments. For example the SVM has two parameters and six different combinations were evaluated.

There are several approaches to generate an averaging ROC curve from the different test sets [18]. In this paper, averaging ROC curves of cross-validation are obtained by first generating an ROC curve for each of validation sets, and then calculating the average scores from them.

The standard deviation of AUC can be attained either using the cross-validation method, or approximated as follows [19]:

$$se = \sqrt{\frac{A(1-A) + (N_p - 1)(Q_1 - A^2) + (N_n - 1)(Q_2 - A^2)}{N_n N_p}}, \tag{7}$$

where A denotes AUC, N_n and N_p are the number of negative and positive examples respectively, and $Q_1 = \frac{A}{2-A}$, $Q_2 = \frac{2A^2}{1+A}$.

7 Results

7.1 Cross Validation

In this experiment, we trained and tested the classifiers using 5-fold cross-validation as described above. The best set of parameters for each classifier were selected and the resulting AUC value (averaged over the 5-fold validation) is shown in Table 3. Table 3 also shows standard deviations computed using cross-validation. For single inputs, the SVM outperformed the SLN and C4.5-Rules, while the C4.5-Rules have the best performance with windowed inputs. In addition, due to the different size of the training sets (see Table 1), all classifiers have smaller standard deviations with windowed inputs than single inputs.

7.2 Classification Results on the Consistent Test Set with Single and Windowed Inputs

This test set has 1087 data points (see section 3) in both the single and windowed versions.

The results are shown in Table 4, together with the best base algorithm (the one with the highest F-score).

Compared with the best base algorithm, all classifiers, except MV increase the F-score and decrease the fp_rate. It can be seen that with single inputs, the SVM is clearly the best classifier - it outperforms the others in terms of all the performance metrics. However this is at a cost: in comparison to the best

Table 3. Cross Validation Results with Different classifiers

input	classifier	Mean of AUC	std
single	SLN	79.20	3.81
	SVM	81.52	3.76
	C4.5-Rules	71.42	1.61
windowed	SLN	78.49	0.44
	SVM	79.98	0.30
	C4.5-Rules	84.15	0.63

base algorithm the recall has been decreased. The classifier has become more conservative, predicting binding sites less often but with greater accuracy. With windowed inputs the SVM has a slightly better AUC score while the C4.5-Rules do best on all the other measures. When comparing the single and windowed results the only major difference is that C4.5-Rules does a lot better with windowed data.

Figure 3 shows ROC curves obtained using the consistent test set and single inputs. The curves confirm that the SVM gives best performance and MV and C4.5-Rules are the weakest.

7.3 Classification Results on the Consistent Test Set with Windowed Inputs

The results on the windowed, consistent test set, containing 17045 data points are given in Table 5 with the corresponding ROC curves in Figure 4. Both the SVM and the SLN do better than the best base algorithm, though once again the SVM is the best performer. The ROC curve clearly shows the poor performance of C4.5-Rules. It appears to be overfitting the training set, and this is confirmed by its excellent performance in the cross validation using the training sets, see Table 3.

Table 4. Common performance metrics (%) tested on the same consistent possible binding sites with single and windowed inputs separately. Some of the best results are shown in bold.

input	Classifier	recall	precision	F-score	Accuracy	fp_rate	AUC±se
single	best Alg.	58.23	14.74	23.53	72.49	26.39	-
	SLN	37.97	24.0	29.41	86.75	9.42	72.23±3.33
	SVM	37.97	28.85	**32.79**	88.68	7.34	**73.25±3.30**
	C4.5-Rules	37.97	19.11	25.42	83.81	12.60	67.45±3.43
	MV	48.10	15.57	23.53	77.28	20.44	64.38±3.46
	WMV	53.16	19.35	28.38	80.50	17.36	70.92±3.36
windowed	SLN	50.63	17.78	26.32	79.39	18.35	72.73±3.32
	SVM	51.90	20.20	29.08	81.06	16.07	**73.23±3.30**
	C4.5-Rules	46.84	22.70	**30.58**	84.54	12.50	72.58±3.32

7.4 Classification Results on the Full Test Set with Single and Windowed Inputs

In this experiment, we use the full contiguous test set. All the results are presented in Table 6, and the ROC curve for single input vectors shown in Figure 5. The ROC curve for windowed vectors is almost exactly the same as Figure 4.

Table 5. Common performance metrics (%) tested on 17045 consistent data points with windowed inputs.

input	Classifier	recall	precision	F-score	Accuracy	fp_rate	AUC±se
windowed	best Alg.	38.28	16.68	23.24	83.41	13.42	-
	SLN	31.40	18.75	23.48	86.58	9.55	65.59±0.91
	SVM	35.15	20.18	**25.64**	86.63	9.76	**67.03±0.91**
	C4.5-Rules	18.69	17.01	17.81	88.68	6.40	57.69±0.92

Table 6. Common performance metrics (%) tested on the full test set with single and windowed inputs.

input	Classifier	recall	precision	F-score	Accuracy	fp_rate	AUC±se
single	best Alg.	36.36	18.40	24.44	85.97	10.73	-
	SLN	12.28	21.23	15.56	91.68	3.03	66.13±0.81
	SVM	15.07	26.44	19.20	92.08	2.79	**66.23±0.81**
	C4.5-Rules	11.24	16.84	13.48	91.00	3.69	49.35±0.78
	MV	35.73	15.12	21.25	83.48	13.35	61.66±0.81
	WMV	34.75	20.04	**25.42**	87.28	9.23	63.75±0.81
windowed	SLN	30.01	20.78	24.56	88.50	7.61	67.16±0.89
	SVM	33.43	22.10	**26.61**	88.50	7.84	**68.0±0.80**
	C4.5-Rules	16.89	17.70	17.29	89.92	5.22	58.04±0.81

Looking at the results for the single inputs once again the SVM performs well. Although its recall is lower than the best base algorithm, this is explained by its far lower fp_rate. The C4.5-Rules perform particularly poorly, as is shown in Figure 5, where over most of the range it is predicting below random.

With windowed inputs the story is very much the same. In fact the windowed SVM is the overall best performer across single and windowed classifiers.

8 Conclusions

The significant result presented here is that by integrating the 12 algorithms we can considerably improve binding site prediction. In fact when considering the full contiguous test set, we are able to reduce the false positive predictions of the best base algorithm by 27%, whilst maintaining about the same number of true positive predictions. As expected the SVM gave a better classification result than the SLN and the decision trees. Majority voting was actually worse than the best individual algorithm. However, weighted majority voting was a little better. C4.5 has a tendency to badly overfit the training data and produce very poor predictions, sometimes worse than random.

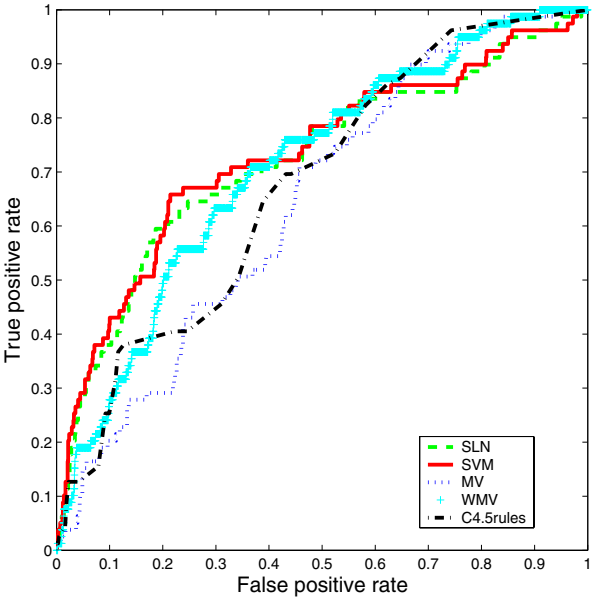


Fig. 3. ROC graph: five classifiers applied to the consistent test set with single inputs

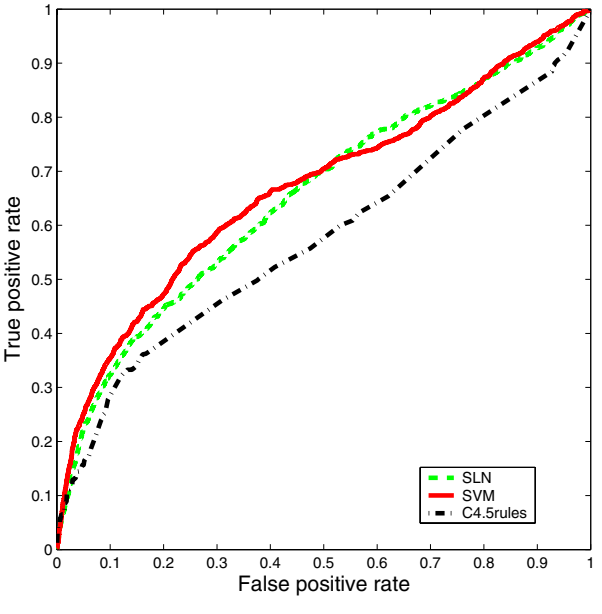


Fig. 4. ROC graph: three classifiers applied to the consistent test set using windowed inputs

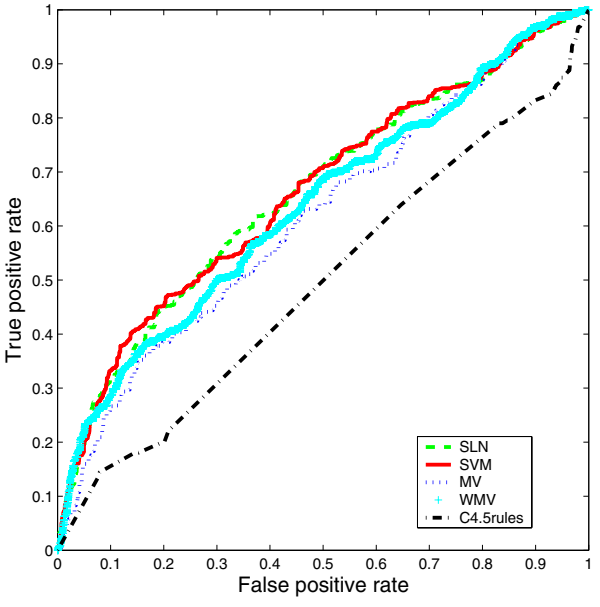


Fig. 5. ROC graph: five classifiers applied to the full test set with single inputs.

Future work will investigate i) using real valued algorithm results in the input vector; ii) using algorithm based technologies to cope with the imbalanced dataset; iii) considering a wider range of supervised meta-classifiers or ensemble learning algorithms. Another important avenue to explore will be to examine the biological significance of the results and we are currently working on using a visualisation tool.

References

1. *http://www.hgmp.mrc.ac.uk/Software/EMBOSS/.*
2. Bailey, T.L. & Elkan, C. (1994) Fitting a mixture model by expectation maximization to discover motifs in biopolymers, *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology*, 28-36, AAAI Press.
3. *http://family.caltech.edu/SeqComp/index.html.*
4. Blanchette, M. & Tompa, M. (2003) FootPrinter: a program designed for phylogenetic footprinting, *Nucleic Acids Research*, Vol. 31, No. 13, 3840-3842.
5. Markstein, M., Stathopoulos, A., Markstein, V., Markstein, P., Harafuji, N., Keys, D., Lee, B., Richardson, P., Rokshar, D., Levine, M.,: Decoding Noncoding Regulatory DNAs in Metazoan Genomes. *proceeding of 1st IEEE Computer Society Bioinformatics Conference (CSB 2002)*, 14-16 August 2002, Stanford, CA, USA.
6. Arnone, M. I. and Davidson, E. H.: The hardwiring of development: Organization and function of genomic regulatory systems. Development 124, 1851-1864, 1997
7. Apostolico, A., Bock, M.E, Lonardi, S., & Xu, X.(2000) Efficient Detection of Unusual Words, *Journal of Computational Biology*, Vol.7, No.1/2.

8. Rajewsky, N., Vergassola, M., Gaul, U. & Siggia, E.D. (2002) Computational detection of genomic cis regulatory modules, applied to body patterning in the early Drosophila embryo, BMC *Bioinformatics*, **3**:30.
9. Thijs, G., Marchal, K., Lescot, M., Rombauts, S., De Moor B, RouzP, & Moreau, Y. (2001) A Gibbs Sampling method to detect over-represented motifs in upstream regions of coexpressed genes, *Proceedings Recomb'2001,* 305-312.
10. Hughes, J.D., Estep, P.W., Tavazoie, S., & Church, G.M. (2000) Computational identification of cis-regulatory elements associated with groups of functionally related genes in Saccharomyces cerevisiae, *Journal of Molecular Biology*, Mar 10;**296**(5):1205-1214
11. Japkowicz, N.: Class imbalances: Are we focusing on the right issure? *Workshop on learning from imbalanced datasets, II, ICML*, Washington DC, 2003.
12. Wu, G and Chang, E. Y.: Class-boundary alignment for imbalanced dataset learning. *Workshop on learning from imbalanced datasets, II, ICML*, Washington DC, 2003.
13. Chawla, N. V., Bowyer, K. W., Hall, L. O. and Kegelmeyer, W. P.: SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*. Vol. 16, pp. 321-357, 2002.
14. Bishop, C.M.: *Neural Networks for Pattern Recognition*. Oxford University Press, New York (1995).
15. Scholköpf, B and Smola, A. J.:*Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, The MIT Press, 2002.
16. Buckland, M. and Gey, F.: The relationship between Recall and Precision, *Journal of the American Society for Information Science*, Vol. 45, No. 1, pp. 12–19, 1994.
17. Joshi, M., Kumar, V., and Agarwal, R.: Evaluating Boosting algorithms to classify rare classes: Comparison and improvements, *First IEEE International Conference on Data Mining*, San Jose, CA, 2001.
18. Fawcett, T.: ROC graphs: notes and practical considerations for researchers. Kluwer Academic publishers, 2004.
19. Hanley, J.A. and McNeil, B.J.: The meaning and use of the area under a receiver operating characteristic (ROC) curve, *Radiology*, 143, 29-36, 1982.
20. Quinlan, J.R.: *C4.5: Programs for Machine Learning*, Morgan Kauffman, 1993.
21. Fawcett, T.: Using rule sets to maximize ROC performance. *Proceedings of the IEEE International Conference on Data Mining (ICDM-2001)*, Los Alamitos, CA, pp 131-138, IEEE Computer Society, 2001.
22. Wu, T.F., Lin, C.J. and Weng, R.C.: Probability Estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research*, 5 pp. 975-1005, 2004.

Support Vector Machine to Synthesise Kernels

Hongying Meng, John Shawe-Taylor, and Sandor Szedmak,
and Jason D.R. Farquhar

School of Electronics and Computer Science, University of Southampton,
Southampton SO17 1BJ, UK
{hm1, jst, sso3v, sjdrf}@ecs.soton.ac.uk

Abstract. In this paper, we introduce a new method (SVM_2K) which amalgamates the capabilities of the Support Vector Machine (SVM) and Kernel Canonical Correlation Analysis (KCCA) to give a more sophisticated combination rule that the boosting framework allows. We show how this combination can be achieved within a unified optimisation model to create a consistent learning rule which combines the classification abilities of the individual SVMs with the synthesis abilities of KCCA. To solve the unified problem, we present an algorithm based on the Augmented Lagrangian Method. Experiments show that SVM_2K performs well on generic object recognition problems in computer vision.

1 Introduction

The Support Vector Machine [1] is a new generation of learning systems based on advances in statistical learning theory. It delivers state-of-the-art performance in real world applications, such as text categorisation, hand-written character recognition, image classification, bio-sequence analysis, etc.

Increasing amounts of multimedia data have become easily accessible in recent years – driving the need to develop methods to efficiently analyse and search this data. In [2], it has been shown that the combination of different types of features is able to give a more accurate result than each component can separately. Similarly, Hardoon D.R. et al.[3] show how Kernel Canonical Correlation Analysis (KCCA)[4] can be used to combine image and text extracted from the web to improve web page classification. KCCA has been successfully applied in information retrieval applications where one of two views is used to retrieve the other, such as of cross-lingual retrieval [5] and content-based image retrieval[3,6].

In [7], the authors combine different components for a generic object classification task. Again, KCCA is used to learn the semantic feature space between different features from the same image and produce a new SVM kernel function. The new kernel mapping can efficiently combine two distinctive features into a semantic one, and significantly improve classification accuracy.

The traditional SVM can not deal with multiple types of features directly, limiting its application in this area. Based on kernel methods, recent progress has focused on developing different optimization models to solve this kind of problem.

J. Winkler, N. Lawrence, M. Niranjan (Eds.): Machine Learning Workshop, LNAI 3635, pp. 242–255, 2005.

Lanckriet G.R. and his co-researchers [8,10] describe a method for combining multiple kernel representations in an optimal fashion, by formulating a convex optimization problem which is solvable by Semidefinite Programming (SDP). In [11], they used Sequential Minimal Optimization (SMO) to solve this problem efficiently. The method was applied to the problem of predicting yeast protein functional classifications. On this problem their method performs better than a SVM trained on any single source of the data and a previously-described Markov random field based algorithm.

Andrews S. et al. [12] presented two new formulations for multiple-instance learning as maximum margin problems solvable by mixed integer quadratic programs. These extensions created a state of the art classification technique making all the SVM learning approaches, including non-linear classification via kernels, available to an area that up to now has been largely dominated by special purpose methods.

The main common point of these approaches is that they extended standard SVM optimization problem by adding some specific rules to the kernels or to the features.

In this paper, we propose a new method (SVM_2K) to solve this kind of problem. by exploiting the interaction of the decision functions provided by the per-feature set SVM subproblems. The subproblem SVMs focus on classification based upon each distinct feature set, whilst a KCCA like algorithm manages their interaction to synthesise a single learner from the sub-SVMs and hence improve generalisation performance. We show how these disparate functions can be formulated in a unified optimisation model to create a consistent learning rule. To cope with the complexity of solving this unified problem we have developed a new algorithm based on Augmented Lagrangian method.

2 Problem Addressed

Learning via two feature set recently attracts several researchers to improve the capability of known learning techniques. Dasgupta et al. [9] showed a learner can exploit explicit or implicit interactions between different features to decrease the generalization error. Analogue approaches call this kind of approaches as co-training, multi-task or multi-view learning to define their underlying concepts. This techniques are mostly applied for semi-supervised learning, where the second source of the features relates to a set of unlabeled data. The base idea behind these approaches if the learners highly agree on the training set then their generalization error is probably smaller. Some examples are of these papers Blum et al. [18], Evgeniou et al. [19] and Muslea et al. [20].

Conjecturing the extendibility of this result we created a maximum margin framework for a pair of SVM classifiers working on two sources of input features.

We have a set $\mathcal{S}_t = \{\mathbf{s}_i^{(t)} = (\mathbf{x}_i^{(t)}, y_i),\ i = 1, \ldots, m\},\ t = 1, \ldots, T$ of samples drawn from the same unknown distribution. Every sample $\mathcal{S}_t$ comprises the same object with the same labels $y = \{y_i = \{-1, +1\}\}$ but the sets of feature vectors $\{\mathbf{x}_i^{(t)} \in \mathcal{X}_t\}$ are different. Furthermore, for each set of features there is

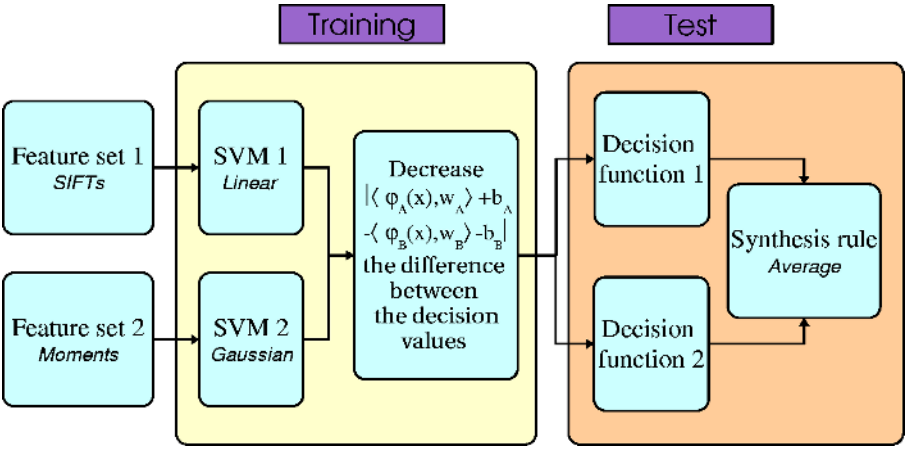


Fig. 1. This is the outline of our method. Two distinct feature sets were combined in the training by an integrated SVM optimization problem. And the decision functions were combined in the test process.

a mapping $\phi_t : \mathcal{X}_t \to R^{n_t}$ defined on the feature vectors. The task is to find a function $f : R^T \to \{-1, 1\}$ and a set functions $h_t : R^{n_t} \to R$ such that they give the best prediction for the labels y with respect to a given loss functional following the form $y_i \sim f(h_1(\phi_t(\mathbf{x}_i^{(t)})), \ldots, h_T(\phi_T(\mathbf{x}_i^{(T)})))$. Assume the functions $\{h_t\}$ are linear, i.e., $h_t(z) = \mathbf{z}^T\mathbf{w}_t + b_t$, for all $t = 1, \ldots, T$.

To solve this type of problem the trivial approach of simply concatenating the feature vectors $\mathbf{x}_i^{(t)}$ together to form a single long feature vector to which conventional learning methods, such as an SVM, can be applied appears promising. However, one problem with this approach is that different kernels may work best with different feature sets, for example linear with feature set 1 and Gaussian with feature set 2. As a single SVM can only use one kernel, using the concatenated features we will be forced to use a kernel which is not suited to the data – potentially reducing performance.

The question is now how to create a classification method based upon these distinct features that is able to realise the potential advantages of the distinct sources of information.

Let us reduce the problem size T given by the number of distinct features to 2 and used the indeces A and B instead of the numbers. Consider the SVM style optimization problem

$$\begin{gathered}\min \tfrac{1}{2}(||\mathbf{w}_A||_2^2 + ||\mathbf{w}_B||_2^2) + \mathbf{1}^T(C^A\boldsymbol{\xi}^A + C^B\boldsymbol{\xi}^B + D\boldsymbol{\eta}) \\ \text{with respect to} \\ \mathbf{w}_A,\ \mathbf{w}_B,\ b_A,\ b_B,\ \boldsymbol{\xi}^A,\ \boldsymbol{\xi}^B,\ \boldsymbol{\eta} \\ \text{subject to}\end{gathered}$$

$$\begin{array}{ll} Synthesis & \psi(\langle \mathbf{w}_A, \phi_A(\mathbf{x}_i^A)\rangle + b_A, \langle \mathbf{w}_B, \phi_B(\mathbf{x}_i^B)\rangle + b_B) \leq \eta_i + \epsilon, \\ subSVM1 & y_i(\langle \mathbf{w}_A, \phi_A(\mathbf{x}_i^A)\rangle + b_A) \geq 1 - \boldsymbol{\xi}_i^A, \\ subSVM2 & y_i(\langle \mathbf{w}_B, \phi_B(\mathbf{x}_i^B)\rangle + b_B) \geq 1 - \boldsymbol{\xi}_i^B, \\ & \boldsymbol{\xi}^A \geq 0,\ \boldsymbol{\xi}^B \geq 0,\ \boldsymbol{\eta} \geq 0,\ i = 1, \ldots, m, \\ & \boldsymbol{\xi}^A = (\xi_1^A, \ldots, \xi_m^A),\ \boldsymbol{\xi}^B = (\xi_1^B, \ldots, \xi_m^B), \\ & \boldsymbol{\eta} = (\eta_1, \ldots, \eta_m). \end{array} \tag{1}$$

In this formulation $\mathbf{1}$ is a vector for which every component equals to 1 and Y is a diagonal matrix containing the labels $\{y_i,\ i = 1, \ldots, m\}$. The constants C^A, C^B and D are penalty parameters. The important part of this formulation is the synthesis function ψ which links the 2 SVM subproblems by forcing them to be similar with respect to the values of the decision functions.

In this paper we define ψ in one of the simplest ways, namely using the absolute value of the differences for every $i = 1, \ldots, m$. That is,

$$\psi(\langle \mathbf{w}_A, \phi_A(\mathbf{x}_i^A)\rangle + b_A, \langle \mathbf{w}_B, \phi_B(\mathbf{x}_i^B)\rangle + b_B) = |\langle \mathbf{w}_A, \phi_A(\mathbf{x}_i^A)\rangle + b_A - \langle \mathbf{w}_B, \phi_B(\mathbf{x}_i^B)\rangle - b_B|.$$

3 Framework of the Solution

3.1 Problems Arising

Before detailing our approach we need to make some remarks about the problem at hand. First we remove the absolute value by unfolding the synthesis constraint as a pair of constraints for all i,

$$\begin{array}{c} +\langle \mathbf{w}_A, \phi_A(\mathbf{x}_i^A)\rangle + b_A - \langle \mathbf{w}_B, \phi_B(\mathbf{x}_i^B)\rangle - b_B \leq \eta_i + \epsilon, \\ -\langle \mathbf{w}_A, \phi_A(\mathbf{x}_i^A)\rangle - b_A + \langle \mathbf{w}_B, \phi_B(\mathbf{x}_i^B)\rangle + b_B \leq \eta_i + \epsilon, \\ i = 1, \ldots, m. \end{array} \tag{2}$$

Now, let $K^A{=}(K_{ij}^A = \langle \phi_A(\mathbf{x}_i^A), \phi_A(\mathbf{x}_j^A)\rangle$ and $K^B = (K_{ij}^B{=}\langle \phi_B(\mathbf{x}_i^B), \phi_B(\mathbf{x}_j^B)\rangle$, where $i, j{=}1, \ldots, m$. Then the dual of (1) gives

$$\begin{array}{c} \min \tfrac{1}{2}(\mathbf{g}^A)^T K^A \mathbf{g}^A + \tfrac{1}{2}(\mathbf{g}^B)^T K^B \mathbf{g}^B - \mathbf{1}^T \boldsymbol{\alpha}^A - \mathbf{1}^T \boldsymbol{\alpha}^B + \epsilon(\mathbf{1}^T \boldsymbol{\beta}^+ + \mathbf{1}^T \boldsymbol{\beta}^-) \\ \text{with respect to} \\ \boldsymbol{\alpha}^A, \boldsymbol{\alpha}^B, \boldsymbol{\beta}^+, \boldsymbol{\beta}^- \\ \text{subject to} \\ \mathbf{1}^T \mathbf{g}^A = 0,\ \mathbf{1}^T \mathbf{g}^B = 0, \\ \mathbf{g}^A = Y\boldsymbol{\alpha}^A - \boldsymbol{\beta}^+ + \boldsymbol{\beta}^-,\ \mathbf{g}^B = Y\boldsymbol{\alpha}^B + \boldsymbol{\beta}^+ - \boldsymbol{\beta}^-, \\ 0 \leq \boldsymbol{\alpha}^A \leq C^A,\ 0 \leq \boldsymbol{\alpha}^B \leq C^B, \\ 0 \leq \boldsymbol{\beta}^+ + \boldsymbol{\beta}^- \leq D,\ 0 \leq \boldsymbol{\beta}^+,\ 0 \leq \boldsymbol{\beta}^-, \end{array} \tag{3}$$

where $\boldsymbol{\alpha}^A$ and $\boldsymbol{\alpha}^B$ are the dual vector variables belonging to the sub-SVM problems, and $\boldsymbol{\beta}^+$ and $\boldsymbol{\beta}^-$ are the dual vector variables corresponding to the unfolded synthesis constraints. The sign in the superscript of $\boldsymbol{\beta}$ indicates the sign of the expression within the absolute value.

The main difficulties with this problem are;

- The dimensionality of the problem blows up by the factor 4 compared to the single SVM case.
- The constraints (2) are pairwise mutually exclusive at a fixed index i. The consequence of this is an additional constraint set

$$\beta_i^+ \beta_i^- = 0,\ i = 1, \ldots, m \tag{4}$$

which makes the dual feasibility domain be non-convex.

Table (4) shows that applying a general purpose solver to this problem results in excessively long computational times. In the next subsection we show that the structure of the optimization problem has high redundancy which a specialised algorithm can exploit to significantly reduce these times.

3.2 Components

Let us consider first the general form of a quadratic programming problem with linear equality and some additional box constraints.

$$\begin{array}{c} \min_{\mathbf{z}} \frac{1}{2}\mathbf{z}^T Q \mathbf{z} + \mathbf{q}^T \mathbf{z} \\ \text{subject to} \\ A\mathbf{z} = \mathbf{d}, \\ \mathbf{z} \in \hat{\mathcal{Z}}. \end{array} \tag{5}$$

Casting the dual problem (3) into this form we have matrices

$$Q = \begin{bmatrix} YK^AY & \emptyset & -YK^A & YK^A \\ \emptyset & YK^BY & YK^B & -YK^B \\ -YK^A & YK^B & K^A + K^B & -K^A - K^B \\ YK^A & -YK^B & -K^A - K^B & K^A + K^B \end{bmatrix}, \tag{6}$$

$$A = \begin{bmatrix} \mathbf{1}^T Y & \emptyset & -\mathbf{1}^T & \mathbf{1}^T \\ \emptyset & \mathbf{1}^T Y & \mathbf{1}^T & -\mathbf{1}^T \end{bmatrix}, \tag{7}$$

and vectors like these

$$\mathbf{q} = \begin{bmatrix} -\mathbf{1} \\ -\mathbf{1} \\ \epsilon\mathbf{1} \\ \epsilon\mathbf{1} \end{bmatrix}, \mathbf{d} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \mathbf{z} = \begin{bmatrix} \boldsymbol{\alpha}^A \\ \boldsymbol{\alpha}^B \\ \boldsymbol{\beta}^+ \\ \boldsymbol{\beta}^- \end{bmatrix}, \tag{8}$$

and finally the set $\hat{\mathcal{Z}}$ covers the box and the additional complementarity constraints and it is given by

$$\begin{array}{c} \hat{\mathcal{Z}} = \{\mathbf{z} | 0 \leq \boldsymbol{\alpha}^A \leq C^A,\ 0 \leq \boldsymbol{\alpha}^B \leq C^B, \\ 0 \leq \boldsymbol{\beta}^+ + \boldsymbol{\beta}^- \leq D,\ 0 \leq \boldsymbol{\beta}^+,\ 0 \leq \boldsymbol{\beta}^-, \\ \beta_i^+ \beta_i^- = 0,\ i = 1, \ldots, m\}. \end{array} \tag{9}$$

The structure of Q reveals an additional difficulty, namely the third and fourth row of sub-blocks can be expressed as linear combinations of the first and the second row. These relationships can be shown by multiplying the first and second row with Y and computing the difference of the second and the first row giving the third one and the difference of the first and the second row giving the fourth one.

These matrices and vectors comprise highly redundant components. Applying an optimization algorithm exploiting the particular structure of the problem we can gain one or two orders of magnitude reduction in the processing time.

3.3 Reformulation of the Dual

To decrease the dimension of the problem and get rid of the non-convexity a substitution is introduced. One can recognise the dual variables occur pairwise in the objective function as well as in the constraints. Let $\mathbf{u}^+ = \boldsymbol{\beta}^+ + \boldsymbol{\beta}^-$ and $\mathbf{u}^- = \boldsymbol{\beta}^+ - \boldsymbol{\beta}^-$. Exploiting the complementarity constraints (4) then we can derive

$$\mathbf{u}^+ = |\mathbf{u}^-|,\ -D \leq \mathbf{u}^- \leq D. \tag{10}$$

Thus, we can drop the variable $\mathbf{u}_+$ and the feasibility domain becomes convex, but, unfortunately, the objective function has an un-differentiable term $\epsilon(\boldsymbol{\beta}^+ + \boldsymbol{\beta}^-) = \epsilon|\mathbf{u}_N|$. To relax that we replace it with $\epsilon(\mathbf{u}^{-T}\mathbf{u}^-)$. After these modifications the components are changed to

$$Q = \begin{bmatrix} YK^AY & \emptyset & -YK^A \\ \emptyset & YK^BY & YK^B \\ -YK^A & YK^B & K^A + K^B + \epsilon I \end{bmatrix}, \tag{11}$$

where I is the identity matrix, and

$$A = \begin{bmatrix} \mathbf{1}^TY & \emptyset & -\mathbf{1}^T \\ \emptyset & \mathbf{1}^TY & \mathbf{1}^T \end{bmatrix}, \tag{12}$$

furthermore the vectors become

$$\mathbf{q} = \begin{bmatrix} -\mathbf{1} \\ -\mathbf{1} \\ \mathbf{0} \end{bmatrix}, d = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \mathbf{z} = \begin{bmatrix} \boldsymbol{\alpha}^A \\ \boldsymbol{\alpha}^B \\ \mathbf{u}^- \end{bmatrix}. \tag{13}$$

Finally, the box constraints are simplified as well

$$\begin{gathered} \hat{\mathcal{Z}} = \{\mathbf{z} | 0 \leq \boldsymbol{\alpha}^A \leq C^A,\ 0 \leq \boldsymbol{\alpha}^B \leq C^B, \\ -D \leq \mathbf{u}^- \leq D\}. \end{gathered} \tag{14}$$

3.4 Augmented Lagrangian Formulation

We are given a constrained optimization problem

$$\begin{gathered} \min_{\mathbf{z}} f(\mathbf{z}) \\ \text{subject to} \\ h(\mathbf{z}) = 0, \end{gathered} \tag{15}$$

where $\mathbf{z} \in R^n$, f is a scalar and h is a vector valued function. Let $\boldsymbol{\lambda}$ be the Lagrangian multiplier vector corresponding to the constraint $h(\mathbf{z}) = 0$. The Augmented Lagrangian function of (15) is

$$L_c(\mathbf{z}, \boldsymbol{\lambda}) = f(\mathbf{z}) + \boldsymbol{\lambda}^T h(\mathbf{z}) + \tfrac{c}{2}||h(\mathbf{z})||^2, \tag{16}$$

where c is a large positive scalar. By using the Augmented Lagrangian form one can transform a constrained problem into an unconstrained one. It is possible to make a partial elimination where the aim is to reduce the complexity of the set of the constraints into and easy and a hard part. This case gives,

$$\begin{array}{c} \min_{\mathbf{z}} f(\mathbf{z}) \\ \text{subject to} \\ h(\mathbf{z}) = 0, \\ \mathbf{z} \in \hat{\mathcal{Z}}, \end{array} \tag{17}$$

where $\hat{\mathcal{Z}}$ is the easy constraint and $h(\mathbf{z}) = 0$ the hard one. We can then use the Augmented Lagrangian function of (17) to remove the hard constrain, giving

$$\begin{array}{c} \min_{\mathbf{z},\boldsymbol{\lambda}} f(\mathbf{z}) + \boldsymbol{\lambda}^T h(\mathbf{z}) + \frac{c}{2}||h(\mathbf{z})||^2 \\ \text{subject to} \\ \mathbf{z} \in \hat{\mathcal{Z}}, \end{array} \tag{18}$$

The following statement gives the fundamental convergence result explaining how the Augmented Lagrangian can work for a constrained optimization. For a proof and more background the reader is referred to Bertsekas [13]

Proposition 1. *Assume that f and h are continuous functions, that $\hat{\mathcal{Z}}$ is a closed set, and that the constraint set $\{\mathbf{z} \in \hat{\mathcal{Z}} | h(\mathbf{z}) = 0\}$ is nonempty. For $k = 1, \ldots,$ let $\mathbf{z}^k$ be a global minimum of the problem*

$$\begin{array}{c} \min L_{c^k}(\mathbf{z}, \boldsymbol{\lambda}^k) \\ \text{subject to} \\ \mathbf{z} \in \hat{\mathcal{Z}}, \end{array} \tag{19}$$

where $\{\boldsymbol{\lambda}^k\}$ is a bounded sequence of the approximations for the dual variables corresponding to the constraint $h(\mathbf{z}) = 0$, $0 < c^k < c^{k+1}$ for all k, and $c^k \to \infty$. Then every limit point of the sequence $\{\mathbf{z}^k\}$ is a global minimum of the original problem (15).

