CAMBRIDGE UNIVERSITY
ENGINEERING DEPARTMENT

Maximum Margin Training of Generative Kernels

M.I. Layton and M.J.F. Gales

CUED/F-INFENG/TR.484

June 2004

Cambridge University Engineering Department
Trumpington Street
Cambridge. CB2 1PZ
United Kindom

E-mail: {ml362,mjfg}@eng.cam.ac.uk

Abstract

Generative kernels, a generalised form of Fisher kernels, are a powerful form of kernel that allow the kernel parameters to be tuned to a specific task. The standard approach to training these kernels is to use maximum likelihood estimation. This paper describes a novel approach based on maximum-margin training of both the kernel parameters and a Support Vector Machine (SVM) classifier. It combines standard SVM training with a gradient-descent based kernel parameter optimisation scheme. This allows the kernel parameters to be explicitly trained for the data set and the SVM score-space. Initial results on an artificial task and the Deterding data show that such an approach can reduce classification error rates.

1 Introduction

Many problems, for example, speech, text and DNA-sequence data, consist of variable length sequences and can be difficult to process within the standard classification framework. For such data, it is usual to estimate a class-conditional generative model, such as an HMM, and then to use Bayes rule to calculate the posterior probability of the labels. However, many discriminative techniques, which directly estimate the posterior probability of the class labels, have in other areas, proved to be superior to generative models.

Generative models are commonly trained using Maximum-Likelihood Estimation (MLE). This views the model parameters, $\boldsymbol{\theta}$, as quantities that are fixed but unknown. Given a training set of independent and identically distributed examples, $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$, with labels, $Y = (y_1, \ldots, y_n)$, MLE chooses parameter values that maximise the joint probability of the training set. The MLE objective function is given by,

$$\mathcal{F}_{ML} = p(\boldsymbol{X}|\boldsymbol{\theta}) = \prod_{i=1}^{n} p(\boldsymbol{x}_i|\boldsymbol{\theta}) \tag{1}$$

For many generative models, a closed-form solution for estimating the model parameters does not exist. Iterative schemes, based upon the Expectation-Maximisation (EM) algorithm, are commonly used for a wide-range of models, including HMMs. However, MLE converges to the true parameter values only when the underlying model is correct and infinite training data exists. Under realistic conditions, the underlying models are not known and training data is limited. In such situations, discriminative training may be used. These techniques aim to train model parameters to directly minimise the error rate, or more commonly, a differentiable approximation to the error rate. One common discriminative training criterion is Maximum Mutual Information (MMIE) [3]. In addition to maximising the likelihood of correct hypotheses, this scheme attempts to reduce the probability of incorrect hypotheses. The MMIE objective function is given by,

$$\mathcal{F}_{MMI}(\boldsymbol{\theta}) = \sum_{i=1}^{n} \log \left[\frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_i|y_i)p(y_i)}{\sum_{j=1}^{C} p_{\boldsymbol{\theta}}(\boldsymbol{x}_i|y_j)p(y_j)} \right] \tag{2}$$

where C is the number of classes.

Another popular and successful discriminative classifier is the Support Vector Machine (SVM) [28]. SVMs have the desirable property that they attempt to minimise an upper bound on the generalisation error by maximising the margin of separation between two competing classes. Advantages often cited for using SVMs are: a strong statistical basis; the existence of a global optimum; and good generalisation to unseen data. However, SVM performance is intrinsically linked to the choice of kernel and the values of its associated parameters. For a particular task, the kernel is usually selected from a set of proposed kernels and parameters for that kernel are chosen either experimentally (to minimise test error on a validation set) or through some simple heuristic. Furthermore, although SVMs have been successfully applied to many static classification tasks, the SVM framework cannot directly classify variable length sequences, such as speech and text data. These limitations may be addressed through the use of *Generative kernels* [18, 25, 26]. These are a generalised form of Fisher kernel. For generative kernels, kernel parameters are learnt directly from the data, typically using Maximum Likelihood Estimation (MLE) [25, 26]. Maximum Mutual Information Estimation (MMIE) [3], a discriminative training scheme often used in speech processing, has also been used. However, these schemes do not guarantee an optimal kernel for the particular SVM classifier and score-space being used. In this report a more principled approach to kernel parameter estimation is described.

Although many techniques for adapting parameters are described in the Neural network literature, including maximum margin training of perceptrons [21], limited research has been conducted for SVMs. Early SVM parameter optimisation methods were aimed at finding parameters for specific kernel, such as [8, 11, 12] for GRBF kernels. Recently, however, more general kernel parameter optimisation schemes have been introduced. For example, Chapelle [9] proposed a computationally efficient technique for estimating the leave-one-out error (and thus the generalisation error) using support vectors; this was minimised by gradient descent. Subsequently, this was extended to minimise an analytic upper bound on the generalisation error [10]. Later, Ayat [2, 1] introduced a similar minimisation scheme based upon an empirical test error (calculated using a validation set). Faster convergence was achieved through the use of a quasi-Newton optimisation algorithm.

Alternative techniques, that aim to improve performance by adjusting parameters other than the kernel parameters, have also been developed. For example, Grandvalet [16] preprocessed the input-data using a diagonal scaling matrix; the components of the matrix were learnt by minimising an estimate of the empirical risk. In [7], Bousquet minimised the Rademacher Complexity of the kernel matrix, and in [20] Kandola varied the feature-space to maximise a measure of agreement between the kernel and data.

In this report, techniques for embedding generative models, such as Gaussian mixture models (GMMs), within SVM kernels are reviewed. A method of explicitly maximising the margin of separation between classes by adjusting the parameters of the generative model (the kernel parameters) is then proposed. This process amounts to training the generative model using a maximum-margin criterion. Although the techniques introduced in this report can be applied to any non-linear kernel, this report will concentrate solely on generative kernels.

2 Support Vector Machines

SVMs are an approximate implementation of the method of structural risk minimisation. This states that the generalisation error of a classifier is bounded by the sum of the training error and a term that depends on the Vapnik-Chervonenkis (VC) dimension [13]. For separable data, SVMs yield a training error of 0 and minimise the VC dimension. Vapnik [28] showed that the upper bound on generalisation error does not depend on the dimensionality of the space and is minimised by locating the unique hyperplane that maximises the distance between the hyperplane and the closest examples (as shown in figure 1).

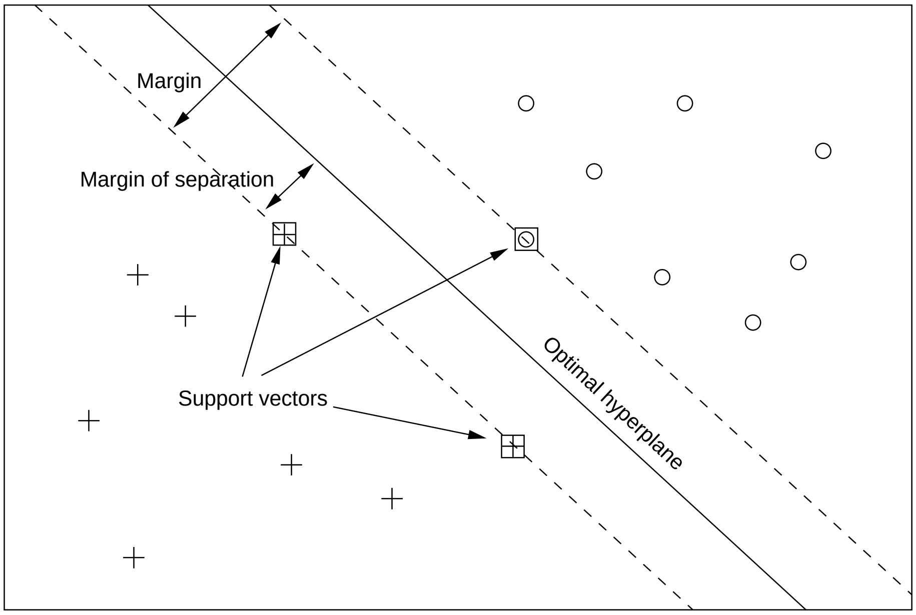

Figure 1: The optimal hyperplane for separable data

In figure 1, the examples in classes A and B are depicted by $+$ and $\circ$ respectively. The examples in each class closest to the optimal hyperplane are known as the support vectors and are depicted by $\boxplus$ and $\boxcircle$. The margin is defined as the distance (perpendicular to the optimal hyperplane) between the closest examples of class A and the closest examples of class B.

2.1 The optimal hyperplane

The concept of maximising the margin between the classes is an intuitive one. Since the training and test sets are assumed to have been generated by the same underlying process, it is reasonable to assume that the test data will lie 'close' to the training data. This means that large-margin classifiers have a degree of immunity to noise in both the training examples and the location of the hyperplane (since examples are far from the decision surface, small perturbations are unlikely to cause them to cross it and be misclassified). Accordingly, large-margin classifiers generalise well despite not incorporating problem-specific knowledge.

