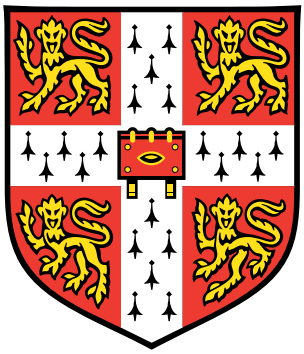
Kernel Methods for Text-Independent Speaker Verification

Chris Longworth

Cambridge University Engineering Department
and
Christ's College
February 23, 2010



Dissertation submitted to the University of Cambridge
for the degree of Doctor of Philosophy

Declaration

This dissertation is the result of my own work and includes nothing that is the outcome of work done in collaboration. It has not been submitted in whole or in part for a degree at any other university. Some of the work has been published previously in conference proceedings [129–132, 134] and a journal article [133]. The length of this thesis including references and footnotes is approximately 64,000 words. This thesis contains 28 figures and 23 tables.

Summary

In recent years, systems based on support vector machines (SVMs) have become standard for speaker verification (SV) tasks. An important aspect of these systems is the dynamic kernel. These operate on sequence data and handle the dynamic nature of the speech. In this thesis a number of techniques are proposed for improving dynamic kernel-based SV systems.

The first contribution of this thesis is the development of alternative forms of dynamic kernel. Several popular dynamic kernels proposed for SV are based on the Kullback-Leibler divergence between Gaussian mixture models. Since this has no closed-form solution, typically a matched-pair upper bound is used instead. This places significant restrictions on the forms of model structure that may be used. In this thesis, dynamic kernels are proposed based on alternative, variational approximations to the divergence. Unlike standard approaches, these allow the use of a more flexible modelling framework. Also, using a more accurate approximation may lead to performance gains.

The second contribution of this thesis is to investigate the combination of multiple systems to improve SV performance. Typically, systems are combined by fusing the output scores. For SVM classifiers, an alternative strategy is to combine at the kernel level. Recently an efficient maximum-margin scheme for learning kernel weights has been developed. In this thesis several modifications are proposed to allow this scheme to be applied to SV tasks. System combination will only lead to gains when the kernels are complementary. In this thesis it is shown that many commonly used dynamic kernels can be placed into one of two broad classes, derivative and parametric kernels. The attributes of these classes are contrasted and the conditions under which the two forms of kernel are identical are described. By avoiding these conditions gains may be obtained by combining derivative and parametric kernels.

The final contribution of this thesis is to investigate the combination of dynamic kernels with traditional static kernels for vector data. Here two general combination strategies are available: static kernel functions may be defined over the dynamic feature vectors. Alternatively, a static kernel may be applied at the observation level. In general, it is not possible to explicitly train a model in the feature space associated with a static kernel. However, it is shown in this thesis that this form of kernel can be computed by using a suitable metric with approximate component posteriors. Generalised versions of standard parametric and derivative kernels, that include an observation-level static kernel, are proposed based on this approach.

Keywords: speaker recognition; dynamic kernels; support vector machines; classifier combination.

Acknowledgements

First and foremost, I would like to thank my supervisor Mark Gales. Throughout my time at Cambridge he has consistently provided the right balance of support, criticism and encouragement and has offered countless helpful suggestions. I will always be grateful for the remarkable level of dedication that he shows to his students. I would also like to thank the Schiff Foundation for generously funding my studies and allowing me to travel to several international conferences. Without this support I would not have had the opportunity to study at Cambridge. I would also like to thank Christ's College for providing additional financial assistance towards conference expenses.

There are many people in the speech group to whom I owe thanks. I am indebted to Patrick Gosling and Anna Langley, both for their commitment to maintaining our computing systems and for their patience in handling my numerous queries and problems during the last few years. Thanks are also due to Rachel Fogg and Janet Milne, for the many occasions in which they have helped me with administrative issues. I would also like to thank everybody in the group who has made it such a pleasant and creative environment in which to work. In particular, special thanks are due to the following: Sarah Airey, Graeme Blackwood, Jamie Brunning, Bill Byrne, Catherine Breslin, Arantza Del Pozo, Federico Flego, Matt Gibson, Zeynep Inanoglu, Martin Layton, Hank Liao, CK Raut, Zoi Roupakia, Matt Shannon, Matt Stuttle, Rogier van Dalen, Phil Woodland and Kai Yu. I would also like to thank Mark, Rogier and Matt for their assistance in proofreading parts of this thesis.

Finally, I'd like to thank my parents and my brother Ricky, for putting up with me during the final days of thesis writing, and my young sister Rosie for allowing me the rent-free use of her tree house as a temporary office. Finally, my deepest thanks go to my partner Emily for her incredible level of support and for always being ready with a cup of tea, even when I didn't know that I needed one.

Acronyms

ASR	Automatic speech recognition
CAT	Cluster-adaptive training
CMLLR	Constrained MLLR
CRF	Conditional random fields
DAG	Directed acyclic graph
DBN	Dynamic Bayesian network
DCF	Detection cost function
DET	Detection error trade-off
EER	Equal error rate
EM	Expectation maximisation
GMM	Gaussian mixture model
GLDS	Generalised linear discriminant sequence
GDK	Generalised derivative kernel
GPK	Generalised parametric kernel
HCRF	Hidden conditional random fields
HMM	Hidden Markov model
KL	Kullback-Leibler
LLR	Log-likelihood ratio
MAP	Maximum a-posteriori
MFCC	Mel-frequency cepstral coefficients
MKL	Multiple kernel learning
ML	Maximum likelihood
MLLR	Maximum likelihood linear regression
MMI	Maximum mutual information
NAP	Nuisance attribute projection
PCA	Principal component analysis
NIST	National Institute of Standards and Technology
PLP	Perceptual linear prediction
SNERF	Syllable-based nonuniform extraction region features
SRE	Speaker recognition evaluation
SV	Speaker verification
SVM	Support vector machine
WCCN	Within-class covariance normalisation
UBM	Universal background model

Notation

These are the terms and notation used throughout this work.

Vectors and Matrices

$\boldsymbol{x}$	vector
x_j	scalar value that is the jth element of $\boldsymbol{x}$
$\boldsymbol{A}$	matrix
$\boldsymbol{A}^{\mathtt{T}}$	transpose of matrix $\boldsymbol{A}$
$\mathtt{Tr}[\boldsymbol{A}]$	trace of matrix $\boldsymbol{A}$
$\mathtt{diag}[\boldsymbol{A}]$	a diagonalised version of matrix $\boldsymbol{A}$
$\lvert\boldsymbol{A}\rvert$	determinant of matrix $\boldsymbol{A}$
$\boldsymbol{A}^{\text{-}1}$	inverse of matrix $\boldsymbol{A}$
$\boldsymbol{a}_i$	row vector that is i^{th} row of $\boldsymbol{A}$
a_{ij}	scalar value that is the element in row i and column j of $\boldsymbol{A}$

Variables, Symbols and Operations

$\approx$	approximately equal to
x	scalar quantity
$\underset{x}{\operatorname{argmax}} f(x)$	the value of x that maximises the value of $f(x)$
$\max_x f(x)$	the maximum value of $f(x)$ as x is varied
$\log(x)$	logarithm base e of x
$\exp(x)$	exponential of x
$\mathcal{E}\{f(x)\}$	the expected value of $f(x)$
$\sum_{n=1}^{N} a_n$	summation from $n = 1$ to N—that is, $a_1 + a_2 + \cdots + a_N$
$\nabla_{\boldsymbol{x}} f(\boldsymbol{x})$	vector derivative of a function $f(\boldsymbol{x})$ with respect to $\boldsymbol{x}$
$\int f(\boldsymbol{x})\, dx$	indefinite integral of $f(\boldsymbol{x})$ with respect to $\boldsymbol{x}$

$\boldsymbol{I}$	identity matrix
$\langle \boldsymbol{a}, \boldsymbol{b} \rangle$	inner product of $\boldsymbol{a}$ and $\boldsymbol{b}$

Observations

$\boldsymbol{O}$	sequence of observations
T	number of frames in a sequence of observations $\boldsymbol{O}$
t	time frame index
$\boldsymbol{o}_t$	speech observation vector at time t
D	number of dimensions of full feature vector
d	dimension index

GMM Parameters

$\boldsymbol{\lambda}$	set of current acoustic model parameters
$\boldsymbol{\lambda}^{(s)}$	set of acoustic model parameters associated with speaker s
M	number of GMM components
m	index for the mth component of an GMM
c_m	mixture weight associated with GMM component m
$\boldsymbol{\mu}_m$	mean of component m
$\boldsymbol{\Sigma}_m$	variance of component m

SVM Parameters

$\boldsymbol{\alpha}$	dual variables
$\boldsymbol{w}$	primal weight vector
$\boldsymbol{\xi}$	slack variables
b	bias parameter
C	regularisation constant

Kernels and Feature spaces

$k(\boldsymbol{x}_i, \boldsymbol{x}_j)$	static kernel function between vectors $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$
$K(\boldsymbol{O}_i, \boldsymbol{O}_j)$	dynamic kernel function between sequences $\boldsymbol{O}_i$ and $\boldsymbol{O}_j$
$\boldsymbol{\psi}(\boldsymbol{x}; \boldsymbol{\lambda})$	static feature expansion of vector $\boldsymbol{x}$
$\boldsymbol{\phi}(\boldsymbol{O}; \boldsymbol{\lambda})$	dynamic feature expansion of sequence $\boldsymbol{O}$

Parameter Estimation

$P(\theta_t = m \mid \boldsymbol{o}_t)$	posterior probability of component m at time t

Probability and Distributions

$f(\cdot)$	probability distribution
$P(\cdot)$	probability mass function
$p(\cdot)$	probability density function
$p(x, y)$	joint probability density function—that is, the probability density of x and y
$p(x \mid y)$	conditional probability density of x given y
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	multivariate Gaussian distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$
$\mathcal{N}(\boldsymbol{o}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$	probability of vector $\boldsymbol{o}$ given a multivariate Gaussian distribution
$KL(f_i \mid\mid f_j)$	Kullback-Leibler divergence between distributions f_i and f_j

Table of Contents

List of Tables

List of Figures

Chapter 1
Introduction

Many speech processing tasks, such as speech recognition [61], speaker [23, 172] and language [145] identification and gender detection [155] require the classification of speech utterances. Many speech classification techniques have been developed, however the most successful have been based on statistical classifiers [91]. Here, instead of applying hard-coded rules, a set of models are learned from the data that capture the relationships between the speech and the class labels. By analysing these models, unlabeled speech data may be classified.

Statistical techniques for speech classification generally fall into one of two broad categories: generative and discriminative classification schemes [17]. In the first approach, a generative model is trained to approximate the probability density function of the speech observations associated with each class. A classification decision is then made using a combination of Bayes' rule and Bayes' decision rule. Discriminative schemes are an alternative classification approach. These include discriminative models, which are trained to model the distribution of the class posteriors, and discriminative classifiers, which model the location of a decision boundary that separates the classes. A popular example of this last approach is the support vector machine (SVM) [198], which has been successfully applied to many different applications [10, 21, 33].

1.1 Kernel-based speech classification

A useful property shared by several forms of discriminative classifier, including the support vector machine, is that they can be kernelised [181]. Here, all references to the data are in the form of an inner product between two data examples. Instead of applying an inner product directly, a kernel function may be defined that implicitly maps each data example into a high-dimensional feature space where the inner product is evaluated. Hence, through

the selection of a suitable kernel function, classification may be performed in a space in which the classes are more easily separated. Many kernels used with discriminative classifiers only handle data of a fixed-dimensionality. In this thesis, these are referred to as *static kernels*. In contrast, speech utterances are typically parameterised as variable-length sequences of observation vectors. This has led to the use of *dynamic kernels* [204], also known as sequence kernels. These dynamic kernels map variable-length sequences into a fixed dimension feature space in which the inner product, or static kernel, can be computed. Hence, through the use of an appropriate dynamic kernel, discriminative classifiers can used to classify variable-length sequences such as speech.

The development of dynamic kernels is an important area of research. The features extracted from each sequence by a dynamic kernel should emphasize information useful for discriminating between classes and remove any non-useful information that may affect the classification decision. Ideally, dynamic kernels should also be efficient to compute. Many forms of dynamic kernel have been proposed for classification of speech sequences. An early example of this type of kernel is the generalised linear discriminant sequence (GLDS) kernel [24, 142]. Here, each observation vector is initially mapped into a high-dimensional space using a static kernel. A fixed dimensional set of features is then obtained by taking the mean of the expanded observations.

More recent approaches have examined dynamic kernels based on generative models, examples include the Fisher kernel [90, 205], the Gaussian mixture model (GMM) supervector kernel [28], the maximum likelihood linear regression (MLLR) kernel [189] and the cluster-adaptive training (CAT) kernel [211]. These kernels use the generative model to extract structure from the utterance. Despite the large number of dynamic kernels that have been proposed in the literature, there have so far been few attempts to identify relationships between these kernels. One of the contributions of this thesis is to show that many commonly used dynamic kernels can be placed into one of two broad classes. In this thesis, these are referred to as *parametric kernels* and *derivative kernels*. The two types of kernel are closely related and under certain conditions the features obtained can be shown to be identical. By avoiding these conditions, kernels may be obtained that express complementary information. The framework may also be used to motivate new forms of dynamic kernel.

1.2 Speaker verification

In this thesis, discriminative classification schemes using dynamic kernels are applied to the task of speaker verification (SV). Here the objective is to decide, given an utterance of speech and an associated identity claim, whether the speech was uttered by the claimed target speaker or by an imposter. Speaker verification has several practical applications, including biometrics for authentication [113], forensics [31], and as a component in speech recognition systems [215]. This thesis examines the text-independent task, where users of the system are not constrained to speak a particular sequence of words. Text-independent SV has been the focus of considerable research effort and is the subject of annual NIST speaker recognition evaluations [144]. The task is complicated by the fact that speaker-dependent characteristics of the speech may be masked by environmental or channel conditions. Additionally, the distinctive characteristics of each speaker's voice may also vary between recording sessions, due to changes in emotional state, health or age.

Traditional approaches to text-independent SV have made use of generative models, normally Gaussian mixture models, to approximate the distribution of the speech associated with each target speaker [170]. A standard approach is then to base a classification decision on the log-likelihood ratio between the target speaker model and a universal background model trained to represent all speakers [169]. Recent approaches have examined how to apply support vector machines to the SV task using dynamic kernels [24, 205]. SVMs have generally been found to outperform traditional log-likelihood based approaches [28, 48, 190] and a variety of dynamic kernels have been successfully applied to the SV task [28, 189, 205, 211].

The performance of an SVM-based speaker verification system depends on a number of factors. These include: the form of speech parameterisation used, the scheme used to estimate the generative model parameters, the choice of dynamic kernel and any schemes used to normalise the classifier output. This thesis concentrates on the aspect of the dynamic kernel and proposes a number of techniques that may be used to obtain dynamic kernels that are more effective at discriminating between speakers. Although this thesis focuses on the application to SVM-based speaker verification, the approaches described in this thesis are suitable for any task involving sequences of continuous observations where dynamic kernels can be applied.

Many state-of-the-art SV systems make use of distributional kernels, such as the GMM-supervector [28] and the non-linear GMM-supervector kernels [48]. For these kernels, a GMM is trained to represent each utterance in the dataset. The kernel function between a pair of utterances is then derived from a Kullback-Leibler (KL) divergence measure between the corresponding models. However, many standard forms of distributional kernel require that all GMMs have the same structure and that they are adapted from a single background model. This restriction may limit the performance of an SV system. In this thesis, alternative forms of dynamic kernel are proposed that are derived from two variational approximations to the KL divergence. Unlike standard approaches, these variational kernels do not restrict all GMMs to have the same structure allowing more complex training schemes to be used. For example, GMMs may be adapted from a range of gender or noise condition-dependent background models. Additionally, the use of a kernel that more accurately reflects the true KL divergence between GMMs may lead to gains.

An alternative approach that may yield more discriminative kernels is to combine multiple complementary dynamic kernels. This requires selecting a weight for each kernel. An efficient scheme for learning a suitable set of weights was proposed in [168, 188]. Here kernel weights are obtained using a maximum-margin criterion and trained using a standard SVM implementation. In this thesis, a number of modifications to this scheme are proposed to allow this scheme to be used for SV. The standard scheme has a known tendency to find sparse kernel weightings. In this thesis, a regularisation term is applied to the objective function to allows the user to select the desired level of sparsity. Kernel weights are also tied over multiple target speakers to obtain more robust parameter estimates.

The final contribution of this thesis is to combine dynamic kernels with traditional static kernels to potentially yield more discriminative features. Two general approaches are available. In the first approach, dynamic feature vectors are obtained from each utterance as before. Then, instead of taking the inner product of the feature vectors, a static kernel function is evaluated. For the second approach, instead of calculating dynamic features using the original observations, each observation is initially mapped into the higher-dimensional space associated with a static kernel. Dynamic features are then calculated based on the expanded observations. Hence, verification may be based on higher order observation level

features while exploiting the nature of the generative model to obtain a fixed set of features. In this thesis, generalised versions of standard derivative and parametric kernels are derived that include a observation-level static kernel. These kernels generalise standard derivative and parametric kernels respectively, and also the GLDS kernel, providing a theoretical link between these forms of kernel.

1.3 Thesis organisation

This thesis is organised into three main parts. Chapters 2-4 give a general background to speech classification techniques and describe how a speaker verification system may be constructed. Chapters 5-7 contain the main contributions of the thesis. Alternative forms of dynamic kernel are proposed based on variational approximations to the KL divergence. Schemes are described for establishing whether certain forms of dynamic kernels are complementary and finally techniques for combining multiple static and dynamic kernels are presented. Chapters 8-9 present detailed experimental results and offer a number of conclusions and suggestions for future work.

A detailed breakdown of the structure of the thesis is given below.

Chapter 2 provides an introduction to generative and discriminative approaches for classifying speech. Various generative classification schemes and discriminative functions are introduced. The use of static kernels within a discriminative framework is also described.

Chapter 3 introduces dynamic kernels and shows how they can be applied to classify variable-length sequences. Various forms of dynamic kernel suitable for classifying sequences of discrete and continuous observations are described. Several forms of distributional kernel are also introduced and a scheme described for applying these to sequence classification.

Chapter 4 describes how the theory introduced in the previous two chapters may be applied to build a state-of-the-art speaker verification system. Systems based on both generative models and discriminative classifiers are discussed.

Chapter 5 proposes alternative forms of dynamic kernels, suitable for SVM-based speaker verification. Two recently developed variational approximations to the KL divergence between Gaussian mixture models are first described. Then variational dynamic kernels, derived from these approximations, are proposed. The properties of the variational kernels are then contrasted with standard forms of dynamic kernel.

Chapter 6 examines the combination of multiple dynamic kernels to improve SV performance. Two approaches, score- and kernel-level combination, are first described and contrasted. Then modifications to a scheme for automatically weighting kernels are proposed for the SV task. Two general categories of dynamic kernel, derivative and parametric kernels, are then introduced and the conditions under which they will be complementary described.

Chapter 7 examines the combination of dynamic kernels with traditional static kernels. Combination of static kernels at both the observation level and dynamic feature vector level is described. Generalised derivative and parametric kernels, formed by combining observation-level static kernels with standard derivative and parametric kernels respectively, are proposed and methods for evaluating these kernels described. Finally, two schemes for handling limited amounts of data, data partitioning and speaker tying, are described.

Chapter 8 presents detailed experimental results on the 2002 NIST SRE dataset. Variational kernels are compared against standard forms of dynamic kernel. Then combinations of multiple dynamic kernels are evaluated. Finally combinations of static and dynamic kernels are evaluated.

Chapter 9 gives a summary of the thesis and proposes topics for future research.

Chapter 2
Approaches for Speech Classification

There are many tasks that require assigning utterances of speech to a particular class. These include isolated word recognition, language identification as well as gender/accent detection. Speaker verification, discussed in detail in chapter 4, is another example of a classification task. Modern schemes for classifying speech data generally fall into one of two broad approaches, based on either generative or discriminative classification schemes. Both of these approaches are described in this chapter.

In the first approach, generative classification, statistical models are trained to assign the likelihood of a speech utterance given a particular class. Bayes' decision rule is then applied to classify the utterance. The choice of statistical model used is important and the appropriate form of model to use is usually task and data dependent. This chapter introduces two forms of model suitable for speech classification, the Gaussian mixture model (GMM) and the hidden Markov model (HMM). Both of these models are examples of parametric approaches, where the probability densities are determined by an associated set of model parameters. This chapter describes several techniques that may be applied to obtain suitable parameter estimates for these models.

Discriminative schemes are an alternative approach to speech classification. Unlike generative approaches, these attempt to model either the class boundaries or posterior probabilities directly. Discriminative schemes described in this chapter include discriminative models, such as the conditional random field (CRF) [116] and hidden CRF [165], and distance-based discriminative functions such the support vector machine (SVM) [198]. Discriminative functions are normally applied to static data, where all examples contain the same number of features.

However, through the use of an appropriate kernel function, introduced here and described further in chapter 3, these techniques may also be applied to sequence data such as speech.

2.1 Generative classification schemes

Generative classification is a popular statistical approach that is suitable for classifying both static data and sequence data, such as speech. Generative classification schemes for speech make use of statistical models to represent the probability density function associated with a sequence of observations. These *generative models* can then be used to randomly generate observations according to the probability density function associated with the model. Alternatively, given a speech utterance composed of T observations, $\boldsymbol{O} = \{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_T\}$, where $\boldsymbol{o}_t$ is a vector of D features extracted from the speech signal at time t, generative models can be used to estimate the likelihood of the sequence given the associated density function. It is this ability that allows generative models to be used for speech classification.

For speech classification tasks, the objective is to label $\boldsymbol{O}$ as belonging to a particular class $\omega \in \Omega$, where Ω is the set of all potential class labels. If the conditional likelihood of an observation sequence given a class $p(\boldsymbol{O}|\omega)$ is known, Bayes' rule can be used to convert the likelihood into a class posterior [56].

$$P(\omega|\boldsymbol{O}) = \frac{P(\omega)p(\boldsymbol{O}|\omega)}{p(\boldsymbol{O})} \tag{2.1}$$

where $P(\omega)$ is a prior distribution over the class labels ω and $p(\boldsymbol{O})$ is the class-independent likelihood of $\boldsymbol{O}$. This may be calculated by marginalising over all classes, $p(\boldsymbol{O}) = \sum_{\omega\in\Omega} P(\omega)p(\boldsymbol{O}|\omega)$. A label $y \in \Omega$ can then be assigned to $\boldsymbol{O}$ using Bayes' decision rule [56].

$$\begin{aligned} y &= \operatorname*{argmax}_{\omega\in\Omega} \{P(\omega|\boldsymbol{O})\} \\ &= \operatorname*{argmax}_{\omega\in\Omega} \left\{ \frac{P(\omega)p(\boldsymbol{O}|\omega)}{p(\boldsymbol{O})} \right\} \end{aligned} \tag{2.2}$$

In the binary case, where $\Omega = \{\omega_1, \omega_2\}$, Bayes' decision rule can be expressed as

$$\log p(\boldsymbol{O}|\omega_1) - \log p(\boldsymbol{O}|\omega_2) \begin{array}{c} y=\omega_1 \\ > \\ < \\ y=\omega_2 \end{array} b \tag{2.3}$$

where b is a classification threshold defined by the class priors, $b = \log P(\omega_2) - \log P(\omega_1)$. In order to classify utterances using Bayes' decision rule, a method is needed of estimating $p(\boldsymbol{O}|\omega)$. This is generally achieved by training a generative model $p(\boldsymbol{O}; \boldsymbol{\lambda}^{(\omega)})$ to represent the conditional likelihood of $\boldsymbol{O}$ given class ω, where $\boldsymbol{\lambda}^{(\omega)}$ is a set of class-dependent model parameters associated with the generative model. Two common forms of generative model for speech classification are Gaussian mixture models (GMMs) and hidden Markov models (HMMs). These are described in the following subsections.

2.1.1 Gaussian mixture models

One of the most popular forms of distribution for modelling speech observations is the Gaussian distribution. For an observation $\boldsymbol{o}$, the Gaussian likelihood is given by

$$\mathcal{N}(\boldsymbol{o};\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^D\,|\boldsymbol{\Sigma}|}}\exp\left(-\frac{1}{2}\left(\boldsymbol{o}-\boldsymbol{\mu}\right)^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}\left(\boldsymbol{o}-\boldsymbol{\mu}\right)\right) \tag{2.4}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are respectively the mean and covariance of the distribution. The Gaussian distribution has a number of properties that make it attractive for acoustic modelling. For example, it is continuous, symmetric and differentiable. However, in practice speech data can not always be closely approximated by a Gaussian distribution. For example, due to differences in gender, speech distributions will often be bimodal. Also, speech distributions are rarely exactly symmetric.

A form of generative model commonly used to model speech is the Gaussian Mixture Model (GMMs). The GMM is a latent variable extension of the Gaussian distribution comprised of multiple Gaussian components each with distinct mean and covariance $\{\boldsymbol{\mu}_m,\boldsymbol{\Sigma}_m\}$, and an associated prior probability c_m, also known as the component weight. GMMs are able to approximate arbitrary probability distributions, given a sufficient number of Gaussian components. The likelihood of an observation $\boldsymbol{o}$ given an M-component GMM with model parameters $\boldsymbol{\lambda} = (\boldsymbol{\mu}_1,\ldots,\boldsymbol{\mu}_M,\boldsymbol{\Sigma}_1,\ldots,\boldsymbol{\Sigma}_M,c_1,\ldots c_M)$ is a weighted sum of the individual component likelihoods.

$$p(\boldsymbol{o};\boldsymbol{\lambda}) = \sum_{m=1}^{M} c_m\mathcal{N}(\boldsymbol{o};\boldsymbol{\mu}_m,\boldsymbol{\Sigma}_m) \tag{2.5}$$

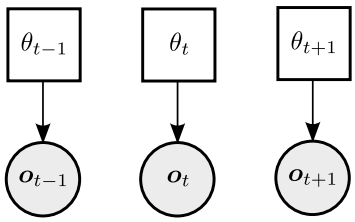


Figure 2.1: A dynamic Bayesian network corresponding to a Gaussian mixture model.

A dynamic Bayesian network corresponding to a GMM is shown in figure 2.1. Here, square nodes represent discrete variables and round nodes represent continuous variables. The lack of a edge from one node to another indicates that the second node is conditionally independent of the first. It can be seen from the figure that successive observations are independently distributed and depend only on the latent component θ. Thus, the likelihood of an utterance $\boldsymbol{O}$ is the product of the individual observation likelihoods.

$$p(\boldsymbol{O};\boldsymbol{\lambda}) = \prod_{t=1}^{T} p(\boldsymbol{o}_t;\boldsymbol{\lambda}) \tag{2.6}$$

Several parameter estimation schemes for generative models are described in section 2.1.3. Many of these schemes require calculating the posterior probability $P(\theta_t = m|\boldsymbol{o}_t)$ of the

latent Gaussian component m given the observation $\boldsymbol{o}_t$. Since observations are generated independently, for GMMs this can be calculated using

$$P(\theta_t = m|\boldsymbol{o}_t) = \frac{c_m \mathcal{N}(\boldsymbol{o}_t; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)}{\sum_{n=1}^{M} c_n \mathcal{N}(\boldsymbol{o}_t; \boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n)} \tag{2.7}$$

For a D-dimensional feature vector, a full covariance matrix has $\mathcal{O}(D^2)$ parameters. When the number of components is large, modelling the shape of the distribution using full-covariances is expensive and requires a large amount of training data. Typically, the elements of the observation vector are assumed to be independent, and diagonal or block-diagonal covariance matrices are used. Approaches based on parameter tying, such as semi-tied covariances [64] or SPAM (Subspace Precision and Mean) [7], may also be used.

2.1.2 Hidden Markov models

For many speech classification tasks, such as ASR, it is desirable to model changes in the speech signal over time. Static models such as Gaussian mixture models are unsuitable for this as they assume that successive observations are independent. Hidden Markov models (HMMs) [167] are dynamic generative models that have become the dominant form of generative model used in state of the art ASR systems [69].

The hidden Markov model is a finite state machine composed of a fixed number of discrete states including non-emitting initial and end states. The HMM starts in the initial state at time $t = 0$. At each subsequent time instance t the HMM transitions into a new state $\theta_t = j$. An observation is then generated based on the output distribution, $b_j(\boldsymbol{o}_t) = p(\boldsymbol{o}_t|\theta_t)$, associated with the new state. Only $\{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_T\}$ are observable, the corresponding sequence of states $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_T\}$ is unobserved, or latent. The HMM makes the following two assumptions.

- **Conditional independence**: the likelihood of generating an observation $\boldsymbol{o}_t$ depends only on the current state θ_t.
- **First order Markov assumption:** the probability of transitioning to a particular state is dependent only on the previous state.

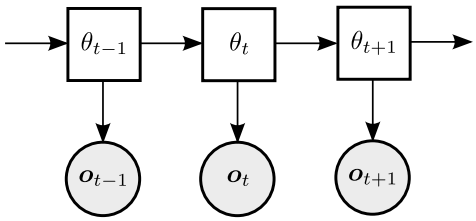


Figure 2.2: A dynamic Bayesian network corresponding to a hidden Markov model.

These dependency assumptions are illustrated by the dynamic Bayesian network in figure 2.2. The horizontal arrows model the first order Markov assumption, while the vertical arrows model the dependence of the observations on the current state. The state transition probabilities are often compactly represented using a matrix $\boldsymbol{a}$, where $a_{ij} = P(\theta_t = j|\theta_{t-1} =$

i). The output distribution $b_j(\boldsymbol{o}_t)$ associated with each state j can take any form. For speech applications, where the observations are continuous, $b_j(\boldsymbol{o}_t)$ is typically a diagonal covariance Gaussian mixture model. In this case the HMM is defined by a set of parameters $\boldsymbol{\lambda} = \{\boldsymbol{a}, \boldsymbol{c}, \boldsymbol{\mu}, \boldsymbol{\Sigma}\}$, which include a distinct prior c_{jm}, mean $\boldsymbol{\mu}_{jm}$ and covariance $\boldsymbol{\Sigma}_{jm}$ for each component m of state j. A particular form of hidden Markov model known as a *left-to-right* HMM is also commonly used. In a left-to-right HMM the transition probabilities between states are constrained to ensure that the states can be ordered such that the only non-zero transition probabilities are from a state to itself or to its immediate successor. An example of a left-to-right hidden Markov model with three emitting states is shown in figure 2.3.

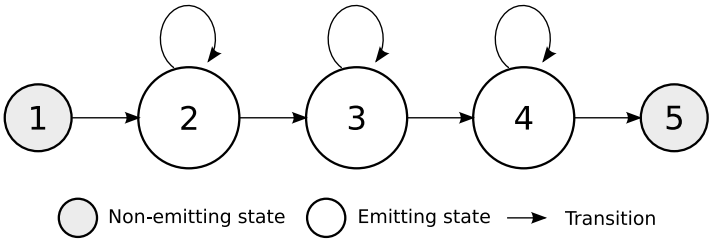


Figure 2.3: A three-state left-to-right hidden Markov model.

Likelihood calculations for HMMs are less straightforward than for GMMs. This is because successive observations are not assumed to be independent. The posterior likelihood of an utterance $\boldsymbol{O}$ given $\boldsymbol{\lambda}$ is obtained by marginalising over all possible latent state sequences Θ.

$$p(\boldsymbol{O}; \boldsymbol{\lambda}) = \sum_{\boldsymbol{\theta} \in \Theta} \prod_{t=1}^{T} P(\theta_t | \theta_{t-1}) p(\boldsymbol{o}_t | \theta_t) \tag{2.8}$$

It is often impractical to calculate equation 2.8 using an explicit summation over all latent state sequences. However, due to the conditional independence and first order-Markov assumptions, dynamic programming algorithms may be used to efficiently calculate both the likelihood of the observation sequence and the posterior probability of each state given the observations. One such algorithm is the forward-backward algorithm [97]. Here, the *forward-probability* $\alpha_j(t)$ is the sum of the likelihoods of all partial paths ending in state j at time t. For a N-state HMM, it can be calculated recursively using

$$\begin{aligned} \alpha_j(t) &= p(\boldsymbol{o}_1, \ldots, \boldsymbol{o}_t, \theta_t = j; \boldsymbol{\lambda}) \\ &= \left[\sum_{i=2}^{N-1} \alpha_i(t-1) a_{ij} \right] b_j(\boldsymbol{o}_t) \end{aligned} \tag{2.9}$$

where the initial forward probability at time $t = 1$ is given by

$$\alpha_j(1) = \alpha_{1j} b_j(\boldsymbol{o}_1) \;\; \forall j \neq 1 \tag{2.10}$$

The likelihood of the utterance $p(\boldsymbol{O}; \boldsymbol{\lambda})$ is given by $\alpha_N(T)$, the forward probability of the end state N at time T. Similarly, the *backward-probability* $\beta_i(t)$ is the conditional probability

that the model will generate the rest of the sequence from time t given $\theta_t = i$. It is defined recursively using

$$\begin{aligned} \beta_i(t) &= p(\boldsymbol{o}_{t+1}, .., \boldsymbol{o}_T | \theta_t = i; \boldsymbol{\lambda}) \\ &= \sum_{j=2}^{N-1} a_{ij} b_j(\boldsymbol{o}_{t+1}) \beta_j(t+1) \end{aligned} \tag{2.11}$$

where the backward probabilities at time T are given by

$$\beta_i(T) = a_{iN} \tag{2.12}$$

Similarly, the likelihood $p(\boldsymbol{O}; \boldsymbol{\lambda})$ is given by $\beta_1(0)$, the backward probability of the initial state at time $t = 0$. Note that this likelihood may be obtained using either the forward or the backward probabilities. It is not necessary to calculate both. Using the definition of the forward and backward probabilities, the posterior probability $P(\theta_t = j | \boldsymbol{O}; \boldsymbol{\lambda})$ of being in state j at time t can be calculated.

$$P(\theta_t = j | \boldsymbol{O}; \boldsymbol{\lambda}) = \frac{\alpha_j(t) \beta_j(t)}{p(\boldsymbol{O}; \boldsymbol{\lambda})} \tag{2.13}$$

The state posterior probabilities are required for various parameter training and adaptation algorithms, described in the following subsections. Although the conditional independence and first order Markov assumptions associated with the HMM are known to be incorrect for speech, the HMM has nonetheless been found to be a useful generative model for speech classification. Several alternative forms of model have since been developed that attempt to relax some of these independence assumptions. These include factor-analysed HMMs [175], buried Markov models [16], and switching linear dynamical systems [176].

2.1.3 Parameter estimation

A standard approach for training generative models is to select values $\boldsymbol{\lambda}^*$ for the model parameters that maximise some training criterion $\mathcal{F}(\boldsymbol{\lambda} | \mathcal{O})$ estimated over a training dataset $\mathcal{O} = \{\boldsymbol{O}_1, \ldots, \boldsymbol{O}_N\}$.

$$\boldsymbol{\lambda}^* = \underset{\boldsymbol{\lambda}}{\text{argmax}} \{\mathcal{F}(\boldsymbol{\lambda} | \mathcal{O})\} \tag{2.14}$$

A number of different criteria have been proposed for training generative model parameters. By far the most popular training criterion is maximum likelihood (ML) estimation. Here, $\boldsymbol{\lambda}$ is selected to maximise the likelihood of the training data. Alternatively, discriminative training criteria can be used. These seek to optimise the model parameters with respect to the expected error rate. Proposed criteria include maximum mutual information (MMI) [9], conditional maximum likelihood (CML) [152], minimum Bayes risk (MBR) [99] and minimum classification error (MCE) [95, 136]. For ASR, a number additional discriminative training criteria have been proposed. These include minimum phone error [162, 163] and minimum word error [105, 151]. Two of the most popular training criteria, ML and MMI, are described below.

2.1.3.1 Maximum likelihood

Maximum likelihood estimation [13, 167] is a standard approach for training the parameters of generative models. ML operates by selecting model parameters $\boldsymbol{\lambda}^*$ such that the likelihood of the training data is maximised. For a training set $\mathcal{O} = \{\boldsymbol{O}_1, \ldots, \boldsymbol{O}_N\}$, this is equivalent to maximising the log-likelihood of the training data using the objective function

$$\mathcal{F}^{\texttt{ML}}(\boldsymbol{\lambda}|\mathcal{O}) = \sum_{i=1}^{N} \log p(\boldsymbol{O}_i; \boldsymbol{\lambda}) \tag{2.15}$$

The ML parameter estimates are obtained by maximising equation 2.15 with respect to the model parameters $\boldsymbol{\lambda}$. For GMM and HMM models, differentiating equation 2.15 with respect to $\boldsymbol{\lambda}$ does not yield simple closed form estimates for the optimal model parameters, due to the latent sequence $\boldsymbol{\theta}$. Instead expectation maximisation (EM) [50] can be used to update the model parameters. Rather than optimising the ML objective function directly, instead an *auxiliary function* is iteratively optimised. For the ML criterion the auxiliary function is

$$\mathcal{Q}^{\texttt{ML}}(\boldsymbol{\lambda}^{(k+1)}, \boldsymbol{\lambda}^{(k)}) = \sum_{\boldsymbol{\theta} \in \Theta} P(\boldsymbol{\theta}|\mathcal{O}; \boldsymbol{\lambda}^{(k)}) \log p(\boldsymbol{\theta}, \mathcal{O}; \boldsymbol{\lambda}^{(k+1)}) \tag{2.16}$$

where $\boldsymbol{\lambda}^{(k)}$ are parameter estimates at iteration k, and the summation is defined over all latent sequences Θ. At each iteration, selecting an updated model parameter set $\boldsymbol{\lambda}^{(k+1)}$ that increases $Q^{\texttt{ML}}(\boldsymbol{\lambda}^{(k+1)}, \boldsymbol{\lambda}^{(k)})$ is guaranteed to lead to an increase in the ML criterion, due to the inequality

$$\mathcal{F}^{\texttt{ML}}(\boldsymbol{\lambda}^{(k+1)}|\mathcal{O}) - \mathcal{F}^{\texttt{ML}}(\boldsymbol{\lambda}^{(k)}|\mathcal{O}) \geq Q^{\texttt{ML}}(\boldsymbol{\lambda}^{(k+1)}, \boldsymbol{\lambda}^{(k)}) - Q^{\texttt{ML}}(\boldsymbol{\lambda}^{(k)}, \boldsymbol{\lambda}^{(k)}) \tag{2.17}$$

EM is therefore a two-step procedure where, at each iteration, the auxiliary function is first calculated (the E step) and then new parameters selected that maximise this function (the M step). By repeating this procedure, optimal parameters $\boldsymbol{\lambda}^*$ will be obtained that locally maximise the ML criterion over the training data. For GMMs, the latent sequence $\boldsymbol{\theta}$ is the sequence of component indices that generate the observations. Omitting the summation over all utterances for clarity, the EM estimate for the new parameter set $\boldsymbol{\lambda}^{(k+1)}$ that maximises the ML auxiliary function at each iteration is given by

$$\boldsymbol{\mu}_m^{(k+1)} = \frac{\sum_{t=1}^{T} P(\theta_t = m|\boldsymbol{o}_t; \boldsymbol{\lambda}^{(k)}) \boldsymbol{o}_t}{\sum_{t=1}^{T} P(\theta_t = m|\boldsymbol{o}_t; \boldsymbol{\lambda}^{(k)})} \tag{2.18}$$

$$\boldsymbol{\Sigma}_m^{(k+1)} = \frac{\sum_{t=1}^{T} P(\theta_t = m|\boldsymbol{o}_t; \boldsymbol{\lambda}^{(k)}) (\boldsymbol{o}_t - \boldsymbol{\mu}_m^{(k+1)})(\boldsymbol{o}_t - \boldsymbol{\mu}_m^{(k+1)})^{\texttt{T}}}{\sum_{t=1}^{T} P(\theta_t = m|\boldsymbol{o}_t; \boldsymbol{\lambda}^{(k)})} \tag{2.19}$$

$$c_m^{(k+1)} = \frac{\sum_{t=1}^{T} P(\theta_t = m|\boldsymbol{o}_t; \boldsymbol{\lambda}^{(k)})}{\sum_{m=1}^{M} \sum_{t=1}^{T} P(\theta_t = m|\boldsymbol{o}_t; \boldsymbol{\lambda}^{(k)})} \tag{2.20}$$

where $P(\theta_t = m|\boldsymbol{o}_t; \boldsymbol{\lambda}^{(k)})$ is the posterior probability of component m generating observation $\boldsymbol{o}_t$ given the parameter set $\boldsymbol{\lambda}^{(k)}$. This can be calculated using equation 2.7.