In order to apply the Augmented Lagrangian approach for our problem we need to modify the matrix Q and the vector q given in the objective function of the original quadratic problem. These modifications are

$$\tilde{Q} = Q + \frac{c}{2} \begin{bmatrix} Y\mathbf{1}\mathbf{1}^T Y & \emptyset & -Y\mathbf{1}\mathbf{1}^T \\ \emptyset & Y\mathbf{1}\mathbf{1}^T Y & Y\mathbf{1}\mathbf{1}^T \\ -\mathbf{1}\mathbf{1}^T Y & \mathbf{1}\mathbf{1}^T Y & 2\mathbf{1}\mathbf{1}^T \end{bmatrix}, \tag{20}$$

and

$$\tilde{\mathbf{q}} = \mathbf{q} + \begin{bmatrix} \lambda_A Y \mathbf{1} \\ \lambda_B Y \mathbf{1} \\ (-\lambda_A + \lambda_B)\mathbf{1} \end{bmatrix}, \tag{21}$$

where λ_A and λ_B are the Lagrangian multipliers corresponding to the constraints $\mathbf{1}^T g^A = 0$ and $\mathbf{1}^T g^B = 0$ respectively

3.5 Algorithm Skeleton

The basic procedure for solving an Augmented Lagrangian problem is as follows,

Step 1. Let $\boldsymbol{\lambda}^0$ be an initial solution for the Lagrangian multipliers, c^0 be a given initial value for the scalar c, γ_c is the multiplier for c, $\epsilon_{\boldsymbol{\lambda}}$ be a required accuracy and $k = 0$.

Step 2. Solve the problem for $\mathbf{z}$ at fixed $\boldsymbol{\lambda}^k$

$$\begin{gathered} \min_{\mathbf{z}} L_{c^k}(\mathbf{z}, \boldsymbol{\lambda}^k) \\ \text{subject to} \\ \mathbf{z} \in \hat{\mathcal{Z}}, \end{gathered} \tag{22}$$

The optimum solution is denoted by $\mathbf{z}^k$.

Step 3. Update the Lagrangian multipliers by $\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + c^k h(\mathbf{z}^k)$ and the running constant by $c^{k+1} = \gamma_c c^k$.

Step 4. If $||h(\mathbf{z}^k)|| < \epsilon_h$
Then Stop!
Else Set $k = k + 1$ and go to Step 2!

The next subsection presents the algorithm for (22)

3.6 Conditional Gradient Method for solving the Lagrangian Subproblem

The conditional gradient method is a simple gradient descent method suited to constrained problems. Let us consider problem (15). Based on the first order optimality condition and assuming $\mathbf{z}_*$ is an optimum solution then the inequality

$$\nabla f(\mathbf{z}_*)(\mathbf{z} - \mathbf{z}_*) \geq 0 \tag{23}$$

has to hold for any $\mathbf{z}$ satisfying $h(\mathbf{z}) = 0$. The basic idea exploited in this algorithm is to find a feasible solution which minimises (23) at an approximation of the optimum solution and hence get closer to the real optimum. The nice fact in our case is that this optimisation problem has a very simple linear form which is solvable in linear time with respect to the dimension of the gradient. The schema of the algorithm reads as follows

Step 1. Let $\mathbf{z}_0^k$ be an initial solution, $\epsilon_z > 0$ be expected accuracy and t=0.

Step 2. Solve the linear programming problem

$$\begin{array}{c} \min_{\mathbf{z}} \nabla L_{c^k}(\mathbf{z}_t^k, \boldsymbol{\lambda}^k)^T(\mathbf{z} - \mathbf{z}_t^k) \\ \text{subject to} \\ \mathbf{z} \in \hat{\mathcal{Z}}. \end{array} \tag{24}$$

Let the optimum solution be denoted by $\mathbf{z}_*^k$

Step 3. Compute the next approximation of the solution using

$$\mathbf{z}_{t+1}^k = \mathbf{z}_t^k + \tau(\mathbf{z}_*^k - \mathbf{z}_t^k), \tag{25}$$

where τ is derived by line search.

Step 4. If $||\nabla L_{c^k}(\mathbf{z}_t^k, \boldsymbol{\lambda}^k)|| \geq -\epsilon_z$ holds

Then Stop!

Else set $t = t + 1$ and go to Step 2

The closed form of the solution for (24) is presented in the next subsection.

3.7 Solution of the Linear Subproblem

The subproblem (24) is a linear programming problem in the form

$$\begin{array}{c} \min_{\mathbf{z}} \mathbf{d}^T\mathbf{z} \\ \text{subject to} \\ \mathbf{z} = \hat{\mathcal{Z}}, \end{array} \tag{26}$$

where $\hat{\mathcal{Z}}$ is defined by the constraints

$$\begin{array}{l} 0 \leq \boldsymbol{\alpha}^A \leq C^A, \\ 0 \leq \boldsymbol{\alpha}^B \leq C^B, \\ -D \leq \mathbf{u}^- \leq D, \end{array} \tag{27}$$

and $\mathbf{d} = L_{c^k}(\mathbf{z}_t^k, \boldsymbol{\lambda}_t^k)$ for a fixed k and t.

Let $\mathbf{d}$ be split into three parts $\mathbf{d} = (\mathbf{d}_{\boldsymbol{\alpha}^A}, \mathbf{d}_{\boldsymbol{\alpha}^B}, \mathbf{d}_{u^-})$ corresponding to the subsets of the vector variable $\mathbf{z}$. The components of an optimum solution can be computed by

$$\boldsymbol{\alpha}_i^A = \begin{cases} C^A & \text{if } (\mathbf{d}_{\boldsymbol{\alpha}^A})_i < 0, \\ 0 & \text{otherwise} \end{cases}, \tag{28}$$

$$\boldsymbol{\alpha}_i^B = \begin{cases} C^B & \text{if } (\mathbf{d}_{\boldsymbol{\alpha}^B})_i < 0, \\ 0 & \text{otherwise} \end{cases}, \tag{29}$$

$$u_i^- = \begin{cases} D & \text{if } (\mathbf{d}_{\mathbf{u}_-})_i < 0, \\ -D & \text{otherwise.} \end{cases}. \tag{30}$$

for any $i = 1, \ldots, m$.

4 Application to Generic Object Recognition

The proposed method was applied to a generic object recognition task for a computer vision problem.

4.1 Data Set and Features

Two data sets were used in our experiments. The first one is a difficult dataset[1] used by Opelt et. al[14]). These images contain the objects at arbitrary scales and poses with highly textured background. There are two categories of objects, persons (P) and bikes (B), and images containing none of these objects (N). We tested the images containing an object (e.g. of categories B and P) against non-object images from the database (e.g. of category N). The performance was measured with the receiver-operating characteristic (ROC) corresponding error rate ([14,15]). The training set contains 100 positive and 100 negative images. The tests are carried out on 100 new images, half belonging to the learnt class and half not.

The second dataset[2] is commonly used for generic object recognition, for example by Opelt et al. [14], Fergus et al. [15] and other papers. The three object classes in this dataset are; motorbikes, aeroplanes and faces. It also contains an additional background class.

For each image two sets of low level features were computed. One[3] used the affine invariant Harris detector[16] developed by K.Mikolajczyk and C.Schmid to detect interest points within an image and Invariant Moment's as patch descriptors. The other used David Lowe's keypoint detector[4] to detect interesting patches with SIFT[17] patch descriptors. These sets of image patch descriptors then form the basis of the feature generation.

Because different images have different numbers of interest points vector quantisation was used to map these sets of points into a fixed length feature vector. Specifically, k-means was used to learn K cluster centers based upon the features from all images. For each image a fixed length K feature vector was then created by recording the minimum distance between an image feature and each of these K centers. In all the following experiments, the parameter for clustering was chosen as $K = 400$.

4.2 Experimental Results

The results were compared to the one in which individual features were the input of the SVM. It is also compared with the SVM solution working on the mixed features, where the feature vectors were concatenated into a high dimensional vector. These results were also compared to the state of the art performance obtained by other methods.

State of the Art Performance. The state of the art performance of these two data sets by using different methods are listed in Table 1 and Table 2. In Table 1, a complex boosting algorithm[14] was used on the Invariant Moment features and SIFT features of dataset 1. In Table 2, the boosting algorithm[14] was applied

[1] Available at http://www.emt.tugraz.at/∼pinz/data/
[2] Available at http://www.robots.ox.ac.uk/∼vgg/data/
[3] Available at $http://lear.inrialpes.fr/people/Mikolajczyk/$.
[4] Available at $http://www.cs.ubc.ca/ \sim lowe/keypoints/$

on dataset 2 and compared to the previous state of the art performance obtained by Fergus et al.[15].

Table 1. Classification accuracy based on boosting algorithm[14] according to ROC Equal Error Rate on dataset 1

Dataset	Moment	SIFT
Bikes	76.5	86.5
Persons	68.7	80.8

Table 2. Classification accuracy based on the algorithms in [15] and[14] according to ROC Equal Error Rate on dataset 2

Dataset	Fergus et al.[15]	Opelt et al.[14]
Motorbikes	92.5	92.2
Airplanes	90.2	88.9
Faces	96.4	93.5

Single and Concatenated Feature-Set SVM Performance. A SVM was trained for each of the Invariant Moment and SIFT features separately and on the features created by concatenating them. The results are listed in the following Table 3.

Results on SVM_2K. In Table 4 the computation times are presented for each data set. This compares a general purpose solver (from the MATLAB Optimisation Toolbox) with the algorithm presented in this paper. These results clearly show that our specialised solver can solve the SVM_2K problem in a very short time without convergence difficulties.

The classification performance of our new method is presented in Table 5. Two sets of results are presented to demonstrate how classification performance is influenced by the setting of various algorithm parameters. The columns labelled "preselected" contain the results corresponding to the fixed training and test sets those are defined in [15]. The columns labelled "random" show the accuracies when 2-fold cross-validation was computed with 5 repetitions on the union of the preselected training and test sets. For the latter case the mean and the standard deviation of the accuracies were computed.

5 Conclusion and Discussion

In this paper we proposed a new method to combine KCCA and SVMs into a single classifier. Real world applications show promising performance with this approach.

Table 3. Classification accuracy based on SVM according to ROC Equal Error Rate on dataset 1

Dataset	Moment	SIFT	Concatenated
Bikes	74.1011	76.0141	75.9375
Persons	75.8411	72.8348	73.9102
Motorbikes	95.318	94.9583	95.0871
Airplanes	92.3925	97	97.25
Faces	97.9527	96.4747	98.5148

Table 4. Computation times for each dataset using different optimization methods

Data sets	MATLAB Optimization Toolbox	Augmented Lagrangian
Bike	1320s	8.5s
Persons	1080s	10.8s
Motorbikes	>1 day(43h)	43s
Aeroplanes	>1 day	42s
Faces	>1 day	65s

Table 5. Classification accuracies computed by the Augmented Lagrangian method

	Accuracies(%): mean(std)			
	$C^A = 1$, $C^B = 1$, $D = 0.5$ $\epsilon = 0.001$		$C^A = 0.2$, $C^B = 0.2$, $D = 0.1$ $\epsilon = 0.001$	
Data sets	Preselected	Random	Preselected	Random
Bike	84.00	76.85(3.75)	81.00	81.65(7.00)
Persons	80.00	72.56(10.80)	82.00	73.74(8.07)
Motorbikes	92.82	90.35(4.68)	93.53	95.06(0.94)
Aeroplanes	97.88	92.59(2.67)	97.88	96.22(1.34)
Faces	99.25	98.75(1.19)	99.10	99.16(0.46)

From Table 3, it is clear that the standard SVM based on individual or concatenated features can give good performance on dataset 2. The results are much better than the previous results. But they are not so good as for dataset 1.

From Table 5, one can see that SVM_2K outperforms both the standard SVM and other approaches – especially, for data set 2. Meanwhile, the algorithms based on Augmented Lagrangian method provides efficient implementation.

The experiments show that the performance for different feature sets is highly dependant on the choice of penalty parameters. It claims further investigation on finding an optimal or a nearly optimal configuration for a given learning problem.

To compare the SVM_2K approach correctly with the standard SVM, theoretical analysis of the statistical learning theory based generalisation performance should be studied.

Acknowledgements

This work is supported by the European project LAVA Num. IST-2001-34405.

References

1. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines (and other kernel-based learning methods). Cambridge University Press, Cambridge, UK (2000)
2. Kolenda, T., Hansen, L.K., Larsen, J., Winther, O.: Independent component analysis for understanding multimedia content. In Bourlard, H., Adali, T., Bengio, S., Larsen, J., Douglas, S., eds.: Proceedings of IEEE Workshop on Neural Networks for Signal Processing XII, Piscataway, New Jersey, IEEE Press (2002) 757–766 Martigny, Valais, Switzerland, Sept. 4-6, 2002.
3. Hardoon, D.R., Shawe-Taylor, J.: KCCA for different level precision in content-based image retrieval. In: Proceedings of Third International Workshop on Content-Based Multimedia Indexing, IRISA, Rennes, France (2003)
4. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press, Cambridge, UK (2004)
5. Vinokourov, A., Shawe-Taylor, J., Cristianini, N.: Inferring a semantic representation of text via cross-language correlation analysis. In: Advances of Neural Information Processing Systems 15. (2002)
6. Vinokourov, A., Hardoon, D.R., Shawe-Taylor, J.: Learning the semantics of multimedia content with application to web image retrieval and classification. In: Proceedings of Fourth International Symposium on Independent Component Analysis and Blind Source Separation, Nara, Japan (2003)
7. Meng, H., Hardoon, D.R., Shawe-Taylor, J., Szedmak, S.: Generic object recognition by distinct features combination in machine learning. In: Proceedings of SPIE, Vol.5673. (Jan., 2005)
8. Lanckriet, G.R., Cristianini, N., Ghaoui, P.B.L.E., Jordan, M.I.: Learning the kernel matrix with semidefinite programming. Journal of machine learning research (2004) 27–72
9. Dasgupta S., Littman M.L., McAllester D. : PAC generalization bounds for co-training. Advances in Neural Information Processing Systems (NIPS). (2001)
10. Lanckriet, G., Deng, M., Cristianini, N., Jordan, M., Noble, W.: Kernel-based data fusion and its application to protein function prediction in yeast. In: Proceedings of the Pacific Symposium on Biocomputing. (2004) 300–311
11. Bach, F., Lanckriet, G., Jordan, M.: Multiple kernel learning, conic duality, and the smo algorithm. In: Proceedings of the 21st International Conference on Machine Learning, Canada (2004)
12. Andrews, S., Tsochantaridis, I., Hofmann, T.: Support vector machines for multiple-instance learning. In: 15th Advances in Neural Information Processing Systems. (2002)

13. Bertsekas, D.: Nonlinear Programming. Second edition edn. Athena Scientific (1999)
14. Opelt, A., M.Fussenegger, Pinz, A., Auer, P.: Weak hypotheses and boosting for generic object detection and recognition. In: Proceedings of the 2004 European Conference on Computer vision, "Prague Czech Republic" (2004) 71–84
15. Fergus, R., Perona, P., Zisserman, A.: Object class recognition by unsupervised scale-invariant learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2003)
16. Mikolajczyk, K., Schmid, C.: An affine invariant interest point detector. In: Proceedings of the 2002 European Conference on Computer vision, "Copenhagen Denmark" (2002) 128–142
17. Lowe, D.: Object recognition from local scale-invariant features. In: Proceedings of the 7th IEEE International Conference on Computer vision, "Kerkyra Greece" (1999) 1150–1157
18. Blum A., Mitchell T.: Combining Labeled and Unlabeled Data with Co-Training. Proceedings of the 11th Annual Conference on Computational Learning Theory, (1998) 92–100
19. Evgeniou T., Micchelli C.A., Pontil M. : Learning multiple tasks with kernel methods. Journal of Machine Learning Research, 6(Apr) (2005) 615–637
20. Muslea I., Minton S., Knoblock C.A. : Active + semi-supervised learning = robust multi-view learning. In Proceedings of the 19th International Conference on Machine Learning (ICML-2002), Sydney, Australia, (2002) pages 435–442

Appropriate Kernel Functions for Support Vector Machine Learning with Sequences of Symbolic Data

Bram Vanschoenwinkel* and Bernard Manderick

Vrije Universiteit Brussel,
Computational Modeling Lab,
Pleinlaan 2, 1050 Brussel, Belgium
{bvschoen, bmanderi}@vub.ac.be
http://como.vub.ac.be/

Abstract. In classification problems, machine learning algorithms often make use of the assumption that (dis)similar inputs lead to (dis)similar outputs. In this case, two questions naturally arise: what does it mean for two inputs to be similar and how can this be used in a learning algorithm? In support vector machines, *similarity* between input examples is implicitly expressed by a kernel function that calculates inner products in the feature space. For numerical input examples the concept of an inner product is easy to define, for discrete structures like sequences of symbolic data however these concepts are less obvious. This article describes an approach to SVM learning for symbolic data that can serve as an alternative to the bag-of-words approach under certain circumstances. This latter approach first transforms symbolic data to vectors of numerical data which are then used as arguments for one of the standard kernel functions. In contrast, we will propose kernels that operate on the symbolic data directly.

1 Introduction

The similarity between examples of a set of data often gives much information about the patterns that may be present in that data. Therefore some machine learning (ML) algorithms try to make use of as many similarity information about the input examples as possible. One of the best known examples is the k-nearest neighbours algorithm in which a metric defined on the input examples determines the distance between a new unseen example and the examples stored in memory. This new example is then assigned to one of the classes based on the k nearest examples in memory. In these applications the choice of metric is a very important one as it will have a direct influence on the classification of a new example. Therefore it is important to choose a metric in function of the relevant characteristics of the application under consideration. For real input

* Author funded by a doctoral grant of the institute for advancement of scientific technological research in Flanders (IWT).

J. Winkler, N. Lawrence, M. Niranjan (Eds.): Machine Learning Workshop, LNAI 3635, pp. 256–280, 2005.

examples the Euclidean distance is often used, for discrete data like strings, sequences or natural language data the choice of metric is often much harder to make as the concept of a distance is not always easy to define in this context. Moreover, once we have a similarity measure on the input examples it will not always be directly usable in the learning algorithm as it is the case for k-nearest neighbours. In some algorithms the similarity measure is not directly observable and subject to a number of conditions imposed by the learning algorithm. In that sense it is not always easy to use the best available measure in the learning algorithm.

In support vector machine (SVM) learning, similarity information is implicitly contained in the kernel function. Positive definite (PD) kernel functions are related to one of the most simple similarity measures available: inner products. In practice however, we often have dissimilarity measures under the form of distance functions, the larger the distance the larger the dissimilarity. Dissimilarity information can be incorporated in a PD kernel through the normed distance between examples or by considering the class of conditionally positive definite (CPD) kernels, which are kernels that calculate generalized distances in the feature space [1]. Concepts like inner products and metrics are clearly understood and straightforward in the case of real input examples, for discrete examples like strings these concepts are not straightforward. Moreover when working with large amounts of textual data there is also the issue of computational performance. Therefore, in SVM learning, symbolic data is often first transformed to real data and then a standard kernel working on the transformed input examples is used to do classification. This approach is generally known as the bag-of-words (BOW) approach and it was first introduced to do classification of large amounts of text [2].

The problems we will consider here are 1) *language independent named entity recognition* (LINER) and 2) *protein secondary structure prediction* (PSSP). In LINER it is the purpose to determine for all the words in a text whether the word refers to a proper name or not, and, if it refers to a proper name it will also be indicated what kind of proper name, i.e. name of a person, organization, location etc. In PSSP it is the purpose to decide for each amino acid in a sequence of amino acids whether it belongs to a α-helix, β-sheet or a coil. Each instance of both problems can be represented by a sequence of symbols where each symbol belongs to a given dictionary, moreover we will look at the context of each symbol in a given sequence, i.e. the p symbols before and the q symbols after that symbol. Additionally, in the case of LINER we will extend a context by using additional features, like part-of-speech tags for example. In that case we talk about an extended context. In the case of LINER a symbol can be a word, a proper name or a punctuation mark etc., the dictionary contains all words of the language in consideration together with proper names, punctuation marks etc. A sequence of symbols is a sentence and the context of a given word are the p words in front and the q words after the given word. For PSSP a symbol represents an amino acid, the dictionary is the set of all existing amino acids, a sequence is (part of) a protein formed by a succession of amino acids and the

context of a given amino acid are the p acids in front and the q acids after the given acid.

Because of its great success the BOW approach quickly became very popular. Although the approach was designed to do classification of large texts, variants have been described to do classification of the symbols in a sequence in the way described above. In this way however a lot of useful information can get lost because a standard kernel is often not specifically designed to capture the similarity between contexts. Therefore it can be worthwhile to look for metrics that operate directly on such contexts. It is our purpose to show that an alternative approach to SVM learning for applications where the data can be represented by sequences of (extended) contexts can be used instead of the BOW approach.

We start this work with a short description of SVM theory from a kernel viewpoint, as kernel functions will be the main topic of this paper. Next we describe an alternative approach to SVM learning with symbolic data that gives a much more intuitive basis with respect to the similarity of the input examples. We start by describing a variant of a simple pseudo-inner product that has been introduced in [3] and is based on a very simple metric defined on contexts. We will show that this inner product is in fact equivalent to the standard inner product applied to the orthonormal representation of these contexts. But, in the same time we will show that the discrete approach based on the similarity between contexts is a much more intuitive viewpoint to incorporate special purpose, complex similarity measures into the learning process. Two inner products, with increasingly complex similarity measures will be considered and for both cases it will be shown that they satisfy the necessary conditions to be used in a kernel function. One such inner product will be based on a weighted version of the simple metric and the second one will be based on the entries of a similarity matrix.

Finally, a number of the described kernel functions will be applied to the problem of PSSP, compared with each other and a number of standard kernels. From the results it will become clear that using special purpose similarity information, in the way proposed in this paper, has a positive effect on the classification results.

2 Support Vector Machines

The following sections give a brief overview of the theory of SVMs together with a discussion of different classes of kernel functions. Next, because we want to make kernels based on specific (dis)similarity information about the input examples, we will show the importance of and the relation between similarity measures, distances and kernels. Subsequently, all introduced concepts will be used in Section (4) to construct kernels based on the (dis)similarity between contexts. Note that, for simplicity we will only consider the case of binary classification. For a more comprehensive description of SVMs we refer to [4,5].

2.1 Maximum-Margin Separation

Consider an input space X with input vectors $\mathbf{x}$, a target space $Y = \{1, -1\}$ and a training set $Tr = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$ with $\mathbf{x}_i \in X$ and $y_i \in Y$.

In SVM classification, separation of the two classes $Y = \{1, -1\}$ is done by means of the *maximum margin* hyperplane, i.e. the hyperplane that maximizes the distance to the closest data points and guarantees the best generalization on new, unseen examples. Let us consider two hyperplanes :

$$\langle \mathbf{w}, \mathbf{x}_i \rangle + b \geq 1 \quad \text{if } (y_i = 1) \tag{1}$$

$$\langle \mathbf{w}, \mathbf{x}_i \rangle + b \leq -1 \quad \text{if } (y_i = -1) \tag{2}$$

The distance from the hyperplane to a point $\mathbf{x}_i$ can be written as :

$$d(\mathbf{w}, b; \mathbf{x}_i) = \frac{|\langle \mathbf{w}, \mathbf{x}_i \rangle + b|}{\|\mathbf{w}\|}$$

Consequently the margin between two hyperplanes can be written as :

$$\min_{\mathbf{x}_i; y_i = 1} d(\mathbf{w}, b; \mathbf{x}_i) + \min_{\mathbf{x}_i; y_i = -1} d(\mathbf{w}, b; \mathbf{x}_i)$$

To maximize this margin we have to minimize $\|\mathbf{w}\|$. This comes down to solving a quadratic optimization problem with linear constraints. Notice however that we assumed that the data in Tr are perfectly linear separable. In practice however this will often not be the case. Therefore we employ the so called *soft-margin* method in contrast to the *hard-margin* method. Omitting further details we can rewrite the soft-margin optimization problem by stating the hyperplane in its dual form, i.e. find the Lagrange multipliers $\alpha_i \geq 0$ $(i = 1, ..., N)$ so that :

$$\begin{aligned} \text{Maximize: } & L(\alpha_1, \cdots, \alpha_N) = \textstyle\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \\ \text{Subject to: } & \textstyle\sum_{i=1}^{N} \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq C \end{aligned} \tag{3}$$

Considering the dual problem above we can now write the maximum margin hyperplane as a linear combination of support vectors. By definition the vectors $\mathbf{x}_i$ corresponding with non-zero α_i are called the *support vectors* SV and this set consists of those data points that lie closest to the hyperplane and thus are the most difficult to classify. In order to classify a new point $\mathbf{x}_{new}$, one has to determine the sign of

$$\sum_{\mathbf{x}_i \in SV} \alpha_i y_i \langle \mathbf{x}_i, \mathbf{x}_{new} \rangle + b \tag{4}$$

If this sign is positive $\mathbf{x}_{new}$ belongs to class 1, if negative to class -1, if zero $\mathbf{x}_{new}$ lies on the decision boundary. Note that we have restricted the summation to the set SV of support vectors because the other α_i are zero anyway.

2.2 The Kernel Trick

In practice it will often be the case that the data can not be separated linearly by means of a hyperplane. One of the basic ideas behind SVMs is to have a mapping ϕ from the original input space X into a high-dimensional *feature space* F that

is a Hilbert space, i.e. a complete vector space provided with an inner product. Separation of the transformed feature vectors $\phi(\mathbf{x}_i)$ in F is done linearly, i.e. by a hyperplane. The decision boundary which is linear in F corresponds to a non-linear decision boundary in X.

However, transforming the vectors in the training set Tr into such a higher-dimensional space incurs computational problems. The high dimensionality of F makes it very expensive both in terms of memory and time to represent the feature vectors $\phi(\mathbf{x}_i)$ corresponding to the training vectors $\mathbf{x}_i$. Moreover, it might be very hard to find the transformation ϕ that separates linearly the transformed data.

Notice that the objective function $L(\alpha_1, ..., \alpha_N)$ in (3) and the definition of the hyperplane in (4) depend only on inner products between vectors. If there would exist a function that allowed us to directly calculate the inner products between the transformed feature vectors $\phi(\mathbf{x}_i)$ from the $\mathbf{x}_i$ without actually having to consider $\phi(\mathbf{x}_i)$, we can reduce the computational complexity considerably. It turns out that such a function exists, it is called a *kernel function* and it is defined as follows [5]:

Definition 1. *A* kernel *is a symmetric function $K : X \times X \to \mathbb{R}$ so that for all $\mathbf{x}_i$ and $\mathbf{x}_j$ in X, $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$ where ϕ is a (non-linear) mapping from the input space X into the Hilbert space F provided with the inner product $\langle ., . \rangle$.*

2.3 Different Classes of Kernel Functions

The selection of an appropriate kernel K is the most important design decision in SVMs since it implicitly defines the feature space F and the map ϕ. A SVM will work correctly even if we don't know the exact form of the features that are used in F. Moreover, the kernel expresses prior knowledge about the patterns being modeled, encoded as a similarity measure between two input vectors.

But not all symmetric functions over $X \times X$ are kernels that can be used in a SVM. Since a kernel K is related to an inner product, cfr. the definition above, it has to satisfy some conditions that arise naturally from the definition of an inner product and are given by Mercer's theorem: the kernel function has to be positive definite (PD). Therefore, we have the following definition:

Definition 2. *A symmetric function $K : X \times X \to \mathbb{R}$ which for all $m \in \mathbb{N}$, $\mathbf{x}_i$, $\mathbf{x}_i \in X$ gives rise to a positive semi-definite (PSD) kernel matrix, i.e. for which for all $c_i \in \mathbb{R}$ we have:*

$$\sum_{i,j=1}^{m} c_i c_j \mathbf{K}_{ij} \geq 0, \text{ where } \mathbf{K}_{ij} = K(\mathbf{x}_i, \mathbf{x}_j) \tag{5}$$

is called a positive definite (PD) kernel.

When K is not PD it will be unclear what kind of classification problems we are solving and convexity of the optimization problem can no longer be guaranteed.

In practice, the requirement of a kernel to be PD turns out to be a very strict assumption. Many special purpose or sophisticated similarity and dissimilarity measures that one would like to incorporate in the learning process do not satisfy the requirement of being PD.

One particular class of non-PD kernels is the class of *conditionally positive definite* (CPD) kernels. For this type of kernel functions it has been shown that they can be used as generalized distances in the feature space [1]. Section (2.4) will describe one such kernel function known as the negative distance kernel. For now we start by giving a definition of the class of CPD kernels:

Definition 3. *A symmetric function $K : X \times X \to \mathbb{R}$ which satisfies (5) for all $m \in \mathbb{N}$, $\mathbf{x}_i \in X$ and for all $c_i \in \mathbb{R}$ with the extra condition:*

$$\sum_{i=1}^{m} c_i = 0, \tag{6}$$

is called a conditionally positive definite (CPD) kernel.

2.4 Kernel Functions and Similarity Measures

Next, different relations between kernel functions and similarity measures are described. It will be shown that there is a very close relationship between kernel functions, similarity measures and metrics. We start with the trivial case of PD kernels and their relation to the concept of similarity and we show how a distance can be derived from a PD kernel. Next, it is shown that CPD kernels can be used to define distances in the feature space.

Notice that in SVMs similarity between examples is measured by inner products, through the calculation of PD kernels. We start by giving an interesting property that relates a PD kernel function $K : X \times X \to \mathbb{R}$ to a metric d on the input space X:

Distance derived from a PD kernel: *Let K be a PD kernel over $X \times X$, then d defined as :*

$$\begin{aligned} d(\mathbf{x}, \mathbf{x}') &= \|\phi(\mathbf{x}) - \phi(\mathbf{x}')\| \\ &= \sqrt{K(\mathbf{x}, \mathbf{x}) - 2K(\mathbf{x}, \mathbf{x}') + K(\mathbf{x}', \mathbf{x}')} \end{aligned} \tag{7}$$

is a distance on X and consequentially (X, d) is a metric space.

Although PD kernels define inner products, they can also use a form of dissimilarity in their calculation. An example of this is the *radial basis kernel* function $K_{rb} : X \times X \to \mathbb{R}$ that explicitly makes use of the distance between two points in the input space X to calculate inner products in the feature space F:

$$K_{rb}(\mathbf{x}, \mathbf{x}') = exp\left(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2\right) \tag{8}$$

In general $\|\mathbf{x} - \mathbf{x}'\|$ can be substituted by any metric that calculates the distance between $\mathbf{x}$ and $\mathbf{x}'$, the kernel that is so defined is called a generalized radial basis function [6]. In this way one can plug in a special purpose distance function for the application in consideration.

Next, let us focus on the class of CPD kernels. The following will state only those results that are needed to justify the approach in this work, for more details we refer to [1] and [7]. We start by considering the connection between PD and CPD kernels:

Proposition 1. *Let $\mathbf{x_0} \in X$, and let K be a symmetric kernel on $X \times X$. Then $\tilde{K}(\mathbf{x}, \mathbf{x}') = K(\mathbf{x}, \mathbf{x}') - K(\mathbf{x}, \mathbf{x_0}) + K(\mathbf{x_0}, \mathbf{x_0})$ is positive definite if and only if K is conditionally positive definite.*

Proof. For the proof of this proposition we refer to [1].

Next, based on Equation (1) we can construct a feature map for K. Because $\tilde{K}$ is PD we can employ the Hilbert space representation $\phi : X \to F$, with $\langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle = \tilde{K}(\phi(\mathbf{x}), \phi(\mathbf{x}'))$:

$$\|\phi(\mathbf{x}) - \phi(\mathbf{x}')\|^2 = \tilde{K}(\mathbf{x}, \mathbf{x}) + \tilde{K}(\mathbf{x}', \mathbf{x}') - 2\tilde{K}(\mathbf{x}, \mathbf{x}')$$

Substituting (1) yields,

$$\|\phi(\mathbf{x}) - \phi(\mathbf{x}')\|^2 = -K(\mathbf{x}, \mathbf{x}') + \frac{1}{2}\left(K(\mathbf{x}, \mathbf{x}) + K(\mathbf{x}', \mathbf{x}')\right) \tag{9}$$

Now we have the following result that relates a CPD to a metric on X, comparable to the result in (7):

Distance Derived from a CPD Kernel: *Let K be a CPD kernel over $X \times X$, satisfying $K(\mathbf{x}, \mathbf{x}) = 0$ for all $\mathbf{x} \in X$, then d defined as :*

$$\begin{aligned} d(\mathbf{x}, \mathbf{x}') &= \|\phi(\mathbf{x}) - \phi(\mathbf{x}')\| \\ &= \sqrt{-K(\mathbf{x}, \mathbf{x}')} \end{aligned} \tag{10}$$

is a distance on X and consequentially (X, d) is a metric space.

Notice that the second term of (9) is dropped in (10) because we assumed that $K(\mathbf{x}, \mathbf{x}) = 0$. In the case that this assumption would not be true, (10) should be adapted accordingly and d would no longer be a metric but a *semi-metric*. One CPD kernel function that we will consider here is the negative distance (ND) kernel [5]:

$$K_{ND}(\mathbf{x}, \mathbf{x}') = -\|\mathbf{x} - \mathbf{x}'\|^{\beta} \text{ with } 0 < \beta \leq 2 \tag{11}$$

3 Implications of Working with Sequences of Symbols

In the previous we always assumed that the vectors $\mathbf{x}$ and $\mathbf{x}'$ were vectors belonging to a real vector space X. In this section however we will consider data

that are sequences of symbols and the context of each symbol in such a sequence, i.e. a fixed number of symbols before and after that symbol in the sequence, see Section (3.1).