Given linearly separable training data $\boldsymbol{X} = (\boldsymbol{x}_1, ..., \boldsymbol{x}_n)$ with labels $Y = (y_1, ..., y_n)$ where $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1, +1\}$, a decision surface that separates the two classes is sought. The simplest such surface is a hyperplane,

$$\boldsymbol{w}^T \boldsymbol{x} + b = 0 \tag{3}$$

where $\boldsymbol{w}$ is an adjustable weight vector and b is the bias. The distance between the hyperplane and the closest example of each class is known as the *margin of separation*, denoted ρ. The hyperplane that maximises this separation is referred to as the *optimal* or *maximum-margin* hyperplane (figure 1); the optimal hyperplane is unique [28]. The geometric distance between the two classes, herein referred to as the margin of the SVM, is given by,

$$\text{Margin} = 2\rho = \frac{2}{|\boldsymbol{w}|} \tag{4}$$

The learning task for a SVM can thus be formulated as maximising the margin over $\boldsymbol{w}$ and b. An equivalent, and simpler to optimise, objective function is,

$$\min_{\boldsymbol{w},b} \tfrac{1}{2}(\boldsymbol{w} \cdot \boldsymbol{w}) \tag{5}$$

where optimisation is constrained by the decision rule, $y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1$. These constraints may be incorporated into the objective function using the method of Lagrange multipliers. The new objective function, L, is minimised over $\boldsymbol{w}$ and b and maximised over the Lagrange multipliers,

$$\max_{\boldsymbol{\alpha}} \min_{\boldsymbol{w},b} L(\boldsymbol{\alpha}, \boldsymbol{w}, b) = \frac{1}{2}(\boldsymbol{w} \cdot \boldsymbol{w}) - \sum_{i=1}^{n} \alpha_i (y_i(\boldsymbol{w} \cdot \boldsymbol{x} + b) - 1) \tag{6}$$

subject to $\sum_{i=1}^{n} \alpha_i y_i = 0$ and $\alpha_i \geq 0$. This is a constrained quadratic (convex) optimisation problem with a unique solution; many efficient algorithms have been developed for solving such problems. The optimal solution must also satisfy the Karush-Kuhn-Tucker (KKT) conditions,

$$\alpha_i \big[y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) - 1 \big] = 0 \tag{7}$$

At optimality, only examples that satisfy $\boldsymbol{w}^T \boldsymbol{x}_i + b = 1$ have $\alpha_i > 0$. These examples are known as the *support vectors* [28] and lie on the margin (see figures 1 and 2). In general, there are significantly fewer support vectors than training examples. This sparseness allows classification of test data to be performed efficiently.

Optimisation may also be performed using the dual objective function,

$$\max_{\alpha} W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j (\boldsymbol{x}_i \cdot \boldsymbol{x}_j) \tag{8}$$

subject to $\sum_{i=1}^{n} \alpha_i y_i = 0$ and $\alpha_i \geq 0$. This objective function is cast entirely in terms of the training data. Additionally, it depends only on the dot products of the input vectors $(\boldsymbol{x}_i \cdot \boldsymbol{x}_j)$ which are trivial to compute.

2.2 Soft-margin SVMs

In general, it is often not possible to construct a separating hyperplane between classes with no classification errors. In these situations, a hyperplane is found that minimises the probability of error, averaged over the training set. This is accomplished by introducing *soft margins*. The margin between classes is said to be soft if there exist training examples that violate the margin constraint, $y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1$. To permit this, slack variables, $\epsilon_i \geq 0$, can be introduced to measure the deviation of examples from the ideal condition of pattern separability ($\epsilon_i = 0$). The constraint thus becomes,

$$y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1 - \epsilon_i \tag{9}$$

Margin constraints may be violated in one of two ways:

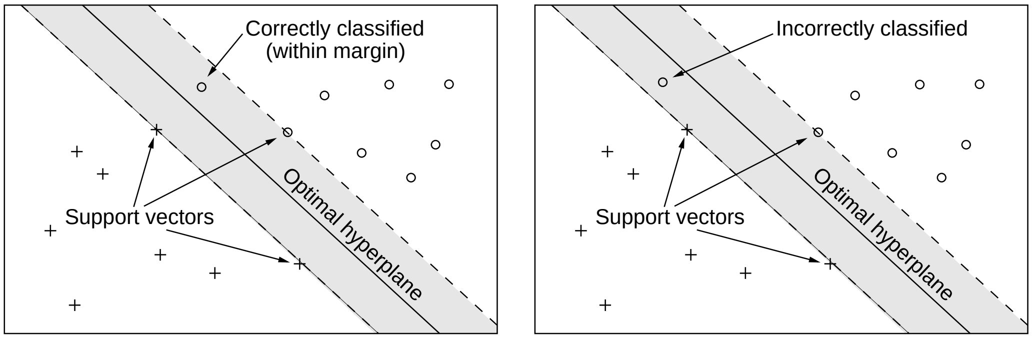

Figure 2: Example falls (a) within margin of separation; (b) on the wrong side of the decision hyperplane

- The example is on the correct side of the decision hyperplane but falls inside the margin of separation: $0 \leq \epsilon_i \leq 1$; figure 2(a)
- The example is on the incorrect side of the decision hyperplane: $\epsilon_i \geq 1$; figure 2(b)

The first example is correctly classified; the second is misclassified.

To reflect the need to maximise the margin whilst simultaneously minimising the number of misclassified examples, the primal objective function is rewritten as,

$$\min_{\boldsymbol{w},b} \tfrac{1}{2}(\boldsymbol{w} \cdot \boldsymbol{w}) + C \sum_{i=1}^{n} \epsilon_i \tag{10}$$

C acts as a regularisation parameter and controls the trade-off between the margin and the number of misclassified points. The value of C is selected by the user and is often determined experimentally using a validation set. The soft-margin dual optimisation problem may be written as,

$$\max_{\boldsymbol{\alpha}} W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j (\boldsymbol{x}_i \cdot \boldsymbol{x}_j) \tag{11}$$

subject to $\sum_{i=1}^{n} \alpha_i y_i = 0$ and $0 \leq \alpha_i \leq C$. This is similar to the separable case since neither the slack variables, ϵ_i, nor their Lagrange multipliers appear in the dual problem. However, unlike the separable case, the upper limit on the Lagrange multipliers, $\boldsymbol{\alpha}_i$, limits the influence of individual examples (which may be outliers).

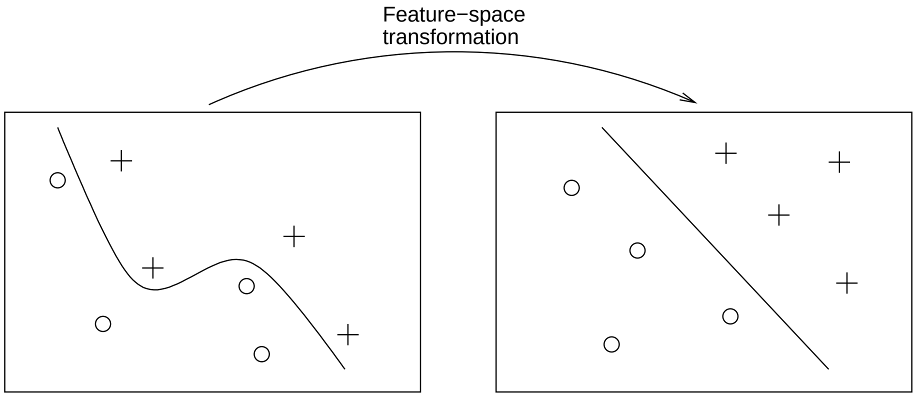

Figure 3: Use of a feature-space to make data linearly separable

2.3 Kernels

For non-linearly separable data, Cover's theorem [17] states that examples may be made linearly separable with a high probability given a non-linear transformation, $\boldsymbol{\phi}$, from input-space to a *feature-space* of sufficient dimensionality (figure 3). Using this mapping, the kernelised form of the dual-objective (11) is defined,

$$\max_{\alpha} W(\boldsymbol{\alpha},\boldsymbol{\theta}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j y_i y_j K(\boldsymbol{\theta},\boldsymbol{x}_i,\boldsymbol{x}_j) \tag{12}$$

where the *Kernel*, K, is given by,

$$K(\boldsymbol{\theta},\boldsymbol{x}_i,\boldsymbol{x}_j) = \boldsymbol{\phi}(\boldsymbol{\theta},\boldsymbol{x}_i)\cdot\boldsymbol{\phi}(\boldsymbol{\theta},\boldsymbol{x}_j) \tag{13}$$

An explicit feature-space need not be specified when defining the kernel. In general, the kernel may be any symmetric non-linear function that satisfies Mercer's condition [28, 13]. This states that the kernel may be any symmetric non-linear function whose kernel matrix,

$$\boldsymbol{K} = \left\{K(\boldsymbol{\theta},\boldsymbol{x}_i,\boldsymbol{x}_j)\right\}_{(i,j)=1}^{n} \tag{14}$$

is positive semi-definite. This ensures that an expansion,

$$K(\boldsymbol{\theta},\boldsymbol{x}_i,\boldsymbol{x}_j) = \sum_{k=1}^{\infty} a_k \boldsymbol{z}_k(\boldsymbol{x}_i)\boldsymbol{z}_k(\boldsymbol{x}_j) \tag{15}$$

into some unknown feature-space $(\boldsymbol{z}_1(\boldsymbol{x}),\ldots,\boldsymbol{z}_k(\boldsymbol{x}),\ldots)$ exists, where this function generates the inner-product, thus satisfying the requirements of the SVM. In this report, only normalised inner-product kernels, of the form,

$$K(\boldsymbol{\theta},\boldsymbol{x}_i,\boldsymbol{x}_j) = \boldsymbol{\phi}(\boldsymbol{x}_i,\boldsymbol{\theta})\boldsymbol{G}^{-1}(\boldsymbol{\theta})\boldsymbol{\phi}(\boldsymbol{x}_j,\boldsymbol{\theta}) \tag{16}$$

will be considered; these always satisfy Mercer's condition.

The computational cost of mapping input-data into a high-dimensional feature-space is prohibitive. However, since the kernel depends only on the inner-product of the mapped examples, $\boldsymbol{\phi}(\cdot)\cdot\boldsymbol{\phi}(\cdot)$, and not explicitly on $\boldsymbol{\phi}(\cdot)$, this may be overcome by defining kernels that combine the mapping and inner-product into one operation. Such kernels are useful since the feature-space is never explicitly calculated, allowing high-dimensional feature-spaces to be used efficiently. Examples of such kernels are given in table 1.

