

**CAMBRIDGE UNIVERSITY**  
ENGINEERING DEPARTMENT

**Maximum Margin Training of  
Generative Kernels**

M.I. Layton  
First year report  
August 2004

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

E-mail: [m1362@eng.cam.ac.uk](mailto:m1362@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 report 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,  $\theta$ , as quantities that are fixed but unknown. Given a training set of independent and identically distributed examples,  $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n)$ , with labels,  $Y = (y_1, \dots, y_n)$ , MLE chooses parameter values that maximise the joint probability of the training set. The MLE objective function is given by,

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

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) [4]. 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,

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

where  $C$  is the number of classes.

Another popular and successful discriminative classifier is the Support Vector Machine (SVM) [42]. 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* [39, 40]. These are a generalised form of Fisher kernel [29]. For generative kernels, kernel parameters are learnt directly from the data, typically using Maximum Likelihood Estimation (MLE) [39, 40]. Maximum Mutual Information Estimation (MMIE) [4], 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 [32], limited research has been conducted for SVMs. Early SVM parameter optimisation methods were aimed at finding parameters for specific kernels, such as [10, 15, 16] for GRBF kernels. Recently, however, more general kernel parameter optimisation schemes have been introduced. For example, Chapelle [11] 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 [12]. Later, Ayat [3, 2] 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 [25] 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 [8], Bousquet minimised the Rademacher Complexity of the kernel matrix, and in [31] Kandola varied the feature-space to maximise a measure of agreement between the kernel and data.

A further problem of SVMs is that they are intrinsically binary classifiers. Real-world problems typically have many classes – ranging from relatively few (such as the 3-class Iris data set) to several thousand (when discriminating between context-dependent models in Large Vocabulary Continuous Speech Recognition (LVCSR) systems). Multiclass SVM systems typically take one of two approaches: decomposition into binary problems; or extending the SVM framework to handle multiple classes (all-in-one methods). Examples of binary decomposition techniques are 1-versus-all, 1-versus-1 (pairwise) and Error Correcting Output Codes [20, 21, 33, 14]. All-in-one methods have been considered by Weston [43], Bredensteiner [9] and Hsu [28]. Multiclass methods are discussed in more detail in section 3.

In this report, techniques for embedding generative models, such as Gaussian mixture models (GMMs), within SVM kernels are reviewed (section 4). 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 (section 5). 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 [17]. For separable data, SVMs yield a training error of 0 and minimise the VC dimension. Vapnik [42] 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).

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

$$\mathbf{w}^T \mathbf{x} + b = 0 \quad (3)$$

where  $\mathbf{w}$  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  $\rho$ . The hyperplane that

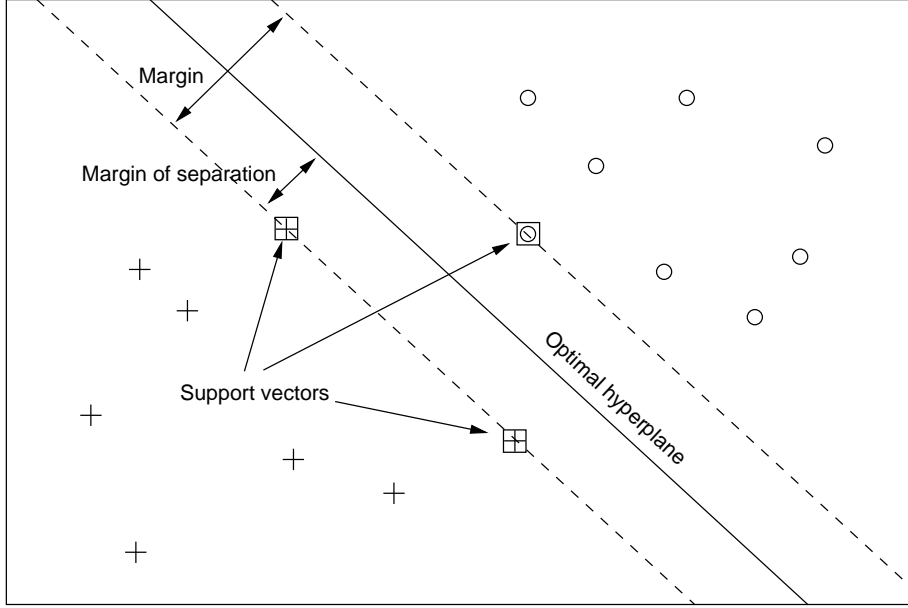


Figure 1: The optimal hyperplane for separable data

maximises this separation is referred to as the *optimal* or *maximum-margin* hyperplane (figure 1); the optimal hyperplane is unique [42]. The geometric distance between the two classes, herein referred to as the margin of the SVM, is given by,

$$\text{Margin} = 2\rho = \frac{2}{|\mathbf{w}|} \quad (4)$$

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

$$\min_{\mathbf{w}, b} \frac{1}{2}(\mathbf{w} \cdot \mathbf{w}) \quad (5)$$

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

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

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,

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

At optimality, only examples that satisfy  $\mathbf{w}^T \mathbf{x}_i + b = 1$  have  $\alpha_i > 0$ . These examples are known as the *support vectors* [42] 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,

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

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 ( $\mathbf{x}_i \cdot \mathbf{x}_j$ ) which are trivial to compute.

## 2.2 Soft-margin SVMs

In general, it is 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,  $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1$ . 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,

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

Margin constraints may be violated in one of two ways:

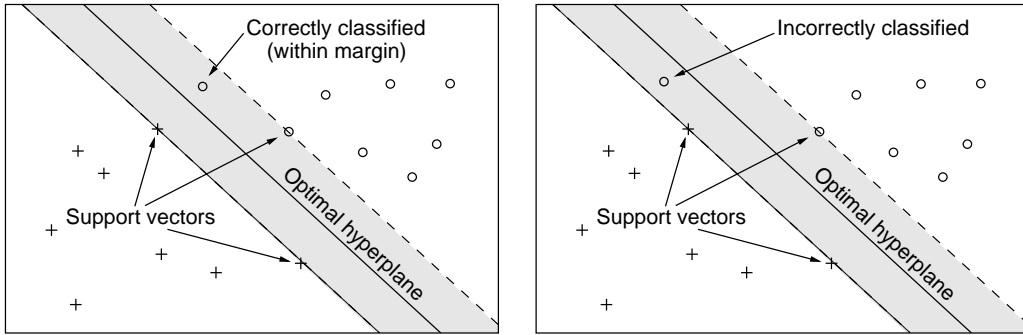


Figure 2: Example falls (a) within margin of separation; (b) on the wrong side of the decision hyperplane

- 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 reformulated as,

$$\min_{\mathbf{w}, b} \frac{1}{2}(\mathbf{w} \cdot \mathbf{w}) + C \sum_{i=1}^n \epsilon_i \quad (10)$$

$C$  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,

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

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,  $\epsilon_i$ , nor their Lagrange multipliers appear in the dual problem. However, unlike the separable case, the upper limit on the Lagrange multipliers,  $\alpha_i$ , limits the influence of individual examples (which may be outliers).

## 2.3 Kernels

For non-linearly separable data, Cover's theorem [26] states that examples may be made linearly separable with a high probability given a non-linear transformation,  $\phi$ , from input-space to a *feature-space* of sufficient dimensionality (figure 3). Using this mapping, the kernelised form of the dual-objective (11) is

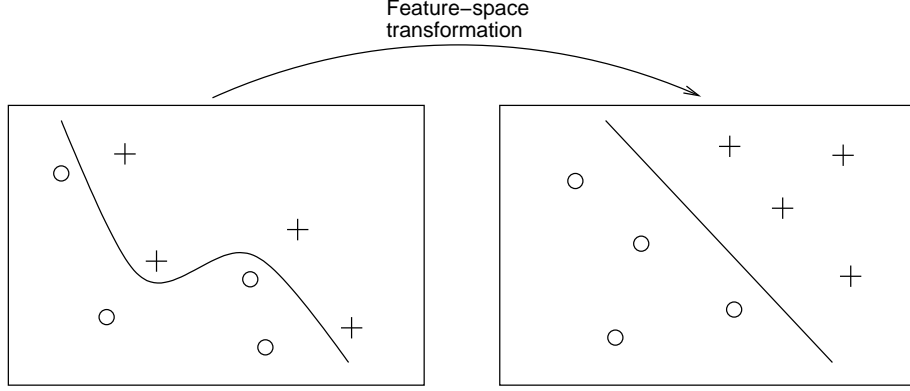


Figure 3: Use of a feature-space to make data linearly separable

defined,

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

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

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

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 [42, 17]. This states that the kernel may be any symmetric non-linear function whose kernel matrix,

$$\mathbf{K} = \{K(\theta, \mathbf{x}_i, \mathbf{x}_j)\}_{(i,j)=1}^n \quad (14)$$

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

$$K(\theta, \mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^{\infty} a_k z_k(\mathbf{x}_i) z_k(\mathbf{x}_j) \quad (15)$$

into some unknown feature-space ( $z_1(\mathbf{x}), \dots, z_k(\mathbf{x}), \dots$ ) 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,

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

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,  $\phi(\cdot) \cdot \phi(\cdot)$ , and not explicitly on  $\phi(\cdot)$ , 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	$K(\mathbf{x}_i, \mathbf{x}_j)$	Parameters
Inner-product	$\mathbf{x}_i \cdot \mathbf{x}_j$	
Inhomogeneous polynomial	$(\mathbf{x}_i \cdot \mathbf{x}_j + c)^p$	Power ( $p \in \mathbb{N}$ ), $c > 0$
Sigmoid	$\tanh(\kappa(\mathbf{x}_i \cdot \mathbf{x}_j) + \theta)$	$\kappa > 0, \theta < 0$
GRBF	$\exp\left(-\frac{\ \mathbf{x}_i - \mathbf{x}_j\ ^2}{2\sigma^2}\right)$	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 [37] (and thus not strictly a SVM kernel), in practice, it has been used successfully.

### 3 Multiclass Support Vector Machines

SVMs, as formulated by Vapnik [42], position a hyperplane to maximally separate observations from two competing classes. However, many practical problems have multiple classes and thus cannot be directly classified using SVMs. Instead, a more complex setup is required. This is typically achieved using one of two approaches: decomposition into binary problems (and then training separate SVMs on each problem); or extending the SVM framework to directly handle multiple classes.

Decomposition is a popular technique and consists of splitting the multiclass problem into a series of binary problems. A classifier is then trained on each binary problem. Finally, using a predefined algorithm, the results are combined to yield a class label. Common decomposition techniques are 1-versus-all, 1-versus-1 (pairwise), and Error Correcting Output Codes (ECOCs). The alternative approach is to extend the SVM framework to directly handle multiple classes. This typically involves redefining the SVM loss function to include between-class dependencies. The resulting classifiers are sufficiently similar to SVMs that standard SVM techniques such as SMO [35] and chunking [30]) may be adapted to the new system.

#### 3.1 1-versus-all

1-versus-all decomposition is one of the earliest (and also the most popular) decomposition methods for SVMs. For a  $k$ -class problem,  $k$  binary classifiers are constructed. Each classifier is trained to separate examples of one class from examples of all other classes. This method is popular since the number of classifiers scales linearly with the number of classes, making it computationally efficient.