For HMMs, the Baum-Welch algorithm [13] is an implementation of EM. Here $\boldsymbol{\theta}$ is a latent sequence over both states and components. Again omitting the summation over all utterances, the HMM parameter estimates at iteration $k+1$ are given by

$$a_{ij}^{(k+1)} = \frac{\sum_{t=2}^{T} \zeta_{ij}(t)}{\sum_{t=2}^{T} \sum_{j=1}^{N} \zeta_{ij}(t)} \tag{2.21}$$

$$\boldsymbol{\mu}_{jm}^{(k+1)} = \frac{\sum_{t=1}^{T} P(\theta_t = \{j,m\}|\boldsymbol{O};\boldsymbol{\lambda}^{(k)})\boldsymbol{o}_t}{\sum_{t=1}^{T} P(\theta_t = \{j,m\}|\boldsymbol{O};\boldsymbol{\lambda}^{(k)})} \tag{2.22}$$

$$\boldsymbol{\Sigma}_{jm}^{(k+1)} = \frac{\sum_{t=1}^{T} P(\theta_t = \{j,m\}|\boldsymbol{O};\boldsymbol{\lambda}^{(k)})(\boldsymbol{o}_t - \boldsymbol{\mu}_{jm}^{(k+1)})(\boldsymbol{o}_t - \boldsymbol{\mu}_{jm}^{(k+1)})^{\mathsf{T}}}{\sum_{t=1}^{T} P(\theta_t = \{j,m\}|\boldsymbol{O};\boldsymbol{\lambda}^{(k)})} \tag{2.23}$$

$$c_{jm}^{(k+1)} = \frac{\sum_{t=1}^{T} P(\theta_t = \{j,m\}|\boldsymbol{O};\boldsymbol{\lambda}^{(k)})}{\sum_{m=1}^{M} \sum_{t=1}^{T} P(\theta_t = \{j,m\}|\boldsymbol{O};\boldsymbol{\lambda}^{(k)})} \tag{2.24}$$

where $P(\theta_t = \{j,m\}|\boldsymbol{O};\boldsymbol{\lambda}^{(k)})$ is the posterior probability of being in state j and component m at time t, given parameter estimates $\boldsymbol{\lambda}^{(k)}$. This may be estimated using the forward-backward algorithm. $\zeta_{ij}(t)$ is the posterior probability of being in state i at time $t-1$ and state j at time t, again this can be calculated using the forward-backward probabilities.

$$\begin{aligned} \zeta_{ij}(t) &= P(\theta_{t-1} = i, \theta_t = j|\boldsymbol{O};\boldsymbol{\lambda}^{(k)}) \\ &= \frac{\alpha_i(t-1)a_{ij}b_j(\boldsymbol{o}_t)\beta_j(t)}{\alpha_N(T)} \end{aligned} \tag{2.25}$$

2.1.3.2 Maximum mutual information

Discriminative training criteria provide a method of directly optimising the model parameters to reduce the classification error. One of the most popular discriminative criteria is the maximum mutual information (MMI) criterion [9]. For a training set $\mathcal{O} = \{\boldsymbol{O}_1, \ldots, \boldsymbol{O}_N\}$ with associated class labels $\mathcal{Y} = \{y_1, \ldots, y_N\}$, MMI training explicitly seeks to maximise the mutual information between the label sequence and the information extracted from the observation sequence by a model with parameters $\boldsymbol{\lambda}$. This can be expressed in terms of generative model likelihoods $p(\boldsymbol{O};\boldsymbol{\lambda}^{(\omega)})$ and class priors $P(\omega)$, where $\boldsymbol{\lambda}^{(\omega)}$ is a set of generative model parameters associated with class $\omega \in \Omega$.

$$\mathcal{F}^{\texttt{MMI}}(\boldsymbol{\lambda}|\mathcal{O}) = \sum_{i=1}^{N} \log \frac{p(y_i, \boldsymbol{O}_i;\boldsymbol{\lambda}^{(y_i)})}{P(y_i)p(\boldsymbol{O}_i;\boldsymbol{\lambda}^{(y_i)})} \tag{2.26}$$

When the prior probability of the classes is fixed, MMI training is equivalent to conditional maximum likelihood training [116, 152], used in section 2.2.2 for estimating HCRF parameters. Here the objective function maximises the posterior of the correct class and has the form

$$\begin{aligned} \mathcal{F}^{\texttt{CML}}(\boldsymbol{\lambda}|\mathcal{O}) &= \log P(\mathcal{Y}|\mathcal{O};\boldsymbol{\lambda}) \\ &= \sum_{i=1}^{N} \log P(y_i|\boldsymbol{O}_i;\boldsymbol{\lambda}) \end{aligned} \tag{2.27}$$

By applying Bayes' rule, the MMI criterion may also be expressed in the form of a summation over all class labels.

$$\mathcal{F}^{\texttt{MMI}}(\boldsymbol{\lambda}|\mathcal{O}) = \sum_{i=1}^{N} \log \left(\frac{P(y_i)p^{\kappa}(\boldsymbol{O}_i;\boldsymbol{\lambda}^{(y_i)})}{\sum_{\omega\in\Omega} P(\omega)p^{\kappa}(\boldsymbol{O}_i;\boldsymbol{\lambda}^{(\omega)})} \right) \tag{2.28}$$

Equation 2.28 includes a scaling factor, κ. This allows the less likely classes to contribute to the criterion [177]. For ASR tasks, κ is usually set to the inverse of the language model scaling factor.

In common with other discriminative estimation schemes it is difficult to obtain a suitable auxiliary function, such that increasing the auxiliary function is guaranteed to not decrease the MMI objective function. An extended Baum-Welch (EBW) algorithm was proposed in [153, 210]. This includes a smoothing term to ensure that the auxiliary function remains convex. Alternatively, a weak-sense auxiliary function may be used [164]. This shares the same gradient around the the current parameter estimates, but increasing a weak-sense auxiliary function does not guarantee to increase the original function. For MMI training a suitable weak-sense auxiliary function is

$$\mathcal{Q}^{\texttt{MMI}}(\boldsymbol{\lambda}^{(k+1)},\boldsymbol{\lambda}^{(k)}) = \mathcal{Q}^{\texttt{num}}(\boldsymbol{\lambda}^{(k+1)},\boldsymbol{\lambda}^{(k)}) - \mathcal{Q}^{\texttt{den}}(\boldsymbol{\lambda}^{(k+1)},\boldsymbol{\lambda}^{(k)}) + \mathcal{Q}^{\texttt{sm}}(\boldsymbol{\lambda}^{(k+1)},\boldsymbol{\lambda}^{(k)}) \tag{2.29}$$

where $\mathcal{Q}^{\texttt{num}}(\boldsymbol{\lambda}^{(k+1)},\boldsymbol{\lambda}^{(k)})$ is the standard ML auxiliary function, defined by equation 2.16. This function is defined using the model in the numerator of equation 2.28, associated with the correct class label. Similarly, $\mathcal{Q}^{\texttt{den}}(\boldsymbol{\lambda}^{(k+1)},\boldsymbol{\lambda}^{(k)})$ is the equivalent auxiliary function associated with the denominator model in equation 2.28, representing every possible class label. Finally, $\mathcal{Q}^{\texttt{sm}}(\boldsymbol{\lambda}^{(k+1)},\boldsymbol{\lambda}^{(k)})$ is a smoothing term, required to ensure convergence. For GMMs, a suitable smoothing term is [217]

$$\mathcal{Q}^{\texttt{sm}}(\boldsymbol{\lambda}^{(k+1)},\boldsymbol{\lambda}^{(k)}) = \sum_{m=1}^{M} D_m \int_{\boldsymbol{o}} p(\boldsymbol{o}|\theta=m;\boldsymbol{\lambda}^{(k)}) \log p(\boldsymbol{o}|\theta=m;\boldsymbol{\lambda}^{(k+1)}) \; d\boldsymbol{o} \tag{2.30}$$

where $p(\boldsymbol{o}|\theta=m;\boldsymbol{\lambda})$ is the likelihood of $\boldsymbol{o}$ being generated by component m and D_m is a component specific smoothing term that ensures that the weak-sense auxiliary function is convex. This is typically set to the maximum of a) the scaled component-occupancy of the denominator model and b) the minimum value $\tilde{D}_m$ that ensures that the covariance matrices obtained are positive semi-definite [209].

$$D_m = \max(E\sum_{t=1}^{T} \gamma_m^{\texttt{den}}(t), \tilde{D}_m) \tag{2.31}$$

where E is a constant, normally fixed around 1-2 for ASR tasks. For GMMs, optimising using the auxiliary function defined by equation 2.29 with a single training utterance yields the same update rules as the EBW algorithm [164].

$$\boldsymbol{\mu}_m^{(k+1)} = \frac{D_m\boldsymbol{\mu}_m^{(k)} + \sum_{t=1}^{T} \gamma_m^{\texttt{num}}(t)\boldsymbol{o}_t - \gamma_m^{\texttt{den}}(t)\boldsymbol{o}_t}{D_m + \sum_{t=1}^{T} \gamma_m^{\texttt{num}}(t) - \gamma_m^{\texttt{den}}(t)} \tag{2.32}$$

$$\boldsymbol{\Sigma}_m^{(k+1)} = \frac{D_m(\boldsymbol{\mu}_m^{(k)}\boldsymbol{\mu}_m^{(k)\texttt{T}} + \boldsymbol{\Sigma}_m^{(k)}) + \sum_{t=1}^{T} \gamma_m^{\texttt{num}}(t)\boldsymbol{o}_t\boldsymbol{o}_t^{\texttt{T}} - \gamma_m^{\texttt{den}}(t)\boldsymbol{o}_t\boldsymbol{o}_t^{\texttt{T}}}{D_m + \sum_{t=1}^{T} \gamma_m^{\texttt{num}}(t) - \gamma_m^{\texttt{den}}(t)} - \boldsymbol{\mu}_m^{(k+1)}\boldsymbol{\mu}_m^{(k+1)\texttt{T}} \tag{2.33}$$

Here, $\gamma_m^{\texttt{num}}(t)$ and $\gamma_m^{\texttt{den}}(t)$ are respectively the posterior probabilities associated with the numerator and denominator of equation 2.28.

$$\gamma_m^{\texttt{num}}(t) = P(\theta_t = m|\boldsymbol{O}_i; \boldsymbol{\lambda}^{y_i}) \quad (2.34)$$

$$\gamma_m^{\texttt{den}}(t) = \sum_{\omega \in \Omega} P(\theta_t = m|\boldsymbol{O}; \boldsymbol{\lambda}^{(\omega)}) \quad (2.35)$$

A similar approach may be used to update the parameters of a hidden Markov model [210].

2.1.4 Model adaptation

Training generative model parameters using the ML or MMI criteria will only yield robust parameter estimates when sufficient training data is available. For many speech classification tasks, the amount of training data available per class is limited and this will not be the case. One method for handling this issue is to apply Bayesian approaches [173, 207]. Instead of training 'point estimates' for the model parameter values, here the model parameters are assumed to be random variables associated with a distribution. The likelihood of the training data is then obtained by marginalising over this distribution and model training consists of selecting a distribution for the parameters that maximises some training criterion, e.g. ML.

An alternative method of obtaining robust parameter estimates is though model adaptation. Here a robust class-independent model is used as a starting point. This is then transformed into a class-dependent model using a small amount of class-dependent adaptation data. For speech classification tasks, the most common adaptation schemes are maximum a-posteriori (MAP) adaptation and maximum likelihood linear regression (MLLR). These are introduced in the following subsections. Cluster-adaptive training (CAT), an adaptive training scheme where parameters are tied over multiple class-dependent models, is also discussed. This scheme allows a set of robust class-dependent models to be trained in parallel even when only minimal training data is available per class.

2.1.4.1 MAP adaptation

Maximum a-posteriori (MAP) adaptation [71] is an adaptation scheme for generative models that seeks to maximises the posterior probability of the model parameters $\boldsymbol{\lambda}$ given the observations. This leads to the following training criterion.

$$\mathcal{F}^{\texttt{map}}(\boldsymbol{\lambda}|\mathcal{O}) = \log p(\boldsymbol{\lambda}|\boldsymbol{O}) \quad (2.36)$$

Using Bayes' rule the optimal parameters $\boldsymbol{\lambda}^*$ can be expressed in terms of a generative model $p(\boldsymbol{O}; \boldsymbol{\lambda})$ and a prior distribution $p(\boldsymbol{\lambda})$ over the model parameters.

$$\boldsymbol{\lambda}^* = \underset{\boldsymbol{\lambda}}{\text{argmax}} \ \log p(\boldsymbol{O}; \boldsymbol{\lambda}) + \log p(\boldsymbol{\lambda}) \quad (2.37)$$

The advantage of the MAP approach is that given a robust prior distribution over the model parameters, the training process will yield robust estimates for the updated parameters, even when the amount of training data is limited. By using a robust class-independent prior, MAP can therefore be used to robustly estimate parameters of a class-dependent model given a small amount of adaptation data. The choice of a suitable prior distribution is problematic since there is no conjugate prior for a GMM. However, by assuming that the component priors

and Gaussians are independent, a suitable form of prior may be obtained [71]. Typically, prior distributions are used that have the form

$$p(\boldsymbol{\lambda}) = p^{\mathtt{D}}(\boldsymbol{c}^{\mathtt{prior}}) \prod_{m=1}^{M} p^{\mathtt{W}}(\boldsymbol{\mu}_m^{\mathtt{prior}}, \boldsymbol{\Sigma}_m^{\mathtt{prior}}) \tag{2.38}$$

where $p^{\mathtt{D}}(\cdot)$ is a Dirichlet distribution over the component priors and $p^{\mathtt{W}}(\cdot)$ is a normal-Wishart distribution. The parameters of the prior distribution may be obtained from a well-trained class-independent GMM $p(\boldsymbol{O}; \boldsymbol{\lambda}^{\mathtt{prior}})$ where $\boldsymbol{\lambda}^{\mathtt{prior}} = \{\boldsymbol{c}^{\mathtt{prior}}, \boldsymbol{\mu}^{\mathtt{prior}}, \boldsymbol{\Sigma}^{\mathtt{prior}}\}$. For GMMs, EM training using the MAP criterion with this choice of prior yields the following parameter updates. Similar expressions also exist to adapt HMM parameters [71].

$$\boldsymbol{\mu}_m^{\mathtt{map}} = \frac{\sum_{t=1}^{T} P(\theta_t = m|\boldsymbol{o}_t; \boldsymbol{\lambda}) \boldsymbol{o}_t + \tau^{\mathtt{map}} \boldsymbol{\mu}_m^{\mathtt{prior}}}{\sum_{t=1}^{T} P(\theta_t = m|\boldsymbol{o}_t; \boldsymbol{\lambda}) + \tau^{\mathtt{map}}} \tag{2.39}$$

$$\begin{aligned} \boldsymbol{\Sigma}_m^{\mathtt{map}} = & \frac{\sum_{t=1}^{T} P(\theta_t = m|\boldsymbol{o}_t; \boldsymbol{\lambda})(\boldsymbol{o}_t - \boldsymbol{\mu}_m^{\mathtt{map}})(\boldsymbol{o}_t - \boldsymbol{\mu}_m^{\mathtt{map}})^{\mathtt{T}}}{\sum_{t=1}^{T} P(\theta_t = m|\boldsymbol{o}_t; \boldsymbol{\lambda}) + \tau^{\mathtt{map}}} \\ & + \frac{\tau^{\mathtt{map}}(\boldsymbol{\mu}_m^{\mathtt{prior}} - \boldsymbol{\mu}_m^{\mathtt{map}})(\boldsymbol{\mu}_m^{\mathtt{prior}} - \boldsymbol{\mu}_m^{\mathtt{map}})^{\mathtt{T}} + \tau^{\mathtt{map}} \boldsymbol{\Sigma}_m^{\mathtt{prior}}}{\sum_{t=1}^{T} P(\theta_t = m|\boldsymbol{o}_t; \boldsymbol{\lambda}) + \tau^{\mathtt{map}}} \end{aligned} \tag{2.40}$$

$$c_m^{\mathtt{map}} = \frac{\sum_{t=1}^{T} P(\theta_t = m|\boldsymbol{o}_t; \boldsymbol{\lambda}) + \tau^{\mathtt{map}}}{\sum_{n=1}^{M} \sum_{t=1}^{T} P(\theta_t = n|\boldsymbol{o}_t; \boldsymbol{\lambda}) + \tau^{\mathtt{map}}} \tag{2.41}$$

Here $\tau^{\mathtt{map}}$ is a parameter that controls the influence of the prior on the MAP estimates. When $\tau^{\mathtt{map}}$ equals zero, MAP training will yield the standard ML parameter estimates. As τ approaches infinity the MAP estimate of the model parameters tends towards the prior. By comparing the update equations, MAP adaptation can be seen to be closely related to the parameter smoothing used in MMI training, discussed in section 2.1.3.2.

In situations when only limited adaptation data is available many components will not be moved far from the prior. It is therefore common to carry out multiple iterations of MAP to successively increase the likelihood of the adaptation data. In static-prior MAP [71], the original well-trained model is used as the prior distribution after every iteration, while the statistics $P(\theta_t = m|\boldsymbol{o}_t; \boldsymbol{\lambda})$ are generate using the current parameter estimates. This ensures that new parameter estimates remain robust as the influence of the well-trained prior distribution is not reduced after successive iterations. Discriminative adaptation schemes based on MAP have also been developed. MMI-MAP [164] is an example of this approach.

2.1.4.2 Maximum likelihood linear regression

Maximum Likelihood Linear Regression (MLLR) [63, 121, 122] is a popular technique for adapting generative model parameters. MLLR operates by linearly transforming the mean [121] and/or variance [68] of a model to better represent a particular class. Transforms are typically tied over multiple model components using a regression class tree [62, 122].

An example of a regression class tree is shown in figure 2.4. Here each leaf node represents a set of model parameters and is known as a base-class. For each node in the tree, a transform is generated using the adaptation data associated with the descendant base-classes. If there is not enough data to robustly generate a transform for a node, the transform associated with

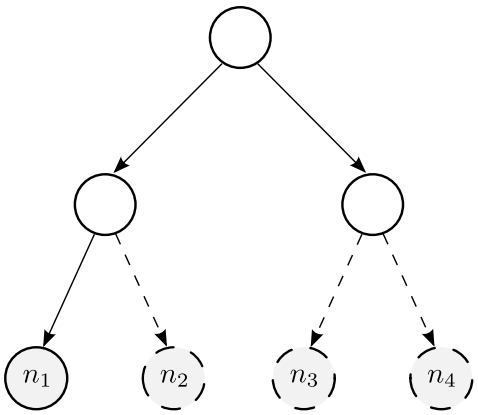


Figure 2.4: An example regression tree for a MLLR transform.

the parent node will be used instead. In the worst case a single global transform, estimated over the entire training set, will be used to adapt all parameters. The sets of base-classes that share a transform are known as a regression classes. The regression class tree in figure 2.4, contains four base-classes. Of these, nodes n_2, n_3 and n_4 represent base-classes for which there is insufficient adaptation data to generate a transform. The transform for class n_2 will be generated from data occupying base-classes n_1 and n_2 and the parameters associated with base-classes n_3 and n_4 form a regression class and will share a transform. The structure of a regression class tree may be based either on expert knowledge or, more commonly, a suitable tree may be trained automatically by assuming that 'close' Gaussians are adapted using the same linear transform [121].

For GMMs, the adapted mean $\boldsymbol{\mu}_m^{\mathtt{MLLR}}$ and covariance $\boldsymbol{\Sigma}_m^{\mathtt{MLLR}}$ associated with component m is given by

$$\boldsymbol{\mu}_m^{\mathtt{MLLR}} = \boldsymbol{A}_r\boldsymbol{\mu}_m + \boldsymbol{b}_r \tag{2.42}$$

$$\boldsymbol{\Sigma}_m^{\mathtt{MLLR}} = \boldsymbol{H}_r\boldsymbol{\Sigma}_m\boldsymbol{H}_r^{\mathtt{T}} \tag{2.43}$$

where $\boldsymbol{A}_r$ and $\boldsymbol{H}_r$ are DxD transforms associated with regression class r and $\boldsymbol{b}_r$ is a Dx1 bias vector. CMLLR [52, 63] is a variant of MLLR where $\boldsymbol{A}_r$ and $\boldsymbol{H}_r$ are constrained to be identical. In that case, an equivalent transform may alternatively be applied directly to the observations. The transform parameters can be estimated using maximum likelihood estimation and expectation maximisation. For the case when diagonal covariances are used, this yields a closed form solution. For a regression class r, the adaptation parameters $\{\boldsymbol{a}_{ir}, b_{ir}\}$ associated with row i of the mean transform $\boldsymbol{A}_r$ and bias $\boldsymbol{b}_r$ are given by

$$[\boldsymbol{a}_{ir} \quad b_{ir}]^{\mathtt{T}} = \boldsymbol{G}_{ir}^{-1}\boldsymbol{k}_{ir} \tag{2.44}$$

where $\boldsymbol{G}_{ir}$ and $\boldsymbol{k}_{ir}$ are sufficient statistics defined by

$$\boldsymbol{G}_{ir} = \sum_{m\in r}\frac{1}{\sigma_{im}^2}\sum_{t=1}^{T}P(\theta_t = m|\boldsymbol{O};\boldsymbol{\lambda})\boldsymbol{\xi}_m\boldsymbol{\xi}_m^{\mathtt{T}} \tag{2.45}$$

$$\boldsymbol{k}_{ir} = \sum_{m\in r}\frac{1}{\sigma_{im}^2}\sum_{t=1}^{T}P(\theta_t = m|\boldsymbol{O};\boldsymbol{\lambda})o_{it}\boldsymbol{\xi}_m \tag{2.46}$$

where σ_{im}^2 is the ith diagonal element of $\boldsymbol{\Sigma}_m$ and $\boldsymbol{\xi}_m$ is the extended mean vector associated with component m.

$$\boldsymbol{\xi}_m = \begin{bmatrix} \boldsymbol{\mu}_m \\ 1 \end{bmatrix} \tag{2.47}$$

In general, when only a small amount of adaptation data is available, MLLR tends to yield more robust parameter estimates than MAP. However, for larger amounts of data, better performance is usually obtained using MAP.

2.1.4.3 Cluster-adaptive training

Adaptive training is a technique that combines the parameter estimation and adaptation approaches described in the previous sections. For a set of classes, $\Omega = \{\omega_1, \ldots, \omega_K\}$, adaptive training can be used to robustly obtain a set of class-dependent generative models, by simultaneously training a single class-independent model, known as the canonical model, and a set of class-dependent adaptation parameters. An early form of adaptive training was speaker adaptive training (SAT) [1]. This uses MLLR transforms to define the parameters of the class-dependent GMM or HMM. Cluster-adaptive training (CAT) [65], described here, is a related approach originally developed for ASR applications where is it used both to compensate for the variation between training corpora and as a robust speaker-adaptation scheme. For GMM models, the mean parameters of a class-dependent model $p(\boldsymbol{o}; \boldsymbol{\lambda}^{(\omega)})$ are defined by a class-dependent weighted interpolation of a set of P class-independent means, known as the clusters. For a particular Gaussian component m, $\boldsymbol{\mu}_m^{(\omega)}$ is defined by

$$\boldsymbol{\mu}_m^{(\omega)} = \boldsymbol{M}_m \boldsymbol{w}_r{}^{(\omega)} \tag{2.48}$$

where $\boldsymbol{M}_m$ is a matrix of P class-independent mean vectors associated with component m

$$\boldsymbol{M}_m = [\boldsymbol{\mu}_{m1} \ldots \boldsymbol{\mu}_{mP}] \tag{2.49}$$

and $\boldsymbol{w}_r^{(\omega)}$ is the vector of cluster weights associated with class ω.

$$\boldsymbol{w}_r^{(\omega)} = \begin{bmatrix} w_{r1}^{(\omega)} \\ \vdots \\ w_{rP}^{(\omega)} \end{bmatrix} \tag{2.50}$$

The cluster weights associated with each class therefore define coordinates within a P-dimensional space allowing a class-dependent model to be represented compactly. This approach is closely related to eigenvoices [114]. However, while eigenvoice cluster means are typically obtained using PCA, CAT parameters are trained using maximum-likelihood estimation.

As in MLLR, components may be partitioned into R regression-classes. This allows an independent set of cluster weights to be applied to each class. Here the subscript r is used to indicate the regression class associated with the current component. A bias cluster can also be used to represent any class-independent aspect of the mean-vectors. In this case, the final cluster weight $w_P^{(\omega)}$ is fixed at 1 for all classes. Component priors

and variances are usually tied over all classes. Thus the parameters $\boldsymbol{\lambda}$ of the CAT model are given by $\boldsymbol{\lambda} = \{\boldsymbol{M}, \boldsymbol{w}^{(1)}, \ldots, \boldsymbol{w}^{(K)}, \boldsymbol{c}, \boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_M\}$ where $\boldsymbol{w}^{(\omega)} = \{\boldsymbol{w}_1^{(\omega)}, \ldots, \boldsymbol{w}_R^{(\omega)}\}$ and $\boldsymbol{M} = \{\boldsymbol{M}_1, \ldots, \boldsymbol{M}_M\}$. Given these parameters, the likelihood of generating an observation $\boldsymbol{o}$ with label $y = \omega$ can be calculated using

$$p(\boldsymbol{o}; \boldsymbol{\lambda}) = \sum_{m=1}^{M} c_m \frac{1}{\sqrt{(2\pi)^D \left|\boldsymbol{\Sigma}_m\right|}} \exp\left(-\frac{1}{2}\left(\boldsymbol{o} - \boldsymbol{M}_m \boldsymbol{w}_r^{(\omega)}\right)^{\mathtt{T}} \boldsymbol{\Sigma}_m^{-1} \left(\boldsymbol{o} - \boldsymbol{M}_m \boldsymbol{w}_r^{(\omega)}\right)\right) \quad (2.51)$$

CAT systems are typically trained using maximum likelihood estimation, although discriminative criteria may also be used [217]. For a set of training data $\mathcal{O} = \{\boldsymbol{O}_1, \ldots, \boldsymbol{O}_N\}$ with associated class labels $\mathcal{Y} = \{y_1, \ldots, y_N\}$ it is not possible to find a closed form solution for the ML parameter estimates. However by alternatively updating either the cluster means $\boldsymbol{M}$ or the the weights $\boldsymbol{w}$, while the other is kept fixed, it is possible to iteratively increase the likelihood of the training data.

When the cluster means are assumed to be fixed, new ML estimates for the weights can be obtained via EM by optimising the following auxiliary function [65].

$$\mathcal{Q}(\boldsymbol{\lambda}^{(k+1)}, \boldsymbol{\lambda}^{(k)}) = \sum_{\omega \in \Omega} \sum_{r=1}^{R} \boldsymbol{w}_r^{(\omega)\mathtt{T}} \boldsymbol{k}^{\mathtt{w}(\omega)} - \frac{1}{2} \boldsymbol{w}_r^{(\omega)\mathtt{T}} \boldsymbol{G}_r^{\mathtt{w}(\omega)} \boldsymbol{w}_r^{(\omega)} \quad (2.52)$$

where $\boldsymbol{G}_r^{\mathtt{w}(\omega)}$ and $\boldsymbol{k}_r^{\mathtt{w}(\omega)}$ are sufficient statistics accumulated over training utterances $\boldsymbol{O}_i$ with label $y_i = \omega$.

$$\boldsymbol{G}_r^{\mathtt{w}(\omega)} = \sum_{m \in r} \sum_{t=1}^{T} P(\theta_t = m | \boldsymbol{o}_t; \boldsymbol{\lambda}^{(k)}) \boldsymbol{M}_m^{\mathtt{T}} \boldsymbol{\Sigma}_m^{-1} \boldsymbol{M}_m \quad (2.53)$$

$$\boldsymbol{k}_r^{\mathtt{w}(\omega)} = \sum_{m \in r} \boldsymbol{M}_m^{\mathtt{T}} \boldsymbol{\Sigma}_m^{-1} \sum_{t=1}^{T} P(\theta_t = m | \boldsymbol{o}_t; \boldsymbol{\lambda}^{(k)}) (\boldsymbol{o}_t - \boldsymbol{\mu}_{mP}) \quad (2.54)$$

Here $P(\theta_t = m | \boldsymbol{o}_t; \boldsymbol{\lambda}^{(k)})$ is the probability of the observation at time t being generated by component m given the previous estimate of the cluster weights and means. $\boldsymbol{\mu}_{mP}$ is the cluster mean vector associated with the bias cluster P. This term is required to ensure that the cluster weights associated with the bias cluster are fixed at 1. A new estimate of the cluster weights can be obtained by differentiating the auxiliary function with respect to $\boldsymbol{w}_r$ and equating to zero.

$$\boldsymbol{w}_r^{(\omega)} = \boldsymbol{G}_r^{\mathtt{w}(\omega)-1} \boldsymbol{k}_r^{\mathtt{w}(\omega)} \quad (2.55)$$

Similarly, by keeping the cluster weights $\boldsymbol{w}$ fixed, a new ML estimate for the cluster means can be obtained via EM by maximising the following auxiliary function [65].

$$\mathcal{Q}(\boldsymbol{\lambda}^{(k+1)}, \boldsymbol{\lambda}^{(k)}) = -\frac{1}{2} \sum_{\omega \in \Omega} \sum_{m=1}^{M} \sum_{t=1}^{T} P(\theta_t = m | \boldsymbol{o}_t; \boldsymbol{\lambda}^{(k)}) \left(\boldsymbol{o}_t - \boldsymbol{M}_m \boldsymbol{w}_r^{(\omega)}\right)^{\mathtt{T}} \boldsymbol{\Sigma}_m^{-1} \left(\boldsymbol{o}_t - \boldsymbol{M}_m \boldsymbol{w}_r^{(\omega)}\right) \quad (2.56)$$

Differentiating equation 2.56 with respect to the cluster means $\boldsymbol{M}_m$ associated with each component m yields the following update equation.

$$\boldsymbol{M}_m^{\mathtt{T}} = \boldsymbol{G}_m^{-1}\boldsymbol{K}_m \tag{2.57}$$

where $\boldsymbol{G}_m$ and $\boldsymbol{K}_m$ are sufficient statistics accumulated over all training utterances.

$$\boldsymbol{G}_m = \sum_{\omega\in\Omega}\sum_{t=1}^{T} P(\theta_t = m|\boldsymbol{o}_t;\boldsymbol{\lambda}^{(k)})\boldsymbol{w}_r^{(\omega)}\boldsymbol{w}_r^{(\omega)\mathtt{T}} \tag{2.58}$$

$$\boldsymbol{K}_m = \sum_{\omega\in\Omega}\sum_{t=1}^{T} P(\theta_t = m|\boldsymbol{o}_t;\boldsymbol{\lambda}^{(k)})\boldsymbol{w}_r^{(\omega)}\boldsymbol{o}(t)^{\mathtt{T}} \tag{2.59}$$

A new estimate of the covariances may be obtained in a similar manner and (assuming diagonal covariances) is given by

$$\boldsymbol{\Sigma}_m = \mathtt{diag}\left(\frac{\boldsymbol{L}_m - \boldsymbol{M}_m\boldsymbol{K}_m}{\sum_{t=1}^{T} P(\theta_t = m|\boldsymbol{o}_t,\boldsymbol{\lambda})}\right) \tag{2.60}$$

where $\boldsymbol{G}_m$ and $\boldsymbol{K}_m$ are defined by equations 2.58 and 2.59 and $\boldsymbol{L}_m$ is defined as follows.

$$\boldsymbol{L}_m = \sum_{t=1}^{T} P(\theta_t = m|\boldsymbol{o}_t;\boldsymbol{\lambda}^{(k)})\boldsymbol{o}_t\boldsymbol{o}_t^{\mathtt{T}} \tag{2.61}$$

In order to train a CAT model it is necessary to obtain initial estimates for either $\boldsymbol{M}_m$ for each component m or $\boldsymbol{w}_r^{(\omega)}$ for each class ω and regression class r. For CAT models that contain two non-bias clusters, gender-based initialisation schemes may be used. Alternatively, when a larger number of clusters is required, an eigenvoice initialisation scheme [65] may be applied.

2.2 Discriminative classification schemes

There has been considerable interest in applying discriminative techniques to improve speech classification performance. One approach, discussed in the previous section, is to combine generative models with discriminative training criteria such as maximum-mutual information. An alternative is to make use of discriminative classification schemes. Whereas generative classification schemes attempt to model the likelihood of speech observations given a particular class, discriminative schemes attempt to model the class boundaries directly.

Many of the approaches discussed in this section are examples of discriminative models. These attempt to directly model the posterior probability of the class (or class-sequence) given an observation sequence. Conditional random fields (CRFs), discussed first, and a latent variable extension, HCRFs, are an example of this approach. Discriminative schemes may also be based on non-statistical discriminative functions. Here a (typically linear) decision boundary is trained that separates the classes. The support vector machine, described here, is an example of this approach and is the primary form of discriminative classifier used in this thesis. A closely related discriminative model, the relevance vector machine, is also introduced and schemes are presented for applying these techniques to multi-class tasks.

Finally, kernel functions are introduced. These allow linear schemes such as the support vector machine or relevance vector machine to be applied to non-linear data. The static kernel functions described in this chapter only operate on fixed-dimensional data, and hence can not be used for speech classification. However, they provide an introduction to the dynamic kernels described in the following chapter that can be used to classify speech.

2.2.1 Conditional random fields

Conditional random fields (CRFs) [116] are a form of conditional model, related to Maximum Entropy Markov Models (MEMMs) [20, 148]. Unlike generative models such as GMMs or HMMs, conditional models directly approximate the conditional likelihood of the class labels given an observation sequence. The majority of the models described in this chapter operate on data where each observation sequence, $\boldsymbol{O} = \{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_T\}$, is assigned a single class label, $y \in \Omega$. Unlike these models, CRFs operate on data where each observation is associated with a distinct class label. Thus, CRFs model the conditional likelihood of a label sequence $\boldsymbol{Y} = \{y_1, \ldots, y_T\}$. CRFs have the form of a normalised log-linear model and the conditional probability of a label sequence, $\boldsymbol{Y}$, given an observation sequence, $\boldsymbol{O}$, is defined by

$$P(\boldsymbol{Y}|\boldsymbol{O};\boldsymbol{\alpha}) = \frac{1}{Z(\boldsymbol{O};\boldsymbol{\alpha})} \exp\left(\sum_{t=1}^{T}\sum_{j} \alpha_j f_j(y_t, y_{t-1}, \boldsymbol{O})\right) \tag{2.62}$$

where $Z(\boldsymbol{O};\boldsymbol{\alpha})$ is a normalisation term, required to ensure that $P(\boldsymbol{Y}|\boldsymbol{O};\boldsymbol{\alpha})$ is a valid probability mass function. $Z(\boldsymbol{O};\boldsymbol{\alpha})$ is calculated over all possible label sequences $\bar{\boldsymbol{Y}}$.

$$Z(\boldsymbol{O};\boldsymbol{\alpha}) = \sum_{\boldsymbol{Y}\in\bar{\boldsymbol{Y}}} \exp\left(\sum_{t=1}^{T}\sum_{j} \alpha_j f_j(y_t, y_{t-1}, \boldsymbol{O})\right) \tag{2.63}$$

The conditional model is defined using a fixed set of functions, $f_j(y_t, y_{t-1}, \boldsymbol{O})$, each weighted by a model parameter α_j. Functions are dependent on both the observation and label sequence. Although the CRF framework does not limit the dependencies associated with the functions, additional constraints are typically introduced to allow $Z(\boldsymbol{O};\boldsymbol{\alpha})$ to be efficiently calculated. In equations 2.62 and 2.63 a second order Markov dependency over the label sequences is applied. Under these conditions, equation 2.63 may be efficiently calculated using a forward-backward style algorithm [13, 116]. The dynamic Bayesian network associated with the observation and label sequences is shown in figure 2.5.

Determining appropriate functions for particular applications is an important area of research. These may be based on acoustic features only or more complex functions may be used that incorporate label information. One approach is to use expert knowledge to derive an appropriate set of features [116, 157, 180]. Alternatively, functions may be based on sufficient statistics obtained from a generative model [74][1]. Many dynamic kernels for continuous data, introduced in chapter 3, also have explicit expansion functions that can be used to obtain CRF features. Given an initial set of functions, post-processing algorithms may be applied to refine the set by iteratively combining and pruning functions [147, 157].

[1]Note that the experiments in [74] were performed using HCRFs. However, the same sufficient statistics may be used for CRFs

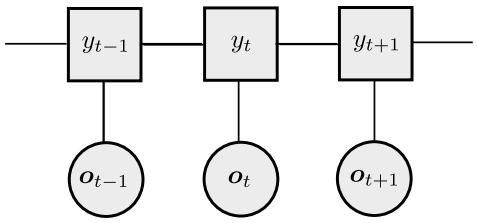


Figure 2.5: A dynamic Bayesian network corresponding to a conditional random field containing a second order Markov dependency over labels. The nature of the dependencies is dependent on the form of $f_j(y_t, y_{t-1}, \boldsymbol{O})$. Additional dependencies between labels and observations are not shown, but may be included.

One criterion suitable for training the parameters of a CRF is the conditional maximum likelihood (CML) criterion [116, 152], introduced in section 2.1.3.2. For a training set of N utterances, this seeks to select the model parameters $\boldsymbol{\alpha}$ that maximise the conditional likelihood of the label sequences $\mathcal{Y} = \{\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_N\}$ given the utterances $\mathcal{O} = \{\boldsymbol{O}_1, \ldots, \boldsymbol{O}_N\}$.

$$\mathcal{F}^{\texttt{CML}} = \sum_{i=1}^{N} \log P(\boldsymbol{Y}_i|\boldsymbol{O}_i; \boldsymbol{\alpha}) \tag{2.64}$$

When the prior distribution of the class labels is kept fixed, the CML criterion is identical in form to the MMI criterion in equation 2.26. The difference is due to the form of model to which it is applied. Whereas the MMI criterion is used to train generative models, the CML criterion is applied to conditional models. It is not possible to obtain a solution in closed form that maximises equation 2.64. Instead, iterative approaches may be used. These may be based on expectation maximisation, such as the generalised iterative scaling [42, 116] or improved iterative scaling [157] schemes. Alternatively, gradient descent-based algorithms may be used [180, 202]. These often have simpler implementations and in many cases have been found to converge faster.

2.2.2 Hidden conditional random fields

The hidden conditional random field (HCRF) [165, 166] is a latent variable extension of the conditional random field described in section 2.2.1. In the CRF model it is assumed that there exists a one-to-one relationship between an observation $\boldsymbol{o}_t$ and label y_t. For many applications, including ASR and SV, this assumption is not appropriate.