One of the most obvious implications of working with sequences is that we will have to define an "inner product" between contexts. One popular approach to this problem is to transform sequences to a real representation and then applying the standard SVM and kernel functions, this is explained in Section (3.2). Finally, in Section (3.3) we give an alternative approach to SVM learning with sequences. We will argue that especially for contexts that result from sliding a window over a sequence of symbols, the use of special purpose, distance based kernel functions working on the contexts themselves should be preferred over the use of standard kernel functions working on the transformed real data.

3.1 Contexts

Consider a collection S of sequences $\mathbf{s}$. Every sequence $\mathbf{s}$ consists of an ordered succession of symbols, i.e. $\mathbf{s} = \left(s_{k_0} \dots s_{k_{|\mathbf{s}|-1}}\right)$ with $|\mathbf{s}|$ the length of the sequence and with $s_{k_i} \in D$ a set (henceforth called a dictionary) of symbols indexed according to $k_i \in \{1, \dots, n\}$ with $|D| = n$ the cardinality of the dictionary D and $i = 0, \dots, |\mathbf{s}| - 1$. Contexts are now formed by sliding a window over the sequences $\mathbf{s} \in S$, i.e. for every sequence $\mathbf{s}$ a set of instances $I(\mathbf{s})$ containing $|\mathbf{s}|$ contexts with a window size $r = (p + q + 1)$ is constructed as follows $I(\mathbf{s}) = \{\mathbf{s}[(i-p) : (i+q)] \mid 0 \leq i \leq |\mathbf{s}| - 1\}$ with p the size of the left context, q the size of the right context and with $\mathbf{s}[i : j] = \left(s_{k_i} \dots s_{k_j}\right)$ the subsequence of symbols from index i to j in the sequence $\mathbf{s}$. The total set of contexts is now formed by taking the union of all the $I(\mathbf{s})$, i.e. $I(S) = \bigcup_S I(\mathbf{s})$. Notice that for subsequences with indices $i < 0$ and $j > |\mathbf{s}| - 1$ corresponding positions in the sequence are filled with the special symbol ' $-$ ' which can be considered as the empty symbol. In the following we will give an example of this in the setting of language independent named entity recognition (LINER).

Example 1. In LINER it is the purpose to distinguish between different types of named entities. Named entities are phrases that contain the names of persons, organizations, locations, times and quantities. Consider the following sentence: [B-PER Wolff] , currently a journalist in [B-LOC Argentina] , played with [B-PER Del I-PER Bosque] in the final years of the seventies in [B-ORG Real I-ORG Madrid]. Here we recognize 5 types of named entities (B-PER, I-PER, B-LOC, B-ORG and I-ORG), all other words are tagged as O (outside named entity). Instances for such classification problems are often composed by the word for which we want to determine the entity class and a context of a number of words before and after the word itself. Furthermore, sometimes additional features, like the position of a word in a sentence, are used, but we will not consider these features here. For simplicity we will use a context of 2 words on the left and the right of the focus word (i.e. a window size of 5), in our setting we now have:

1. The dictionary D is the set of all English words and names of persons, organizations etc.

2. The set of sequences S is a set of sentences, i.e. we consider a sentence as being a sequence with the words as components. Notice that in this example there is only one sequence in S as there is only one sentence.
3. For the above sentence we have following sequence $\mathbf{s}$ = Wolff , currently a journalist in Argentina , played with Del Bosque in the final years of the seventies in Real Madrid, with $|\mathbf{s}| = 23$.

Next, the set of contexts $I(S)$ is defined as:

$$\begin{aligned}\bigcup_S I(\mathbf{s}) &= \{\mathbf{s}[(i-2):(i+2)] \mid \ 0 \leq \ i \leq \ 22\ \} \\ &= \{\mathbf{s}[-2:2],\ \mathbf{s}[-1:3], \mathbf{s}[0:4], \ldots, \mathbf{s}[18:22], \mathbf{s}[19:23], \mathbf{s}[20:24]\} \\ &= \{(-\ -\ Wolff\ ,\ currently), (-\ Wolff\ ,\ currently\ a), (Wolff\ , \\ &\quad currently\ a\ journalist),\ \ldots\ ,\ \ldots\ , (seventies\ in\ Real\ Madrid), \\ &\quad (in\ Real\ Madrid-), (Real\ Madrid-\ -)\}\end{aligned}$$

Notice that the '$-$' represent values that are not present because they fall before the beginning or after the end of a sequence.

Next, it is illustrated how the same approach can be applied to *protein secondary structure prediction* (PSSP).

Example 2. Proteins are sequences of amino acids joined together by peptide bonds. There are 20 different amino acids that make up all proteins on earth. Amino acids are represented by a one letter alphabet, in total 20 letters, one for every different amino acid that exists. The order of the amino acids in a sequence is known as its primary structure. Secondary structure is the term protein chemists give to the arrangement of the peptide backbone in space. The backbone can form regular, repeating structures held together due to interactions between chemical groups on the amino acids. These repeating structures are called secondary structure. In PSSP it is the purpose to decide for each amino acid of the protein whether it belongs to an α-helix, a β-sheet or a coil. Contexts are formed in a way similar to the LINER problem, i.e. we take a central amino acid and a context of a number of amino acids in front and after the acid for which we want to determine the secondary structure, this time we will consider a context of 5 acids both to the left and the right of the focus acid:

1. The dictionary D with $|D| = 20$, is the set of all amino acids, i.e. $D = \{A, C, D, E, F,\ G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y\}$
2. The set of sequences S is the set of all amino acid sequences corresponding to existing proteins or multiple alignments of such proteins [8].

Next consider the following protein (taken from the CB513 data set, see Section (5.2)) A Y V I N D S C I A C G A C K P E C P V N I I Q G S I Y A I D A D S C I D C G S C A S V C P V G A P N P E D $\in S$, from this sequence we derive following set of contexts $I(S)$:

$$\begin{aligned}\bigcup_S I(\mathbf{s}) &= \{\mathbf{s}[(i-5):(i+5)] \mid 0 \leq i \leq 53\} \\ &= \{\mathbf{s}[-5:5], \mathbf{s}[-4:6], \mathbf{s}[-3:7], \ldots, \mathbf{s}[51:56], \mathbf{s}[52:57], \mathbf{s}[53:58]\} \\ &= \{(-,-,-,-,-,A,Y,V,I,N,D), (-,-,-,-,A,Y,V,I,N,D,S), \\ &\quad (-,-,-,A,Y,V,I,N,D,S,C), \ldots, (A,Y,V,I,N,D,S,C,I,A,C), \\ &\quad (Y,V,I,N,D,S,C,I,A,C,G), \ldots, (G,A,P,N,P,E,D,-,-,-,-), \\ &\quad (A,P,N,P,E,D,-,-,-,-,-)\}\end{aligned}$$

Finally, in many cases one would like to use additional features next to the symbols in the sequences themselves. These additional features are defined in function of the symbols in the sequences and possibly some other information, like the position of the symbol in the sequence for example. The possible values the new features can adopt are added to the dictionary D and indexed accordingly. The new features themselves can be added everywhere in the context as long as they are put on corresponding positions in all contexts. The most simple way is to just add them at the end of the contexts as they are formed by sliding the window over the original sequences. Consider the following example:

Example 3. In the case of LINER we distinguish 3 types of additional features:

1. *Part-of-speech (POS) tags*: POS tags are defined for every symbol in the original dictionary D and describe what type of word we are considering, e.g. V(erb), N(oun), etc. This feature is defined by the function $f_{POS} : D \rightarrow P$ that maps every symbol from D to some POS tag p from the set P of all possible POS tags. This feature is defined for every symbol in the original context $I(\mathbf{s})$.
2. *Orthographic* feature: Like information about the capitalization of a word, digits or hyphens in a word, etc. This feature is defined by the function $f_{ORT} : D \rightarrow O$. This feature is also defined for every symbol in the original context $I(\mathbf{s})$.
3. *Position* feature: This feature is binary and tells something about the position of the focus word in the original sequence, i.e. whether the focus word is the first word in the sentence or not. This feature is defined not only in function of the set D but also in function of the position of the symbol in the original sequence $\mathbf{s}$, i.e. $f_{wp} : D \times \{1 \ldots |\mathbf{s}| - 1\} \rightarrow \{fw, ow\}$ with fw referring to first word and ow referring to other word.

The total dictionary D_{tot} is now formed by $D \bigcup P \bigcup O \bigcup \{fw, ow\}$ and an extended context based on a sequence $\mathbf{s}$ is made out of 4 blocks as follows:

$$(s_{k_i}, \ldots, s_{k_j}, f^i_{POS}, \ldots, f^j_{POS}, f^i_{ORT}, \ldots, f^j_{ORT}, f_{wp})$$

From now on we will consider D to be the dictionary containing all symbols including the values of the additional features. Moreover, note that for the following it does not matter whether we are working with contexts or extended contexts. Therefore, from now on we will not distinguish between contexts and extended contexts.

Next, different ways to represent contexts, in a way suitable for a SVM to work with them, will be described. For this purpose we will show different ways to represent the contexts as vectors $\mathbf{x}, \mathbf{x}' \in X$.

3.2 Bag-of-Words and Orthonormal Vectors

The bag-of-words (BOW) approach involves a transformation of the discrete instances to a real vector representation [2] and subsequently applying the standard kernel functions to the resulting real vectors. In this case we don't slide a window over a sequence of symbolic data with the purpose to assign every symbol of the sequence to a class (like in LINER). Here it is more the purpose to assign complete texts to different classes, in this context a complete text with (possibly many) different sentences or sequences is considered as one instance. In the bag-of-words vector representation these instances are described by a vector in which each component of the vector represents the value of one feature of that instance. Each symbol that occurs in the data is treated as one such feature. The value of a feature for a particular example can be, for example, the frequency of occurrence of that symbol or it can take a binary form taking the value of 1 if the symbol occurs in the instance and the value of 0 if it doesn't occur in the instance [2]. Note that this approach is based on the assumption that the order in which the symbols occur does not play an important role in the determination of the classification problem. Also note that the dimensionality of the input vectors becomes very high in this way because we have a vector component for every distinct symbol occurring in the data. Moreover, most of the components will take a value of 0 since only a few symbols of the total number of symbols of the data will be present in one example. However, this does not impose any problems for the SVM as it has been proved that it is capable of learning with very high dimensional input data [2]. To handle the sparseness of the input vectors many SVM implementations have been specifically designed so that only the non-zero components of the input vectors should be represented in the data. In the past the BOW approach has been successfully applied to the classification of large texts. The approach was very successful for this type of tasks because the word order in a text does not seem to be important for the classification of the text as features like term frequency give much more relevant information. Moreover, for large texts, working directly with the strings could cause computational problems [9,2].

For other problems however the order in which the symbols appear does play an important role. One such an example is LINER, because here the word order in the sentence plays an important role in the determination of the entity class. Also, for PSSP the order of the amino acids is very important. It is obvious that for these problems the BOW will not work. For these problems it is necessary to include information about the position in the transformation to the real format. This is done by encoding each symbol from the set D as an orthonormal vector. We illustrate this by the following example:

Example 4. For the PSSP problem of Example 2 every amino acid is represented by a 20-dimensional orthonormal vector with only 1 non-zero component, i.e.

the component corresponding to the index of the symbol it represents in the dictionary D. Note that the empty symbol ' − ' is represented by the zero-vector. If we assume that the symbols (or amino acids in this example) are indexed alphabetically in D then we have following orthonormal vectors:

$$
\begin{array}{ll}
-= & (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0) \\
\text{A}= & (1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0) \\
\text{C}= & (0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0) \\
\ldots & \ldots\ldots\ldots \quad \ldots\ldots\ldots \quad \ldots\ldots\ldots \quad \ldots\ldots\ldots \\
\text{W}= & (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0) \\
\text{Y}= & (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1)
\end{array}
$$

In this context vectors $\mathbf{x}, \mathbf{x}' \in X$ with $X \subseteq \mathbb{R}^l$ the space of orthonormal vectors of dimensionality $l = r * n$, where r is the context size and n is the size of the dictionary, represent complete contexts formed by concatenating the above vectors according to the succession of the different amino acids in the order as they appear in the sequence.

Because the position of the symbols in the sequence can now be encoded into real vectors, the method also becomes available to problems that depend on the order of the symbols in the sequence. In general the data is recorded to the orthonormal format and represented by a sparse vector format, i.e. 0-valued components are not represented, and subsequently standard kernel functions like the radial basis or polynomial kernel functions are applied. This approach has been successful for different types of applications, like LINER, PSSP and other problems in both natural language and biology that rely on sliding a window over a sequence of symbols as explained above [10,11,12,13,14].

Although the method has been successful for several applications it suffers from a number of drawbacks:

1. It is only possible to see whether two symbols occur at the same position in two contexts, it is not possible to measure a degree of similarity between the symbols.
2. By using standard kernel functions useful information like metrics defined on the contexts can get lost. In Section (2.4) we saw the close relation between and importance of similarity measures and kernel functions. These similarity measures are defined on the input data, i.e. on strings, sequences and contexts and not on orthonormal vectors.
3. An orthonormal vector does not give much information about the context it represents, i.e. it is very hard to see from which context it has been derived.

For these reasons we believe that, for applications with symbolic data and with contexts that are formed by sliding a window over a sequence of symbols it is better to work directly with the contexts themselves.

3.3 Contexts

Contexts $\mathbf{x}, \mathbf{x}' \in X$ are vectors with $X = \bigcup_S I(\mathbf{s}) \subseteq \mathbb{S}^l$ the space of contexts of size l. The vectors $\mathbf{x}, \mathbf{x}'$ represent contexts as discussed in Section (3.1) and as

illustrated in examples 1 and (2) and take the exact same form as the contexts themselves. In this light $\mathbb{S}^l$ can be considered as the space of all contexts formed by sliding a window over all possible sequences formed by the symbols in D. From all these sequences we consider only those that are of interest to us (i.e. correct Dutch sentences or existing proteins). Note that in this way the components of the vectors $\mathbf{x}$ and $\mathbf{x}'$ are symbols from the dictionary D. Such contexts have a number of advantages that are in direct contrast with the disadvantages of the orthonormal vectors:

1. As we are working with sequences of symbols it is possible to define a degree of similarity between the different symbols. In this context we can distinguish 2 types of degrees of similarity. In the first type the form of the symbols themselves determines the degree of similarity e.g. we could say that *book* is more similar to *brook* then it is similar to *blood* because they have more characters in common. An example of this is the classification of dialects making use of the Levenshtein distance [15,16]. In the second type it is more a degree of similarity where we say that one symbol is more similar to another based on some form of domain knowledge and not based on the form (i.e. the number of characters they have or do not have in common) of the symbols. In PSSP for example we say that some pairs of amino acids are more similar to each other then other pairs based on evolutionary information represented by a matrix, for an example see Section (4.3).
2. Many similarity measures are defined on the contexts themselves, incorporating these measures into the kernel function often has a positive influence on the classification results. As an example of this consider the distance functions in [17] defined on contexts and used for memory-based learning in natural language problems. Moreover in [3,18] we used these distance functions as a basis to design kernel functions to do SVM learning in natural language problems.
3. A context vector is easy to interpret as it is directly observable what context it represents.

However, working with symbolic data also involves a number of issues. First of all, the data has to be mapped from a discrete input space $X \subseteq \mathbb{S}$ to a real Hilbert space F. Of course by making use of the kernel trick, this mapping ϕ can stay implicit. Nevertheless, there still remains the issue of finding a valid kernel formulation between contexts that captures as much similarity or dissimilarity information as possible.

Next, notice that working with contexts incurs some computational problems because the complexity of comparing 2 strings is proportional to the length of the longest string in the comparison. Remember that one of the most important requirements for a kernel is for it to be computationally efficient as it has to be calculated for all pairs of vectors in the trainingset. As the kernel functions we present here rely on the comparison of symbols and that we only use a degree of similarity of the second type it is possible to represent the symbols not by strings and characters but by their index k_i in the dictionary D. In this way we only have

to compare integers and not strings. For PSSP this won't make a big difference as all strings are single characters of length 1, for LINER however the gain will be considerable. Additionally, some of the kernels that will be introduced in the next section are calculated through a matrix containing entries for every pair of symbols in the set D, by representing the symbols by their index in D we can access such a matrix in $O(1)$.

In the next section we introduce a natural framework for designing kernel functions for contexts based on metrics defined on the contexts themselves.

4 Appropriate Kernel Functions for Contexts

In this section we will always start from a (dis)similarity measure defined on the input examples and derive simple, linear kernels for that measure. In analogy with the real case these linear kernels take the form of simple inner products between the input vectors. Subsequently these inner products will be used as a basis to define more complex kernels and distances in the feature space F.

Section (4.1) starts with the description of a simple inner product based on a simple distance function defined on contexts. For this inner product we will show that it is equivalent with the standard inner product between the corresponding orthonormal vectors, but in the same time we will show that the context approach making use of (dis)similarity measures defined on the contexts themselves is much more intuitive. For this purpose we will consider two extensions of the simple inner product by incorporating domain knowledge in the learning process. Considering increasingly complex (dis)similarity measures it will become clear that reasoning about these measures at the level of the contexts and subsequently deriving kernel functions is much more intuitive than working with orthonormal vectors.

Section (4.2) describes an extension based on a weighted version of the simple distance function from Section (4.1) and Section (4.3) shows how the same method can be applied for problems where a metric is not directly at one's disposal, more specifically we will look at problems where similarity information is available under the form of a matrix. Finally Section (4.4) gives an overview of how to use the different simple inner products to define distances in the feature space F and more complex kernels suited for SVM learning.

Notice that from now on we will be working with vectors $\mathbf{x}, \mathbf{x'} \in X^l$ with $X^l \subseteq \mathbb{S}^l$ and vectors $\tilde{\mathbf{x}}, \tilde{\mathbf{x}'} \in \tilde{X}^{n*l} \subseteq \mathbb{R}^{n*l}$ with l the length of the contexts and n the cardinality of D as before. The components are denoted by $x_i, x'_i \in D$ and $\tilde{x}_i, \tilde{x}'_i \in \mathbb{R}$ respectively. In practice however the contexts will be represented by the index of the symbols in D. Finally, $\langle \ . \ . \ \rangle$ will be used to denote the standard inner product in real Euclidean space and $\langle \ . \mid . \ \rangle$ will be used to denote the pseudo-inner product between contexts. We call it *pseudo* because, although it will satisfy all necessary conditions of an inner product, it is strictly mathematically spoken not an inner product as it is defined on strings.

4.1 A Simple Overlap Kernel

The most basic metric for contexts is the *Simple Overlap Metric* (SOM) [19,3]:

Definition 4. *Let $X^l \subseteq \mathbb{S}^l$ be a discrete space with l-dimensional contexts $\mathbf{x}$ and $\mathbf{x}'$ with components x_i and $x'_i \in D$ with $|D| = n$ as before. Then the distance $d_{SOM} : X^l \times X^l \to \mathbb{R}^+$ is defined as*

$$d_{SOM} : X^l \times X^l \to \mathbb{R}^+ : d_{SOM}(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{l} \delta(x_i, x'_i) \tag{12}$$

with $\delta(x_i, x'_i) = 1 - \delta_{x_i}(x'_i)$ and $\delta : D \times D \to \{1, 0\}$:

$$\delta_{x_i}(x'_i) = 1 \; if \; x_i = x'_i \neq -, \; else \; 0 \tag{13}$$

with l the context size and ' $-$ ' referring to the empty token.

In previous work we constructed a kernel function by defining a mapping $\phi_{SOM} : X^l \to F$ based on the distance function from Equation (12) and working out the inner product in F. Moreover by making use of Equation (7) we showed that the distance $\|\phi_{SOM}(\mathbf{x}) - \phi_{SOM}(\mathbf{x}')\|$ in the feature space F (calculated through the newly derived kernel function) is very closely related to d_{SOM} [3]. Next we will give a more general form of this kernel function and a more formal proof of its positive definiteness.

Proposition 2. *Let $X^l \subseteq \mathbb{S}^l$ be a discrete space with l-dimensional contexts $\mathbf{x}$ and $\mathbf{x}'$, with components x_i and $x'_i \in D$ with $|D| = n$ as before. Let $\tilde{X}^{n*l} \subseteq \mathbb{R}^{n*l}$ be a binary space with $n * l$-dimensional vectors $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{x}'}$ the corresponding orthonormal vector representation of the vectors $\mathbf{x}$ and $\mathbf{x}'$. Then the function $\langle \, . \mid . \, \rangle_{SOM} : X^l \times X^l \to \mathbb{R}$ defined as $\langle \mathbf{x} | \mathbf{x}' \rangle_{SOM} = \sum_{i=1}^{l} \delta_{x_i}(x'_i)$ with δ as defined in (13) is positive definite (PD) and $\langle \mathbf{x} | \mathbf{x}' \rangle_{SOM} = \langle \tilde{\mathbf{x}}, \tilde{\mathbf{x}}' \rangle$.*

Proof. We will prove that $\langle \mathbf{x} | \mathbf{x}' \rangle_{SOM} = \langle \tilde{\mathbf{x}}, \tilde{\mathbf{x}}' \rangle$, the positive definiteness of the function $\langle \, . \mid . \, \rangle_{SOM}$ will follow automatically. We start by noting that the vectors $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{x}'}$ are composed out of l orthonormal vectors. We will denote this as follows: $\tilde{\mathbf{x}} = ([\tilde{\mathbf{x}}]_1, \ldots [\tilde{\mathbf{x}}]_l)$ with $[\tilde{\mathbf{x}}]_i$ the orthonormal vector corresponding with symbol x_i from the vector $\mathbf{x}$. For $l = 1$ and $\forall \mathbf{x}, \mathbf{x}' \in X^1$ we have the following:

$$\langle \mathbf{x} | \mathbf{x}' \rangle_{SOM} = \delta_{x_1}(x'_1) = \begin{cases} 1 & \text{if} \quad x_1 = x'_1 \\ 0 & \text{if} \quad x_1 \neq x'_1 \end{cases}$$

For the discrete case this follows directly from the definition of the kernel function and the distance function it is based on, see Equation (13). For the orthonormal case it is sufficient to note that for the inner products between all orthonormal vectors $[\tilde{\mathbf{x}}], [\tilde{\mathbf{x}'}]$ it holds that:

$$\left\langle [\tilde{\mathbf{x}}], [\tilde{\mathbf{x}'}] \right\rangle = \begin{cases} 1 & \text{if} \quad [\tilde{\mathbf{x}}] = [\tilde{\mathbf{x}'}] \\ 0 & \text{if} \quad [\tilde{\mathbf{x}}] \neq [\tilde{\mathbf{x}'}] \end{cases} \tag{14}$$

Next, because $l = 1$ we need only one orthonormal vector to construct complete instances, i.e. $\tilde{\mathbf{x}} = [\tilde{\mathbf{x}}]_1$ and $\tilde{\mathbf{x}'} = [\tilde{\mathbf{x}'}]_1$ and thus:

$$\langle \tilde{\mathbf{x}}, \tilde{\mathbf{x}}' \rangle = \sum_{i=1}^{l=1} \left\langle [\tilde{\mathbf{x}}]_i, [\tilde{\mathbf{x}'}]_i \right\rangle \tag{15}$$

$$= \left\langle [\tilde{\mathbf{x}}]_1, [\tilde{\mathbf{x}'}]_1 \right\rangle$$
$$= \delta_{x_1}(x'_1)$$

Where the last step is justified by Equation (14) and by the assumption that $[\tilde{\mathbf{x}}]_i$ is the orthonormal vector corresponding to the token x_i.
Now assume that the proposition holds for $l = m$. Next, we will prove that the proposition holds for $l = m+1$ and by induction we will be able to conclude that it holds for all l. We start by showing that the calculation of the kernel values for $l = m + 1$ can be decomposed in terms of $l = m$:

$$\langle \mathbf{x}|\mathbf{x}' \rangle_{SOM} = \sum_{i=1}^{l=m} \delta_{x_i}(x'_i) + \begin{cases} 1 & \text{if} \quad x_{m+1} = x'_{m+1} \\ 0 & \text{if} \quad x_{m+1} \neq x'_{m+1} \end{cases} \tag{16}$$

Now it can be readily seen that the the proposition holds for $l = m + 1$ because we now by assumption that for the left part of the RHS of Equation (16) it holds that $\sum_{i=1}^{l=m} \delta_{x_i}(x'_i) = \sum_{i=1}^{l=m} \left\langle [\tilde{\mathbf{x}}]_i, [\tilde{\mathbf{x}'}]_i \right\rangle$ and for the right part of the RHS making use of Equations (14) and (15) $x_{m+1} = x'_{m+1}$ and $x_{m+1} \neq x'_{m+1}$ implies $\left\langle [\tilde{\mathbf{x}}]_{m+1}, [\tilde{\mathbf{x}'}]_{m+1} \right\rangle = 1$ and 0 respectively. □

Although we have shown that $\langle \mathbf{x}|\mathbf{x}' \rangle_{SOM} = \langle \tilde{\mathbf{x}}, \tilde{\mathbf{x}}' \rangle$ we will argue that it is better to work with contexts in stead of orthonormal vectors because this gives a much more intuitive basis to incorporate extra domain knowledge into the learning algorithm under the form of special purpose distance functions. We start with a simple extension of $\langle \ . \mid . \ \rangle_{SOM}$ by introducing weights into d_{SOM}.

4.2 A Simple Weighted Overlap Kernel

Consider the SOM from Equation (12) and suppose we add weights w_i as follows:

$$d_{WOM} : X^l \times X^l \to \mathbb{R}^+ : d_{WOM}(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{l} w_i \delta(x_i, x'_i) \tag{17}$$

with δ as before and $\forall i : w_i \geq 0$.

Next we can define an inner product or simple kernel based on the distance function from Equation (17) in the same way as before.

Proposition 3. *Let $X^l \subseteq \mathbb{S}^l$ be a discrete space with l-dimensional contexts $\mathbf{x}$ and $\mathbf{x}'$, with components x_i and $x'_i \in D$ with $|D| = n$ as before. Then the function $\langle \ . \mid . \ \rangle_{SOM} : X^l \times X^l \to \mathbb{R}$ defined as $\langle \mathbf{x}|\mathbf{x}' \rangle_{WOM} = \sum_{i=1}^{l} w_i \delta_{x_i}(x'_i)$ with δ as defined in (13) is a positive definite (PD) kernel function.*

Proof. The proof is very straightforward, i.e. in a similar way as for Proposition 2 it can be proved that $\langle \ . \mid . \ \rangle_{WOM}$ corresponds to the inner product between orthonormal vectors that are not binary but where the non-zero component of the orthonormal vector corresponding to token x_i takes the value $\sqrt{w_i}$. The details are left to the reader. □

4.3 A Simple Similarity Matrix Based Kernel

This section discusses the case where there is a degree of similarity between the different symbols in D. More precisely, the case where such similarity information is organized as a matrix will be discussed. We start by giving the definition of a similarity matrix for a dictionary D.

Definition 5. *Given a dictionary D a similarity matrix $\mathbf{S}$ is the $n \times n$ matrix with two entries for every pair of symbols in D, i.e. $s_{jk}, s_{kj} \in \mathbb{R}$ for $j, k \in \{1, 2, \ldots, n\}$ the indices of the symbols in D. Large values for the entries indicate a high degree of similarity between the corresponding symbols in D and small values indicate a low degree of similarity between the corresponding symbols in D.*

Many examples of similarity matrices can be found in the field of biology, i.e. for nucleotide scoring in DNA sequences, protein scoring for amino acid sequences or PSSP. Next, above similarity measure, that is defined on contexts and symbols will be used to define a simple pseudo-inner product. It will be shown how the pseudo-inner product is calculated through the entries of a similarity matrix $\mathbf{S}$ and what properties $\mathbf{S}$ should satisfy in order for the inner product to make sense.

Definition 6. *Let $X^l \subseteq \mathbb{S}^l$ be a discrete space with l-dimensional contexts $\mathbf{x}$ and $\mathbf{x}'$ with components x_i and $x'_i \in D$ with $|D| = n$ as before and let j_i and $k_i \in \{1, 2, \ldots, n\}$ be the indices of the components x_i and $x'_i \in D$ such that $\mathbf{S}(j_i, k_i) = s_{jk}$, with $\mathbf{S}$ a $n \times n$ similarity matrix as defined in Definition 5 with the additional properties so that for all j, k:*

$$\begin{aligned} s_{jk} &= s_{kj} \\ s_{jj} &\geq 0 \\ |s_{jk}|^2 &\leq s_{jj} s_{kk} \end{aligned}$$

Then the similarity matrix based inner product : $\langle \, . \mid . \, \rangle_{MAT} : X^l \times X^l \to \mathbb{R}$ is defined in the following way:

$$\langle \mathbf{x} | \mathbf{x}' \rangle_{MAT} = \sum_{i=1}^{l} \mathbf{S}(j_i, k_i) \tag{18}$$

Nevertheless, even if a matrix $\mathbf{S}$ satisfies the above properties the pseudo-inner product is still not sure to be PD. For the pseudo-inner product to be PD and hence usable as a kernel function in a SVM the similarity matrix $\mathbf{S}$ should also be PSD and positive valued. This is expressed in the following proposition.

Proposition 4. *Let $X^l \subseteq \mathbb{S}^l$ be a discrete space with l-dimensional contexts $\mathbf{x}$ and $\mathbf{x}'$ with components x_i and $x'_i \in D$ with $|D| = n$ as before, with $j_i, k_i \in \{1, 2, \ldots, n\}$ and with $\mathbf{S}$ as in Definition 6. Then the function $\langle \, . \mid . \, \rangle_{MAT} : X^l \times X^l \to \mathbb{R}$ defined as in Equation 18 is PD if and only if $\mathbf{S}$ is PSD and $s_{jk} \geq 0$ for all j, k.*

Proof. We start with the binary, 1-dimensional case. Assume that D contains only 2 symbols, i.e. $n = 2$, also assume that $l = 1$. We have to prove that, given the input vectors $\mathbf{x}_i$ and $\mathbf{x}_j \in X^1$, the kernel matrix $\mathbf{K}_{ij} = \langle \mathbf{x}_i, \mathbf{x}_j \rangle_{MAT}$ is PSD.

For $l = 1$ we have a kernel matrix that takes the following form:

$$\mathbf{K_1} = \begin{pmatrix} s_{1,1} & s_{1,2} \\ s_{2,1} & s_{2,2} \end{pmatrix}$$

$\mathbf{K_1} = \mathbf{S}$ is a PSD matrix by requirement of the proposition and thus the proposition is true for $l = 1$.

Now assume that the proposition holds for $l = m$. We will prove now that the proposition holds for $l = m+1$ and by induction we will be able to conclude that it holds for all l. We start by showing that $\mathbf{K_{m+1}}$ can be decomposed in terms of $\mathbf{K_m}$ as follows:

$$\mathbf{K_{m+1}} = \begin{pmatrix} \mathbf{K_m} + (s_{1,1})_m & \mathbf{K_m} + (s_{1,2})_m \\ \mathbf{K_m} + (s_{2,1})_m & \mathbf{K_m} + (s_{2,2})_m \end{pmatrix} \tag{19}$$

$$= \begin{pmatrix} \mathbf{K_m} & \mathbf{K_m} \\ \mathbf{K_m} & \mathbf{K_m} \end{pmatrix} + \begin{pmatrix} (s_{1,1})_m & (s_{1,2})_m \\ (s_{2,1})_m & (s_{2,2})_m \end{pmatrix} \tag{20}$$

With the $m \times m$ matrices $(s_{jk})_m$ containing the element s_{jk} at every position and $\mathbf{K_{m+1}}$ a $2^{m+1} \times 2^{m+1}$ symmetric matrix. Given that $\mathbf{K_m}$ is PSD the first matrix in (20) is PSD as well. Next, given that $\forall j, k : s_{jk} \geq 0$ the matrices $(s_{jk})_m$ are all PSD and hence the second matrix in (20) is PSD as well.

Next we will consider the more general case for a dictionary with a cardinality c, i.e. D does not contain only 2 elements but c elements with c finite, i.e. $n = c$. Start by considering $\mathbf{K_1}$ which is a $c \times c$ matrix with $\mathbf{K_1} = \mathbf{S}$ like for the binary case and thus the proposition is true for $l = 1$. Next, in the same way as before we will show how the $c^{(m+1)} \times c^{(m+1)}$ matrix $\mathbf{K_{(m+1)}}$ can be decomposed in terms of $\mathbf{K_m}$:

$$\mathbf{K_{m+1}} = \begin{pmatrix} \mathbf{K_m} + (s_{1,1})_m & \mathbf{K_m} + (s_{1,2})_m & \dots & \mathbf{K_m} + (s_{1,c})_m \\ \mathbf{K_m} + (s_{2,1})_m & \mathbf{K_m} + (s_{2,2})_m & \dots & \mathbf{K_m} + (s_{2,c})_m \\ \vdots & & \ddots & \vdots \\ \mathbf{K_m} + (s_{c,1})_m & \dots & & \mathbf{K_m} + (s_{c,c})_m \end{pmatrix} \tag{21}$$

$$= \begin{pmatrix} \mathbf{K_m} & \mathbf{K_m} & \dots & \mathbf{K_m} \\ \mathbf{K_m} & \mathbf{K_m} & \dots & \mathbf{K_m} \\ \vdots & \dots & \ddots & \vdots \\ \mathbf{K_m} & \dots & & \mathbf{K_m} \end{pmatrix} + \begin{pmatrix} (s_{1,1})_m & (s_{1,2})_m & \dots & (s_{1,c})_m \\ (s_{2,1})_m & (s_{2,2})_m & \dots & (s_{2,c})_m \\ \vdots & \dots & \ddots & \vdots \\ (s_{c,1})_m & \dots & & (s_{c,c})_m \end{pmatrix}$$

Next in a similar way as before we can show that the first and second matrix in (22) are PSD and as a result the matrix $\mathbf{K_{m+1}}$ is PSD as well. Hence by induction, the proposition is true. □

4.4 More Complex Kernels in Tree Flavors

In the following section tree kinds of kernels, based on the pseudo-inner products of the previous sections, will be defined. The first type that will be discussed is based on the polynomial kernel [5]. The second and the third type are formed by defining distances making use of the pseudo-inner products of the previous sections and Equations (8) and (11).

Start by considering the standard polynomial kernel in real space and applied to orthonormal vectors $\tilde{\mathbf{x}}, \tilde{\mathbf{x}}' \in \tilde{X}^{n*l} \subseteq \mathbb{R}^{n*l}$:

$$K_{poly}(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}') = (\langle \tilde{\mathbf{x}}, \tilde{\mathbf{x}}' \rangle + c)^d \tag{22}$$

Now, we can make tree kernels for contexts $\mathbf{x}, \mathbf{x}' \in X^l \subseteq \mathbb{S}^l$ by substituting the standard inner product in Equation (22) by one of the pseudo-inner products from Propositions 2, 3 and 4 and by normalizing them as:

$$\tilde{K}(\mathbf{x}, \mathbf{x}') = \frac{K(\mathbf{x}, \mathbf{x}')}{\sqrt{K(\mathbf{x}, \mathbf{x})}\sqrt{K(\mathbf{x}', \mathbf{x}')}}$$

The resulting kernels are called *Simple Overlap Kernel* (SOK), *Weighted Overlap Kernel* (WOK) and *Pseudo-Inner product Kernel* (PIK) respectively. Note that for our purposes we always choose $c = 0$ and $d = 2$. Also note that the normalization step has a similar effect as the normalization of real inputs, but that we cannot normalize the inputs here as they are discrete.