To classify a test example, a result is obtained for each binary classifier. These results are then combined using a voting algorithm. The example is labelled as the class that receives the most votes. In practice, however, the vote may be tied. Since a single class label is required, the tie must be broken. This is best done by backing-off to an alternative classifier. However, for convenience the tie is commonly broken by comparing the value of the decision function for each class – the class with the largest decision function is chosen. This assumes that the values of the decision function for different classifiers are comparable. However, since the SVM output scale depends on only a small fraction of the training data (the support vectors), the scale is not robust and so reliable comparisons cannot be made.

#### 3.2 1-versus-1 (pairwise) decomposition

Another popular decomposition technique is pairwise decomposition. This requires classifiers to be trained between every possible pair of classes – for a  $k$ -class problem, this results in  $k(k - 1)/2$  classifiers. This

technique has limited practical use since the number of classifiers scales quadratically with the number of classes. For systems with a large number of classes, for example, phonemes in speech recognition systems, it quickly becomes impractical.

Similarly to 1-versus-all systems, test examples are classified by obtaining a result from each binary classifier. A simple majority voting scheme is then used to determine the class label. When votes are tied between two classes,  $A$  and  $B$ , a casting vote may be given to the classifier  $A$ -vs- $B$ . When more classes are tied, an alternative voting scheme may be used or an alternative classifier used.

Unfortunately, this system suffers from the problem of 'incompetent' classifiers. This occurs when the class of the test example does not match the classes used when training the classifier, for example, presenting an example from class  $A$  to a classifier trained to separate classes  $B$  and  $C$ . Depending on the nature of the data set, this may generate a significant bias towards specific classes. Although positive voting is common, Cutzu [18] asserts that, providing individual classifiers are well trained (and so expected to make few errors), negative voting schemes can minimise the effect of incompetent classifiers. This is because if, for an example  $x$ , the classifier  $A$ -vs- $B$  outputs the label  $A$ , one cannot conclude that  $x \in A$  (since  $x$  may belong to neither  $A$  nor  $B$ ). However, since the classifier is assumed to output the correct label (given an example from classes  $A$  or  $B$ ) one can conclude that  $x \notin B$  (otherwise  $B$  would have been output). Alternatively, Angulo, Parra and Catalá [1] attempted to minimise this problem by defining a SVM that, in addition to labelling an example as being from either of the two training classes, can emit an output indicating that the test example belongs to neither class (and thus contributes no votes).

If probabilistic outputs can be derived from the SVMs, more sophisticated combination techniques may be used. The most common such technique for generating probabilistic outputs was introduced by Platt [36] and involves fitting a probability distribution to the output scale of the SVM. Unfortunately, since the output scale of SVMs is dependent on only the support vectors, these estimates may be severely distorted by outliers.

### 3.3 Error correcting output codes

Error correcting output codes (ECOCs) [21, 20, 33] are an alternative decomposition technique. Each class  $k$  is labelled with a fixed length multi-bit output code. Although, in general, output codes for a  $k$ -class problem can be represented by  $\log_2 k$  bits, Diettrich and Bakiri proposed adding redundant bits to allow error correction. Given all training examples, separate binary classifiers can be trained to recognise each bit of the output codes.

For each test example, the classifiers corresponding to each bit of the output code are evaluated. The resulting output code is then compared to output codes of each of the classes and the class with the nearest match is chosen (redundant bits allow a degree of error correction). When finding the nearest match, a common metric is the Hamming Distance. This measures the number of bit differences between two code words.

Unfortunately, the assignment of output codes to classes can affect the performance. This is because different arrangements of codes change the single-bit classification problems, potentially altering their complexity. Additionally, the use of 'optimal' codes is not practical since, as Crammer and Singer [14] proved, the problem of finding optimal output codes is NP-complete. For this reason, codes are typically chosen using either expert knowledge or through experimentation. An alternative method, proposed by Crammer and Singer [14], involved extending ECOCs to use continuous output codes. A set of 'good' codes (based upon the minimising the complexity of the binary classifiers) can then be learnt using a quadratic optimisation procedure.

### 3.4 All-in-one

These techniques aim to generalise the SVM framework to multiple classes. This is done by generalising the SVM loss function. For example, Hsu and Lin [28] show that many all-in-one formulations may be

reduced to the following primal loss function,

$$\begin{aligned} \min_{\mathbf{w}, b, \epsilon} \frac{1}{2} \sum_{m=1}^k \mathbf{w}_m^T \mathbf{w}_m + C \sum_{i=1}^n \sum_{\substack{m=1 \\ m \neq y_i}}^k \epsilon_i^m & \quad (17) \\ \mathbf{w}_{y_i}^T \phi(\mathbf{x}_i) + b_{y_i} \geq \mathbf{w}_m^T \phi(\mathbf{x}_i) + b_m + 2 - \epsilon_i^m & \\ \epsilon_i^m \geq 0 & \end{aligned}$$

where  $k$  is the number of classes,  $n$  is the number of training examples.  $\mathbf{w}_m$  and  $b_m$  are the weight vector and bias of the hyperplane for class  $m$ , and  $\epsilon_i^m$  are the slack vectors associated with example  $i$  and class  $m$ .  $C$  is the standard SVM regularisation constant.

Although all-in-one techniques avoid the problems of decomposition techniques (combining results from separate classifiers, and determination of class labels), they suffer from complexity problems. This is because the dual quadratic problem has  $k(l-1)$  variables that must be trained using the whole training data set. This means that all-in-one systems can take significantly longer to train than equivalent pairwise or ECOC systems. Additionally, the benefits obtained from such systems have, to date, been minimal. However, as shown by Hill [27], an all-in-one optimisation can be formulated such that it can be initialised with the results from a set of pairwise classifiers (optimising the decision boundary for each pair of classes without regard for any others). This acts to kick-start the optimisation and can significantly reduce the overall training time.

## 4 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 [5, 38, 23, 24]. An alternative approach, first introduced by Jaakkola and Hausser [29] and later extended in [34, 39, 40], is to train generative models on the data and then define kernels based upon them. These kernels are known as generative kernels.

### 4.1 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,

$$\phi(\mathbf{x}, \boldsymbol{\theta}_A, \boldsymbol{\theta}_B) = \begin{bmatrix} \ln p(\mathbf{x} | \boldsymbol{\theta}_A) \\ \ln p(\mathbf{x} | \boldsymbol{\theta}_B) \end{bmatrix} \quad (18)$$

where,

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

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,

$$\begin{aligned}
\phi(\mathbf{x}, \boldsymbol{\theta}_A, \boldsymbol{\theta}_B) &= \ln \left[ \frac{p(\mathbf{x}|\boldsymbol{\theta}_A)}{p(\mathbf{x}|\boldsymbol{\theta}_B)} \right] \\
&= \ln p(\mathbf{x}|\boldsymbol{\theta}_A) - \ln p(\mathbf{x}|\boldsymbol{\theta}_B) \\
&= \frac{1}{2} \ln \left( \frac{|\boldsymbol{\Sigma}_B|}{|\boldsymbol{\Sigma}_A|} \right) - \frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_A)^T \boldsymbol{\Sigma}_A^{-1} (\mathbf{x} - \boldsymbol{\mu}_A) \\
&\quad + \frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_B)^T \boldsymbol{\Sigma}_B^{-1} (\mathbf{x} - \boldsymbol{\mu}_B)
\end{aligned} \tag{19}$$

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).

## 4.2 Fisher score-space

The LL and LLR feature-spaces are defined using the posterior probabilities of the class-conditional generative models. In [29], 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) [29, 34] is thus defined as,

$$\phi(\mathbf{x}, \boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \ln p(\mathbf{x}|\boldsymbol{\theta}) \tag{20}$$

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

$$K(\boldsymbol{\theta}, \mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i, \boldsymbol{\theta})^T \mathbf{G}^{-1} \phi(\mathbf{x}_j, \boldsymbol{\theta}) \tag{21}$$

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

$$\mathbf{G} = \mathcal{E} \left\{ \left( \phi(\mathbf{x}, \boldsymbol{\theta}) - \boldsymbol{\mu}_{\phi} \right) \left( \phi(\mathbf{x}, \boldsymbol{\theta}) - \boldsymbol{\mu}_{\phi} \right)^T \right\} \tag{22}$$

$$\boldsymbol{\mu}_{\phi} = \mathcal{E} \{ \phi(\mathbf{x}, \boldsymbol{\theta}) \} \tag{23}$$

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

Jaakkola and Haussler's original Fisher score-spaces were based upon a single generative model, trained on examples from both classes. However, when mixture models are used, these score-spaces suffer from wrap-around [39, 41]. 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 [39, 40]. The score-spaces can be further generalised to include higher-order terms as well as the log-likelihood ratio [39, 40].

## 4.3 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 [41]. Independent generative models are trained for each class; the log-likelihood-ratio of these models is then used to generate the score-space,

$$\begin{aligned}
\phi(\mathbf{x}, \boldsymbol{\theta}_A, \boldsymbol{\theta}_B) &= \nabla_{\boldsymbol{\theta}_A, \boldsymbol{\theta}_B} \ln \left( \frac{p(\mathbf{x}|\boldsymbol{\theta}_A)}{p(\mathbf{x}|\boldsymbol{\theta}_B)} \right) \\
&= \begin{bmatrix} \nabla_{\boldsymbol{\theta}_A} \ln p(\mathbf{x}|\boldsymbol{\theta}_A) \\ \nabla_{\boldsymbol{\theta}_B} \ln p(\mathbf{x}|\boldsymbol{\theta}_B) \end{bmatrix}
\end{aligned} \tag{24}$$

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,

$$\phi(\mathbf{x}, \boldsymbol{\theta}_A, \boldsymbol{\theta}_B) = \begin{bmatrix} \ln p(\mathbf{x}|\boldsymbol{\theta}_A) - \ln p(\mathbf{x}|\boldsymbol{\theta}_B) \\ \nabla_{\boldsymbol{\theta}_A} \ln p(\mathbf{x}|\boldsymbol{\theta}_A) \\ \nabla_{\boldsymbol{\theta}_B} \ln p(\mathbf{x}|\boldsymbol{\theta}_B) \end{bmatrix} \quad (25)$$

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

$$K(\boldsymbol{\theta}, \mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i, \boldsymbol{\theta})^T \mathbf{G}^{-1} \phi(\mathbf{x}_j, \boldsymbol{\theta}) \quad (26)$$

$$\mathbf{G} = \mathcal{E} \left\{ \left( \phi(\mathbf{x}, \boldsymbol{\theta}) - \boldsymbol{\mu}_\phi \right) \left( \phi(\mathbf{x}, \boldsymbol{\theta}) - \boldsymbol{\mu}_\phi \right)^T \right\} \quad (27)$$

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

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,

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

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

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

#### 4.4 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 log posterior probability of the GMM is given by,

$$\ln p(\mathbf{x}|\boldsymbol{\theta}) = \ln \left\{ \sum_{j=1}^m c_j \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \right\} \quad (29)$$

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

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

where,

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

and  $j$  is the GMM mixture component. A GMM is specified for each class. The kernel normalisation term is calculated using either equation (27) 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.