For HCRFs, this one-to-one relationship between observations and labels is relaxed by incorporating a latent state sequence, $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_T)$, that allows complex dependencies in the observation structure to be included in the model. The posterior probability of a class label, $y \in \Omega$, given an observation sequence $\boldsymbol{O} = \{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_T\}$ is then obtained by marginalising over all latent state sequences Θ.

$$P(y|\boldsymbol{O}; \boldsymbol{\alpha}) = \frac{1}{Z(\boldsymbol{O}; \boldsymbol{\alpha})} \sum_{\boldsymbol{\theta} \in \Theta} \exp\left(\sum_j \alpha_j f_j(y, \boldsymbol{O}, \boldsymbol{\theta}) \right) \tag{2.65}$$

where $\boldsymbol{\alpha}$ are the model parameters and $f(y, \boldsymbol{O}, \boldsymbol{\theta})$ defines a set of sufficient statistics that are dependent on the observations, the latent state sequences and the class label. $Z(\boldsymbol{O}; \boldsymbol{\alpha})$ is a

normalisation term, equal to the expectation of the unnormalised model over all classes Ω and latent sequences Θ.

$$Z(\boldsymbol{O};\boldsymbol{\alpha}) = \sum_{y\in\Omega}\sum_{\boldsymbol{\theta}\in\Theta} \exp\left(\sum_j \alpha_j f_j(y,\boldsymbol{O},\boldsymbol{\theta})\right) \tag{2.66}$$

As with CRFs, additional constraints may be included in the model to allow $Z(\boldsymbol{O};\boldsymbol{\alpha})$ to be calculated. When a Markov dependency is imposed on the latent state sequence, equation 2.66 can be efficiently estimated using a forward-backward algorithm [74, 165]. In equation 2.65 this dependency is implicitly incorporated into $f(y,\boldsymbol{O},\boldsymbol{\theta})$.

The nature of the functions $f(y,\boldsymbol{O},\boldsymbol{\theta})$ determines the form of dependencies included in the model. One option for defining the latent variable and sufficient statistics associated with the HCRF is to emulate the structure and sufficient statistics associated with latent variable generative models such as HMMs [74]. For example, when the latent sequences follows a first order Markov assumption the following HMM-style sufficient statistics may be used.

$$f^{\texttt{TR}}_{\theta\theta'}(y,\boldsymbol{O},\boldsymbol{\theta}) = \sum_{t=1}^{T}\delta(\theta_{t-1}=\theta)\delta(\theta_t=\theta') \qquad \forall\theta\forall\theta' \tag{2.67}$$

$$f^{\texttt{OCC}}_{\theta}(y,\boldsymbol{O},\boldsymbol{\theta}) = \sum_{t=1}^{T}\delta(\theta_t=\theta) \qquad \forall\theta \tag{2.68}$$

$$f^{\texttt{M1}}_{\theta d}(y,\boldsymbol{O},\boldsymbol{\theta}) = \sum_{t=1}^{T}\delta(\theta_t=\theta)o_{dt} \qquad \forall\theta\forall d \tag{2.69}$$

$$f^{\texttt{M2}}_{\theta d}(y,\boldsymbol{O},\boldsymbol{\theta}) = \sum_{t=1}^{T}\delta(\theta_t=\theta)o^2_{dt} \qquad \forall\theta\forall d \tag{2.70}$$

where $\delta(\theta_t = \theta)$ is the Kronecker function, equal to one if the condition is true or zero otherwise. These (Viterbi) statistics correspond respectively to the transition probabilities, state occupancies and first and second order Gaussian sufficient statistics associated with a single-component per state, diagonal covariance HMM. Similar statistics may easily be defined for the general HMM case.

By appropriately selecting a set of sufficient statistics and model parameters $\boldsymbol{\alpha}$, it is possible to obtain an HCRF that yields the conditional probability density function associated with a hidden Markov model [74]. Given a diagonal covariance, single-component HMM defined by the parameters $\{\boldsymbol{\mu},\boldsymbol{\Sigma},\boldsymbol{a}\}$, the HCRF will yield an identical conditional likelihood when the sufficient statistics defined by equations 2.67-2.70 are used and corresponding elements of $\boldsymbol{\alpha}$ are set to

$$\alpha^{\texttt{TR}}_{\theta\theta'} = \log a_{\theta\theta'} \qquad \forall\theta\forall\theta' \tag{2.71}$$

$$\alpha^{\texttt{OCC}}_{\theta} = -\frac{1}{2}\left(\log 2\pi^D|\boldsymbol{\Sigma}_\theta| + \boldsymbol{\mu}^{\mathsf{T}}_\theta\boldsymbol{\Sigma}^{-1}_\theta\boldsymbol{\mu}_\theta\right) \qquad \forall\theta \tag{2.72}$$

$$\alpha^{\texttt{M1}}_{\theta d} = \frac{\mu_{d\theta}}{\sigma^2_{d\theta}} \qquad \forall\theta\forall d \tag{2.73}$$

$$\alpha^{\texttt{M2}}_{\theta d} = \frac{1}{2\sigma^2_{d\theta}} \qquad \forall\theta\forall d \tag{2.74}$$

where $\sigma^2_{d\theta}$ is the dth diagonal element of $\boldsymbol{\Sigma}_\theta$. Similarly, by selecting appropriate functions and model parameters, the HCRF model will yield the conditional likelihood associated a general multiple component, full covariance HMM. Since negative values of $\alpha^{\texttt{M2}}_{\theta d}$ do not correspond to valid covariance matrices, equations 2.71-2.74 appear to indicate that the HCRF is able to model more general dependencies than the HMM. In fact, this is not the case. The decision boundary associated with the HCRF can be shown to be invariant to certain transforms of $\boldsymbol{\alpha}$ [84]. By performing an appropriate set of invariance transforms, it is possible to transform any Gaussian-like HCRF into a corresponding HMM [84, 85]. Hence the two forms of model are equivalent. Given a suitable training set $\mathcal{O} = \{\boldsymbol{O}_1, \ldots, \boldsymbol{O}_N\}$ with associated class labels $\mathcal{Y} = \{y_1, \ldots, y_N\}$, the model parameters $\boldsymbol{\alpha}$ are usually selected to maximise the conditional likelihood of the class labels given the data.

$$\mathcal{F}^{\texttt{CML}}(\boldsymbol{\alpha}|\mathcal{O}) \quad = \quad \sum_{i=1}^{N} \log P(y_i|\boldsymbol{O}_i;\boldsymbol{\alpha}) \tag{2.75}$$

Since HCRFs do not have normalised output distributions or transition probabilities, it is not necessary to use specialized algorithms such as extended Baum-Welch. Instead the CML criterion is typically optimised directly. The CML criterion yields a convex objective function for conditional random fields. However, due to the additional latent structure, this is not the case for HCRFs. To avoid local maxima when iteratively updating $\boldsymbol{\alpha}$, stochastic gradient based methods are usually applied [74, 138, 165, 206].

2.2.3 Support vector machines

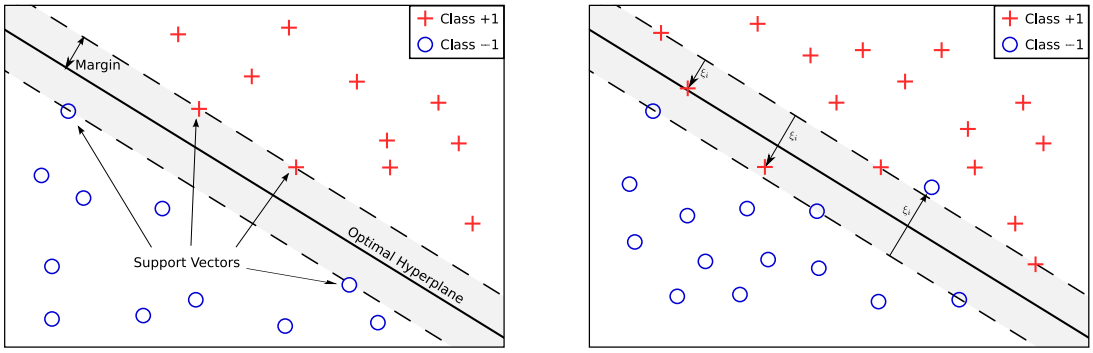
The support vector machine (SVM) [19, 35, 198] is a binary discriminative classifier that has been found to yield good performance on a wide range of machine learning tasks, including gene classification [21], handwriting recognition [10] and image classification [33]. Unlike (H)CRFs, SVMs are distance based classifiers that operate by finding a linear decision boundary according to a maximum-margin criterion.

Consider a training set, $\mathcal{O} = \{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_N\}$, where each $\boldsymbol{o}_i$ is a vector of D elements and has an associated binary label $y_i = \omega$ where $\omega \in \{-1, +1\}$. For each example $\boldsymbol{o}_i$, the elements of $\boldsymbol{o}_i$ define the position of the example within a D-dimensional space. When the training set is linearly separable, it is possible to locate a separating hyperplane within this space such that all training examples are correctly classified. Given this hyperplane, defined by weight vector $\boldsymbol{w}$ and bias b, a test example $\boldsymbol{o}^{\texttt{v}}$ may be classified according to

$$y^{\texttt{v}} = \text{sign}\left(\langle \boldsymbol{w}, \boldsymbol{o}^{\texttt{v}} \rangle + b\right) \tag{2.76}$$

where $\langle \boldsymbol{o}_i, \boldsymbol{o}_j \rangle$ indicates the inner product between $\boldsymbol{o}_i$ and $\boldsymbol{o}_j$. For the linearly separable case, also known as the hard margin case, the task of finding a separating hyperplane is ill-posed. There will usually be infinitely many hyperplanes that correctly classify all training examples. It is therefore necessary to define additional requirements in order to obtain a unique solution. For SVMs, the optimal separating hyperplane is one which maximises the perpendicular distance between the decision boundary and the closest training examples, known as the margin. This *maximum-margin* decision boundary has been shown to minimise a bound on the generalisation error [198].

Equation 2.76 is invariant under a positive rescaling of the hyperplane parameters. Thus, in order to obtain a unique solution it is necessary to introduce additional constraints. For



(a) hard margin (b) soft margin

Figure 2.6: Optimal SVM hyperplane for (a) linearly separable data using a hard margin and (b) non-linearly separable data using a soft margin with slack variables.

SVMs this is achieved by defining canonical hyperplanes on either side of the decision hyperplane. For a fixed value of $\boldsymbol{w}$ and b these are defined by the values of $\boldsymbol{o}$ that form solutions to $\langle \boldsymbol{w}, \boldsymbol{o} \rangle + b = 1$ and $\langle \boldsymbol{w}, \boldsymbol{o} \rangle + b = -1$. Training examples are then constrained to lie outside this region. This arrangement is depicted in figure 2.6(a) for two-dimensional data. Under these conditions the size of the margin can be calculated using the expression [41]

$$\text{Margin} = \frac{1}{\langle \boldsymbol{w}, \boldsymbol{w} \rangle} \tag{2.77}$$

The maximum-margin decision boundary is therefore defined by the parameters $\boldsymbol{w}, b$ that maximise equation 2.77 such that all training examples lie outside the margin. This yields the following quadratic optimisation problem, known as the (hard margin) primal SVM problem.

$$\begin{aligned} \min \quad & \frac{1}{2}\langle \boldsymbol{w}, \boldsymbol{w} \rangle \\ \text{w.r.t} \quad & \boldsymbol{w}, b \\ \text{s.t.} \quad & y_i(\langle \boldsymbol{w}, \boldsymbol{o}_i \rangle + b) \geq 1 \quad \forall i \end{aligned} \tag{2.78}$$

In many situation, particularly when dealing with noisy data, it is not possible to linearly separate the training set. To allow SVMs to be trained in such conditions, the margin constraints, $y_i(\langle \boldsymbol{w}, \boldsymbol{o}_i \rangle + b) \geq 1 \quad \forall i$, are often relaxed to allow some training examples to be misclassified. A slack variable ξ_i is introduced for each training example $\boldsymbol{o}_i$ to provide a measure of the training error associated with the example. For each training example $\boldsymbol{o}_i$, the slack variable ξ_i is non-negative, and is equal to the distance by which $\boldsymbol{o}_i$ violates the original margin constraints. For correctly-classified training examples, $\xi_i = 0$. This is known as the soft margin case and is depicted in figure 2.6(b). To avoid increasing the margin at the expense of misclassifying the training examples, the objective function is then altered to additionally penalize training errors. The (soft margin) primal SVM problem is defined by

$$\begin{aligned} \min \quad & \frac{1}{2}\langle \boldsymbol{w}, \boldsymbol{w} \rangle + C \sum_{i=1}^{N} \xi_i \\ \text{w.r.t.} \quad & \boldsymbol{w}, b \\ \text{s.t.} \quad & y_i(\langle \boldsymbol{w}, \boldsymbol{o}_i \rangle + b) \geq 1 - \xi_i \qquad \forall i \\ & \xi_i \geq 0 \qquad \forall i \end{aligned} \tag{2.79}$$

The constant C acts as a regularisation term controlling the trade-off between reducing the training set error and maximising the margin. For small values of C the margin will be maximised at the expense of making more training set errors. When C is large the number of training set errors will be reduced at the expense of a smaller margin. For linearly separable data, the optimal $\boldsymbol{w}, b$ tend to the hard margin solution as $C \to \infty$. A suitable value for C may be selected using a development set. Alternatively, a data-dependent algorithm may be used to select C. In SVM^{light} [94], by default C is selected according to equation 2.80. In this case, the optimal decision boundary is also invariant to global scalings of the dataset.

$$C = \frac{N}{\sum_{i=1}^{N} \langle \boldsymbol{o}_i, \boldsymbol{o}_i \rangle} \tag{2.80}$$

By introducing Lagrange multipliers associated with each constraint in equation 2.79, a dual form of the SVM optimisation function can be derived.

$$\begin{aligned} \max \quad & \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \langle \boldsymbol{o}_i, \boldsymbol{o}_j \rangle \\ \text{w.r.t.} \quad & \boldsymbol{\alpha} \\ \text{s.t.} \quad & \sum_{i=1}^{N} y_i \alpha_i = 0 \\ & 0 \leq \alpha_i \leq C \quad \forall i \end{aligned} \tag{2.81}$$

In the dual form, the decision boundary is parameterised by the Lagrange variables $\boldsymbol{\alpha}$. Each element of $\boldsymbol{\alpha}$ corresponds to a training example and determines the influence of the example on the position of the decision boundary. Given the dual variables, $\boldsymbol{\alpha}$, the primal weights and bias can be reclaimed using equations 2.82 and 2.83.

$$\boldsymbol{w} = \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{o}_i \tag{2.82}$$

$$b = -\frac{\max_{y_i=-1}(\langle \boldsymbol{w}, \boldsymbol{o}_i \rangle) + \min_{y_i=1}(\langle \boldsymbol{w}, \boldsymbol{o}_i \rangle)}{2} \tag{2.83}$$

where $\max_{y_i=-1}(\cdot)$ selects the training example from class $y = -1$ that is closest to the decision boundary and $\min_{y_i=1}(\cdot)$ selects the example from class $y = 1$ that is closest. By combining equations 2.82 and 2.76 the classification function can also be expressed in the dual form.

$$y^{\mathrm{v}} = \mathrm{sign}\left(\sum_{i=1}^{N} \alpha_i y_i \langle \boldsymbol{o}_i, \boldsymbol{o}^{\mathrm{v}} \rangle + b\right) \tag{2.84}$$

The optimal decision boundary may be obtained by solving either the primal or dual problems using standard quadratic programming methods such as gradient decent or quadratic programming (QP) optimisation [88]. Specialized algorithms such as Sequential Minimal Optimisation (SMO) [160] and the decomposition and chunking algorithms [94, 178] have also been developed to efficiently solve SVM optimisation problems. At optimality, the conditions expressed in equation 2.85, known as the Karush-Kuhn-Tucker (KKT) conditions, are true [41].

$$\alpha_i \left[y_i (\langle \boldsymbol{w}, \boldsymbol{o}_i \rangle + b) - 1 + \xi_i \right] = 0 \quad \forall i \tag{2.85}$$

An interesting consequence of the KKT conditions is that only the elements of $\boldsymbol{\alpha}$ associated with training examples lying on or within the margin are non-zero. This subset of training examples entirely determines the position of the decision boundary and is collectively known as the *support vectors* of the training set. Equation 2.84 can therefore be evaluated efficiently by only including support vectors in the summation.

The use of a soft margin allows a robust decision boundary to be obtained when linearly separable data is corrupted by noise. However, for many classification tasks the boundary between the classes is not linearly separable and a more complex representation is needed. One method of separating non-linear data without introducing additional model parameters is to introduce a static kernel function to implicitly map training examples into a a high-dimensional, separable, feature space. A linear decision boundary can then be obtained in the feature space that corresponds to a non-linear boundary in the input space. Static kernels, and their relationship to SVMs, are discussed in further detail in section 2.2.6.

2.2.4 Relevance vector machines

The relevance vector machine (RVM) [194, 195] is a form of discriminative classifier that is an implementation of the sparse Bayesian learning framework introduced in [137]. The RVM is closely related to the support vector machine and is defined in a similar form. However unlike the SVM, where the distance between a training example and the decision boundary does not have a probabilistic interpretation, for the RVM the output scores model the conditional likelihood of a class given the example and the current hyperparameter estimates. Thus the RVM is a conditional model, similar to the CRF and HCRF described in sections 2.2.1 and 2.2.2.

A common approach in statistical classification is to classify a test example $\boldsymbol{o}^{\mathrm{v}}$ based on a weighted linear combination of a set of basis functions $f_i(\boldsymbol{o}^{\mathrm{v}})$.

$$S(\boldsymbol{o}^{\mathrm{v}}; \boldsymbol{\alpha}, b) = \sum_{i=1}^{N} \alpha_i f_i(\boldsymbol{o}^{\mathrm{v}}) + b \tag{2.86}$$

where $f_i(\cdot)$ is typically non-linear. In the RVM model, each function $f_i(\cdot)$ is defined by a common function $k(\boldsymbol{o}^{\mathrm{v}}, \boldsymbol{o}_i)$ between $\boldsymbol{o}^{\mathrm{v}}$ and a training example $\boldsymbol{o}_i$. This yields a model of

similar functional form to the support vector machine.

$$S(\boldsymbol{o}^{\mathrm{v}};\boldsymbol{\alpha},b)=\sum_{i=1}^{N}\alpha_i k(\boldsymbol{o}^{\mathrm{v}},\boldsymbol{o}_i)+b \tag{2.87}$$

The properties of the RVM model are defined by the function $k(\boldsymbol{o}^{\mathrm{v}},\boldsymbol{o}_i)$. One class of function that may be applied is the class of static kernel functions. These implicitly map each vector $\boldsymbol{o}$ into a higher dimensional feature space, where an inner product is evaluated. Static kernel functions are described in more detail in section 2.2.6. For classification of speech data, dynamic kernel functions may be used as described in chapter 3. This approach was used successfully in [76]. Unlike standard kernel-based techniques such as the SVM classifier, where $k(\boldsymbol{o},\boldsymbol{o}_i)$ represents an inner product in some feature space, for the RVM $k(\boldsymbol{o},\boldsymbol{o}_i)$ is only used to derive a suitable set of basis functions. Hence, unlike standard kernel functions, for RVMs there is no requirement that $k(\boldsymbol{o},\boldsymbol{o}_i)$ satisfies Mercer's conditions.

In the SVM, a preference for smoother functions is introduced by using a maximum-margin training criterion. For the RVM, a similar preference for smooth functions is introduced using a Bayesian approach by defining a prior distribution over $\boldsymbol{\alpha}$. Here the prior distribution of the weights $\boldsymbol{\alpha}$ and bias b is modeled by a zero-mean Gaussian of the form

$$p(\boldsymbol{\alpha},b;\boldsymbol{\gamma})=\mathcal{N}(b;0,\gamma_0^{-1})\prod_{i=1}^{N}\mathcal{N}(\alpha_i;0,\gamma_i^{-1}) \tag{2.88}$$

By fixing the prior means at zero, the resultant weight vector tends to be extremely sparse. Hence the number of *relevance vectors* $\boldsymbol{o}_i$ with non-zero weights α_i is low. The relative sparsity of the RVM model, compared to the SVM, is one of its main advantages. Classification may be performed more efficiently, and the model is less prone to overfitting the data. Unlike the SVM, where the support vectors lie on the class boundary, here the relevance vectors tend to be representative of a particular class and typically lie far from the decision boundary [195].

For each weight α_i and the bias b, the inverse variance of the prior is defined by a hyperparameter γ_i. To complete the Bayesian specification for the model, priors must also be defined over γ_i. The hyperparameters γ_i define the scale of each basis function $f_i(\cdot)$. Hence, a suitable form of prior is the Gamma function. Alternatively, a non-informative prior over a logarithmic scale may be used [195]. One advantage of this approach is that the RVM predictions will be invariant to linear scalings of the basis function outputs.

The RVM classifier output is the posterior probability of one of the classes given $\boldsymbol{\alpha}$ and the test example $\boldsymbol{o}^{\mathrm{v}}$. For the binary case, $y^{\mathrm{v}}\in\{\omega_1,\omega_2\}$, the probability of the first class, ω_1 is obtained by applying the sigmoid link function to $S(\boldsymbol{o}^{\mathrm{v}};\boldsymbol{\alpha},b)$.

$$P(\omega_1|\boldsymbol{o}^{\mathrm{v}};\boldsymbol{\alpha},b)=\frac{1}{1+e^{-S(\boldsymbol{o}^{\mathrm{v}};\boldsymbol{\alpha},b)}} \tag{2.89}$$

The probability of the second class is equal to $1-P(\omega_1|\boldsymbol{o}^{\mathrm{v}};\boldsymbol{\alpha},b)$. For a training set $\mathcal{O}=\{\boldsymbol{o}_1,\ldots,\boldsymbol{o}_N\}$ with associated labels $\mathcal{Y}=\{y_1,\ldots,y_N\}$. the conditional likelihood of the training set labels is given by

$$P(\mathcal{Y}|\mathcal{O};\boldsymbol{\alpha},b)=\prod_{i=1}^{N}P(y_i|\boldsymbol{o}_i;\boldsymbol{\alpha},b) \tag{2.90}$$

The optimal hyperparameters $\{\boldsymbol{\gamma}^*, \boldsymbol{\alpha}^*, b^*\}$ are then obtained by maximising the posterior of the unknowns given the data.

$$\{\boldsymbol{\gamma}^*, \boldsymbol{\alpha}^*, b^*\} = \underset{\boldsymbol{\gamma},\boldsymbol{\alpha},b}{\operatorname{argmax}} \{p(\boldsymbol{\gamma}, \boldsymbol{\alpha}, b|\mathcal{Y}, \mathcal{O})\} \tag{2.91}$$

It is not possible to obtain a closed form solution to equation 2.91. Instead, an iterative procedure, based on Laplace's method, can be applied to obtain a suitable set of hyperparameter estimates [137]. At each iteration k, the most probable weights $\boldsymbol{\alpha}^{(k)}$ and bias $b^{(k)}$ are obtained, given the fixed values of $\boldsymbol{\gamma}^{(k-1)}$. This procedure is equivalent to optimising a penalized logistic model [195] and may be performed using iterative, second order Newton methods. The Hessian of equation 2.91 with respect to the weights and bias is then inverted and negated to obtain the covariance $\boldsymbol{\Sigma}^{(k)}$ of a Gaussian approximation to the posterior over the weights and bias centered at $\boldsymbol{\alpha}^{(k)}, b^{(k)}$. An update for the hyperparameters $\boldsymbol{\gamma}^{(k)}$ can then be obtained using the statistics $\boldsymbol{\alpha}^{(k)}$ and $\boldsymbol{\Sigma}^{(k)}$ [76].

$$\gamma_0^{(k)} = \frac{\rho_0}{b^{(k)2}} \tag{2.92}$$

$$\gamma_i^{(k)} = \frac{\rho_i}{\alpha_i^{(k)2}} \quad i > 0 \tag{2.93}$$

$$\rho_i = 1 - \gamma_i^{(k-1)} \sigma_i^{(k)2} \tag{2.94}$$

where $\sigma_i^{(k)2}$ is the ith diagonal element of $\boldsymbol{\Sigma}^{(k)}$. This iterative procedure is then repeated until convergence. Relevance vector machines may also be applied directly to multi-class classification [195]. However, as the number of classes increases, training quickly becomes computationally infeasible. Alternatively, a multi-class classifier may be constructed by combining multiple binary classifiers. Examples of this approach are given in the next section.

2.2.5 Multi-class classification

There are many classification problems where the number of potential labels is greater than two. These tasks are known as multi-class, or polychotomous, problems. For generative classification schemes, Bayes' decision rule can be applied to solve multi-class classification problems directly. In contrast, many discriminative classifiers, including the SVM, are inherently binary in nature. Although there has been interest in adapting the SVM to handle the multi-class problem directly, approaches such as [198, 208] yield quadratic optimisation problems where the size is proportional to the number of classes. Although more efficient approaches have since been developed [40] multi-class classification is still significantly more complex than the binary case. Similarly, although the relevance vector machine can be applied to multi-class tasks, training quickly becomes infeasible as the number of classes increases [195]. An alternative approach to solving multi-class problems is to break them down into a series of binary classification tasks. This process is known as a *reduction*. A selection of reduction schemes are presented in the following subsections.

2.2.5.1 One-versus-one classifiers

A popular scheme is the one-versus-one reduction scheme used in [59, 89]. Here a distinct binary classifier is trained to distinguish each pair of classes. For a classification task involving

K classes, a total of $K(K-1)/2$ different binary classifiers are required. For each pair of classes (ω_i, ω_j), the associated classifier is trained using only data labeled either ω_i or ω_j. One advantage of this scheme is that since each classifier is independent, they may be efficiently trained in parallel.

Given a set of trained one-versus-one classifiers, a number of methods are available for classifying test data. A commonly used scheme is majority voting [59]. Here, a test example is classified using each of the one-versus-one classifiers in turn. A vote is then assigned to class ω_i for each classifier that hypothesizes ω_i. Finally, the test example is assigned to the class that received the most votes. In the event of a tie between two classes (ω_i, ω_j), the test example is assigned to the class hypothesized by the classifier associated with pair (ω_i, ω_j). It is unclear how ties between larger numbers of competing classes should be resolved. One approach is to randomly assign the test example to a class. Alternatively, a radically different form of multi-class classifier, such as a generative classifier, may be used to distinguish between the competing classes.

Consider the case where each binary classier is *optimal*, i.e. the classifier associated with pair (ω_i, ω_j) correctly classifies any data belonging to class ω_i or ω_i and labels examples randomly otherwise. Under these conditions, the majority voting scheme will correctly classify all examples since the true class is guaranteed to receive a minimum of $K-1$ votes and each competing class will receive a maximum of $K-2$ votes. In practice this assumption is unrealistic and successive binary classification errors can cause the scheme to misclassify examples.

2.2.5.2 Directed acyclic graph classifiers

Various reduction schemes have been developed that use rooted binary directed acyclic graphs (DAGs) to classify test examples [161, 201]. Typically each leaf node is associated with a particular class label and a binary classifier is associated with each non-leaf node. A test example is then assigned to a particular class by following a path from the root to one of the leaf nodes. At each node, the test example is classified using the associated binary classifier. Depending on the classification decision, one of the two child nodes is then selected. This process is repeated until a leaf node is reached. The exact nature of the reduction scheme is dependent on the structure of the DAG and the binary decision made at each node.

One example of a reduction scheme based on DAGs is the decision DAG classifier [161]. An example binary-DAG for this scheme is depicted in figure 2.7(a). Each node in the graph is associated with a binary classifier, trained to distinguish between two classes ω_i and ω_j, as in the one-versus-one scheme. During classification, a set of potential classes is maintained. Initially, this set contains every class. At each level of the graph, test examples are classified using the binary classifier associated with the current node. After each decision, the losing class is discarded from the set and the corresponding edge is followed to determine the next node. Once a child node is reached only one possible hypothesised class remains and the test example can be assigned to this remaining class. Like the one-versus-one scheme, the decision DAG approach requires a total of $K(K-1)/2$ binary classifiers to be trained. However, only $K-1$ classification decisions are needed to assign a class to each test example. Although the structure of the decision DAG is dependent on the ordering of the classes, in practice, re-ordering the classes has not been found to significantly effect the performance of the scheme [161].

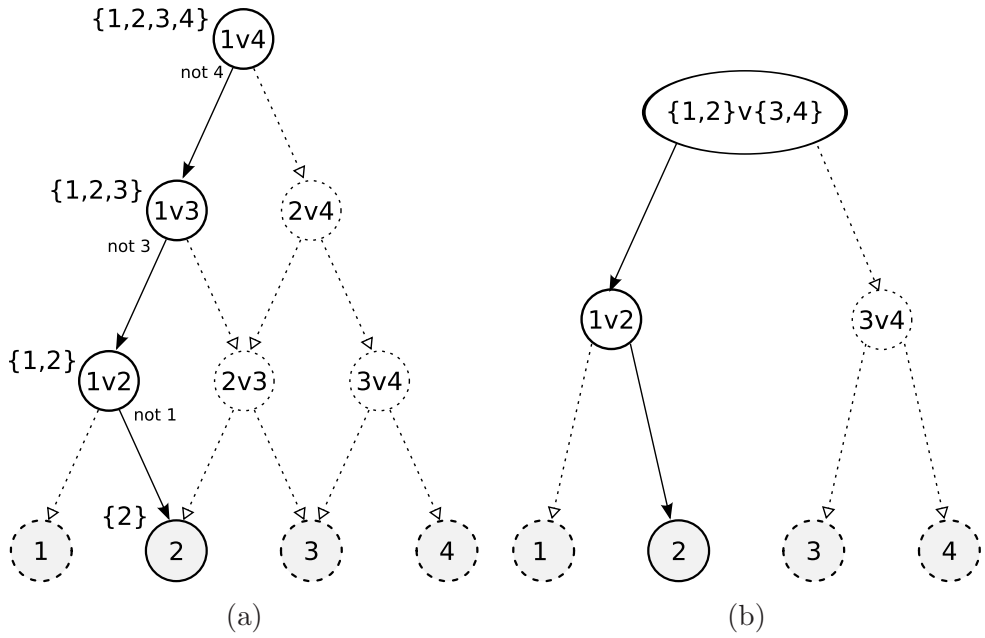


Figure 2.7: Multi-class classification using (a) a decision-DAG (b) a divide-by-two DAG.

An alternative reduction scheme based on binary DAGs is the divide-by-two (DB2) scheme proposed in [201]. An example DAG for the DB2 scheme is depicted in figure 2.7(b). Unlike the decision DAG classifier which discards a single class label at each level of the graph, here the number of potential labels is reduced by half after each classification decision. At each node, the remaining classes are partitioned into two subsets and a binary classifier is then trained to distinguish between these subsets. A suitable partitioning scheme may be obtained via clustering algorithms or by simply splitting the classes such that the number of training examples associated with each subset is roughly equal [201]. Unlike the decision DAG approach, only $K-1$ binary classifiers need to be trained and test examples may be classified using only $\log_2 K$ binary classifiers.

2.2.5.3 Filter trees

The filter tree [14, 70] is a DAG-based reduction scheme closely related to the DB2 scheme described in section 2.2.5.2. As in the DB2 scheme, the DAG is constrained to be a binary tree, and at each level a binary classifier is trained to distinguish between two competing sets of classes. Unlike the DB2 scheme, the filter tree is fundamentally a bottom-up approach. The training set associated with each node is dependent on the output of the classifiers associated with its descendants. An example filter tree is shown in figure 2.8.

The filter tree was developed for *cost-sensitive* classification tasks. For these tasks, there is a cost, $cost(\boldsymbol{o}, \omega_k)$, associated with classifying each example $\boldsymbol{o}$ as belonging to class ω_k. The cost associated with the true label is typically zero and the cost associated with misclassification is typically positive. The objective is then to classify the data in a way that minimises the total cost. Note that a cost-insensitive problem can be easily transformed into a cost-sensitive problem by setting $cost(\boldsymbol{o}, \omega_k) = 0$ if $\boldsymbol{o}$ belongs to class ω_k and $cost(\boldsymbol{o}, \omega_k) = 1$

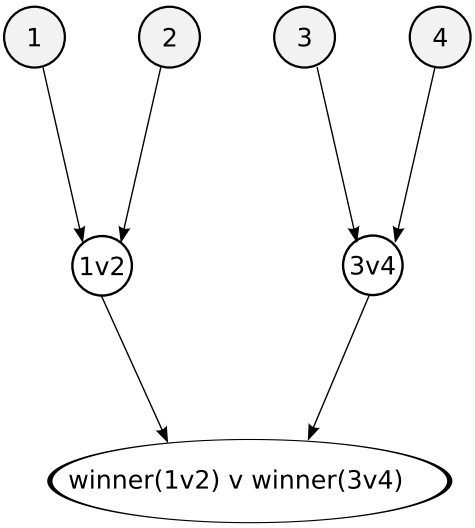


Figure 2.8: Multi-class classification using a filter tree.

otherwise. For cost-sensitive tasks, filter trees require the use of an *importance-weighted* binary classifier [14]. Here each training example is augmented with a parameter measuring the importance of it being classified correctly, determining its influence on the obtained decision boundary. For tasks where the importance is binary, the importance-weighted classifier may be substituted with a standard classifier, such as an SVM, by removing any zero-importance examples from the training set.

A test example is classified using a filter tree by an elimination contest. Starting from the leaf nodes, classes are initially grouped into pairs according to the graph. At each node, a binary classifier is used to assign the test example to one of the input classes. The predicted class label is then propagated to the next layer until the root node is reached. Finally, the test example is labeled with the class predicted by the root node. By tracing directly from the root node to the leaf, a test example can be classified using only $\log_2 K$ binary classifier evaluations.

The filter tree is also trained using a bottom-up procedure. Training examples are propagated through the tree as for classification. At each node, an importance-weighted binary classifier is trained to distinguish between the two sets of classes associated with the node. The importance associated with each training example is set equal to the difference between the costs of the classes predicted by the left and right subtrees. The effect of this is to penalize training examples misclassified by nodes closer to the leaves, yielding a more robust set of classifiers. The classification error associated with filter trees has been shown [14] to be bounded by $e \log_2 K$, where e is the average error rate of the binary classifiers.

2.2.5.4 Error correcting output codes

Error correcting output codes (ECOCs) [51] are a reduction scheme based on the analogy of transmitting data over a noisy channel. Here the class labels represent the data transmitted and any mistakes made by the binary classifiers are analogous to the effects of noise altering

individual bits of the transmitted data. Unlike more basic schemes, ECOCs are explicitly designed to recover from individual classifier mistakes.

ECOCs operate by assigning a codeword to each class. Each codeword is unique and consists of a length-L bitstring. Table 2.1 shows example codes for a three class problem. For each position l, the classes are partitioned into two sets depending on the value of the lth bit in the associated codewords. A binary classifier r_l is then trained to distinguish between the two sets. During test, examples are classified by each of the L binary classifiers and the outputs converted into a bitstring. By comparing this bitstring to the codewords associated with each potential label, the example can be assigned to a class. If any of the binary classifiers make mistakes, the resultant bitstring may not correspond directly to any of the codewords. In this case the example is assigned to the class with the 'nearest' codeword. A useful distance measure is the Hamming distance, defined as the number of bits that differ between the two strings.

Class	Codeword				
	r_1	r_2	r_3	r_4	r_5
ω_0	0	1	1	0	0
ω_1	1	0	1	0	1
ω_2	0	1	0	1	1

Table 2.1: A 5-bit error correcting output code for a three class problem.

The ability of the ECOC scheme to detect and correct for classification errors is related to the choice of codewords. If the minimum Hamming distance between any two codewords is equal to d, then the ECOC scheme can correct at most $(d-1)/2$ bit errors. In table 2.1, the minimum Hamming distance is three. Hence this code is able to correct a single binary misclassification. By choosing a set of codewords such that the minimum Hamming distance is maximised, the robustness of the scheme to classification errors is increased.

The use of the Hamming distance as a metric assumes that errors in any of the bit positions are uncorrelated. When two classifiers are trained to distinguish the same sets of classes this will not be the case. Hence, each column of the code table should be unique. Additionally, columns should not be complements of each other. This is not the case for the code defined in table 2.1. Here, the columns r_1 and r_2 are complementary and the associated classifiers will not be independent. Similarly columns r_3 and r_4 are complementary. For a K class classification task, a maximum of $2^{K-1}-1$ independent classifiers can be used (all-zero and all-one columns are not discriminative).

In an ideal code, the minimum Hamming distance between the rows and between both the columns and their complements will be large. Unless K is at least five it is difficult to satisfy both of these conditions. For larger number of classes obtaining a suitable code is a non-trivial task. One approach is to choose codewords by assigning meaning to individual bit positions. However this typically yields codes where the corresponding columns are highly correlated. Alternatively, suitable codes may be obtained algorithmically, either by selecting a subset of columns from an exhaustive list, or by iteratively updating a code using randomised hill climbing [51]. A number of adaptations to the basic ECOC scheme have also been proposed, including an extension to cost-sensitive classification [117] and schemes based on continuous rather than discrete codings [39].

2.2.6 Kernel functions

Many machine learning algorithms, including the support vector machine, can be expressed using a representation where the only reference to the data is via the inner product between training examples. One advantage of this representation is that only the inner products need to be stored in memory. When dealing with a small number of high dimensional training examples this can be significantly more efficient. A second benefit is that by replacing the inner product with a suitable *kernel function*, classifiers that are inherently linear, such as the support vector machine, can yield non-linear decision boundaries.

One method of separating non-linear data without introducing additional model parameters is to apply a non-linear function $\boldsymbol{\psi}(\boldsymbol{o})$ to map each example $\boldsymbol{o}$ into a high-dimensional *feature space*. The function $\boldsymbol{\psi}(\cdot)$ is often referred to as a *score operator* and the corresponding feature space as a *score space*. The mapped examples are known as *feature vectors*. According to Cover's theorem [38], given a sufficiently high-dimensional feature space, the classes can be made linearly separable with high probability. A linear decision boundary is then trained in the feature space, which will correspond to a non-linear boundary in the input space. When all references to data are in the form of inner products, rather than explicitly evaluating $\boldsymbol{\psi}(\boldsymbol{o})$, a (static) kernel function $k(\boldsymbol{o}_i, \boldsymbol{o}_j)$ may be defined that calculates the inner product in the feature space defined by $\boldsymbol{\psi}(\boldsymbol{o})$.

$$k(\boldsymbol{o}_i, \boldsymbol{o}_j) = \langle \boldsymbol{\psi}(\boldsymbol{o}_i), \boldsymbol{\psi}(\boldsymbol{o}_j) \rangle \tag{2.95}$$

By replacing the inner product with a kernel function, a *kernelised* form of the SVM decision function defined in equation 2.84 is obtained.

$$y^{\mathrm{v}} = \mathrm{sign}\left(\sum_{i=1}^{N} \alpha_i y_i k(\boldsymbol{o}_i, \boldsymbol{o}^{\mathrm{v}}) + b \right) \tag{2.96}$$

Similarly, the kernelised dual objective function has the form

$$\begin{aligned} \max \quad & \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j k(\boldsymbol{o}_i, \boldsymbol{o}_j) \\ \text{w.r.t.} \quad & \boldsymbol{\alpha} \\ \text{s.t.} \quad & \sum_{i=1}^{N} y_i \alpha_i = 0 \\ & 0 \leq \alpha_i \leq C \quad \forall i \end{aligned} \tag{2.97}$$

The advantage of this approach is that it is often possible to efficiently calculate $k(\boldsymbol{o}_i, \boldsymbol{o}_j)$ without explicitly mapping examples into the feature space. Since the dimension of feature spaces are typically extremely high, and potentially infinite, this can often yield considerable computational savings. To demonstrate this consider the following expansion function for two dimensional observations.

$$\boldsymbol{\psi}\left(\begin{bmatrix} o_1 \\ o_2 \end{bmatrix} \right) = \begin{bmatrix} o_1^2 \\ \sqrt{2} o_1 o_2 \\ o_2^2 \end{bmatrix} \tag{2.98}$$

The kernel function associated with this expansion can be expressed as follows:

$$\begin{aligned} k(\boldsymbol{o}_i, \boldsymbol{o}_j) &= \left\langle \begin{bmatrix} o_{i1}^2 \\ \sqrt{2}o_{i1}o_{i2} \\ o_{i2}^2 \end{bmatrix}, \begin{bmatrix} o_{j1}^2 \\ \sqrt{2}o_{j1}o_{j2} \\ o_{j2}^2 \end{bmatrix} \right\rangle \\ &= \left(\left\langle \begin{bmatrix} o_{i1} \\ o_{i2} \end{bmatrix}, \begin{bmatrix} o_{j1} \\ o_{j2} \end{bmatrix} \right\rangle \right)^2 \end{aligned} \tag{2.99}$$

Hence this kernel can be efficiently calculated as a function of the inner product of the original observations without performing an explicit expansion. To be a valid kernel, a function $k(\boldsymbol{o}_i, \boldsymbol{o}_j)$ must correspond to some feature space, however it is not necessary that this feature space can be explicitly expressed or even that it has finite dimension, so long as it can be proved to exist. Mercer's conditions [149, 181, 198] state that a necessary and sufficient condition for a function $k(\boldsymbol{o}_i, \boldsymbol{o}_j)$ to correspond to a valid feature space is that the Gram (or kernel) matrix $\boldsymbol{K}$ induced by $k(\boldsymbol{o}_i, \boldsymbol{o}_j)$ is symmetric and positive semi-definite, where $K_{ij} = k(\boldsymbol{o}_i, \boldsymbol{o}_j) \ \forall i \forall j$. Many forms of kernel function have been proposed for mapping vector data into a high-dimensional space. Several standard kernel functions are given in table 2.2. With the exception of the linear kernel, these kernels all correspond to feature spaces where the dimension varies with the kernel parameters. Although many kernels, such the polynomial kernel, have fixed-dimensional feature spaces, others, such as the Laplacian or Gaussian kernels, generate feature spaces where the dimension varies with the number of training examples.