The simple inner-product kernel (discussed above) performs no feature-space transformation. The polynomial kernel transforms the input-data into a feature-space of monomials; the dimensionality is determined

Kernel	$K(\boldsymbol{x}_i, \boldsymbol{x}_j)$	**Parameters**
Inner-product	$\boldsymbol{x}_i \cdot \boldsymbol{x}_j$	
Inhomogeneous polynomial	$(\boldsymbol{x}_i \cdot \boldsymbol{x}_j + c)^p$	Power ($p \in \mathbb{N}$), $c > 0$
Sigmoid	$\tanh\left(\kappa(\boldsymbol{x}_i \cdot \boldsymbol{x}_j) + \theta\right)$	$\kappa > 0, \theta < 0$
GRBF	$\exp\left(-\frac{\|\|\boldsymbol{x}_i - \boldsymbol{x}_j\|\|^2}{2\sigma^2}\right)$	Kernel width ($\theta > 0$)

Table 1: Commonly used kernel functions

by both the number of support vectors and a user-specified power, p. For the GRBF kernel, the number of radial-basis functions and their centres are determined automatically by the number and values of the support vectors respectively. The Sigmoid kernel creates a two-layer perceptron SVM. The number of hidden neurons and their weight vectors are determined by the number of support vectors and their values. Although the Sigmoid kernel is never positive definite [23] (and thus not strictly a SVM kernel), in practice, it has been used successfully.

3 Score-spaces

Dynamic data, such as speech, typically consists of variable-length sequences of observations. However, since SVMs require a feature-space with a fixed dimensionality, a method of mapping variable-length data into fixed-length vectors is required. Although several methods have been proposed for performing such a mapping, many rely on sampling techniques, and thus discard (potentially) useful information. Examples of such techniques are [4, 24, 14, 15]. An alternative approach, first introduced by Jaakkola and Hausser [18] and later extended in [22, 25, 26], is to train generative models on the data and then define kernels based upon them. These kernels are known as generative kernels.

3.1 Log-likelihood and log-likelihood-ratio feature-spaces

The simplest generative feature-space is a log-likelihood (LL) feature-space. SVM training and classification are performed using the log posterior probabilities of the generative model. The resulting feature-space is 2-dimensional,

$$\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{\theta}_A, \boldsymbol{\theta}_B) = \begin{bmatrix} \ln p(\boldsymbol{x}|\boldsymbol{\theta}_A) \\ \ln p(\boldsymbol{x}|\boldsymbol{\theta}_B) \end{bmatrix} \tag{17}$$

where,

$$\ln p(\boldsymbol{x}|\boldsymbol{\theta}) = -\tfrac{1}{2}\ln\left[(2\pi)^d|\boldsymbol{\Sigma}|\right] - \tfrac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})$$

and d is the dimensionality of the input-data. A disadvantage of log-likelihood score-spaces is that competing classes are not considered during classification. A more discriminative score-space without this limitation is the log-likelihood-ratio (LLR) score-space,

$$\begin{aligned} \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{\theta}_A, \boldsymbol{\theta}_B) &= \ln\left[\frac{p(\boldsymbol{x}|\boldsymbol{\theta}_A)}{p(\boldsymbol{x}|\boldsymbol{\theta}_B)}\right] \\ &= \ln p(\boldsymbol{x}|\boldsymbol{\theta}_A) - \ln p(\boldsymbol{x}|\boldsymbol{\theta}_B) \\ &= \tfrac{1}{2}\ln\left(\frac{|\boldsymbol{\Sigma}_B|}{|\boldsymbol{\Sigma}_A|}\right) - \tfrac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_A)^T\boldsymbol{\Sigma}_A^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_A) \\ &\qquad + \tfrac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_B)^T\boldsymbol{\Sigma}_B^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_B) \end{aligned} \tag{18}$$

A simple inner-product kernel may be defined for each of these feature-spaces.

Although the use of generative models allows SVMs to classify sequences of observations, preprocessing creates a layer of abstraction between the input-data and the classifier, limiting the information available to the classifier. This significantly reduces its power by limiting the flexibility in the positioning of the decision surface (the input-data is reduced from being d-dimensional to 2-dimensional).

3.2 Fisher score-space

The LL and LLR feature-spaces are defined using the posterior probabilities of the class-conditional generative models. In [18], Jaakkola and Haussler proposed score-spaces that capture the differences in the generative process between examples using the gradient-space (a sufficient statistic) of the generative model. The likelihood score-space (also known as the Fisher score-space) [18, 22] is thus defined as,

$$\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \ln p(\boldsymbol{x}|\boldsymbol{\theta}) \tag{19}$$

Since the score-space components have different dynamic ranges, the kernel is defined by a normalised inner-product,

$$K(\boldsymbol{\theta}, \boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{\phi}(\boldsymbol{x}_i, \boldsymbol{\theta})^T \boldsymbol{G}^{-1} \boldsymbol{\phi}(\boldsymbol{x}_j, \boldsymbol{\theta}) \tag{20}$$

where $\boldsymbol{G}$ is the covariance of the score-space,

$$\boldsymbol{G} = \mathcal{E}\Big\{ \Big(\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{\theta}) - \boldsymbol{\mu}_{\boldsymbol{\phi}}\Big)\Big(\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{\theta}) - \boldsymbol{\mu}_{\boldsymbol{\phi}}\Big)^T \Big\} \tag{21}$$

$$\boldsymbol{\mu}_{\boldsymbol{\phi}} = \mathcal{E}\big\{\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{\theta})\big\} \tag{22}$$

When the kernel parameters are estimated using MLE, the score-space has a zero-mean and $\boldsymbol{G}$ reduces to the Fisher information matrix. Under these circumstances, the kernel is known as a *Fisher Kernel*.

3.3 Log-likelihood ratio score-spaces

A more discriminative score-space can be defined using the first-derivatives of the log-likelihood-ratio; these are known as log-likelihood-ratio (LLR) score-spaces [27],

$$\begin{aligned} \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{\theta}_A, \boldsymbol{\theta}_B) &= \nabla_{\boldsymbol{\theta}_A, \boldsymbol{\theta}_B} \ln \left(\frac{p(\boldsymbol{x}|\boldsymbol{\theta}_A)}{p(\boldsymbol{x}|\boldsymbol{\theta}_B)} \right) \\ &= \begin{bmatrix} \nabla_{\boldsymbol{\theta}_A} \ln p(\boldsymbol{x}|\boldsymbol{\theta}_A) \\ \nabla_{\boldsymbol{\theta}_B} \ln p(\boldsymbol{x}|\boldsymbol{\theta}_B) \end{bmatrix} \end{aligned} \tag{23}$$

However, when mixture models are used, LL and LLR score-spaces suffer from wrap-around [25, 27]. This occurs when different parts of input-space map to the same point in the feature-space, increasing confusion between classes and reducing performance. This can be avoided by using multiple generative models, one per class [25, 26]. The score-spaces can be further generalised to include higher-order terms as well as the log-likelihood ratio [25, 26]. The LLR term also increases the information explicitly available to the classifier (the gradient-space forms a sufficient statistic and so implicitly contains the posterior probability). An example score-space, in which the SVM can be trained, is given by,

$$\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{\theta}_A, \boldsymbol{\theta}_B) = \begin{bmatrix} \ln p(\boldsymbol{x}|\boldsymbol{\theta}_A) - \ln p(\boldsymbol{x}|\boldsymbol{\theta}_B) \\ \nabla_{\boldsymbol{\theta}_A} \ln p(\boldsymbol{x}|\boldsymbol{\theta}_A) \\ \nabla_{\boldsymbol{\theta}_B} \ln p(\boldsymbol{x}|\boldsymbol{\theta}_B) \end{bmatrix} \tag{24}$$

Similarly to the Fisher score-space, the kernel is defined by the normalised inner-product,

$$K(\boldsymbol{\theta}, \boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{\phi}(\boldsymbol{x}_i, \boldsymbol{\theta})^T \boldsymbol{G}^{-1} \boldsymbol{\phi}(\boldsymbol{x}_j, \boldsymbol{\theta}) \tag{25}$$

$$\boldsymbol{G} = \mathcal{E}\Big\{ \Big(\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{\theta}) - \boldsymbol{\mu}_{\boldsymbol{\phi}}\Big)\Big(\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{\theta}) - \boldsymbol{\mu}_{\boldsymbol{\phi}}\Big)^T \Big\} \tag{26}$$

$$\boldsymbol{\mu}_{\boldsymbol{\phi}} = \mathcal{E}\big\{\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{\theta})\big\} \tag{27}$$

In general, it is not possible to calculate the expectation values directly. However, since the training data is independent and identically distributed, the calculations may be approximated by a summation over the training data,

$$\boldsymbol{G} = \frac{1}{n}\sum_{i=1}^{n}\Big(\boldsymbol{\phi}(\boldsymbol{x}_i,\boldsymbol{\theta}) - \boldsymbol{\mu}_{\boldsymbol{\phi}}\Big)\Big(\boldsymbol{\phi}(\boldsymbol{x}_i,\boldsymbol{\theta}) - \boldsymbol{\mu}_{\boldsymbol{\phi}}\Big)^T \tag{28}$$

$$\boldsymbol{\mu}_{\boldsymbol{\phi}} = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{\phi}(\boldsymbol{x}_i,\boldsymbol{\theta}) \tag{29}$$

For efficient inversion, it is advantageous to approximate $\boldsymbol{G}$ as either block-diagonal or diagonal.