## 4.5 HMM-based LLR score-spaces

Although, with sufficient components, GMMs can model static data well, they are unable to model sequences of observations. For such data, Hidden Markov Models (HMMs) may be used. HMMs are finite state machines that change state at every time  $t$ . Every time a state  $j$  is entered, an observation  $\mathbf{x}_t$  is emitted according to the output probability density function associated with that state,  $p(\mathbf{x}_t | s(t) = j, \mathcal{M})$ . The HMM state is not directly observable.

This report uses HMMs with GMM output probability density functions for each state. To limit the number of model parameters, diagonal covariance matrices are used. The log posterior probability of the HMM is given by,

$$\ln p(\mathbf{X} | \mathcal{M}) = \ln \left\{ \sum_{\phi} \prod_{t=1}^T a_{\phi_t \phi_{t-1}} p(\mathbf{x}_t | \phi_t, \mathcal{M}) \right\} \quad (30)$$

where  $a_{ij}$  is the probability of transitioning from state  $i$  to state  $j$ ,  $\phi = (\phi_1, \dots, \phi_T)$  is the current state sequence,  $\Phi$  is the set of all state sequences, and  $p(\mathbf{x}_t | \phi_t, \mathcal{M}) = b_{\phi_t}(\mathbf{x}_t)$  is the output probability function for the state  $\phi_t$ ,

As for the GMM-based score-space, the HMM-based score-space contains the first derivatives of the LLR with respect to the model parameters. These are given by (see appendix C for derivation) [39, 40],

$$\nabla_{\boldsymbol{\mu}_{jm}} \ln p(\mathbf{X} | \mathcal{M}) = \sum_{\tau=1}^T \gamma_{jm}(\tau) \boldsymbol{\Sigma}_{jm}^{-1} (\mathbf{x}_{\tau} - \boldsymbol{\mu}_{jm}) \quad (31)$$

$$\nabla_{\boldsymbol{\Sigma}_{jm}} \ln p(\mathbf{X} | \mathcal{M}) = \sum_{\tau=1}^T \frac{\gamma_{jm}(\tau)}{2} \text{diag} \left[ -\boldsymbol{\Sigma}_{jm}^{-1} + \boldsymbol{\Sigma}_{jm}^{-1} (\mathbf{x} - \boldsymbol{\mu}_{jm}) (\mathbf{x} - \boldsymbol{\mu}_{jm})^T \boldsymbol{\Sigma}_{jm}^{-1} \right] \quad (32)$$

where,

$$\gamma_{jm}(\tau) = p(\phi_{\tau} = \{s_j, m\} | \mathbf{X}, \mathcal{M}) \quad (33)$$

## 5 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 [6], 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.

### 5.1 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 [39, 40] 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 [12]). Instead, the soft-margin dual objective-function (12) is generalised to include kernel parameter optimisation. Referring to the primal SVM in (6) and noting that minimising with respect to  $\mathbf{w}$  is equivalent to minimising with respect to  $\mathbf{w} \cdot \mathbf{x} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} = \sum_{i=1}^n \alpha_i y_i K(\boldsymbol{\theta}, \mathbf{x}_i, \mathbf{x})$  (for the non-linear case), allows us to conclude that this is identical to minimising with respect to  $\boldsymbol{\theta}$ . The dual objective-function thus becomes,

$$\max_{\boldsymbol{\alpha}} \min_{\boldsymbol{\theta}} W(\boldsymbol{\theta}, \boldsymbol{\alpha}) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j K(\boldsymbol{\theta}, \mathbf{x}_i, \mathbf{x}_j) \quad (34)$$

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 ( $\mathbf{G} = \mathbf{I}$ ), 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 can 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  $k$ :
    - $\boldsymbol{\theta}^{(k)} = \arg \min_{\boldsymbol{\theta}} W(\boldsymbol{\theta}; \boldsymbol{\alpha}^{(k-1)})$  (A)
    - $\boldsymbol{\alpha}^{(k)} = \arg \max_{\boldsymbol{\alpha}} W(\boldsymbol{\alpha}; \boldsymbol{\theta}^{(k)})$  (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.

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \eta \nabla_{\boldsymbol{\theta}} W(\boldsymbol{\theta}, \boldsymbol{\alpha}) \Big|_{\boldsymbol{\theta}^{(k)}} \quad (35)$$

where  $\eta$  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  $W$ .

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,  $\eta$ , 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

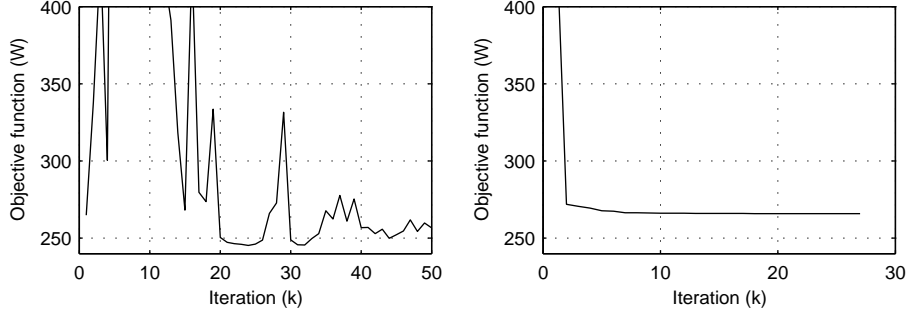


Figure 4: Objective function for a LLR kernel; constant step-size (left) and with back-off

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,

$$\nabla_{\boldsymbol{\theta}} W(\boldsymbol{\alpha}, \boldsymbol{\theta}) = -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \nabla_{\boldsymbol{\theta}} K(\boldsymbol{\theta}, \mathbf{x}_i, \mathbf{x}_j) \quad (36)$$

## 5.2 Kernel gradients

For the normalised inner-product kernel defined in section 4.3, the kernel gradient is given by,

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} K(\boldsymbol{\theta}, \mathbf{x}_i, \mathbf{x}_j) &= \left[ \nabla_{\boldsymbol{\theta}} \phi(\mathbf{x}_i, \boldsymbol{\theta}) \right]^T \mathbf{G}^{-1}(\boldsymbol{\theta}) \phi(\mathbf{x}_j, \boldsymbol{\theta}) + \\ &\phi(\mathbf{x}_i, \boldsymbol{\theta})^T \left[ \nabla_{\boldsymbol{\theta}} \mathbf{G}^{-1}(\boldsymbol{\theta}) \right] \phi(\mathbf{x}_j, \boldsymbol{\theta}) + \phi(\mathbf{x}_i, \boldsymbol{\theta})^T \mathbf{G}^{-1}(\boldsymbol{\theta}) \left[ \nabla_{\boldsymbol{\theta}} \phi(\mathbf{x}_j, \boldsymbol{\theta}) \right] \end{aligned} \quad (37)$$

By symmetry, the first and last terms are identical over the summation in (36). The normalisation term,  $\mathbf{G}$ , can be calculated using either (27) or the approximation in (28).

Consider the example score-space given in (25). 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),

$$\nabla_{\boldsymbol{\theta}_k} (\nabla_{\boldsymbol{\theta}_j} \ln p(\mathbf{x}|\boldsymbol{\theta})) = \frac{\nabla_{\boldsymbol{\theta}_j, \boldsymbol{\theta}_k}^2 p(\mathbf{x}|\boldsymbol{\theta})}{p(\mathbf{x}|\boldsymbol{\theta})} - \left[ \nabla_{\boldsymbol{\theta}_j} \ln p(\mathbf{x}|\boldsymbol{\theta}) \right] \left[ \nabla_{\boldsymbol{\theta}_k} \ln p(\mathbf{x}|\boldsymbol{\theta}) \right]^T \quad (38)$$

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,

$$\nabla_{\boldsymbol{\theta}_k} (\mathbf{G}^{-1}) = -\mathbf{G}^{-1} \mathcal{E} \left\{ 2 \left( \phi(\mathbf{x}, \boldsymbol{\theta}) - \boldsymbol{\mu}_{\phi} \right) \left( \nabla_{\boldsymbol{\theta}_k} \phi(\mathbf{x}, \boldsymbol{\theta}) \right)^T \right\} \mathbf{G}^{-1} \quad (39)$$

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

$$\nabla_{\boldsymbol{\theta}_k} (\mathbf{G}^{-1}) = -\mathbf{G}^{-1} \left[ \frac{2}{n} \sum_{i=1}^n \left( \phi(\mathbf{x}_i, \boldsymbol{\theta}) - \boldsymbol{\mu}_{\phi} \right) \left( \nabla_{\boldsymbol{\theta}_k} \phi(\mathbf{x}_i, \boldsymbol{\theta}) \right)^T \right] \mathbf{G}^{-1} \quad (40)$$

### 5.3 Application to GMMs

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

$$p(\mathbf{x}|\boldsymbol{\theta}) = \sum_{j=1}^m c_j \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \quad (41)$$

Using separate GMMs for each class, the LLR score-space (25) is defined by,

$$\boldsymbol{\phi}(\mathbf{x}, \boldsymbol{\theta}_A, \boldsymbol{\theta}_B) = \begin{bmatrix} \ln p(\mathbf{x}|\boldsymbol{\theta}_A) - \ln p(\mathbf{x}|\boldsymbol{\theta}_B) \\ \nabla_{\boldsymbol{\theta}_A} \ln p(\mathbf{x}|\boldsymbol{\theta}_A) \\ \nabla_{\boldsymbol{\theta}_B} \ln p(\mathbf{x}|\boldsymbol{\theta}_B) \end{bmatrix} \quad (42)$$

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 (42) may be rewritten more explicitly as,

$$\boldsymbol{\phi}(\mathbf{x}, \boldsymbol{\theta}_A, \boldsymbol{\theta}_B) = \begin{bmatrix} \ln p(\mathbf{x}|\boldsymbol{\theta}_A) - \ln p(\mathbf{x}|\boldsymbol{\theta}_B) \\ \nabla_{\boldsymbol{\mu}_A} \ln p(\mathbf{x}|\boldsymbol{\theta}_A) \\ \nabla_{\boldsymbol{\Sigma}_A} \ln p(\mathbf{x}|\boldsymbol{\theta}_A) \\ \nabla_{\boldsymbol{\mu}_B} \ln p(\mathbf{x}|\boldsymbol{\theta}_B) \\ \nabla_{\boldsymbol{\Sigma}_B} \ln p(\mathbf{x}|\boldsymbol{\theta}_B) \end{bmatrix} \quad (43)$$

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

$$\nabla_{\boldsymbol{\theta}_A} \boldsymbol{\phi}(\mathbf{x}, \boldsymbol{\theta}_A) = \begin{bmatrix} \nabla_{\boldsymbol{\theta}_A} l(\mathbf{x}|\boldsymbol{\theta}_A) \\ \nabla_{\boldsymbol{\mu}_A, \boldsymbol{\theta}_A}^2 l(\mathbf{x}|\boldsymbol{\theta}_A) \\ \nabla_{\boldsymbol{\Sigma}_A, \boldsymbol{\theta}_A}^2 l(\mathbf{x}|\boldsymbol{\theta}_A) \\ 0 \\ 0 \end{bmatrix}, \quad \nabla_{\boldsymbol{\theta}_B} \boldsymbol{\phi}(\mathbf{x}, \boldsymbol{\theta}_B) = \begin{bmatrix} -\nabla_{\boldsymbol{\theta}_B} l(\mathbf{x}|\boldsymbol{\theta}_B) \\ 0 \\ 0 \\ \nabla_{\boldsymbol{\mu}_B, \boldsymbol{\theta}_B}^2 l(\mathbf{x}|\boldsymbol{\theta}_B) \\ \nabla_{\boldsymbol{\Sigma}_B, \boldsymbol{\theta}_B}^2 l(\mathbf{x}|\boldsymbol{\theta}_B) \end{bmatrix} \quad (44)$$

where,

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

The cross-derivatives,  $\nabla_{\boldsymbol{\theta}_A, \boldsymbol{\theta}_B}^2$  and  $\nabla_{\boldsymbol{\theta}_B, \boldsymbol{\theta}_A}^2$  are zero since classes  $A$  and  $B$  are independent. Using the general equation, (38), the within-class derivatives may be calculated using,