Kernel	Functional form $k(\boldsymbol{o}_i, \boldsymbol{o}_j)$	Kernel parameters
Linear	$\langle \boldsymbol{o}_i, \boldsymbol{o}_j \rangle$	None
Homogeneous polynomial	$\langle \boldsymbol{o}_i, \boldsymbol{o}_j \rangle^p$	Power, p
Inhomogeneous polynomial	$(\langle \boldsymbol{o}_i, \boldsymbol{o}_j \rangle + c)^p$	Power, p; offset, c
Laplacian	$\exp\left(-\| \boldsymbol{o}_i - \boldsymbol{o}_j \| / 2\sigma^2\right)$	Width, σ
Gaussian	$\exp\left(-\| \boldsymbol{o}_i - \boldsymbol{o}_j \|^2 / 2\sigma^2\right)$	Variance, σ
Sigmoid	$\tanh\left(\kappa \langle \boldsymbol{o}_i, \boldsymbol{o}_j \rangle + c\right)$	Scale, κ; offset, c

Table 2.2: A summary of several standard kernel functions.

There is no requirement that kernel functions operate on vector data, as long as the corresponding kernel matrix satisfies Mercer's conditions. Hence, kernel functions may be defined that operate on alternative data structures such as sequences, graphs, text documents and sets. The modular nature of the kernel function allows standard machine learning techniques, such as SVM-based classification, to be applied to a wide range of tasks by selecting a task-appropriate kernel function. The predominant forms of kernel used in this thesis operate on sequences, and allow SVM classifiers to be applied to speech classification tasks. To distinguish these from kernels that operate on vector data, the term *dynamic kernel* is used to refer to kernels that operate on variable-length sequences and *static kernel* to refer to kernels that operate on vectors. Many examples of dynamic kernels suitable for speech classification are given in chapter 3.

Given a kernel function, it is possible to define new forms of kernel by modifying the function directly. For kernel functions $k^{\mathsf{a}}(\boldsymbol{o}_i, \boldsymbol{o}_j)$ and $k^{\mathsf{b}}(\boldsymbol{o}_i, \boldsymbol{o}_j)$ the following operations can be shown to yield valid kernel functions [181].

$$
\begin{aligned}
k(\boldsymbol{o}_i, \boldsymbol{o}_j) &= a k^{\mathsf{a}}(\boldsymbol{o}_i, \boldsymbol{o}_j) && (2.100)\\
k(\boldsymbol{o}_i, \boldsymbol{o}_j) &= k^{\mathsf{a}}(\boldsymbol{o}_i, \boldsymbol{o}_j) + \alpha && (2.101)\\
k(\boldsymbol{o}_i, \boldsymbol{o}_j) &= (k^{\mathsf{a}}(\boldsymbol{o}_i, \boldsymbol{o}_j))^p && (2.102)\\
k(\boldsymbol{o}_i, \boldsymbol{o}_j) &= \exp(k^{\mathsf{a}}(\boldsymbol{o}_i, \boldsymbol{o}_j)) && (2.103)\\
k(\boldsymbol{o}_i, \boldsymbol{o}_j) &= \exp(-\alpha ||\boldsymbol{o}_i - \boldsymbol{o}_j||^2) && (2.104)\\
k(\boldsymbol{o}_i, \boldsymbol{o}_j) &= k^{\mathsf{a}}(\boldsymbol{o}_i, \boldsymbol{o}_j) + k^{\mathsf{b}}(\boldsymbol{o}_i, \boldsymbol{o}_j) && (2.105)\\
k(\boldsymbol{o}_i, \boldsymbol{o}_j) &= k^{\mathsf{a}}(\boldsymbol{o}_i, \boldsymbol{o}_j) k^{\mathsf{b}}(\boldsymbol{o}_i, \boldsymbol{o}_j) && (2.106)
\end{aligned}
$$

where α is a positive real scalar and p is a positive integer. In chapters 6 and 7, these relationships are used to derive new forms of kernel function for speaker-verification by combining multiple dynamic and static kernel functions.

If a function $k(\boldsymbol{o}_i, \boldsymbol{o}_j)$ represents a valid kernel, it may be decomposed into the following form.

$$
k(\boldsymbol{o}_i, \boldsymbol{o}_j) = \boldsymbol{\psi}(\boldsymbol{o}_i)^{\mathsf{T}} \boldsymbol{G}^{-1} \boldsymbol{\psi}(\boldsymbol{o}_j) \tag{2.107}
$$

where $\boldsymbol{\psi}(\boldsymbol{o})$ is the associated score operator and $\boldsymbol{G}^{-1}$ is a square (potentially identity) matrix that does not depend on the data. The matrix $\boldsymbol{G}^{-1}$ therefore defines a distance metric in the feature space defined by $\boldsymbol{\psi}(\boldsymbol{o})$. When a kernel function is defined explicitly via a score operator, it is often not clear what form of metric should be used. Many kernel-based algorithms, including SVMs, are not invariant to scaling individual dimensions of the feature-space. Thus, the choice of unit used to measure each dimension of the feature-space will affect the decision boundary obtained. It is therefore useful to use a metric that is maximally non-committal with regards to the data since this will cause all dimensions to be treated equally regardless of scale. One such metric is given by

$$
\begin{aligned}
\boldsymbol{G} &= \mathcal{E}\left\{(\boldsymbol{\psi}(\boldsymbol{o}) - \boldsymbol{\mu}_{\boldsymbol{\psi}})(\boldsymbol{\psi}(\boldsymbol{o}) - \boldsymbol{\mu}_{\boldsymbol{\psi}})^{\mathsf{T}}\right\} && (2.108)\\
\boldsymbol{\mu}_{\boldsymbol{\psi}} &= \mathcal{E}\{\boldsymbol{\phi}(\boldsymbol{o})\} && (2.109)
\end{aligned}
$$

where $\mathcal{E}\{\}$ is the expectation with respect to $\boldsymbol{o}$. Since equation 2.108 will often have no closed form solution, the metric is often approximated by calculating the expectation over the available training examples [44]. This is equivalent to setting $\boldsymbol{G}$ to the covariance matrix of the feature vectors obtained from the training data.

$$
\boldsymbol{G} = \frac{1}{N-1} \sum_{i=1}^{N} \left[\boldsymbol{\psi}(\boldsymbol{o}_i) - \boldsymbol{\mu}_{\boldsymbol{\psi}}\right] \left[\boldsymbol{\psi}(\boldsymbol{o}_i) - \boldsymbol{\mu}_{\boldsymbol{\psi}}\right]^{\mathsf{T}} \tag{2.110}
$$

Rather than use equation 2.110 directly, it is common to use a further approximation and constrain $\boldsymbol{G}$ to be diagonal. This significantly reduces the computational cost of calculating and inverting $\boldsymbol{G}$ for high dimensional feature spaces.

For high-dimensional data, the dynamic range of the feature vectors may vary greatly, potentially reducing classification performance. One method of avoiding this issue is to

normalise all feature vectors to unit length. Unfortunately, such an approach can lead to wrap-around, where two distinct vectors map to the same point in feature space. Spherical normalisation [205] operates by mapping each feature vector to the surface of a unit hypersphere containing one more dimension than the original feature vector. The resulting feature vector has unit magnitude but information about the original magnitude is retained by the additional dimension. Feature vectors $\boldsymbol{\psi}(\boldsymbol{o})$ can be transformed to spherically normalised vectors $\boldsymbol{\psi}^{\mathtt{SN}}(\boldsymbol{o})$ using equation 2.111.

$$\boldsymbol{\psi}^{\mathtt{SN}}(\boldsymbol{o}) = \frac{1}{\sqrt{\langle \boldsymbol{\psi}(\boldsymbol{o}), \boldsymbol{\psi}(\boldsymbol{o}) \rangle + c^2}} \begin{bmatrix} \boldsymbol{\psi}(\boldsymbol{o}) \\ c \end{bmatrix} \tag{2.111}$$

where c is a positive constant. Alternatively spherical normalisation can be expressed in terms of the kernel function

$$k^{\mathtt{SN}}(\boldsymbol{o}_i, \boldsymbol{o}_j) = \frac{k(\boldsymbol{o}_i, \boldsymbol{o}_j) + c^2}{\sqrt{k(\boldsymbol{o}_i, \boldsymbol{o}_i) + c^2}\sqrt{k(\boldsymbol{o}_j, \boldsymbol{o}_j) + c^2}} \tag{2.112}$$

2.3 Summary

In this chapter, two general approaches for classifying speech utterances were introduced. Generative classification, discussed first, uses generative models to approximate the probability distribution associated with the process of generating speech observations. Given a generative model trained to model the likelihood of the speech observations associated with each class, classification may be performed using Bayes' decision rule. Two popular forms of generative model were presented: Gaussian mixture models, which model the likelihood of each observation using a mixture of Gaussian density functions, and hidden Markov models, which include latent variables to model additional temporal dependencies. Schemes for estimating and adapting generative model parameters were discussed.

The second half of this chapter introduced discriminative classification schemes. Unlike generative schemes these attempt to model the class boundaries directly. Two different types of discriminative scheme were discussed in this chapter, based on either discriminative models or discriminative functions. In the first approach, a conditional model is trained to directly model the posterior probability of the class labels. Two forms of conditional model were introduced in this chapter: the conditional random field and the related hidden conditional random field, which includes a latent state sequence to model additional dependencies. Discriminative functions are an alternative, non-statistical approach where a decision boundary is trained to separate the classes. A popular form of discriminative function, described in this chapter, is the support vector machine. This is a binary, distance-based classifier trained using a maximum-margin criterion. The relevance vector machine, a conditional model defined in a similar form to the support vector machine was also described. Schemes were then presented to allow these classifiers to be applied to multi-class tasks. A common property of relevance and support vector machines is that they can be kernelised. Here, a kernel function is applied to implicitly map examples into a high-dimensional feature space allowing non-linear decision boundaries to be efficiently trained.

Chapter 3
Dynamic Kernels

In the previous chapter it was shown that, through the use of a suitable kernel function, classifiers that are inherently linear can yield non-linear decision boundaries. Although kernel functions were originally developed for fixed-dimensional vector data, it possible to derive valid kernel functions that operate on many forms of data structure including graphs, sets and text documents. This has allowed many standard classification algorithms, such as the support vector machine, to be applied to a wide range of tasks by defining an appropriate kernel function.

In this chapter, a number of kernels are introduced that are designed to allow kernel-based classifiers to handle sequence data. To distinguish these from the static kernels introduced in section 2.2.6 for fixed-dimensional data, these kernels will be referred to as *dynamic* or sequence kernels[1], as in [204]. These dynamic kernels implicitly map each variable-length sequence into a fixed-dimensional feature space, before calculating the inner product in that space. Many of the kernels described in this chapter are examples of *generative kernels*. These combine generative and discriminative approaches by using generative models to define a suitable feature space within a discriminative kernel-based framework.

The dynamic kernels introduced in this chapter are divided into three broad categories based on the form of data to which they can be applied. The first two forms of dynamic kernel are discrete-observation kernels and continuous-observation kernels. These are designed to handle sequences of discrete or continuous data respectively. The final category of dynamic kernel described here is the distributional kernel. These kernels are closely related to the first two categories. However, rather than operate on sequence data directly, distributional kernels

[1] In this work the notation $K(\boldsymbol{O}_i, \boldsymbol{O}_j)$ is used to denote dynamic kernel functions and $\phi(\boldsymbol{O})$ to denote the associated dynamic score operator. $k(\boldsymbol{o}_i, \boldsymbol{o}_j)$ and $\psi(\boldsymbol{o})$ are reserved for static kernels.

define a function between two probability distributions. An example scheme for applying distributional kernels to the classification of sequence data is also described.

3.1 Discrete-observation kernels

Discrete-observation kernels are characterised by their ability to handle sequences of the form $\boldsymbol{O} = \{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_T\}$, where $\boldsymbol{o}_t$ is a discrete valued observation. Discrete-observation kernels were primarily developed for use in text-processing and biological applications. The extension of these kernels to speech-based classification tasks is described in further detail in chapter 4. A large number of dynamic kernels have been developed for discrete data including the bag-of-words kernel [93], string kernels [123, 128] and the marginalised count kernel [197].

3.1.1 Bag-of-words kernel

The bag-of-words (BOW) [93] kernel was one of the earliest kernels developed for handling sequences of discrete data and was originally applied to text classification. Here $\boldsymbol{O}$ represents a text sequence of length T and each observation $\boldsymbol{o}_t$ represents a word within that sequence. The set of possible words $\mathcal{W}$ that may occur within the text sequence is known as the vocabulary. In the simplest form, each text sequence is then mapped into a fixed-dimensional representation by taking a count, $count(w_k, \boldsymbol{O})$, of the number of times each word w_k appears in $\boldsymbol{O}$. In the resulting feature vector, the element at index k consists of the count associated with word w_k.

$$\boldsymbol{\phi}^{\texttt{BOW}}(\boldsymbol{O}) = \begin{bmatrix} count(w_1, \boldsymbol{O}) \\ \vdots \\ count(w_K, \boldsymbol{O}) \end{bmatrix} \tag{3.1}$$

The kernel function is then defined as the inner product of the feature vectors.

$$K^{\texttt{BOW}}(\boldsymbol{O}_i, \boldsymbol{O}_j) = \boldsymbol{\phi}^{\texttt{BOW}}(\boldsymbol{O}_i)^{\texttt{T}} \boldsymbol{\phi}^{\texttt{BOW}}(\boldsymbol{O}_j) \tag{3.2}$$

An issue that restricts the use of this form of kernel in practice is that the dimensionality of the feature space K, equal to the size of the vocabulary $\mathcal{W}$, will often be extremely large. Since $\boldsymbol{\phi}^{\texttt{BOW}}(\boldsymbol{O})$ will typically be sparse, it usually more efficient to only evaluate the kernel function over the set of words $\tilde{\mathcal{W}}$ known to appear in both $\boldsymbol{O}_i$ and $\boldsymbol{O}_j$.

$$K(\boldsymbol{O}_i, \boldsymbol{O}_j) = \sum_{w_k \in \tilde{\mathcal{W}}} count(w_k, \boldsymbol{O}_i)\, count(w_k, \boldsymbol{O}_j) \tag{3.3}$$

For text applications, rather than using each word directly, it is common to replace each word with its stem [93]. This reduces the total size of the vocabulary and ensures that related words such as `running`, `runner` and `runners` are treated identically. Similar procedures suitable for speaker-verification are discussed in more detail in chapter 4.

The BOW kernel defined in equation 3.3 is not invariant to changes in sequence length. By repeating the words contained in $\boldsymbol{O}_i$ the value of the kernel function will be doubled without necessarily changing the semantic content of the sequence. To compensate for this

a normalisation term may be included. In [93] the kernel function is normalised by the magnitude of each feature vector.

$$\hat{K}(\boldsymbol{O}_i, \boldsymbol{O}_j) = \frac{K(\boldsymbol{O}_i, \boldsymbol{O}_j)}{\sqrt{K(\boldsymbol{O}_i, \boldsymbol{O}_i)}\sqrt{K(\boldsymbol{O}_j, \boldsymbol{O}_j)}} \tag{3.4}$$

Alternatively the kernel function may be normalised by the length of each utterance to obtain a length-independent function.

$$\hat{K}(\boldsymbol{O}_i, \boldsymbol{O}_j) = \frac{K(\boldsymbol{O}_i, \boldsymbol{O}_j)}{T_i T_j} \tag{3.5}$$

Although conceptually simple, the BOW kernel is limited in practice since it does not take any form of word context into account. Permuting the words within each sentence will result in an identical kernel function even if the meaning of the sentence is radically changed. N-gram and marginalised count kernels, discussed in the following sections, are an attempt to overcome this restriction.

3.1.2 N-gram kernel

The N-gram kernel [124], also known as the spectrum kernel, extends the approach used in the BOW kernel to handle local context. Like the BOW kernel, the N-gram kernel was originally developed to handle classification of text sequences. Rather than treating each document as a sequence of words, the N-gram kernel is normally implemented on the level of individual characters. The kernel operates by comparing the substrings found within each sequence. For each sequence, the number of times that each possible substring of length N appears is counted. These counts are then used as a set of fixed-dimensional features that describe the sequence. For example, when N=3 the feature space has the following form.

$$\boldsymbol{\phi}^{\texttt{trigram}}(\boldsymbol{O}) = \begin{bmatrix} count(\texttt{aaa}, \boldsymbol{O}) \\ count(\texttt{aab}, \boldsymbol{O}) \\ \vdots \end{bmatrix} \tag{3.6}$$

where $count(v_k, \boldsymbol{O})$ indicated the number of time substring v_k occurs in sequence $\boldsymbol{O}$. For the case when N=1, the N-gram kernel has the form of a character-level BOW kernel. For larger values of N, the dimension of the feature vector is equal to K^N, where K is the number of characters (including the empty character) that may appear in the string. Hence for all but small values of N, evaluating the N-gram kernel can become infeasible. One approach for handling this issue is to exclude from the feature space any substrings that are considered unlikely to occur. For text classification using English documents this may include substrings such as 'hxj' or 'zqx'. Schemes for identifying likely substrings will be application dependent.

A closely related form of string kernel is the binary spectrum kernel [124]. Rather than recording a count of substrings, the element of the feature vector corresponding to each substring v_k is set to 1 if v_k occurs in $\boldsymbol{O}$ or 0 otherwise. One limitation of the N-gram approach is that only substrings of a particular fixed length are compared. For applications where insertions or deletions into the sequences are common, this may not be optimal. A number of kernels have been developed that extend the basic N-gram approach such as the mismatch kernel [125], the wildcard kernel [123] and the gappy N-gram kernel discussed in the following section.

3.1.3 Gappy N-gram kernel

The gappy N-gram kernel [123, 128], also known as the string subsequence kernel (SSK), is closely related to the N-gram kernel. Whereas the N-gram kernel measures the similarity of two sequences by considering the number of shared substrings of a given length, the gappy N-gram kernel attempts to extend this approach to non-contiguous substrings, where the elements of the subsequence are separated by one or more gaps. Thus the gappy N-gram kernel is designed to be tolerant to insertions of elements into the sequence.

Like the N-gram kernel, the feature space contains a dimension for each possible substring v_k of length N. However rather than taking a simple count of the number of times v_k appears in $\boldsymbol{O}$, the influence of each substring v_k on the kernel function is weighted according to how separated the elements of v_k are within $\boldsymbol{O}$. Substrings with consecutive elements are assigned a higher weighting than substrings with widely spaced elements. This is implemented by introducing a decay factor, ρ, where $0 \leq \rho \leq 1$. For each occurrence of a substring v_k the weighting is set to $\rho^{l(v_k)}$ where $l(v_k)$ is the length of the substring of $\boldsymbol{O}$ that completely encompasses the elements of v_k. Thus the element of the feature vector associated with substring v_k is given by

$$\phi_k^{\texttt{gappy}}(\boldsymbol{O}) = \sum_{v_k \in \boldsymbol{O}} \rho^{l(v_k)} \tag{3.7}$$

where the summation is over all instances of substring v_k in $\boldsymbol{O}$. To demonstrate the gappy N-gram kernel consider the text strings `ran`, `rat`, `can` and `cat`. For these sequences, there are eight possible substrings of length 2. The associated feature spaces for these strings are given below

	ϕ(ran)	ϕ(rat)	ϕ(can)	ϕ(cat)
r-a	ρ^2	ρ^2	0	0
a-n	ρ^2	0	ρ^2	0
r-n	ρ^3	0	0	0
r-t	0	ρ^3	0	0
a-t	0	ρ^2	0	ρ^2
c-a	0	0	ρ^2	ρ^2
c-t	0	0	0	ρ^3
c-n	0	0	ρ^3	0

Evaluating the gappy N-gram kernel by explicitly calculating the feature space is only possible for extremely short sequences and when N is small. However it is possible to evaluate the kernel more efficiently by using either dynamic programming techniques [128] or by using a retrieval tree data structure [123]. For both of these approaches the complexity of the algorithms scales linearly with both sequence length and N. Alongside many other forms of discrete-observation kernel, including the bag-of-words and N-gram kernels, gappy N-gram kernels may also be efficiently applied using weighted finite state transducers under a rational kernel framework [36, 37].

The effectiveness of the gappy N-gram kernel compared to the standard N-gram kernel is likely to be application-dependent. The gappy N-gram kernel was originally applied to the classification biological sequences [128]. In this case the insertion or deletion of sequence elements may occur frequently without substantially changing the relationships between sequences. By comparison, for text-processing applications the addition of characters within a

word often substantially changes the meaning, For example in the words `rat` and `rapt`. The suitability of the gappy N-gram kernel for a particular application is therefore related to how the users concept of sequence similarity is influenced by the insertion or deletion of sequence elements.

3.1.4 Term frequency log likelihood ratio kernel

The term frequency LLR (TF-LLR) kernel is a kernel function that operates on discrete sequence data and has been successfully applied to model higher-level features for speaker verification [29]. The TF-LLR kernel is motivated from an information retrieval perspective and is related to the term-frequency inverse-document-frequency (TF-IDF) measure. Rather than using substring counts directly as features, the probability $P(v_k|\boldsymbol{O})$ of the occurrence of a particular substring v_k in sequence $\boldsymbol{O}$ is calculated.

$$P(v_k|\boldsymbol{O}) = \frac{count(v_k, \boldsymbol{O})}{\sum_v count(v, \boldsymbol{O})} \tag{3.8}$$

where $count(v_k, \boldsymbol{O})$ is the count of the number of times that substring v_k appears in $\boldsymbol{O}$ and the summation is over all possible substrings. Typically only the substrings that are present in a large background dataset $\mathcal{O}^{\mathsf{B}} = \{\boldsymbol{O}^{\mathsf{B}}_1, \ldots, \boldsymbol{O}^{\mathsf{B}}_N\}$ are included. For two utterances $\boldsymbol{O}_i$ and $\boldsymbol{O}_j$, a similarity measure is obtained by determining whether the substrings present in $\boldsymbol{O}_i$ were more likely to be generated by the model $P(v_k|\boldsymbol{O}_j)$ associated with $\boldsymbol{O}_j$ or by the model $P(v_k|\mathcal{O}^{\mathsf{B}})$ associated with the background data. A suitable function $l(\boldsymbol{O}_i, \boldsymbol{O}_j)$ may be defined by taking the log-likelihood ratio of the two models averaged over all substrings present in $\boldsymbol{O}_i$.

$$l(\boldsymbol{O}_i, \boldsymbol{O}_j) = \frac{1}{\Gamma_i} \sum_{t=1}^{\Gamma_i} \log \frac{P(h_{it}|\boldsymbol{O}_j)}{P(h_{it}|\mathcal{O}^{\mathsf{B}})} \tag{3.9}$$

$$= \sum_{v_k} P(v_k|\boldsymbol{O}_i) \log \frac{P(v_k|\boldsymbol{O}_j)}{P(v_k|\mathcal{O}^{\mathsf{B}})} \tag{3.10}$$

where h_{it} is the tth sequential substring in $\boldsymbol{O}_i$ and Γ_i is the total number of substrings (including duplicates) present in $\boldsymbol{O}_i$. It is not possible to use $l(\boldsymbol{O}_i, \boldsymbol{O}_j)$ as a kernel function directly since it is not symmetric and does not satisfy Mercer's conditions. However, by taking a first order Taylor approximation to the $\log(\cdot)$ in equation 3.10, a valid TF-LLR kernel function may be obtained.

$$K^{\texttt{TFLLR}}(\boldsymbol{O}_i, \boldsymbol{O}_j) = \sum_{v_k} \frac{P(v_k|\boldsymbol{O}_i)}{\sqrt{P(v_k|\mathcal{O}^{\mathsf{B}})}} \frac{P(v_k|\boldsymbol{O}_j)}{\sqrt{P(v_k|\mathcal{O}^{\mathsf{B}})}} \tag{3.11}$$

For each utterance, the dynamic features associated with the TFLLR kernel are the frequency that each substring occurs normalised by the square root of the frequency that the substring occurs in the background data. This normalisation is similar to the inverse-document-frequency (IDF) measure in information retrieval. For IDF, typically the log of the background frequency is used rather than the square root. The TFLOG kernel [29], defined by equation 3.12, is a variant of the TF-LLR kernel based on this approach.

$$K^{\texttt{TFLOG}}(\boldsymbol{O}_i, \boldsymbol{O}_j) = \sum_{v_k} \frac{P(v_k|\boldsymbol{O}_i)}{\log P(v_k|\mathcal{O}^{\mathsf{B}}) + 1} \frac{P(v_k|\boldsymbol{O}_j)}{\log P(v_k|\mathcal{O}^{\mathsf{B}}) + 1} \tag{3.12}$$

3.1.5 Marginalised count kernel

Marginalised kernels were proposed in [197] as a principled method of incorporating context-dependency into a kernel function by modelling the context using a latent state sequence. Unlike the previous kernels discussed in this section, marginalised kernels combine standard kernel techniques with the generative approaches introduced in chapter 2. Here a latent variable model $P(\boldsymbol{O};\boldsymbol{\lambda})$ is introduced. For example, $P(\boldsymbol{O};\boldsymbol{\lambda})$ may be defined by a HMM with a discrete output distribution[1]. Each sequence, $\boldsymbol{O} = \{\boldsymbol{o}_1,\ldots,\boldsymbol{o}_T\}$, is assumed to be generated by the model according to a latent sequence $\boldsymbol{\theta} = \{\theta_1,\ldots,\theta_T\}$ that incorporates context information. Given the existence of a suitable kernel function, $\hat{K}(\{\boldsymbol{O}_i,\boldsymbol{\theta}_i\},\{\boldsymbol{O}_j,\boldsymbol{\theta}_j\})$, defined over joint observation/latent variable sequences, the marginalised kernel is obtained by marginalising over all possible latent state sequences Θ.

$$K(\boldsymbol{O}_i,\boldsymbol{O}_j) = \sum_{\boldsymbol{\theta}_i\in\Theta}\sum_{\boldsymbol{\theta}_j\in\Theta} P(\boldsymbol{\theta}_i|\boldsymbol{O}_i;\boldsymbol{\lambda})P(\boldsymbol{\theta}_j|\boldsymbol{O}_j;\boldsymbol{\lambda})\hat{K}(\{\boldsymbol{O}_i,\boldsymbol{\theta}_i\},\{\boldsymbol{O}_j,\boldsymbol{\theta}_j\}) \tag{3.13}$$

The precise nature of the marginalised kernel is dependent on both the latent variable model $P(\boldsymbol{O};\boldsymbol{\lambda})$ and the joint kernel function $\hat{K}(\{\boldsymbol{O}_i,\boldsymbol{\theta}_i\},\{\boldsymbol{O}_j,\boldsymbol{\theta}_j\})$. In practice, the form of joint kernel function that may be used is limited since, for many forms of joint-kernel, evaluating the summation over all possible latent state-sequences will be infeasible. One example of a joint kernel that leads to a tractable kernel function is the count kernel. This is closely related to the N-gram kernel described in section 3.1.2 except that for each substring v, distinct counts are maintained for each context in which the substring appears. For first (`mc1`) and second (`mc2`) order count kernels, the associated marginalised kernels have the following feature space.

$$\boldsymbol{\phi}^{\texttt{mc1}}(\boldsymbol{O}) = \begin{bmatrix} \vdots \\ \sum_{t=1}^{T} P(\theta_t = p \mid \boldsymbol{O};\boldsymbol{\lambda}) \\ \vdots \end{bmatrix} \tag{3.14}$$

$$\boldsymbol{\phi}^{\texttt{mc2}}(\boldsymbol{O}) = \begin{bmatrix} \vdots \\ \sum_{t=2}^{T} P(\theta_{t-1} = p, \theta_t = q \mid \boldsymbol{O};\boldsymbol{\lambda}) \\ \vdots \end{bmatrix} \tag{3.15}$$

where p and q indicate distinct contexts. When $\boldsymbol{\lambda}$ has the form of an HMM, p and q will represent HMM state indices and $P(\theta_t = p \mid \boldsymbol{O};\boldsymbol{\lambda})$ and $P(\theta_{t-1} = p, \theta_t = q \mid \boldsymbol{O};\boldsymbol{\lambda})$ may be efficiently estimated using the forward-backward algorithm described in chapter 2. For an N-state HMM, the total number of elements in the feature vector will be N for a first order marginalised count kernel and N^2 for a second order kernel. This approach may also be extended to higher order N-grams to incorporate more complex dependencies [197].

[1] Marginalised kernels may also be applied to sequences of continuous observations. In this case both GMMs and HMMs with GMM output distributions are appropriate forms of latent variable models to use.

3.2 Continuous-observation kernels

Continuous-observation kernels are a form of dynamic kernel designed to operate on sequences of continuous data. Unlike the discrete-observation kernels described in section 3.1, the dynamic kernels described in this section may be applied directly to utterances of speech, $\boldsymbol{O} = \{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_T\}$, where $\boldsymbol{o}_t$ is a vector of real-valued observations made at frame t. The majority of the continuous-observation kernels described in this section are examples of generative kernels, and make use of generative models to map each variable-length sequence into a fixed-dimensional space. Continuous-observation kernels described in this section include the GLDS kernel [24],the probabilistic sequence kernel [120] , the Fisher kernel [90], the LLR kernel [185], the MLLR kernel [189] and the CAT kernel [211].

3.2.1 GLDS Kernel

The Generalised Linear Discriminant Sequence (GLDS) kernel [24] was one of the earliest forms of dynamic kernel successfully applied to classification of speech sequences. It operates by initially mapping each observation $\boldsymbol{o}_t$ into a higher order feature space $\boldsymbol{\psi}(\boldsymbol{o}_t)$. Here $\boldsymbol{\psi}(\cdot)$ represents the feature-mapping associated with one of the static kernels introduced in chapter 2 for fixed-dimensional data. A duration-independent fixed-dimensional vector is then obtained by taking the mean of the expanded observations.

$$\boldsymbol{\phi}^{\texttt{GLDS}}(\boldsymbol{O}) = \frac{1}{T}\sum_{t=1}^{T}\boldsymbol{\psi}(\boldsymbol{o}_t) \tag{3.16}$$

The dimension of $\boldsymbol{\phi}^{\texttt{GLDS}}(\cdot)$ is dependent on both the dimension of the original observations $\boldsymbol{o}_t$ and the form of static expansion used. Typically, explicitly evaluating $\boldsymbol{\phi}(\boldsymbol{O})$ is only possible for limited forms for kernel function such as linear, or low order polynomial kernels. However, this issue may be avoided by expressing the GLDS kernel in terms of a series of static kernel evaluations. When an identity metric is used, this yields the dynamic kernel function.

$$K^{\texttt{GLDS}}(\boldsymbol{O}_i, \boldsymbol{O}_j) = \frac{1}{T_i T_j}\sum_{t=1}^{T_i}\sum_{\tau=1}^{T_j} k(\boldsymbol{o}_{it}, \boldsymbol{o}_{j\tau}) \tag{3.17}$$

where $k(\boldsymbol{o}_{it}, \boldsymbol{o}_{j\tau})$ is a static kernel. When equation 3.17 is used to evaluate the kernel function, standard forms of static kernel, such as polynomial or Gaussian kernels, may be efficiently applied.

The GLDS kernel makes use of static kernels defined at the level of individual observations to map sequences into a more separable feature space. In chapter 7 this approach is extended and combined with generative schemes to derive more general dynamic kernels suitable for speech classification. One disadvantage to using the GLDS kernel is that the static kernel function must be calculated between all pairs of observations. For longer utterances this can be computationally expensive. The generalised kernels described in chapter 7 do not suffer from this restriction.

3.2.2 Log-likelihood kernels

Given a suitable function, $\boldsymbol{\phi}(\boldsymbol{O})$, that maps sequences of continuous observations into a fixed-dimensional feature space, a dynamic kernel can easily be defined. The GLDS kernel, described in section 3.2.1, maps each sequence into a fixed-dimensional space by taking an average of the (expanded) sequence elements. This approach is unlikely to be robust, since useful information is lost by the averaging process. A more principled way of obtaining a suitable function $\boldsymbol{\phi}(\boldsymbol{O})$ is through the use of generative models, introduced in chapter 2. Given a generative model, the output of $p(\boldsymbol{O};\boldsymbol{\lambda})$ maps each sequence into a fixed (one) dimensional feature space.

$$\boldsymbol{\phi}^{\texttt{likelihood}}(\boldsymbol{O}) = \frac{1}{T}\left[\log p(\boldsymbol{O};\boldsymbol{\lambda})\right] \tag{3.18}$$

Equation 3.18 includes a length normalisation term. This is important when the length of sequences vary greatly within the dataset. For GMM models with parameters $\boldsymbol{\lambda} = \{\boldsymbol{c}, \boldsymbol{\mu}, \boldsymbol{\Sigma}\}$ the likelihood of $\boldsymbol{O}$ is given by

$$p(\boldsymbol{O};\boldsymbol{\lambda}) = \prod_{t=1}^{T}\sum_{m=1}^{M} c_m \mathcal{N}(\boldsymbol{o}_t;\boldsymbol{\mu}_m,\boldsymbol{\Sigma}_m) \tag{3.19}$$

Alternatively, the likelihood may be obtained using a hidden Markov model. Here the likelihood is obtained as the summation over the set Θ of latent state sequences.

$$p(\boldsymbol{O};\boldsymbol{\lambda}) = \sum_{\boldsymbol{\theta}\in\Theta}\prod_{t=1}^{T} P(\theta_t|\theta_{t-1})p(\boldsymbol{o}_t|\theta_t) \tag{3.20}$$

where $p(\boldsymbol{o}_t|\theta_t)$ is a state-dependent output distribution typically modelled using a GMM. There has been interest in extending this approach to obtain higher-dimensional feature sets. One simple approach used in [192, 218] is anchor modelling. Here a fixed-dimensional set of features is obtained by evaluating the log-likelihood of $\boldsymbol{O}$ given a set of N reference models. This yields the following feature space.

$$\boldsymbol{\phi}^{\texttt{anchor}}(\boldsymbol{O}) = \frac{1}{T}\begin{bmatrix} \log p(\boldsymbol{O};\boldsymbol{\lambda}^{(1)}) \\ \vdots \\ \log p(\boldsymbol{O};\boldsymbol{\lambda}^{(N)}) \end{bmatrix} \tag{3.21}$$

The nature of this kernel is determined by the set of reference models used. To ensure that the features obtained are complementary, it is important that the models are varied and the probability density functions effectively span the dataset. In practice, anchor modelling is rarely used as it has been found to perform poorly compared to alternative forms of speech classifier [192]. However it provides an interesting contrast to the remaining dynamic kernels described in this section which also attempt to extend the basic likelihood kernel approach.

3.2.3 Fisher kernel

The Fisher kernel [90] is one of the most commonly used forms of dynamic kernel for classifying sequences of continuous data. Like the related likelihood kernel, introduced in section 3.2.2, the feature space associated with the Fisher kernel is derived using a generative model $p(\boldsymbol{O};\boldsymbol{\lambda})$. Thus the Fisher kernel is a member of the family of generative kernels [184]. Since the likelihood kernel only extracts a single feature from each utterance, its ability to discriminate between classes is limited. The Fisher kernel attempts to exploit the nature of the generative model to capture differences in the generative processes for different classes. This is done by defining the feature space using the gradients of the generative model.

$$\boldsymbol{\phi}^{\mathtt{F}}(\boldsymbol{O};\boldsymbol{\lambda}) \quad = \quad [\boldsymbol{\nabla}_{\boldsymbol{\lambda}} \log p(\boldsymbol{O};\boldsymbol{\lambda})] \tag{3.22}$$

where $\boldsymbol{\nabla}_{\boldsymbol{\lambda}} \log p(\boldsymbol{O};\boldsymbol{\lambda})$ indicates the vector of partial derivatives of $\log p(\boldsymbol{O};\boldsymbol{\lambda})$ with respect to each of the generative model parameters, $\boldsymbol{\lambda}$. Since these features correspond to the score of the likelihood function, the feature vectors associated with the Fisher kernel are often referred to as *Fisher scores*. When $p(\boldsymbol{O};\boldsymbol{\lambda})$ is a member of the exponential family, the Fisher scores form sufficient statistics for $\log p(\boldsymbol{O};\boldsymbol{\lambda})$ under the natural parameterisation $\boldsymbol{\lambda}$ [90]. Thus they provide a well-motivated expansion of the likelihood function to multiple dimensions. Fisher kernels are usually defined using a maximally non-committal metric.

$$K^{\mathtt{F}}(\boldsymbol{O}_i,\boldsymbol{O}_j) = \boldsymbol{\phi}^{\mathtt{F}}(\boldsymbol{O}_i;\boldsymbol{\lambda})^{\mathtt{T}}\boldsymbol{G}^{-1}\boldsymbol{\phi}^{\mathtt{F}}(\boldsymbol{O}_j;\boldsymbol{\lambda}) \tag{3.23}$$

where the metric $\boldsymbol{G}$ is given by

$$\boldsymbol{G} = \mathcal{E}\left\{(\boldsymbol{\phi}^{\mathtt{F}}(\boldsymbol{O};\boldsymbol{\lambda}) - \boldsymbol{\mu}_{\boldsymbol{\phi}})(\boldsymbol{\phi}^{\mathtt{F}}(\boldsymbol{O};\boldsymbol{\lambda}) - \boldsymbol{\mu}_{\boldsymbol{\phi}})^{\mathtt{T}}\right\} \tag{3.24}$$

where $\mathcal{E}\{\}$ is defined with respect to $\boldsymbol{O}$ and $\boldsymbol{\mu}_{\boldsymbol{\phi}}$ is the expected value of the feature vectors defined by $\boldsymbol{\mu}_{\boldsymbol{\phi}} = \mathcal{E}\{\boldsymbol{\phi}^{\mathtt{F}}(\boldsymbol{O};\boldsymbol{\lambda})\}$. When the parameters, $\boldsymbol{\lambda}$, of the generative model are estimated to maximise the ML criterion the score space has zero mean. Under these conditions, the metric has the form of the Fisher information matrix $\boldsymbol{G}^{\mathtt{FI}}$, defined as the variance of the Fisher scores.

$$\boldsymbol{G}^{\mathtt{FI}} = \mathcal{E}\left\{[\boldsymbol{\nabla}_{\boldsymbol{\lambda}} \log p(\boldsymbol{O};\boldsymbol{\lambda})]\,[\boldsymbol{\nabla}_{\boldsymbol{\lambda}} \log p(\boldsymbol{O};\boldsymbol{\lambda})]^{\mathtt{T}}\right\} \tag{3.25}$$

The elements of the Fisher information matrix provide a measure of the amount of information that $\boldsymbol{O}$ carries about each parameter in $\boldsymbol{\lambda}$. As discussed in section 2.2.6, there is often no closed form for equation 3.25. Instead, the metric is usually approximated using a diagonalised expectation over the examples in the training dataset $\mathcal{O} = \{\boldsymbol{O}_1,\ldots,\boldsymbol{O}_N\}$ [44].