The second type of kernels are formed by defining distances by making use of the pseudo-inner products from the previous sections and the fact that we can define distances for these inner products in a similar way as in Equation (7), i.e.

$$\|\mathbf{x} - \mathbf{x}'\| = \sqrt{\langle \mathbf{x}|\mathbf{x} \rangle - 2 \langle \mathbf{x}|\mathbf{x}' \rangle + \langle \mathbf{x}'|\mathbf{x}' \rangle} \tag{23}$$

with $\langle \, . \mid . \, \rangle$ one of the inner products from Propositions 2, 3 and 4. Next, by substituting Equation (23) in Equation (8) the analogy is perfect. The resulting kernel functions are called *Overlap Radial Basis Kernel* (ORBK), *Weighted Radial Basis Kernel* (WRBK) and *Substitution Radial Basis Kernel* (SRBK) respectively.

Finally we also consider a type of conditionally positive definite kernels by making use of the same distances as discussed above and substituting them into the negative distance kernel from Equation (11). The resulting kernels for the pseudo-inner products from Propositions 2, 3 and 4 are called the *Negative Overlap Distance Kernel* (NODK), *Weighted Negative Distance Kernel* (WNDK) and *Negative Substitution Distance Kernel* (NSDK) respectively.

5 Experiments

In this section the kernel functions described in Section (4) will be tested on the problem of PSSP as it is described in Example 2.

Notice that it is not our goal to do better than the state of the art methods as we will do classification based only on the primary structure sequences and a number of multiple sequence alignments. Our goal is to:

1. Compare the orthonormal vector approach with the kernel functions SOK, ORBK and NODK making use of contexts.
2. Compare the kernel functions SOK, ORBK and NODK with the kernel functions PIK, SRBK and NSDK making use of the BLOSUM62 matrix (see Section (5.1)).

Also note that we will not do experiments with the WOK, WRBK and WNDK as we don't have the space to do so, in previous work however [3,18] we already did some experiments with similar kernels making use of information gain weights. At the moment we are doing some more optimized experiments for LINER making use of the weighted kernels, information gain ratio weights using grid search to optimize all parameters of the learning process.

Finally note that we will not consider related methods like the mismatch, spectrum and profile kernels [20,21] and the syllable and other string kernel extensions of [22]. Although these methods use similar concepts they are designed to do classification of the sequences themselves while our method aims to do classification of the different symbols in the sequences. In that sense the named methods calculate kernel values for the sequences while we calculate kernel values for all symbols in the sequences.

5.1 Protein Secondary Structure Prediction

Overview. For a description of the problem of protein secondary structure prediction (PSSP) we refer to Example 2. The current state of the art PSSP methods achieve prediction accuracies around 77%. These methods do not only make use of the similarity between the primary structure sequences but they also make use of evolutionary information, profiles, chemical properties, etc. Moreover the best performing methods combine different machine learning methods to come to their good results. For a comprehensive overview of the state of the art PSSP methods we refer to [23].

It is also important to notice that an important aspect in protein secondary structure prediction is the use of multiple alignments. A multiple alignment is a set of amino acid sequences in a rectangular arrangement, where each row consists of one sequence padded by gaps, such that the columns highlight similarity/conservation between positions. An optimal multiple alignment is one that has optimal score, i.e. the highest degree of similarity, or the lowest cost (calculated by the Levenshtein Distance for example). Multiple alignments contain more information than the sequences alone, the additional information comes from the fact that the pattern of substitutions in these alignments reflects the family's protein fold [8].

Similarity Between Amino Acid Sequences. The simplest way to measure the similarity between two amino acid sequences is to simply count the number of (mis)matching amino acids in both sequences making use of the SOM from Equation (12). However, for amino acids this turns out to be a rather naive approach. Some pairs of amino acids are more *similar* than other pairs. With

similar we mean here that, for a given period of evolution, some amino acids are more likely to mutate than other amino acids. The fact that some amino acids can mutate in other amino acids with high probability and without changing the function of the protein can be considered as a form of similarity. Therefore similarity among amino acid sequences can be modeled with a substitution matrix. The entries of a 20×20 substitution matrix describe, for a certain period of evolution, the probabilities of amino acid mutations for all possible pairs of amino acids. Two well-known families of substitution matrices are the family of *Point Accepted Mutation* (PAM) [24] matrices and the family of *BLOck SUbstitution Matrix* (BLOSUM) [25] matrices. These matrices are created based on sets of well-known proteins. For a set of well-known proteins : i) align the sequences, ii) count the mutations at each position, iii) for each substitution set the score to the *log-odd ratio* :

$$\log_2 \left(\frac{\text{observed mutation rate}}{\text{mutation rate expected by chance}} \right)$$

We will not go in further detail about the construction of such matrices as this would lead us to far. As an example we will consider one particular BLOSUM matrix, called the BLOSUM62 matrix.

5.2 Data and Software

For the experiments we used the **CB513** data set [26]. This dataset is compiled by Cuff and Barton and consists of 513 proteins. The experiments are conducted with LIBSVM [9] a Java/C++ library for SVM learning. An amino acid that is to be predicted is represented by a context vector of 11 amino acids, 5 acids before and 5 acids after the central amino acid that we wish to classify.

5.3 Results

We start by giving the 6-fold crossvalidation results of the SOK, ORBK and NODK. The main goal here is to determine good values for the SVM cost parameter C and the kernel parameters γ and β. Table **??** gives these results. Notice that k refers to the context size, which in our case is 11. Notice that we also included results for the standard (normalized) polynomial and radial basis kernels applied to the orthonormal representation of the contexts. Completely in the line of the expectations they give identical results as the SOK and ORBK are calculated through the pseudo-inner product from Proposition 2 for which we proved that it is equal to the standard inner product applied to the orthonormal vector representation of the contexts. Next, in Table 2 we show results of the precision and recall accuracies for every class of the SOK, ORBK and NODK. We train probability estimates on a data set that includes 3 alignments for every protein, resulting in a data set with 244021 instances. For the multi-class classification a *one-against-one* method is used and to obtain the probability estimates for every output LIBSVM internally performs crossvalidation and estimates pairwise probabilities based on the training data and their decision values, for more details see

Table 1

Kernel function	Overall Accuracy
$K_{SOK}(C = 1, d = 2, c = 0)$	62.61%
$K_{poly}(C = 1, d = 2, c = 0)$	62.61%
$K_{ORBK}(C = 1, \gamma = 1/k)$	63.98%
$K_{radial}(C = 1, \gamma = 1/k)$	63.98%
$K_{NODK}(C = 0.3, \beta = 1)$	63.06%

[9]. In the test set we included 35 alignments per protein, resulting in a test set with 321532 instances. At prediction time we determined the class label for every acid by performing majority weighting on the estimated probabilities for all the alignments [27]. For the kernel and SVM parameters we used the values that have been determined by the 6-fold crossvalidation. Note that the significance intervals for the overall accuracies have been obtained with bootstrap resampling making use of 500 samples [28]. Accuracies outside of these intervals are assumed to be significantly different from the related accuracy ($p < 0.05$). Also note that the recall for *β-sheets* is very low, this is because *β sheets* are governed by long range interactions that can not be captured by our kernel functions.

Next we will make use of a similarity matrix in the way described in Section (4.3) to do PSSP. The matrix we will be using is the BLOSUM62 matrix (see Section (5.1) and [25]). First of all it is important to notice that the BLOSUM62 matrix does satisfy the conditions of Definition 6 but it does not satisfy the conditions of Proposition 4, i.e. it is not PSD and not positive valued. Therefore we perform a sort of normalization:

$$\tilde{B}_{62} = \frac{B_{62} - min(B_{62})}{max(B_{62}) - min(B_{62})}$$

with the functions *min* and *max* giving the smallest and largest elements of the matrix. After performing this operation, the matrix contains all elements between 0 and 1. By doing so, the matrix becomes almost PSD, i.e. it has one slightly negative eigenvalue. However, this does not cause any problems as LIBSVM can cope with this.

For training and testing we used the same data sets as before and we also trained probability estimates and used majority weighting on the estimated probabilities of the different alignments as before, the results are given in Table 3.

The results show that every kernel that makes use of the BLOSUM62 matrix outperforms its counterpart that doesn't with a significant difference. This shows that the use of special purpose (dis)similarity measures defined on the input examples can have a positive influence on the classification results.

In the light of these results it is our belief that the use of these kernel functions in a complete PSSP system can lead to a system with really competitive results. Moreover, experts might be able to apply post-processing heuristics or to refine the metric defined on the input examples, for example by making use of other substitution matrices like the BLOSUM40, BLOSUM100 or even a totally different family of matrices. Additionally it could also be tried to use a

Table 2

Kernel	Class	Precision	Recall	Overall Accuracy
K_{SOK}	α	76.78%	74.85%	71.89% ± 0.55
	β	73.73%	50.08%	
	coil	67.87%	80.38%	
K_{ORBK}	α	77.06%	76.38%	72.96% ± 0.54
	β	72.58%	54.72%	
	coil	69.97%	79.22%	
K_{NODK}	α	76.69%	76.54%	73.07% ± 0.57
	β	72.17%	55.96%	
	coil	70.59%	78.74%	

Table 3

Kernel	Class	Precision	Recall	Overall Accuracy
K_{PIK}	α	77.70%	75.81%	73.05% ± 0.56
	β	75.52%	52.72%	
	coil	68.94%	80.95%	
K_{SRBK}	α	78.17%	76.07%	73.49% ± 0.52
	β	75.69%	53.90%	
	coil	69.43%	81.18%	
K_{NSDK}	α	77.86%	76.78%	73.67% ± 0.54
	β	73.44%	57.53%	
	coil	70.55%	79.14%	

weighted kernel function like one of the WOK, WRBF or WNDK making use of information gain ratio weights or a combination of such weighting with substitution matrices. Finally, optimizing the left and right context might also further improve the accuracy a little.

6 Conclusion

This article gave a concise overview of the theory of support vector machines with a strong focus on the relationship between kernel functions, similarity measures, metrics and distances. It was shown that the use of (dis)similarity measures, defined on the input examples, in the kernel function of a SVM is an important design decision. In this context we argued that for discrete symbolic input spaces the use of the orthonormal vectors is not the best choice because the (dis)similarity measures one would like to use in the kernel functions are defined on strings and contexts and not on real, orthonormal vectors. In particular we considered the case where input examples are formed by sliding a window over a sequence of strings or characters. For this type of problems a number of kernel functions, working directly on contexts, and based on metrics defined on those contexts, have been described. We started out with a very simple linear kernel that serves as a basis for designing other kernels in a similar way. Two such other

kernels, one based on a weighted distance function and one based on similarity matrices, were described. In general, a more intuitive framework for designing kernel functions, based on (dis)similarity measures on contexts, was proposed. Moreover, for every kernel function it was shown that it satisfies the necessary conditions of a kernel. Finally in the experiments the proposed kernel functions have been applied to the problem of protein secondary structure prediction and compared with each other. The results showed that the proposed method can serve as a valid and competitive alternative for SVM learning with symbolic data and input examples formed by a sliding window.

References

1. Schölkopf, B.: The kernel trick for distances. Technical report, Microsoft Research (2000)
2. Joachims, T.: Learning to Classify Text Using Support Vector Machines. Kluwer Academic Publishers (2002)
3. Vanschoenwinkel, B., Manderick, B.: A weighted polynomial information gain kernel for resolving pp attachment ambiguities with support vector machines. In: The Proceedings of the Eighteenth International Joint Conferences on Artificial Intelligence (IJCAI-03). (2003) 133–138
4. Vapnik, V.: The Nature of Statistical Learning Theory. Springer-Verlag, New York (1998)
5. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines and other Kernel-based Learning Methods. Cambridge University Press (2000)
6. Haussler, D.: Convolution kernels on discrete structures (1999)
7. Hein, M., Bousquet, O.: Maximal margin classification for metric spaces. In: Proceedings of Learning Theory and Kernel Machines: 16th Annual Conference on Learning Theory and 7th Kernel Workshop, Colt/Kernel 2003 Washington, Dc. Usa, August 24-27, Springer-Verlag (2003) 72–86
8. Rost, B., Sander, C.: Improved prediction of protein secondary structure by use of sequence profiles and neural networks. In: Proceedings of the National Academy of Science USA. Volume 90. (1993) 7558–7562
9. Chih-Chung, C., Chi-Jen, L.: LIBSVM : A Library for Support Vector Machines. (2004)
10. McNamee, P., Mayfield, J.: Entity extraction without language-specific resources. In: Proceedings of CoNLL-2002, Taipei, Taiwan (2002) 183–186
11. Kudoh, T., Matsumoto, Y.: Use of support vector learning for chunk identification. In: Proceedings of CoNLL-2000 and LLL-2000, (Lisbon, Portugal)
12. Takeuchi, K., Collier, N.: Use of support vector machines in extended named entity. In: Proceedings of CoNLL-2002, Taipei, Taiwan (2002) 119–125
13. Hua, S., Sun, Z.: A novel method of protein secondary structure prediction with high segment overlap measure: Support vector machine approach. Journal Of Molecular Biology **308 (2)** (2001) 397–407
14. Degroeve, S.: Design and Evaluation of a Linear Classification Strategy for Gene Structural Element Recognition. PhD thesis, Universiteit Gent, Faculty of Sciences, Gent, Belgium (2004)
15. Levenshtein, V.: Binary codes capable of correcting deletions, insertions and reversals. Soviet Physics - Doklady 10, **10** (1966) 707–710

16. Nerbonne, J., Heeringa, W., Kleiweg, P.: Comparison and classification of dialects. In: Preceedings of EACL 99. (1999)
17. Daelemans, W., Zavrel, J., van der Sloot, K., van den Bosch, A.: Timbl: Tilburg memory-based learner, version 4.3. Technical report, Tilburg University and University of Antwerp (2002)
18. Vanschoenwinkel, B.: A discrete kernel approach to support vector machine learning in language independent named entity recognition. In: Proceedings of BENELEARN-04 (Annual Machine Learning Conference of Belgium and The Netherlands). (2004) 154–161
19. Zavrel, J., Daelemans, W., Veenstra, J.: Resolving pp attachment ambiguities with memory-based learning. In: Proceedings CoNNL, Madrid, Computational Linguistics, Tilburg University (1997) 136 – 144
20. Leslie, C., Eskin, E., Noble, W.S.: The spectrum kernel : A string kernel for svm protein classification. In: Proceedings of the Pacific Symposium on Biocomputing. (2002) 564–575
21. Kuang, R., Ie, E., Wang, K., Siddiqi, M., Freund, Y., Leslie, C.: Profile-based string kernels for remote homology detection and motif extraction. In: Computational Systems Biology Conference. (2004)
22. Saunders, C., Tschach, H., Shawe-Taylor, J.: Syllables and other string kernel extensions. In: Proceedings of the 19th International Conference on Machine Learning, Morgan Kaufmann (2002) 530–537
23. Rost, B.: Rising accuracy of protein secondary structure prediction. New York: Dekker (2003)
24. Dayhoff, M., Schwartz, R.: Matrices for detecting distant relationships. Atlas of Protein Sequences (1979) 353–358
25. Henikoff, S., Henikoff, J.: Amino acid substitution matrices from protein blocks. In: Proceedings of the National Academy of Science USA. Volume 89(2). (1992) 10915–10919
26. Cuff, J., Barton, G.: Evaluation and improvement of multiple sequence methods for protein secondary structure prediction. Proteins: Struct., Funct., Genet. **35** (1999) 508–519
27. Sren Kamaric Riis, A.K.: Improving prediction of protein secondary structure using structured neural networks and multiple sequence alignments. Journal Of Computational Biology **3** (1996) 163–183
28. Noreen, E.W.: Computer-Intensive Methods for Testing Hypotheses. John Wiley & Sons (1989)

Variational Bayes Estimation of Mixing Coefficients

Bo Wang and D. M. Titterington

Department of Statistics, University of Glasgow, Glasgow G12 8QQ, UK

Abstract. We investigate theoretically some properties of variational Bayes approximations based on estimating the mixing coefficients of known densities. We show that, with probability 1 as the sample size n grows large, the iterative algorithm for the variational Bayes approximation converges locally to the maximum likelihood estimator at the rate of $O(1/n)$. Moreover, the variational posterior distribution for the parameters is shown to be asymptotically normal with the same mean but a different covariance matrix compared with those for the maximum likelihood estimator. Furthermore we prove that the covariance matrix from the variational Bayes approximation is 'too small' compared with that for the MLE, so that resulting interval estimates for the parameters will be unrealistically narrow.

1 Introduction

Variational Bayes approximations, introduced by Attias [1,2], have been successfully applied to complex models involving incomplete-data where computational difficulties arise in the ideal Bayesian approaches, as for instance with hidden Markov models and mixture models. The approximations are widely recognised to be effective and promising in a series of papers, such as [3,4,5,6,7,8,9,10] and the references therein. In these earlier contributions, the approximations were shown empirically to be convergent and consistent. In Attias [1,2] and Penny and Roberts [10] the authors claimed that the variational Bayes estimator approaches the maximum likelihood estimator in the large sample limit, but no rigorous proof was given. While experimental results are often satisfactory in practice, exact theoretical assessment of the quality of this method is an important issue.

In this paper we study some properties of variational Bayes approximations theoretically for certain mixture models. Based on estimating the mixing coefficients of known densities, we show that, with probability 1 as the sample size n grows large, the iterative algorithm for the variational Bayes approximation converges locally to the maximum likelihood estimator at the rate of $O(1/n)$. Moreover, we prove that the variational posterior distribution for the parameters is asymptotically normal with the same mean but a different covariance matrix compared with those for the maximum likelihood estimator. Further developments show that the covariance matrix from the variational Bayes approximation is 'too small' compared with that for the MLE, so that resulting interval

J. Winkler, N. Lawrence, M. Niranjan (Eds.): Machine Learning Workshop, LNAI 3635, pp. 281–295, 2005.

estimates for the parameters will be too narrow. Numerical examples reinforce the theoretical analysis.

2 The Mixture Model and the Variational Approximation

We consider a model in which we have a mixture of m known densities $p_1, p_2, \ldots, p_m$. The density of an observation is given by

$$p(y_i|\Theta) = \sum_{s=1}^{m} p_s(y_i) p(s_i = s|\Theta) \ , \tag{1}$$

where $y_i \in \mathbb{R}^d$ denotes the ith observed data vector, and s_i indicates the hidden component that generated it. The components are labelled by $s = 1, 2, \ldots, m$, and the component s has mixing coefficient $\theta_s = p(s_i = s|\Theta)$ for any i. We write the parameters collectively as $\Theta = (\theta_1, \theta_2, \ldots, \theta_m)'$, and assign a Dirichlet prior distribution $\mathcal{D}(a_1^{(0)}, a_2^{(0)}, \ldots, a_m^{(0)})$ to Θ.

Suppose that we have (complete) data consisting of a random sample of size n, with $Y = (y_1, y_2, \ldots, y_n)'$ and $S = (s_1, s_2, \ldots, s_n)'$. Then the joint density of Θ, S and Y is

$$p(\Theta, S, Y) \propto \Big\{ \prod_{s=1}^{m} \theta_s^{a_s^{(0)}-1} \Big\} \prod_{i=1}^{n} \{\theta_{s_i} p_{s_i}(y_i)\} \ .$$

Since the exact posterior distribution of Θ requires marginalising $p(\Theta, S|Y)$ over S, which leads to intractable calculations when the sample size is large, alternative approximate distributions need to be developed. In the variational Bayes approach, we use an approximating density $q(S, \Theta)$ for $p(S, \Theta|Y)$, which factorises as $q(S, \Theta) = q^{(S)}(S) q^{(\Theta)}(\Theta)$, and is chosen to maximise

$$\int \sum_{\{S\}} q(S, \Theta) \log \frac{p(\Theta, S, Y)}{q(S, \Theta)} d\Theta \ .$$

It follows that $q^{(S)}(S)$ factorises as $q^{(S)}(S) = \prod_{i=1}^{n} q_i^{(S)}(s_i)$ and the variational posterior can be obtained by the following iterative procedure. In turn, we perform the following two stages.

(i) Optimise $q^{(\Theta)}(\Theta)$ for fixed $\{q_i^{(S)}(s_i), i = 1, \ldots, n\}$. This step results in

$$q^{(\Theta)}(\Theta) \sim \mathcal{D}(\{a_s^{(0)} + \sum_{i=1}^{n} r_{is}\}_{s=1}^{m}) \ , \tag{2}$$

where $r_{is} = q_i^{(S)}(s_i = s)$.

(ii) Optimise $\{q_i^{(S)}(s_i), s_i = 1, \ldots, m, i = 1, \ldots, n\}$ for fixed $q^{(\Theta)}(\Theta)$. This results in

$$r_{is} = q_i^{(S)}(s_i = s) = \frac{p_s(y_i)\phi_s}{\sum_{s=1}^{m} p_s(y_i)\phi_s} \ , \quad s = 1, \ldots, m, \tag{3}$$

where

$$\phi_s = \exp\{\int q^{(\Theta)}(\alpha) \log \alpha_s d\alpha\},\ \alpha = (\alpha_1, \ldots, \alpha_m)' \ .$$

We write $\phi = (\phi_1, \ldots, \phi_m)'$.

This iterative procedure can be initialised by taking, for each i and s,

$$r_{is} = \frac{p_s(y_i) a_s^{(0)}}{\sum_{s=1}^m p_s(y_i) a_s^{(0)}} \ .$$

3 Local Convergence of the Iterative Procedure

An iterative procedure is said to *converge locally* to a limit if the iterates converge to that limit whenever the starting values are sufficiently near to the limit. We suppose that the true value of the parameter Θ is Θ^*, with $0 < \theta_s^* < 1,\ s = 1, \ldots, m$ and $\sum_{s=1}^m \theta_s^* = 1$. In this section we shall show that the algorithm presented in the previous section converges locally to Θ^*.

In the kth iteration, we write

$$\theta_s^{(k)} = \frac{1}{n} \sum_{i=1}^n r_{is}^{(k)} \ , \quad \Theta^{(k)} = (\theta_1^{(k)}, \ldots, \theta_m^{(k)})' \ ,$$

where the notation for r now recognises the fact that the r-values change from iteration to iteration.

We define the variational Bayesian estimator of a parameter as its variational posterior mean. Then the procedure given by (2) and (3) suggests the following algorithm for calculating the variational Bayesian estimate of Θ: starting with some initial value $\Theta^{(1)}$, successive iterates are defined inductively by

$$\Theta^{(k+1)} = \Phi_n(\Theta^{(k)}) \tag{4}$$

for $k = 1, 2, \ldots$, where $\Phi_n = (\Phi_n^1, \ldots, \Phi_n^m)'$,

$$\Phi_n^s(\Theta^{(k)}) \triangleq \frac{1}{n} \sum_{i=1}^n \frac{p_s(y_i) \phi_s^{(k)}}{\sum_{s=1}^m p_s(y_i) \phi_s^{(k)}} \ ,$$

$$\phi_s^{(k)} = \exp\{\int q^{(\Theta)(k)}(\alpha) \log \alpha_s d\alpha\} \ , \tag{5}$$

and

$$q^{(\Theta)(k)}(\alpha) \sim \mathcal{D}(\{a_s^{(0)} + n\theta_s^{(k)}\}_{s=1}^m) \ . \tag{6}$$

We have the following theorem.

Theorem 1. *With probability 1 as n approaches infinity, the iterative procedure (4) converges locally to the true value Θ^*.*

Proof. We first prove that, with probability 1 as n approaches infinity, the operator Φ_n is *locally contractive*; that is, there exists a number λ, $0 \leq \lambda < 1$, such that

$$\|\Phi_n(\bar{\Theta}) - \Phi_n(\Theta^*)\| \leq \lambda \|\bar{\Theta} - \Theta^*\| \ ,$$

whenever $\bar{\Theta}$ lies sufficiently near Θ^*.

Since $\bar{\Theta}$ is near Θ^*, one can write

$$\Phi_n(\bar{\Theta}) - \Phi_n(\Theta^*) = \nabla\Phi_n(\Theta^*)(\bar{\Theta} - \Theta^*) + O(\|\bar{\Theta} - \Theta^*\|^2) \ ,$$

where $\nabla\Phi_n(\Theta^*)$ denotes the gradient of $\Phi_n(\Theta)$ evaluated at Θ^*. Consequently, it is sufficient to show that $\nabla\Phi_n(\Theta^*)$ converges with probability 1 to an operator which has norm less than 1.

From the definition of the operator Φ_n, we have, for $s, j = 1, \ldots, m$, that

$$\nabla_j \Phi_n^s(\Theta^*) = \frac{1}{n} \sum_{i=1}^{n} \frac{p_s(y_i)\phi_s^j(\sum_{t=1}^m p_t(y_i)\phi_t) - p_s(y_i)\phi_s(\sum_{t=1}^m p_t(y_i)\phi_t^j)}{(\sum_{t=1}^m p_t(y_i)\phi_t)^2} \ ,$$

where ϕ_s is as given in (5)-(6) but with $\Theta^{(k)}$ replaced by Θ^* in (6), and where ϕ_s^j denotes the derivative of ϕ_s with respect to θ_j. However, it is obvious that, as n tends to infinity, the mean of θ_s corresponding to the density $q^{(\Theta)}(\Theta)$ is

$$\frac{a_s^{(0)} + n\theta_s^*}{\sum_{s=1}^m a_s^{(0)} + n} \to \theta_s^* \ ,$$

the covariance between θ_s and θ_t, for $s \neq t$, is

$$- \frac{(a_s^{(0)} + n\theta_s^*)(a_t^{(0)} + n\theta_t^*)}{(\sum_{s=1}^m a_s^{(0)} + n)^2(\sum_{s=1}^m a_s^{(0)} + n + 1)} = O(\frac{1}{n}) \to 0 \ ,$$

and the variance of θ_s is

$$\frac{(a_s^{(0)} + n\theta_s^*)(\sum_{s=1}^m a_s^{(0)} - a_s^{(0)} + n - n\theta_s^*)}{(\sum_{s=1}^m a_s^{(0)} + n)^2(\sum_{s=1}^m a_s^{(0)} + n + 1)} = O(\frac{1}{n}) \to 0 \ .$$

From these we show in Appendix A that, as n tends to infinity,

$$\phi_s \to \theta_s^* \ ,$$

$$\phi_s^j \to \begin{cases} 1, \text{ if } j = s \ ; \\ 0, \text{ if } j \neq s \ . \end{cases}$$

Thus from Appendix B we obtain that, with probability 1, as n tends to infinity,

$$\begin{aligned} \nabla_j \Phi_n^s(\Theta^*) &\to \begin{cases} \mathbb{E}\left\{ \frac{p_s(y_i)(\sum_{s=1}^m p_s(y_i)\theta_s^*) - p_s^2(y_i)\theta_s^*}{(\sum_{s=1}^m p_s(y_i)\theta_s^*)^2} \right\} \ , \text{ if } j = s \ ; \\ -\mathbb{E}\left\{ \frac{p_s(y_i)p_j(y_i)\theta_s^*}{(\sum_{s=1}^m p_s(y_i)\theta_s^*)^2} \right\} \ , \qquad\qquad \text{if } j \neq s \ , \end{cases} \\ &= \mathbb{E}\left\{ \frac{p_s(y_i)}{\sum_{s=1}^m p_s(y_i)\theta_s^*} \right\} \delta_{js} - \theta_s^* \cdot \mathbb{E}\left\{ \frac{p_s(y_i)p_j(y_i)}{(\sum_{s=1}^m p_s(y_i)\theta_s^*)^2} \right\} \ , \end{aligned}$$

where δ_{js} is the Kronecker delta function and the expectation corresponds to the true model in which $\Theta = \Theta^*$. Note that here y_i represents any random vector distributed according to the probability density of the form (1).

Since

$$\mathbb{E}\left\{\frac{p_s(y_i)}{\sum_{s=1}^m p_s(y_i)\theta_s^*}\right\} = \int \frac{p_s(y_i)}{\sum_{s=1}^m p_s(y_i)\theta_s^*} \cdot p(y_i)dy_i = 1 \ , \tag{7}$$

where $p(y_i) = \sum_{s=1}^m p_s(y_i)\theta_s^*$ is the (true unconditional) probability density of each observation, the last expression can be rewritten as

$$\nabla\Phi_n(\Theta^*) \to I - \Xi\Psi \ ,$$

where I denotes the identity matrix, $\Xi \stackrel{\triangle}{=} \mathrm{diag}(\theta_1^*, \theta_2^*, \cdots, \theta_m^*)$ and

$$\Psi = \mathbb{E}\begin{pmatrix} \nu_1 \\ \nu_2 \\ \vdots \\ \nu_m \end{pmatrix}\begin{pmatrix} \nu_1 \ \nu_2 \cdots \nu_m \end{pmatrix} \ , \quad \nu_s = \frac{p_s(y_i)}{\sum_{s=1}^m p_s(y_i)\theta_s^*} \ . \tag{8}$$

Obviously Ψ is a positive definite matrix. Therefore, as n tends to infinity,

$$\nabla\Phi_n(\Theta^*) < I \ .$$

Next we prove that $I - \Xi\Psi \geq 0$. Since Ξ is a positive diagonal matrix, it suffices to show that

$$\Theta'\Xi^{-1}\Theta \geq \Theta'\Xi^{-1}\Xi\Psi\Theta = \Theta'\Psi\Theta \tag{9}$$

for any $\Theta = (\theta_1, \ldots, \theta_m)'$.

In fact, one has

$$\begin{aligned}\Theta'\Psi\Theta &= \mathbb{E}(\theta_1\nu_1 + \theta_2\nu_2 + \cdots + \theta_m\nu_m)^2 \\ &= \mathbb{E}\left[\sum_{s=1}^m (\theta_s^{*-1}\theta_s \cdot \theta_s^*\nu_s)\right]^2 \ .\end{aligned}$$

As a corollary of Schwarz's inequality we have that, if $\eta_s \geq 0$ for $s = 1, \cdots, m$ and $\sum_{s=1}^m \eta_s = 1$, then

$$|\sum_{s=1}^m \xi_s\eta_s|^2 \leq \sum_{s=1}^m \xi_s^2\eta_s \tag{10}$$

for all $\{\xi_s\}_{s=1,\cdots,m}$ (see [11]).

Applying this result and noting that $\sum_{s=1}^m \theta_s^*\nu_s = 1$, we obtain

$$\begin{aligned}\Theta'\Psi\Theta &\leq \mathbb{E}\left[\sum_{s=1}^m (\theta_s^{*-1}\theta_s)^2\theta_s^*\nu_s\right] = \mathbb{E}\left[\sum_{s=1}^m \theta_s^{*-1}\theta_s^2\nu_s\right] \\ &= \sum_{s=1}^m \theta_s^{*-1}\theta_s^2\mathbb{E}\nu_s = \sum_{s=1}^m \theta_s^{*-1}\theta_s^2 = \Theta'\Xi^{-1}\Theta \ ,\end{aligned}$$

because of (7).

Thus we have proved that $\nabla\Phi_n(\Theta^*)$ converges with probability 1 to an operator with norm less than 1, and consequently the operator Φ_n is *locally contractive*.

From the above proof, it is then obvious that $\Phi_n(\Theta^*)$ tends to Θ^* as n approaches infinity. Since

$$\begin{aligned}\|\Theta^{(k+1)} - \Theta^*\| &\leq \|\Phi_n(\Theta^{(k)}) - \Phi_n(\Theta^*)\| + \|\Phi_n(\Theta^*) - \Theta^*\| \\ &\leq \lambda\|\Theta^{(k)} - \Theta^*\| + \|\Phi_n(\Theta^*) - \Theta^*\| \ ,\end{aligned}$$

the iterative procedure (4) converges locally to the true value Θ^* as n approaches infinity . □

4 The Convergence Rate of the Variational Bayes Estimator

In this section we consider the rate at which the variational Bayes estimator converges to the maximum likelihood estimator (MLE). Suppose the sample size n is large. Let $\hat{\Theta}^n = (\hat{\theta}_1^n, \ldots, \hat{\theta}_m^n)'$ and $\tilde{\Theta}^n = (\tilde{\theta}_1^n, \ldots, \tilde{\theta}_m^n)'$ denote the fixed point of iteration (4) in the neighbourhood of the true value and the variational Bayes estimator, respectively; that is, from (4),

$$\hat{\theta}_s^n = \frac{1}{n}\sum_{i=1}^{n} \frac{p_s(y_i)\phi_s}{\sum_{s=1}^m p_s(y_i)\phi_s} \ , \tag{11}$$

$$q^{(\Theta)}(\alpha) \sim \mathcal{D}(\{a_s^{(0)} + n\hat{\theta}_s^n\}_{s=1}^m) \ ,$$

$$\phi_s = \exp\{\int q^{(\Theta)}(\alpha)\log\alpha_s d\alpha\} \ ,$$

and

$$\tilde{\theta}_s^n \triangleq \int \alpha q^{(\Theta)}(\alpha)d\alpha = \frac{a_s^{(0)} + n\hat{\theta}_s^n}{\sum_{s=1}^m a_s^{(0)} + n} \ .$$

Hence $\tilde{\Theta}^n = \hat{\Theta}^n + O(1/n)$.

Suppose that $\bar{\Theta}^n = (\bar{\theta}_1^n, \ldots, \bar{\theta}_m^n)'$ is the strongly consistent MLE of the parameter Θ; that is, $\bar{\theta}_s^n$ is the solution of the following likelihood equation (see Peters and Walker [11], Redner and Walker [12]):

$$l_s^n(\Theta) \triangleq \theta_s - \frac{1}{n}\sum_{i=1}^{n} \frac{p_s(y_i)\theta_s}{\sum_{s=1}^m p_s(y_i)\theta_s} = 0 \ ,$$

for $s = 1, \ldots, m$. Let $l^n = (l_1^n, \ldots, l_m^n)'$. Then we have the following lemma.