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

where

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

and

$$\begin{aligned} \nabla_{\boldsymbol{\mu}_j, \boldsymbol{\mu}_j}^2 l_j(\mathbf{x}|\boldsymbol{\theta}) &= -\boldsymbol{\Sigma}_j^{-1} \\ \nabla_{\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j}^2 l_j(\mathbf{x}|\boldsymbol{\theta}) &= \text{Diag} \left[ -\boldsymbol{\Sigma}_j^{-1} \boldsymbol{\Sigma}_j^{-1} (\mathbf{x} - \boldsymbol{\mu}_j) \right] \\ \nabla_{\boldsymbol{\Sigma}_j, \boldsymbol{\Sigma}_j}^2 l_j(\mathbf{x}|\boldsymbol{\theta}) &= \frac{1}{2} \boldsymbol{\Sigma}_j^{-1} \boldsymbol{\Sigma}_j^{-1} - \text{Diag} \left\{ \text{diag} \left[ \left( \boldsymbol{\Sigma}_j^{-1} (\mathbf{x} - \boldsymbol{\mu}_j) \right) \left( \boldsymbol{\Sigma}_j^{-1} \boldsymbol{\Sigma}_j^{-1} (\mathbf{x} - \boldsymbol{\mu}_j) \right)^T \right] \right\} \end{aligned}$$

Derivations are given in appendix B.

## 5.4 Application to HMMs

To allow classification of dynamic data, HMMs can be used as the base distributions. A score-space may then be created. For clarity of presentation, and without loss of generality, the discussion below will only consider score-spaces that contain the log-likelihood ratio and the first derivatives with respect to the means of the GMM output probability distributions,

$$\phi(\mathbf{x}, \boldsymbol{\theta}) = \begin{bmatrix} \ln p(\mathbf{x}|\boldsymbol{\theta}_A) - \ln p(\mathbf{x}|\boldsymbol{\theta}_B) \\ \nabla_{\boldsymbol{\mu}_A} \ln p(\mathbf{x}|\boldsymbol{\theta}_A) \\ \nabla_{\boldsymbol{\mu}_B} \ln p(\mathbf{x}|\boldsymbol{\theta}_B) \end{bmatrix} \quad (45)$$

As for GMMs, maximum margin training for HMMs requires the differential of the score-space with respect to the model parameters. Consider the differential of the score-space with respect to the mean of the  $n$ th mixture component of the  $k$ th state,  $\boldsymbol{\mu}_{kn}$ . The differential of the log-likelihood ratio with respect to  $\boldsymbol{\mu}_{kn}$  is trivial since it is contained within the score-space. The derivatives of the other components with respect to  $\boldsymbol{\mu}_{kn}$  are given by (see appendix C for derivation),

$$\begin{aligned} \nabla_{\boldsymbol{\mu}_{kn}} \nabla_{\boldsymbol{\mu}_{jm}}^T \ln p(\mathbf{X}|\mathcal{M}) &= \sum_{t=1}^T \sum_{\tau=1}^T \left( p(\phi_t = \{s_j, m\}, \phi_\tau = \{s_k, n\} | \mathbf{X}, \mathcal{M}) - \gamma_{jm}(t) \gamma_{kn}(\tau) \right) \\ &\quad \boldsymbol{\Sigma}_{kn}^{-1}(\mathbf{x}_\tau - \boldsymbol{\mu}_{kn})(\mathbf{x}_t - \boldsymbol{\mu}_{jm})^T \boldsymbol{\Sigma}_{jm}^{-1} \\ &\quad - \delta_{jk} \delta_{mn} \sum_{t=1}^T \gamma_{jm}(t) \boldsymbol{\Sigma}_{jm}^{-T} \end{aligned} \quad (46)$$

By symmetry, the first term of equation (46) may be separated into off-diagonal and diagonal components,

$$\begin{aligned} \nabla_{\boldsymbol{\mu}_{kn}} \nabla_{\boldsymbol{\mu}_{jm}}^T \ln p(\mathbf{X}|\mathcal{M}) &= 2 \sum_{t=1}^T \sum_{\tau=t+1}^T \left( p(\phi_t = \{s_j, m\}, \phi_\tau = \{s_k, n\} | \mathbf{X}, \mathcal{M}) - \gamma_{jm}(t) \gamma_{kn}(\tau) \right) \\ &\quad \boldsymbol{\Sigma}_{kn}^{-1}(\mathbf{x}_\tau - \boldsymbol{\mu}_{kn})(\mathbf{x}_t - \boldsymbol{\mu}_{jm})^T \boldsymbol{\Sigma}_{jm}^{-1} \\ &\quad + \sum_{t=1}^T \left( \gamma_{jm}(t) - \gamma_{jm}(t)^2 \right) \boldsymbol{\Sigma}_{jm}^{-1}(\mathbf{x}_t - \boldsymbol{\mu}_{jm})(\mathbf{x}_t - \boldsymbol{\mu}_{jm})^T \boldsymbol{\Sigma}_{jm}^{-1} \\ &\quad - \delta_{jk} \delta_{mn} \sum_{t=1}^T \gamma_{jm}(t) \boldsymbol{\Sigma}_{jm}^{-T} A \end{aligned} \quad (47)$$

where  $\delta_{ab}$  is the Kronecker delta ( $\delta_{ab} = 1$  if  $a = b$ , otherwise  $\delta_{ab} = 0$ ). The mixture component occupancy,  $\gamma_{jm}(t)$ , is given by,

$$\gamma_{jm}(t) = \gamma_j(t) \frac{c_{jm} \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}_{jm}, \boldsymbol{\Sigma}_{jm})}{\sum_{m=1}^M c_{jm} \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}_{jm}, \boldsymbol{\Sigma}_{jm})} \quad (48)$$

where  $\gamma_j(t)$  is the state occupancy. This may be calculated using the Forward-Backward algorithm [22, 19],

$$\gamma_j(t) = p(s(t) = j | \mathbf{X}, \mathcal{M}) = \frac{\alpha_j(t) \beta_j(t)}{p(\mathbf{X} | \mathcal{M})} \quad (49)$$

where,

$$\alpha_j(t) = p(\mathbf{x}_1, \dots, \mathbf{x}_t, s(t) = j | \mathcal{M}) = \left[ \sum_{i=2}^{N-1} \alpha_i(t-1) a_{ij} \right] b_j(\mathbf{x}_t) \quad (50)$$

$$\beta_j(t) = p(\mathbf{x}_{t+1}, \dots, \mathbf{x}_T | s(t) = j, \mathcal{M}) = \sum_{i=2}^{N-2} a_{ji} b_i(\mathbf{x}_{t+1}) \beta_i(t+1) \quad (51)$$

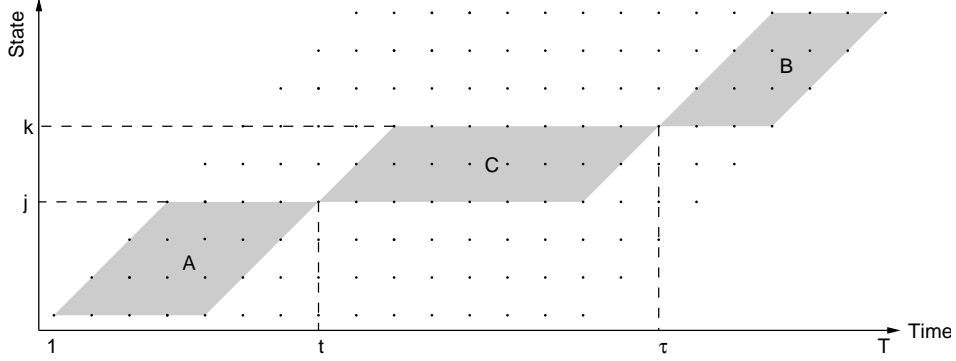


Figure 5: Valid paths for calculating  $p(\phi_t = \{s_j, m\}, \phi_\tau = \{s_k, n\} | \mathbf{X}, \mathcal{M})$

The probability of passing through both states  $\{s_j, m\}$  and  $\{s_k, n\}$  (at times  $t$  and  $\tau$  respectively),  $p(\phi_t = \{s_j, m\}, \phi_\tau = \{s_k, n\} | \mathbf{X}, \mathcal{M})$  is significantly more complex to calculate. Given these constraints, figure 5 depicts the valid paths through the HMM. The probability of passing through states  $\{s_j, m\}$  and  $\{s_k, n\}$  at times  $t$  and  $\tau$  is given by,

$$\begin{aligned}
 p(\phi_t = \{s_j, m\}, \phi_\tau = \{s_k, n\} | \mathbf{X}, \mathcal{M}) &= p(\mathbf{x}_1, \dots, \mathbf{x}_t, s(t) = j, m(t) = m | \mathcal{M}) \\
 &\quad p(\mathbf{x}_{t+1}, \dots, \mathbf{x}_\tau, s(\tau) = k, m(\tau) = n | s(t) = j, \mathcal{M}) p(\mathbf{x}_{\tau+1}, \dots, \mathbf{x}_T | s(\tau) = k, \mathcal{M})
 \end{aligned} \tag{52}$$

where the three probabilities correspond to the regions labelled A, C and B in figure 5. The first (region A) and last (region B) terms are identical to the standard Forward-Backward terms,  $\alpha_j(t)$  and  $\beta_k(\tau)$ . Equation (52) may thus be written,

$$p(\phi_t = \{s_j, m\}, \phi_\tau = \{s_k, n\} | \mathbf{X}, \mathcal{M}) = \frac{\alpha_j(t) \gamma_{jm}(t) p(\mathbf{x}_{t+1}, \dots, \mathbf{x}_\tau, s(\tau) = k | s(t) = j, \mathcal{M}) \gamma_{kn}(\tau) \beta_k(\tau)}{p(\mathbf{X} | \mathcal{M})} \tag{53}$$

Using this expanded expression, the first term of equation (47) may be calculated using the following, modified, Forward-Backward algorithm,

- 
1. Initialise  $\alpha_j(t)$  and  $\beta_j(t)$  for all times,  $t$ , and states  $j$
  2. **Foreach**  $t = 1 \rightarrow T$ 
    - Foreach**  $j = 1 \rightarrow N$ 
      - Initialise  $\gamma_k(t) = 0$  for all states,  $k$
      - Set  $\gamma_j(t) = \alpha_j(t)$
      - **Foreach**  $\tau = t + 1 \rightarrow T$ 
        - \* Forward step: calculate  $\gamma_k(\tau)$  for all states,  $k$
        - \*  $p(s(t) = j, s(\tau) = k | \mathbf{X}, \mathcal{M}) = \gamma_k(\tau) \beta_k(\tau)$
      - **End**
    - End**
  3. **End**
- 

To avoid underflows, log-likelihoods may be used. The computational cost of this algorithm scales  $\mathcal{O}(N^2 T^2)$  where  $N$  is the number of states and  $T$  is the number of frames per utterance. Since the correlation between distant frames is expected to be minimal, the second summation over  $T$  (labelled A above) may be approximated by a summation from  $t + 1$  through to  $t + c$  where  $c$  is the number of frames before (by symmetry) and after the current frame. Complexity then becomes  $N^2 c T$  where  $c < T$ .