The exact nature of the Fisher kernel features is dependent on the form of the generative model $p(\boldsymbol{O};\boldsymbol{\lambda})$. For GMMs, first order derivatives of $\log p(\boldsymbol{O};\boldsymbol{\lambda})$ with respect to the mean ($\boldsymbol{\mu}_m$), covariance ($\boldsymbol{\Sigma}_m$) and component prior (c_m) associated with component m are defined

by

$$\nabla_{\boldsymbol{\mu}_m} \log p(\boldsymbol{O};\boldsymbol{\lambda}) = \sum_{t=1}^{T} P(\theta_t = m|\boldsymbol{o}_t;\boldsymbol{\lambda})\boldsymbol{\Sigma}_m^{-1}(\boldsymbol{o}_t - \boldsymbol{\mu}_m) \quad (3.26)$$

$$\nabla_{\boldsymbol{\Sigma}_m} \log p(\boldsymbol{O};\boldsymbol{\lambda}) = \sum_{t=1}^{T} \frac{P(\theta_t = m|\boldsymbol{o}_t;\boldsymbol{\lambda})}{2}\left[-\boldsymbol{\Sigma}_m^{-1} + \boldsymbol{\Sigma}_m^{-1}(\boldsymbol{o}_t-\boldsymbol{\mu}_m)(\boldsymbol{o}_t-\boldsymbol{\mu}_m)^{\mathsf{T}}\boldsymbol{\Sigma}_m^{-1}\right] \quad (3.27)$$

$$\nabla_{c_m} \log p(\boldsymbol{O};\boldsymbol{\lambda}) = \sum_{t=1}^{T} \frac{P(\theta_t = m|\boldsymbol{o}_t;\boldsymbol{\lambda})}{c_m} - 1 \quad (3.28)$$

where $P(\theta_t = m|\boldsymbol{o}_t;\boldsymbol{\lambda})$ is the posterior probability of $\boldsymbol{o}_t$ being emitted by component m. It is also possible to include higher order derivative features into the score space, however these have previously been found to contain little useful discriminative information [119].

Alternatively, a HMM may be used as the generative model. Here the first order derivatives with respect to the mean ($\boldsymbol{\mu}_{jm}$), covariance ($\boldsymbol{\Sigma}_{jm}$), component priors (c_{jm}) associated with component m of state j and transition probability (a_{ij}) from state i to state j are given by

$$\nabla_{\boldsymbol{\mu}_{jm}} \log p(\boldsymbol{O};\boldsymbol{\lambda}) = \sum_{t=1}^{T} P(\theta_t = \{j,m\}|\boldsymbol{O};\boldsymbol{\lambda})\boldsymbol{\Sigma}_{jm}^{-1}(\boldsymbol{o}_t - \boldsymbol{\mu}_{jm}) \quad (3.29)$$

$$\nabla_{\boldsymbol{\Sigma}_{jm}} \log p(\boldsymbol{O};\boldsymbol{\lambda}) = \sum_{t=1}^{T} \frac{P(\theta_t = \{j,m\}|\boldsymbol{O};\boldsymbol{\lambda})}{2}\left[-\boldsymbol{\Sigma}_{jm}^{-1}+\boldsymbol{\Sigma}_{jm}^{-1}(\boldsymbol{o}_t-\boldsymbol{\mu}_{jm})(\boldsymbol{o}_t-\boldsymbol{\mu}_{jm})^{\mathsf{T}}\boldsymbol{\Sigma}_{jm}^{-1}\right] \quad (3.30)$$

$$\nabla_{c_{jm}} \log p(\boldsymbol{O};\boldsymbol{\lambda}) = \sum_{t=1}^{T} \frac{P(\theta_t = \{j,m\}|\boldsymbol{O};\boldsymbol{\lambda})}{c_{jm}} - P(\theta_t = j|\boldsymbol{O};\boldsymbol{\lambda}) \quad (3.31)$$

$$\nabla_{a_{ij}} \log p(\boldsymbol{O};\boldsymbol{\lambda}) = \sum_{t=1}^{T} \frac{P(\theta_{t-1} = i, \theta_t = j|\boldsymbol{O};\boldsymbol{\lambda})}{a_{ij}} - P(\theta_{t-1} = i|\boldsymbol{O};\boldsymbol{\lambda}) \quad (3.32)$$

The derivative features associated with the HMM component priors and transition probabilities are closely related to the features of the first and second order marginalised count kernels. Unlike the marginalised count kernels, equations 3.31 and 3.32 both contain an additional term involving the state posterior probabilities. These terms arises from the sum-to-one constraint on the component priors and state transition probabilities and act to centre the derivatives within the feature space. The derivatives also contain additional scaling terms, c_{jm} and a_{ij}. Although these alter the dynamic range of the features, under a maximally non-committal metric they will be normalised. Depending on the amount of available data it may not be possible to robustly generate a complete set of derivative features. In this case it is common to only include a restricted set of derivative features, such as first order derivatives with respect to component means. Since these derivatives typically scale with the duration of the utterance, it is also common to include an additional duration normalisation term. When GMMs are used, this yields the following score space.

$$\boldsymbol{\phi}(\boldsymbol{O};\boldsymbol{\lambda}) = \frac{1}{T}\left[\sum_{t=1}^{T} P(\theta_t = m|\boldsymbol{O};\boldsymbol{\lambda})\boldsymbol{\Sigma}_m^{-1}(\boldsymbol{o}_t - \boldsymbol{\mu}_m)\right] \quad (3.33)$$

Although Fisher kernels are typically defined using a maximally non-committal metric it is possible to combine the score space in equation 3.33 with alternative forms of metric, such

as the identity metric. Since each feature is normalised by the component variance, this can yield features that vary greatly in dynamic range. SVM classifiers are not scale-invariant so this will affect the decision. One approach to handling this issue, used in [67], is to instead normalise each feature by the component standard deviation. This keeps the dynamic range of each set of feature consistent. Note that this is not normally an issue when a maximally non-committal metric is used, as the dynamic range effects are handled by the metric.

3.2.4 Log-likelihood ratio kernel

A potential issue that can occur when using the Fisher kernel with latent-variable models such as GMMs or HMMs is 'wrap-around' [184]. This occurs when multiple regions of the input space map to the same point in the feature space. To illustrate this, consider a GMM with two widely spaced Gaussian components. For an observation located exactly at the mean of one of these components, the derivative with respect to either of the component means is zero. The first due to a zero gradient, the second due to a zero posterior. Similarly, a second observation emitted at the mean of the other component will also have a zero derivative with respect to both component means. Thus, the two distinct observations yield identical feature vectors.

One approach to resolving this problem is to additionally include derivative features that do not suffer from 'wrap-around', such as derivatives with respect to the component priors. For binary classification problems, another solution to this issue is to instead take derivatives with respect to the log-likelihood ratio (LLR) between the two classes, defined by

$$\text{LLR} = \log p(\boldsymbol{O}; \boldsymbol{\lambda}^{(1)}) - \log p(\boldsymbol{O}; \boldsymbol{\lambda}^{(2)}) \tag{3.34}$$

Unlike the Fisher kernel which uses a single generative model, the LLR kernel [185] requires two generative models $\boldsymbol{\lambda}^{(1)}$ and $\boldsymbol{\lambda}^{(2)}$, trained using ML on the data from each class respectively. By incorporating multiple generative models into the feature space, wrap-around is less likely to occur. The LLR feature space is defined by

$$\boldsymbol{\phi}^{\text{LLR}}(\boldsymbol{O}; \boldsymbol{\lambda}^{(1)}, \boldsymbol{\lambda}^{(2)}) = \frac{1}{T} \begin{bmatrix} \log p(\boldsymbol{O}; \boldsymbol{\lambda}^{(1)}) - \log p(\boldsymbol{O}; \boldsymbol{\lambda}^{(2)}) \\ \boldsymbol{\nabla}_{\boldsymbol{\lambda}^{(1)}} \log p(\boldsymbol{O}; \boldsymbol{\lambda}^{(1)}) \\ -\boldsymbol{\nabla}_{\boldsymbol{\lambda}^{(2)}} \log p(\boldsymbol{O}; \boldsymbol{\lambda}^{(2)}) \end{bmatrix} \tag{3.35}$$

Note that the LLR itself is usually included as a feature in the score space. Since the generative model parameters $\boldsymbol{\lambda}^{(1)}$ $\boldsymbol{\lambda}^{(2)}$ are assumed to be independent, equation 3.35 does not include any cross derivative terms of the form $\boldsymbol{\nabla}_{\boldsymbol{\lambda}^{(1)}} \log p(\boldsymbol{O}; \boldsymbol{\lambda}^{(2)})$. Like the Fisher Kernel, the LLR kernel is usually defined using a maximally non-committal metric.

$$K^{\text{LLR}}(\boldsymbol{O}_i, \boldsymbol{O}_j) = \boldsymbol{\phi}^{\text{LLR}}(\boldsymbol{O}_i; \boldsymbol{\lambda}^{(1)}, \boldsymbol{\lambda}^{(2)})^{\text{T}} \boldsymbol{G}^{-1} \boldsymbol{\phi}^{\text{LLR}}(\boldsymbol{O}_j; \boldsymbol{\lambda}^{(1)}, \boldsymbol{\lambda}^{(2)}) \tag{3.36}$$

where $\boldsymbol{G}$ is the expected value of the covariance of the score space. Unlike the Fisher kernel, $\boldsymbol{G}$ does not have the form of a Fisher information matrix due to the additional LLR score space term. The Fisher kernel may be applied to either labeled or unlabeled data, since the class labels are not required to estimate $\boldsymbol{\lambda}$. In contrast, the LLR kernel can only be applied when labeled data from both classes is available to train $p(\boldsymbol{O}; \boldsymbol{\lambda}^{(1)})$ and $p(\boldsymbol{O}; \boldsymbol{\lambda}^{(2)})$.

3.2.5 Probabilistic sequence kernel

The probabilistic sequence kernel (PSK) [120] is an alternative approach to extend the likelihood kernel to obtain a higher-dimensional feature space. Rather than use $p(\boldsymbol{O};\boldsymbol{\lambda})$ directly, for the PSK a fixed-dimensional set of features is obtained by evaluating the posterior probability of each model component given $\boldsymbol{O}$. For GMMs this yields the following PSK feature space.

$$\boldsymbol{\phi}^{\text{PSK}}(\boldsymbol{O}) = \frac{1}{T}\begin{bmatrix} \vdots \\ \sum_{t=1}^{T} P(\theta_t = m|\boldsymbol{o}_t;\boldsymbol{\lambda}) \\ \vdots \end{bmatrix} \tag{3.37}$$

where $P(\theta_t = m|\boldsymbol{O};\boldsymbol{\lambda})$ is the posterior probability of $\boldsymbol{o}_t$ being emitted by component m. From Bayes' theorem this is defined by

$$P(\theta_t = m|\boldsymbol{O};\boldsymbol{\lambda}) = \frac{c_m \mathcal{N}(\boldsymbol{o};\boldsymbol{\mu}_m,\boldsymbol{\Sigma}_m)}{\sum_{n=1}^{M} c_n \mathcal{N}(\boldsymbol{o};\boldsymbol{\mu}_n,\boldsymbol{\Sigma}_n)} \tag{3.38}$$

The features associated with the PSK are closely related to the partial derivatives of the utterance log-likelihood with respect to the component priors. These are given by

$$\boldsymbol{\nabla}_{c_m} \log p(\boldsymbol{O};\boldsymbol{\lambda}) \quad = \quad \sum_{t=1}^{T} \frac{P(\theta_t = m|\boldsymbol{o}_t;\boldsymbol{\lambda})}{c_m} - 1 \tag{3.39}$$

The only difference between the two sets of features is that each derivative is scaled by the associated component prior and that a constant term is subtracted from all derivatives. Since this corresponds to a constant scaling and translation of each dimension of the feature vector, both sets of features will yield the same decision boundary when the kernels are combined with (linear) SVMs using a maximally non-committal metric. Thus, the features of the PSK are closely related to the features of the Fisher kernel, when partial derivatives with respect to the component priors are used.

The PSK is also related to the GLDS kernel. Whereas the GLDS kernel utilises a static kernel function to map each sequence element into a higher dimensional space, for the PSK the static mapping function is defined by a generative model [120]. A suitable kernel function may be defined by taking the inner product of the features generated by equation 3.37 using either an identity metric or a maximally non-committal metric. Alternatively, as in [120], a suitable decorrelating metric may be motivated from an RBF network [139].

3.2.6 MLLR kernel

A recently developed form of dynamic kernel that has been successfully applied to classify speech utterances is the MLLR kernel [104, 189, 190]. Here the feature space is defined by the parameters of an MLLR transform [122]. Given an utterance $\boldsymbol{O}_i$ and a suitable HMM/GMM background model $\boldsymbol{\lambda}$, a set of utterance-dependent transforms are trained, as described in chapter 2. For GMM models, the adapted mean $\boldsymbol{\mu}_m^{(i)}$ and covariance $\boldsymbol{\Sigma}_m^{(i)}$ parameters associated with component m are defined by

$$\boldsymbol{\mu}_m^{(i)} \quad = \quad \boldsymbol{A}_r^{(i)}\boldsymbol{\mu}_m + \boldsymbol{b}_r^{(i)} \tag{3.40}$$

$$\boldsymbol{\Sigma}_m^{(i)} \quad = \quad \boldsymbol{H}_r^{(i)}\boldsymbol{\Sigma}_m\boldsymbol{H}_r^{(i)^{\mathsf{T}}} \tag{3.41}$$

where r is the regression class associated with component m and $\boldsymbol{A}_r^{(i)}$,$\boldsymbol{b}_r^{(i)}$ and $\boldsymbol{H}_r^{(i)}$ are the utterance-dependent transform parameters associated with regression class r. The MLLR feature space is constructed by concatenating the set of transform parameters into a feature vector.

$$\boldsymbol{\phi}^{\texttt{MLLR}}(\boldsymbol{O}_i) = \begin{bmatrix} \text{vec}(\boldsymbol{A}_1^{(i)}) \\ \boldsymbol{b}_1^{(i)} \\ \text{vec}(\boldsymbol{H}_1^{(i)}) \\ \vdots \\ \text{vec}(\boldsymbol{A}_R^{(i)}) \\ \boldsymbol{b}_R^{(i)} \\ \text{vec}(\boldsymbol{H}_R^{(i)}) \end{bmatrix} \tag{3.42}$$

where vec$(\cdot)$ is a function that transforms the elements of a matrix row-wise into a vector. Since the MLLR transform parameters are sufficient to encapsulate all the utterance-dependent information contained within the adapted model, they can potentially be used to generate an extremely compact set of utterance-dependent features. The dimension of the feature space can be adjusted, for example by varying the number of regression classes, or by using block-diagonal transforms. Alternatively, only a subset of transform parameters may be included in the feature space. The performance of the MLLR kernel is closely related to the form of the background model used. For speech classification applications, typically the background model consists of a trained HMM-based speech recogniser [189, 190] although GMM background models have also been used [103, 104]. It is important that all MLLR transforms are generated from the same background model otherwise the feature space obtained will not be consistent.

A closely related form of kernel is the CMLLR kernel used in [58]. Instead of adapting the background model using MLLR, CMLLR transforms are generated for each utterance. For each regression class r, a single transform is used to adapt both the mean and the covariance parameters associated with each component.

$$\boldsymbol{\mu}_m^{(i)\texttt{CMLLR}} = \boldsymbol{A}_r^{(i)}\boldsymbol{\mu}_m + \boldsymbol{b}_r^{(i)} \tag{3.43}$$

$$\boldsymbol{\Sigma}_m^{(i)\texttt{CMLLR}} = \boldsymbol{A}_r^{(i)}\boldsymbol{\Sigma}_m\boldsymbol{A}_r^{(i)^{\texttt{T}}} \tag{3.44}$$

The CMLLR kernel is then defined in a similar manner to equation 3.42.

3.2.7 CAT kernel

Cluster-adaptive training (CAT), described in chapter 2, is an adaptive training scheme originally designed to allow generative models to be robustly trained using data from multiple corpora recorded under different environmental conditions. It can be interpreted as a robust adaptation scheme where the means of an utterance-dependent GMM are given by the weighted interpolation of P mean clusters. For a particular Gaussian component m the mean for the adapted GMM, $\boldsymbol{\mu}_m^{(i)}$ is defined by

$$\boldsymbol{\mu}_m^{(i)} = \boldsymbol{M}_m\boldsymbol{w}_r^{(i)} \tag{3.45}$$

(3.46)

where $\boldsymbol{M}_m$ is a matrix of P cluster mean vectors associated with component m

$$\boldsymbol{M}_m = [\boldsymbol{\mu}_{m1} \ldots \boldsymbol{\mu}_{mP}] \tag{3.47}$$

and $\boldsymbol{\mu}_{mp}$ is the mean of component m of cluster p. $\boldsymbol{w}_r^{(i)}$ is the vector of cluster weights associated with utterance i and regression class r.

$$\boldsymbol{w}_r^{(i)} = \begin{bmatrix} w_{r1}^{(i)} \\ \vdots \\ w_{rP}^{(i)} \end{bmatrix} \tag{3.48}$$

Given a trained set of cluster means, $\boldsymbol{w}_r^{(i)}$ may be trained for each regression class using the EM update rules described in chapter 2. The CAT kernel [211] is closely related to the (C)MLLR kernels described in the previous section. However rather than obtaining a set of utterance dependent features from an (C)MLLR transform, here the cluster weights are used as features. The feature space associated with the CAT kernel has the form.

$$\boldsymbol{\phi}^{\texttt{CAT}}(\boldsymbol{O}_i) = \begin{bmatrix} \boldsymbol{w}_1^{(i)} \\ \vdots \\ \boldsymbol{w}_R^{(i)} \end{bmatrix} \tag{3.49}$$

Like the MLLR kernel, the CAT kernel yields an extremely compact set of dynamic features. The performance of a classifier based on the CAT kernel will therefore be heavily dependent on the nature of the background CAT model.

3.3 Distributional kernels

Distributional kernels are a family of dynamic kernels that operate directly on probabilistic distributions. For two distributions f_i and f_j defined over the same domain $\boldsymbol{o}$, a distributional kernel $K(f_i, f_j)$ implicitly defines an inner product between f_i and f_j. Although these kernels cannot handle sequence data directly, they are closely related to the sequence kernels described in sections 3.1 and 3.2 and may, through use of an appropriate scheme, be applied to the classification of sequences containing either discrete or continuous observations. One approach for applying distributional kernels to the classification of variable-length sequences is to train a distinct distribution to model each sequence in the dataset. For each utterance, $\boldsymbol{O}_i$, the parameters $\boldsymbol{\lambda}^{(i)}$ of distribution f_i are normally selected to maximise the likelihood of $\boldsymbol{O}_i$. Thus

$$\boldsymbol{\lambda}^{(i)} = \underset{\boldsymbol{\lambda}}{\text{argmax}}\{\log p(\boldsymbol{O}_i; \boldsymbol{\lambda})\} \tag{3.50}$$

Alternatively, the adaptation schemes described in chapter 2, such as MAP or MLLR or discriminative training criteria such as MMI, may be applied to obtain an appropriate set of distributions. Although most of the kernels described in this section may be applied to either discrete or continuous distributions, this section primarily considers their application to classification of speech sequences. Thus, the forms of distribution considered in the section, such as Gaussians and GMMs, are appropriate for modelling speech data. To emphasize that the distributional kernels are being applied to sequence classification, the notation $K(\boldsymbol{O}_i, \boldsymbol{O}_j)$

is used in this section. Distributional kernels are typically motivated from commonly used measures of similarity between distributions. In this section, distributional kernels are introduced based on both the Kullback-Leibler divergence [115] and the Bhattacharyya affinity measure [15].

3.3.1 GMM-supervector kernel

The Kullback-Leibler (KL) divergence [115] defines the relative entropy between two distributions. For distributions f_i and f_j over $\boldsymbol{o}$, the divergence, $KL(f_i||f_j)$, is defined by

$$KL(f_i||f_j) = \int f_i(\boldsymbol{o}) \log \frac{f_i(\boldsymbol{o})}{f_j(\boldsymbol{o})} d\boldsymbol{o} \tag{3.51}$$

The KL divergence has a number of useful properties. The divergence between two distributions f_i and f_j is equal to zero if and only if $f_i = f_j$. Otherwise the divergence is positive. Note that the KL divergence is not symmetric, $KL(f_i||f_j) \neq KL(f_j||f_i)$.

For Gaussian distributions a closed form solution to the KL divergence exists. For Gaussians $\tilde{f}_i(\boldsymbol{o}) = \mathcal{N}(\boldsymbol{o}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ and $\tilde{f}_j(\boldsymbol{o}) = \mathcal{N}(\boldsymbol{o}; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$ the KL divergence can be expressed as

$$\begin{aligned} KL(\tilde{f}_i||\tilde{f}_j) &= \frac{1}{2}\big[\log \frac{|\boldsymbol{\Sigma}_j|}{|\boldsymbol{\Sigma}_i|} + \mathtt{Tr}[\boldsymbol{\Sigma}_j{}^{-1}\boldsymbol{\Sigma}_i] - D \\ &\quad +(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)^{\mathsf{T}}\boldsymbol{\Sigma}_j{}^{-1}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)\big] \end{aligned} \tag{3.52}$$

where $\mathtt{Tr}[\cdot]$ defines the trace of a matrix and D is the dimensionality of $\boldsymbol{o}$. For GMM distributions $f_i(\boldsymbol{o}) = \sum_{n=1}^{N} c_{in}\mathcal{N}(\boldsymbol{o}; \boldsymbol{\mu}_{in}, \boldsymbol{\Sigma}_{in})$ and $f_j(\boldsymbol{o}) = \sum_{m=1}^{M} c_{jm}\mathcal{N}(\boldsymbol{o}; \boldsymbol{\mu}_{jm}, \boldsymbol{\Sigma}_{jm})$ there exists no closed form solution. Instead the divergence must be approximated. This may be achieved either by using sampling approaches or by finding a closed form expression that approximates the true KL divergence. Here the latter approach is examined. When $N = M$, a commonly used upper bound to the KL divergence is the *matched-pair* bound [53]. This is defined by

$$KL^{\mathtt{MP}}(f_i||f_j) = \sum_{m=1}^{M} c_{im}\left[\log \frac{c_{im}}{c_{jm}} + KL(f_{im}||f_{jm})\right] \tag{3.53}$$

where $KL(f_{in}||f_{jm})$ indicates the divergence between component n of f_i and component m of f_j. Permuting the components of one distribution will affect the obtained bound. Hence the matched-pair bound is only suitable when there is a clearly defined coordination between pairs of components from the two distributions. This may be the case when both are adapted from the same background distribution but will not be true generally. In chapter 5, related forms of kernel are proposed that do not suffer from this limitation.

For the GMM-supervector kernel [28], f_i and f_j are constrained to be GMMs that differ only in the means. Under these conditions, the matched-pair bound, defined in equation 3.53, may be used. Since the KL divergence is asymmetric, the symmetric KL divergence, $KL(f_i||f_j) + KL(f_j||f_i)$, is often used instead. This equals zero if and only if $f_i = f_j$, otherwise it is positive. These properties are typical of a distance metric rather than an inner product and the symmetric KL divergence does not yield a positive semi-definite Gram matrix. Thus, it is common to alter the function prior to use such that it behaves more like an inner product. One approach is to make use of the polarisation identity, which defines a

relationship between a distance and a kernel function in a particular space. For a distance $D(f_i, f_j)$ defined between distributions this identity has the following form.

$$D(f_i, f_j)^2 = K(\boldsymbol{O}_i, \boldsymbol{O}_i) - 2K(\boldsymbol{O}_i, \boldsymbol{O}_j) + K(\boldsymbol{O}_j, \boldsymbol{O}_j) \tag{3.54}$$

For the GMM-supervector kernel, the distance between f_i and f_j is defined by

$$D(f_i, f_j)^2 = KL^{\text{MP}}(f_i||f_j) + KL^{\text{MP}}(f_j||f_i) \tag{3.55}$$

This distance is related, via the polarisation identity, to the standard GMM-supervector [28] kernel function.

$$K^{\text{GMM-SV}}(\boldsymbol{O}_i, \boldsymbol{O}_j) = \sum_{m=1}^{M} c_m \boldsymbol{\mu}_{im}{}^{\text{T}} \boldsymbol{\Sigma}_m^{-1} \boldsymbol{\mu}_{jm} \tag{3.56}$$

The GMM-supervector kernel features are closely related to those associated with the Fisher kernel. The relationship between these two forms of kernel is discussed in more detail in chapter 6.

3.3.2 Non-linear GMM-supervector kernel

For the GMM-supervector kernel, a suitable kernel function was derived from an approximation to the KL divergence using the polarisation identity. An alternative procedure for obtaining a suitable kernel function from a distance metric is exponentiation [181]. Here a suitable kernel function is obtained by taking the exponent of the negative symmetric KL divergence.

$$K(\boldsymbol{O}_i, \boldsymbol{O}_j) = \exp\left(-\alpha \left[KL(f_i||f_j) + KL(f_j||f_i)\right]\right) \tag{3.57}$$

where α is a constant scaling term, require for numerical stability. When $KL(f_i||f_j)$ can be evaluated explicitly, for example when f_i and f_j are Gaussians, equation 3.57 may be used directly. The KL kernel, used in [150] has this form. Alternatively, the true KL divergence may be approximated. The non-linear GMM-supervector kernel [45], like the GMM-supervector kernel described earlier, operates on GMMs where all distributions are constrained to be adapted from the same background distribution f and only the mean parameters differ. Under these conditions the matched-pair bound to the KL divergence may be used.

$$KL^{\text{MP}}(f_i||f_j) = \sum_{m=1}^{M} c_{im} \left[\log \frac{c_{im}}{c_{jm}} + KL(f_{im}||f_{jm})\right] \tag{3.58}$$

This yields the following kernel function.

$$K^{\text{MP}}(\boldsymbol{O}_i, \boldsymbol{O}_j) = \exp\left(-\alpha \sum_{m=1}^{M} c_m (\boldsymbol{\mu}_{im} - \boldsymbol{\mu}_{jm})^{\text{T}} \boldsymbol{\Sigma}_m{}^{-1} (\boldsymbol{\mu}_{im} - \boldsymbol{\mu}_{jm})\right) \tag{3.59}$$

Equation 3.59 has the form of the non-linear GMM-supervector kernel used in [45]. Polarisation and exponentiation are closely related. When $\alpha = 1/2\sigma^2$ equation 3.59 is equivalent

to applying an RBF kernel to standard GMM-supervector features. However, unlike the GMM-supervector kernel, $K^{\text{MP}}(\boldsymbol{O}_i, \boldsymbol{O}_j)$ does not have an explicit associated feature space.

One issue with using the non-linear GMM-supervector kernel in practice is that due to the non-linear feature space, the non-linear kernel is known to perform poorly when the KL divergence between distributions varies greatly within the dataset. In [48] it was suggested that this issue can be avoided by normalising all distributions f_i such that the divergence between f_i and the background distribution f is equal to one. When the KL divergence is approximated using the matched-pair bound this may be implemented by normalising the mean parameters associated with each GMM distribution using equation 3.60

$$\boldsymbol{\mu}_{im}^{\text{mnorm}} = \frac{1}{KL^{\text{MP}}(f_i||f)}\boldsymbol{\mu}_{im} + \left(1 - \frac{1}{KL^{\text{MP}}(f_i||f)}\right)\boldsymbol{\mu}_m \tag{3.60}$$

3.3.3 Bhattacharyya kernel

An alternative measure of similarity between two distributions is the Bhattacharyya affinity measure [15, 98]. For two distributions f_i and f_j defined over $\boldsymbol{o}$, this is defined as

$$\mathcal{B}(f_i||f_j) = \int \sqrt{f_i(\boldsymbol{o})}\sqrt{f_j(\boldsymbol{o})}\ d\boldsymbol{o} \tag{3.61}$$

The Bhattacharya affinity measure, which is closely related to Hellinger's distance, forms a bound on the KL divergence between distributions [196]. Unlike the KL divergence, equation 3.61 is symmetric and may be shown to be positive semi-definite [112]. Additionally, $0 \leq \mathcal{B}(f_i||f_j) \leq 1$ for all f_i and f_j and $\mathcal{B}(f_i||f_i) = 1$ for all f_i. The Bhattacharyya measure is therefore particularly suited for use as a kernel function and has been applied successfully in applications such as text-categorisation [92] and speaker recognition [32]. For Gaussian distributions, $f_i(\boldsymbol{o}) = \mathcal{N}(\boldsymbol{o}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ and $f_j(\boldsymbol{o}) = \mathcal{N}(\boldsymbol{o}; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$, the Bhattacharyya affinity measure may be evaluated explicitly. In this case, the Bhattacharyya kernel has the form

$$K^{\text{bhatta}}(\boldsymbol{O}_i, \boldsymbol{O}_j) = \sqrt{\frac{|\tilde{\boldsymbol{\Sigma}}|}{\sqrt{|\boldsymbol{\Sigma}_i||\boldsymbol{\Sigma}_j|}}} \exp\left[-\frac{1}{4}\left(\boldsymbol{\mu}_i^{\text{T}}\boldsymbol{\Sigma}_i\boldsymbol{\mu}_i + \boldsymbol{\mu}_j^{\text{T}}\boldsymbol{\Sigma}_j\boldsymbol{\mu}_j - 2\tilde{\boldsymbol{\mu}}^{\text{T}}\tilde{\boldsymbol{\Sigma}}\tilde{\boldsymbol{\mu}}\right)\right] \tag{3.62}$$

where $\tilde{\boldsymbol{\Sigma}} = 2(\boldsymbol{\Sigma}_i^{-1} + \boldsymbol{\Sigma}_j^{-1})^{-1}$ and $\tilde{\boldsymbol{\mu}} = \frac{1}{2}(\boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i + \boldsymbol{\Sigma}_j^{-1}\boldsymbol{\mu}_j)$. For more complex forms of distribution, such as when f_i and f_j are GMMs, the Bhattacharyya affinity measure is not guaranteed to have a closed form solution. Instead sampling approaches, such as the scheme used in [92], may be used to estimate equation 3.61.

As with the Kullback-Leibler divergence, for families of distributions where it is not possible to evaluate equation 3.61 directly, alternative forms of kernels may be derived by approximating the true Bhattacharyya affinity. The GMM-UBM mean interval (GUMI) kernel [213, 214] is an example of this approach. For the distributional kernels described previously in this section, the kernel function $K(\boldsymbol{O}_i, \boldsymbol{O}_j)$ is motivated directly from a statistical measure between f_i and f_j. The GUMI kernel operates in a different manner. Here for each utterance $\boldsymbol{O}_i$, with associated distribution f_i, the feature vector $\boldsymbol{\phi}(\boldsymbol{O}_i; f)$ is derived from an approximation to the Bhattacharyya affinity between f_i and a general background distribution f representing all the data[1]. For M-component GMM distributions

[1] The Bhattacharyya affinity is defined slightly differently in the derivation of the Bhattacharyya and GUMI kernels. Equation 3.63 is derived from an affinity measure defined as $\mathcal{B}^{\text{GUMI}} = -\log \mathcal{B}$

$f_i(\boldsymbol{o}) = \sum_{m=1}^{M} c_{jm}\mathcal{N}(\boldsymbol{o}; \boldsymbol{\mu}_{im}, \boldsymbol{\Sigma}_{im})$ and background distribution $f = \sum_{m=1}^{M} c_m\mathcal{N}(\boldsymbol{o}; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)$ the GUMI feature space associated with component m has the following form.

$$\phi_m^{\texttt{GUMI}}(\boldsymbol{O}_i; f) = \left[\frac{\boldsymbol{\Sigma}_{im} + \boldsymbol{\Sigma}_m}{2}\right]^{-\frac{1}{2}} (\boldsymbol{\mu}_{im} - \boldsymbol{\mu}_m) \tag{3.63}$$

Equation 3.63 is closely related to the GMM-supervector feature space defined in equation 3.56. Unlike the GMM-supervector the GUMI features also includes a sequence-dependent covariance term. When the covariances are tied over all distributions, a maximally non-committal metric is applied and the background distribution f has zero mean, the GUMI and GMM-supervector features will be identical.

3.4 Summary

This chapter has examined three different classes of dynamic kernels that may be applied to sequence data. The first two classes of dynamic kernel were discrete and continuous observation kernels, which operate directly on sequences of discrete and continuous data respectively. The final form of dynamic kernel examined in this chapter were distributional kernels. These implicitly calculate the inner product between probabilistic distributions in some potentially high-dimensional space. By training a distribution to represent each sequence in a dataset, distributional kernels may also be applied to classify sequences of continuous or discrete data, including speech utterances. The next chapter discusses how these dynamic kernels may be combined with the classifiers described in chapter 2 to build a speaker verification system.

Chapter 4
Speaker Verification

Speaker verification (SV) is a binary classification task in which the objective is to determine, given an utterance of speech and an associated identity claim, whether or not the utterance was spoken by the specific claimed speaker. This chapter describes how the theory introduced in chapters 2 and 3 may be applied to construct a state-of-the-art system for text-independent speaker verification. Current SV systems are typically based around one of two general frameworks. In the first approach, one generative model is trained to represent a general background population of speakers, and a second to represent the claimed speaker. The speech utterance is then classified, as to whether the claimed identity is correct, using Bayes' decision rule. Recently there has been interest in applying discriminative classifiers such as SVMs to the SV task. Dynamic kernels, described in chapter 3, are used to allow the SVM to handle speech data. Both generative and discriminative approaches are described in this chapter.

An open problem in SV research is how to handle unwanted variability in the speech data. This includes additive or channel noise that distorts the speech signal and makes it harder to distinguish speakers. A related issue is inter-session variability, due to either changing environmental or recording conditions, or the emotional state or age of the speaker. A variety of techniques have been developed to handle these issues. Techniques described in this chapter include frontend techniques such as cepstral mean normalisation [60] and feature-warping [156], model based techniques such as nuisance attribute projection (NAP) variability compensation [186] and factor analysis [109] and score-normalisation techniques such as T-norm [6] and Z-norm [171]. State-of-the art SV systems will typically include multiple forms of noise-compensation.

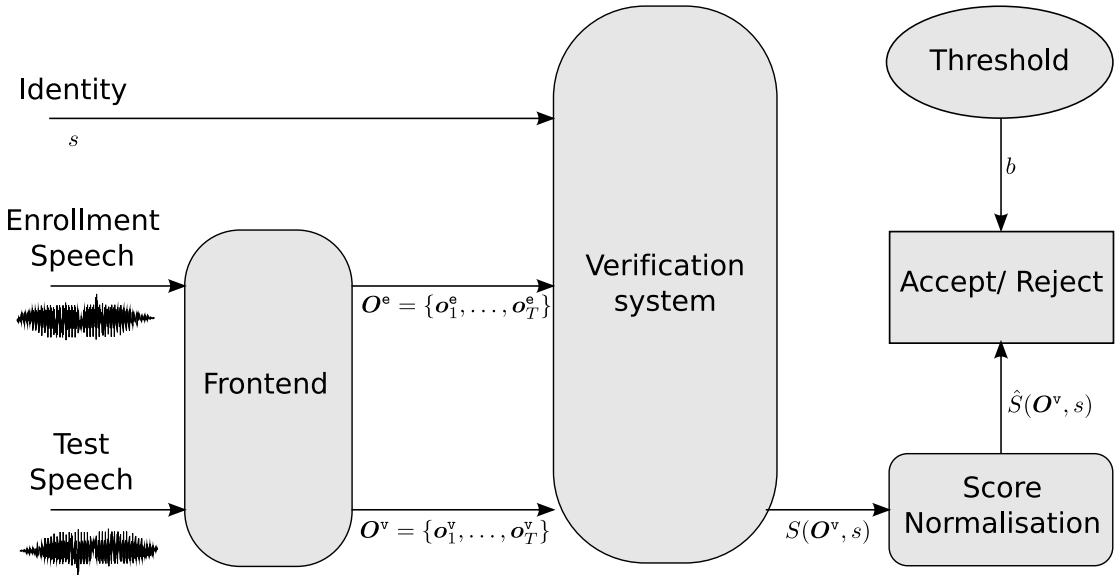
This chapter is organised as follows: the next section gives an overview of the components of a general speaker-verification system. In section 4.2, appropriate frontend schemes

for parameterising the data are discussed. Section 4.3 describes a generative model-based framework for SV. A closely related approach, where session and channel variability is modelled using factor analysis is described in section 4.4. Section 4.5 describes how the dynamic kernels introduced in chapter 3 may be applied to construct an SVM-based speaker verification system. Finally, section 4.6 discusses techniques for improving system performance by normalising the output scores and section 4.7 describes metrics that may be used for assessing the performance of a speaker-verification system.

4.1 Overview

The process of speaker verification is normally divided into two distinct phases, enrollment and verification. During enrollment, each user of the system provides a series of utterances $\mathcal{O}^{\mathsf{e}} = \{\boldsymbol{O}^{\mathsf{e}}_1, \ldots, \boldsymbol{O}^{\mathsf{e}}_N\}$. These are then used to train a classifier capable of distinguishing the characteristics of the user's voice from other speakers. During the verification stage, an utterance $\boldsymbol{O}^{\mathsf{v}}$ and an identity s are provided. The system is then required to determine whether the utterance was generated by the target speaker s or by an imposter. Depending on the particular application, test utterances may be guaranteed to have been spoken by one of the enrolled speakers (closed-set verification) or they may be from a previously unseen speaker (open-set verification). SV tasks are also typically divided into two groups, text-dependent and text-independent, depending on whether the speaker is prompted to speak a specific sequence of words. Most current SV research, including the annual NIST speaker recognition evaluations (SREs) [144], is concerned with text-independent speaker verification. Here utterances are classified based purely on their acoustic content, without requiring a known transcription. In this thesis, only text-independent speaker verification is considered. In general, the number of enrollment utterances that each user provides is likely to be limited. Often only a single enrollment utterance per speaker is available. A common requirement for SV systems is that they must be robust to these conditions. Techniques for improving the robustness of SV systems are discussed in more detail in chapter 7.

In this chapter, SV systems are divided into three primary components: the frontend, the core verification system and a score-normalisation component. The relationship between these components is shown in figure 4.1. Although these components are considered separately in this chapter, in the literature the distinction between these components is often blurred, particularly for SVM-based systems. Each enrollment and test utterance is initially processed by a frontend system that converts a speech waveform into a sequence of observations. Next, the utterances are passed to a core verification system where a distinct classifier is trained for each target speaker using the enrollment data. A score, $S(\boldsymbol{O}^{\mathsf{v}}, s)$, can then be assigned to each test utterance using the classifier associated with the claimed speaker identity. Classifiers used in SV systems are usually based on either generative models, such as GMMs, or discriminative techniques, such as SVMs. Both approaches are described in this chapter. Finally each score is post-processed to normalise unwanted session and environmental variability and map the scores from each classifier to a consistent range. The output of the SV system is a normalised score $\hat{S}(\boldsymbol{O}, s)$. A accept/reject design can be made by comparing this score to a fixed threshold.



Enrollment $\boldsymbol{O}^{\mathrm{e}}$ and test $\boldsymbol{O}^{\mathrm{v}}$ utterances are initially processed using a frontend to parameterise each utterance into a sequence of observations. The parameterised utterances are then passed to the core verification system, where a distinct classifier is then trained for each target speaker using the available enrollment data. These classifiers are then used to assign a score $S(\boldsymbol{O}^{\mathrm{v}}, s)$ to each test utterance given the target speaker identity s. Test scores are normalised to reduce unwanted variation and compared to a fixed threshold to obtain a classification decision.