3.4 GMM-based LLR score-spaces

In this report, m-component GMMs with diagonal covariance matrices are used as an example generative model (one per class). These are a common form of generative model since, given sufficient mixture components, GMMs can (with mild restrictions) approximate any source distribution. The use of diagonal covariance matrices significantly reduces the number of parameters to be estimated (compared to full covariance matrices) and so increases efficiency of calculations. The posterior probability of the GMM is given by,

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \sum_{j=1}^{m} c_j\mathcal{N}(\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j) \tag{30}$$

The score-space contains the derivatives of the LLR. For a particular class, these are given by,

$$\begin{aligned}
\nabla_{\boldsymbol{\mu}_j}\ln\left(p(\boldsymbol{x}|\boldsymbol{\theta})\right) &= \gamma_j(\boldsymbol{x})\Big[\boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_j)\Big]\\
\nabla_{\boldsymbol{\Sigma}_j}\ln\left(p(\boldsymbol{x}|\boldsymbol{\theta})\right) &= \frac{\gamma_j(\boldsymbol{x})}{2}\mathrm{diag}\Big[-\boldsymbol{\Sigma}_j^{-1} + \Big(\boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_j)\Big)\Big(\boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_j)\Big)^T\Big]
\end{aligned}$$

where,

$$\gamma_j(\boldsymbol{x}) = \frac{c_j\mathcal{N}(\boldsymbol{x};\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j)}{\sum_{j=1}^{m} c_j\mathcal{N}(\boldsymbol{x};\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j)}$$

and j is the GMM mixture component. A GMM is specified for each class. The kernel normalisation term is calculated using either equation (26) or the approximation given in (28).

Training the SVM with an inner-product GMM-based LLR kernel requires explicit mapping of training examples into the score-space. This operation has a computational cost of $\mathcal{O}(nmd)$ multiply-accumulates, where n is the number of training examples, m is the number of mixture-components per generative model and d is the dimensionality of the data.

4 Maximum margin training of generative kernels

For a given learning task, the performance of the classifier is affected by both its structure and the values of its embedded parameters. If these are not well chosen, classification performance will be reduced. For SVMs, performance is governed by: the kernel, the parameters of the kernel, and the SVM regularisation parameter (C). The kernel is chosen either by prior knowledge or selected experimentally from a set of proposed kernels (choosing the one that minimises the test error on a validation set). The regularisation parameter, C, can also be selected experimentally. Alternatively, an automated technique, such as [5], can be used. In this report, to allow classification of dynamic data, a generative kernel is assumed. For simplicity, the regularisation parameter, C, is set empirically and assumed to be fixed.

4.1 Kernel parameter optimisation

This section considers the problem of choosing kernel parameters to optimise classification performance. A standard parameter selection technique involves performing a search over the kernel parameter-space to find parameters that minimise error on a validation set. Since an explicit search of parameter-space is infeasible for generative kernels, parameters are usually learnt from the data using MLE [25, 26] or MMIE. However, these training schemes do not guarantee an optimal kernel for the SVM classifier and score-space being used.

For a given score-space and kernel, this report presents a method of determining the kernel parameters by explicitly maximising the margin of separation. This amounts to maximum-margin (MM) training of the generative model. Unlike the simpler search of parameter-space, a validation set is not required. Nor does it require time-consuming estimates of the generalisation error to be calculated (as required, for example, in [10]). Instead, the soft-margin dual objective-function (12) is generalised to include kernel parameter optimisation,

$$\max_{\boldsymbol{\alpha}} \min_{\boldsymbol{\theta}} W(\boldsymbol{\theta}, \boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\boldsymbol{\theta}, \boldsymbol{x}_i, \boldsymbol{x}_j) \tag{31}$$

From this expression, it is clear that minimising the objective function, W, with respect to the kernel parameters is equivalent to maximising the kernel. Using this approach, the kernel parameters may be explicitly tuned to the score-space being used. Note that, for unnormalised generative kernels ($\boldsymbol{G} = \boldsymbol{I}$), the system will attempt to attain an infinitely large margin by shifting the model parameters away from the underlying data.

Although the Lagrange multipliers and kernel parameters could be optimised simultaneously, standard (and efficient) SVM techniques rely on the convex structure of the conventional SVM objective function, and so would be inapplicable. Instead, an iterative process is proposed: the kernel parameters and Lagrange multipliers (support vectors) are each updated whilst assuming the other fixed. The algorithm is given below:

---

1. Initialise parameters, $\boldsymbol{\theta}^{(0)}$, of generative model using MLE
2. Train SVM to locate initial support vectors, $\boldsymbol{\alpha}^{(0)}$
3. Calculate initial value of objective function, $W^{(0)} = W(\boldsymbol{\theta}^{(0)}, \boldsymbol{\alpha}^{(0)})$
4. For each iteration k:
   - $\boldsymbol{\theta}^{(k)} = \arg\min_\theta \left(W(\boldsymbol{\theta}; \boldsymbol{\alpha}^{(k-1)})\right)$ (A)
   - $\boldsymbol{\alpha}^{(k)} = \arg\max_\alpha \left(W(\boldsymbol{\alpha}; \boldsymbol{\theta}^{(k)})\right)$ (B)
   - Recalculate objective function, $W^{(k)} = W(\boldsymbol{\theta}^{(k)}, \boldsymbol{\alpha}^{(k)})$

   Repeat until convergence: $|W^{(k)} - W^{(k-1)}| < \epsilon$

Step (B) in the above process is the standard SVM optimisation. Step (A) is an unconstrained optimisation that adjusts the kernel parameters, $\boldsymbol{\theta}$, to minimise the objective function, W. No general closed-form solution is available, so gradient descent optimisation may be used.

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \eta \nabla_{\boldsymbol{\theta}} W(\boldsymbol{\theta}, \boldsymbol{\alpha}) \big|_{\boldsymbol{\theta}^{(k)}} \tag{32}$$

where η is the gradient-descent step-size. Since the kernel is changing, it is not possible to guarantee that the KKT conditions are still satisfied. This may result in W increasing after the support vectors have been updated. Additionally, since the objective function may vary discontinuously with the support vectors, the system can become unstable with respect to changes in $\boldsymbol{\theta}$ since even small changes can result in large changes in W.

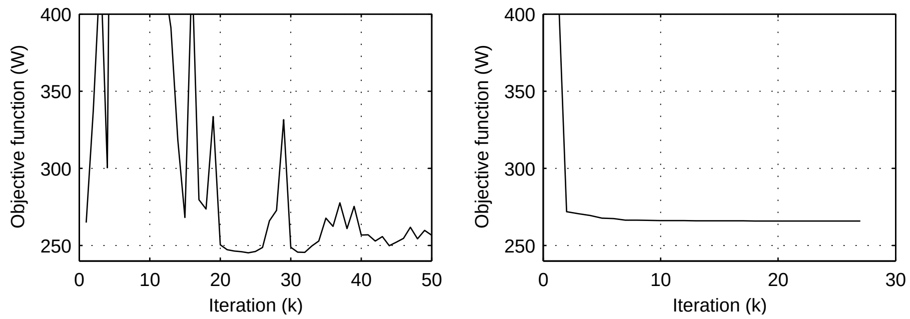

Figure 4: Objective function for a LLR kernel; constant step-size (left) and with back-off

This problem is illustrated on the left-hand side of figure 4. Here, both steps (A) and (B) independently improve the objective function, but the combined effect is a non-monotonically changing function. Discontinuities mean that convergence is not guaranteed.

The optimisation may be stabilised through the use of a back-off algorithm. After each iteration, if $W^{(k)} > W^{(k-1)}$, the system backs-off to its previous state ($\boldsymbol{\theta}^{(k-1)}, \boldsymbol{\alpha}^{(k-1)}$) and the step-size, η, for the gradient descent is reduced. This ensures that the objective function can never increase and thus ensures that W will be lower than (or at worst, equal to) that for a SVM with constant kernel parameters. Performance should therefore be improved. The disadvantage of this technique is that, whilst guaranteeing stability, the system is biased against large changes in kernel parameters, increasing the likelihood of convergence to a local minimum. The right-hand side of figure 4 shows the stabilised version.

In order to optimise the kernel parameters using gradient descent, the derivative of the objective function with respect to the kernel parameters is required. This is given by the gradient of the kernel summed over the current support vectors,

$$\nabla_{\boldsymbol{\theta}} W(\boldsymbol{\alpha}, \boldsymbol{\theta}) = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \nabla_{\boldsymbol{\theta}} K(\boldsymbol{\theta}, \boldsymbol{x}_i, \boldsymbol{x}_j) \tag{33}$$

4.2 Kernel gradients

For the normalised inner-product kernel defined in section 3.3, the kernel gradient is given by,

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} K(\boldsymbol{\theta}, \boldsymbol{x}_i, \boldsymbol{x}_j) = &\Big[\nabla_{\boldsymbol{\theta}} \boldsymbol{\phi}(\boldsymbol{x}_i, \boldsymbol{\theta})\Big]^T \boldsymbol{G}^{-1}(\boldsymbol{\theta}) \boldsymbol{\phi}(\boldsymbol{x}_j, \boldsymbol{\theta}) + \\ &\boldsymbol{\phi}(\boldsymbol{x}_i, \boldsymbol{\theta})^T \Big[\nabla_{\boldsymbol{\theta}} \boldsymbol{G}^{-1}(\boldsymbol{\theta})\Big] \boldsymbol{\phi}(\boldsymbol{x}_j, \boldsymbol{\theta}) + \boldsymbol{\phi}(\boldsymbol{x}_i, \boldsymbol{\theta})^T \boldsymbol{G}^{-1}(\boldsymbol{\theta}) \Big[\nabla_{\boldsymbol{\theta}} \boldsymbol{\phi}(\boldsymbol{x}_j, \boldsymbol{\theta})\Big]\end{aligned} \tag{34}$$

By symmetry, the first and last terms are identical over the summation in (33). The normalisation term, $\boldsymbol{G}$, can be calculated using either (26) or the approximation in (28).