## 5.5 Computational cost

As discussed in section 4.4, training the SVM with GMM- and HMM-based LLR kernels requires training examples to be mapped into the generative score-space. For GMMs, 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. HMMs have the additional cost of computing the occupancy for each mixture of each state. Using the Forward-Backward algorithm, this may be calculated in  $\mathcal{O}(TN^2m)$  time, where  $T$  is the number of frames per utterance and  $N$  is the number of states per HMM. Thus, for HMMs, the computational cost of mapping input data to the score-space is  $\mathcal{O}(nTN^2m^2d)$ . However, the computational cost of generating the score-space may be significantly reduced by only including selected derivatives (features). In this report, the most discriminative features were chosen using Fisher ratios. These are features that maximise,

$$F = \text{tr}(\mathbf{W}^{-1}\mathbf{B}) \quad (54)$$

where  $\mathbf{W}$  and  $\mathbf{B}$  are the within- and between-class covariances of the component in the feature-space.

The first step of the MM parameter adjustment requires the gradient,  $\nabla_{\theta}$ , to be calculated. From equation (36), it is clear that, for GMMs, 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  $\mathcal{O}(m^2d^2n)$ .

For HMMs, the time-correlation element of the derivatives significantly increases the computational cost. However, since the correlation between distance frames is expected to be negligible, the double summation over  $T$  may be approximated using a localised subset of frames. Alternatively, since preliminary experiments indicate that the time correlation component of the derivative is negligible in comparison with other terms, it may be ignored.

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<sup>light</sup>, mitigate this problem by using chunking algorithms [30].

## 6 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 [7]. All SVMs were trained using a modified version of SVM<sup>light</sup> [30] configured with a linear kernel and  $C = 1$ . All experiments in this section were performed using score-spaces based upon GMM probability distributions.

### 6.1 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).

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

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%.

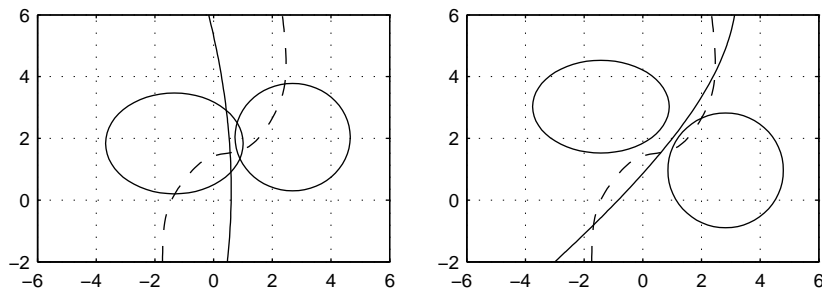


Figure 6: LLR (solid) and Bayes (dashed) decision boundaries before (left) and after (right) parameter optimisation

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 on the LLR. Figure 6 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.

A number of configurations of generative kernel and score-space were examined. Three different LLR score-spaces were examined, LLR,  $\text{LLR} + \nabla_{\mu}$  (LLR term plus derivatives of the LLR with respect to the means of both classes  $A$  and  $B$ ) and  $\text{LLR} + \nabla_{\mu} + \nabla_{\Sigma}$  (including derivatives with respect to covariances). Initially, the normalisation term  $\mathbf{G}$  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 6, the MM algorithm shifted the distributions away from the underlying data, moving the decision

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

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%.

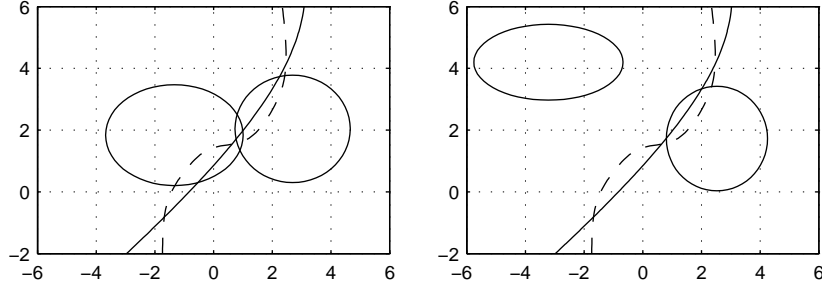


Figure 7: Decision boundary (a) before and (b) after parameter optimisation; LLR+ $\nabla_{\mu}$  +  $\nabla_{\Sigma}$  score-space

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 7, the powerful LLR+ $\nabla_{\mu}$ + $\nabla_{\Sigma}$  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+ $\nabla_{\mu}$ + $\nabla_{\Sigma}$  score-space, the 2-component systems achieved training and test error rates equal to the Bayes error after MM training. The 2-component LLR+ $\nabla_{\mu}$ + $\nabla_{\Sigma}$  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  $\mathbf{G}$  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 +  $\nabla_{\mu}$  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.

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  $C = 0.1$ ).

C	Training (%)		Test (%)		Objective		Margin	
	Initial	Final	Initial	Final	Initial	Final	Initial	Final
<b>1-Component</b>								
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
<b>2-Component</b>								
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 +  $\nabla_{\mu}$  +  $\nabla_{\Sigma}$  score-space with varying  $C$

## 6.2 Deterding data

The Deterding dataset [7] 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  $\omega_a$ -vs- $\omega_b$ . If more classes were tied, the tie was resolved using MLE GMMs.

Classifier	Components	Training (%)		Test (%)	
		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 + $\nabla_{\mu}$ )	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% [40, 23]. 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.

## 7 Future plans

Over the next two years, I plan to concentrate my work on two areas of research: MM training for HMMs and Multiclass SVMs. Below is a detailed plan of how I intend to split my time.

**MM training for HMMs (3 months)** Complete the implementation of the MM training algorithm for HMM-based score-spaces (section 4.5). Initially, I will use binary subsets of the Isolet [13] data set to test my code and to compare MM trained performance with that of ML and MMI trained classifiers. In addition, I will perform basic multiclass tests on the Isolet E-set using pairwise classifiers and a majority voting scheme (using both positive and negative voting). Finally, I intend to test performance using pruned (to leave two options) confusion networks from a Large Vocabulary Continuous Speech Recognition (LVCSR) task. The task will be simplified by only training SVMs for particularly confusable pairs of words. For all other words, the label will be chosen using the posterior probabilities of each word.

**ECOC implementation (2 months)** Implement basic ECOC multi-class SVMs using the techniques detailed by Dietterich [21] (section 3.3). The initial implementation will be written using the Perl text processing language. Experiments will be performed to determine the effect of different output codes on system performance. In particular, I will analyse the cost-vs-benefits of increasing the code length. Additionally, I will analyse the performance variability induced by different code-class assignments.

Experiments will be performed using the Isolet [13] E-set (additional experiments using all letters of the alphabet will be performed if time permits) and unpruned (or partially pruned) confusion networks from a LVCSR system. Results will be compared against baseline 1-versus-1 (using positive and negative voting) and 1-versus-all binary decomposition techniques (sections 3.2 and 3.1).

**Multiclass MM training – all-in-one method (6 months)** Formulate a multi-class version of maximum-margin training using existing formulations of the all-in-one method. This will involve examining previous all-in-one theories to determine the benefits and pitfalls of each formulation.

Since the new algorithm needs to scale to large numbers of classes and large data sets, efficiency is of prime importance. To achieve this, I will analyse the performance of the algorithm and search for approximations that will allow the system to scale. Of particular interest are methods of bootstrapping the optimisation by solving smaller binary (for example, Hill *et al.* [27]) or multiclass problems.

**Multiclass MM training – implementation (10 months)** Using the theory derived previously, I will implement a multiclass system based upon the all-in-one theory. Where possible, existing tools and libraries will be used to maximise the time available for experimentation and adaptation of the framework. Initially, I plan to test the framework using small scale test sets (such as the Iris data set). Next I will test performance using the Isolet E-set. Finally, in order to confirm the scalability of the system, I will work with unpruned (partially pruned) confusion networks from a LVCSR task.

**Write-up and examination (3 months)** Complete the formal write-up of a thesis for submission to the Degree of Doctor of Philosophy.

## 8 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 [3]. This difference may be due to the greater complexity of LLR kernels compared to the kernels considered by Ayat. Although, to date, experiments have only been performed using GMM-based LLR kernels, future experiments will be performed using

dynamic data (such as the Isolet [13] isolated letter task) using HMM-based score-spaces. The theory for this work is contained in sections 4.5, 5.4 and appendix C.

Preliminary research on multiclass SVMs has also been conducted. This is contained in section 3. Detailed plans for the future progress of this work are contained in section 7. In particular, it is expected that the research will concentrate on examining alternative forms of the all-in-one SVM formulation. Unlike some all-in-one theories, focus will be placed upon creating a practical theory that can be scaled to large systems (substituting approximations for exact results where necessary).

## Acknowledgements

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

## References

- [1] C. Angulo, X. Parra, and A. Catalá. K-SVRC. a support vector machine for multi-class classification. *Neurocomputing*, 55:57–77, 2002.
- [2] 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.
- [3] 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.
- [4] 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.
- [5] 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.
- [6] 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.
- [7] C.L. Blake and C.J. Merz. UCI repository of machine learning databases. Technical report, University of California, Irvine, School of Information and Computer Science, 1998.
- [8] O. Bousquet and D.J.L. Herrmann. On the complexity of learning the kernel matrix. In *Advances in Neural Information Processing Systems*, 2002.
- [9] E.J. Bredensteiner and K.P. Bennett. Multicategory classification by support vector machines. *Computational Optimizations and Applications*, 12:53–79, 1999a.
- [10] 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.
- [11] O. Chapelle and V. Vapnik. Model selection for support vector machines. In *Advances in Neural Information Processing Systems*, 1999.
- [12] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support sector machines. In *Advances in Neural Information Processing Systems*, 2001.