Figure 4.1: Overview of a text-independent speaker verification system.

$$\hat{S}(\boldsymbol{O}^{\mathtt{v}}, s) - b \begin{array}{c} \text{accept} \\ > \\ < \\ \text{reject} \end{array} 0 \tag{4.1}$$

where b is a threshold, set by the user. A speaker verification system can make two distinct types of error: miss (false rejection) and false alarm (false acceptance). For many situations the penalties associated with each form of error will not be equal. By adjusting the threshold b, the operator can compensate. A low threshold will reduce the number of misses at a cost of increasing the number of false-alarms. Setting a high threshold will give the opposite behaviour. If $\hat{S}(\boldsymbol{O}^{\mathtt{v}}, s)$ approximates a log-likelihood ratio and a known cost $(C^{\mathtt{false-alarm}}, C^{\mathtt{miss}})$ is associated with each form of error, the optimal trade-off can be determined and the threshold may be explicitly set using the Bayes' detection criterion to minimise the total cost.

$$b^{\mathtt{bayes}} = \log \frac{(1 - P(\omega^{\mathtt{tar}}))C^{\mathtt{false-alarm}}}{P(\omega^{\mathtt{tar}})C^{\mathtt{miss}}} \tag{4.2}$$

where $P(\omega^{\mathtt{tar}})$ is the prior probability that $\boldsymbol{O}^{\mathtt{v}}$ was spoken by the target identity. When the objective is to compare performance of multiple SV systems it is not necessary to explicitly set a threshold. Threshold-independent evaluation metrics are discussed in detail in section 4.7.

4.2 Frontend processing

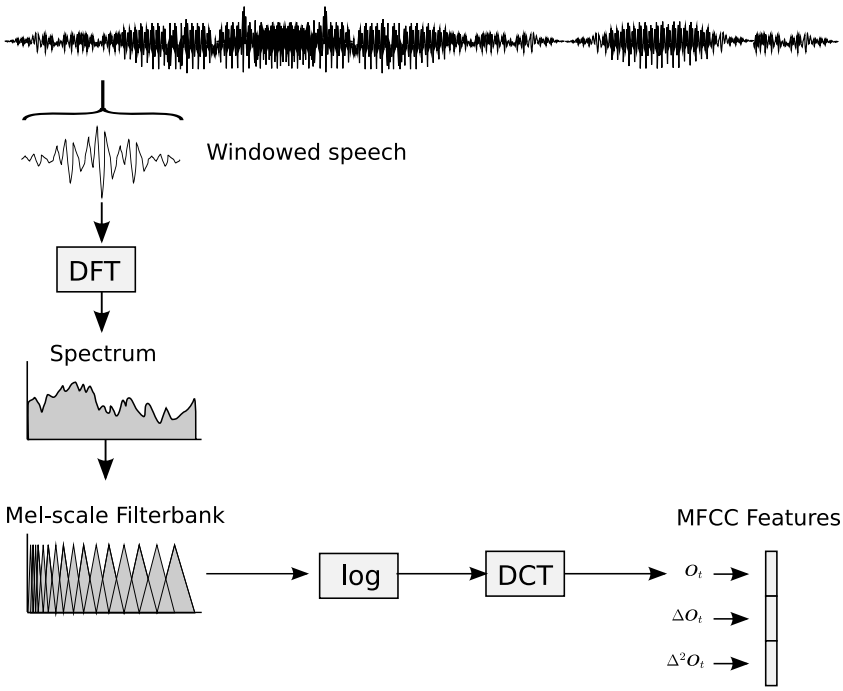
Frontend processing is the first stage in constructing a speaker verification system. The objective is to convert a speech waveform into a compact representation that retains all the information useful for discrimination between speakers and discards non-useful information. Frontend precessing schemes for SV have traditionally been adapted from schemes designed for ASR. These include mel-frequency cepstral coefficients (MFCCs) [43] and perceptual linear prediction (PLP) coefficients [86]. These are based on short-term spectral features of the waveform. Since one of the objectives in ASR frontend processing is to minimise interspeaker variation, these features may not be optimal. However, they have been found to perform well in practice [55].

Recently, there has been interest in designing systems based on long-term prosodic and language features. These are typically obtained using a two stage process. First short-term spectral features are obtained from the waveform. These are then processed using an ASR system to obtain longer-term features. Both short and long term feature extraction is described in this section.

4.2.1 Short term feature extraction

The dominant parameterisation for SV systems is based on short term spectral features. Here, each utterance is initially partitioned into a series of overlapping frames, each comprised of K samples $\{x_1, \ldots, x_K\}$, and a single observation vector $\boldsymbol{o}_t$ is extracted from each frame. For SV, windows sizes of 30ms duration and frame shifts of 10ms are commonly used [12]. One set

of spectral features that are commonly used for both ASR and SV are mel-frequency cepstral coefficients (MFCCs) [43]. Figure 4.2 shows the extraction process required to obtain MFCCs from the speech waveform. Within each speech frame, the waveform is stationary and a



Short-term spectral features are obtained from the waveform by computing a discrete Fourier transform of the speech signal contained within a 30ms sliding window. Filterbank coefficients are extracted using triangular bandpass filters spaced according to the mel-scale. A discrete cosine transform is then applied and dynamic terms appended to obtain the MFCC feature vector.

Figure 4.2: MFCC feature extraction.

discrete Fourier transform (DFT) is applied to obtain the short term spectrum. Alternatively, when the number of samples contained in each frame is a power of two, an efficient Fast Fourier Transform (FFT) may be computed instead. To avoid the introduction of unwanted high frequencies caused by discontinuities at the edge of the window, a hamming window is typically applied before computing the spectrum. In this case, each frame may also be optionally padded with zero-valued samples to allow the use of a FFT. This process is known as *zero-padding*. Finally, a pre-emphasis filter is often applied to the short term spectrum to boost the energy at higher frequencies.

The spectral envelope is sampled using triangular bandpass-filters to obtain a series of R filterbank coefficients $\{f_1, \ldots, f_R\}$. The center of the filters are spaced according to the mel

scale

$$f^{\texttt{mel}} = (1127)\log(1 + \frac{f^{\texttt{Hz}}}{700}) \tag{4.3}$$

where $f^{\texttt{mel}}$ is the mel frequency and $f^{\texttt{Hz}}$ is the linear frequency. The mel-scale is proportional to the log of the linear frequency and reflects changes in the pitch of the signal as perceived by the human auditory system. The filterbank coefficients are then converted into MFCC features by taking a discrete cosine transform of the log-spectral amplitudes. The MFCC features $o_{dt}^{\texttt{MFCC}}$ are given by

$$o_{dt}^{\texttt{MFCC}} = \sqrt{\frac{2}{R}} \sum_{i=1}^{R} \log(f_{it}) \cos(\frac{\pi d}{R}(i - 0.5)) \tag{4.4}$$

where f_{it} is the amplitude of filterbank i at frame t. The lowest cepstral feature $o_{0t}^{\texttt{MFCC}}$ measures the average log-energy. MFCCs were originally developed for use in ASR, where it has become standard to extract 13 MFCC features from each window. For SV this may not be optimal [12]. Current state-of-the-art SV system use a range of parameterisations [22, 102].

An alternative cepstral parameterisation may be obtained via linear prediction (LP) analysis of the waveform. In the LP model, the amplitude of a sample x_k is predicted using the amplitudes of the previous P samples. The predicted value of sample x_k is given by

$$\tilde{x}_k = \sum_{p=1}^{P} a_p x_{k-p} \tag{4.5}$$

where $\boldsymbol{a} = \{a_1, \ldots, a_P\}$ are the LP coefficients. In LP analysis, the vocal tract is modelled as an all-pole filter. The LP coefficients represent the impulse response of the vocal tract. The difference between the actual samples x_k and the prediction $\tilde{x}_k$ models the source excitation. Speech is quasi-stationary since the vocal tract apparatus takes a finite time to change position, therefore the LP coefficients may be assumed to be constant within a particular window of K samples extracted from the waveform at time t. The optimal LP coefficients $\boldsymbol{a}_t^*$ associated with these speech samples are those that minimise the square of the error over all samples within the window.

$$\boldsymbol{a}_t^* = \underset{\boldsymbol{a}}{\operatorname{argmin}} \left\{ \frac{1}{K} \sum_{k=1}^{K} (x_k - \tilde{x}_k)^2 \right\} \tag{4.6}$$

Equation 4.6 may be efficiently minimised using algorithms described in [5, 140]. The LP coefficients form a compact representation of speech, however they are typically insufficiently robust to be used as a parameterisation for SV. Instead they can be used to derive linear predictive cepstral coefficients (LPCC) [4, 60]. These can be computed efficiently using the recursion $o_{t1}^{\texttt{LPCC}} = a_1^*$ and

$$o_{td}^{\texttt{LPCC}} = a_d^* + \sum_{j=1}^{d-1} \left(\frac{d-j}{d} \right) o_{t(d-j)}^{\texttt{LPCC}} a_j^* \quad 1 < d \leq P \tag{4.7}$$

$$o_{td}^{\texttt{LPCC}} = \sum_{j=1}^{d-1} \left(\frac{d-j}{d} \right) o_{t(d-j)}^{\texttt{LPCC}} a_j^* \qquad d > P \tag{4.8}$$

Unlike ASR tasks, where they perform poorly [220], there is some evidence [141] to suggest that LPCC coefficients can outperform MFCCs on SV tasks.

A closely related form of parameterisation that is commonly used in both ASR and SV are perceptual linear predictive (PLP) coefficients [86]. These are obtained in a similar manner to LPCC coefficients, except that the spectrum at each time instance is initially modified by a series of psycho-acoustically motivated transforms. These are intended to generate a set of coefficients that more accurately reflect perceived changes in the speech signal and apply a greater weight to perceptually important parts of the frequency range. The short term power-spectrum is initially convolved with a series of critical band filters. These are spaced according to the Bark scale [221] and correspond to roughly equal distances along the basilar membrane in the inner ear, which vibrates in response to incoming sound.

$$f^{\texttt{bark}} = 6 \log \left[\sqrt{1 + \left(\frac{f^{\texttt{Hz}}}{1200\pi} \right)^2} + \frac{f^{\texttt{Hz}}}{1200\pi} \right] \tag{4.9}$$

An equal loudness pre-emphasis filter is applied to normalise the relative perceived loudness of each frequency. The amplitude of all frequencies is then warped using a cubed root so that changes in intensity are proportional to perceived changes in loudness. Finally, an inverse DCT is performed to transform the speech back into the time domain. PLP coefficients are then obtained by performing an LP analysis and converting the LP coefficients to cepstral coefficients using equation 4.7. Mel-PLP coefficients are a related parameterisation that have also been used for SV [12]. Here, instead of using the Bark scale the coefficients are derived using the mel scale defined by equation 4.3.

Speech signals are known to be strongly correlated over time. Since standard forms of generative model, such as HMMs and GMMs do not explicitly model temporal correlation (for HMMs, observations are assumed to be conditionally independent given the current state) it is common to append additional coefficients to the static MFCC/LPCC/PLP feature vector to model the changing speech signal at each time instance. Often, first (delta Δ) and second (acceleration Δ^2) order derivatives of the static coefficients are appended. For computational efficiency, delta coefficients are typically approximated using linear regression.

$$\Delta \boldsymbol{o}_t = \frac{\sum_{\delta=1}^{\Delta} \delta (\boldsymbol{o}_{t+\delta}^{\texttt{static}} - \boldsymbol{o}_{t-\delta}^{\texttt{static}})}{2 \sum_{\delta=1}^{\Delta} \delta^2} \tag{4.10}$$

where $\boldsymbol{o}_t^{\texttt{static}}$ is a vector of MFCC/LPCC/PLP features extracted from frame t. When $\Delta = 1$ the delta coefficients are simply the difference between the previous and following frame. The use of larger values of Δ will lead to more robust estimates of the dynamic features. Acceleration coefficients may be calculated from the delta coefficients in a similar manner using equation 4.10. However, unlike ASR where they improve classification accuracy, MFCC and PLP acceleration coefficients have been found to contain little useful speaker-discriminative information [55]. Static, delta and (optionally) acceleration coefficients are concatenated to obtain the final feature vector $\boldsymbol{o}_t$ at time t.

$$\boldsymbol{o}_t = \begin{bmatrix} \boldsymbol{o}_t^{\texttt{static}} \\ \Delta \boldsymbol{o}_t \\ \Delta^2 \boldsymbol{o}_t \end{bmatrix} \tag{4.11}$$

4.2.2 Short term feature normalisation

Short term cepstral features obtained using the parameterisation schemes in section 4.2.1 are often distorted by additive and convolutional noise. This typically introduces additional dependencies between the elements of the feature vector [127], which can not effectively be modelled using diagonal covariance matrices. Three normalisation techniques are discussed in this section that attempt to reduce the effect of noise on the features. These are cepstral mean and variance normalisation [60], cepstral feature warping [156] and principal component analysis [17].

4.2.2.1 Cepstral mean and variance normalisation

In the cepstral domain, convolutional noise is additive. Cepstral mean normalisation (CMN) [60], also known as cepstral mean subtraction, is a noise-compensation approach that operates by normalising the mean of the cepstral features to zero, effectively removing fixed convolutional noise. The CMN normalised feature vector is given by

$$\boldsymbol{o}_t^{CMN} = \boldsymbol{o}_t - \frac{1}{T}\sum_{\tau=1}^{T}\boldsymbol{o}_\tau \tag{4.12}$$

CMN is known to remove speaker-specific information about the vocal tract length [156]. Under clean conditions, applying CMN has been found to degrade SV performance [170]. However under different channel conditions applying CMN can lead to gains [12]. A related form of normalisation is cepstral variance normalisation (CVN). Here the variance of each cepstral feature is normalised to one. Since the variance is estimated independently for each cepstral feature, CVN cannot be used to decorrelate the feature vector. If decorrelation is required, then a PCA transform, discussed in section 4.2.2.3, may be applied instead.

4.2.2.2 Cepstral feature warping

Cepstral mean normalisation only removes convolutional effects from the speech signal. When additive noise is present then CMN will not necessarily improve SV performance. Cepstral feature warping [156] is a form of short-term gaussianisation developed for SV that can remove both additive and convolutional noise. The objective of feature warping is to apply a (non-linear) transformation to the features such that the transformed observations $\boldsymbol{O}^{\mathtt{FW}} = \{\boldsymbol{o}_1^{\mathtt{FW}}, \ldots, \boldsymbol{o}_N^{\mathtt{FW}}\}$ will be normally distributed.

$$\boldsymbol{o}^{\mathtt{FW}} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \tag{4.13}$$

Unlike CMN and CVM, which normalise the first and second order moments of the data respectively, cepstral feature warping also normalises higher order moments.

Cepstral feature warping operates by normalising each observation relative to its neighbours in the sequence. To simplify the transformation, each dimension is normalised independently. Hence, the transformed features will only be approximately Gaussian. A fixed-width window of N observations is moved along the utterance, one observation at a time. At each interval, the features contained within the window are sorted by value into descending order and the position R_{dt} of the central feature o_{dt} within the sorted list is determined. The feature value is then warped such that the cumulative density to o_{dt} in the observed distribution is

equal to the cumulative density to the transformed feature within a normal distribution. The value of the transformed feature $o_{dt}^{\texttt{FW}}$ is given by

$$\frac{N + \frac{1}{2} - R_{dt}}{N} = \int_{z=-\infty}^{o_{dt}^{\texttt{FW}}} \frac{1}{\sqrt{2\pi}} \exp\left(\frac{-z^2}{2}\right) \, dz \tag{4.14}$$

For a fixed window size, the transformation may be efficiently implemented using a lookup table [156]. Typically windows of three seconds duration are used. Since cepstral feature warping normalises the global mean and variance of the cepstral features, it is not necessary to perform additional CMN or CVN normalisation. A closely related form of short-term Gaussianisation was used in [34]. Unlike cepstral feature warping, which is a histogram-based scheme, the scheme proposed in [34] uses GMMs to model the observed distribution of the features. Thus, the user is required to select an appropriate model complexity. This is not necessary for cepstral feature warping.

4.2.2.3 Principal component analysis

Cepstral mean normalisation and cepstral feature warping are techniques that can be applied to independently normalise the mean and variance of each observation. When the speech signal contains additive noise, the cepstral features will often be highly correlated [127]. One technique that can be used to decorrelate the feature vector is principal component analysis (PCA) [17]. PCA is an unsupervised technique that applies a linear transform to each observation.

$$\boldsymbol{o}_t^{\texttt{PCA}} = \boldsymbol{A}^{\texttt{PCA}} \boldsymbol{o}_t \tag{4.15}$$

The PCA transform $\boldsymbol{A}^{\texttt{PCA}}$ is obtained by performing an eigen-decomposition $\boldsymbol{\Sigma}_{\boldsymbol{o}} = \boldsymbol{A}^{\texttt{PCA}^{\texttt{T}}} \Lambda \boldsymbol{A}^{\texttt{PCA}}$, where $\boldsymbol{\Sigma}_{\boldsymbol{o}}$ is the covariance of the features. PCA may also be used as a feature-selection technique, by retaining the transformed features that correspond the the largest eigenvalues. This assumes that dimensions with the largest variance are most important, PCA is therefore not robust to scalings of individual dimensions of the observation vector.

4.2.3 Long term feature extraction

A recent trend in SV research has been the use of parameterisations based on long-term features. Unlike parameterisations such as PLP or MFCC coefficients that represent the spectral distribution over a 30ms window, long term features are calculated over an interval that can span multiple seconds and incorporate prosodic or language information associated with the speaker. When a suitable speech recogniser is available, a simple parameterisation that may be applied is to tokenise each utterance into a sequence of T words [54]. Here each utterance has the form

$$\boldsymbol{O} = [w_1, \ldots, w_T] \tag{4.16}$$

where w_t is the tth word in the sequence. Alternatively, $\boldsymbol{O}$ may be tokenised at the phone level [2, 3].

A more complex form of parameterisation that has been applied successfully to SV are SNERFs (syllable-based nonuniform extraction region features) [182]. SNERFs are obtained

by processing the output of a speech recogniser using a set of handcrafted rules to tokenise the speech utterance into a sequence of T syllables.

$$\boldsymbol{O} = [c_1, \ldots, c_T] \tag{4.17}$$

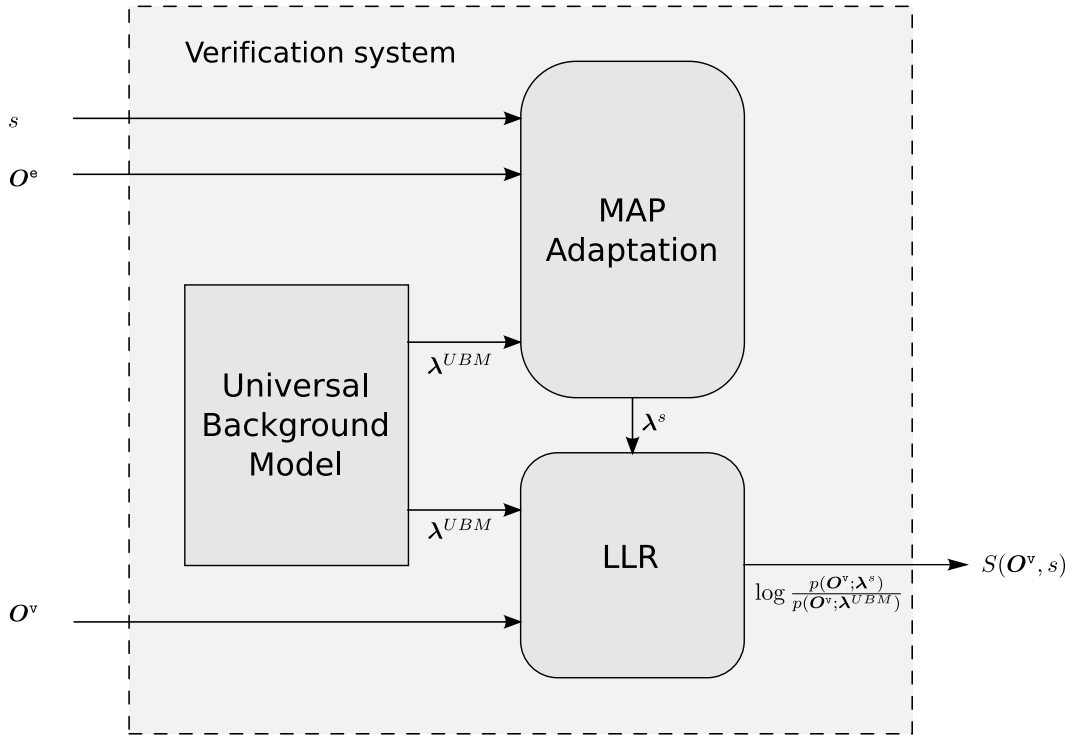
In the absence of an ASR system, pseudo-syllable regions may be identified as the regions between consecutive local minima in the energy signal over voiced regions [102]. For each syllable region c_t a phone alignment is obtained from the speech recogniser and used to extract a variety of features related to the duration, pitch and energy of the speech segment. A vector of duration features $\boldsymbol{d}_t$ is extracted from c_t using the alignments of the speech recogniser. Distinct duration features are extracted from different regions of the syllable: onset, nucleus, coda, onset+nucleus, nucleus + coda and the full syllable. Two vectors of pitch features, $\boldsymbol{p}_t^{\mathtt{v}}$ and $\boldsymbol{p}_t^{\mathtt{u}}$ are extracted from voiced and unvoiced regions of the syllable respectively. Initially, the fundamental frequency f_0 is calculated for each sample. Then for each region, the following pitch features are extracted: the maximum pitch, minimum pitch, mean pitch, the difference between the maximum and minimum pitch, the number of frames that are rising/falling/doubled/halved/voiced, the length of the first/last slope, the number of changes from falling to rising, the value of first/last average slope and the maximum positive/negative slope. Vectors of energy features are calculated from four different regions: the nucleus $\boldsymbol{e}_t^{n}$, the nucleus minus unvoiced frames $\boldsymbol{e}_t^{\mathtt{n-u}}$, the whole syllable $\boldsymbol{e}_t^{\mathtt{s}}$ and the whole syllable minus unvoiced frames $\boldsymbol{e}_t^{\mathtt{s-u}}$. Energy features are obtained from each region in a similar manner to pitch features. Finally, a SNERF feature vector $\boldsymbol{o}_t^{\mathtt{SNERF}}$ is obtained from each syllable region c_t by concatenating the respective duration, pitch and energy features.

$$\boldsymbol{o}_t^{\mathtt{SNERF}} = \begin{bmatrix} \boldsymbol{d}_t \\ \boldsymbol{p}_t^{\mathtt{v}} \\ \boldsymbol{p}_t^{\mathtt{u}} \\ \boldsymbol{e}_t^{\mathtt{n}} \\ \boldsymbol{e}_t^{\mathtt{n-u}} \\ \boldsymbol{e}_t^{\mathtt{s}} \\ \boldsymbol{e}_t^{\mathtt{s-u}} \end{bmatrix} \tag{4.18}$$

SNERFs contain a combination of discrete and continuous features. To enable a consistent modelling approach, the continuous elements of the SNERF feature vector are discretised using a series of bins. The bin thresholds are set such that each bin contains roughly equal amounts of data. While short-term parameterisations generally involve sequences of continuous observations, the features associated with long term parameterisations are generally discrete. Generative classification schemes can usually be applied to both continuous and discrete data, given an appropriate generative model. However, for SVM-based classifiers, it is often not possible to use kernels developed for sequences of continuous observations to classify discrete data. This is because many popular forms of dynamic kernel, such as the GMM-supervector [28] and MLLR [189] kernels are explicitly defined in terms of specific continuous variable generative models, such as GMMs and HMMs.

4.3 GMM-based speaker verification

Traditionally, speaker verification systems have made use of generative models. Generative models, introduced in chapter 2, can be used to assign a likelihood to a speech utterance.



A GMM universal background model (UBM) is initially trained using data from many speakers. A set of target speaker-dependent GMMs are then MAP adapted from the UBM using the available target-specific enrollment data $\boldsymbol{O}^{\mathrm{e}}$. For each test utterance $\boldsymbol{O}^{\mathrm{v}}$ and identity claim s the classifier output is computed as the log-likelihood ratio between the target speaker model $\boldsymbol{\lambda}^{(s)}$ and the UBM.

Figure 4.3: Speaker verification using generative models.

The nature of the model is usually determined by an associated set of parameters $\boldsymbol{\lambda}$. Given two models, the first $p(\boldsymbol{O};\boldsymbol{\lambda}^{(s)})$ representing the target speaker s, and the second $p(\boldsymbol{O};\boldsymbol{\lambda}^{I(s)})$ representing all competing speakers, a classification score $S(\boldsymbol{O}^{\mathrm{v}},s)$ is assigned to each test utterance $\boldsymbol{O}^{\mathrm{v}}$ using a log-likelihood ratio [170].

$$S(\boldsymbol{O}^{\mathrm{v}},s) = \log p(\boldsymbol{O}^{\mathrm{v}};\boldsymbol{\lambda}^{(s)}) - \log p(\boldsymbol{O}^{\mathrm{v}};\boldsymbol{\lambda}^{I(s)}) \tag{4.19}$$

The dominant form of generative model used in text-independent SV is the GMM. For efficiency, diagonal covariance matrices are normally used with up to 1000 Gaussian components. GMMs are suitable when utterances are parameterised as sequences of continuous observations. This includes widely used cepstral parameterisations such as MFCCs and PLP coefficients. For discrete parameterisations such SNERFs or word sequences, N-gram models may be used instead [3, 11, 77, 79].

4.3.1 Parameter estimation

The background model $p(\boldsymbol{O}^{\mathrm{v}};\boldsymbol{\lambda}^{I(s)})$ acts to normalise some speaker-independent aspects of $\boldsymbol{O}^{\mathrm{v}}$ and can therefore be interpreted as a form of score-normalisation. Alternative score-normalisation schemes are discussed in section 4.6. In early SV systems, $p(\boldsymbol{O};\boldsymbol{\lambda}^{I(s)})$ was modelled using a cohort approach [174].

$$p^{\mathtt{cohort}}(\boldsymbol{O};\boldsymbol{\lambda}^{I(s)}) = \mathcal{Q}(p(\boldsymbol{O};\boldsymbol{\lambda}^{(1)}),\ldots,p(\boldsymbol{O};\boldsymbol{\lambda}^{(S)})) \tag{4.20}$$

where $\mathcal{Q}(\cdot)$ is a function (i.e. max, average) over all the competing speaker models. This approach has a number of disadvantages. It is only suitable for closed-set identification, where the identities of the potential imposters is known in advance. It is also inefficient, since a unique background model is used for each target speaker. Instead, current GMM-based SV systems typically use universal background models (UBMs). Here a single generative model is trained using data from many speakers. This approach is more efficient, since a single background model can be used to classify all speakers, and can be applied to open-set SV tasks.

$$p(\boldsymbol{O};\boldsymbol{\lambda}^{I(s)}) \quad = \quad p(\boldsymbol{O};\boldsymbol{\lambda}^{UBM}) \tag{4.21}$$

Figure 4.3 outlines the main components of a GMM/UBM-based SV system. Given a large background dataset $\{\boldsymbol{O}^{\mathtt{B}}_1,\ldots,\boldsymbol{O}^{\mathtt{B}}_N\}$, containing utterances from a wide range of speakers, the UBM parameters $\boldsymbol{\lambda}^{UBM}$ are typically trained using a maximum likelihood criterion.

$$\boldsymbol{\lambda}^{UBM} \quad = \quad \underset{\boldsymbol{\lambda}}{\mathrm{argmax}} \left\{ \sum_{i=1}^{N} \log p(\boldsymbol{O}^{\mathtt{B}}_i;\boldsymbol{\lambda}) \right\} \tag{4.22}$$

If the gender of the target speaker is provided, or can be detected from each utterance, gender-dependent UBMs can be used. This approach is most effective when the background utterances are recorded under the same environmental conditions as the enrollment and test data. For applications where the noise-conditions vary between recording sessions, it may be more appropriate to use a range of noise condition-dependent background models. Alternatively, factor analysis-based approaches, described in section 4.4, may be used instead.

The amount of enrollment data provided by each speaker is usually limited and is typically insufficient to robustly estimate the parameters of a speaker-dependent GMM. Instead, robust speaker-dependent models may be obtained by adapting the parameters of the UBM using a small amount of speaker-dependent enrollment data $\boldsymbol{O}^{\mathtt{e}} = \{\boldsymbol{o}_1^{(s)}, \ldots, \boldsymbol{o}_T^{(s)}\}$ [169]. A commonly used technique is MAP, described in chapter 2. When the UBM is used as a robust prior distribution, the adapted speaker-dependent GMM means are given by

$$\boldsymbol{\mu}_m^{(s)} = \frac{\sum_{t=1}^{T} P(\theta_t = m|\boldsymbol{o}_t^{(s)}; \boldsymbol{\lambda}^{UBM})\boldsymbol{o}_t^{(s)} + \tau^{\mathtt{map}}\boldsymbol{\mu}_m^{UBM}}{\sum_{t=1}^{T} P(\theta_t = m|\boldsymbol{o}_t^{(s)}; \boldsymbol{\lambda}^{UBM}) + \tau^{\mathtt{map}}} \tag{4.23}$$

where $P(\theta_t = m|\boldsymbol{o}_t^{(s)}; \boldsymbol{\lambda}^{UBM})$ is the posterior probability of $\boldsymbol{o}_t^{(s)}$ being emitted by UBM component m. The MAP adaptation constant $\tau^{\mathtt{map}}$ controls the trade-off between the prior and posterior parameter estimates. Suitable values for $\tau^{\mathtt{map}}$ are dependent on the amount of available adaptation data and the size of model used. Depending on the amount of available adaptation data, variance and component priors can also be adapted. Multiple iterations of MAP may also be used to obtain a speaker-dependent model. Discriminative adaptation schemes, such as MMI-MAP have been applied to SV [129]. However they have not been found to provide significant gains over the standard MAP approach.

4.4 Factor analysis-based speaker verification

One of the principal causes of performance degradation in SV systems is due to unwanted variation in the speech signal. This variation may be either speaker-dependent, caused by natural variation in a user's speech between recording sessions, or speaker-independent, caused by environment or channel effects. There has been significant interest in attempting to effectively model this variation to improve the performance of SV systems. One approach that has been found to be successful is factor analysis modelling [109, 110]. Here a fixed GMM structure is used, where each component has associated mean and covariance parameters. The value of each component mean is then assumed to be distributed according to a small number of latent factors, each associated with either the current speaker or current session (channel) conditions. All latent factors are assumed to be normally distributed with zero mean and unit variance. Factor analysis has been successfully applied to model both cepstral [111] and continuous prosodic [46] features.

The factor analysis model is defined as follows. For each utterance $\boldsymbol{O}_i^{(s)}$, spoken by speaker s during recording session i, the observations generated by mixture component m are assumed to be distributed with covariance $\boldsymbol{\Sigma}_m$ and mean $\boldsymbol{\mu}_{im}^{(s)}$. The session and speaker dependent mean $\boldsymbol{\mu}_{im}^{(s)}$ is then modelled as the sum of a session-independent, speaker-dependent mean $\boldsymbol{\mu}_m^{(s)}$ and a term dependent on a vector $\boldsymbol{x}_i$ of session factors associated with $\boldsymbol{O}_i^{(s)}$.

$$\boldsymbol{\mu}_{im}{}^{(s)} = \boldsymbol{\mu}_m{}^{(s)} + \boldsymbol{U}_m\boldsymbol{x}_i \tag{4.24}$$

where $\boldsymbol{U}_m$ is a factor loading matrix that defines the orientation of a low dimensional subspace containing all session-dependent variation.

Similarly, for each speaker s, the speaker-dependent, session-independent component mean $\boldsymbol{\mu}_m{}^{(s)}$ is modelled as the sum of a speaker-independent mean parameter $\boldsymbol{\mu}_m$ and two speaker-dependent terms.

$$\boldsymbol{\mu}_m^{(s)} = \boldsymbol{\mu}_m + \boldsymbol{V}_m\boldsymbol{y}^{(s)} + \boldsymbol{D}_m\boldsymbol{z}_m^{(s)} \tag{4.25}$$

Vector $\boldsymbol{y}^{(s)}$ consists of latent speaker factors which determine the position of speaker s within a low dimensional speaker subspace defined by the factor loading matrix $\boldsymbol{V}_m$. The factor analysis model can also optionally includes an additional set of speaker and component-dependent factors $\boldsymbol{z}_m^{(s)}$, where the associated factor-loading matrix $\boldsymbol{D}_m$ is diagonal. The purpose of these additional factors is to model any additional speaker-dependent variation not modelled by $\boldsymbol{y}_m$. For example, when $\boldsymbol{V}_m = \boldsymbol{0}$ and $\boldsymbol{U}_m = \boldsymbol{0}$ the model resembles the standard GMM approach. The inclusion of the $\boldsymbol{D}_m\boldsymbol{z}_m^{(s)}$ term dramatically increases the complexity of the model. It is therefore common to make the assumption $\boldsymbol{D}_m = \boldsymbol{0}$. In the literature this approach is referred to as the PCA case. A related model, in which only channel variability is modelled with latent factors was used in [200].

For each component m the parameters of the factor-analysis model are defined by a tuple $\boldsymbol{\lambda}_m = \{\boldsymbol{\mu}_m, \boldsymbol{U}_m, \boldsymbol{V}_m, \boldsymbol{D}_m, \boldsymbol{\Sigma}_m\}$. The complete set of model parameters is defined by $\boldsymbol{\lambda} = \{\boldsymbol{\lambda}_1 \ldots \boldsymbol{\lambda}_M\}$. This form of model is closely related to cluster-adaptive training with a Gaussian prior defined over the cluster weights. The factor analysis model described by equation 4.24 is analogous to a CAT system in which a subset of cluster weights are tied over all utterances belonging to a particular speaker. Schemes for calculating likelihoods and for iteratively updating the model parameters are described in the following subsections.

4.4.1 Likelihood function

The joint likelihood of a set of N utterances $\{\boldsymbol{O}_1^{(s)}, \ldots \boldsymbol{O}_N^{(s)}\}$ associated with speaker s is dependent on both the model parameters $\boldsymbol{\lambda}$ and on the latent factors $\bar{\boldsymbol{x}}^{(s)}$ associated with speaker s and the session conditions for each utterance .

$$\bar{\boldsymbol{x}}^{(s)} = [\boldsymbol{x}_1^{\mathsf{T}}, \ldots, \boldsymbol{x}_N^{\mathsf{T}}, \boldsymbol{y}^{(s)\mathsf{T}}, \boldsymbol{z}_1^{(s)\mathsf{T}}, \ldots, \boldsymbol{z}_M^{(s)\mathsf{T}}]^{\mathsf{T}} \tag{4.26}$$

If the values of the latent factors were known then $\boldsymbol{\mu}_{im}{}^{(s)}$ could be explicitly calculated for each utterance and the associated likelihood estimated directly. Since $\bar{\boldsymbol{x}}^{(s)}$ is unknown, calculating $p(\boldsymbol{O}_1^{(s)}, \ldots, \boldsymbol{O}_N^{(s)}; \boldsymbol{\lambda})$ requires marginalising over the latent factors.

$$p(\boldsymbol{O}_1^{(s)}, \ldots, \boldsymbol{O}_N^{(s)}; \boldsymbol{\lambda}) = \int p(\boldsymbol{O}_1^{(s)}, \ldots, \boldsymbol{O}_N^{(s)} | \bar{\boldsymbol{x}}^{(s)}; \boldsymbol{\lambda}) \mathcal{N}(\bar{\boldsymbol{x}}^{(s)}; \boldsymbol{0}, \boldsymbol{I}) \quad d\bar{\boldsymbol{x}}^{(s)} \tag{4.27}$$

A closed form solution to this integral can be calculated [106] in terms of the expected value $\mathcal{E}\{\bar{\boldsymbol{x}}\}$ and covariance $\boldsymbol{\Sigma}_{\bar{\boldsymbol{x}}}$ of the latent variables.

$$\begin{aligned} \log p(\boldsymbol{O}_1^{(s)}, \ldots, \boldsymbol{O}_N^{(s)}; \boldsymbol{\lambda}) \;=\; & G + \frac{1}{2}\log|\boldsymbol{\Sigma}_{\bar{\boldsymbol{x}}}| + \frac{1}{2}\sum_{i=1}^{N}\Bigg[\mathcal{E}\{\boldsymbol{x}_i^{\mathsf{T}}\}\sum_{m=1}^{M}\boldsymbol{U}_m^{\mathsf{T}}\boldsymbol{\Sigma}_m^{-1}\boldsymbol{f}_{im} \\ & + \mathcal{E}\{\boldsymbol{y}^{(s)\mathsf{T}}\}\sum_{m=1}^{M}\boldsymbol{V}_m^{\mathsf{T}}\boldsymbol{\Sigma}_m^{-1}\boldsymbol{f}_{im} + \sum_{m=1}^{M}\mathcal{E}\{\boldsymbol{z}_m^{(s)\mathsf{T}}\}\boldsymbol{D}_m\boldsymbol{\Sigma}_m^{-1}\boldsymbol{f}_{im}\Bigg] \end{aligned} \tag{4.28}$$

$$G \;=\; \sum_{i=1}^{N}\sum_{m=1}^{M}\left[N_{im}\log\frac{1}{(2\pi)^{\frac{D}{2}}|\boldsymbol{\Sigma}_m|^{\frac{1}{2}}} - \frac{1}{2}\mathtt{Tr}(\boldsymbol{\Sigma}_m^{-1}\boldsymbol{S}_{im})\right] \tag{4.29}$$

where D is the dimension of the observations and $\boldsymbol{f}_{im}$, $\boldsymbol{S}_{im}$, N_{im} are sufficient statistics calculated for each utterance $\boldsymbol{O}_i^{(s)} = \{\boldsymbol{o}_{i1}, \ldots, \boldsymbol{o}_{iT}\}$.

$$\boldsymbol{f}_{im} = \sum_t P(\theta_t = m|\boldsymbol{o}_{it})(\boldsymbol{o}_{it} - \boldsymbol{\mu}_m) \tag{4.30}$$

$$\boldsymbol{S}_{im} = \sum_t P(\theta_t = m|\boldsymbol{o}_{it})(\boldsymbol{o}_{it} - \boldsymbol{\mu}_m)(\boldsymbol{o}_{it} - \boldsymbol{\mu}_m)^{\mathsf{T}} \tag{4.31}$$

$$N_{im} = \sum_t P(\theta_t = m|\boldsymbol{o}_{it}) \tag{4.32}$$

where $P(\theta_t = m|\boldsymbol{o}_t)$ is the posterior probability of component m given observation $\boldsymbol{o}_t$. These probabilities may be estimated from a background GMM, such as the UBM.

In order to calculate the likelihood function and re-estimate the model parameters it is necessary to first calculate the expected value $\mathcal{E}\{\bar{\boldsymbol{x}}_i^{(s)}\}$ of the latent factors as well as the determinant of their covariance $|\boldsymbol{\Sigma}_{\bar{\boldsymbol{x}}}^{(s)}|$. The details of obtaining $\mathcal{E}\{\bar{\boldsymbol{x}}_i^{(s)}\}$ and $|\boldsymbol{\Sigma}_{\bar{\boldsymbol{x}}}^{(s)}|$ are omitted here for brevity, but can be found in [106]. When $\boldsymbol{D}_m = 0$ and $\boldsymbol{V}_m = 0$ for all m, the expected value associated with $\boldsymbol{x}_i$ is

$$\mathcal{E}\{\boldsymbol{x}_i\} = \left[\boldsymbol{I} + \sum_{m=1}^{M} N_{im}\boldsymbol{U}_m^{\mathsf{T}}\boldsymbol{\Sigma}_m^{-1}\boldsymbol{U}_m\right]^{-1}\left[\sum_{m=1}^{M}\boldsymbol{U}_m^{\mathsf{T}}\boldsymbol{\Sigma}_m^{-1}\boldsymbol{f}_{im}\right] \tag{4.33}$$

Aside from the identity matrix, this has a similar form to the cluster weight re-estimation formula for CAT, described in chapter 2.