Lemma 1. *With probability 1 as n tends to infinity, the gradient of the likelihood function l^n, evaluated at the true value Θ^*, converges uniformly to $\Xi\Psi$, where Ξ and Ψ are defined as in the previous section.*

Proof. After some straightforward manipulations, the convergence is a direct corollary of the strong law of large numbers. □

Meanwhile, from (11) we have that

$$\begin{aligned} 0 &= \hat{\theta}_s^n - \frac{1}{n}\sum_{i=1}^{n} \frac{p_s(y_i)\phi_s}{\sum_{s=1}^m p_s(y_i)\phi_s} \\ &= l_s^n(\hat{\Theta}^n) + \frac{1}{n}\sum_{i=1}^{n}\left[\frac{p_s(y_i)\hat{\theta}_s^n}{\sum_{s=1}^m p_s(y_i)\hat{\theta}_s^n} - \frac{p_s(y_i)\phi_s}{\sum_{s=1}^m p_s(y_i)\phi_s}\right] \\ &= l_s^n(\hat{\Theta}^n) + \frac{1}{n}\sum_{i=1}^{n}\frac{\sum_{j=1}^m p_s(y_i)p_j(y_i)(\hat{\theta}_s^n\phi_j - \phi_s\hat{\theta}_j^n)}{[\sum_{s=1}^m p_s(y_i)\hat{\theta}_s^n][\sum_{s=1}^m p_s(y_i)\phi_s]} \ . \end{aligned}$$

It follows from Appendix A that $\phi_s = \hat{\theta}_s^n + O(1/n)$, so $\hat{\theta}_s^n\phi_j - \phi_s\hat{\theta}_j^n = O(1/n)$. Thus the second term is of order $O(1/n)$. From Taylor's expansion the first term can be rewritten as

$$\begin{aligned} l^n(\hat{\Theta}^n) &= l^n(\bar{\Theta}^n) + \nabla l^n(\bar{\Theta}^n + \lambda(\hat{\Theta}^n - \bar{\Theta}^n))(\hat{\Theta}^n - \bar{\Theta}^n) \\ &= \nabla l^n(\bar{\Theta}^n + \lambda(\hat{\Theta}^n - \bar{\Theta}^n))(\hat{\Theta}^n - \bar{\Theta}^n) \ , \end{aligned}$$

where $0 \le \lambda \le 1$. Thus, we obtain

$$0 = \nabla l^n(\bar{\Theta}^n + \lambda(\hat{\Theta}^n - \bar{\Theta}^n))(\hat{\Theta}^n - \bar{\Theta}^n) + O(\frac{1}{n}) \ .$$

We have proved that $\hat{\Theta}^n$ converges to Θ^*, and it is well known that $\bar{\Theta}^n$ tends to Θ^*, so it follows from Lemma 1 that $\nabla l^n(\bar{\Theta}^n + \lambda(\hat{\Theta}^n - \bar{\Theta}^n))$ converges to $\Xi\Psi$. Therefore, we have that $\hat{\Theta}^n = \bar{\Theta}^n + O(1/n)$, and consequently that $\tilde{\Theta}^n = \bar{\Theta}^n + O(1/n)$.

Remark 1. It is known that the (non-variational) Bayes estimator and the MLE get closer to each other at rate $O(1/n)$, so the variational Bayes estimator is asymptotically consistent at the same rate.

5 Asymptotic Normality of the Variational Posterior Distribution

In this section, we show that the variational posterior distribution for the parameter Θ obtained by the iterative procedure has also the property of asymptotic normality. This implies that the variational posterior becomes more and more concentrated around the true parameter value as the sample size grows.

Suppose the sample size n is large. Denote by $\hat{\Theta}^n = (\hat{\theta}_1^n, \ldots, \hat{\theta}_m^n)'$ the fixed point of the iteration (4). Thus the variational posterior density of Θ at $\hat{\Theta}^n$ is

$$q_n^{(\Theta)}(\Theta) \sim \mathcal{D}(\{\hat{a}_s^{(n)}\}_{s=1}^m) \ , \tag{12}$$

where $\hat{a}_s^{(n)} = a_s^{(0)} + \sum_{i=1}^n \hat{r}_{is}$ and $\hat{r}_{is} = q_i^{(S)}(s_i = s)$.

In the rest of the paper, we express θ_m explicitly as $1 - \sum_{s=1}^{m-1} \theta_s$. Thus the density (12) can be rewritten as a density of exponential family type:

$$\begin{aligned} q_n^{(\Theta)}(\Theta) &\propto \exp\left\{(\hat{a}_1^{(n)} - 1)\log\theta_1 + \cdots + (\hat{a}_m^{(n)} - 1)\log(1 - \textstyle\sum_{s=1}^{m-1}\theta_s)\right\} \\ &\triangleq \exp\{h'(\Theta)\beta - \alpha\psi(\Theta)\} \ , \end{aligned} \tag{13}$$

where

$$\beta \triangleq (\hat{a}_1^{(n)} - 1, \ldots, \hat{a}_{m-1}^{(n)} - 1)' \ ,$$

$$h(\Theta) \triangleq (\log\theta_1, \ldots, \log\theta_{m-1})' \ ,$$

$$\alpha \triangleq 1 - \hat{a}_m^{(n)}, \text{ and } \psi(\Theta) \triangleq \log(1 - \sum_{s=1}^{m-1} \theta_s) \ .$$

In Wang and Titterington [13] we have proved the asymptotic normality of the variational posterior distributions associated with natural exponential families with missing values. Since the algorithm (4) has been proved to be convergent, along the same lines it can be easily checked that the result holds for the density (13). Therefore, if $\tilde{\Theta}^n = (\tilde{\theta}_1^n, \ldots, \tilde{\theta}_{m-1}^n)'$ is the maximiser of $h'(\Theta)\beta - \alpha\psi(\Theta)$ and we define $\Sigma^n = -[\nabla^2 \log q_n^{(\Theta)}(\tilde{\Theta}^n)]^{-1}$, we have the following theorem.

Theorem 2. *As n approaches infinity, the variational posterior density $q_n^{(\Theta)}(\cdot)$ converges in distribution to a normal density with mean $\tilde{\Theta}^n$ and covariance matrix Σ^n.*

More clearly, from the definition we can actually evaluate $\tilde{\Theta}^n$ explicitly as

$$\tilde{\theta}_s^n = \frac{\hat{a}_s^{(n)} - 1}{\sum_{s=1}^m (\hat{a}_s^{(n)} - 1)} = \frac{a_s^{(0)} + \sum_{i=1}^n \hat{r}_{is} - 1}{\sum_{s=1}^m a_s^{(0)} + n - m} \ , \quad s = 1, \ldots, m-1 \ .$$

Thus the covariance matrix Σ^n can be expressed as

$$\begin{pmatrix} \frac{\hat{a}_1^{(n)}-1}{(\tilde{\theta}_1^n)^2} + \frac{\hat{a}_m^{(n)}-1}{(\tilde{\theta}_m^n)^2} & & & & \\ & \ddots & & \frac{\hat{a}_m^{(n)}-1}{(\tilde{\theta}_m^n)^2} & \\ & & \ddots & & \\ & \frac{\hat{a}_m^{(n)}-1}{(\tilde{\theta}_m^n)^2} & & \ddots & \\ & & & & \frac{\hat{a}_{m-1}^{(n)}-1}{(\tilde{\theta}_{m-1}^n)^2} + \frac{\hat{a}_m^{(n)}-1}{(\tilde{\theta}_m^n)^2} \end{pmatrix}^{-1} ,$$

where $\tilde{\theta}_m^n$ is defined as $1 - \sum_{s=1}^{m-1} \tilde{\theta}_s^n$.

It follows from Theorem 1 and the law of large numbers that $\frac{1}{n}\sum_{i=1}^{n}\hat{r}_{is}$ converges to θ_s^*, and hence we obtain that $\tilde{\Theta}^n$ converges to the true value Θ^* and $n\Sigma^n$ converges to the matrix

$$\begin{pmatrix} \theta_1^{*-1}+\theta_m^{*-1} & & & & \\ & \ddots & & \theta_m^{*-1} & \\ & & \ddots & & \\ & \theta_m^{*-1} & & \ddots & \\ & & & & \theta_{m-1}^{*-1}+\theta_m^{*-1} \end{pmatrix}^{-1}$$

which is equal to

$$\begin{pmatrix} \theta_1^*(1-\theta_1^*) & & & & \\ & \ddots & & -\theta_i^*\theta_j^* & \\ & & \ddots & & \\ & -\theta_i^*\theta_j^* & & \ddots & \\ & & & & \theta_{m-1}^*(1-\theta_{m-1}^*) \end{pmatrix} \triangleq \Lambda \ . \tag{14}$$

There have been a large number of contributions about the asymptotic normality of maximum likelihood estimators and posterior distributions; see for instance Walker [14], Heyde and Johnstone [15], Chen [16], Bernardo and Smith [17] and Ghosal, Ghosh and Samanta [18]. In the setting of mixture models, letting $\bar{\Theta}^n = (\bar{\theta}_1^n, \ldots, \bar{\theta}_{m-1}^n)'$ be the strongly consistent maximum likelihood estimator of the parameter Θ and $\mathcal{I}(\Theta)$ the Fisher information matrix, Redner and Walker [12] showed the following theorem.

Theorem 3. *As n tends to infinity, $\sqrt{n}(\bar{\Theta}^n - \Theta^*)$ is asymptotically normally distributed with mean zero and covariance matrix $\mathcal{I}(\Theta^*)^{-1}$.*

From Theorems 2 and 3, the limiting densities of the variational Bayes density and the MLE have the same mean (i.e. the true value Θ^*). However one can see that their covariance matrices are not equal in general. An interesting question is: how can they be compared? In the next section we will study this issue.

6 Fisher Information and Covariance Associated with Variational Bayes Approximation

In this section, we denote by y any random vector distributed according to the probability density of the form (1). Therefore, the Fisher information matrix is given by

$$\mathcal{I}(\Theta) = \int [\nabla \log p(y|\Theta)][\nabla \log p(y|\Theta)]' p(y|\Theta) dy \ . \tag{15}$$

Let $L(\Theta) = \log p(y|\Theta)$ and define ν as in (8). After a straightforward calculation the Fisher information matrix (15) can be rewritten as

$$\mathcal{I}(\Theta) = \int VV' p(y|\Theta) dy = \mathbb{E}[VV'] \ ,$$

where $V = (\nu_1 - \nu_m, \ldots, \nu_{m-1} - \nu_m)'$.

Proposition 1. *Letting Λ denote the limiting covariance matrix of the variational posterior of the parameter Θ, we have that $\mathcal{I}(\Theta^*)^{-1} \geq \Lambda$.*

Proof. Since Λ is positive definite, it is sufficient to show that

$$\Theta' \mathcal{I}(\Theta^*) \Theta \leq \Theta' \Lambda^{-1} \Theta \tag{16}$$

for any $\Theta = (\theta_1, \ldots, \theta_{m-1})'$.

In fact, along the same lines as the proof of inequality (9) one can obtain

$$\Theta' \mathbb{E}(VV') \Theta \leq \sum_{s=1}^{m} \theta_s^2 \theta_s^{*-1} \ ,$$

where $\theta_m \stackrel{\triangle}{=} -\sum_{s=1}^{m-1} \theta_s$.

On the other hand, by (14) it can be easily checked that

$$\Theta' \Lambda^{-1} \Theta = \sum_{s=1}^{m} \theta_s^2 \theta_s^{*-1} \ .$$

The proof is complete. □

By (10) equality in (16) holds if and only if the mixture model (1) has only one component or the supports of the component densities are disjoint. If the components are well separated or have little overlap, the mixture distribution can be regarded approximately as multinomial. In this case, at a given observation y_i, there exists one $p_s(y_i)$ which is far larger than the others, and therefore the inverse of the Fisher information matrix is close to the covariance matrix of the variational posterior distribution. Proposition 1 shows that in general the covariance matrix from the variational Bayes approximation is 'too small' compared with that for the MLE, so that resulting interval estimates for the parameters will be too narrow.

7 Numerical Experiments

We demonstrate our results with a simple mixture of two known normal densities.

First we fix the two normal densities to have means of 2 and 4 and unit variance, and generate a total of 100 observations using $\theta = 0.65$. For different sample sizes up to $n = 100$ the MLE $\bar{\theta}^n$ and the variational Bayes estimate $\hat{\theta}^n$ based on a Beta prior distribution for θ with $a_1^{(0)} = a_2^{(0)} = 1$ are computed using

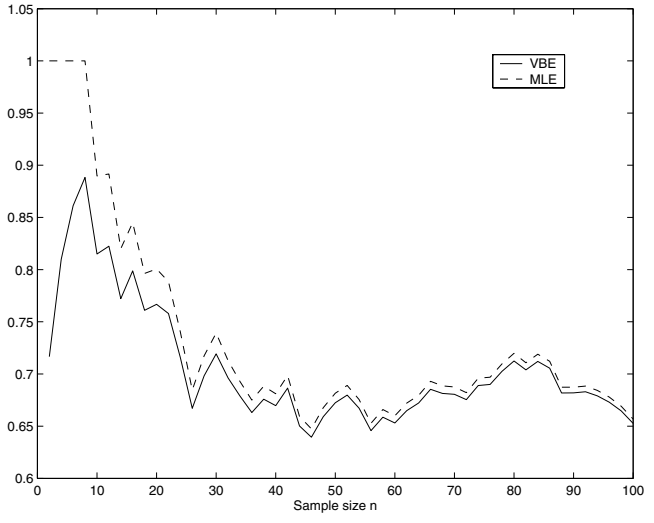


Fig. 1. Variational Bayesian estimate of a mixing weight and MLE plotted against the sample size

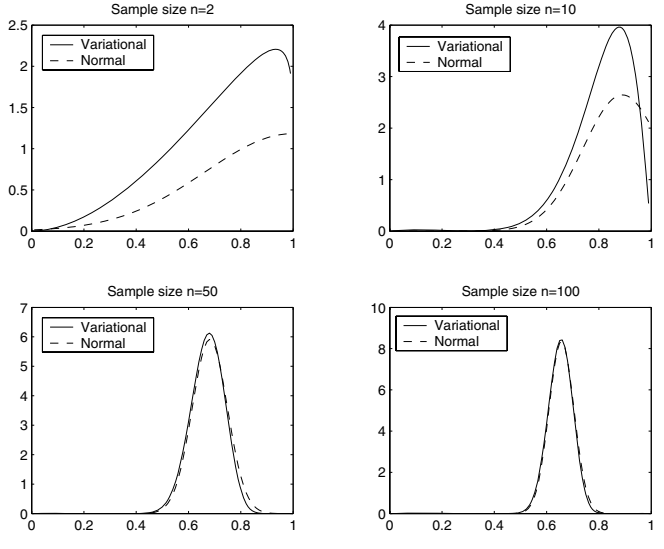


Fig. 2. Variational posteriors of the parameter θ and normal densities for different sample sizes

the first n observations, and are plotted in Figure 1. When the sample size is small, there is a gap between the two estimates, but quickly they track each other very closely as the sample size grows.

In Figure 2, we plot the corresponding variational posterior densities and the normal density $\mathcal{N}(\bar{\theta}^n, \Lambda/n)$ for the sample sizes $n = 2, 10, 50, 100$, where Λ is

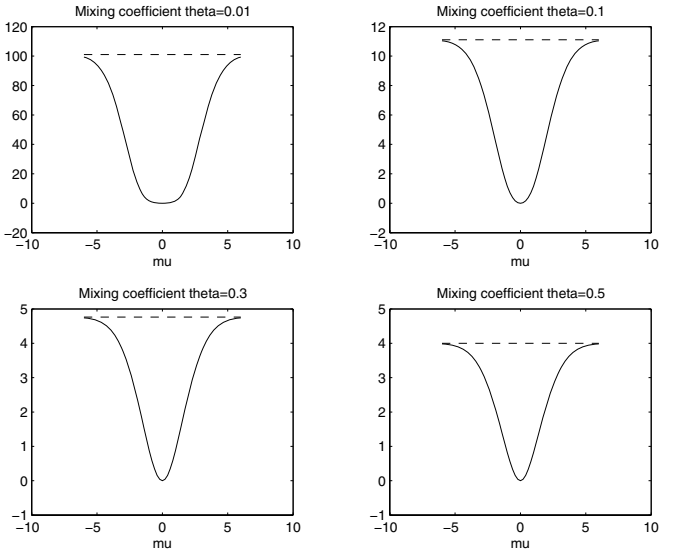


Fig. 3. The inverse of the variance associated with the variational Bayes approximation and the Fisher information for different mixing coefficients. The solid lines denote the Fisher information and the dashed horizontal lines indicate the inverse of the variance for the variational Bayes approximation.

defined in (14). It can be seen that the variational posterior density becomes closer and closer to the limiting normal density.

Finally, to compare the variance Λ associated with the variational Bayes approximation with the Fisher information, we fix one component to have mean zero and unit variance and compute the inverse of the variance Λ and the Fisher information, allowing the other component to have varying mean μ. The results are plotted in Figure 3 for various values of the mixing coefficient θ. The inverse of the variance associated with the variational Bayes approximation does not vary with the changes of μ, whereas the Fisher information does. Obviously, if the components in the mixture model are widely separated, these two quantities are very similar, whereas, if the components are nearly identical, they are very different.

8 Conclusion

We have investigated some properties of variational Bayes approximations, namely consistency and asymptotic normality, and compared the true covariance matrix of the posterior distribution (the inverse of the Fisher information) with the covariance matrix associated with its variational Bayes approximation. The results reveal that in mixture models the point estimate obtained by using a factorised form $q^{(S)}(S)q^{(\Theta)}(\Theta)$ for the posterior distributions of Θ and S does

not lead to bias for large samples, but the interval estimates for the parameters will be too narrow in general.

Acknowledgement

This work was supported by a grant from the UK Science and Engineering Research Council. This work was also supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.

References

1. Attias, H.: Inferring parameters and structure of latent variable models by variational Bayes. In Prade, H., Laskey, K., eds.: Proc. 15th Conference on Uncertainty in Artificial Intelligence, Stockholm, Sweden, Morgan Kaufmann Publishers (1999) 21–30
2. Attias, H.: A variational Bayesian framework for graphical models. In Solla, S., Leen, T., Muller, K.R., eds.: Advances in Neural Information Processing Systems 12. MIT Press, Cambridge, MA (2000) 209–215
3. Beal, M.J.: Variational Algorithms for Approximate Bayesian Inference. PhD thesis, University College London (2003)
4. Corduneanu, A., Bishop, C.M.: Variational Bayesian model selection for mixture distributions. In Richardson, T., Jaakkola, T., eds.: Proceedings Eighth International Conference on Artificial Intelligence and Statistics, Morgan Kaufmann (2001) 27–34
5. Ghahramani, Z., Beal, M.J.: Propagation algorithms for variational Bayesian learning. In Leen, T., Dietterich, T., Tresp, V., eds.: Advances in Neural Information Processing Systems 13. MIT Press, Cambridge, MA (2001) 507–513
6. Humphreys, K., Titterington, D.M.: Approximate Bayesian inference for simple mixtures. In Bethlehem, J.G., van der Heijden, P.G.M., eds.: COMPSTAT2000. Physica-Verlag, Heidelberg (2000) 331–336
7. Jaakkola, T.S., Jordan, M.I.: Bayesian logistic regression: a variational approach. Statistics and Computing **10** (2000) 25–37
8. Jordan, M.I., Ghahramani, Z., Jaakkola, T.S., Saul, L.K.: An introduction to variational methods for graphical models. In Jordan, M.I., ed.: Learning in Graphical Models. MIT Press, Cambridge (1999) 105–162
9. MacKay, D.J.C.: Ensemble learning for hidden Markov models. Technical report, Cavendish Laboratory, University of Cambridge (1997)
10. Penny, W.D., Roberts, S.J.: Variational Bayes for 1-dimensional mixture models. Technical Report PARG-2000-01, Oxford University (2000)
11. Peters, B.C., Walker, H.F.: An iterative procedure for obtaining maximum-likelihood estimates of the parameters for a mixture of normal distributions. SIAM J. Appl. Math. **35** (1978) 362–378
12. Redner, R.A., Walker, H.F.: Mixture densities, maximum likelihood and the EM algorithm. SIAM Review **26** (1984) 195–239
13. Wang, B., Titterington, D.M.: Convergence and asymptotic normality of variational Bayesian approximations for exponential family models with missing values. In Chickering, M., Halpern, J., eds.: Proceedings of the twentieth conference on Uncertainty in Artificial Intelligence, Banff, Canada, AUAI Press (2004) 577–584

14. Walker, A.M.: On the asymptotic behaviour of posterior distributions. J. R. Statist. Soc. B **31** (1969) 80–88
15. Heyde, C.C., Johnstone, I.M.: On asymptotic posterior normality for stochastic processes. J. R. Statist. Soc. B **41** (1979) 184–189
16. Chen, C.F.: On asymptotic normality of limiting density functions with Bayesian implications. J. R. Statist. Soc. B **47** (1985) 540–546
17. Bernardo, J.M., Smith, A.F.M.: Bayesian Theory. John Wiley & Sons, Inc, New York (1994)
18. Ghosal, S., Ghosh, J.K., Samanta, T.: On convergence of posterior distributions. The Annals of Statistics **23** (1995) 2145–2152

Appendix A

First the following conclusion holds. Suppose that $p_n(x)$ is the probability density function of the random variable X_n, that $\mathbb{E}X_n = \mu_n \to \mu$, as $n \to \infty$, and that $\mathrm{Cov}(X_n) = O(1/n)$. Then, for any function $f(\cdot)$ with continuous second-order derivative near μ, we have

$$\mathbb{E}f(X_n) = f(\mu_n) + O(\frac{1}{n}) \ .$$

This follows from the Taylor expansion

$$\begin{aligned} f(X_n) = f(\mu_n) &+ \nabla f(\mu_n) \cdot (X_n - \mu_n) + \frac{1}{2}(X_n - \mu_n)' \nabla^2 f(\mu_n)(X_n - \mu_n) \\ &+ o(\|X_n - \mu_n\|^2) \ , \end{aligned}$$

because then

$$\begin{aligned} \mathbb{E}f(X_n) &= f(\mu_n) + \nabla f(\mu_n) \cdot (\mathbb{E}X_n - \mu_n) \\ &\quad + \frac{1}{2}\mathbb{E}\left[(X_n - \mu_n)' \nabla^2 f(\mu_n)(X_n - \mu_n)\right] + o(\mathbb{E}\|X_n - \mu_n\|^2) \\ &= f(\mu_n) + \frac{1}{2}\mathbb{E}\left[(X_n - \mu_n)' \nabla^2 f(\mu_n)(X_n - \mu_n)\right] + o(\mathbb{E}\|X_n - \mu_n\|^2) \\ &= f(\mu_n) + O(\frac{1}{n}) \ . \end{aligned}$$

Applying this to the case of

$$f(x) = \log x, \quad \text{and} \quad X_n : \ q^{(\Theta)}(\alpha) \sim \mathcal{D}(\{a_s^{(0)} + n\theta_s\}_{s=1}^m) \ ,$$

we easily obtain that

$$\mathbb{E}f(X_n) = \int q^{(\Theta)}(\alpha) \log \alpha_s d\alpha = \log(\mathbb{E}X_n) + O(\frac{1}{n}) \ .$$

Hence, from Taylor expansion, we have

$$\phi_s = \exp\{\int q^{(\Theta)}(\alpha) \log \alpha_s d\alpha\} = \exp\{\log(\mathbb{E}X_n) + O(\frac{1}{n})\}$$

$$
\begin{aligned}
&= \exp\{\log(\mathbb{E}X_n)\} + \exp\{\log(\mathbb{E}X_n)\} \cdot O(\frac{1}{n}) + O(\frac{1}{n^2}) \\
&= \mathbb{E}X_n + \mathbb{E}X_n \cdot O(\frac{1}{n}) \\
&= \theta_s + O(\frac{1}{n}) \to \theta_s \ .
\end{aligned}
$$

We have assumed that $0 < \theta_s^* < 1$ and our conclusions are local in nature, so there is no loss of generality in restricting the discussion to $0 < \varepsilon_0 \leq \theta_s^* \leq 1 - \varepsilon_0 < 1$ for a small positive constant ε_0. Thus the above convergences are also uniform in θ_s. As a result we have

$$
\phi_s^j \to \begin{cases} 1, \text{ if } j = s \ ; \\ 0, \text{ if } j \neq s \ . \end{cases}
$$

Appendix B

We show that, if $F_n(\cdot) \to F_0(\cdot)$ uniformly, then, with probability 1,

$$
\frac{1}{n} \sum_{i=1}^{n} F_n(X_i) \to \mathbb{E}F_0(X_i)
$$

for any sequence $\{X_n\}$ of independent and identically distributed random variables.

This is a direct corollary of the strong law of large numbers and the following calculation:

$$
\begin{aligned}
&|\frac{1}{n} \sum_{i=1}^{n} F_n(X_i) - \mathbb{E}F_0(X_i)| \\
\leq\ &|\frac{1}{n} \sum_{i=1}^{n} F_n(X_i) - \frac{1}{n} \sum_{i=1}^{n} F_0(X_i)| + |\frac{1}{n} \sum_{i=1}^{n} F_0(X_i) - \mathbb{E}F_0(X_i)| \\
\leq\ &\frac{1}{n} \sum_{i=1}^{n} |F_n(X_i) - F_0(X_i)| + |\frac{1}{n} \sum_{i=1}^{n} F_0(X_i) - \mathbb{E}F_0(X_i)| \\
\leq\ &\sup_x |F_n(x) - F_0(x)| + |\frac{1}{n} \sum_{i=1}^{n} F_0(X_i) - \mathbb{E}F_0(X_i)| \ .
\end{aligned}
$$

A Comparison of Condition Numbers for the Full Rank Least Squares Problem

Joab R. Winkler

Department of Computer Science, The University of Sheffield,
Regent Court, 211 Portobello Street, Sheffield S1 4DP,
United Kingdom
j.winkler@dcs.shef.ac.uk

Abstract. Condition numbers of the full rank least squares (LS) problem $\min_x \|Ax - b\|_2$ are considered theoretically and their computational implementation is compared. These condition numbers range from a simple normwise measure that may overestimate by several orders of magnitude the true numerical condition of the LS problem, to refined componentwise and normwise measures. Inequalities that relate these condition numbers are established, and it is concluded that the solution x_0 of the LS problem may be well-conditioned in the normwise sense, even if one of its components is ill-conditioned. It is shown that the refined condition numbers are ill-conditioned in some circumstances, the cause of this ill-conditioning is identified, and its implications are discussed.

1 Introduction

Four condition numbers of the least squares (LS) problem

$$\min_x \|Ax - b\|, \qquad \|\cdot\| = \|\cdot\|_2, \tag{1}$$

where $A \in \mathbb{R}^{m\times n}$, $x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $m \geq n$, rank $A = n$, and A is known exactly, are considered theoretically and computationally. These condition numbers range from a simple normwise measure to refined measures that consider the numerical condition of each component of the solution x_0 of the LS problem. The simple normwise condition number, which is given by $\kappa_{LS}(A, \theta) = \kappa(A)/\cos\theta$, where $\kappa(A)$ is equal to the ratio of the largest to the smallest singular value of A, and $\cos\theta \neq 1$ if b does not lie in the column space of A, reduces to $\kappa(A)$ if $m = n$. It is shown that although the refined condition numbers yield much more information than $\kappa_{LS}(A, \theta)$, their computational implementation may pose problems because they are ill-conditioned in some circumstances.

The solution x_0 of (1) satisfies

$$A^T Ax = A^T b,$$

and is given by

$$x_0 = A^\dagger b = VS^\dagger U^T b = \sum_{i=1}^{n} \frac{u_i^T b}{\sigma_i} v_i,$$

J. Winkler, N. Lawrence, M. Niranjan (Eds.): Machine Learning Workshop, LNAI 3635, pp. 296–318, 2005.

where $X^\dagger = \left(X^T X\right)^{-1} X^T$ is the left inverse of X, USV^T is the singular value decomposition (SVD) of A, v_i is the i'th column of V, u_i^T is the i'th row of U, and $\sigma_i, i = 1, \ldots, n$, are the singular values of A, arranged in non-increasing order.

Four condition numbers of the LS problem are considered in Section 2, and it is shown in Section 3 that they are related by a series of inequalities. The numerical stability of the condition numbers is discussed in Section 4, and it is shown that the simplest condition number $\kappa_{LS}(A, \theta)$ can be computed reliably, assuming that $\cos\theta \approx 1$, but it may overestimate, by several orders of magnitude, the true numerical condition of x_0. These properties of $\kappa_{LS}(A, \theta)$ must be compared with those of the refined normwise and componentwise condition numbers, which provide more detailed information but are, as noted above, ill-conditioned in some circumstances. This instability limits their practical value, and a regularised approximation to the refined normwise condition number is considered in Section 5. Several examples that illustrate the results are given, and Section 6 contains a summary of the paper.

2 Condition Numbers for the Least Squares Problem

Four condition numbers of the solution x_0 of the LS problem are considered in this section. These range from a simple normwise measure to refined componentwise measures.

2.1 A Simple Normwise Condition Number

It is shown in [8] that a simple normwise condition number of x_0 is

$$\kappa_{LS}(A, \theta) = \max_{\delta b, b \in \mathbb{R}^m} \frac{\Delta x_0}{\Delta b} = \frac{\sigma_1}{\sigma_n} \frac{1}{\cos\theta} = \frac{\kappa(A)}{\cos\theta}, \tag{2}$$

where

$$\Delta x_0 = \frac{\|\delta x_0\|}{\|x_0\|}, \qquad \Delta b = \frac{\|\delta b\|}{\|b\|},$$

the residual r and angle θ satisfy

$$r := Ax_0 - b, \qquad \|r\| = \|b\| \sin\theta \qquad \text{and} \qquad \|Ax_0\| = \|b\| \cos\theta, \tag{3}$$

and $\kappa(A) = \sigma_1/\sigma_n$ is the condition number of A.

Practical problems rarely generate situations in which $\theta \approx \pi/2$ because it is usual to pose the LS problem such that b can be approximated, with a small error, by a linear combination of the columns of A. The matrix A incorporates the model that represents the data, for example, the basis functions that are used in regression, and a poor choice of basis functions may lead to a large value of θ. Even if θ is small, $\kappa_{LS}(A, \theta)$ is large if A is ill-conditioned.

The main disadvantage of $\kappa_{LS}(A, \theta)$ follows immediately from (2). In particular, it is a worst case upper bound of the true numerical condition of x_0, taken over all vectors b and δb, and the following example shows that $\kappa_{LS}(A, \theta)$ may overestimate the true numerical condition by several orders of magnitude.

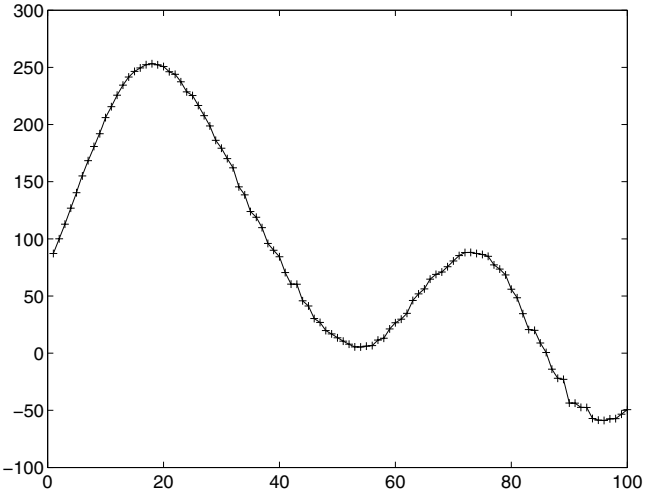


Fig. 1. A set of 100 points and their approximating curve for $b = b_1$

Example 1. Regression was performed on two sets of 100 data points $(x_i, y_i)\,, i = 1, \ldots, 100$, in the interval $I = [1, \ldots, 20]$ using a linear combination of 33 radial basis functions,

$$y_i = \sum_{k=1}^{33} a_k \exp\left(-\frac{(x_i - d_k)^2}{2\sigma_d^2}\right), \qquad i = 1, \ldots, 100,$$

where $\sigma_d = 1.35$ and the centres d_k of the basis functions are not uniformly distributed in I. The coefficient matrix A is therefore of order 100×33, b stores the function values y_i, and x stores the coefficients a_k of the radial basis functions.

The first set of 100 points, which are stored in the vector $b = b_1$, and their regression curve are shown in Figure 1, and it is seen that the error between the exact data points and the approximating curve is small. The LS problem was solved twice, once for the exact data $b = b_1$, and once for perturbed data $b_1 + \delta b_1$, where the elements of δb_1 are drawn from a zero mean Gaussian distribution, such that $\|b_1\| \,/\, \|\delta b_1\| = 2.9 \times 10^5$. The solutions of the LS problem for these problems are shown in Figure 2, and it is seen that the noise has a significant effect on the coefficients of the radial basis functions, such that the solution in the presence of noise has no value.

The experiment was repeated for the second set of 100 data points, which are stored in the vector $b = b_2$. These points are shown in Figure 3, which also shows the regression curve that is obtained using the same matrix A because the same radial basis functions were used. The LS problem was therefore solved twice, once for the exact data $b = b_2$, and once for perturbed data $b_2 + \delta b_2$ where $\|b_2\| \,/\, \|\delta b_2\| = 1.2 \times 10^3$. Figure 4 shows the coefficients of the regression curve for these two cases, and it is seen that the noise has very little effect on the exact coefficients.

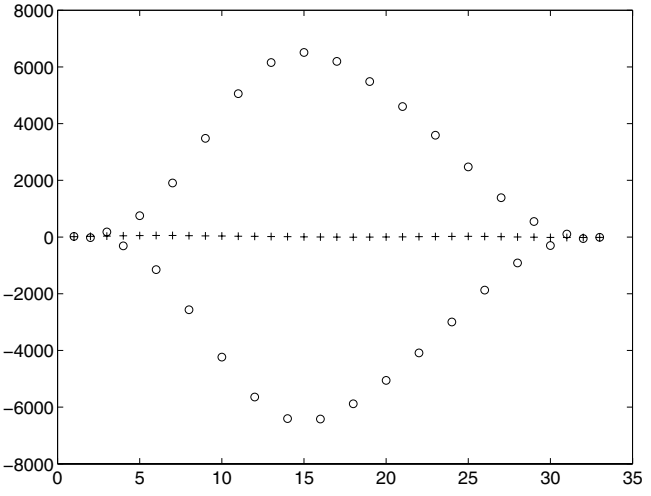


Fig. 2. The coefficients of the radial basis functions for the exact (+) and noisy (o) data for the curve shown in Figure 1

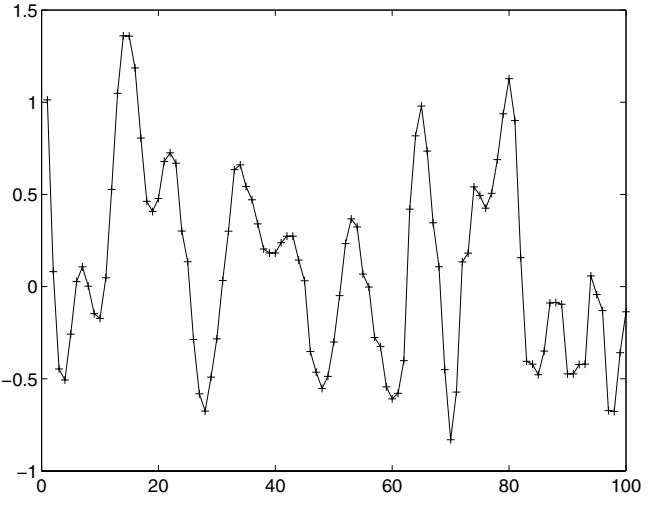


Fig. 3. A set of 100 points and their approximating curve for $b = b_2$

The condition number of A is $\kappa(A) = 5.1 \times 10^8$ and the vectors b_1 and b_2 lie in the column space of A, from which it follows that $\kappa_{LS}(A, \theta) = \kappa(A)$ for $b = b_1$ and $b = b_2$. Figures 2 and 4 show that this measure fails to distinguish between the vectors b_1 and b_2, that is, the vectors b for which the LS problem is well-conditioned, and the vectors b for which this problem is ill-conditioned. □

The failure of $\kappa_{LS}(A, \theta)$ to quantify the true numerical condition of a linear algebraic equation is a disadvantage, which is overcome by the effective condition number.