- [13] R. Cole, Y. Muthusamy, and M. Fanty. The ISOLET spoken letter database. Technical Report CSE 90-004, Oregon Graduate Institute of Science and Technology, March 1990.
- [14] K. Crammer and Y. Singer. On the learnability and design of output codes for multiclass problems. *Machine Learning*, 47, 2002.
- [15] 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.
- [16] N. Cristianini, C. Campbell, and J. Shawe-Taylor. Dynamically adapting kernels in support vector machines. In *Advances in Neural Information Processing Systems*, 1998.
- [17] N. Cristianini and J. Shawe-Taylor. *Support Vector Machines and other Kernel-based Learning Methods*. Cambridge University Press, 2000.
- [18] F. Cutzu. Polychotomous classification with pairwise classifiers: a new voting principle. Technical Report TR573, Department of Computer Science, Indiana University, January 2003.
- [19] J.R. Deller, J.G. Proakis, and J.H.L. Hansen. *Discrete-Time Processing of Speech Signals*. Prentice Hall, 1993.
- [20] T.G. Dietterich and G. Bakiri. Error-correcting output codes: A general method for improving multiclass inductive learning programs. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, pages 572–577. AAAI Press, 1991.
- [21] T.G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.
- [22] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. John Wiley & Sons, 2nd edition, 2001.
- [23] A. Ganapathiraju, J. Hamaker, and J. Picone. Support Vector Machines for speech recognition. In *International Conference on Spoken Language Processing*, 1998.
- [24] A. Ganapathiraju, J. Hamaker, and J. Picone. Hybrid SVM/HMM architectures for speech recognition. In *International Conference on Spoken Language Processing*, 2000.
- [25] Y. Grandvalet and S. Canu. Adaptive scaling for feature selection in SVMs. In *Advances in Neural Information Processing Systems*, 2002.
- [26] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice-Hall International, 2nd edition, 1999.
- [27] S.I. Hill. *Applications of Statistical Learning Theory to Signal Processing Problems*. PhD thesis, University of Cambridge, April 2003.
- [28] C.-W. Hsu and C.-J. Lin. A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, 13:415–425, 2002.
- [29] T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In S.A. Solla and D.A. Cohn, editors, *Advances in Neural Information Processing Systems 11*, pages 487–493. MIT Press, 1999.
- [30] 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.
- [31] J. Kandola, J. Shawe-Taylor, and N. Cristianini. Optimizing kernel alignment over combinations of kernels. *Journal of Machine Learning Research*, 2002.
- [32] 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.

- [33] E. Mayoraz and M. Moreira. On the decomposition of polychotomies into dichotomies. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 219–226, July 1997.
- [34] 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.
- [35] J. Platt. Fast training of support vector machines using Sequential Minimal Optimization. In B. Scholkopf, C. Burges and A. Smola, editor, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1999.
- [36] J.C. Platt. Probabilities for sv machines. In A.J. Smolai, P.L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, 2000.
- [37] B. Schölkopf and A.J. Smola. *Learning with kernels*. MIT Press, 2002.
- [38] 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.
- [39] N. Smith and M. Gales. Speech recognition using SVMs. In *Advances in Neural Information Processing Systems*, 2001.
- [40] N.D. Smith. *Using Augmented Statistical Models and Score Spaces for Classification*. PhD thesis, University of Cambridge, September 2003.
- [41] 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.
- [42] V.N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, 1998.
- [43] J. Weston and C. Watkins. Multi-class support vector machines. Technical Report CSD-TR-98-04, Royal Holloway, University of London, May 1998.

## A Derivatives of the dual SVM objective function

The dual objective function is given by,

$$\max_{\alpha} W(\boldsymbol{\theta}, \boldsymbol{\alpha}) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j K(\boldsymbol{\theta}, \mathbf{x}_i, \mathbf{x}_j) \quad (55)$$

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,

$$\nabla_{\boldsymbol{\theta}} W(\boldsymbol{\alpha}, \boldsymbol{\theta}) = -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \nabla_{\boldsymbol{\theta}} K(\boldsymbol{\theta}, \mathbf{x}_i, \mathbf{x}_j) \quad (56)$$

where,

$$\begin{aligned} \nabla_{\boldsymbol{\theta}_k} K(\boldsymbol{\theta}, \mathbf{x}_i, \mathbf{x}_j) &= \left[ \nabla_{\boldsymbol{\theta}_k} \boldsymbol{\phi}(\mathbf{x}_i, \boldsymbol{\theta}) \right]^T \mathbf{G}^{-1} \boldsymbol{\phi}(\mathbf{x}_j, \boldsymbol{\theta}) \\ &+ \boldsymbol{\phi}(\mathbf{x}_i, \boldsymbol{\theta})^T \left[ \nabla_{\boldsymbol{\theta}_k} \mathbf{G}^{-1} \right] \boldsymbol{\phi}(\mathbf{x}_j, \boldsymbol{\theta}) \\ &+ \boldsymbol{\phi}(\mathbf{x}_i, \boldsymbol{\theta})^T \mathbf{G}^{-1} \left[ \nabla_{\boldsymbol{\theta}_k} \boldsymbol{\phi}(\mathbf{x}_j, \boldsymbol{\theta}) \right] \end{aligned} \quad (57)$$

The derivative of the score-space with respect to the parameter  $\theta_k$  is given by,

$$\begin{aligned}
\nabla_{\theta_k} [\phi(\mathbf{x}, \boldsymbol{\theta})]_j &= \nabla_{\theta_k} \left( \nabla_{\theta_j} \ln p(\mathbf{x}|\boldsymbol{\theta}) \right) \\
&= \nabla_{\theta_k} \left( \frac{1}{p(\mathbf{x}|\boldsymbol{\theta})} \nabla_{\theta_j} p(\mathbf{x}|\boldsymbol{\theta}) \right) \\
&= \frac{1}{p(\mathbf{x}|\boldsymbol{\theta})^2} \left\{ p(\mathbf{x}|\boldsymbol{\theta}) \left( \nabla_{\theta_j, \theta_k}^2 p(\mathbf{x}|\boldsymbol{\theta}) \right) - \left( \nabla_{\theta_j} p(\mathbf{x}|\boldsymbol{\theta}) \right) \left( \nabla_{\theta_k} p(\mathbf{x}|\boldsymbol{\theta}) \right) \right\} \\
&= \frac{1}{p(\mathbf{x}|\boldsymbol{\theta})} \left( \nabla_{\theta_j, \theta_k}^2 p(\mathbf{x}|\boldsymbol{\theta}) \right) - \frac{1}{p(\mathbf{x}|\boldsymbol{\theta})} \left( \nabla_{\theta_j} p(\mathbf{x}|\boldsymbol{\theta}) \right) \frac{1}{p(\mathbf{x}|\boldsymbol{\theta})} \left( \nabla_{\theta_k} p(\mathbf{x}|\boldsymbol{\theta}) \right) \\
&= \frac{1}{p(\mathbf{x}|\boldsymbol{\theta})} \left( \nabla_{\theta_j, \theta_k}^2 p(\mathbf{x}|\boldsymbol{\theta}) \right) - \left( \nabla_{\theta_j} \ln p(\mathbf{x}|\boldsymbol{\theta}) \right) \left( \nabla_{\theta_k} \ln p(\mathbf{x}|\boldsymbol{\theta}) \right) \tag{58}
\end{aligned}$$

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

$$\nabla_{\theta_k} (\mathbf{G}^{-1}) = -\mathbf{G}^{-1} (\nabla_{\theta_k} \mathbf{G}) \mathbf{G}^{-1}$$

Since,

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

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

$$\begin{aligned}
\nabla_{\theta_k} \mathbf{G} &= 2\mathcal{E} \left\{ \left( \phi(\mathbf{x}, \boldsymbol{\theta}) \right) \left( \nabla_{\theta_k} \phi(\mathbf{x}, \boldsymbol{\theta}) \right)^T \right\} \\
&\quad - 2\mathcal{E} \left\{ \phi(\mathbf{x}, \boldsymbol{\theta}) \right\} \mathcal{E} \left\{ \nabla_{\theta_k} \phi(\mathbf{x}, \boldsymbol{\theta}) \right\}^T \\
&= 2\mathcal{E} \left\{ \left( \phi(\mathbf{x}, \boldsymbol{\theta}) \right) \left( \nabla_{\theta_k} \phi(\mathbf{x}, \boldsymbol{\theta}) \right)^T \right\} \\
&\quad - 2\mathcal{E} \left\{ \boldsymbol{\mu}_\phi \left( \nabla_{\theta_k} \phi(\mathbf{x}, \boldsymbol{\theta}) \right)^T \right\} \\
&= 2\mathcal{E} \left\{ \left( \phi(\mathbf{x}, \boldsymbol{\theta}) - \boldsymbol{\mu}_\phi \right) \left( \nabla_{\theta_k} \phi(\mathbf{x}, \boldsymbol{\theta}) \right)^T \right\}
\end{aligned}$$

Therefore,

$$\nabla_{\theta_k} (\mathbf{G}^{-1}) = -2\mathbf{G}^{-1} \mathcal{E} \left\{ \left( \phi(\mathbf{x}, \boldsymbol{\theta}) - \boldsymbol{\mu}_\phi \right) \left( \nabla_{\theta_k} \phi(\mathbf{x}, \boldsymbol{\theta}) \right)^T \right\} \mathbf{G}^{-1} \tag{59}$$

## 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  $\theta_A$  and  $\theta_B$  are given by,

$$\nabla_{\theta_A} \phi(\mathbf{x}, \boldsymbol{\theta}) = \begin{bmatrix} \nabla_{\theta_A} \ln p(\mathbf{x}|\boldsymbol{\theta}) \\ \nabla_{\boldsymbol{\mu}_A, \theta_A}^2 \ln p(\mathbf{x}|\boldsymbol{\theta}) \\ \nabla_{\boldsymbol{\Sigma}_A, \theta_A}^2 \ln p(\mathbf{x}|\boldsymbol{\theta}) \\ 0 \\ 0 \end{bmatrix} \tag{60}$$

and,

$$\nabla_{\boldsymbol{\theta}_B} \phi(\mathbf{x}, \boldsymbol{\theta}) = \begin{bmatrix} \nabla_{\boldsymbol{\theta}_B} \ln p(\mathbf{x}|\boldsymbol{\theta}) \\ 0 \\ 0 \\ \nabla_{\boldsymbol{\mu}_B, \boldsymbol{\theta}_B}^2 \ln p(\mathbf{x}|\boldsymbol{\theta}) \\ \nabla_{\boldsymbol{\Sigma}_B, \boldsymbol{\theta}_B}^2 \ln p(\mathbf{x}|\boldsymbol{\theta}) \end{bmatrix} \quad (61)$$

For each class,

$$\nabla_{\boldsymbol{\mu}_j} = \gamma_j(\mathbf{x}) \left[ \boldsymbol{\Sigma}_j^{-1} (\mathbf{x} - \boldsymbol{\mu}_j) \right] \quad (62)$$

$$\nabla_{\boldsymbol{\Sigma}_j} = \frac{\gamma_j(\mathbf{x})}{2} \text{diag} \left[ -\boldsymbol{\Sigma}_j^{-1} + \left( \boldsymbol{\Sigma}_j^{-1} (\mathbf{x} - \boldsymbol{\mu}_j) \right) \left( \boldsymbol{\Sigma}_j^{-1} (\mathbf{x} - \boldsymbol{\mu}_j) \right)^T \right] \quad (63)$$