In the factor analysis model, it is assumed that speaker-dependent factors are fixed over all recordings from a speaker. Hence equation 4.27 can only be used to calculate the joint-likelihood of all utterances associated with a particular speaker. For a test utterance $\boldsymbol{O}^{\mathsf{v}}$ and enrollment utterances $\{\boldsymbol{O}_1^{\mathsf{e}}, \ldots, \boldsymbol{O}_N^{\mathsf{e}}\}$ associated with the claimed identity s, a score may be assigned to $\boldsymbol{O}^{\mathsf{v}}$ by comparing the joint likelihood of $\boldsymbol{O}^{\mathsf{v}}$ and the enrollment data given that they are independent (different speakers) with the likelihood given that they are not independent (same speaker).

$$S^{\texttt{batch}}(\boldsymbol{O}^{\mathsf{v}}; s) = \log \frac{p(\boldsymbol{O}^{\mathsf{v}}, \boldsymbol{O}_1^{\mathsf{e}}, \ldots, \boldsymbol{O}_N^{\mathsf{e}}; \boldsymbol{\lambda})}{p(\boldsymbol{O}^{\mathsf{v}}, \boldsymbol{\lambda})p(\boldsymbol{O}_1^{\mathsf{e}}, \ldots, \boldsymbol{O}_N^{\mathsf{e}}; \boldsymbol{\lambda})} \tag{4.34}$$

$S^{\texttt{batch}}(\boldsymbol{O}^{\mathsf{v}}; s)$ may be calculated using equation 4.27. However, the expected value and covariance of the latent factors must be estimated independently for each term in equation 4.34. This approach is often computationally infeasible when speakers have many enrollment utterances. However, equation 4.34 may be approximated by adapting $\boldsymbol{\lambda}$ to a particular speaker and evaluating the score.

$$S(\boldsymbol{O}^{\mathsf{v}}; s) = \log \frac{p(\boldsymbol{O}^{\mathsf{v}}; \boldsymbol{\lambda}^{(s)})}{p(\boldsymbol{O}^{\mathsf{v}}; \boldsymbol{\lambda})} \tag{4.35}$$

Equation 4.35 is conceptually similar to the standard LLR approach for GMMs. This approximation is only equal to equation 4.34 in the PCA case ($\boldsymbol{D}_m = 0$) [110]. However, due to the difficulty in evaluating equation 4.34 it is typically also used in the general case. Generally only $\boldsymbol{\mu}_m$, $\boldsymbol{V}_m$ and $\boldsymbol{D}_m$ are adapted to a particular target speaker. $\boldsymbol{U}_m$ and $\boldsymbol{\Sigma}_m$ are kept fixed since it is assumed that channel effects are speaker-independent.

4.4.2 Maximum likelihood parameter estimation

In [108], a parameter re-estimation scheme for factor analysis was proposed that seeks to maximise the total likelihood of the training data. At each iteration k the likelihood of the training data is guaranteed to not decrease.

$$\sum_{s=1}^{S} p(\mathcal{O}^{(s)}; \boldsymbol{\lambda}^{(k+1)}) \geq \sum_{s=1}^{S} p(\mathcal{O}^{(s)}; \boldsymbol{\lambda}^{(k)}) \tag{4.36}$$

where $\mathcal{O}^{(s)}$ is a set of $N^{(s)}$ enrollment utterances associated with speaker s. A suitable estimate of the speaker and session independent mean vector $\boldsymbol{\mu}_m$ may be obtained from the parameters of a UBM, if available. In the following re-estimation scheme $\boldsymbol{\mu}_m$ is assumed to be known and fixed at each iteration. An alternative scheme where $\boldsymbol{\mu}_m$ is additionally re-estimated is provided in [106].

At each iteration, the expected values $\mathcal{E}\{\bar{\boldsymbol{x}}^{(s)}\}$ and second order moments $\mathcal{E}\{\bar{\boldsymbol{x}}^{(s)}\bar{\boldsymbol{x}}^{(s)\mathtt{T}}\}$ of the latent factors associated with each training speaker s are estimated as described in [106]. The sufficient statistics $N_{im}^{(s)}$, $\boldsymbol{S}_{im}^{(s)}$ and $\boldsymbol{f}_{im}^{(s)}$ are also accumulated for each utterance in $\mathcal{O}^{(s)}$ according to equations 4.30, 4.31 and 4.32. Given $\mathcal{E}\{\bar{\boldsymbol{x}}^{(s)}\}$ and $\mathcal{E}\{\bar{\boldsymbol{x}}\bar{\boldsymbol{x}}^{\mathtt{T}}\}$, new estimates of $\boldsymbol{U}_m$, $\boldsymbol{V}_m$ and $\boldsymbol{D}_m$ may then be obtained. For each row j of the factor loading matrices, new estimates of $\boldsymbol{U}_{mj}^{(k+1)}$, $\boldsymbol{V}_{mj}^{(k+1)}$ and $\boldsymbol{D}_{mj}^{(k+1)}$ are given by

$$\begin{pmatrix} \boldsymbol{U}_{mj}^{(k+1)} & \boldsymbol{V}_{mj}^{(k+1)} & \boldsymbol{D}_{mj}^{(k+1)} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\mathcal{C}}_{mj} & \boldsymbol{\mathcal{Q}}_{mj} \end{pmatrix} \begin{pmatrix} \boldsymbol{\mathcal{A}}_m & \boldsymbol{\mathcal{B}}_{mj}^{\mathtt{T}} \\ \boldsymbol{\mathcal{B}}_{mj} & \boldsymbol{\mathcal{P}}_{mj} \end{pmatrix}^{-1} \tag{4.37}$$

$$\tag{4.38}$$

where $\boldsymbol{\mathcal{B}}_{mj}$,$\boldsymbol{\mathcal{C}}_{mj}$ are the jth rows of $\boldsymbol{\mathcal{B}}_m$, $\boldsymbol{\mathcal{C}}_m$ and $\boldsymbol{\mathcal{P}}_{mj}$,$\boldsymbol{\mathcal{Q}}_{mj}$ are the jth entries of the diagonal matrices $\boldsymbol{\mathcal{P}}_m$,$\boldsymbol{\mathcal{Q}}_m$, defined below.

$$\boldsymbol{\mathcal{A}}_m = \sum_{s=1}^{S} \sum_{i}^{N^{(s)}} N_{im}^{(s)} \mathcal{E}\{\boldsymbol{x}_i^{(s)} \boldsymbol{x}_i^{(s)\mathtt{T}}\} \tag{4.39}$$

$$\boldsymbol{\mathcal{B}}_m = \sum_{s=1}^{S} \sum_{i}^{N^{(s)}} N_{im}^{(s)} \mathcal{E}\{\boldsymbol{z}_m^{(s)} \boldsymbol{x}_i^{(s)\mathtt{T}}\} \tag{4.40}$$

$$\boldsymbol{\mathcal{C}}_m = \sum_{s=1}^{S} \sum_{i}^{N^{(s)}} \boldsymbol{f}_{im}^{(s)} \mathcal{E}\{\boldsymbol{x}_i^{(s)\mathtt{T}}\} \tag{4.41}$$

$$\boldsymbol{\mathcal{P}}_m = \sum_{s=1}^{S} \mathtt{diag} \left(\mathcal{E}\{\boldsymbol{z}_m^{(s)} \boldsymbol{z}_m^{(s)\mathtt{T}}\} \sum_{i}^{N^{(s)}} N_{im}^{(s)} \right) \tag{4.42}$$

$$\boldsymbol{\mathcal{Q}}_m = \sum_{s=1}^{S} \mathtt{diag} \left(\mathcal{E}\{z_m^{(s)\mathtt{T}}\} \sum_{i}^{N^{(s)}} \boldsymbol{f}_{im}^{(s)} \right) \tag{4.43}$$

Given these statistics, an updated estimate $\boldsymbol{\Sigma}_m^{(k+1)}$ of the covariance matrices may also be calculated.

$$\boldsymbol{\Sigma}_m^{(k+1)} = \frac{1}{N_m} \left[\boldsymbol{S}_m - \mathtt{diag} \left(\boldsymbol{\mathcal{C}}_m \left[\boldsymbol{U}_m \boldsymbol{V}_m \right]^{\mathtt{T}} + \boldsymbol{\mathcal{Q}}_m \boldsymbol{D}_m \right) \right] \tag{4.44}$$

where

$$\boldsymbol{S}_m = \sum_{s=1}^{S} \sum_{i=1}^{N^{(s)}} \boldsymbol{S}_{im}^{(s)} \tag{4.45}$$

$$N_m = \sum_{s=1}^{S} \sum_{i=1}^{N^{(s)}} N_{im}^{(s)} \tag{4.46}$$

An alternative training algorithm for factor analysis models, based on minimising a divergence measure, was proposed in [107]. This algorithm is known to be faster than the maximum likelihood scheme. However, it keeps the orientation of the speaker and session subspaces fixed throughout successive iterations. Hence, the maximum likelihood algorithm is typically used to train $\boldsymbol{\lambda}$ on a large background dataset, while the minimum divergence is used to adapt the parameters $\boldsymbol{\mu}_m$, $\boldsymbol{V}_m$ and $\boldsymbol{D}_m$ to a particular speaker.

4.5 SVM-based speaker verification

Recently there has been interest in applying discriminative classifiers to the SV task. One form of discriminative classifier that has become popular is the support vector machine (SVM), described in chapter 2. SVM-based classifiers have been found to outperform traditional generative model based approaches [28, 48, 190] and, alongside factor analysis techniques, have become the dominant approach for SV in state of the art systems [75, 102, 126].

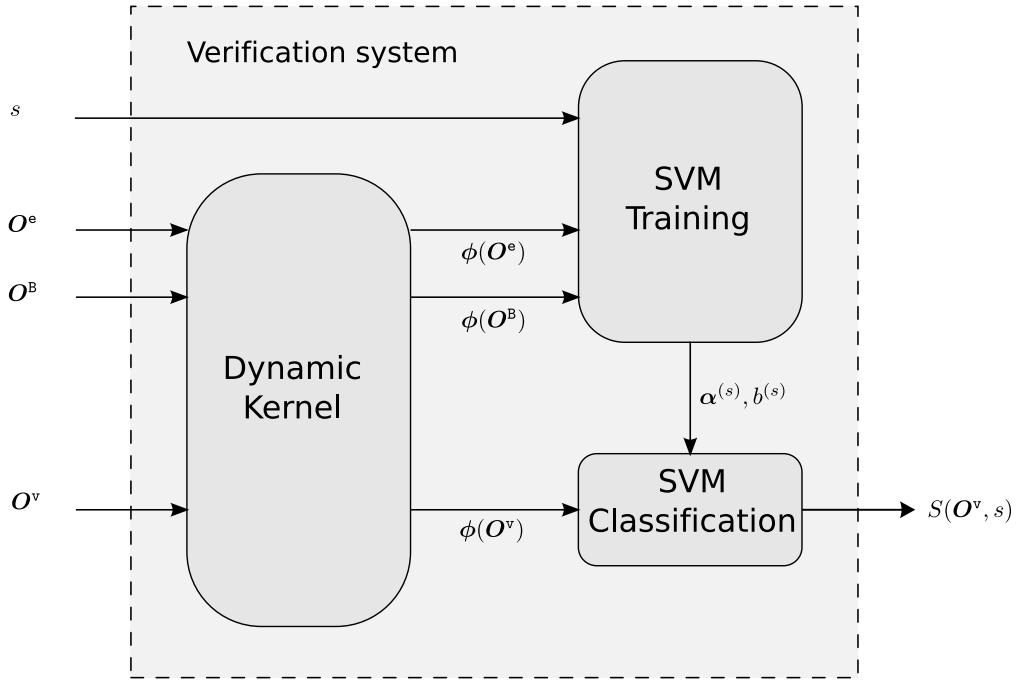
An overview of an SVM-based SV system is given in figure 4.4. Distinct SVM classifier parameters $\{\boldsymbol{\alpha}^{(s)}, b^{(s)}\}$ are trained for each enrolled speaker s. Positive training examples ($y^{(s)} = 1$) are obtained from the enrollment utterances provided by the speaker. Negative examples ($y^{(s)} = -1$) are obtained from a large background dataset that contains utterances from a range of speakers. Given a test utterance $\boldsymbol{O}^{\mathrm{v}}$ with identity claim s, the SVM classifier score is given by

$$S(\boldsymbol{O}^{\mathrm{v}}, s) = \sum_{i=1}^{N} \alpha_i^{(s)} y_i^{(s)} K(\boldsymbol{O}_i, \boldsymbol{O}^{\mathrm{v}}) + b^{(s)} \tag{4.47}$$

where $y_i^{(s)} \in \{-1, 1\}$ is the label associated with training utterance $\boldsymbol{O}_i$ and is dependent on the identity of the enrolled speaker. The bias term $b^{(s)}$ associated with each classifier acts as a speaker-dependent threshold. It also maps the output scores from each classifier into a consistent range, allowing a global threshold to be set for all target speakers.

4.5.1 Dynamic kernels for SVM-based speaker verification

The nature of the SVM classifier is dependent on the form of dynamic kernel $K(\boldsymbol{O}_i, \boldsymbol{O}_j)$ used. Many of the dynamic kernels described in chapter 3 have been applied to SV, see for example [24, 28, 48, 120, 189, 205, 211, 213]. Most state-of-the-art SVM-based SV systems parameterise utterances as sequences of continuous cepstral observations. For these systems, the dominant form of dynamic kernel is the GMM-supervector kernel. Systems submitted to the 2008 NIST SRE that used this form of dynamic kernel include [75, 102, 126]. The GMM-supervector kernel is typically implemented using GMMs with up to 1000 Gaussian



A dynamic kernel is used to (implicitly) transform each variable-length observation sequence $\boldsymbol{O}$ into a fixed-dimensional representation $\boldsymbol{\phi}(\boldsymbol{O})$. A distinct SVM classifier is then trained for each target speaker using the available enrollment data $\boldsymbol{\phi}(\boldsymbol{O}^{\mathrm{e}})$ and a background dataset of imposter utterances $\boldsymbol{O}^{\mathrm{B}}_1, \ldots, \boldsymbol{O}^{\mathrm{B}}_N$. The classifier output for a test utterance $\boldsymbol{O}^{\mathrm{v}}$ with associated identity claim s is obtained using the SVM classifier parameters $(\boldsymbol{\alpha}^s, b^{(s)})$ trained using enrollment data from target speaker s.

Figure 4.4: Speaker verification using support vector machines.

components. A UBM is initially trained using background data from many speakers, as described in section 4.3. The mean parameters of this UBM are then adapted using MAP to generate a range of utterance dependent distributions. Typically the MAP adaptation constant $\tau^{\texttt{map}}$ is in the range 10-25. In practice, the use of a fixed number of components and a single background model may limit the performance of the system. These restrictions are caused by the matched-pair bound on the KL divergence used to derive the GMM-supervector kernel. Variational kernels, closely related to GMM-supervector kernels, are proposed in chapter 5. Unlike the GMM-supervector kernel these are compatible with multiple background model and a range of model sizes.

A second form of dynamic kernel that is used in state-of-the art SV systems [102, 126] is the MLLR kernel, described in chapter 3. Here the kernel is implemented by using MLLR to adapt the mean parameters of a trained HMM-based speech recogniser. Typically up to 16 full MLLR transforms are used, each associated with a distinct regression class. The MLLR transform coefficients are then concatenated to obtain the final feature vector. Dynamic kernels derived from factor analysis modelling have also been applied to SV [47]. Here a factor analysis model is trained as described in section 4.4. For each utterance, the speaker factors $\boldsymbol{y}^{(s)}$ and/or the speaker and component dependent factors $\boldsymbol{z}_m^{(s)}$ are concatenated to obtain an compact utterance-dependent feature vector. However, this combined SVM/FA approach was not found to yield gains over standard factor analysis modelling. SVM-based systems may also be combined with long-term prosodic and language features. Word and SNERF parameterisations, described in section 4.2.3, were used in [102]. Unlike cepstral parameterisations, these features are typically discrete. N-gram and term-frequency log likelihood ratio kernels, described in chapter 3 have been successfully applied [25, 182] using these features. Although systems based on prosodic features tend to perform worse than those built using spectral features, they have been found to give gains in combination [78, 102]. For all SVM-based systems, normalisation techniques such as NAP or WCCN, described in the following section, are usually applied to reduce session-variability.

In recent evaluations, many sites have achieved performance gains by combining multiple classifiers [22, 102]. This is usually implemented by combining the output scores using techniques such as logistic regression [158]. For SVM-based systems, an alternative approach is to combine at the kernel level. The combination of multiple dynamic kernels, as well as a scheme for obtaining suitable kernel weights, is discussed in more detail in chapter 6. In order to effectively combine dynamic kernels, the features must be complementary. The relationship between many forms of dynamic kernel applied to SV are also discussed in chapter 6. Alternatively, dynamic kernels may be combined with traditional static kernels to improve SV performance. Suitable combination approaches are discussed in more detail in chapter 7.

4.5.2 Intersession variability modelling

A significant source of performance degradation in SV systems is caused by variation in acoustic conditions (environment, channel, handset etc) between recording sessions. Recently, there has been interest in explicitly modelling session variability to improve system performance. Factor analysis-based approaches, described in section 4.4, include explicit factors that model channel and speaker variation and have been found to outperform traditional GMM-based SV systems. For SVM-based systems, one approach for modelling intersession variability is via the kernel metric $\boldsymbol{G}$. Here the kernel function is defined by equation 4.48.

$$K(\boldsymbol{O}_i, \boldsymbol{O}_j) = \boldsymbol{\phi}(\boldsymbol{O}_i)^{\mathsf{T}} \boldsymbol{G}^{-1} \boldsymbol{\phi}(\boldsymbol{O}_j) \tag{4.48}$$

In the absence of prior information about session variability, it is appropriate to use a maximally non-committal metric, as described in chapter 2. However, when a suitably-labeled background dataset is available, a metric may be trained that minimises unwanted session variability. Two popular metric-based methods for reducing intersession variability are described in this section. These are nuisance attribute projection (NAP) and within-class covariance normalisation (WCCN).

4.5.2.1 Nuisance attribute projection

Nuisance attribute projection (NAP) [186, 187] is an intersession variability compensation technique that operates by removing a k-dimensional subspace associated with unwanted variability, for example due to changes in handset or background conditions, from the feature space. Here the metric has the form.

$$\boldsymbol{G}^{\mathtt{NAP}^{-1}} = \boldsymbol{P}^{\mathsf{T}} \boldsymbol{P} \tag{4.49}$$

where $\boldsymbol{P}$ is a projection defined as $\boldsymbol{P} = (\boldsymbol{I} - \boldsymbol{V}\boldsymbol{V}^{\mathsf{T}})$ and $\boldsymbol{V}$ is a matrix with k orthonormal columns that defines the subspace to be removed. Given a set of labeled background utterances $\mathcal{O}^{\mathtt{B}} = \{\boldsymbol{O}_1^{\mathtt{B}}, \ldots, \boldsymbol{O}_N^{\mathtt{B}}\}$ the optimal $\boldsymbol{V}$ is obtained when the average cross-channel/session distance between projected feature vectors $\boldsymbol{P}\boldsymbol{\phi}(\boldsymbol{O}_i^{\mathtt{B}})$ is minimised.

$$\boldsymbol{V}^* \quad = \quad \underset{\boldsymbol{V}}{\operatorname{argmin}} \sum_{i,j=1}^{N} W_{ij} ||\boldsymbol{P}\boldsymbol{\phi}(\boldsymbol{O}_i^{\mathtt{B}}) - \boldsymbol{P}\boldsymbol{\phi}(\boldsymbol{O}_j^{\mathtt{B}})||_2^2 \tag{4.50}$$

where W_{ij} defines the relationship between $\boldsymbol{O}_i^{\mathtt{B}}$ and $\boldsymbol{O}_j^{\mathtt{B}}$. W_{ij} can be selected in several different ways, dependent on the type of variability to be removed and the availability of labeled data. When each background utterance $\boldsymbol{O}_i^{\mathtt{B}}$ is labeled with the form of handset used, $h_i \in \{\mathtt{carbon}, \mathtt{electrolet}, \ldots\}$, W_{ij} may be defined to minimise cross-handset variability [187].

$$W_{ij} = \begin{cases} 1 & \text{when } h_i \neq h_j \\ 0 & \text{when } h_i = h_j \end{cases} \tag{4.51}$$

Alternatively, when the background dataset contains multiple sessions from each speaker, W_{ij} may be defined to reduce the global intersession variability of the background data [28].

$$W_{ij} = \begin{cases} 1 & \text{when } s_i = s_j \\ 0 & \text{when } s_i \neq s_j \end{cases} \tag{4.52}$$

where s_i indicates the speaker identity associated with $\boldsymbol{O}_i^{\mathtt{B}}$. Weighted combinations of these two approaches may also be used [187]. The solution to equation 4.50, is obtained by solving an eigenvector problem

$$\boldsymbol{V} = \text{eig}(\boldsymbol{X}\boldsymbol{Z}\boldsymbol{X}^{\mathsf{T}}) \tag{4.53}$$

where the function eig($\boldsymbol{A}$) returns the k eigenvectors of $\boldsymbol{A}$ with the largest eigenvalues and

$$\boldsymbol{X} = \left[\phi(\boldsymbol{O}_1^{\mathtt{B}}) \quad \cdots \quad \phi(\boldsymbol{O}_N^{\mathtt{B}})\right] \tag{4.54}$$

$$\boldsymbol{Z} = \mathtt{diag}(\boldsymbol{W}\boldsymbol{1}) - \boldsymbol{W} \tag{4.55}$$

$\mathtt{diag}(\boldsymbol{a})$ is a function that returns a square diagonal matrix whose diagonal elements are the indices of $\boldsymbol{a}$. $\boldsymbol{1}$ is a column vector with all elements equal to one. The dimension k of the nuisance subspace is usually tuned on a development dataset and will typically be around 1% of the dimension of the feature space.

NAP is closely related to the factor analysis-based approach described in section 4.4. When NAP is applied to appropriately normalised GMM-supervector features and $\boldsymbol{W}$ is selected using equation 4.52 the subspace obtained is identical to the channel-dependent factor loading matrix $\boldsymbol{U}_m$ associated with factor analysis [28]. Although the subspaces are identical, they are used in very different ways. In NAP, the nuisance subspace is projected out using an appropriate metric. In factor analysis, an iterative approach is used to estimate the latent session-dependent factors, which are then subtracted from the mean supervector.

For kernel functions where the associated feature space is sufficiently high-dimensional or does not have an explicit representation, $\phi(\boldsymbol{O})$, it is not possible to evaluate equation 4.53. One alternative is to use the form of kernelised NAP derived in [219]. Here kernel PCA [179] is applied to implement NAP without needing to explicitly evaluate $\phi(\boldsymbol{O})$. The NAP normalised kernel function is defined by

$$K^{\mathtt{NAP}}(\boldsymbol{O}_i, \boldsymbol{O}_j) = K(\boldsymbol{O}_i, \boldsymbol{O}_j) - \boldsymbol{k}_i{}^{\mathtt{T}} \boldsymbol{Z}^{\frac{1}{2}} \boldsymbol{Y}\boldsymbol{Y}^{\mathtt{T}} \boldsymbol{Z}^{\frac{1}{2}} \boldsymbol{k}_j \tag{4.56}$$

where $\boldsymbol{Y}, \boldsymbol{k}_i$ and $\boldsymbol{Z}$ are defined by

$$\boldsymbol{Y} = \mathrm{eig}(\boldsymbol{Z}^{\frac{1}{2}} \boldsymbol{K}^{\mathtt{B}} \boldsymbol{Z}^{\frac{1}{2}}) \tag{4.57}$$

$$\boldsymbol{k}_i = \left[K(\boldsymbol{O}_i, \boldsymbol{O}_1^{\mathtt{B}}), \ldots, K(\boldsymbol{O}_i, \boldsymbol{O}_N^{\mathtt{B}})\right] \tag{4.58}$$

$$\boldsymbol{Z}^{\frac{1}{2}} = \mathtt{diag}(\boldsymbol{W}\boldsymbol{1})^{\frac{1}{2}} \left[\boldsymbol{I} - \boldsymbol{W}\right] \tag{4.59}$$

The elements of the background kernel matrix $\boldsymbol{K}^{\mathtt{B}}$ are defined by $K_{ij}^{\mathtt{B}} = K(\boldsymbol{O}_i^{\mathtt{B}}, \boldsymbol{O}_j^{\mathtt{B}})$. Unlike equation 4.53, equation 4.56 can be evaluated for kernels that do not have an explicit feature space.

4.5.2.2 Within-class covariance normalisation

The effect of a maximally non-committal metric is to normalise the global covariance of the feature space. When a labeled training set is available that contains multiple recording sessions per speaker, an alternative approach is to apply within-class covariance normalisation (WCCN) [81]. WCCN is a metric-based scheme that aims to reduce intersession variability. Here the metric is defined to normalise the expected within-class covariance matrix defined by

$$\boldsymbol{G}^{\mathtt{WCCN}} = \sum_{s=1}^{S} P(s) \boldsymbol{G}^{(s)} \tag{4.60}$$

where $P(s)$ is the prior probability of speaker s and $\boldsymbol{G}^{(s)}$ is the expected within class covariance matrix for speaker s defined by

$$\boldsymbol{G}^{(s)} = \mathcal{E}\left\{(\boldsymbol{\phi}(\boldsymbol{O}) - \boldsymbol{\mu}_\phi)(\boldsymbol{\phi}(\boldsymbol{O}) - \boldsymbol{\mu}_\phi)^{\mathtt{T}}\right\} \quad (4.61)$$

$$\boldsymbol{\mu}_\phi^{(s)} = \mathcal{E}\left\{\boldsymbol{\phi}(\boldsymbol{O})\right\} \quad (4.62)$$

where $\mathcal{E}\{\}$ is the expectation over all utterances associated with speaker s. Like NAP, WCCN uses a labeled training set to define a metric that reduces intersession variability. However, unlike NAP, WCCN also weights each feature space dimension to minimise an upper bound on the error rate [80]. $\boldsymbol{G}^{(s)}$ may be approximated by the covariance of the training set feature vectors associated with speaker s. When the amount of training data available for estimating $\boldsymbol{G}^{\mathtt{WCCN}}$ is limited, the estimate obtained may not be robust. One option is to use a weighted interpolation between $\boldsymbol{G}^{\mathtt{WCCN}}$ and the identity metric. A smoothed estimate $\hat{\boldsymbol{G}}^{\mathtt{WCCN}}$ is then given by

$$\hat{\boldsymbol{G}}^{\mathtt{WCCN}} = (1-\upsilon)\boldsymbol{G}^{\mathtt{WCCN}} + \upsilon\boldsymbol{I} \quad (4.63)$$

where υ is a smoothing factor in the range $[0 \ldots 1]$ that is empirically set by the user.

When the dimension, F, of $\boldsymbol{\phi}(\boldsymbol{O})$ is large, it may not be feasible to invert $\boldsymbol{G}^{\mathtt{WCCN}}$. Since the dimension of the feature space is typically significantly larger than the number of training speakers, one option is to apply PCA to project each feature vector into a smaller, $F^{\mathtt{PCA}}$-dimensional space. WCCN may then be efficiently applied within this space [80, 100]. The projected feature vector $\boldsymbol{\phi}^{\mathtt{PCA}}(\boldsymbol{O})$ is defined by

$$\boldsymbol{\phi}^{\mathtt{PCA}}(\boldsymbol{O}) = \boldsymbol{U}^{\mathtt{T}}\boldsymbol{\phi}(\boldsymbol{O}) \quad (4.64)$$

where $\boldsymbol{U}$ is the eigenvector decomposition of the global covariance of the training data $\boldsymbol{\Sigma}_{\boldsymbol{\phi}}$.

$$\boldsymbol{\Sigma}_{\boldsymbol{\phi}} = \boldsymbol{U}\Lambda\boldsymbol{U}^{\mathtt{T}} \quad (4.65)$$

The PCA feature space contains all of the covariance information contained within the training feature vectors. However test feature vectors will contain additional information that can not be fully expressed within the PCA subspace. This issue can be overcome by defining a PCA-complement feature vector that contains all the information contained in the original features but not in the PCA set. The PCA-complement feature vector is defined by

$$\boldsymbol{\phi}^{\overline{\mathtt{PCA}}}(\boldsymbol{O}) = (\boldsymbol{I} - \boldsymbol{U}\boldsymbol{U}^{\mathtt{T}})\boldsymbol{\phi}(\boldsymbol{O}) \quad (4.66)$$

The PCA-complement feature vector is only non-zero for test utterances. The kernel function is then defined by a weighted concatenation of the PCA and PCA-complement feature vectors.

$$K^{\mathtt{WCCN}}(\boldsymbol{O}_i, \boldsymbol{O}_j) = \begin{bmatrix} \boldsymbol{\phi}^{\mathtt{PCA}}(\boldsymbol{O}_i) \\ \boldsymbol{\phi}^{\overline{\mathtt{PCA}}}(\boldsymbol{O}_i) \end{bmatrix}^{\mathtt{T}} \begin{bmatrix} \gamma^2\boldsymbol{G}^{\mathtt{PCA}\,-1} & 0 \\ 0 & (1-\gamma)^2\boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^{\mathtt{PCA}}(\boldsymbol{O}_j) \\ \boldsymbol{\phi}^{\overline{\mathtt{PCA}}}(\boldsymbol{O}_j) \end{bmatrix} \quad (4.67)$$

where γ is a constant set by the user in the range range $[0 \ldots 1]$. When $\gamma = 1$ only information contained within the PCA subspace is used by the kernel. In this case the kernel function is closely related to the function obtained using NAP with $k = F - F^{\mathtt{PCA}}$ [101]. $\boldsymbol{G}^{\mathtt{PCA}}$ is the metric defined by equation 4.60 when $\boldsymbol{\phi}(\boldsymbol{O}) = \boldsymbol{\phi}^{\mathtt{PCA}}(\boldsymbol{O})$. Since the training set does not contain any information about the covariance of the PCA-complement subspace it is not possible to learn a suitable metric. Instead, an identity metric is used to normalise these features.

4.6 Score-normalisation

Score-normalisation is a post-processing stage applied to the output scores of a speaker verification system. Score-normalisation techniques are generally used to perform two functions. They can be used to perform a classifier-dependent scaling of the output distributions. This allows a global threshold to be set over a range of speaker-dependent classifiers. Score-normalisation techniques can also be used to compensate for environmental or channel conditions. The LLR framework described in section 4.3 is one example of score-normalisation. In this setup the UBM is used to generate a utterance-dependent normalisation term, allowing a single fixed threshold to be used. Similarly, the use of handset-dependent background models [83] is an example of score-normalisation designed to improve noise-robustness. Although the techniques described in this section were developed for use in a GMM-LLR framework, they may also be applied to normalise the output of SVM-based systems.

4.6.1 Zero normalisation

Zero normalisation (Z-norm) [6, 171] was an early form of score-normalisation. The purpose of Z-norm is to normalise the output distribution of each speaker-dependent classifier to zero mean and unit variance, allowing a global threshold to be set. Since the amount of available enrollment data per speaker is typically limited, rather than normalising the global output distribution, only the distribution of imposter scores is normalised. Here the normalised scores are given by

$$\hat{S}(\boldsymbol{O}, s) = \frac{S(\boldsymbol{O}, s) - \mu^{\mathtt{I(s)}}}{\sigma^{\mathtt{I(s)}}} \tag{4.68}$$

Where $\mu^{\mathtt{I(s)}}$ and $\sigma^{\mathtt{I(s)}}$ is the mean and standard deviation of the scores obtained by classifying a series of imposter utterances using the speaker-dependent classifiers. An advantage of this approach is that the speaker-dependent normalisation parameters $\mu^{\mathtt{I(s)}}$ and $\sigma^{\mathtt{I(s)}}$ are not dependent on $\boldsymbol{O}$ and hence may be computed offline. If a development set of utterances is available that were not spoken by any of the enrolled speakers, then the set may be reused to obtain all speaker-dependent normalisation parameters.

A closely related form of score-normalisation is handset normalisation (H-norm) [171]. This form of score-normalisation seeks to normalise out variation due to differences in the type of handset e.g. carbon, electrolet. Here, rather than applying a speaker-dependent normalisation. handset-dependent normalisation parameters $\mu^{(h)}$ and $\sigma^{(h)}$ are learned from a development dataset. The normalised scores are given by

$$\hat{S}(\boldsymbol{O}, s) = \frac{S(\boldsymbol{O}, s) - \mu^{(h)}}{\sigma^{(h)}} \tag{4.69}$$

Handset normalisation is dependent on either handset information h being provided for each utterances, or an effective handset detection scheme being available. In recent systems such as [111, 126], discrete handset compensation techniques have been superseded by more recent techniques such as NAP or factor analysis modelling.

4.6.2 Test normalisation

One issue with applying techniques such as Z-norm, is that normalisation may not be effective when there is acoustic mismatch between the test data and the development set used to train the normalisation parameters. Test normalisation (T-norm) [6] is a widely used form of score-normalisation that does not suffer from this problem. Test normalisation seeks to reduce the session variability associated with each score. In contrast to Z-norm, where each score is normalised using a range of imposter utterances, T-norm normalises each score using a range of classifiers, each trained to distinguish a different speaker. This approach is closely related to the LLR framework when a set of cohort models are used instead of a UBM. The difference here is that both the mean and variance of the output distribution is normalised. If $\mu^{(\boldsymbol{O})}$ and $\sigma^{(\boldsymbol{O})}$ are the mean and standard deviation of the scores obtained by classifying $\boldsymbol{O}$ using a range of speaker-dependent classifiers, the normalised scores are given by

$$\hat{S}(\boldsymbol{O}, s) = \frac{S(\boldsymbol{O}, s) - \mu^{(\boldsymbol{O})}}{\sigma^{(\boldsymbol{O})}} \tag{4.70}$$

When T-norm is applied within a LLR framework, the contribution of the UBM will be normalised away. It is therefore more efficient to score utterances directly using the speaker-dependent likelihood $S(\boldsymbol{O}, s) = \log p(\boldsymbol{O}; \boldsymbol{\lambda}^{(s)})$. Unlike Z-norm, $\mu^{(\boldsymbol{O})}$ and $\sigma^{(\boldsymbol{O})}$ are dependent on $\boldsymbol{O}$ and therefore cannot be computed offline. For GMM-LLR based systems, T-norm can be efficiently approximated by only evaluating the top N most likely Gaussian components given $\boldsymbol{O}$. When the T-norm cohort models are all adapted from the same UBM, the components of the models will be strongly coordinated. Thus the top N components given $\boldsymbol{O}$ may be assumed to be the same for all cohort models allowing efficient evaluation. Z-norm and T-norm normalise the scores in different ways and the two forms of normalisation may be combined. These approaches were found to be complementary in [200].

4.6.3 Adaptive T-norm

In T-norm, a set of cohort speaker-dependent classifiers is used to normalise the score associated with each utterance during test. Adaptive T-norm (AT-norm) [191] is a variant of T-norm where the set of classifiers used varies depending on the target speaker. By automatically selecting a set of cohort classifiers designed for speakers that closely resemble the target speaker, SV performance may be improved.

Given a set of cohort classifiers $\mathcal{P}$, a speaker-dependent subset $\mathcal{P}^{(s)}$ is computed offline for each target speaker s. In [191], cohort sets were selected by scoring a series of imposter utterances $\{\boldsymbol{O}^{\mathrm{I}}_1, .., \boldsymbol{O}^{\mathrm{I}}_N\}$ using both the target speaker classifier and each cohort classifier. The distance D_{sp} between the target speaker classifier and each cohort classifier p is then obtained using a Manhattan block distance between the vector of imposter scores.

$$D_{sp} = \sum_{i=1}^{N} |S(\boldsymbol{O}^{\mathrm{I}}_i, s) - S(\boldsymbol{O}^{\mathrm{I}}_i, p)| \tag{4.71}$$

where $S(\boldsymbol{O}, p)$ is the score obtained by classifying $\boldsymbol{O}$ using the pth cohort classifier. $\mathcal{P}^{(s)}$ is then comprised of the set of k cohort classifiers closest to the target speaker classifier. The effectiveness of AT-norm is dependent on the availability of a suitable development set from which a large, varied set of cohort classifiers can be trained.

4.7 Evaluation metrics

Speaker verification systems make two distinct forms of error. These are: false alarms, where an imposter speaker is incorrectly accepted, and misses, where a genuine speaker is incorrectly rejected. Given the number of false alarms $N^{\texttt{false-alarm}}$ and misses $N^{\texttt{miss}}$ the miss ($P^{\texttt{miss}}$) and false alarm ($P^{\texttt{false-alarm}}$) probabilities are given by

$$P^{\texttt{miss}} = \frac{N^{\texttt{miss}}}{N^{\texttt{true}}} \tag{4.72}$$

$$P^{\texttt{false-alarm}} = \frac{N^{\texttt{false-alarm}}}{N^{\texttt{imposter}}} \tag{4.73}$$

where $N^{\texttt{true}}$ and $N^{\texttt{imposter}}$ are the number of trials involving genuine and imposter speakers respectively. When the costs associated with each form of error are the same, a useful metric is the accuracy. This is defined by

$$\text{Accuracy} = \frac{N^{\texttt{miss}} + N^{\texttt{false-alarm}}}{N^{\texttt{true}} + N^{\texttt{imposter}}} \tag{4.74}$$

For a particular SV system, the relative frequency of each type of error can be adjusted by varying the operating threshold. Thus directly comparing speaker verification systems is difficult. Several evaluation metrics are discussed in this section. A common property of most of these metrics is that they are threshold-independent.

4.7.1 Equal error rate

The equal error rate (EER) is the most commonly used measure of system performance for SV tasks. The equal error rate is the value of the false alarm and the miss probabilities when the operating threshold is adjusted such that they are equal. The EER score provides a threshold-independent score for which the costs of misses and false alarms are equal. This is demonstrated in figure 4.5

4.7.2 Detection cost function

The equal error rate is only a suitable evaluation metric when the costs of misses and false-alarms are equal. For NIST evaluations, the ratio of imposter trials to target speaker trials is extremely high. Hence, the EER is disproportionately biased by those trials involving target speakers. An alternative metric is the detection cost function (DCF) [143]. This metric is commonly used in NIST SRE evaluations and is given by the weighted sum of the false alarm and miss probabilities at a defined threshold.

$$\text{DCF} = P^{\texttt{tar}} C^{\texttt{miss}} P^{\texttt{miss}} + (1 - P^{\texttt{tar}}) C^{\texttt{false-alarm}} P^{\texttt{false-alarm}} \tag{4.75}$$

where $P^{\texttt{tar}}$ is the prior likelihood of an utterance being spoken by the target speaker and $C^{\texttt{miss}}$ and $C^{\texttt{false-alarm}}$ are respectively the costs associated with a miss and false alarm. For NIST evaluations, the costs and prior are fixed at $C^{\texttt{miss}} = 10$, $C^{\texttt{false-alarm}} = 1$ and $P^{\texttt{tar}} = 0.01$ [144]. Rather than using 4.75 directly, the score is typically normalised by the best cost that can be obtained by either naively accepting or rejecting all trials, defined by

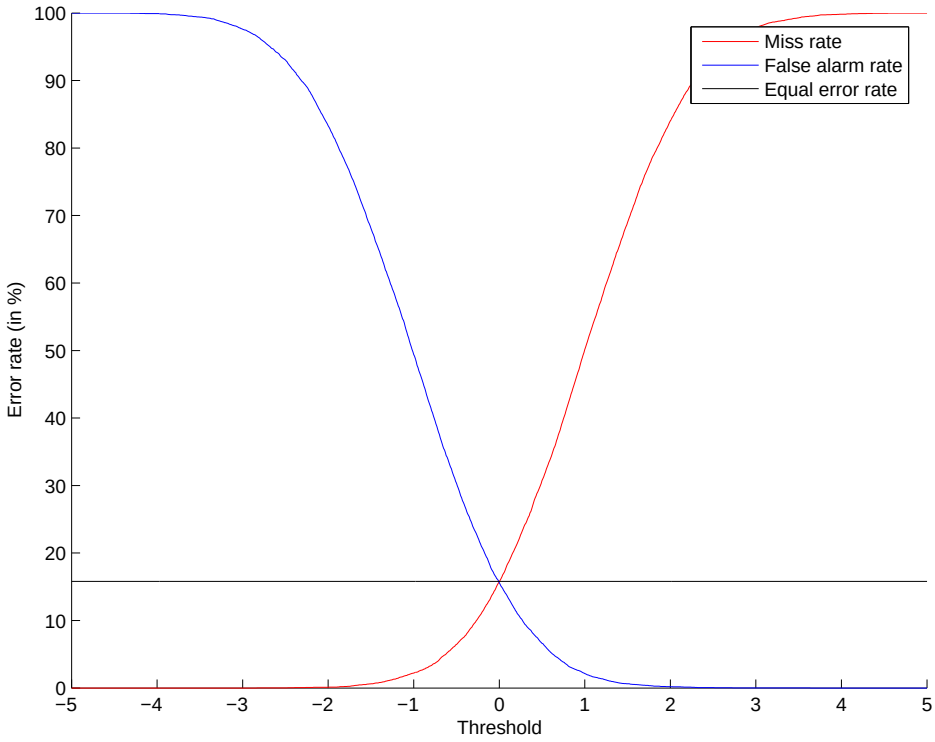


Figure 4.5: Miss and False alarm probabilities for a SV system as the operating threshold varies. The threshold-independent equal error rate (EER) is the error rate obtained when the threshold is adjusted to equalize the miss and false alarm probabilities. For the depicted system an EER of 15.8% is obtained by setting the threshold to zero.