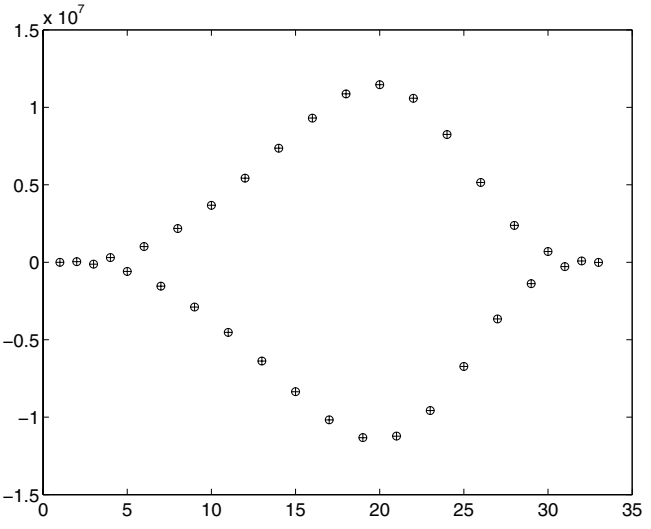


Fig. 4. The coefficients of the radial basis functions for the exact (+) and noisy (o) data for the curve shown in Figure 3

2.2 The Effective Condition Number

The effective condition number $S_{LS}\,(A, b)$ is superior to the condition number $\kappa_{LS}\,(A, \theta)$ because it distinguishes between the vectors b for which the LS problem is ill-conditioned, and the vectors b for which the LS problem is well-conditioned. It is defined as the maximum value of the ratio $\Delta x_0/\Delta b$ over all perturbations $\delta b \in \mathbb{R}^m$,

$$S_{LS}\,(A, b) := \max_{\delta b \in \mathbb{R}^m} \frac{\Delta x_0}{\Delta b} = \frac{1}{\sigma_n} \frac{\|b\|}{\|x_0\|} = \frac{1}{\sigma_n} \frac{\|b\|}{\|S^{\dagger} U^T b\|}, \tag{4}$$

and it satisfies the inequality

$$1 \leq S_{LS}(A, b) \leq \kappa_{LS}\,(A, \theta)\,. \tag{5}$$

The lower bound in (5) follows immediately, and derivation of the upper bound is simplified by the introduction of the vector $c \in \mathbb{R}^m$,

$$c = \{c_i\}_{i=1}^m = U^T b, \tag{6}$$

from which it follows that (4) can be written as

$$S_{LS}(A, Uc) = \frac{\|c\|}{\sigma_n \sqrt{\sum_{i=1}^n \left(\frac{c_i}{\sigma_i}\right)^2}}. \tag{7}$$

Since

$$\sigma_n \sqrt{\sum_{i=1}^n \left(\frac{c_i}{\sigma_i}\right)^2} = \frac{\sigma_n}{\sigma_1} \sqrt{\sum_{i=1}^n \left(\frac{\sigma_1}{\sigma_i} c_i\right)^2} \geq \frac{1}{\kappa(A)} \sqrt{\sum_{i=1}^n c_i^2},$$

it follows that

$$S_{LS}(A, Uc) \leq \kappa(A) \frac{\|c\|}{\sqrt{\sum_{i=1}^{n} c_i^2}}. \tag{8}$$

The application of the SVD of A to (3) shows that

$$\cos\theta = \frac{\|Ax_0\|}{\|b\|} = \frac{\left\|SS^{\dagger}c\right\|}{\|c\|} = \frac{\sqrt{\sum_{i=1}^{n} c_i^2}}{\|c\|},$$

and thus the upper bound in (5) follows from (8).

Example 2. Consider Example 1 in which the LS problem was considered for two right hand side vectors b. The effective condition numbers of this problem for these vectors are

$$S_{LS}(A, b_1) = 4.6 \times 10^8 \qquad \text{and} \qquad S_{LS}(A, b_2) = 7.9.$$

It is clear that these values are consistent with Figures 2 and 4, respectively, and thus the effective condition number is a measure, in the normwise sense, of the numerical condition of the linear algebraic equations that result from these problems in regression. □

The denominator of (7) suggests that the model

$$|c_i| = \sigma_i^{\beta}, \qquad \beta \geq 0, \tag{9}$$

for the absolute value of the components $c_i, i = 1, \ldots, n$, of c be postulated [4, 6]. This model is convenient because it follows from this equation that

$$S_{LS}(A, Uc) = \frac{\|c\|}{\sigma_n \sqrt{\sum_{i=1}^{n} \sigma_i^{2(\beta-1)}}},$$

and thus β controls the effective condition number of x_0. The approximate value of $S_{LS}(A, b)$ for three value of β is shown in Table 1, and it is seen that it increases as β increases. In particular, the LS problem is well-conditioned in the normwise sense if $0 \leq \beta \ll 1$, and it becomes ill-conditioned in this sense as β increases.[1] It follows, therefore, that the variation of β allows a wide class of LS problems, from those that are well-conditioned to those that are ill-conditioned, to be investigated. The form of the ratio $|c_i| / \sigma_i$ for different values of β is shown in Figure 5, and it is seen that its value determines the monotonic nature of the relationship.

The model (9) for $\beta > 1$ arises in the numerical solution of Fredholm integral equations of the first kind, which typically occur in inverse problems [5, 6]. This

[1] A distinction is made between componentwise and normwise stability because it is shown in Section 3 that x_0 may be well-conditioned in the normwise sense, but one or more of its components may be ill-conditioned.

Table 1. Approximate values of $S_{LS}(A,b)$ for different values of β

$\beta \geq 0$	$S_{LS}(A,b)$, $m \neq n$	$S_{LS}(A,b)$, $m = n$
$\ll 1$	$\frac{\|c\|}{\sigma_n^\beta} = \frac{\|c\|}{\|c_n\|}$	$\frac{\|c\|}{\|c_n\|}$
1	$\frac{\|c\|}{\sqrt{n}\sigma_n} = \frac{\|c\|}{\sqrt{n}\|c_n\|}$	$\frac{1}{\sqrt{n}}\kappa(A)$
$\gg 1$	$\frac{\kappa(A)\|c\|}{\sigma_1^\beta} = \frac{\kappa(A)\|c\|}{\|c_1\|}$	$\kappa(A)$

condition on β requires that the magnitude of the coefficients c_i decay to zero faster than the singular values σ_i [4, 6],

$$\frac{|c_i|}{\sigma_i} \to 0 \quad \text{as} \quad i \to n. \tag{10}$$

This requirement is called the discrete Picard condition, and it plays an essential role in the regularisation of ill-conditioned linear algebraic equations by Tikhonov regularisation in standard form and truncated singular value decomposition.

The restriction $\beta > 1$ is imposed in [4, 6] because it is assumed that the LS problem is derived from the discretisation of a Fredholm integral equation of the first kind. The condition $\beta \leq 1$ is, however, included in this paper because many problems that generate the LS problem, for example, regression and polynomial basis conversion [9, 10, 11], may or may not be ill-conditioned, and it is therefore necessary to consider (9) for all values of β.

It has been shown that the effective condition $S_{LS}\,(A,b)$ is superior to the condition number $\kappa_{LS}\,(A,\theta)$ because more accurate information on the true numerical condition of the LS problem can be determined. It would appear, therefore, that the disadvantage of $\kappa(A)$ and $\kappa_{LS}\,(A,\theta)$ has been overcome. The following example shows, however, that more refined condition numbers must be developed.

Example 3. Consider the matrix A and vector b that are given by [2],

$$A = \begin{bmatrix} 0.4919 & 0.1112 & -0.6234 & -0.6228 \\ -0.5050 & -0.6239 & 0.0589 & 0.0595 \\ 0.5728 & -0.0843 & 0.7480 & 0.7483 \\ -0.4181 & 0.7689 & 0.2200 & 0.2204 \end{bmatrix}, \; b = \begin{bmatrix} 0.4351 \\ -0.1929 \\ 0.6165 \\ -0.8022 \end{bmatrix}.$$

The solution x_0 of the equation $Ax = b$ was computed, and b was then perturbed to $b + \delta b$ where the elements of δb are drawn from a zero mean Gaussian

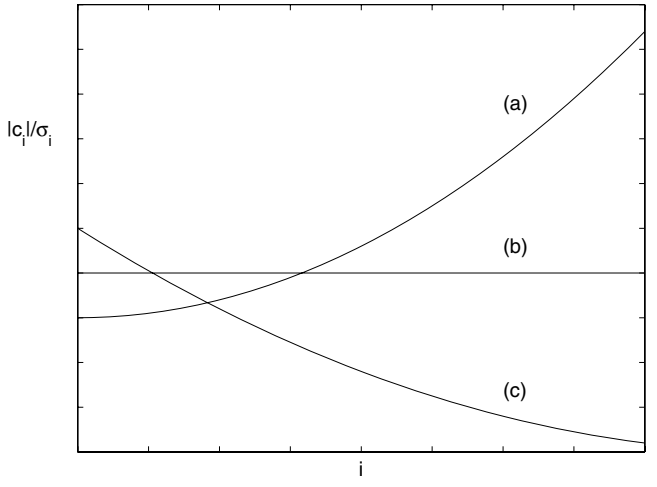


Fig. 5. Graphs of $|c_i| / \sigma_i$ against $i = 1, \ldots, n$, for (a) $\beta < 1$, (b) $\beta = 1$ and (c) $\beta > 1$

distribution such that $\|b\| / \|\delta b\| = 6.39 \times 10^3$. The solution $\hat{x}_0 = x_0 + \delta x_0$ of the equation $Ax = b + \delta b$ was computed. The two solutions are

$$x_0 = \begin{bmatrix} 1.000075414240576 \\ -0.5000879795933287 \\ -0.02425113887960606 \\ 0.02624513954981467 \end{bmatrix}, \qquad \hat{x}_0 = \begin{bmatrix} 1.000133044180548 \\ -0.4999705917859966 \\ 0.08574824910454026 \\ -0.08372428931414788 \end{bmatrix}.$$

The relative errors in the components $x_{0,k}, k = 1, \ldots, 4$, of x_0 are

$$\frac{|\delta x_{0,1}|}{|x_{0,1}|} = 5.76 \times 10^{-5}, \quad \frac{|\delta x_{0,2}|}{|x_{0,2}|} = 2.35 \times 10^{-4},$$
$$\frac{|\delta x_{0,3}|}{|x_{0,3}|} = 4.54, \quad \frac{|\delta x_{0,4}|}{|x_{0,4}|} = 4.19,$$

and the componentwise condition numbers are therefore, approximately,

$$\frac{1}{\Delta b} \begin{bmatrix} \frac{|\delta x_{0,1}|}{|x_{0,1}|} & \frac{|\delta x_{0,2}|}{|x_{0,2}|} & \frac{|\delta x_{0,3}|}{|x_{0,3}|} & \frac{|\delta x_{0,4}|}{|x_{0,4}|} \end{bmatrix}$$
$$= \begin{bmatrix} 0.37 & 1.51 & 2.91 \times 10^4 & 2.69 \times 10^4 \end{bmatrix},$$

where $\Delta b = \|\delta b\| / \|b\| = 1.56 \times 10^{-4}$. It is seen that $x_{0,1}$ and $x_{0,2}$ are well-conditioned, but $x_{0,3}$ and $x_{0,4}$ are ill-conditioned. The effective condition number of the equation is $S_{LS}(A, b) = 1.44 \times 10^3$ and the condition number of A is $\kappa(A) = 2.03 \times 10^3$. It is clear that the large value of $S_{LS}(A, b)$ is a reflection of the ill-conditioned nature of $x_{0,3}$ and $x_{0,4}$, but it fails to consider the stability of $x_{0,1}$ and $x_{0,2}$. □

This example shows that the effective condition number does not reveal how an error in b is distributed among the components of x_0. This information can be obtained by assigning a condition number to each component of x_0, and the next section considers these condition numbers.

2.3 Componentwise Condition Numbers

The condition numbers $\kappa_{LS}(A, \theta)$ and $S_{LS}(A, b)$ measure the errors in the normwise manner, but more refined estimates of the numerical condition of x_0 are obtained by assigning a condition number to each of its components. These condition numbers are introduced in [2] and expressions for them for the LS problem are derived. Since they consider the interaction of A and b, they differ from the collinearity measures that are introduced by Stewart [7], which only consider the linear dependence of the columns of A.

The componentwise condition number $R_{LS}(t_q, b)$ of the q'th component $x_{0,q}$ of x_0 is defined as the maximum value of the ratio of the relative error in $x_{0,q}$ to the relative error in b,

$$R_{LS}(t_q, b) := \max_{\delta b \in \mathbb{R}^m} \frac{\Delta x_{0,q}^{(c)}}{\Delta b}, \qquad \Delta x_{0,q}^{(c)} = \frac{|\delta x_{0,q}|}{|x_{0,q}|},$$

and it is easy to show that

$$R_{LS}(t_q, b) = \frac{\|t_q\| \, \|b\|}{|x_{0,q}|} = \frac{\|t_q\| \, \|b\|}{|t_q^T b|} = \frac{1}{|\cos \alpha_q|}, \qquad q = 1, \ldots, n, \tag{11}$$

where $t_q \in \mathbb{R}^m, q = 1, \ldots, n$, is the q'th row of $A^\dagger$, and α_q is the angle between t_q and b.

The n condition numbers (11) are very refined because they measure the relative error in each component of x_0. They discriminate between the well-conditioned, and ill-conditioned, components of x_0, and they therefore provide information that cannot be deduced from the effective condition number.

The next section considers mixed condition numbers, which, as their name implies, measure the errors in the componentwise and normwise senses. They may therefore be considered as intermediate between the effective condition number and the componentwise condition numbers.

2.4 Mixed Condition Numbers

The mixed condition numbers that are considered in this section use both componentwise and normwise measures, and they therefore display features of the condition numbers that are considered in Sections 2.2 and 2.3.

The mixed condition numbers of the solution of the LS problem are defined as

$$T_{LS}(t_q, b) := \max_{\delta b \in \mathbb{R}^m} \frac{\Delta x_{0,q}^{(m)}}{\Delta b}, \qquad \Delta x_{0,q}^{(m)} = \frac{|\delta x_{0,q}|}{\|x_0\|},$$

and it is easy to show that

$$T_{LS}(t_q, b) = \frac{\|t_q\| \, \|b\|}{\|x_0\|} = \frac{\|t_q\| \, \|b\|}{\|S^\dagger U^T b\|}, \qquad q = 1, \ldots, n. \tag{12}$$

The proof of (12) is very similar to that of the componentwise condition numbers (11).

Example 4. Figure 6 shows the effective, componentwise and mixed condition numbers of the solution x_0 of the LS problem in Example 1 for $b = b_1$. It is seen that the componentwise condition numbers display a large variation in magnitude, and that some of the values are smaller, and some are larger, than the effective condition number. The mixed condition numbers are smaller than their equivalent componentwise condition numbers, and they are also smaller than the effective condition number. □

Figure 6 illustrates the general phenomenon that different condition numbers may differ by several orders of magnitude, and the next section establishes inequalities between the condition numbers.

3 Inequalities Between the Condition Numbers

The condition numbers that are discussed in Section 2 quantify the numerical stability of x_0, and it therefore follows that there must be equations and/or inequalities that unite them. The relationship between the effective condition number and $\kappa_{LS}(A, \theta)$ is established in (5), and this section considers the relationships between these and the other condition numbers.

Consider initially the relationships between the componentwise and mixed condition numbers, and the effective condition number, for some, but not all, of the components of x_0. These are developed by noting that if $p \in \mathbb{R}^n$ is an arbitrary vector that satisfies $\|p\| = 1$, then the magnitude of at least one of its components must be greater than or equal to $1/\sqrt{n}$ [2].

Since t_q^T is the q'th row of $A^\dagger$, it follows that

$$t_q^T = e_q^T A^\dagger = e_q^T V S^\dagger U^T, \tag{13}$$

where e_q is the q'th standard basis vector, and thus

$$\|t_q\| = \left\|e_q^T V S^\dagger\right\| \geq \left|e_q^T V S^\dagger e_n\right| = \left|v_q^T S^\dagger e_n\right| = \frac{|v_{q,n}|}{\sigma_n},$$

where v_q^T is the q'th row of V and $v_{q,n}$ is element (q, n) of V. The result above shows that at least one of the elements of the n'th column of V must be greater than or equal to $1/\sqrt{n}$, and thus there exists at least one integer k, $1 \leq k \leq n$, such that

$$\|t_k\| \geq \frac{1}{\sqrt{n}\sigma_n}.$$

It therefore follows from (11) that there exists at least one integer k for which

$$R_{LS}(t_k, b) = \frac{\|t_k\| \, \|b\|}{|x_{0,k}|} \geq \frac{1}{\sqrt{n}\sigma_n} \frac{\|b\|}{|x_{0,k}|} = \frac{S_{LS}(A, b)}{\sqrt{n}} \frac{\|x_0\|}{|x_{0,k}|}. \tag{14}$$

This inequality is obtained on page 96 in [2], where it assumes a more complex form because it is stated in terms of the condition number $\kappa(A)$, rather than the effective condition number $S_{LS}(A, b)$. It shows that there are two contributory

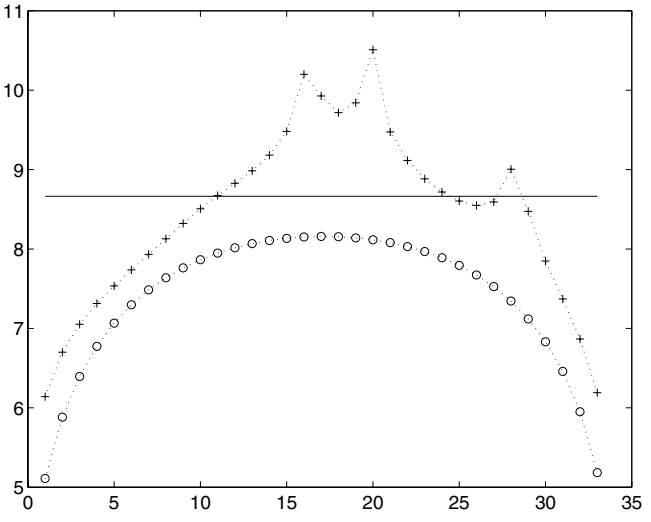


Fig. 6. The effective condition number (-), componentwise condition numbers (+) and mixed condition numbers (o) for the LS problem in Example 1 for $b = b_1$. A logarithmic scale is used for the condition numbers.

factors, a large value of the effective condition number and the condition $\|x_0\| \gg |x_{0,k}|$, to a large value of a componentwise condition number. Similarly, it follows from (12) that

$$T_{LS}\,(t_k, b) = \frac{\|t_k\|\,\|b\|}{\|x_0\|} \geq \frac{1}{\sqrt{n}\sigma_n}\frac{\|b\|}{\|x_0\|} = \frac{S_{LS}(A,b)}{\sqrt{n}}. \tag{15}$$

The inequalities (14) and (15) show that there exists at least one component of x_0 whose componentwise and mixed condition numbers, respectively, are large if the effective condition number is large. This may suggest that these condition numbers are consistent, but the following example shows that this desired consistency is not always present.

Example 5. Consider the equation

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} a \\ 0 \end{bmatrix}, \qquad a \neq 0,$$

for which

$$S_{LS}(A,b) = \kappa(A) = 1 \qquad \text{and} \qquad T_{LS}\,(t_1, b) = T_{LS}\,(t_2, b) = 1,$$

and

$$R_{LS}\,(t_1, b) = 1 \qquad \text{and} \qquad R_{LS}\,(t_2, b) = \infty.$$

It follows that the solution is well-conditioned in the normwise and mixed senses, but x_2 is very ill-conditioned in the componentwise sense. □

The inequalities (14) and (15) define lower bounds for at least one, but not all, of the componentwise and mixed condition numbers of x_0, respectively. Lower bounds of these condition numbers that are satisfied by all the components of x_0 are now developed.

It follows from (13) that

$$\|t_q\| \geq \frac{1}{\sigma_1}, \qquad q = 1, \ldots, n,$$

and thus since

$$R_{LS}(t_q, b) = \frac{\|t_q\| \, \|b\|}{|t_q^T b|} = \frac{\|t_q\| \, \|b\|}{|x_{0,q}|} \geq \frac{\|t_q\| \, \|b\|}{\|x_0\|} = T_{LS}(t_q, b),$$

it follows that

$$R_{LS}(t_q, b) \geq T_{LS}(t_q, b) = \frac{\|t_q\| \, \|b\|}{\|x_0\|} \geq \frac{1}{\sigma_1} \frac{\|b\|}{\|x_0\|}, \qquad q = 1, \ldots, n.$$

Equation (4) therefore yields

$$R_{LS}(t_q, b) \geq T_{LS}(t_q, b) \geq \frac{S_{LS}(A, b)}{\kappa(A)}, \qquad q = 1, \ldots, n, \tag{16}$$

which establishes the inequalities between the effective, componentwise and mixed condition numbers, and $\kappa(A)$.

Lower and upper bounds for the mixed condition numbers are easily established. In particular, it follows from (13) that

$$\|t_q\| \leq \frac{1}{\sigma_n}, \qquad q = 1, \ldots, n,$$

and thus the effective condition number is an upper bound for the mixed condition numbers,

$$T_{LS}(t_q, b) = \frac{\|t_q\| \, \|b\|}{\|x_0\|} \leq S_{LS}(A, b), \qquad q = 1, \ldots, n. \tag{17}$$

This equation can be combined with (5) and (16) to establish lower and upper bounds for the mixed condition numbers,

$$\frac{S_{LS}(A, b)}{\kappa(A)} \leq T_{LS}(t_q, b) \leq S_{LS}(A, b) \leq \kappa_{LS}(A, \theta), \qquad q = 1, \ldots, n, \tag{18}$$

and thus these condition numbers can vary significantly if $\kappa(A) \gg 1$. These inequalities contain condition numbers that are defined wholly or partly in the normwise sense, and the componentwise condition numbers are not present. It is clear that a finite upper bound for the componentwise condition numbers does not exist because one or more of them may be infinite, and thus only lower bounds, which are given in (14) and (16), can be established for them.

The inequality (15) can be combined with (18) to show that there exists at least one integer k such that

$$\frac{S_{LS}(A,b)}{\sqrt{n}} \leq T_{LS}\,(t_k,b) \leq S_{LS}(A,b),$$

which is a tight bound on the mixed condition number of $x_{0,k}$.

The inequalities that have been obtained thus far consider the mixed and componentwise condition numbers of individual components of x_0. It is now shown that they are constrained, such that the n components of each of these condition numbers are not independent.

Consider initially the mixed condition numbers. Since

$$\begin{aligned}\sum_{q=1}^{n} T_{LS}^2\,(t_q,b) &= \frac{\|b\|^2}{\|x_0\|^2}\sum_{q=1}^{n}\|t_q\|^2 \\ &= \frac{\|b\|^2}{\|x_0\|^2}\left\|A^{\dagger}\right\|_F^2 \\ &= \frac{\|b\|^2}{\|x_0\|^2}\sum_{k=1}^{n}\frac{1}{\sigma_k^2} \\ &= S_{LS}^2\,(A,b)\sum_{k=1}^{n}\left(\frac{\sigma_n}{\sigma_k}\right)^2,\end{aligned}$$

where $\|\cdot\|_F$ denotes the Frobenius norm, it follows that the sum of the squares of the mixed condition numbers are constrained. Furthermore, since

$$1 \leq \sum_{k=1}^{n}\left(\frac{\sigma_n}{\sigma_k}\right)^2 \leq n,$$

it follows that

$$S_{LS}(A,b) \leq \sqrt{\sum_{q=1}^{n} T_{LS}^2\,(t_q,b)} \leq \sqrt{n}\,S_{LS}(A,b),$$

which are tight bounds on the square root of the sum of the squares of the mixed condition numbers. It is readily verified that these bounds are consistent with (15) and (17), and they therefore confirm that if the effective condition number is large, there must be at least one component of x_0 whose mixed condition number is large.

This analysis can be repeated for the componentwise condition numbers,

$$\sqrt{\sum_{q=1}^{n} R_{LS}^2\,(t_q,b)} \geq S_{LS}(A,b).$$

4 The Stability of the Condition Numbers

The discussions in the previous sections have shown that the componentwise and mixed condition numbers provide the most refined information on the numerical condition of the solution of the LS problem, and that both of them are superior to the effective condition number, which is superior to $\kappa_{LS}(A, \theta)$. The progression

$$\kappa_{LS}(A, \theta) \rightarrow S_{LS}(A, b) \rightarrow T_{LS}(t_q, b) \rightarrow R_{LS}(t_q, b),$$

is associated with an increase in information, from the (in general) non-sharp upper bound $\kappa_{LS}(A, \theta)$ to the condition number $R_{LS}(t_q, b)$ for each component of x_0. It is natural, therefore, to enquire why the condition numbers $\kappa(A)$ and $\kappa_{LS}(A, \theta)$ are used extensively in the literature, but the condition numbers $S_{LS}(A, b), T_{LS}(t_q, b)$ and $R_{LS}(t_q, b)$ are quoted much less frequently. The reason for this omission is, at first thought, surprising because the derivation of these condition numbers is not difficult.

A condition number must be numerically stable for it to be of practical value. In particular, if ν is a generic condition number, then a change $\delta\nu$ in ν due to a change δb in b must satisfy

$$\frac{|\delta\nu|}{|\nu|} \approx \frac{\|\delta b\|}{\|b\|}. \tag{19}$$

The satisfaction, or otherwise, of this requirement is considered in Section 4.1 for the effective, mixed and componentwise condition numbers, and in Section 4.2 for $\kappa(A)$ and $\kappa_{LS}(A, \theta)$. It is shown that the effective, mixed and componentwise condition numbers may be ill-conditioned, but that $\kappa(A)$ and $\kappa_{LS}(A, \theta)$, assuming $\cos\theta \approx 1$, are well-conditioned.

4.1 The Effective, Mixed and Componentwise Condition Numbers

Equations (4) and (12) show, respectively, that the effective and mixed condition numbers are ill-conditioned (well-conditioned) when x_0 is ill-conditioned (well-conditioned). This qualitative observation can be made more precise by noting that the solution $x_0 + \delta x_0$ of the perturbed LS problem is given by

$$x_0 + \delta x_0 = \sum_{i=1}^{n} \frac{c_i + \delta c_i}{\sigma_i} v_i,$$

and determining the form of this function for different values of β.

Figure 7 shows the variation of $|c_i| / \sigma_i$ and $|c_i + \delta c_i| / \sigma_i$ for different values of β. It is adequate to consider only one of them in order to understand the shape of these graphs because the analysis for the other two graphs follows similarly. Consider Figure 7(iii), for which $\beta > 1$ and thus the discrete Picard condition (10) is satisfied. It follows that the magnitude of the coefficients c_i decays to zero faster than the singular values σ_i, and thus if it is assumed that $|\delta c_i| \approx \epsilon$

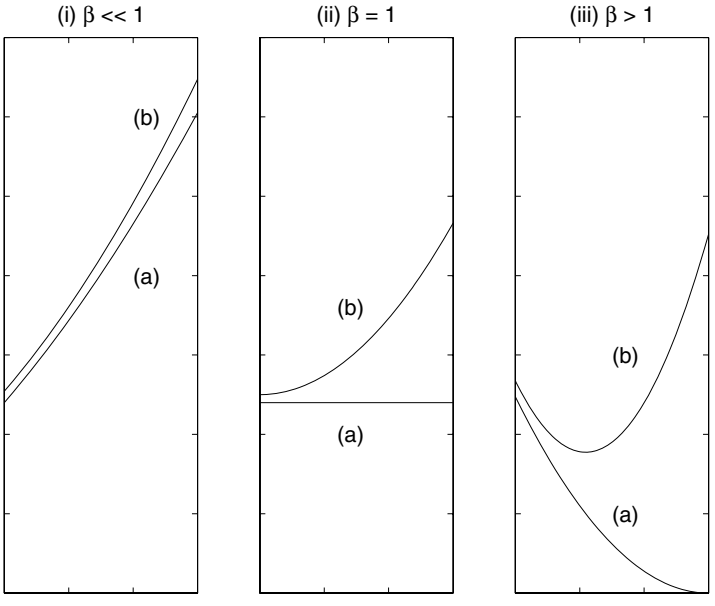


Fig. 7. The variation of (a) $|c_i|/\sigma_i$ and (b) $|c_i + \delta c_i|/\sigma_i$ for (i) $\beta \ll 1$, (ii) $\beta = 1$ and (iii) $\beta > 1$, where $|c_i| = \sigma_i^\beta$

where ϵ is a constant, there exists an integer p such that

$$\frac{|c_i + \delta c_i|}{\sigma_i} \approx \begin{cases} \frac{|c_i|}{\sigma_i} & \text{if} \quad i < p \\ \frac{|c_p + \delta c_p|}{\sigma_p} & \text{if} \quad i = p \\ \frac{\epsilon}{\sigma_i} & \text{if} \quad i > p. \end{cases} \tag{20}$$

Similar analysis for the other values of β shows that

$$\|x_0 + \delta x_0\| \approx \begin{cases} \frac{|c_n|}{\sigma_n} & \text{if} \quad 0 \le \beta \ll 1 \\ \frac{\epsilon}{\sigma_n} & \text{if} \quad 1 \le \beta. \end{cases} \tag{21}$$

It follows from (4) and (6) that

$$S_{LS}(A, U(c + \delta c) = \frac{1}{\sigma_n} \frac{\|c + \delta c\|}{\|x_0 + \delta x_0\|} = \frac{1}{\sigma_n} \frac{\sqrt{\|c\|^2 + 2\sum_{i=1}^m c_i \delta c_i + \|\delta c\|^2}}{\|x_0 + \delta x_0\|},$$

and since it is assumed that $\|\delta c\| \ll \|c\|$ and (from above) $|\delta c_i| \approx \epsilon$, it follows that

$$|\delta c_i| \approx \epsilon \ll \frac{\|c\|}{\sqrt{m}}.$$

Equation (21) therefore shows that

$$S_{LS}(A, U(c + \delta c)) \approx \begin{cases} \frac{\|c\|}{|c_n|} = S_{LS}(A, Uc) & \text{if} \quad 0 \le \beta \ll 1 \\ \frac{\|c\|}{\epsilon} & \text{if} \quad 1 \le \beta, \end{cases}$$

and thus as noted above, the effective condition number is ill-conditioned (well-conditioned) when x_0 is ill-conditioned (well-conditioned). It is clear that identical analysis can be performed for the mixed condition numbers.

The componentwise condition numbers are only considered qualitatively because, as noted in Section 3, x_0 may be well-conditioned in the normwise sense but one or more of its components may be ill-conditioned. Despite this qualification, it follows immediately from (11) that $R_{LS}(t_q, b)$ is ill-conditioned (well-conditioned) when $x_{0,q}$ is ill-conditioned (well-conditioned), and thus some of the qualitative features of the effective and mixed condition numbers are appropriate for the componentwise condition numbers.

The results in this section show that (19) is not satisfied by the effective, mixed and componentwise condition numbers. It follows that although they provide more detailed information than $\kappa(A)$ and $\kappa_{LS}(A, \theta)$, their computational implementation is problematic because incorrect estimates of the true numerical condition of x_0 may be obtained. It is therefore appropriate to consider the stability of $\kappa(A)$ and $\kappa_{LS}(A, \theta)$, and this is discussed in the next section.

4.2 The Condition Numbers $\kappa(A)$ and $\kappa_{LS}(A, \theta)$

The condition number $\kappa(A)$ is equal to σ_1/σ_n, and its stability is therefore defined by the stability with which the singular values of A can be computed. The Wielandt-Hoffman theorem shows that the singular values of A are well-conditioned with respect to perturbations in the elements of A [3]. This stability marks a distinction between this condition number, and the effective, componentwise and mixed condition numbers.

The numerical stability of $\kappa_{LS}(A, \theta)$ follows easily from that of $\kappa(A)$. In particular, a well-formulated problem is one in which the angle θ between b and Ax_0 is small, in which case $\kappa_{LS}(A, \theta) \approx \kappa(A)$. If, however, this angle is large, then it may indicate that the problem is badly posed. This situation may arise, for example, if inappropriate basis functions are used for regression.

This discussion shows that only $\kappa(A)$ and $\kappa_{LS}(A, \theta)$ can be computed reliably in the presence of errors, but they may overestimate, by several orders of magnitude, the true numerical condition of a linear algebraic equation. This suggests that there is a tradeoff between the detail of the information that is revealed by a condition number, and the reliability of its computational implementation. The practical use of the effective, mixed and componentwise condition numbers requires that numerically stable approximations to them be computed, and this topic is considered in the next section.

5 Numerically Stable Approximations to Condition Numbers

The development of regularised approximations to the effective, mixed and componentwise condition numbers requires that the cause of ill-conditioning be identified, and it is readily observed from Section 4.1 that this ill-conditioning is due to division by the small singular values in the term c_i/σ_i.

It follows from (4) and (12) that the effective and mixed condition numbers are ill-conditioned when x_0 is ill-conditioned. Chan and Foulser [1] develop a series of upper bounds for $\|x_0\|$ that avoid division by the small singular values. In particular, since

$$\begin{aligned}\sum_{i=1}^{n}\left(\frac{c_i}{\sigma_i}\right)^2 &\geq \sum_{i=n+1-k}^{n}\left(\frac{c_i}{\sigma_i}\right)^2 \\ &= \frac{1}{\sigma_{n+1-k}^2}\sum_{i=n+1-k}^{n}\left(\frac{\sigma_{n+1-k}}{\sigma_i}\right)^2 c_i^2 \\ &\geq \frac{1}{\sigma_{n+1-k}^2}\sum_{i=n+1-k}^{n} c_i^2, \qquad k=1,\ldots,n, \end{aligned} \tag{22}$$

it follows from (4) that

$$\begin{aligned} S_{LS}\left(A,Uc\right) &= \frac{1}{\sigma_n}\frac{\|c\|}{\|S^{\dagger}c\|} \\ &\leq \frac{\sigma_{n+1-k}}{\sigma_n}\left(\frac{\sum_{i=1}^{m} c_i^2}{\sum_{i=n+1-k}^{n} c_i^2}\right)^{\frac{1}{2}} \\ &:= \hat{S}_{LS}\left(A,Uc;k\right), \qquad k=1,\ldots,n. \end{aligned} \tag{23}$$

These inequalities define a sequence of upper bounds of the effective condition number, where

$$\hat{S}_{LS}\left(A,Uc;n\right) = \kappa_{LS}(A,\theta),$$

and the bound that is the best approximation to the effective condition number is given by the least upper bound of $\hat{S}_{LS}\left(A,Uc;k\right)$,

$$\hat{S}_{LS}(A,Uc) = \min_{k=1,\ldots,n} \hat{S}_{LS}\left(A,Uc;k\right). \tag{24}$$

It is clear that similar bounds can be developed for the mixed condition numbers.