The second-derivatives are given by,

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

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

$$\begin{aligned} \nabla_{\boldsymbol{\mu}_j, \boldsymbol{\mu}_j}^2 l_j(\mathbf{x}|\boldsymbol{\theta}) &= -\boldsymbol{\Sigma}_j^{-1} \\ \nabla_{\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j}^2 l_j(\mathbf{x}|\boldsymbol{\theta}) &= \text{Diag} \left[ -\boldsymbol{\Sigma}_j^{-1} \boldsymbol{\Sigma}_j^{-1} (\mathbf{x} - \boldsymbol{\mu}_j) \right] \\ \nabla_{\boldsymbol{\Sigma}_j, \boldsymbol{\Sigma}_j}^2 l_j(\mathbf{x}|\boldsymbol{\theta}) &= \frac{1}{2} \boldsymbol{\Sigma}_j^{-1} \boldsymbol{\Sigma}_j^{-1} - \text{Diag} \left\{ \text{diag} \left[ \left( \boldsymbol{\Sigma}_j^{-1} (\mathbf{x} - \boldsymbol{\mu}_j) \right) \left( \boldsymbol{\Sigma}_j^{-1} \boldsymbol{\Sigma}_j^{-1} (\mathbf{x} - \boldsymbol{\mu}_j) \right)^T \right] \right\} \end{aligned}$$

and,

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

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

## C Derivatives of the HMM LLR score-space

Data that consists of a sequence of observations may be modelled using a Hidden Markov Model (HMM) with a GMM output probability function. Let  $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_t)$  be the observed vectors (the utterance). Given a model  $\mathcal{M}$ , the log-likelihood of observing  $\mathbf{X}$  is given by,

$$\ln p(\mathbf{X}|\mathcal{M}) = \ln \left\{ \sum_{\phi} \prod_{t=1}^T a_{\phi_t, \phi_{t-1}} p(\mathbf{x}_t|\phi_t, \mathcal{M}) \right\} \quad (64)$$

where  $\phi = (\phi_1, \dots, \phi_t)$  is the hidden state sequence,  $\Phi$  is the set of all state sequences,  $a_{ij}$  is the probability of transitioning from state  $i$  to state  $j$ , and  $p(\mathbf{x}_t|\phi_t, \mathcal{M})$  is the output probability function for state  $\phi_t$ .

Differentiating the log-likelihood with respect to one of the state output function parameters,  $\boldsymbol{\theta}_{jm}$ , yields,

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}_{jm}} \ln p(\mathbf{X}|\mathcal{M}) &= \frac{1}{p(\mathbf{X}|\mathcal{M})} \sum_{\phi}^{\Phi} \nabla_{\boldsymbol{\theta}_{jm}} \left\{ \prod_{t=1}^T a_{\phi_t, \phi_{t-1}} p(\mathbf{x}_t | \phi_t, \mathcal{M}) \right\} \\
&= \frac{1}{p(\mathbf{X}|\mathcal{M})} \sum_{\phi}^{\Phi} \sum_{\tau=1}^T \left\{ \frac{\prod_{t=1}^T a_{\phi_t, \phi_{t-1}} p(\mathbf{x}_t | \phi_t, \mathcal{M})}{p(\mathbf{x}_{\tau} | \phi_{\tau}, \mathcal{M})} \nabla_{\boldsymbol{\theta}_{jm}} p(\mathbf{x}_{\tau} | \phi_{\tau}, \mathcal{M}) \right\} \\
&= \sum_{\phi}^{\Phi} \sum_{\tau=1}^T \left\{ \frac{p(\mathbf{X} | \phi, \mathcal{M})}{p(\mathbf{X} | \mathcal{M})} \frac{\nabla_{\boldsymbol{\theta}_{jm}} p(\mathbf{x}_{\tau} | \phi_{\tau}, \mathcal{M})}{p(\mathbf{x}_{\tau} | \phi_{\tau}, \mathcal{M})} \right\} \tag{65}
\end{aligned}$$

For a GMM output probability distribution,

$$p(\mathbf{x} | \boldsymbol{\theta}_j) = \sum_{m=1}^M c_{jm} \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_{jm}, \boldsymbol{\Sigma}_{jm}) \tag{66}$$

$$\nabla_{\boldsymbol{\mu}_{jm}} p(\mathbf{x}_t | \boldsymbol{\theta}_j) = p(\mathbf{x}_t, m(t) = m | s(t) = j, \mathcal{M}) \boldsymbol{\Sigma}_{jm}^{-1} (\mathbf{x} - \boldsymbol{\mu}_{jm}) \tag{67}$$

$$\nabla_{\boldsymbol{\Sigma}_{jm}} p(\mathbf{x} | \boldsymbol{\theta}_j) = \frac{p(\mathbf{x}_t, m(t) = m | s(t) = j, \mathcal{M})}{2} \text{diag} \left[ -\boldsymbol{\Sigma}_j^{-1} + \left( \boldsymbol{\Sigma}_j^{-1} (\mathbf{x} - \boldsymbol{\mu}_j) \right) \left( \boldsymbol{\Sigma}_j^{-1} (\mathbf{x} - \boldsymbol{\mu}_j) \right)^T \right] \tag{68}$$

### C.1 Differentiating with respect to the mean

From (65),

$$\nabla_{\boldsymbol{\mu}_{jm}} \ln p(\mathbf{X} | \mathcal{M}) = \sum_{\phi}^{\Phi} \sum_{\tau=1}^T \left\{ \frac{p(\mathbf{X} | \phi, \mathcal{M})}{p(\mathbf{X} | \mathcal{M})} \frac{\nabla_{\boldsymbol{\mu}_{jm}} p(\mathbf{x}_{\tau} | \phi_{\tau}, \mathcal{M})}{p(\mathbf{x}_{\tau} | \phi_{\tau}, \mathcal{M})} \right\}$$

Since  $\nabla_{\boldsymbol{\mu}_{jm}} p(\mathbf{x}_{\tau} | \phi_{\tau}, \mathcal{M}) = 0 \quad \forall s \neq j$ , let  $\Phi'$  be the set of paths such that  $s(\tau) = j$ ,

$$\begin{aligned}
&= \sum_{\phi'}^{\Phi'} \sum_{\tau=1}^T \left\{ \frac{p(\mathbf{X} | \phi', \mathcal{M})}{p(\mathbf{X} | \mathcal{M})} \frac{\nabla_{\boldsymbol{\mu}_{jm}} p(\mathbf{x}_{\tau} | s(\tau) = j, \mathcal{M})}{p(\mathbf{x}_{\tau} | s(\tau) = j, \mathcal{M})} \right\} \\
&= \sum_{\tau=1}^T \left\{ \sum_{\phi'}^{\Phi'} \frac{p(\mathbf{X} | \phi', \mathcal{M})}{p(\mathbf{X} | \mathcal{M})} \right\} \frac{p(\mathbf{x}_{\tau}, m(\tau) = m | s(\tau) = j, \mathcal{M})}{p(\mathbf{x}_{\tau} | s(\tau) = j, \mathcal{M})} \boldsymbol{\Sigma}_{jm}^{-1} (\mathbf{x}_{\tau} - \boldsymbol{\mu}_{jm}) \\
&= \sum_{\tau=1}^T \frac{p(\mathbf{X}, s(\tau) = j | \mathcal{M})}{p(\mathbf{X} | \mathcal{M})} p(m(\tau) = m | s(\tau) = j, \mathbf{x}_{\tau}, \mathcal{M}) \boldsymbol{\Sigma}_{jm}^{-1} (\mathbf{x}_{\tau} - \boldsymbol{\mu}_{jm}) \\
&= \sum_{\tau=1}^T p(s(\tau) = j | \mathbf{X}, \mathcal{M}) p(m(\tau) = m | s(\tau) = j, \mathbf{x}_{\tau}, \mathcal{M}) \boldsymbol{\Sigma}_{jm}^{-1} (\mathbf{x}_{\tau} - \boldsymbol{\mu}_{jm}) \\
&= \sum_{\tau=1}^T p(m(\tau) = m, s(\tau) = j | \mathbf{X}, \mathcal{M}) \boldsymbol{\Sigma}_{jm}^{-1} (\mathbf{x}_{\tau} - \boldsymbol{\mu}_{jm}) \\
&= \sum_{\tau=1}^T \gamma_{jm}(\tau) \boldsymbol{\Sigma}_{jm}^{-1} (\mathbf{x}_{\tau} - \boldsymbol{\mu}_{jm}) \tag{69}
\end{aligned}$$

where,

$$\begin{aligned}
\gamma_{jm}(\tau) &= p(m(\tau) = m, s(\tau) = j | \mathbf{X}, \mathcal{M}) \\
&= p(\phi_{\tau} = \{s_j, m\} | \mathbf{X}, \mathcal{M}) \tag{70}
\end{aligned}$$

## C.2 Differentiating with respect to the covariance matrix

Using a derivation similar to that for the differential with respect to the mean, the derivative of the log-likelihood with respect to the diagonal covariance matrix is given by,

$$\nabla_{\Sigma_{j_m}} \ln p(\mathbf{X}|\mathcal{M}) = \sum_{\tau=1}^T \frac{\gamma_{jm}(\tau)}{2} \text{diag} \left[ -\Sigma_j^{-1} + \left( \Sigma_j^{-1}(\mathbf{x} - \boldsymbol{\mu}_j) \right) \left( \Sigma_j^{-1}(\mathbf{x} - \boldsymbol{\mu}_j) \right)^T \right] \quad (71)$$

## C.3 2nd differential with respect to the mean

From (69) and (70),

$$\nabla_{\boldsymbol{\mu}_{j_m}} \ln p(\mathbf{X}|\mathcal{M}) = \sum_{t=1}^T \gamma_{jm}(t) \Sigma_{j_m}^{-1}(\mathbf{x}_t - \boldsymbol{\mu}_{j_m}) \quad (72)$$

$$\gamma_{jm}(t) = p(\phi_t = \{s_j, m\} | \mathbf{X}, \mathcal{M}) \quad (73)$$

Taking the second differential of this expression with respect to the mean gives,

$$\begin{aligned} \nabla_{\boldsymbol{\mu}_{k_n}} \nabla_{\boldsymbol{\mu}_{j_m}}^T \ln p(\mathbf{X}|\mathcal{M}) &= \sum_{t=1}^T \nabla_{\boldsymbol{\mu}_{k_n}} \left[ \gamma_{jm}(t) (\mathbf{x}_t - \boldsymbol{\mu}_{j_m})^T \Sigma_{j_m}^{-1} \right] \\ &= \left( \sum_{t=1}^T (\nabla_{\boldsymbol{\mu}_{k_n}} \gamma_{jm}(t)) (\mathbf{x}_t - \boldsymbol{\mu}_{j_m})^T \Sigma_{j_m}^{-1} \right) - \left( \delta_{jk} \delta_{mn} \sum_{t=1}^T \gamma_{jm}(t) \Sigma_{j_m}^{-T} \right) \end{aligned} \quad (74)$$

where  $\delta_{ab}$  is the Kronecker delta and is equal to 1 if  $a = b$  or 0 otherwise. First consider the term,