Consider the example score-space given in (24). The differentials of the LLR-component of score-space are contained within the score-space itself. The differentials of the first-order score-space components are given by (see Appendix A for derivation),

$$\nabla_{\boldsymbol{\theta}_k} \big(\nabla_{\boldsymbol{\theta}_j} \ln p(\boldsymbol{x}|\boldsymbol{\theta})\big) = \frac{\nabla^2_{\boldsymbol{\theta}_j, \boldsymbol{\theta}_k} p(\boldsymbol{x}|\boldsymbol{\theta})}{p(\boldsymbol{x}|\boldsymbol{\theta})} - \Big[\nabla_{\boldsymbol{\theta}_j} \ln p(\boldsymbol{x}|\boldsymbol{\theta})\Big] \Big[\nabla_{\boldsymbol{\theta}_k} \ln p(\boldsymbol{x}|\boldsymbol{\theta})\Big]^T \tag{35}$$

This equation requires the generative model to be twice differentiable with respect to each of its parameters. The second and third terms are components of the score-space and thus require no further calculation. For the kernel parameter $\boldsymbol{\theta}_k$, the gradient of the normalisation matrix is given by,

$$\nabla_{\boldsymbol{\theta}_k} (\boldsymbol{G}^{-1}) = -\boldsymbol{G}^{-1}\, \mathcal{E}\Big\{ 2 \Big(\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{\theta}) - \boldsymbol{\mu}_{\boldsymbol{\phi}}\Big) \Big(\nabla_{\boldsymbol{\theta}_k} \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{\theta})\Big)^T \Big\}\, \boldsymbol{G}^{-1} \tag{36}$$

Since the training data is independent and identically distributed, the expectation value can be approximated by a summation over the training data,

$$\nabla_{\boldsymbol{\theta}_k}(\boldsymbol{G}^{-1}) = -\boldsymbol{G}^{-1}\left[\frac{2}{n}\sum_{i=1}^{n}\Big(\boldsymbol{\phi}(\boldsymbol{x}_i,\boldsymbol{\theta}) - \boldsymbol{\mu}_{\boldsymbol{\phi}}\Big)\Big(\nabla_{\boldsymbol{\theta}_k}\boldsymbol{\phi}(\boldsymbol{x}_i,\boldsymbol{\theta})\Big)^T\right]\boldsymbol{G}^{-1} \tag{37}$$

4.3 Application to GMMs

An example generative model is a m-component GMM with diagonal covariance matrices,

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \sum_{j=1}^{m} c_j \mathcal{N}(\boldsymbol{x};\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j) \tag{38}$$

Using separate GMMs for each class, the LLR score-space (24) is defined by,

$$\boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{\theta}_A,\boldsymbol{\theta}_B) = \begin{bmatrix} \ln p(\boldsymbol{x}|\boldsymbol{\theta}_A) - \ln p(\boldsymbol{x}|\boldsymbol{\theta}_B) \\ \nabla_{\boldsymbol{\theta}_A} \ln p(\boldsymbol{x}|\boldsymbol{\theta}_A) \\ \nabla_{\boldsymbol{\theta}_B} \ln p(\boldsymbol{x}|\boldsymbol{\theta}_B) \end{bmatrix} \tag{39}$$

where (for constant component priors), $\boldsymbol{\theta}_A = [\boldsymbol{\mu}_A^T, \boldsymbol{\Sigma}_A]^T$ and $\boldsymbol{\theta}_B = [\boldsymbol{\mu}_B^T, \boldsymbol{\Sigma}_B]^T$. Equation (39) may be rewritten more explicitly as,

$$\boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{\theta}_A,\boldsymbol{\theta}_B) = \begin{bmatrix} \ln p(\boldsymbol{x}|\boldsymbol{\theta}_A) - \ln p(\boldsymbol{x}|\boldsymbol{\theta}_B) \\ \nabla_{\boldsymbol{\mu}_A} \ln p(\boldsymbol{x}|\boldsymbol{\theta}_A) \\ \nabla_{\boldsymbol{\Sigma}_A} \ln p(\boldsymbol{x}|\boldsymbol{\theta}_A) \\ \nabla_{\boldsymbol{\mu}_B} \ln p(\boldsymbol{x}|\boldsymbol{\theta}_B) \\ \nabla_{\boldsymbol{\Sigma}_B} \ln p(\boldsymbol{x}|\boldsymbol{\theta}_B) \end{bmatrix} \tag{40}$$

Since classes A and B are independent, the gradient of the score-space $\boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{\theta})$ with respect to the kernel parameters is given by

$$\nabla_{\boldsymbol{\theta}_A}\boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{\theta}_A) = \begin{bmatrix} \nabla_{\boldsymbol{\theta}_A} l(\boldsymbol{x}|\boldsymbol{\theta}_A) \\ \nabla^2_{\boldsymbol{\mu}_A,\boldsymbol{\theta}_A} l(\boldsymbol{x}|\boldsymbol{\theta}_A) \\ \nabla^2_{\boldsymbol{\Sigma}_A,\boldsymbol{\theta}_A} l(\boldsymbol{x}|\boldsymbol{\theta}_A) \\ 0 \\ 0 \end{bmatrix}, \ \nabla_{\boldsymbol{\theta}_B}\boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{\theta}_B) = \begin{bmatrix} -\nabla_{\boldsymbol{\theta}_B} l(\boldsymbol{x}|\boldsymbol{\theta}_B) \\ 0 \\ 0 \\ \nabla^2_{\boldsymbol{\mu}_B,\boldsymbol{\theta}_B} l(\boldsymbol{x}|\boldsymbol{\theta}_B) \\ \nabla^2_{\boldsymbol{\Sigma}_B,\boldsymbol{\theta}_B} l(\boldsymbol{x}|\boldsymbol{\theta}_B) \end{bmatrix} \tag{41}$$

where,

$$l(\boldsymbol{x},\boldsymbol{\theta}) = \ln p(\boldsymbol{x}|\boldsymbol{\theta})$$

The cross-derivatives, $\nabla^2_{\boldsymbol{\theta}_A,\boldsymbol{\theta}_B}$ and $\nabla^2_{\boldsymbol{\theta}_B,\boldsymbol{\theta}_A}$ are zero since classes A and B are independent. Using the general equation, (35), the within-class derivatives may be calculated using,

$$\nabla^2_{\boldsymbol{\theta}_j,\boldsymbol{\theta}_k} l(\boldsymbol{x}|\boldsymbol{\theta}) = \begin{cases} -\Big[\nabla_{\boldsymbol{\theta}_j} l(\boldsymbol{x}|\boldsymbol{\theta})\Big]\Big[\nabla_{\boldsymbol{\theta}_k} l(\boldsymbol{x}|\boldsymbol{\theta})\Big]^T & \text{if } j \neq k \\ \gamma_j\Big\{\nabla^2_{\boldsymbol{\theta}_j^2} l_j(\boldsymbol{x}|\boldsymbol{\theta}) + \Big[1-\gamma_j\Big]\Big[\nabla_{\boldsymbol{\theta}_j} l(\boldsymbol{x}|\boldsymbol{\theta})\Big]\Big[\nabla_{\boldsymbol{\theta}_j} l(\boldsymbol{x}|\boldsymbol{\theta})\Big]^T\Big\} & \text{if } j = k \end{cases}$$

where

$$\gamma_j(\boldsymbol{x}) = \frac{c_j\mathcal{N}(\boldsymbol{x};\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j)}{\sum_{j=1}^{m} c_j\mathcal{N}(\boldsymbol{x};\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j)}, \qquad l_j(\boldsymbol{x}|\boldsymbol{\theta}) = \ln\mathcal{N}(\boldsymbol{x}|\boldsymbol{\theta}_j)$$

and

$$\begin{aligned} \nabla^2_{\boldsymbol{\mu}_j,\boldsymbol{\mu}_j} l_j(\boldsymbol{x}|\boldsymbol{\theta}) &= -\boldsymbol{\Sigma}_j^{-1} \\ \nabla^2_{\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j} l_j(\boldsymbol{x}|\boldsymbol{\theta}) &= \mathrm{Diag}\Big[-\boldsymbol{\Sigma}_j^{-1}\boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_j)\Big] \\ \nabla^2_{\boldsymbol{\Sigma}_j,\boldsymbol{\Sigma}_j} l_j(\boldsymbol{x}|\boldsymbol{\theta}) &= \frac{1}{2}\boldsymbol{\Sigma}_j^{-1}\boldsymbol{\Sigma}_j^{-1} - \mathrm{Diag}\Big\{\mathrm{diag}\Big[\Big(\boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_j)\Big)\Big(\boldsymbol{\Sigma}_j^{-1}\boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_j)\Big)^T\Big]\Big\} \end{aligned}$$

Derivations are given in appendix B.

4.4 Computational cost

As discussed in section 3.4, training the SVM with a GMM-based LLR kernel requires training examples to be mapped into the generative score-space. The computational cost of this operation is $\mathcal{O}(nmd)$ multiply-accumulates, where n is the number of training examples, m is the number of mixture-components and d is the dimensionality of the data.

The first step of the MM parameter adjustment requires the gradient, $\nabla_{\boldsymbol{\theta}}$, to be calculated. From equation (33), it is clear that this calculation scales quadratically with the number of support vectors and requires calculation of the kernel gradient. This requires derivatives of the score-space and normalisation term to be calculated with respect to each of the kernel parameters. These have a cost of $\mathcal{O}(m^2d^2)$ and $\mathcal{O}(m^2d^2n)$ respectively. Thus, the total cost of the gradient descent optimisation for each MM iteration is $\mathcal{O}(m^2d^2(s^2+n))$. Since, for many applications, the number of support vectors is significantly less than the number of training examples, computation scales as $\mathcal{O}(m^2d^2n)$.

The second step of each MM iteration requires retraining the SVM – a time consuming operation. However, since the changes in kernel parameters are small, the changes in Lagrange multipliers are also expected to be small. Thus, if the SVM is initialised with the results of the previous iteration, the SVM is likely to be close to optimality and so will require significantly fewer iterations to train.

Finally, it should be noted that the overall computational cost of training can be significantly reduced by caching results (and partial results) from each stage of training. If caching is used, memory requirements for the gradient descent scheme scale linearly with the number of training examples. Requirements for SVM training scale quadratically with the number of training examples (to store the kernel matrix (14)). This means that, for large data sets, memory requirements can become prohibitive. However, practical SVM trainers, for example SVM$^{\text{light}}$, mitigate this problem by using chunking algorithms [19].