These bounds appear to be numerically stable because division by the small singular values is not performed, which suggests that the disadvantage of the effective and mixed condition numbers is overcome. More careful analysis shows, however, that there exist situations in which these bounds are ill-conditioned. In particular, when b is perturbed to $b+\delta b$, (23) is perturbed to

$$\hat{S}_{LS}\left(A,U(c+\delta c);k\right) = \frac{\sigma_{n+1-k}}{\sigma_n}\left(\frac{\sum_{i=1}^{m}(c_i+\delta c_i)^2}{\sum_{i=n+1-k}^{n}(c_i+\delta c_i)^2}\right)^{\frac{1}{2}},$$

and thus for $k=1$ and $k=2$

$$\hat{S}_{LS}\left(A,U(c+\delta c);1\right) = \frac{\left(\sum_{i=1}^{m}(c_i+\delta c_i)^2\right)^{\frac{1}{2}}}{|c_n+\delta c_n|}, \tag{25}$$

and

$$\hat{S}_{LS}\left(A, U(c+\delta c); 2\right) = \frac{\sigma_{n-1}}{\sigma_n}\left(\frac{\sum_{i=1}^{m}(c_i+\delta c_i)^2}{\left(c_n+\delta c_n\right)^2+\left(c_{n-1}+\delta c_{n-1}\right)^2}\right)^{\frac{1}{2}}, \quad (26)$$

respectively. If the discrete Picard condition (10) is satisfied, (20) shows that there exists an integer p such that

$$|c_i + \delta c_i| \approx |\delta c_i| \quad \text{for} \quad i > p,$$

and thus the bounds in (25) and (26) for $k = 1$ and $k = 2$, respectively, and more generally those for values of k that satisfy $k \leq (n - p + 1)$, are ill-conditioned. It would appear, therefore, that the numerical problems of the effective and mixed condition numbers that were discussed in Section 4.1 persist. A simple solution to this problem can be achieved by applying a threshold η, such that values of $c_i + \delta c_i$ below η are set equal to zero,

$$c_i + \delta c_i = \begin{cases} 0 & \text{if} \quad |c_i + \delta c_i| \leq \eta \\ c_i + \delta c_i & \text{if} \quad |c_i + \delta c_i| > \eta, \end{cases} \quad (27)$$

but the success of this method is dependent upon the value of η. Specifically, a value that is too large will cause components of $c + \delta c$ that have relatively little noise to be filtered out, but a value that is too small will result in components of $c + \delta c$ that have a relatively large amount of noise to be retained.

Example 6. Consider the matrix A in Example 1 and the model (9). One hundred right hand side vectors were selected by allowing β to be a random variable that is uniformly distributed in the interval $[0, \ldots, 2]$. For each value of β, that is, for each right hand side vector, the least upper bound $\hat{S}_{LS}(A, Uc)$ was calculated using (24). The results are shown in Figure 8 and it is seen that $\hat{S}_{LS}(A, Uc)$ is an excellent approximation to the exact value of the effective condition number.

The experiment was repeated by perturbing, for each of the 100 experiments, b to $b + \delta b$ where the elements of δb are drawn from a zero mean Gaussian distribution, such that $\|b\| / \|\delta b\| = 10^5$. The results are shown in Figure 9 and it is seen that the noise degrades the upper bound approximation $\hat{S}_{LS}(A, Uc)$, such that it is now less than the exact value of the effective condition number in many of the experiments.

The simple thresholding strategy (27) was then applied with different values of η. In particular, if the exact value of the signal-to-noise ratio is τ, then the amplitude of the noise is

$$\|\delta b\| = \frac{\|b\|}{\tau} \approx \frac{\|b + \delta b\|}{\tau}, \qquad \|\delta b\| \ll \|b\|,$$

or equivalently

$$\|\delta c\| = \frac{\|c\|}{\tau} \approx \frac{\|c + \delta c\|}{\tau}, \qquad \|\delta c\| \ll \|c\|,$$

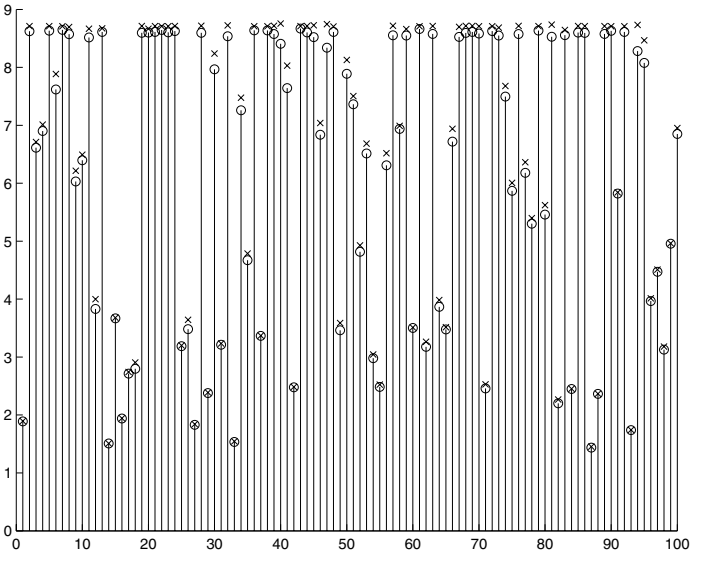


Fig. 8. The least upper bound (×) and the exact value (o) of the effective condition number for 100 right hand side vectors selected at random. A logarithmic scale is used for the condition numbers.

and if it is assumed that $|\delta c_i| \approx \epsilon$ where ϵ is constant, then

$$\epsilon \approx \frac{\|c + \delta c\|}{\sqrt{m}\tau}.$$

The thresholding strategy (27) becomes, therefore,

$$c_i + \delta c_i = \begin{cases} 0 & \text{if} \quad |c_i + \delta c_i| \leq \epsilon \\ c_i + \delta c_i & \text{if} \quad |c_i + \delta c_i| > \epsilon, \end{cases}$$

and thus the threshold ϵ is a function of τ, the exact value of the signal-to-noise ratio. Figures 10-13 show the effect of thresholding on the least upper bound $\hat{S}_{LS}(A, U(c + \delta c))$ for $\epsilon = 10^{-5}, 10^{-4}, 10^{-3}$ and 10^{-2}, respectively. Comparison of these four figures with Figure 9 shows that if $\epsilon \geq 10^{-4}$, the strategy (27) is effective in obtaining computationally reliable upper bounds of the effective condition numbers. Furthermore, although the value $\epsilon = 10^{-4}$ appears to be the optimum of the four values of ϵ that are shown, and the reciprocal of the exact value of the signal-to-noise ratio is $1/\tau = \|\delta b\| / \|b\| = 10^{-5}$, the application of the threshold $\epsilon = 10^{-3}$, that is, a threshold that is two orders of magnitude larger, yields acceptable estimates of the upper bound of the effective condition numbers. □

This example shows that if the signal-to-noise ratio is known or can be estimated, then a simple strategy can be implemented in order to obtain a computationally reliable upper bound for the effective condition number. It is clear that

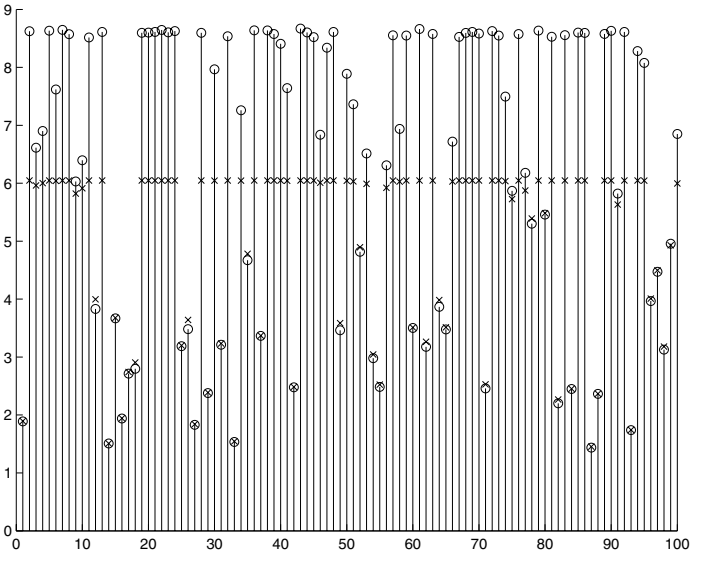


Fig. 9. The least upper bound (×) and the exact value (o) of the effective condition number for the 100 right hand side vectors in Figure 8, in the presence of noise. A logarithmic scale is used for the condition numbers.

the inequalities (22) can also be applied to the denominator of the expression of the mixed condition numbers, and thus computationally reliable upper bounds can also be obtained for these condition numbers. It may or may not be possible to develop upper bounds for the componentwise condition numbers, but it is noted that the inequalities (22) are not appropriate because they are valid for the 2-norm, and therefore not sufficiently refined for componentwise measures.

6 Summary

Four condition numbers of the LS problem have been considered theoretically and computationally, and it has been shown that the simplest measure $\kappa_{LS}(A, \theta)$ can be computed reliably (assuming $\cos\theta \approx 1$), but it may overestimate, possibly by several orders of magnitude, the exact numerical condition of the solution x_0 of the LS problem. This disadvantage, which arises because this condition number is defined as the maximum value of the error magnification, taken over all vectors b and δb, is overcome by the effective, mixed and componentwise condition numbers, because they provide more precise information on the exact numerical condition of x_0. Their computational implementation is, however, problematic because they are ill-conditioned in some circumstances, but it was shown that a simple procedure can be used to obtain computationally reliable upper bounds of the effective and mixed condition numbers.

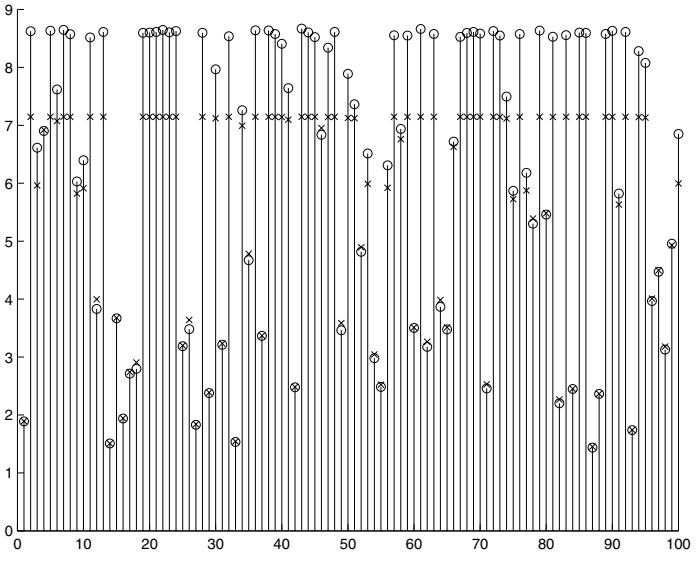


Fig. 10. The least upper bound (×) and the exact value (o) of the effective condition number, for $\epsilon = 10^{-5}$. A logarithmic scale is used for the condition numbers.

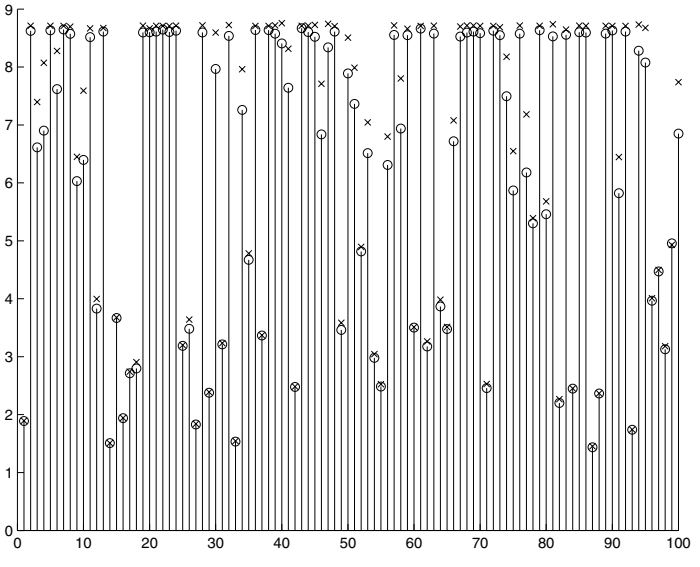


Fig. 11. The least upper bound (×) and the exact value (o) of the effective condition number, for $\epsilon = 10^{-4}$. A logarithmic scale is used for the condition numbers.

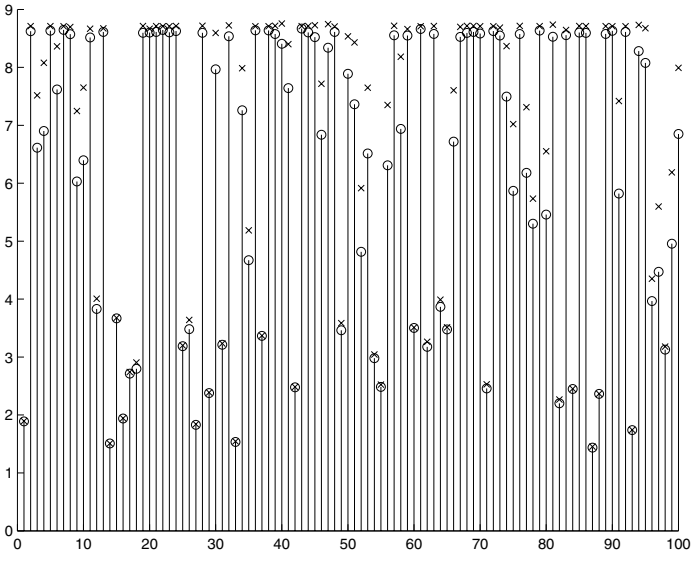


Fig. 12. The least upper bound (×) and the exact value (o) of the effective condition number, for $\epsilon = 10^{-3}$. A logarithmic scale is used for the condition numbers.

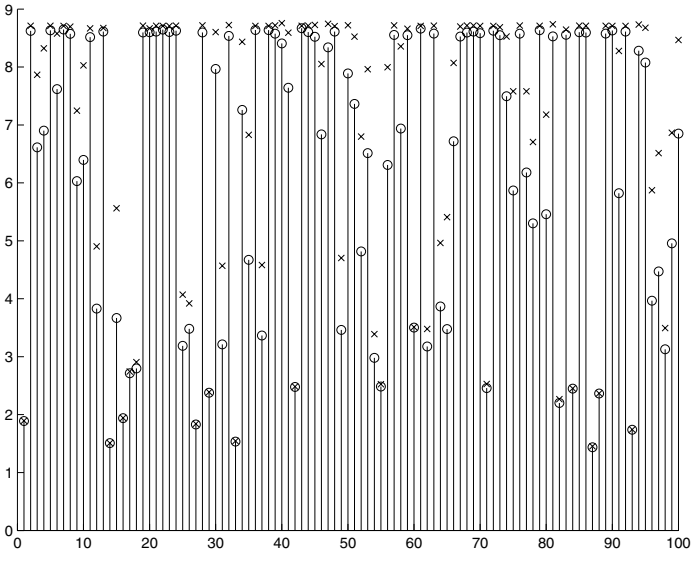


Fig. 13. The least upper bound (×) and the exact value (o) of the effective condition number, for $\epsilon = 10^{-2}$. A logarithmic scale is used for the condition numbers.

References

[1] T. F. Chan and D. E. Foulser. Effectively well-conditioned linear systems. *SIAM J. Sci. Stat. Comput.*, 9:963–969, 1988.

[2] S. Chandrasekaran and I. C. F. Ipsen. On the sensitivity of solution components in linear systems of equations. *SIAM J. Mat. Anal. Appl.*, 16(1):93–112, 1995.

[3] G. H. Golub and C. F. Van Loan. *Matrix Computations*. John Hopkins University Press, Baltimore, USA, 1989.

[4] P. C. Hansen. Truncated singular value decomposition solutions to discrete ill-posed problems with ill-determined numerical rank. *SIAM J. Sci. Stat. Comput.*, 11:503–518, 1990.

[5] P. C. Hansen. Numerical tools for analysis and solution of Fredholm integral equations of the first kind. *Inverse Problems*, 8:849–872, 1992.

[6] P. C. Hansen. *Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion*. SIAM, Philadelphia, USA, 1998.

[7] G. W. Stewart. Collinearity and least squares regression. *Statistical Science*, 2(1):68–100, 1987.

[8] D. S. Watkins. *Fundamentals of Matrix Computations*. John Wiley and Sons, New York, USA, 1991.

[9] J. R. Winkler. Numerically stable conversion between the Bézier and B-spline forms of a curve. In A. Le Méhauté, C. Rabut, and L. L. Schumaker, editors, *Curves and Surfaces with Applications in CAGD*, pages 465–472. Vanderbilt University Press, USA, 1997.

[10] J. R. Winkler. Polynomial basis conversion made stable by truncated singular value decomposition. *Applied Mathematical Modelling*, 21:557–568, 1997.

[11] J. R. Winkler. Tikhonov regularisation in standard form for polynomial basis conversion. *Applied Mathematical Modelling*, 21:651–662, 1997.

SVM Based Learning System for Information Extraction

Yaoyong Li, Kalina Bontcheva, and Hamish Cunningham

Department of Computer Science, The University of Sheffield, Sheffield, S1 4DP, UK
{yaoyong, kalina, hamish}@dcs.shef.ac.uk

Abstract. This paper presents an SVM-based learning system for information extraction (IE). One distinctive feature of our system is the use of a variant of the SVM, the SVM with uneven margins, which is particularly helpful for small training datasets. In addition, our approach needs fewer SVM classifiers to be trained than other recent SVM-based systems. The paper also compares our approach to several state-of-the-art systems (including rule learning and statistical learning algorithms) on three IE benchmark datasets: CoNLL-2003, CMU seminars, and the software jobs corpus. The experimental results show that our system outperforms a recent SVM-based system on CoNLL-2003, achieves the highest score on eight out of 17 categories on the jobs corpus, and is second best on the remaining nine.

1 Introduction

Information Extraction (IE) is the process of automatic extraction of information about pre-specified types of events, entities or relationships from text such as newswire articles or Web pages (see [10] for a comprehensive introduction to IE and its applications). A lot of research has focused on named entity recognition, a basic task of IE, which classifies proper nouns and/or numerical information into classes such as persons, organizations, and dates. Effectively, most IE tasks can be regarded as the task of recognizing some information entities within text. IE can be useful in many applications, such as business intelligence, automatic annotations of web pages with semantic information, and knowledge management.

Over the past ten years, a number of machine learning techniques have been used for IE and they have achieved state-of-the-art results, comparable to manually engineered IE systems. When applying machine learning to IE, a learning algorithm usually learns a model from a set of examples, grouped in documents, which have been manually annotated by the user. Then the model can be used to extract information from new documents. The accuracy of the learned model usually increases with the number of training examples made available to the system. However, as manual annotation is a time-consuming process, it is important for an IE system to have good performance on small training sets.

Machine learning algorithms for IE can be classified broadly into two main categories: rule-based or relational learning on one hand, and statistical learning,

J. Winkler, N. Lawrence, M. Niranjan (Eds.): Machine Learning Workshop, LNAI 3635, pp. 319–339, 2005.

on the other. For each entity class, rule-based methods induce a set of rules from a training set, while statistical methods learn statistical models or classifiers. Some systems based on rule or relational learning are SRV [16], RAPIER [2], WHISK [26], BWI [18], $(LP)^2$ [7], and SNoW [23]). Some example statistical systems are HMMs [14], Maximal Entropy [4], and SVM [19] or [22].

These IE systems also differ from each other in the features that they use. Some use only basic features such as token string, capitalization, and token type (word, number, etc.), e.g. BWI. In addition, others use linguistic features such as part-of-speech, semantic information from gazetteer lists, and the outputs of other IE systems (most frequently general purpose named entity recognizers). A few systems also exploit genre-specific information such as document structure, see e.g. [4]. In general, the more features the system used, the better performance it could achieve.

One of the most successful machine learning methods for IE is Support Vector Machine (SVM), which is a general supervised machine learning algorithm. It has achieved state-of-the-art performance on many classification tasks, including named entity recognition (see e.g. [19], [22]). For instance, [19] compares three commonly used methods for named entity recognition – SVM with quadratic kernel, maximal entropy, and a rule based learning system, and shows that the SVM-based system outperforms the other two. In our view, this comparison [19] is more informative than the comparison in, e.g., the CoNLL-2003 shared task (see [25]), because the former uses both the same corpus and the same features for all three systems, while in the later different systems used the same corpus but different features[1]. As already discussed above, more features usually result in better performance and therefore, it is important to use the same or similar features on the same dataset when comparing different algorithms.

This paper describes an SVM-based learning algorithm for IE and presents detailed experimental results. In contrast to similar previous work, our SVM model (see Section 2) uses an uneven margins parameter which has been shown [21] to improve the performance for document categorization (especially for small categories). Detailed experiments investigating different SVM parameters on three benchmark datasets were carried out (see Section 4.1). The experimental datasets were chosen to enable thorough comparison between our approach and other state-of-the-art learning algorithms (see Section 4.2). The learning curve of the algorithm was also evaluated by providing a small number of initial examples and then incrementally increasing the size of the training data (see Section 4.3). Section 5 covers related work.

2 The SVM Based IE System

Due to named entities often spanning more than one word, a classifier-based IE system needs to be designed to cope with this problem. In our SVM-based

[1] The Pascal Challenge in evaluation of machine learning methods for IE aims to provide a corpus and a pre-defined set of features, so different algorithms can be compared better (http://nlp.shef.ac.uk/pascal/).

system, called GATE-SVM, two SVM classifiers were trained for each entity type – one classifier to recognize the beginning of the entity and another one for the end. One word entities are regarded as both start and end. In contrast, [19] trained four SVM classifiers for each entity type – besides the two SVMs for start and end (like ours), also one for middle words, and one for single word entities. They also trained an extra SVM classifier to recognize words which do not belong to any named entity. Another approach is to train an SVM classifier for every possible transition of tags [22]. In this case, at least five classifiers need to be trained for every entity type: two classifiers for the two transitions between beginning and internal words, another two for the transitions between a beginning word and a non-entity word, and one for the transition from an internal word to a non-entity word. This approach also needs extra classifiers for the transitions between two entity types. Therefore, depending on the number of entities, this approach may result in a large number of SVM classifiers. Hence, in comparison, our approach is simpler than the other two SVM-based systems, in terms of requiring the lowest the number of SVM classifiers.

The rest of this section describes the other features of our system, namely the SVM algorithm, and the pre-processing and post-processing procedures.

2.1 The SVM with Uneven Margins

The GATE-SVM system uses a variant of the SVM, the SVM with uneven margins [21], which has a better generalization performance than the original SVM on imbalanced dataset where the positive examples are much less than the negative ones. The original SVM treats positive and negative examples equally such that the margin of the SVM hyperplane to negative training examples is equal to the margin to positive training examples. However, for imbalanced training data where the positive examples are so rare that they are not representative of the genuine distribution of positive examples, a larger positive margin than the negative one would be beneficial for the generalization of the SVM classifier (see detailed discussion in [21]). Therefore, [21] introduced an uneven margins parameter into the SVM algorithm. The uneven margins parameter is the ratio of the negative margin to the positive margin. By using this parameter the uneven margins SVM is able to handle imbalanced data better than the original even margins SVM model.

The uneven margins parameter has been shown previously to facilitate document classification on unbalanced training data (see [21]). Given that IE classification tasks, particularly when learning from small data sets, often involve imbalanced data (refer to Table 3 below), we expected to gain more benefits from SVM with uneven margins over the original SVM algorithm, which is confirmed in our experimental results presented in Section 4.

Formally, given a training set $\mathbf{Z} = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m))$,where $\mathbf{x}_i$ is the n-dimensional input vector and y_i $(= +1$ or $-1)$ its label, the SVM with uneven margins is obtained by solving the quadratic optimization problem:

$$\text{minimise}_{\mathbf{w},\ b,\ \xi}\ \langle \mathbf{w}, \mathbf{w} \rangle + C \sum_{i=1}^{m} \xi_i$$

$$\begin{aligned} \text{subject to} \quad & \langle \mathbf{w}, \mathbf{x}_i \rangle + \xi_i + b \geq 1 \quad \text{if } y_i = +1 \\ & \langle \mathbf{w}, \mathbf{x}_i \rangle - \xi_i + b \leq -\tau \quad \text{if } y_i = -1 \\ & \xi_i \geq 0 \qquad \text{for } i = 1, ..., m \end{aligned}$$

In these equations, τ is the uneven margins parameter which is the ratio of the negative margin to the positive margin in the classifier and is equal to 1 in the original SVM. Like other parameters of learning algorithms, the value of τ can be empirically determined by, for example, n-fold cross-validation on training set or hold-out development set. Moreover, from Table 5 in Section 4 we can see that the performance of the uneven margins SVM is not sensitive to the value of the uneven margins parameter. Therefore, a reasonable estimation of the τ can help the uneven margins SVM to achieve significantly better results than the original SVM model on imbalanced data.

The solution of the quadratic optimization problem above can be obtained by solving a related SVM problem (see [21]). In other words, it is not necessary to solve the uneven margins SVM problem directly. Instead, we can solve a corresponding standard SVM problem first by using an existing SVM implementation and then obtain the solution of uneven margins SVM through a transformation.

2.2 The Feature Vector Input to the SVM

When statistical learning methods are applied to IE tasks, they are typically formulated as classification, i.e., each word in the document is classified as belonging or not to one of the target classes (e.g., named entity tags). The same strategy is adopted in this work, which effectively means that each word is regarded as a separate instance by the SVM classifier. First of all, each document is processed using the open-source ANNIE system, which is part of GATE[2] [11]. This produces a number of linguistic (NLP) features. The features include token form, capitalization information of words, token kind, lemma, part-of-speech tag, semantic classes from gazetteer lists, and named entity type according to ANNIE's rule-based recognizer. Table 1 shows an example of text with its associated NLP features. Note that a token may not have all the NLP features considered, e.g. the token "Time" did not have the Lookup feature because it does not occur in ANNIE's gazetteer lists.

Given this input, a feature vector was derived from the NLP features of each token in the following way:

1. All possible features from the training documents are collected and indexed with a unique identifier, and each dimension of the feature vector corresponds to one feature (e.g. a given token string such as "Time" or a part-of-speech (POS) category such as "CD").
2. For each token, each component of the feature vector that corresponds to the value of the respective NLP feature are set to 1, and all other components are set to 0.

[2] Available from http://www.gate.ac.uk/

Table 1. NLP Features for the text sample "Time: 3:30 PM". The features are token form, capitalization information (Case), simple token kind (Tokenkind), part-of-speech (Pos), semantic classes from gazetteer lists (Lookup), and named entity types according the ANNIE. The Unknown type for word "Time" meant that ANNIE identified the word "Time" as a named entity but could not recognize its type of entity.

Token	Case	Tokenkind	Lemma	Pos	Lookup	Entity
Time	upperInitial	word	time	NNP		Unknown
:		punctuation	:	:		
3		number	3	CD		Time
:		punctuation	:	:		Time
30		number	3	CD		Time
PM	allCaps	word	pm	NNP	time	Time

Table 2 presents the feature vectors for the tokens listed in Table 1. We can see that the vectors are very sparse – only several components out of an approximately 20000 dimensional vector are non-zero.

Table 2. Feature vectors for the tokens of text "Time: 3:30 PM". The vectors are presented in a compact form, i.e. only the indexes and values of all nonzero components are shown. Refer to Table 1 for the NLP features.

Token	Feature Vector
Time	4835:1 11811:1 11815:1 19009:1 19697:1 19780:1
:	399:1 11816:1 12213:1 19682:1
3	187:1 11818:1 12001:1 19685:1 19778:1
:	399:1 11816:1 12213:1 19682:1 19778:1
30	188:1 11818:1 12002:1 19685:1 19778:1
PM	3621:1 11812:1 11815:1 17292:1 19697:1 19752:1 19778:1

Since in information extraction the context of the word is usually as important as the word itself, the SVM input vector needs to take into account features of the preceding and following words, in addition to those of the given word. In our experiments the same number of left and right words was taken as a context. In other words, the current word was at the centre of a window of words from which the features are extracted. This is called a *window size*. Therefore, for example, when the window size is 3, the algorithm uses features derived from 7 words: the 3 preceding, the current, and the 3 following words. Due to the use of a context window, the SVM input vector is the combination of the feature vector of the current word and those of its neighboring words.

As the input vector of the SVM combines the feature vectors of all words in the context window, these vectors can be weighted differently, depending on the relative importance of the neighboring words. In this work, two weighting schemes for the feature vectors from neighboring words were investigated. The first is the commonly used *equal weighting*, which keeps every nonzero component of the feature vector as 1 in the combined input vector, i.e., treats all

neighboring words as equally important. The second weighting scheme is the *reciprocal scheme*, which weights the surrounding words reciprocally to the distance to the word in the centre of the current window, reflecting the intuition that the nearer a neighboring word is, the more important it is for classifying the given word. Formally it means that the nonzero components of the feature vector corresponding to the jth right or left neighboring word are set to be equal to $1/j$ in the combined input vector. Therefore, we also refer to this scheme as $1/j$ **weighting**.

2.3 Post-processing the Results from the SVM Classifiers

As we train two different SVM classifiers to identity the start or end word for each target class, some post-processing is needed to combine these into a single tag. Therefore, our system has a module with *three different stages* to post-process the results from the SVM classifiers:

- The *first stage* uses a simple procedure to guarantee the consistency of the recognition results. It scans a document to remove start tags without matching end tags and end tags without preceding start tags.
- The *second stage* filters out candidate entities from the output of the first stage, based on their length. Namely, the tags of a candidate entity are removed if the entity's length (the number of words) is not equal to the length of any entity of the same type in the training set (a similar method was used in [18]).
- In contrast to the above two stages where each candidate entity is considered separately, the *third stage* puts together all possible tags for a given word and chooses the best one. In detail, the output x of the SVM classifier (before thresholding) is first transferred into a probability via the Sigmoid function $s(x) = 1/(1 + exp(-\beta x))$ where β is set to 2.0 (also see [19] and [22]). Then a probability for an entity candidate is computed as $s(x_s) * s(x_e)$, where x_s and x_e are the outputs of the SVM classifier for the start and end words of the candidate, respectively. Finally, for each given word, the probabilities for all possible tags are compared to each other and the tag with the highest probability P_h is assigned if P_h is greater than 0.25. Otherwise no tag is assigned to the word.

The three stages of the procedure can be applied sequentially to process the outputs of the classifiers. On the other hand, we can also obtain several post-processing procedures which consist of, e.g. only the first, or the first and the second, or all three stages. We will compare the different kinds of post procedures in Section 4.1. Note that both [19] and [22] used a Viterbi search algorithm as a post-procedure for their SVM classifiers, which corresponds to applying the first and third stages of our algorithm.

3 The Experimental Datasets

The system was evaluated on three corpora covering different IE tasks – named entity recognition (CoNLL-2003) and template filling or scenario templates [24]

(seminars and jobs corpora). There are several reasons for choosing these corpora. Firstly, CoNLL-2003 provides the most recent detailed evaluation results of machine learning algorithms on named entity recognition. Secondly, the seminars and jobs corpora have also been used recently by many learning systems, both wrapper induction and more linguistically oriented ones (see Section 5 for a detailed discussion). Thirdly, the CONLL-2003 corpus differs from the other two corpora in two important aspects: (i) in CONLL-2003 there are many entities per document, whereas the jobs and seminar corpora have only a small number per document; (ii) CONLL-2003 documents are mostly free text, whereas the other two corpora contain semi-structured documents. Therefore, the performance of our SVM algorithm was evaluated thoroughly on these three corpora as our goal was to design a versatile approach, with state-of-the-art performance both on domain-independent IE tasks (e.g., named entity recognition) and domains-specific ones (e.g., template filling).

In more detail, the first corpus is the English part of the CoNLL-2003 shared task dataset — language-independent named entity recognition[3]. This corpus consists of 946 documents for training, 216 documents for development (e.g., tuning the parameters in learning algorithm), and 231 documents for evaluation (i.e., testing), all of which are news articles taken from the Reuters English corpus (RCV1) [20]. The corpus contains four types of named entities — person, location, organization and miscellaneous names.

The other two corpora are the CMU seminar announcements and the software job postings[4], in both of which domain-specific information is extracted into a number of slots. The seminar corpus contains 485 seminar announcements and four slots – start time (stime), end time (etime), speaker and location of the seminar. The job corpus includes 300 computer related job advertisements and 17 slots encoding job details, such as title, salary, recruiter, computer language, application, and platform.

Table 3 shows the statistics for the CoNLL-2003, seminars and jobs datasets, respectively. As can be seen from that table, the non-annotated words are much more than the annotated words, particularly for domain-specific datasets like seminar announcements and software job postings. In other words, all three corpora are imbalanced datasets where the number of positive examples is much lower than the negative ones.

Machine learning systems typically separate the corpus into training and test sets. Since the CoNLL-2003 corpus already has the training, development and test set pre-specified, the system is trained on the training set, different experimental settings are tested on the development set, and the optimal ones are used in the run on the test set in order to obtain the performance results, which are then used for comparison to other systems.

The other two corpora do not provide such different sets, therefore training and testing need to be carried out differently. In our experiments we opted for splitting the corpora into two equal training and test sets by randomly assigning

[3] See http://cnts.uia.ac.be/conll2003/ner/

[4] Available from http://www.isi.edu/info-agents/RISE/repository.html.

Table 3. Number of examples for each entity/slot type, together with the number of non-tagged words, in CoNLL-2003 corpus, seminars announcements, and software jobs postings, respectively.

Conll03	LOC	MISC	ORG	PER	Non-entity	
Training set	7140	3438	6321	6600	191627	
Test-a set	1837	922	1341	1842	47926	
Test-b set	1668	702	1661	1617	43654	
Seminars	Stime	Etime	Speaker	Location	Non-entity	
	980	433	754	643	157647	
Jobs	Id	Title	Company	Salary	Recruiter	State
	304	457	298	141	312	462
	City	Country	Language	Platform	Application	Area
	659	345	851	709	590	1005
	Req-years-e	Des-years-e	Req-degree	Des-degree	Post date	Non-entity
	166	43	83	21	302	127302

documents to one or the other[5]. In order to obtain more representative results, we carried out several runs and the final results were obtained by averaging the results from each run. Many of the learning systems evaluated on these corpora used the same approach and we adopted it to facilitate comparison (see Section 4.2).