$$\begin{aligned} \nabla_{\boldsymbol{\mu}_{k_n}} \gamma_{jm} &= \nabla_{\boldsymbol{\mu}_{k_n}} \left[ p(\phi_t = \{s_j, m\} | \mathbf{X}, \mathcal{M}) \right] \\ &= \nabla_{\boldsymbol{\mu}_{k_n}} \left[ \frac{p(\mathbf{X}, \phi_t = \{s_j, m\} | \mathcal{M})}{p(\mathbf{X} | \mathcal{M})} \right] \\ &= \frac{1}{p(\mathbf{X} | \mathcal{M})} \nabla_{\boldsymbol{\mu}_{k_n}} p(\mathbf{X}, \phi_t = \{s_j, m\} | \mathcal{M}) \\ &\quad - \frac{1}{p(\mathbf{X} | \mathcal{M})^2} p(\mathbf{X}, \phi_t = \{s_j, m\} | \mathcal{M}) \nabla_{\boldsymbol{\mu}_{k_n}} p(\mathbf{X} | \mathcal{M}) \\ &= \frac{1}{p(\mathbf{X} | \mathcal{M})} \nabla_{\boldsymbol{\mu}_{k_n}} p(\mathbf{X}, \phi_t = \{s_j, m\} | \mathcal{M}) \\ &\quad - p(\phi_t = \{s_j, m\} | \mathbf{X}, \mathcal{M}) \nabla_{\boldsymbol{\mu}_{k_n}} \ln p(\mathbf{X} | \mathcal{M}) \\ &= \frac{1}{p(\mathbf{X} | \mathcal{M})} \sum_{\tau=1}^T p(\mathbf{X}, \phi_t = \{s_j, m\}, \phi_\tau = \{s_k, n\} | \mathcal{M}) \Sigma_{k_n}^{-1}(\mathbf{x}_\tau - \boldsymbol{\mu}_{k_n}) \\ &\quad - \gamma_{jm}(t) \sum_{\tau=1}^T \gamma_{kn}(\tau) \Sigma_{k_n}^{-1}(\mathbf{x}_\tau - \boldsymbol{\mu}_{k_n}) \\ &= \sum_{\tau=1}^T p(\phi_t = \{s_j, m\}, \phi_\tau = \{s_k, n\} | \mathbf{X}, \mathcal{M}) \Sigma_{k_n}^{-1}(\mathbf{x}_\tau - \boldsymbol{\mu}_{k_n}) \\ &\quad - \gamma_{jm}(t) \sum_{\tau=1}^T \gamma_{kn}(\tau) \Sigma_{k_n}^{-1}(\mathbf{x}_\tau - \boldsymbol{\mu}_{k_n}) \end{aligned} \quad (75)$$

Substituting (75) into (74) yields,

$$\begin{aligned}
\nabla_{\boldsymbol{\mu}_{kn}} \nabla_{\boldsymbol{\mu}_{jm}}^T \ln p(\mathbf{X}|\mathcal{M}) &= \sum_{t=1}^T \sum_{\tau=1}^T p(\phi_t = \{s_j, m\}, \phi_\tau = \{s_k, n\} | \mathbf{X}, \mathcal{M}) \boldsymbol{\Sigma}_{kn}^{-1} (\mathbf{x}_\tau - \boldsymbol{\mu}_{kn}) (\mathbf{x}_t - \boldsymbol{\mu}_{jm})^T \boldsymbol{\Sigma}_{jm}^{-1} \\
&\quad - \sum_{t=1}^T \sum_{\tau=1}^T \gamma_{jm}(t) \gamma_{kn}(\tau) \boldsymbol{\Sigma}_{kn}^{-1} (\mathbf{x}_\tau - \boldsymbol{\mu}_{kn}) (\mathbf{x}_t - \boldsymbol{\mu}_{jm})^T \boldsymbol{\Sigma}_{jm}^{-1} \\
&\quad - \delta_{jk} \delta_{mn} \sum_{t=1}^T \gamma_{jm}(t) \boldsymbol{\Sigma}_{jm}^{-T}
\end{aligned} \tag{76}$$

#### C.4 2nd differential with respect to the covariance matrix

From (71) and (70),

$$\nabla_{\boldsymbol{\Sigma}_{jm}} \ln p(\mathbf{X}|\mathcal{M}) = \sum_{\tau=1}^T \frac{\gamma_{jm}(\tau)}{2} \text{diag} \left[ -\boldsymbol{\Sigma}_j^{-1} + \boldsymbol{\Sigma}_j^{-1} (\mathbf{x} - \boldsymbol{\mu}_j) (\mathbf{x} - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1} \right] \tag{77}$$

$$\gamma_{jm}(t) = p(\phi_\tau = \{s_j, m\} | \mathbf{X}, \mathcal{M}) \tag{78}$$

Taking the second differential of this expression with respect to the diagonal covariance matrix yields,

$$\nabla_{\boldsymbol{\Sigma}_{kn}} \nabla_{\boldsymbol{\Sigma}_{jm}}^T \ln p(\mathbf{X}|\mathcal{M}) = \sum_{\tau=1}^T \nabla_{\boldsymbol{\Sigma}_{kn}} \left\{ \frac{\gamma_{jm}(\tau)}{2} \text{diag} \left[ -\boldsymbol{\Sigma}_j^{-1} + \boldsymbol{\Sigma}_j^{-1} (\mathbf{x} - \boldsymbol{\mu}_j) (\mathbf{x} - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1} \right]^T \right\} \tag{79}$$

$$\begin{aligned}
&= \frac{1}{2} \sum_{\tau=1}^T \left( \nabla_{\boldsymbol{\Sigma}_{kn}} \gamma_{jm}(\tau) \right) \text{diag} \left[ -\boldsymbol{\Sigma}_j^{-1} + \boldsymbol{\Sigma}_j^{-1} (\mathbf{x} - \boldsymbol{\mu}_j) (\mathbf{x} - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1} \right]^T \\
&\quad + \frac{1}{2} \sum_{\tau=1}^T \gamma_{jm}(\tau) \text{diag} \left[ \boldsymbol{\Sigma}_j^{-1} \boldsymbol{\Sigma}_j^{-1} - 2 \boldsymbol{\Sigma}_j^{-1} \boldsymbol{\Sigma}_j^{-1} (\mathbf{x} - \boldsymbol{\mu}_j) (\mathbf{x} - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1} \right] \tag{80}
\end{aligned}$$

Considering the term,

$$\begin{aligned}
\nabla_{\Sigma_{kn}} \gamma_{jm} &= \nabla_{\Sigma_{kn}} \left[ p(\phi_t = \{s_j, m\} | \mathbf{X}, \mathcal{M}) \right] \\
&= \nabla_{\Sigma_{kn}} \left[ \frac{p(\mathbf{X}, \phi_t = \{s_j, m\} | \mathcal{M})}{p(\mathbf{X} | \mathcal{M})} \right] \\
&= \frac{1}{p(\mathbf{X} | \mathcal{M})} \nabla_{\Sigma_{kn}} p(\mathbf{X}, \phi_t = \{s_j, m\} | \mathcal{M}) \\
&\quad - \frac{1}{p(\mathbf{X} | \mathcal{M})^2} p(\mathbf{X}, \phi_t = \{s_j, m\} | \mathcal{M}) \nabla_{\Sigma_{kn}} p(\mathbf{X} | \mathcal{M}) \\
&= \frac{1}{p(\mathbf{X} | \mathcal{M})} \nabla_{\Sigma_{kn}} p(\mathbf{X}, \phi_t = \{s_j, m\} | \mathcal{M}) \\
&\quad - p(\phi_t = \{s_j, m\} | \mathbf{X}, \mathcal{M}) \nabla_{\Sigma_{kn}} \ln p(\mathbf{X} | \mathcal{M}) \\
&= \frac{1}{2p(\mathbf{X} | \mathcal{M})} \sum_{\tau=1}^T p(\mathbf{X}, \phi_t = \{s_j, m\}, \phi_\tau = \{s_k, n\} | \mathcal{M}) \text{diag} \left[ -\Sigma_{kn}^{-1} + \Sigma_{kn}^{-1}(\mathbf{x}_\tau - \boldsymbol{\mu}_{kn})(\mathbf{x}_\tau - \boldsymbol{\mu}_{kn})^T \Sigma_{kn}^{-1} \right] \\
&\quad - \frac{\gamma_{jm}(t)}{2} \sum_{\tau=1}^T \gamma_{kn}(\tau) \text{diag} \left[ -\Sigma_{kn}^{-1} + \Sigma_{kn}^{-1}(\mathbf{x}_\tau - \boldsymbol{\mu}_{kn})(\mathbf{x}_\tau - \boldsymbol{\mu}_{kn})^T \Sigma_{kn}^{-1} \right] \\
&= \frac{1}{2} \sum_{\tau=1}^T p(\phi_t = \{s_j, m\}, \phi_\tau = \{s_k, n\} | \mathbf{X}, \mathcal{M}) \text{diag} \left[ -\Sigma_{kn}^{-1} + \Sigma_{kn}^{-1}(\mathbf{x}_\tau - \boldsymbol{\mu}_{kn})(\mathbf{x}_\tau - \boldsymbol{\mu}_{kn})^T \Sigma_{kn}^{-1} \right] \\
&\quad - \frac{\gamma_{jm}(t)}{2} \sum_{\tau=1}^T \gamma_{kn}(\tau) \text{diag} \left[ -\Sigma_{kn}^{-1} + \Sigma_{kn}^{-1}(\mathbf{x}_\tau - \boldsymbol{\mu}_{kn})(\mathbf{x}_\tau - \boldsymbol{\mu}_{kn})^T \Sigma_{kn}^{-1} \right] \\
&= \frac{1}{2} \sum_{\tau=1}^T \left[ p(\phi_t = \{s_j, m\}, \phi_\tau = \{s_k, n\} | \mathbf{X}, \mathcal{M}) - \gamma_{jm}(t) \gamma_{kn}(\tau) \right] \\
&\quad \text{diag} \left[ -\Sigma_{kn}^{-1} + \Sigma_{kn}^{-1}(\mathbf{x}_\tau - \boldsymbol{\mu}_{kn})(\mathbf{x}_\tau - \boldsymbol{\mu}_{kn})^T \Sigma_{kn}^{-1} \right]
\end{aligned} \tag{81}$$

Substituting (75) into (74) yields,

$$\begin{aligned}
\nabla_{\Sigma_{kn}} \nabla_{\Sigma_{jm}}^T \ln p(\mathbf{X} | \mathcal{M}) &= \frac{1}{4} \sum_{\tau=1}^T \left[ p(\phi_t = \{s_j, m\}, \phi_\tau = \{s_k, n\} | \mathbf{X}, \mathcal{M}) - \gamma_{jm}(t) \gamma_{kn}(\tau) \right] \\
&\quad \text{diag} \left[ -\Sigma_{kn}^{-1} + \Sigma_{kn}^{-1}(\mathbf{x}_\tau - \boldsymbol{\mu}_{kn})(\mathbf{x}_\tau - \boldsymbol{\mu}_{kn})^T \Sigma_{kn}^{-1} \right] \text{diag} \left[ -\Sigma_{kn}^{-1} + \Sigma_{kn}^{-1}(\mathbf{x}_\tau - \boldsymbol{\mu}_{kn})(\mathbf{x}_\tau - \boldsymbol{\mu}_{kn})^T \Sigma_{kn}^{-1} \right]^T \\
&\quad + \frac{1}{2} \sum_{\tau=1}^T \gamma_{jm}(\tau) \text{Diag} \left( \text{diag} \left[ \Sigma_j^{-1} \Sigma_j^{-1} - 2\Sigma_j^{-1} \Sigma_j^{-1}(\mathbf{x} - \boldsymbol{\mu}_j)(\mathbf{x} - \boldsymbol{\mu}_j)^T \Sigma_j^{-1} \right] \right)
\end{aligned} \tag{82}$$

$$\tag{83}$$