5 Experimental results

To evaluate the performance of maximum margin training of kernel parameters, two sets of experiments were performed. The first was a binary classification task with ‘artificial’ data generated from known sources. The second used a standard task, the Deterding data [6]. All SVMs were trained using a modified version of SVM$^{\text{light}}$ [19] configured with a linear kernel and $C = 1$.

5.1 Artificial data

Two-class artificial data was generated from a 3-component GMM. The training and test sets consisted of 1,000 and 20,000 examples respectively, split equally between the two classes. The training set was restricted to 1,000 examples to limit the computational cost of training. The size of the test set was chosen to allow accurate estimates of the generalisation error of trained classifiers. Given the true source distribution, the Bayes decision rule yielded the optimal decision surface and gave the minimum generalisation error as 9.3%. Since, for most practical problems, the source distribution is unknown, classifiers must be constructed using approximate models. Our experiments simulate this by introducing a mismatch between the true model (a 3-component GMM) and the classifier model (a 1- or 2-component GMM with diagonal covariance matrices).

Table 2 shows the performance of various standard classifiers on this task. For a 1-component system, MLE yielded an error rate of 23.0%. Increasing the number of mixture-components increased the power of the classifier and reduced the error significantly to 9.6%. The discriminative classifier (MMIE) performed significantly better than MLE for the 1-component system, yielding an error of just 11.3%. However, additional components had little effect on the error. A soft-margin SVM with a standard linear kernel gave an error of 10.8%.

As a simple example for MM training, a single Gaussian for each class was used as the kernel. The Gaussians were first initialised by MLE and then trained using the MM algorithm with a score-space based

Classifier	Components	Training (%)	Test (%)
Minimum Bayes Error	3		9.3
GMM (MLE)	1	24.0	23.0
GMM (MLE)	2	10.0	9.6
GMM (MMIE)	1	12.1	11.3
GMM (MMIE)	2	11.2	11.1
SVM with linear kernel		10.8	10.8

Table 2: Baseline error rates for artificial data

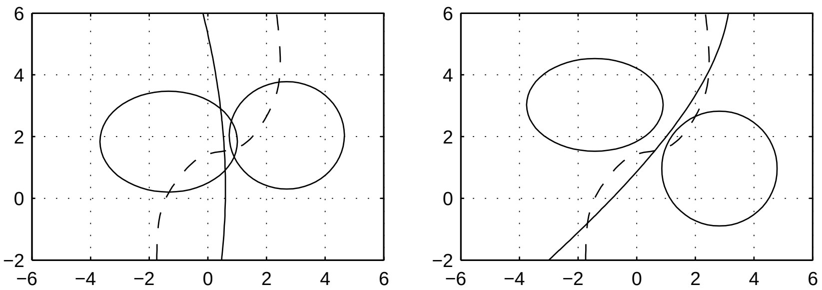

Figure 5: LLR (solid) and Bayes (dashed) decision boundaries before (left) and after (right) parameter optimisation

on the LLR. Figure 5 shows the change in the position of the kernel and the resultant decision boundary. MM-training significantly altered the kernel position, moving the decision boundary closer to the Bayes decision boundary.

	Training (%)		Test (%)		Objective		Margin	
Kernel	**Initial**	**Final**	**Initial**	**Final**	**Initial**	**Final**	**Initial**	**Final**
1-Component								
LLR	23.8	10.5	22.8	10.4	477.1	265.8	1.24	0.70
LLR+$\nabla_{\boldsymbol{\mu}}$ (diag)	10.7	10.4	10.8	10.4	269.8	263.4	0.94	0.99
LLR+$\nabla_{\boldsymbol{\mu}}$ (block)	10.7	10.4	10.8	10.3	268.8	262.9	1.22	1.11
LLR+$\nabla_{\boldsymbol{\mu}}$+$\nabla_{\boldsymbol{\Sigma}}$ (diag)	10.4	10.5	10.4	10.3	263.6	262.8	0.95	1.16
LLR+$\nabla_{\boldsymbol{\mu}}$+$\nabla_{\boldsymbol{\Sigma}}$ (block)	10.5	10.5	10.3	10.3	263.0	262.8	1.08	1.15
2-Component								
LLR	10.2	9.3	10.2	9.3	264.9	234.0	0.49	0.70
LLR+$\nabla_{\boldsymbol{\mu}}$ (diag)	9.3	9.2	9.5	9.4	235.1	227.6	0.87	0.99
LLR+$\nabla_{\boldsymbol{\mu}}$ (block)	9.4	9.3	9.5	9.3	235.0	228.7	0.88	1.02
LLR+$\nabla_{\boldsymbol{\mu}}$+$\nabla_{\boldsymbol{\Sigma}}$ (diag)	9.3	9.0	9.6	9.7	230.5	224.9	0.77	0.68
LLR+$\nabla_{\boldsymbol{\mu}}$+$\nabla_{\boldsymbol{\Sigma}}$ (block)	9.4	9.4	9.7	9.7	230.3	226.6	0.76	0.75

Table 3: SVM error rates and performance of LLR kernels on artificial data

A number of configurations of generative kernel and score-space were examined. Three different LLR score-spaces were examined, LLR, LLR + $\nabla_{\boldsymbol{\mu}}$ (LLR term plus derivatives of the LLR with respect to the means of both classes A and B) and LLR + $\nabla_{\boldsymbol{\mu}}$ + $\nabla_{\boldsymbol{\Sigma}}$ (including derivatives with respect to covariances). Initially, the normalisation term $\boldsymbol{G}$ was approximated by a diagonal matrix. For all score-spaces, MM training was used to update the means and covariances; component priors were kept fixed at the ML-estimates. Table 3 shows the error rates, objective function values and margin sizes. As required by the MM training algorithm, the value of the objective function decreased for all score-spaces. However, the change in margin was dependent on the score-space. For example, despite a significant drop in the value of the objective function for the LLR score-space, the margin also fell. This occurred because, as shown in figure 5, the MM algorithm shifted the distributions away from the underlying data, moving the decision surface. This reduced the margin but significantly increased separability of the data resulting in a decrease in test error from 22.8% to 10.4%.

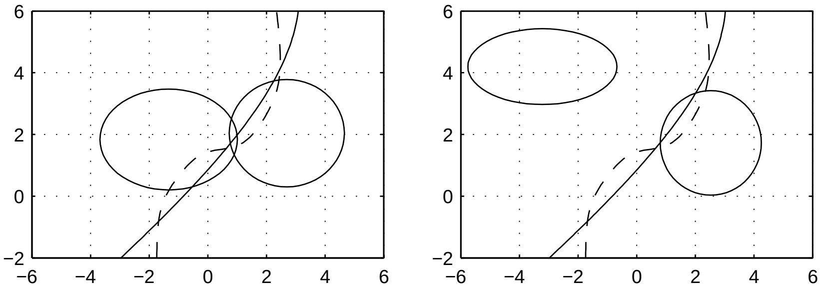

Figure 6: Decision boundary (a) before and (b) after parameter optimisation; LLR+$\nabla_{\boldsymbol{\mu}} + \nabla_{\boldsymbol{\Sigma}}$ score-space

Increasing the complexity of the score-space reduced the initial values of both the objective function and the test error. Accordingly, the scope for gains by the MM algorithm was significantly smaller. As shown in figure 6, the powerful LLR+$\nabla_{\boldsymbol{\mu}}$+$\nabla_{\boldsymbol{\Sigma}}$ score-space initially positions the decision boundary close to the Bayes decision boundary. Thus, although MM training significantly changes the kernel position, the change in decision boundary, and hence change in test error, is negligible for this simple task. With the exception of the LLR+$\nabla_{\boldsymbol{\mu}}$+$\nabla_{\boldsymbol{\Sigma}}$ score-space, the 2-component systems achieved training and test error rates equal to the Bayes error after MM training. The 2-component LLR+$\nabla_{\boldsymbol{\mu}}$+$\nabla_{\boldsymbol{\Sigma}}$ score-space is too powerful for the task, resulting in over-training. The MM algorithm decreased the test error for all score-spaces.

Computing the normalisation term using the block-diagonal representation of $\boldsymbol{G}$ yielded only minor improvements in test error. However, the additional whitening terms significantly reduced the number of iterations required for convergence. For example, for the LLR + $\nabla_{\boldsymbol{\mu}}$ score-space, the number of iterations required halved, from 65 to 36. Compared to the diagonal approximation, margins were also slightly higher. Increasing the complexity of the score-space and kernel reduced the test error.

	Training (%)		**Test (%)**		**Objective**		**Margin**	
C	**Initial**	**Final**	**Initial**	**Final**	**Initial**	**Final**	**Initial**	**Final**
1-Component								
0.1	10.3	10.1	10.6	10.5	27.99	27.39	1.18	1.35
1	10.4	10.5	10.4	10.3	263.6	262.8	0.95	1.16
10	10.5	10.5	10.3	10.3	2615.2	2614.6	0.94	1.12
100	10.5	10.5	10.3	10.3	26132.0	26131.8	0.99	0.93
2-Component								
0.1	9.3	9.0	9.5	9.7	24.25	22.24	1.50	1.61
1	9.3	9.0	9.6	9.7	230.5	224.9	0.77	0.68
10	9.0	9.2	9.6	10.0	2264.1	2137.6	0.51	0.37
100	9.3	9.2	9.6	10.0	22501.3	21351.5	0.19	0.17

Table 4: SVM error rates for LLR + $\nabla_{\boldsymbol{\mu}} + \nabla_{\boldsymbol{\Sigma}}$ score-space with varying C

Experiments were performed to determine the effect of the SVM regularisation parameter C on MM-trained classifiers; results are shown in table 4. Although performance was relatively unaffected by changes in C, as C decreased the number of iterations required for convergence increased significantly (34 iterations were required for $C = 10$, whereas 1,084 were required for $C = 0.1$).