All corpora were also pre-processed with the open-source ANNIE system [11], in order to obtain the linguistic (NLP) features used in the SVM input vector, as discussed in Section 2.2 above. These features are used in addition to information already present in the documents such as words and capitalization information. The NLP features are domain-independent and include token kind (word, number, punctuation), lemma, part-of-speech (POS) tag, gazetteer class, and named entity type according to ANNIE's rule-based recognizer[6]. The following section discusses the experimental results on the three corpora.

4 Experimental Results

As already discussed in Section 2, two SVM classifiers are trained for each entity or slot filler, one for the start and one for the end words. The resulting models are then run on the test set and the post-processing procedures described in Section 2 are applied. The algorithm described in [21] is used to obtain the solution of the SVM with uneven margins by solving an SVM problem. More specifically, the SVM package SVMlight version 3.5[7] is used for solving the SVM problem. Unless otherwise stated, the default values of the parameters in SVMlight 3.5 are used.

[5] As the total number of documents in the seminar corpus is 485, we randomly split the dataset into 243 training documents and 242 testing ones.

[6] We also investigated the effect of the different NLP features, but due to space limitations the results will be included in another paper.

[7] Available from http://www.joachims.org/svm_light

The results below are reported using the F_1-measure, which is the harmonic mean of precision and recall. In other words, $F_1 = (2 * precision * recall)/(precision + recall)$, where *precision* is the percentage of correct entities found by the system and *recall* is the percentage of entities in the test set which are found by the system. A tag is considered correct if it matches exactly the human-annotated tag, both in terms of its type and its start and end offset in the document.

The overall performance of the algorithm on a given corpus can be obtained in two different ways. One is the so called *macro-averaged* F_1, which is the mean of F_1 of all the entity types or slots in the corpus. The other is the *micro-averaged* measure[8], obtained by adding together the recognition results on all entity types first and then computing precision, recall, and F-measure. Some researchers argue that the macro-averaged measure is better than the micro-averaged one (see e.g. [28]), because the micro-averaged measure can be dominated by the larger classes so that it reflects less the performance of the algorithm on smaller classes. On the other hand, if all classes are of a comparable size, as is often the case in IE datasets, then the macro-averaged measure is not very different from the micro-averaged one. Therefore, we use macro-averaged F-measure in Section 4.1 where an overall measure of the system's performance is needed, e.g., for the purpose of establishing the impact of different parameters on the system's performance. In Section 4.2 the macro-averaged F-measure is used for comparing the overall performance of our system with the seminars and jobs datasets, while the micro-averaged F-measure is used on the CoNLL-2003 dataset since it was used in the CoNLL-2003 shared task.

4.1 Influence of Different Parameters on the Algorithm's Performance

First of all, we carried out some preliminary experiments to determine the optimal parameter settings for each of three datasets. In order to avoid testing each possible setting against all others, the different settings of the parameters are investigated sequentially. The (possibly only slightly advantageous) best setting obtained for one parameter from the current experiment was used in subsequent ones.

The first group of experiments is for different sizes of the context window, while linear kernel and all NLP features are used. Then other parameters are investigated sequentially, namely the impact of SVM kernels (linear, quadratic and cubic), two values of the uneven margin parameter τ (1.0 and 0.4), different combinations of NLP features, two weighting schemes for the features from neighboring tokens, and finally three post-processing procedures derived from the three post-processing stages discussed in Section 2. In this way, the optimal settings listed in Table 4 were obtained (note the difference for the different corpora).

[8] See http: //www.itl.nist.gov/iaui/894.02/related_projects/muc/muc_sw/muc_sw_manual.html

Table 4. The optimal settings of system for the three datasets, obtained by the sequentially optimal experiments. For NLP features "all" means all the NLP features obtained from the ANNIE system. For post-processing "all" means that all the three stages of the procedure are used.

Setting	window size	SVM kernel	τ	NLP features	Weighting	Post-processing
Conll03	2	quadratic	0.4	all except POS	$1/j$	all
Seminars	5	quadratic	0.4	all	$1/j$	all
Jobs	3	linear	0.4	all except POS	$1/j$	all

It should be noted that while keeping the number of experiments down, such sequential optimization may not result in the most optimal parameter settings. For instance, the optimal window size from the first group of experiments using linear kernel may not be optimal for the later experiments using quadratic kernel. Hence, the optimal setting obtained at each stage may not be the global optimal value, although we believe the differences to be quite small.

Next, a series of experiments was conducted to investigate the influence of the different parameters. In these experiments, we used different settings of one parameter and adopted the values of all other parameters, as presented in Table 4 for each dataset. Due to space limitations, this paper focuses on the experimental results for the unique features in our system, namely the uneven margins parameter τ, the reciprocal weighting scheme, and the post-processing procedures.

As already discussed above, on the CoNLL-2003 dataset, the system is trained on the train set and the results are reported on the development set. Each of other two datasets is split randomly in two, with one partition used for training and the other for testing, and the results are averaged over ten runs.

Uneven Margins Parameter. Our system uses the uneven margins SVM model, while other SVM-based systems for IE use the original SVM algorithm with even margins (see e.g. [19] and [22]). As discussed in Section 2, the uneven margins parameter τ is the ratio of the negative margin to the positive margin. If $\tau = 1$, the uneven margins SVM is equivalent to the original SVM model. As already discussed, the uneven margins parameter helps the SVM handle imbalanced training sets, i.e., sets where the positive training examples are much rarer compared to the negatives ones (a common problem in classification for IE).

Table 5 presents the results for different values of uneven margins parameter for the three datasets. Firstly, the SVM with uneven margins ($\tau < 1.0$) performs statistically significantly better than the original SVM ($\tau = 1.0$) on the two datasets – Seminars and Jobs. On the other hand, the uneven margins model obtains only marginal improvements over the even margins model on the CoNLL-2003 data. This is because the classification problems on the first two corpora are much more imbalanced than those on the CoNLL-2003 dataset. The more imbalanced a classification problem is, the more helpful the uneven margins parameter is. Also see the results for small training sets in Section 4.3. Secondly, it can be seen that the results for the τ in an interval (e.g. the interval (0.4, 0.6))

Table 5. Results for different settings of uneven margins parameter of the SVM: macro-averaged F_1 (%) on the three datasets. The standard deviation is shown in parenthesis, indicating the statistical significances of the results. The best performance figures on each dataset appear in bold.

τ	1.0	0.8	0.6	0.4	0.2	0.0
Conll03	89.0	89.6	**89.7**	89.2	85.3	65.6
Seminars	81.7(±0.6)	84.0(±0.7)	85.8(±0.8)	**86.2(±0.8)**	82.6(±1.0)	55.4(±1.4)
Jobs	79.0(±1.4)	79.9(±1.2)	**81.0(±0.9)**	80.8(±1.0)	79.0(±1.3)	57.7(±1.5)

Table 6. Two weighting schemes: macro-averaged F_1 (%) on the three datasets. In bold are the best performance figures for every dataset.

	Equal weighting	$1/j$ weighting
Conll03	88.4	**89.2**
Seminars	85.5(±1.0)	**86.2(±0.8)**
Jobs	80.5(±1.0)	**80.8(±1.0)**

are quite similar, showing that the performance is not particularly sensitive to the value of τ. Finally, $\tau = 0.6$ achieves slightly better results than $\tau = 0.4$ on the Jobs data. However, due to the small difference, all other experiments presented in this paper preserve the experimental settings from Table 4, meaning that $\tau = 0.4$ is used on both the Seminar and Jobs datasets.

Two Weighting Schemes. We compared two weighting schemes for combining the features from surrounding words – the commonly used *equal weighting* and the *reciprocal weighting* of features from neighboring words (discussed in Section 2.2 above). Table 6 presents the results of the two weighting schemes on the three datasets. While the reciprocal scheme produces slightly better results than equal weighting on all the three datasets, the difference is not statistically significant.

Post-processing Procedures. As discussed in Section 2.3, a three-stage post-processing procedure is used to combine the results of the SVM classifiers. In brief, in the first stage filters out the spurious start or end tags. The second removes entities with length not equal to the length of any example tag in the training set. The third stage outputs the category with the highest probability. Based on these three different filtering strategies, three post-processing procedures were experimented with: the first strategy only; the first and second strategies; and finally all three in sequence. In the first and second procedures, if one piece of text is assigned more than one tag, then the last tag according to the tag order in Table 3 is assigned.

Table 7 presents the results before post-processing as well as the results for the three procedures on the three corpora. Compared to the results with no post-processing, the results are improved significantly by post-processing. However, procedures 2 and 3 only obtain slightly better results than their previous stages.

Table 7. Comparison of the results without and with the post-processing procedures: macro-averaged F_1 (%) on the three datasets. Proc1, Proc2 and Proc3 denote respectively the three post-processing stages. In bold are the best performance figures for every dataset.

	No post-processing	Proc1	Proc2	Proc3
Conll03	87.5	89.0	89.1	**89.2**
Seminars	81.7(±1.0)	85.5(±0.9)	85.7(±0.9)	**86.2(±0.8)**
Jobs	77.0(±0.7)	79.9(±0.9)	80.3(±0.9)	**80.8(±1.0)**

4.2 Comparison to Other Systems

This section compares our system to other machine learning approaches on the three datasets. Since our system uses the NLP features produced by GATE and the learning algorithm based on SVM, we call our system **GATE-SVM**. In the experiments described in this subsection, we used similar settings to those in the other systems, in order to enable a fair comparison.

The significance boundaries of the results on the CoNLL-2003 corpus, presented in this paper, are estimated via bootstrap sampling method [25] and can be used to determine if a result is significantly different than all others.

For the other two datasets, the significance boundaries of previous results are not available. Since we ran ten experiments on each of the two datasets, the standard deviations from the results of the ten experiments are used as the significance measure in the comparison to other systems.

Named Entity Recognition. The performance of GATE-SVM on named entity recognition is evaluated on the CoNLL-2003 dataset. Since this set comes with development data for tuning the learning algorithm, different settings were tried in order to obtain the best performance on the development set. The different SVM kernel types, window sizes, and values of the uneven margin parameter τ were tested. The results showed that quadratic kernel, window size 4 and $\tau = 0.5$ produce best results on the development set. These optimal values are slightly different from those obtained in Section 4.1, which shows that the values for learning parameters selected through sequential optimization may not be globally optimal. The $1/j$ weighting scheme and all three post-processing stages are used.

Table 8 presents the results of our system on the CoNLL-2003 dataset, together with the results of the top system in the CoNLL-2003 share task evaluation [13] and another participating SVM-based system [22], which are taken from the summary paper [25]. The results of our systems are given using two different settings in order to make a fairer comparison to the SVM based system entered in the shared task. GATE-SVM-1 uses all NLP features obtained from ANNIE, except part-of-speech information. The only difference between GATE-SVM-1 and GATE-SVM-2 is that GATE-SVM-2 does not use the semantic information from ANNIE's gazetteer lists. Therefore, all types of NLP features used by GATE-SVM-2 were also used by the participating SVM-based system. The

Table 8. Comparison to other systems on the CoNLL-2003 shared task: F-measure (%) on each entity type and the overall micro-averaged F-measures. The macro-averaged F-measure (MA F_1) is also included for comparison. Test-a denotes the development set and test-b is as the test set. The only difference between GATE-SVM-1 and GATE-SVM-2 is in the NLP feature used (see text). The best performance figures for each entity type and overall appear in bold.

System	test set	LOC	MISC	ORG	PER	MA F_1	Overall
GATE-SVM-1	test-a	93.70	86.13	87.00	93.03	89.96	90.83
	test-b	89.25	77.79	82.29	90.92	85.06	86.30
GATE-SVM-2	test-a	93.46	86.36	86.76	92.24	89.70	90.49
	test-b	89.20	77.66	81.60	90.68	84.78	86.00
Best one	test-a	**96.12**	**89.06**	**90.24**	**96.60**	**93.01**	**93.87**
	test-b	**91.15**	**80.44**	**84.67**	**93.85**	**87.53**	**88.76±0.7**
Participating	test-a	93.75	86.02	85.90	93.91	89.90	90.85
SVM Based System	test-b	88.77	74.19	79.00	90.67	83.16	84.67±1.0

results of both GATE-SVM-1 and GATE-SVM-2 are significantly better the participating SVM-based system. However, our results are significantly worse than the best result, which was obtained by combining the outputs of four different classifiers and other information.

Template Filling. The seminar corpus has been used to evaluate quite a few learning systems. Those include rule learning approaches such as SRV [17], Whisk [26], Rapier [2], BWI [18], SNoW [23] and $(LP)^2$ [7], as well as statistical learning systems such as HMM [14] and maximum entropy (MaxEnt) [4]. See Section 5 for more details.

The major problem with carrying out comparisons on the seminar corpus is that the different systems used different experimental setups. For instance, SRV, SNoW and MaxEnt reported results averaged over 5 runs. In each run the dataset was randomly divided into two partitions of equal size – one used for training and one for testing. Furthermore, SRV used a randomly selected third of the training set for validation. WHISK's results were from 10-fold cross validation on a randomly selected set of 100 documents. Rapier's and $(LP)^2$'s results were averaged over 10 runs, instead of the 5 runs used in SRV, SNoW and MaxEnt. Finally, BWI's and HMM results were obtained via standard cross validation.

The GATE-SVM results reported here are the average over ten runs, following the methodology of Rapier and $(LP)^2$. Table 9 presents the results of our system on seminar announcements, together with the results of the other systems. As far as it was possible, we use the same features as the other systems to enable a more informative comparison. In particular, the results listed in Table 9, including our system, did not use any gazetteer information and named entity recognizer output. The only features in this case are words, capitalization information, token types, lemmas, and POS tags. The settings for the SVM parameters were taken from Table 4, i.e., window size 5, quadratic kernel, and $\tau = 0.4$.

The F_1 measure for each slot was computed together with the macro-averaged F_1 for the overall performance of the system. Note that the majority of systems evaluated on the seminars and jobs corpora only reported per slot F-measures, without overall results. However, an overall measure is useful when comparing different systems on the same dataset. Hence, we computed the macro-averaged F_1 for the other systems from their per-slot F_1.

Table 9. Comparison to other systems on CMU seminar corpus: F_1 (%) on each slot and overall performance (macro-averaged F_1). Standard deviation for the MA F_1 of our system is presented in parenthesis. The best results for each slot and the overall performance appear in bold font.

	Speaker	Location	Stime	Etime	MA F_1
GATE-SVM	69.0	81.3	94.8	92.7	84.5($\pm$0.8)
$(LP)^2$	**77.6**	75.0	99.0	95.5	**86.8**
SNoW	73.8	75.2	**99.6**	**96.3**	86.2
MaxEnt	65.3	82.3	**99.6**	94.5	85.4
BWI	67.7	76.7	**99.6**	93.9	84.6
HMM	71.1	**83.9**	99.1	59.5	78.4
Rapier	53.1	73.4	95.9	94.6	79.1
Whisk	18.3	66.6	92.6	86.1	65.7
SRV	56.3	72.2	98.5	77.9	76.0

Table 9 shows that the best results on the different slots are achieved by different system and that the best overall performance is achieved by the $(LP)^2$. The GATE-SVM did not perform as well as the best results on the Seminar data. But it still outperformed many other systems.

However, if information from the ANNIE gazetteer and named entity recognizer is used as additional features, then the macro-averaged F_1 for GATE-SVM is 0.862, which is better than the 0.857 for $(LP)^2$ using the same features (see [7]). While this is still slightly worse than the 0.872 of the maximum entropy system (see [4]), a direct comparison between the GATE-SVM and that system cannot be made. This is due to our system just using general NLP features while [4] used genre-specific features (see Section 5 for further details). Furthermore, GATE-SVM's parameter settings were not optimized specifically for this corpus (see the discussions about optimizing the experimental settings for the CoNLL-2003 dataset above).

On the **jobs postings corpus**, GATE-SVM is compared to two rule learning systems, Rapier [2] and $(LP)^2$ [7], which are among the few evaluated on this dataset.

Again, in order to make the comparison as informative as possible, we adopted the same settings in our experiments as those used by the system which reported the highest results on this dataset, i.e., $(LP)^2$ [8]. In particular, the results are obtained by averaging the performance in ten runs, using a random half of the corpus for training and the rest for testing. In contrast, Rapier's results were obtained via 10-fold cross validation over the entire dataset, thus making

Table 10. Comparison to other systems on the jobs corpus: F_1 (%) on each entity type and overall performance as macro-averaged F_1. Standard deviation for the MA F_1 of our system is presented in parenthesis. The highest score on each slot and overall performance appears in bold.

Slot	**GATE-SVM**	$(LP)^2$	Rapier	Slot	**GATE-SVM**	$(LP)^2$	Rapier
Id	97.7	**100**	97.5	Platform	80.1	**80.5**	72.5
Title	**49.6**	43.9	40.5	Application	70.2	**78.4**	69.3
Company	**77.2**	71.9	70.0	Area	46.8	**53.7**	42.4
Salary	**86.5**	62.8	67.4	Req-years-e	**80.8**	68.8	67.2
Recruiter	78.4	**80.6**	68.4	Des-years-e	81.9	60.4	**87.5**
State	**92.8**	84.7	90.2	Req-degree	**87.5**	84.7	81.5
City	**95.5**	93.0	90.4	Des-degree	59.2	65.1	**72.2**
Country	**96.2**	81.0	93.2	Post date	99.2	**99.5**	**99.5**
Language	86.9	**91.0**	81.8	MA F_1	**80.8(±1.0)**	77.2	76.0

it impossible to adopt a unified approach. As in the previous experiment, only basic NLP features are used: word, capitalization information, token types, and lemmas. The parameter values of window size 3, linear kernel and $\tau = 0.4$) are used here. The macro-averaged F_1 for the other two systems is computed for overall performance comparison.

Table 10 presents the results of our system as well as the results of the other two systems on the Jobs corpus. GATE-SVM achieves the best results among all three on eight out of the 17 slots and the second best results on the remaining nine. Overall, the macro-averaged F_1 of GATE-SVM is significantly better than the other two systems.

4.3 Information Extraction from Small Training Sets

The application of SVM (or other supervised learning algorithms) to IE requires a manually annotated training set. Since manual annotation is a time-consuming process. learning from small data sets is highly desirable.

Consequently, we evaluated the learning algorithm on a growing number of examples. For both the seminar and jobs corpora, a small number of documents from the corpus were selected randomly as the training set and the remaining ones were used for testing. For the CoNLL-2003 dataset the training documents were chosen randomly from the training set and the results are reported on the development set. In order to factor out randomness of results, the mean of ten runs is reported. The same features and system parameters were used on each of the three datasets as those in Section 4.2.

Figure 1 shows the learning curves of the SVM models with and without uneven margins on the three datasets. The 95% confidence intervals for the data points are also shown for the Seminars and Jobs datasets. System performance improves consistently as more training documents become available. In addition, the uneven margins SVM model demonstrates clearly better results than the original SVM, in particular on a small number of training documents.

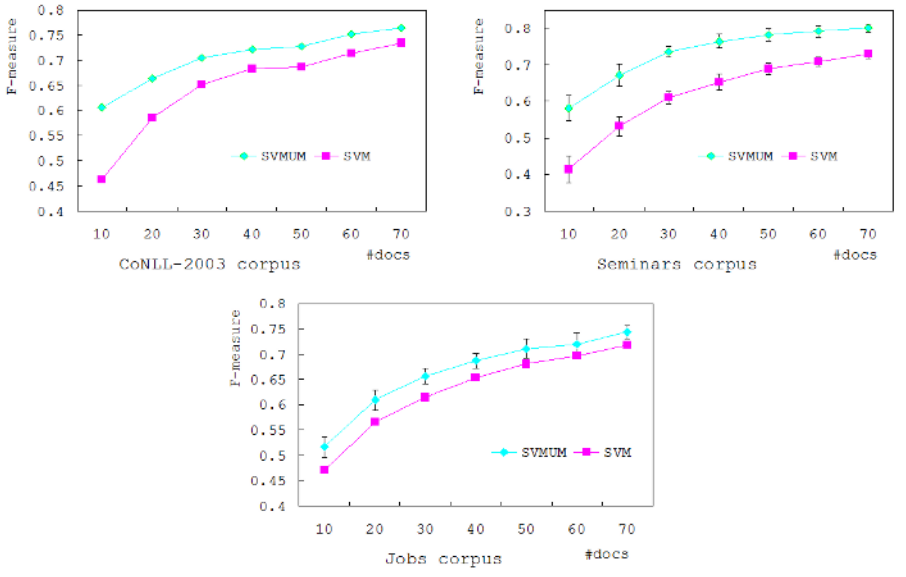


Fig. 1. Learning curves for overall F1 with a growing number of training documents on the three datasets. On each dataset the uneven margins SVM is compared to the original SVM model. For the Seminar and Jobs datasets, the error bar at a data point show the 95% confidence interval derived from the standard deviations. For clarity, we just show the confidence intervals for one curve in the Jobs graph – the confidence intervals are similar for another curve.

Table 11. Different numbers of documents for training: macro-averaged F_1 (%) on seminars dataset for every entity type

	10	20	30	40	50	60	70
stime	82.4	86.6	88.9	90.5	91.4	92.0	92.4
etime	70.7	80.6	85.9	88.1	88.5	89.2	90.6
speaker	30.7	42.4	55.6	60.6	63.4	65.5	65.9
location	48.4	58.6	63.7	67.0	69.6	69.9	70.9

Table 11 shows that some types of entities can be learned faster than others, due to their more fixed internal structure. For example, start and end times can be learned from as little as 10 documents, while at least 60 documents are required to reach similar performance on speaker and location. When interpreting these results one must bear in mind that most documents in the seminars dataset provide only one, or maximum two, examples of each slot (the ratio between number of documents and number of examples per slot in the corpus ranges between 0.9 and 2). Therefore, in this case learning after 10 documents is almost equivalent to learning from 10 to 15 examples per slot.

Another system which carried out such experiments on the seminars dataset is $(LP)^2$ [9]. Table 12 compares our system with the system based on $(LP)^2$. In a nutshell, our system is better than $(LP)^2$ on the stime and speaker categories

Table 12. Comparison of GATE-SVM with $(LP)^2$: F_1 (%) for each slot of the seminars corpus and the macro averaged F_1 for overall performance. The highest score on each slot and overall performance appears in bold.

	Number of training docs	$(LP)^2$	GATE-SVM
stime	30	84.0	**88.9**
etime	20	**82.3**	80.6
speaker	25	50.6	**60.6**
location	30	**70.0**	63.7
MA F_1		71.7	**73.5**

but is worse on etime and location. The overall performance of our system is slightly better than that of $(LP)^2$.

5 Related Work

This section briefly discusses previous work on applying machine learning to IE, in particular those systems which were evaluated on the three datasets used in our experiments. We first describe the applications of SVM to IE. Then we look at the other algorithms evaluated on the CoNLL-2003 dataset. Finally, the rule learning and statistical learning IE systems on the seminar announcements and job postings corpora are reviewed.

5.1 SVM-Based Systems

The SVM based system in [19] trained four SVM classifiers for each named entity type – besides the two SVMs for start and end words like ours, one for middle words, and one for single word entities. They also trained an extra SVM classifier to recognize the words which do not belong to any named entity. [19] used a sigmoid function to transfer the SVM output into a probability and then applied the Viterbi algorithm to determine the optimal label sequence for a sentence. The system was evaluated on a Japanese IE corpus. They used the neighboring words with window size 2. Their experiments showed that the SVM based system performed better than both maximum entropy and rule learning systems on the same dataset using the same features. They also showed that quadratic kernel was better than both linear and cubic kernels on their dataset. [19] also described an efficient implementation of the SVM with quadratic kernel.

[22] used a lattice-based approach to named entity recognition and employed SVM with cubic kernel to compute transition probabilities in a lattice. They trained an SVM classifier for every possible transition of tags, meaning that they may have a large number of SVM classifiers. They tested the system on the CoNLL-2003 dataset using cubic kernel. They also took into account the features from neighboring words with window size 3. Their result on the CoNLL-2003 corpus is comparable to ours (see Table 8). There are some other applications of SVM for bio-named entity recognition (see e.g. [27]).

5.2 Other Learning Methods Evaluated on the CONLL'2003 Corpus

CoNLL-2003 corpus is a typical named entity recognition corpus with newswire articles and entity types similar to the earlier MUC-6 and MUC-7 corpora [24]. Sixteen systems participated in the evaluation. All of them were based on statistical learning, except one system which used rule learning as one of four algorithms which were combined into one classifier. The system with the best score was exactly this combined system, based on robust risk minimization, maximum entropy, transformation based learning and HMMs, respectively (see [13].

Another system only based on maximum entropy obtained slightly worse results (see [5]). They used many different features, including some genre-specific one, such as the so-called zone related features which are dependent on the structure of documents. Note that another two participating systems were also only based on maximum entropy (see [1], [12]). In particular, the probability discriminate model used in [12] was quite similar to the one in [5]. The features used in [12] were general and less than those in [5]. Both the scores of these two systems ([1] and [12]) were significantly worse than the system described in [5], which confirms the conjecture that appropriate features are at least as important as the learning algorithm.

The SVM based participating system was discussed above (see [22]).

5.3 Learning Systems Evaluated on Template Filling

SRV is a relational learning (or inductive logic programming) algorithm for IE, which deduces a set of rules for one type of information entity from training examples (see [15]). It checked every text fragments of appropriate size in document in order to identify if the fragment belongs to an entity or not. [16,17] tested SRV for IE on three datasets – the CMU seminars corpus, a collection of 600 newswire articles on corporate acquisitions from Reuters and a collection of web pages of university computer science departments.

WHISK [26], another relational learning system for IE, was tested on collections of structured, semi-structured and free-text documents, such as CNN weather domain, seminar announcements, software jobs postings, and news story articles. WHISK's results on the seminars corpus were not as good as SRV's, which may be attributed to the fact that WHISK used less features – only the token and its semantic class.

Rapier is also a rule based learning IE system (see [2]). It was tested on two dataset: software jobs and seminar announcements. Its results on seminar announcements are better than SRV.

BWI (Boosted Wrapper Induction) involved learning a wrapper (boundary detector) for an information entity via a boosting procedure (see [18]). It was evaluated on several collections such as seminar announcements, software job postings, Reuters articles, and web pages. [18] also considered the neighboring words as context and found, similar to us, that different datasets have a different optimal window size. One should note that for rule based learning algorithms the training time increases exponentially with window size.

$(LP)^2$ is also a rule learning algorithm for IE (see e.g. [7]). $(LP)^2$ was tested on three datasets: seminar announcements, software job postings, and a collection of 103 web pages describing computer science courses (see [9]). It also compared three different sets of features. The effect of window size on the different entity types was also studied [9].

[23] presented another relational learning based IE system, SNoW. It learned rules via a multi-class classifier by looking at a target fragment and its left and right windows. It was evaluated on the seminar announcements dataset.

[14] exploited a general statistical model, Hidden Markov Models (HMMs), for IE. It also used the shrinkage technique to deal with data sparseness for HMM parameter estimations. It was tested on two corpora, the seminar announcements, and a collection of newswire articles from Reuters. It used similar experimental settings to SRV and obtained better results on the seminars corpus.

[4] used a probabilistic discriminate model for IE and used maximum entropy for parameter estimations. It was tested on several corpora including seminar announcements, the CoNLL-2003 corpus (see [5]) and the datasets from MUC-6 and MUC-7 (see [3]).

All previous work used features from a window surrounding the current word, as well as features of the word itself. Both [18] and [7] investigated the effect of window size on the performance of rule-based learning and noticed that the computation times of both the rule learning algorithms BWI and $(LP)^2$ increased exponentially as the window size grew. On the other hand, the computation time in an SVM based system only increases linearly with window size. Therefore, it is easier for the SVM algorithm to select and use the optimal window size.

Basically the rule learning IE systems did not do any post-processing other than simple consistency checking – they treated each type of entity separately. The statistical learning algorithms compute a probability for each entity (or transfer the output into a probability as in the SVM based IE algorithms), such that they can select the best label for a fragment of text based on these probabilities. In order to select the best labels for a sentence, a Viterbi-like search algorithm was usually employed as a post-processor in the statistical learning systems.

[2] and [7] also investigated the effects of growing quantities of training data. [2] also considered active learning, where the system learns an initial model from a small pool of annotated examples and then, based on the learned model, selects additional examples for training.

6 Conclusions

This paper presents an SVM-based algorithm for IE and the experiments on three benchmark datasets – the CoNLL-2003 dataset, the CMU seminars corpus, and the software jobs corpus. The results show that our system is comparable to other state of the art systems for IE.

While other SVM-based IE systems used the original SVM model which treats the negative and positive margins equally, our system uses the SVM with

uneven margins. Our experiments show that the uneven margins SVM performs significantly better than the original SVM on the three datasets, particularly for small training sets.

In comparison to other similar SVM-based algorithms, our system is simpler, i.e., it needs a smaller number of SVM classifiers per entity type than the other two systems discussed respectively in [19] and [22]. Our system also obtained better results than the SVM-based system in [22] on the CoNLL-2003 corpus.

We investigated two weighting schemes for the features of the surrounding words and showed that the reciprocal weighting scheme performs slightly better than the commonly used equal weighting. We also investigated several post-processing procedures, ranging from using the SVM outputs for begin and end tags separately to selecting the highest probability label based on the output of all SVM classifiers. We found that, while overall post-processing can improve the results significantly, some of its stages only obtain small improvements.

Acknowledgements. We would like to thank the two anonymous reviewers for detailed comments and valuable suggestions. This research is supported by the EU-funded SEKT project (http://www.sekt-project.com).

References

1. Bender, O., Och, F. J., Ney, H.: Maximum entropy models for named entity recognition. In Walter Daelemans and Miles Osborne, editors, Proceedings of CoNLL-2003, 148–151. Edmonton, Canada, 2003
2. Califf, M. E.: Relational learning techniques for natural language information extraction. PhD thesis, University of Texas at Austin, 1998
3. Chieu, H. L., Ng, H. T.: A Maximum Entropy Approach to Information Extraction from Semi-Structured and Free Text. In Proceedings of the Eighteenth National Conference on Artificial Intelligence, 786–791, 2002
4. Chieu, H. L., Ng, H. T.: Named entity recognition: A maximum entropy approach using global information. In Proceedings of the 19th International Conference on Computational Linguistics (COLING'02), Taipei, Taiwan, 2002
5. Chieu, H. L., Ng, H. T.: Named entity recognition with a maximum entropy approach. In Walter Daelemans and Miles Osborne, editors, Proceedings of CoNLL-2003, 160–163. Edmonton, Canada, 2003
6. Cimiano, P., Handschuh, S., Staab, S.: Towards the self-Annotating Web. In Proceedings of WWW'04, 2004
7. Ciravegna, F.: $(LP)^2$, an adaptive algorithm for information extraction from web related texts. In Proceedings of the IJCAI-2001 Workshop on Adaptive Text Extraction and Mining, Seattle, 2001
8. Ciravegna, F.: $(LP)^2$, Rule Induction for Information Extraction Using Linguistic Constraints. Technical Report CS-03-07, Department of Computer Science, University of Sheffield, Sheffield, September 2003
9. Ciravegna, F., Wilks, Y.: Designing adaptive information extraction for the semantic Web in Amilcare. In S. Handschuh and S. Staab, editors, Annotation for the Semantic Web. IOS Press, Amsterdam, 2003
10. Cunningham, H.: Information extraction, automatic. Encyclopedia of Language and Linguistics, 2nd Edition, 2005

11. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V.: GATE: A framework and graphical development environment for robust NLP tools and applications. In Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02), 2002
12. Curran, J. R., Clark, S.: Language independent NER using a maximum entropy tagger. In Walter Daelemans and Miles Osborne, editors, Proceedings of CoNLL-2003, 164–167. Edmonton, Canada, 2003
13. Florian, R., Ittycheriah, A., Jing, H., Zhang, T.: Named entity recognition through classifier combination. In Walter Daelemans and Miles Osborne, editors, Proceedings of CoNLL-2003, pages 168–171. Edmonton, Canada, 2003
14. Freigtag, D., McCallum, A. K.: Information extraction with HMMs and shrinkage. In Proceedings of Workshop on Machine Learning for Information Extraction, pages 31–36, 1999
15. Freitag, D.: Information extraction from html: Application of a general learning approach. Proceedings of the Fifteenth Conference on Artificial Intelligence AAAI-98, pages 517–523, 1998
16. Freitag, D.: Machine Learning for Information Extraction in Informal Domains. PhD thesis, Carnegie Mellon University, 1998.
17. Freitag, D.: Machine Learning for Information Extraction in Informal Domains. Machine Learning, **39** (2000) 169–202
18. Freitag, D., Kushmerick, N.: Boosted Wrapper Induction. In Proceedings of AAAI 2000, 2000.
19. Isozaki, H., Kazawa, H.: Efficient Support Vector Classifiers for Named Entity Recognition. In Proceedings of the 19th International Conference on Computational Linguistics (COLING'02), pages 390–396, Taipei, Taiwan, 2002
20. Lewis, D. D., Yang, Y., Rose, T. G., Li, F.: Rcv1: A new benchmark collection for text categorization research. Journal of Machine Learning Research, **5** (2004) 361–397
21. Li, Y., Shawe-Taylor, J.: The SVM with uneven margins and Chinese document categorization. In Proceedings of The 17th Pacific Asia Conference on Language, Information and Computation (PACLIC17), pages 216–227, Singapore, Oct. 2003.
22. Mayfield, J., McNamee, P., Piatko, C.: Named entity recognition using hundreds of thousands of features. In Walter Daelemans and Miles Osborne, editors, Proceedings of CoNLL-2003, pages 184–187. Edmonton, Canada, 2003.
23. Roth, D., Yih, W. T.: Relational learning via propositional algorithms: an information extraction case study. In Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI), pages 1257 – 1263, 2001.
24. SAIC. Proceedings of the Seventh Message Understanding Conference (MUC-7). http://www.itl.nist.gov/iaui/894.02/related projects/muc/index.html, 1998.
25. Sang, E. F., Meulder, F. D.: Introduction to the CoNLL-2003 shared task: language-independent named entity recognition. In Walter Daelemans and Miles Osborne, editors, Proceedings of CoNLL-2003, pages 142–147. Edmonton, Canada, 2003.
26. Soderland, S.: Learning information extraction rules for semi-structured and free text. Machine Learning, **34** (1999) 233–272
27. Song, Y., Yi, E., Kim, E., Lee, G. G.: POSBIOTM-NER: a machine learning approach for bio-named entity recognition. In Workshop on a critical assessment of text mining methods in molecular biology, Granada, Spain (http://www.pdg.cnb.uam.es/BioLINK/workshop BioCreative 04/), 2004.
28. Yang, Y., Liu, X.: A re-examination of text categorization methods. In Proceedings of ACM SIGIR Conference on Research and Development in Information Retrieval, pages 42–49, 1999.

Author Index