5.2 Deterding data

The Deterding dataset [6] consists of the steady-state portion of 11 vowels in British English, spoken in the context of h*d. The recorded speech samples were low-pass filtered at 4.7KHz before being digitised at 10KHz with a 12-bit resolution. The steady-state portion of the vowel in each utterance was partitioned into six 512 Hamming windowed segments. Linear predictive analysis was then performed; linear prediction reflection coefficients were calculated and used to generate 10 log area parameters; these parameters were recorded and constitute a 10-dimensional input space. The dataset is partitioned into a training set of 528 examples and a test set of 462 examples.

Since the MM training algorithm only trains decision boundaries between two classes, a higher level classification scheme was used. In this report a simple majority voting scheme was used. If two classes, $\omega_a < \omega_b$, were tied after voting, a casting vote was given to the classifier ω_a-vs-ω_b. If more classes were tied, the tie was resolved using MLE GMMs.

		Training (%)		Test (%)	
Classifier	**Components**	**Initial**	**Final**	**Initial**	**Final**
GMM (MLE)	1	40.0		55.8	
GMM (MLE)	2	27.7		45.2	
SVM (LLR)	1	38.1	1.9	58.0	47.4
SVM (LLR)	2	26.3	0.8	48.5	38.8
SVM (LLR + $\nabla\boldsymbol{\mu}$)	1	10.6	1.0	46.3	48.1

Table 5: Error rates of LLR kernels on Deterding data

Baseline GMM classifiers were trained using MLE. Insufficient data resulted in over-training for all classifiers. Despite having more parameters than the GMM classifiers (which had tied covariances) and being highly over-trained (with training errors $< 2\%$), the LLR score-spaces yielded significantly better results. The relatively simple LLR score-space with 2-components yielded an impressive error rate of just 38.8%. State-of-the-art performance on this task, using a highly tuned system, is about 30% [26, 14]. Increasing the complexity of the 1-component score-space did not decrease the error. This is due to both the limited ability of the single component to model the data and severe overtraining.

6 Conclusions

A method of maximum margin training for generative kernels has been described. This explicitly tunes the kernel parameters to the score-space and kernel used. Preliminary results demonstrate the effect of MM training on GMM parameters and show significant reductions in error. The results on the Deterding data, despite over-training, indicate the power of MM training and contradict Ayat's findings that explicit maximisation of the margin does not improve generalisation error [2]. This difference may be due to the greater complexity of LLR kernels compared to the kernels considered by Ayat. Although the algorithm outlined in this report can be applied to any kernel, future work will concentrate on extending the MM implementation to more complex generative models such as HMMs. This will allow dynamic data, such as speech, to be classified.

Acknowledgements

M. Layton would like to thank the Schiff Foundation for funding.

References

[1] N.E. Ayat, M. Cheriet, and C.Y. Suen. Empirical error based optimization of SVM kernels. Application to digit image recognition. In *8th International Workshop on Frontiers of Handwriting Recognition (IWFHR'8)*, pages 105–110, Niagara-on-the-Lake, CA, August 2002.

[2] N.E. Ayat, M. Cheriet, and C.Y. Suen. Optimization of the SVM kernels using an empirical error minimization scheme. In *Lecture Notes in Computer Science*, volume 2388, pages 354–369. Springer Verlag, 2002.

[3] L.R. Bahl, P. Brown, P. de Souza, and R. Mercer. Maximum mutual information estimation of hidden markov model parameters for speech recognition. ICASSP86, Tokyo, 1986.

[4] I. Bazzi and D. Katabi. Using support vector machines for spoken digit recognition. In *6th International Conference on Spoken Language Processing*, Beijing, China, October 2000.

[5] T. de Bie, G.R.G. Lanckriet, and N. Cristianini. Convex tuning of the soft margin parameter. Technical Report UCB/CSD-03-1289, Computer Science Division (EECS), University of California, Berkeley, November 2003.

[6] C.L. Blake and C.J. Merz. UCI repository of machine learning databases. Technical report, University of California, Invine, School of Information and Computer Science, 1998.

[7] O. Bousquet and D.J.L. Herrmann. On the complexity of learning the kernel matrix. In *Advances in Neural Information Processing Systems*, 2002.

[8] R. Burbidge. Adaptive kernels for support vector classification. In *Proceedings of the MSRI Workshop on non-linear estimation and classification*, Lecture notes in Computer Science, Berkeley, Ca., March 2002. Springer-Verlag.

[9] O. Chapelle and V. Vapnik. Model selection for support vector machines. In *Advances in Neural Information Processing Systems*, 1999.

[10] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support sector machines. In *Advances in Neural Information Processing Systems*, 2001.

[11] N. Cristianini, C. Campbell, and J. Shawe-Taylor. Dynamically adapting kernels in support vector machines. Technical Report NC2-TR-1998-017, Royal Holloway, University of London, May 1998.

[12] N. Cristianini, C. Campbell, and J. Shawe-Taylor. Dynamically adapting kernels in support vector machines. In *Advances in Neural Information Processing Systems*, 1998.

[13] N. Cristianini and J. Shawe-Taylor. *Support Vector Machines and other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[14] A. Ganapathiraju, J. Hamaker, and J. Picone. Support Vector Machines for speech recognition. In *International Conference on Spoken Language Processing*, 1998.

[15] A. Ganapathiraju, J. Hamaker, and J. Picone. Hybrid SVM/HMM architectures for speech recognition. In *International Conference on Spoken Language Processing*, 2000.

[16] Y. Grandvalet and S. Canu. Adaptive scaling for feature selection in SVMs. In *Advances in Neural Information Processing Systems*, 2002.

[17] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice-Hall International, 2nd edition, 1999.

[18] T. Jaakkola and D. Hausser. Exploiting generative models in disciminative classifiers. In S.A. Solla and D.A. Cohn, editors, *Advances in Neural Information Processing Systems 11*, pages 487–493. MIT Press, 1999.

[19] T. Joachims. Making large-scale SVM learning practical. In B. Scholkopf, C. Burges and A. Smola, editor, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1999.

[20] J. Kandola, J. Shawe-Taylor, and N. Cristianini. Optimizing kernel alignment over combinations of kernels. *Journal of Machine Learning Research*, 2002.

[21] A. Kowalczyk. Maximum margin perceptron. In A.J. Smolai, P.L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 75–113. MIT Press, 2000.

[22] N. Oliver, B. Schölkopf, and A.J. Smola. Natural regularization from generative models. In A.J. Smola, P.L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 51–60. MIT Press, 2000.

[23] B. Schölkopf and A.J. Smola. *Learning with kernels*. MIT Press, 2002.

[24] H. Shimodaira, K.-I. Noma, M. Nakai, and S. Sagayama. Dynamic time-alignment kernel in support vector machine. In *Advances in Neural Information Processing Systems*, 2001.

[25] N. Smith and M. Gales. Speech recognition using SVMs. In *Advances in Neural Information Processing Systems*, 2001.

[26] N.D. Smith. *Using Augmented Statistical Models and Score Spaces for Classification*. PhD thesis, University of Cambridge, September 2003.

[27] N.D. Smith and M.J.F. Gales. Using SVMs to classify variable length speech patterns. Technical Report CUED/F-INFENG/TR.412, Department of Engineering, University of Cambridge, April 2002.

[28] V.N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, 1998.

A Derivatives of the dual SVM objective function

The dual objective function is given by,

$$\max_{\alpha} W(\boldsymbol{\theta}, \boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(\boldsymbol{\theta}, \boldsymbol{x}_i, \boldsymbol{x}_j) \tag{42}$$

The gradient descent optimisation of kernel parameters requires the gradient of the dual objective function. This is given by the derivative of the kernel summed over the support vectors,

$$\nabla_{\boldsymbol{\theta}} W(\boldsymbol{\alpha}, \boldsymbol{\theta}) = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \nabla_{\boldsymbol{\theta}} K(\boldsymbol{\theta}, \boldsymbol{x}_i, \boldsymbol{x}_j) \tag{43}$$

where,

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}_k} K(\boldsymbol{\theta}, \boldsymbol{x}_i, \boldsymbol{x}_j) &= \Big[\nabla_{\boldsymbol{\theta}_k} \boldsymbol{\phi}(\boldsymbol{x}_i, \boldsymbol{\theta})\Big]^T \boldsymbol{G}^{-1} \boldsymbol{\phi}(\boldsymbol{x}_j, \boldsymbol{\theta}) \\
&\quad + \boldsymbol{\phi}(\boldsymbol{x}_i, \boldsymbol{\theta})^T \Big[\nabla_{\boldsymbol{\theta}_k} \boldsymbol{G}^{-1}\Big] \boldsymbol{\phi}(\boldsymbol{x}_j, \boldsymbol{\theta}) \\
&\quad + \boldsymbol{\phi}(\boldsymbol{x}_i, \boldsymbol{\theta})^T \boldsymbol{G}^{-1} \Big[\nabla_{\boldsymbol{\theta}_k} \boldsymbol{\phi}(\boldsymbol{x}_j, \boldsymbol{\theta})\Big]
\end{aligned} \tag{44}$$

The derivative of the score-space with respect to the parameter $\boldsymbol{\theta}_k$ is given by,

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}_k}\big[\boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{\theta})\big]_j &= \nabla_{\boldsymbol{\theta}_k}\Big(\nabla_{\boldsymbol{\theta}_j}\ln p(\boldsymbol{x}|\boldsymbol{\theta})\Big) \\
&= \nabla_{\boldsymbol{\theta}_k}\left(\frac{1}{p(\boldsymbol{x}|\boldsymbol{\theta})}\nabla_{\boldsymbol{\theta}_j}p(\boldsymbol{x}|\boldsymbol{\theta})\right) \\
&= \frac{1}{p(\boldsymbol{x}|\boldsymbol{\theta})^2}\Big\{p(\boldsymbol{x}|\boldsymbol{\theta})\Big(\nabla^2_{\boldsymbol{\theta}_j,\boldsymbol{\theta}_k}p(\boldsymbol{x}|\boldsymbol{\theta})\Big) - \Big(\nabla_{\boldsymbol{\theta}_j}p(\boldsymbol{x}|\boldsymbol{\theta})\Big)\Big(\nabla_{\boldsymbol{\theta}_k}p(\boldsymbol{x}|\boldsymbol{\theta})\Big)\Big\} \\
&= \frac{1}{p(\boldsymbol{x}|\boldsymbol{\theta})}\Big(\nabla^2_{\boldsymbol{\theta}_j,\boldsymbol{\theta}_k}p(\boldsymbol{x}|\boldsymbol{\theta})\Big) - \frac{1}{p(\boldsymbol{x}|\boldsymbol{\theta})}\Big(\nabla_{\boldsymbol{\theta}_j}p(\boldsymbol{x}|\boldsymbol{\theta})\Big)\frac{1}{p(\boldsymbol{x}|\boldsymbol{\theta})}\Big(\nabla_{\boldsymbol{\theta}_k}p(\boldsymbol{x}|\boldsymbol{\theta})\Big) \\
&= \frac{1}{p(\boldsymbol{x}|\boldsymbol{\theta})}\Big(\nabla^2_{\boldsymbol{\theta}_j,\boldsymbol{\theta}_k}p(\boldsymbol{x}|\boldsymbol{\theta})\Big) - \Big(\nabla_{\boldsymbol{\theta}_j}\ln p(\boldsymbol{x}|\boldsymbol{\theta})\Big)\Big(\nabla_{\boldsymbol{\theta}_k}\ln p(\boldsymbol{x}|\boldsymbol{\theta})\Big) \\
&= \frac{1}{p(\boldsymbol{x}|\boldsymbol{\theta})}\Big(\nabla^2_{\boldsymbol{\theta}_j,\boldsymbol{\theta}_k}p(\boldsymbol{x}|\boldsymbol{\theta})\Big) - \Big(\nabla_{\boldsymbol{\theta}_j}\boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{\theta})\Big)\Big(\nabla_{\boldsymbol{\theta}_k}\boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{\theta})\Big) \qquad (45)
\end{aligned}$$

The derivative of the normalisation term $\boldsymbol{G}^{-1}$ with respect to the parameter $\boldsymbol{\theta}_k$ is given by,

$$\nabla_{\boldsymbol{\theta}_k}\Big(\boldsymbol{G}^{-1}\Big) = -\boldsymbol{G}^{-1}\Big(\nabla_{\boldsymbol{\theta}_k}\boldsymbol{G}\Big)\boldsymbol{G}^{-1}$$

Since,

$$\begin{aligned}
\boldsymbol{G} &= \mathcal{E}\Big\{\Big(\boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{\theta}) - \boldsymbol{\mu}_{\boldsymbol{\theta}}\Big)\Big(\boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{\theta}) - \boldsymbol{\mu}_{\boldsymbol{\theta}}\Big)^T\Big\} \\
&= \mathcal{E}\Big\{\boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{\theta})\boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{\theta})^T\Big\} - \mathcal{E}\Big\{\boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{\theta})\Big\}\mathcal{E}\Big\{\boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{\theta})\Big\}^T
\end{aligned}$$

The derivative of $\boldsymbol{G}$ is given by,

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}_k}\boldsymbol{G} &= 2\mathcal{E}\Big\{\Big(\boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{\theta})\Big)\Big(\nabla_{\boldsymbol{\theta}_k}\boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{\theta})\Big)^T\Big\} \\
&\qquad - 2\mathcal{E}\Big\{\boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{\theta})\Big\}\mathcal{E}\Big\{\nabla_{\boldsymbol{\theta}_k}\boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{\theta})\Big\}^T \\
&= 2\mathcal{E}\Big\{\Big(\boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{\theta})\Big)\Big(\nabla_{\boldsymbol{\theta}_k}\boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{\theta})\Big)^T\Big\} \\
&\qquad - 2\mathcal{E}\Big\{\boldsymbol{\mu}_{\boldsymbol{\phi}}\Big(\nabla_{\boldsymbol{\theta}_k}\boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{\theta})\Big)^T\Big\} \\
&= 2\mathcal{E}\Big\{\Big(\boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{\theta}) - \boldsymbol{\mu}_{\boldsymbol{\phi}}\Big)\Big(\nabla_{\boldsymbol{\theta}_k}\boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{\theta})\Big)^T\Big\}
\end{aligned}$$

Therefore,

$$\nabla_{\boldsymbol{\theta}_k}\Big(\boldsymbol{G}^{-1}\Big) = -2\boldsymbol{G}^{-1}\mathcal{E}\Big\{\Big(\boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{\theta}) - \boldsymbol{\mu}_{\boldsymbol{\phi}}\Big)\Big(\nabla_{\boldsymbol{\theta}_k}\boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{\theta})\Big)^T\Big\}\boldsymbol{G}^{-1} \qquad (46)$$

B Derivatives of the GMM LLR score-space

Since classes A and B are assumed to be independent, derivatives of the score-space with respect to parameters $\boldsymbol{\theta}_A$ and $\boldsymbol{\theta}_B$ are given by,

$$\nabla_{\boldsymbol{\theta}_A}\boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{\theta}) = \begin{bmatrix} \nabla_{\boldsymbol{\theta}_A} \ln p(\boldsymbol{x}|\boldsymbol{\theta}) \\ \nabla^2_{\boldsymbol{\mu}_A,\boldsymbol{\theta}_A} \ln p(\boldsymbol{x}|\boldsymbol{\theta}) \\ \nabla^2_{\boldsymbol{\Sigma}_A,\boldsymbol{\theta}_A} \ln p(\boldsymbol{x}|\boldsymbol{\theta}) \\ 0 \\ 0 \end{bmatrix} \tag{47}$$

and,

$$\nabla_{\boldsymbol{\theta}_B}\boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{\theta}) = \begin{bmatrix} \nabla_{\boldsymbol{\theta}_B} \ln p(\boldsymbol{x}|\boldsymbol{\theta}) \\ 0 \\ 0 \\ \nabla^2_{\boldsymbol{\mu}_B,\boldsymbol{\theta}_B} \ln p(\boldsymbol{x}|\boldsymbol{\theta}) \\ \nabla^2_{\boldsymbol{\Sigma}_B,\boldsymbol{\theta}_B} \ln p(\boldsymbol{x}|\boldsymbol{\theta}) \end{bmatrix} \tag{48}$$

For each class,

$$\nabla_{\boldsymbol{\mu}_j} = \gamma_j(\boldsymbol{x})\Big[\boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_j)\Big] \tag{49}$$

$$\nabla_{\boldsymbol{\Sigma}_j} = \frac{\gamma_j(\boldsymbol{x})}{2}\mathrm{diag}\Big[-\boldsymbol{\Sigma}_j^{-1} + \Big(\boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_j)\Big)\Big(\boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_j)\Big)^T\Big] \tag{50}$$

The second-derivatives are given by,

$$\nabla^2_{\boldsymbol{\theta}_j,\boldsymbol{\theta}_k} l(\boldsymbol{x}|\boldsymbol{\theta}) = \begin{cases} -\gamma_j\gamma_k\Big[\nabla_{\boldsymbol{\theta}_j} l_j(\boldsymbol{x}|\boldsymbol{\theta})\Big]\Big[\nabla_{\boldsymbol{\theta}_k} l_k(\boldsymbol{x}|\boldsymbol{\theta})\Big]^T & \text{if } j \neq k \\ \gamma_j\Big\{\nabla^2_{\boldsymbol{\theta}_j^2} l_j(\boldsymbol{x}|\boldsymbol{\theta}) + \Big[1-\gamma_j\Big]\Big[\nabla_{\boldsymbol{\theta}_j} l(\boldsymbol{x}|\boldsymbol{\theta})\Big]\Big[\nabla_{\boldsymbol{\theta}_j} l(\boldsymbol{x}|\boldsymbol{\theta})\Big]^T\Big\} & \text{if } j = k \end{cases}$$

where (Diag[$\cdot$] transforms a d-dimensional vector into a $d \times d$ diagonal matrix),

$$\begin{aligned} \nabla^2_{\boldsymbol{\mu}_j,\boldsymbol{\mu}_j} l_j(\boldsymbol{x}|\boldsymbol{\theta}) &= -\boldsymbol{\Sigma}_j^{-1} \\ \nabla^2_{\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j} l_j(\boldsymbol{x}|\boldsymbol{\theta}) &= \mathrm{Diag}\Big[-\boldsymbol{\Sigma}_j^{-1}\boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_j)\Big] \\ \nabla^2_{\boldsymbol{\Sigma}_j,\boldsymbol{\Sigma}_j} l_j(\boldsymbol{x}|\boldsymbol{\theta}) &= \frac{1}{2}\boldsymbol{\Sigma}_j^{-1}\boldsymbol{\Sigma}_j^{-1} - \mathrm{Diag}\Big\{\mathrm{diag}\Big[\Big(\boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_j)\Big)\Big(\boldsymbol{\Sigma}_j^{-1}\boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_j)\Big)^T\Big]\Big\} \end{aligned}$$

and,

$$\begin{aligned} \gamma_j(\boldsymbol{x}) &= \frac{c_j\mathcal{N}(\boldsymbol{x};\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j)}{\sum_{j=1}^m c_j\mathcal{N}(\boldsymbol{x};\boldsymbol{\mu}_j,\boldsymbol{\Sigma})_j)} \\ l_j(\boldsymbol{x},\boldsymbol{\theta}) &= \ln\mathcal{N}(\boldsymbol{x}|\boldsymbol{\theta}_j) \end{aligned}$$

Between-class derivatives, for example, $\nabla_{\boldsymbol{\theta}_{Aj},\boldsymbol{\theta}_{Bk}} \ln p(\boldsymbol{x}|\boldsymbol{\theta})$, are zero.