$\min\{P^{\mathtt{tar}}C^{\mathtt{miss}}, (1-P^{\mathtt{tar}})C^{\mathtt{false-alarm}}\}$. When NIST costs are used this normalised DCF score takes the form

$$DCF^{\mathtt{norm}} = P^{\mathtt{miss}} + 9.9P^{\mathtt{false-alarm}} \tag{4.76}$$

A threshold independent version of this score known as minDCF can also be used. Here, minDCF is the minimum DCF obtained a-posteriori by adjusting the decision threshold.

4.7.3 Receiver operating characteristics graphs

The EER gives the error rate at a single threshold value. It is often more informative to observe the trade-off between miss and false alarms over a range of operating thresholds. Receiver Operating Characteristics (ROC) curves [193] are a method of graphically comparing systems over the entire operating range. In an ROC curve the hit probability (equal to 1 - miss probability) is plotted against the false-alarm probability as the threshold is adjusted. An example ROC curve is given in figure 4.6. There are a number of issues related to ROC curves that have led to them being superseded by alternative graphical techniques. When comparing two systems, the area under the graph provides a better measure of relative performance than the absolute position of the line. This makes them hard to interpret. A second drawback to the ROC graph is that in practice most systems occupy the top left corner of the graph making it hard to distinguish systems performing at a similar level.

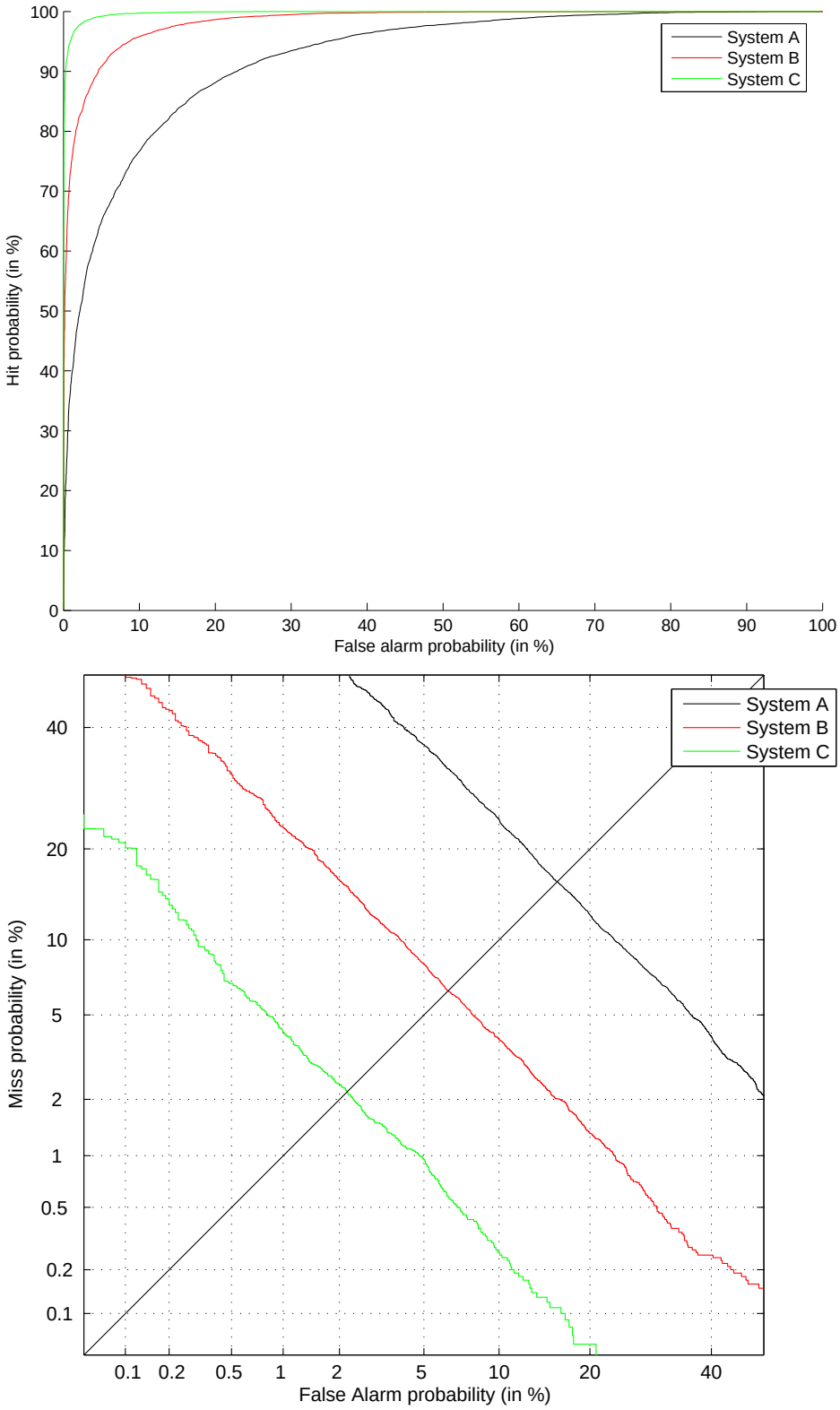


Figure 4.6: a) ROC and b) DET curves for three different SV systems. ROC curves plot the hit probability (equal to 1 - miss probability) against false alarm probability as the operating threshold is adjusted. This allows a threshold-independent comparison between systems to be displayed. Similarly, DET curves plot miss probabilities against false alarm probabilities. The logarithmic scale associated with a DET curve ensures that plots of system performance appear close to linear making the graph more readable.

4.7.4 Detection error threshold graphs

DET curves [146] are a recently developed alternative to ROC curves. For each system, miss probabilities are plotted against false alarm probabilities over a range of operating thresholds. An example DET curve is shown in figure 4.6. An advantage to using DET curves compared to ROC curves is that system performance is plotted on a logarithmic scale. For SV systems where the true and imposter scores are approximately Gaussian distributed, the line associated with each system will be approximately linear. This makes it easier to effectively compare system performance. The EER associated with each system is indicated by the point on the line where the miss and false alarm probabilities are equal.

4.7.5 Significance testing

When comparing SV systems it is useful to be able to determine whether the results obtained are statistically significant. One significance measure suitable for comparing SV systems evaluated on the same test set is McNemar's test [73]. For two systems this test assesses the probability of the null hypothesis, that both systems perform at the same level given the observed results. For a test set consisting of N trials, McNemar's test compares the unique errors made by each system. Here N_{10} is the number of trials misclassified by the first system but correctly classified by the second system. Similarly, N_{01} is the number of unique errors made by the second system. The set of trials that are classified either correctly or incorrectly by both systems is not used since these trials contain no useful information for discriminating between the systems. If the null hypothesis is true, then N_{10} should be binomially distributed according to $\mathcal{B}(k, 1/2)$ where $k = N_{10} + N_{01}$. The null hypothesis may be tested by applying a two-tailed test to the observation of a random variable M drawn from $\mathcal{B}(k, 1/2)$. The null hypothesis may be rejected when the P-value, $P^{\texttt{null}}$, is less that the significance level, where $P^{\texttt{null}}$ is given by

$$P^{\texttt{null}} = \begin{cases} 2P(N_{10} \leq M \leq k) & \text{when } N_{10} > k/2 \\ 2P(0 \leq M \leq N_{10}) & \text{when } N_{10} < k/2 \\ 1.0 & \text{when } N_{10} = k/2 \end{cases} \tag{4.77}$$

These probabilities may be computed explicitly by

$$P^{\texttt{null}} = \begin{cases} 2\sum_{m=N_{10}}^{k} \binom{k}{m} \left(\frac{1}{2}\right)^k & \text{when } N_{10} > k/2 \\ 2\sum_{m=0}^{N_{10}} \binom{k}{m} \left(\frac{1}{2}\right)^k & \text{when } N_{10} < k/2 \end{cases} \tag{4.78}$$

For large values of k ($k > 50$), explicitly computing these probabilities is expensive. However under these conditions the actual binomial probability may be approximated using a normal distribution. If the null hypothesis is true, the expected value of N_{10} is $k/2$ and the variance of N_{10} is $k/4$. If Z is a random variable, distributed according to $\mathcal{N}(0, 1)$, $P^{\texttt{NULL}} = 2P(Z \geq z)$ where z is given by

$$z = \frac{|N_{10} - K/2| - 1/2}{\sqrt{K/4}} \tag{4.79}$$

Equation 4.79 has the form of a χ^2 test [159] that includes a correction factor [212] for continuity. The significance measure provided by McNemar's test is associated with a particular point on the operating range of the SV systems and will not apply if the thresholds are varied. For systems that do not have explicitly defined operating thresholds, one option is to evaluate McNemar's test at the thresholds associated with either the EER or minDCF metric.

4.8 Summary

This chapter has described how generative and discriminative classification schemes may be applied to construct a state-of-the-art speaker recognition system. Three approaches for scoring test utterances were described, given a particular identity claim. The first approach uses the log-likelihood ratio between a speaker dependent model and a background model representing all speakers. An adapted version of this scheme, that uses factor analysis to model session and channel variability, was also discussed. Lastly an SV framework based on discriminative classifiers was introduced. Here dynamic kernels, introduced in chapter 3, are applied to convert each utterance into a fixed-dimensional representation, suitable for classification using an SVM. Front end processing schemes based on short-term cepstral features and long term prosodic and language features were described in this chapter. Score-normalisation techniques for reducing intersession variability and mapping all classifier scores into a consistent range were also discussed. Finally, metrics suitable for evaluating SV system performance were presented.

Chapter 5
Variational Dynamic Kernels

In chapter 3, a range of dynamic kernels were introduced that are suitable for SVM-based speaker verification. Several of these, including the GMM-supervector [28] and the non-linear GMM-supervector kernels [45], are directly motivated from the Kullback-Leibler (KL) divergence between two distributions. For text-independent SV, GMM distributions are commonly used. In this case there exists no closed form solution of the KL divergence and a suitable approximation, typically the matched-pair bound, is used instead. However, the use of this approximation requires that all GMM distributions contain the same number of components and that components with the same index are coordinated. In practice, this means that all distributions must be adapted from a single background distribution. This restriction may limit the performance of a speaker verification system.

Recently, alternative approximations to the KL divergence between GMMs have been derived, based on introducing additional variational distributions over Gaussian components. In this chapter, new forms of dynamic kernel are proposed based on two of these approximations, the variational upper bound [87, 217] and the variational approximation [87]. Unlike the matched-pair bound, for the variational approximations discussed in this chapter there is no requirement that all distributions contain the same number of components or that components are coordinated between distributions. Thus, variational approximations may be used to motivate dynamic kernels that do not contain the same restrictions on model structure as the dynamic kernels in Chapter 3.

In the next section the use of the KL divergence to motivate dynamic kernels is reviewed. The following section then introduces two approximations to the KL divergence based on

variational techniques. Finally, new forms of dynamic kernel are derived based on these variational approximations and the properties of these kernels discussed.

5.1 Distributional kernels for speaker verification

Distributional kernel functions were introduced in chapter 3. In this section, the use of distributional kernels for sequence classification is briefly reviewed. These kernels operate very differently from continuous-observation kernels such as the Fisher kernel [90] or the GLDS kernel [24]. Instead of evaluating a kernel function using two utterance $\boldsymbol{O}_i$ and $\boldsymbol{O}_j$ directly, a distinct distribution f_i and f_j is trained to model each utterance. For an utterance, $\boldsymbol{O}_i$, the parameters $\boldsymbol{\lambda}_i$ of distribution f_i are typically selected to maximise the likelihood of $\boldsymbol{O}_i$. Thus

$$\boldsymbol{\lambda}_i^{\texttt{ML}} = \underset{\boldsymbol{\lambda}}{\text{argmax}}\{\log p(\boldsymbol{O}_i; \boldsymbol{\lambda})\} \tag{5.1}$$

For speaker verification, there is normally too little speech available to yield robust ML estimates for $\boldsymbol{\lambda}_i$. Instead, $\boldsymbol{\lambda}_i$ is normally obtained using MAP adaptation, described in section 2.1.4.1.

$$\boldsymbol{\lambda}_i^{\texttt{MAP}} = \underset{\boldsymbol{\lambda}}{\text{argmax}}\{\log p(\boldsymbol{O}_i; \boldsymbol{\lambda}) + \log p(\boldsymbol{\lambda})\} \tag{5.2}$$

where $\log p(\boldsymbol{\lambda})$ defines a prior over the model parameters. The density function associated with the prior is typically defined using the parameters of the distribution $f^{\texttt{UBM}}$ associated with the UBM. A distributional kernel $K(\boldsymbol{O}_i, \boldsymbol{O}_j)$ can then be defined as a function of f_i and f_j allowing an SVM-based speaker verification system to be implemented as described in chapter 4. The process of applying distributional kernels to SV is shown in figure 5.1.

In chapter 3 dynamic kernels were defined using several different functions over distributions. However the most commonly used forms of distributional kernels, such as the GMM-supervector [28] and the non-linear GMM-supervector [45] kernels, use the Kullback-Leibler divergence [115] to define a suitable function between distributions. The KL divergence was introduced in section 3.3.1 and defines the relative-entropy between two distributions. For distributions f_i and f_j, the divergence is defined by

$$KL(f_i||f_j) = \int f_i(\boldsymbol{o}) \log \frac{f_i(\boldsymbol{o})}{f_j(\boldsymbol{o})}\, d\boldsymbol{o} \tag{5.3}$$

For Gaussian distributions $\tilde{f}_i(\boldsymbol{o}) = \mathcal{N}(\boldsymbol{o}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ and $\tilde{f}_j(\boldsymbol{o}) = \mathcal{N}(\boldsymbol{o}; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$ a closed form solution for the divergence is given by equation 5.4.

$$KL(\tilde{f}_i||\tilde{f}_j) = \frac{1}{2}\left[\log \frac{|\boldsymbol{\Sigma}_j|}{|\boldsymbol{\Sigma}_i|} + \texttt{Tr}[{\boldsymbol{\Sigma}_j}^{-1}\boldsymbol{\Sigma}_i] - D + (\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)^{\texttt{T}}{\boldsymbol{\Sigma}_i}^{-1}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)\right] \tag{5.4}$$

where $\texttt{Tr}[\cdot]$ defines the trace of a matrix and D is the dimensionality of $\boldsymbol{o}$. For state-of-the art text-independent speaker verification systems it is common to model the distribution of speech utterances using Gaussian mixture models. Here the distributions are defined by

$$\begin{aligned} f_i(\boldsymbol{o}) &= \sum_{n=1}^{N} c_{in}\mathcal{N}(\boldsymbol{o}; \boldsymbol{\mu}_{in}, \boldsymbol{\Sigma}_{in}) \\ f_j(\boldsymbol{o}) &= \sum_{m=1}^{M} c_{jm}\mathcal{N}(\boldsymbol{o}; \boldsymbol{\mu}_{jm}, \boldsymbol{\Sigma}_{jm}) \end{aligned} \tag{5.5}$$

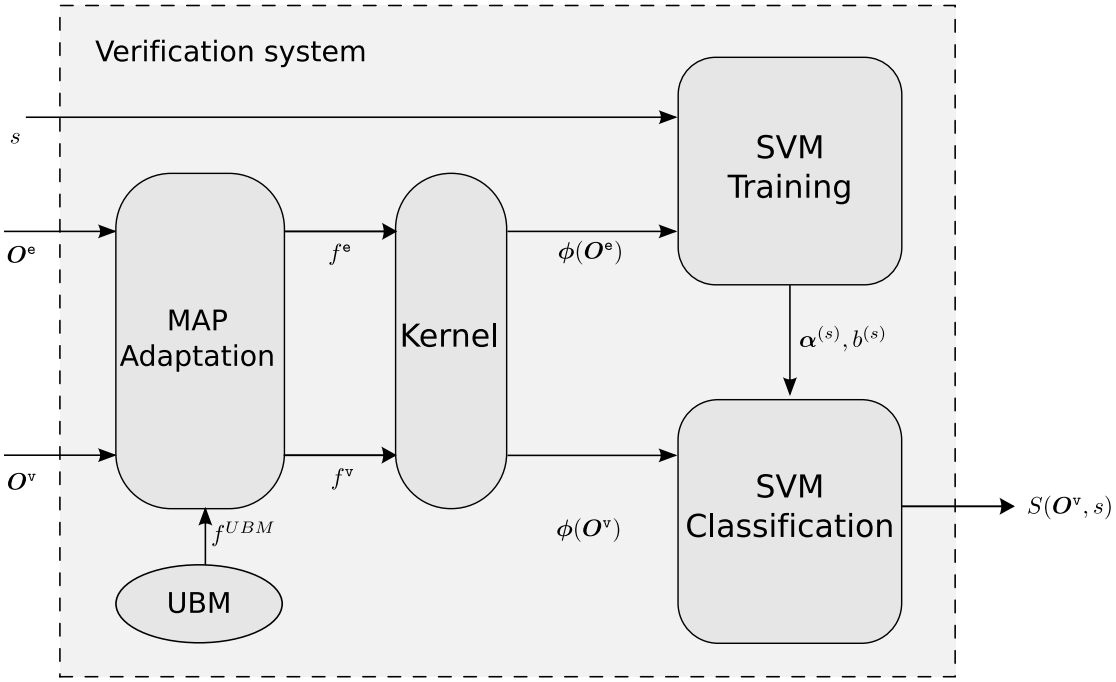


Figure 5.1: Speaker verification using distributional kernels. For distributional kernels, a distinct distribution f_i is trained to model each utterance $\boldsymbol{O}_i$ by MAP adapting the UBM.

Unfortunately no such closed form expression exists for the divergence between two GMMs. Instead the divergence must be estimated. This may be achieved either by using sampling approaches, such as Monte Carlo sampling [87], or by finding a closed form expression that approximates the true KL divergence. Although sampling is the only method that can be used to obtain the KL divergence to an arbitrary degree of accuracy, calculating the divergence via sampling approaches is usually far slower than evaluating a closed form expression. Since SV typically requires a large number of kernel evaluations, this chapter focuses on the latter approach.

5.2 KL divergence for GMMs

Both the GMM-supervector kernel and the non-linear GMM-supervector kernel are derived using the matched-pair bound [53] on the KL divergence. This approximation requires that both GMMs contain the same number of Gaussian components.

$$\begin{aligned} f_i(\boldsymbol{o}) &= \sum_{m=1}^{M} c_{im}\mathcal{N}(\boldsymbol{o};\boldsymbol{\mu}_{im},\boldsymbol{\Sigma}_{im}) \\ f_j(\boldsymbol{o}) &= \sum_{m=1}^{M} c_{jm}\mathcal{N}(\boldsymbol{o};\boldsymbol{\mu}_{jm},\boldsymbol{\Sigma}_{jm}) \end{aligned} \tag{5.6}$$

When f_i and f_j are distributed according to equation 5.6 the log-sum inequality may be applied to obtain an upper bound to the true divergence. This matched-pair bound is defined as follows.

$$KL^{\text{MP}}(f_i||f_j) \leq \sum_{m=1}^{M} c_{im}\left[\log\frac{c_{im}}{c_{jm}} + KL(f_{im}||f_{jm}))\right] \tag{5.7}$$

where $KL(f_{in}||f_{jm})$ indicates the divergence between component n of f_i and component m of f_j. This may be calculated explicitly using equation 5.4. This upper bound is dependent on the ordering of the Gaussian components. Permuting the components of one distribution will affect the obtained bound. Thus the matched-pair bound is only suitable when there is a clearly defined coordination between pairs of components from the two distributions. This may be the case when both are adapted from the same background distribution but will not be true generally. In the following subsections, two variational approximations are described that do not suffer from these restrictions.

5.2.1 The variational approximation

A variational approximation to the KL divergence between two Gaussian mixture models was derived in [87]. Here an approximation to the KL divergence is obtained by introducing an additional variational distribution over components. The parameters of this distribution are then optimised to tighten the approximation. Unlike the matched-pair bound, this approximation does not yield an upper bound to the true KL divergence. However it has been shown experimentally to give a closer approximation to the true KL divergence. A further advantage of this approach is that unlike the matched-pair bound, the variational approximation does

not require any restrictions on the structure of the distributions. Thus, it may be applied to the GMMs defined by equation 5.5.

The variational approximation is derived by initially decomposing the KL divergence into the difference between two expected values

$$KL(f_i||f_j) = \int f_i(\boldsymbol{o}) \log f_i(\boldsymbol{o})\ d\boldsymbol{o} - \int f_i(\boldsymbol{o}) \log f_j(\boldsymbol{o})\ d\boldsymbol{o} \tag{5.8}$$

A lower bound can then be derived separately for each of the terms in 5.8. Starting with the second term, a bound may be obtained by introducing a discrete variational distribution $q_{m|n}$, such that $q_{m|n} > 0$ and $\sum_{m=1}^{M} q_{m|n} = 1$. For clarity of notation $f_{in}(\boldsymbol{o})$ will be used to indicate likelihood of $\boldsymbol{o}$ given the Gaussian distribution associated with component n of distribution f_i.

$$\begin{aligned}\int f_i(\boldsymbol{o}) \log f_j(\boldsymbol{o}) d\boldsymbol{o} &= \int \sum_{n=1}^{N} c_{in} f_{in}(\boldsymbol{o}) \log \left(\sum_{m=1}^{M} q_{m|n} \frac{c_{jm} f_{jm}(\boldsymbol{o})}{q_{m|n}} \right) d\boldsymbol{o} \\ &\geq \int \sum_{n=1}^{N} c_{in} f_{in}(\boldsymbol{o}) \sum_{m=1}^{M} q_{m|n} \log \left(\frac{c_{jm} f_{jm}(\boldsymbol{o})}{q_{m|n}} \right) d\boldsymbol{o}\end{aligned} \tag{5.9}$$

This lower bound is tightened when the expression is maximised with respect to $q_{m|n}$. The optimal value $q^*_{m|n}$ is given by

$$q^*_{m|n} = \frac{c_{jm} e^{-KL(f_{in}||f_{jm})}}{\sum_{m'=1}^{M} c_{jm'} e^{-KL(f_{in}||f_{jm'})}} \tag{5.10}$$

Similarly a lower bound may be obtained to the first term in equation 5.8 by introducing a discrete variational distribution $v_{n'|n}$ such that $v_{n'|n} > 0$ and $\sum_{n'=1}^{N} v_{n'|n} = 1$.

$$\int f_i(\boldsymbol{o}) \log f_i(\boldsymbol{o}) d\boldsymbol{o} \geq \int \sum_{n=1}^{N} c_{in} f_{in}(\boldsymbol{o}) \sum_{n'=1}^{N} v_{n'|n} \log \frac{c_{in'} f_{in'}(\boldsymbol{o})}{v_{n'|n}} d\boldsymbol{o} \tag{5.11}$$

This bound is tightest when the expression is maximised with respect to $v_{n'|n}$. The optimal value $v^*_{n'|n}$ is given by

$$v^*_{n'|n} = \frac{c_{in'} e^{-KL(f_{in}||f_{in'})}}{\sum_{n''=1}^{N} c_{in''} e^{-KL(f_{in}||f_{in''})}} \tag{5.12}$$

Substituting $q^*_{m|n}$ into equation 5.9 and $v^*_{n'|n}$ into equation 5.11 and taking the difference between the two bounds yields a variational approximation to the KL divergence.

$$KL^{\texttt{VAR}}(f_i||f_j) \approx \sum_{n=1}^{N} c_{in} \log \frac{\sum_{n'=1}^{N} c_{in'} e^{-KL(f_{in}||f_{in'})}}{\sum_{m=1}^{M} c_{jm} e^{-KL(f_{in}||f_{jm})}} \tag{5.13}$$

As the difference of two lower bounds, equation 5.13 is not itself a strict bound on the KL divergence. However, in practice it forms a close approximation. This variational approximation is related to the matched-pair bound. For the case where f_i and f_j have equal numbers of components and when the variational distributions $q_{m|n}$ and $v_{n'|n}$are only non-zero when $m = n = n'$, the approximation becomes an upper bound and is equivalent to the form given in equation 5.7.

5.2.2 The variational upper bound

A variational upper bound was derived independently in [217] and [87]. Like the variational approximation, two discrete variational distributions $v_{n|m} \geq 0$ and $q_{m|n} \geq 0$ are introduced. Here, these distributions satisfy the following constraints.

$$\sum_{m=1}^{M} q_{m|n} = c_{in} \tag{5.14}$$

$$\sum_{n=1}^{N} v_{n|m} = c_{jm} \tag{5.15}$$

Using the log-sum inequality the following bound may be defined

$$KL(f_i||f_j) = -\int f_i(\boldsymbol{o}) \log \sum_{n=1}^{N}\sum_{m=1}^{M} \frac{q_{m|n} f_{in}(\boldsymbol{o})}{f_i(\boldsymbol{o})} \frac{v_{n|m} f_{jm}(\boldsymbol{o})}{q_{m|n} f_{in}(\boldsymbol{o})} d\boldsymbol{o} \tag{5.16}$$

$$\leq -\int f_i(\boldsymbol{o}) \sum_{n=1}^{N}\sum_{m=1}^{M} \frac{q_{m|n} f_{in}(\boldsymbol{o})}{f_i(\boldsymbol{o})} \log \frac{v_{n|m} f_{jm}(\boldsymbol{o})}{q_{m|n} f_{in}(\boldsymbol{o})} d\boldsymbol{o} \tag{5.17}$$

$$KL^{\text{UP}}(f_i||f_j) = \sum_{n=1}^{N}\sum_{m=1}^{M} q_{m|n} \left[\log \frac{q_{m|n}}{v_{n|m}} + KL(f_{in}||f_{jm})\right] \tag{5.18}$$

This bound is tightest when q and v are selected to minimise equation 5.18. Unlike the variational approximation there is no closed form expression for the optimal q and v. However, by fixing one set of variational parameters and optimising the other the following update rules are obtained.

$$v_{n|m}^{(k+1)} = \frac{c_{jm} q_{m|n}^{(k)}}{\sum_{n'=1}^{N} q_{m|n'}^{(k)}} \tag{5.19}$$

$$q_{m|n}^{(k+1)} = \frac{c_{in} v_{n|m}^{(k)} e^{-KL(f_{in}||f_{jm})}}{\sum_{m'=1}^{M} v_{n|m'}^{(k)} e^{-KL(f_{in}||f_{jm'})}} \tag{5.20}$$

By iteratively reapplying equations 5.19 and 5.20 the upper bound will be tightened. In [87] the variational parameters were initialised to $v_{n|m} = q_{m|n} = c_{jm}c_{in}$ since any parameters that are set to zero will be unchanged after each iteration.

Like the variational approximation, the variational upper bound is related to the matched-pair approximation. When both distributions contain the same number of components and $q_{m|n} = c_{in}$ and $v_{n|m} = c_{jm}$ where $m = n$ otherwise $q_{m|n} = v_{n|m} = 0$ the variational upper bound will have the form

$$KL(f_i||f_j) = \sum_{m=1}^{M} c_{im} \left[\log \frac{c_{im}}{c_{jm}} + KL(f_{in}||f_{jm})\right] \tag{5.21}$$

This is identical to the form of the matched-pair bound in equation 5.7. The likelihood of this occurring in practice is related to the dimension D of the distribution. When D is high $q_{m|n}$ and $v_{n|m}$ are more likely to be sparse.

5.3 Variational dynamic kernels

In chapter 3 a number of distributional kernels were described that were motivated directly from an approximation to the KL divergence. These included the GMM-supervector kernel and the non-linear GMM-supervector kernel. Typically the KL divergence is not used directly, but instead is used to derive a kernel function with similar characteristics. Since no closed form solution exists for the KL divergence between two GMMs, it is necessary to make use of an approximation. The matched-pair bound was used for the kernels described in chapter 3, however, as discussed in section 5.2, this approximation has a number of disadvantages: it may not be particularly accurate and restricts the structure of distributions that may be used.

In this section, new forms of dynamic kernel are proposed, motivated from the two variational approximations introduced in sections 5.2.1 and 5.2.2. These variational kernels do not suffer the same restrictions and may be applied using GMMs adapted from different background models and consisting of varying numbers of components.

5.3.1 Symmetric KL divergence

The KL divergence between two distributions is not suitable for use directly as a kernel function. As described in chapter 2, to be a valid kernel a function must be symmetric and the associated Gram matrix must be positive semi-definite. This is not the case for the KL divergence, which is asymmetric. Additionally, the KL divergence equals zero if and only if $f_i = f_j$, otherwise it is positive. These properties are typical of a distance metric rather than an inner product. Thus the KL divergence must be modified in some way before it may be used as a kernel function. A symmetric version of the KL divergence may be defined as follows

$$KL^{\mathtt{sym}}(f_i||f_j) = KL(f_i||f_j) + KL(f_j||f_i) \tag{5.22}$$

When $KL(f_i||f_j)$ is approximated by the variational upper bound, the following expression is obtained.

$$\begin{aligned} KL(f_i||f_j) \quad = \quad & KL(q^{\mathtt{L}}||v^{\mathtt{L}}) + KL(v^{\mathtt{R}}||q^{\mathtt{R}}) \\ & + \sum_{n=1}^{N}\sum_{m=1}^{M} q^{\mathtt{L}}_{m|n} KL(f_{in}||f_{jm}) + v^{\mathtt{R}}_{n|m} KL(f_{jm}||f_{in}) \end{aligned} \tag{5.23}$$

where $q^{\mathtt{L}}$ and $v^{\mathtt{L}}$ indicate the variational distributions associated with the left asymmetric term and $q^{\mathtt{R}}$ and $v^{\mathtt{R}}$ indicate the distributions associated with the right asymmetric term. Here $KL(q||v)$ defines the KL divergence between the discrete variational distributions.

$$KL(q||v) = \sum_{n=1}^{N}\sum_{m=1}^{M} q_{m|n} \log \frac{q_{m|n}}{v_{n|m}} \tag{5.24}$$

Evaluating equation 5.23 requires optimising four sets of variational parameters. (This process is implicit when the variational approximation is used.) An alternative approach is to obtain a variational upper bound to the true symmetric KL divergence. In this case only two sets of variational parameters must be optimised. However, this approach yields a weaker bound and hence is not considered here.

5.3.2 Variational kernel functions

Although the divergence defined by equation 5.22 is symmetric, it does not yield a valid kernel function as the corresponding Gram matrix is not positive semi-definite. It is therefore common to alter the function prior to use so that it behaves more like an inner product. One approach, introduced in section 3.3.1 for the GMM-supervector kernel, is to make use of the polarisation identity. This defines a relationship between a distance and a kernel function in a particular space. Given a distance metric $D(f_i, f_j)$ defined between two distributions, an appropriate kernel function $K^{\mathtt{pol}}(\boldsymbol{O}_i, \boldsymbol{O}_i)$ is one that satisfies the following relationship.

$$D(f_i, f_j)^2 = K^{\mathtt{pol}}(\boldsymbol{O}_i, \boldsymbol{O}_i) - 2K^{\mathtt{pol}}(\boldsymbol{O}_i, \boldsymbol{O}_j) + K^{\mathtt{pol}}(\boldsymbol{O}_j, \boldsymbol{O}_j) \tag{5.25}$$

The GMM-supervector kernel was then defined using the matched-pair bound by using the following distance metric.

$$D(f_i, f_j)^2 = KL(f_i||f_j) + KL(f_j||f_i) \tag{5.26}$$

Given a kernel function, the polarisation identity defines a unique distance metric. However the inverse is not true. Unfortunately, due to the form of the variational approximations it is difficult to obtain a suitable kernel using the polarisation identity. For example, even when component variance and prior parameters are tied over all models, both approximations still require evaluating the KL divergence between Gaussian components with different variances yielding a more complex distance measure.

In this work, an alternative approach is used, based on exponentiation. This makes use of the following relation.

$$K^{\mathtt{exp}}(\boldsymbol{O}_i, \boldsymbol{O}_j) \quad = \quad \exp\left[-\alpha D(f_i, f_j)^2\right] \tag{5.27}$$

where α is a positive real scalar. By applying the operations described in section 2.2.6, this can be shown to yield a valid kernel function. This is the same approach used to derive the non-linear GMM-supervector, described in section 3.3.2.

When the distance metric in equation 5.26 is used, the following kernel can be defined using the variational approximation.

$$K^{\mathtt{VAR}}(\boldsymbol{O}_i, \boldsymbol{O}_j) = \exp\left(-\alpha\left[KL^{\mathtt{VAR}}(f_i||f_j) + KL^{\mathtt{VAR}}(f_j||f_i)\right]\right) \tag{5.28}$$

Similarly, a kernel may also be derived using the variational upper bound.

$$K^{\mathtt{UP}}(\boldsymbol{O}_i, \boldsymbol{O}_j) = \exp\left(-\alpha\left[KL^{\mathtt{UP}}(f_i||f_j) + KL^{\mathtt{UP}}(f_j||f_i)\right]\right) \tag{5.29}$$

When both GMMs consist of M components and the variational parameters between components n and m are non-zero only when $n = m$, the kernels obtained will be identical to the non-linear GMM-supervector kernel.

5.3.3 Comparison to non-variational kernels

Unlike the GMM-supervector or non-linear GMM-supervector kernels, for the variational kernels there is no requirement that all GMMs have the same structure or are adapted from the same background model. There are a number of situations where this is useful. If the duration of utterances vary greatly within the dataset, performance gains may be obtained by allowing the number of components per GMM to vary. Hence an utterance-dependent trade-off could be made between increasing model flexibility and avoiding overfitting the data. Alternatively, when utterances come from speakers of varying genders or dialects, or are recorded under a range of different noise-conditions it may be advantageous to adapt each utterance from a background model that more closely resembles the characteristics of the utterance. This approach may also be combined with speaker-clustering schemes to obtain more accurate background models.

A second advantage of using variational kernels that they do not require that components are coordinated across distributions. For kernels such as the GMM-supervector kernel that use the matched-pair bound, any experimental conditions that weaken the coordination between components are likely to also degrade performance. This includes performing multiple iterations of adaptation or applying MAP with low values of $\tau^{\texttt{map}}$. In contrast, variational kernels are likely to be robust to these conditions.

5.4 Summary

In this chapter new forms of dynamic kernel were proposed, derived from two different variational approximations to the KL divergence between GMMs. Unlike standard forms of dynamic kernel, such as the GMM-supervector kernel, these variational kernels do not require all distributions to be adapted from a single background distribution, allowing more complex training schemes to be used. For example, multiple background distributions may be used to reflect clusters of dialect or gender. Alternatively, a range of model sizes may be used if the training set contains large variations in utterance duration, potentially yielding gains over standard approaches.

Chapter 6
Dynamic Kernel Combination

There has been considerable interest and success in improving the performance of speaker verification systems by fusing the output scores from multiple classifiers. For SVM classifiers, a known alternative strategy is to combine systems at the dynamic kernel level. This involves finding a suitable weighting for each kernel, known as multiple kernel learning (MKL). Recently, an efficient maximum-margin scheme for MKL was proposed in [168]. This scheme allows kernel weights to be trained without the need for an auxiliary dataset, in contrast to standard score-fusion techniques. In this chapter, a number of modifications are proposed to allow this scheme to be successfully applied to speaker-verification[1]. The standard scheme has a known tendency towards sparse weightings, which may not be optimal for SV. Here a regularisation term is proposed, allowing the appropriate level of sparsity to be selected. Cross-speaker tying of kernel weights is also proposed to improve the robustness of the parameter estimates.

Dynamic kernel combination will only provide gains if the features expressed by the kernels are complementary. In the second half of this chapter two general categories of dynamic kernel are defined. These are *parametric kernels*, where the feature space consists of parameters from an utterance-dependent generative model, and *derivative kernels*, where the derivatives of the log-likelihood with respect to parameters of a generative model are used. It is then shown that many of the dynamic kernels introduced in chapter 3 can be placed into one of these two

[1] A similar approach for learning kernel weights for SV was independently proposed in [49]. The main differences between that approach and the one proposed in this chapter are the form of the objective function used and the use of cross-speaker tying of kernel weights in this work.

classes. Finally, the attributes of these two classes of kernel are contrasted and the conditions under which they yield identical feature spaces are described. By avoiding these conditions, the combination of derivative and parametric kernels may lead to gains.

6.1 Classifier combination

In recent SV evaluations there has been a focus on combining multiple classifiers to improve overall performance. For SVM classifiers two general combination strategies are available, score-fusion and kernel combination. These are discussed in the following sections.

6.1.1 Score-fusion

Score-fusion is a standard technique used to combine classifiers that has been effectively used to improve performance in a number of SV systems, such as [22, 30, 126, 154]. Score-fusion is applied as follows: given a training set, K independent classifiers are trained. During verification, a score is then obtained from each classifier which are then combined using a *post-classifier* to provide a final output. This process is illustrated in figure 6.1.

The fused output score $S(\boldsymbol{O}^{\mathtt{v}}, s; \boldsymbol{\beta})$ associated with a verification utterance $\boldsymbol{O}^{\mathtt{v}}$ is typically a weighted sum of the individual classifier scores.

$$S(\boldsymbol{O}^{\mathtt{v}}, s; \boldsymbol{\beta}) = \sum_{k=1}^{K} \beta_k S_k(\boldsymbol{O}^{\mathtt{v}}, s) \tag{6.1}$$

where $S_k(\boldsymbol{O}^{\mathtt{v}}, s)$ is the score output for utterance $\boldsymbol{O}^{\mathtt{v}}$ by classifier k and β_k is the weight associated with classifier k. When only SVM classifiers are used and $K_k(\boldsymbol{O}_i, \boldsymbol{O}_j)$ is the kernel associated with classifier k the decision may be expressed as

$$S(\boldsymbol{O}^{\mathtt{v}}, s) = \sum_{k=1}^{K} \beta_k \left[\sum_{i=1}^{N} \alpha_{ik}^{(s)} y_i^{(s)} K_k(\boldsymbol{O}_i, \boldsymbol{O}^{\mathtt{v}}) + b_k^{(s)} \right] \tag{6.2}$$

where $\boldsymbol{\alpha}_k^{(s)}$ and $b_k^{(s)}$ are the dual variables and bias for classifier k trained to discriminate speaker s. Score-fusion is most effective when the classifiers used are complementary. This is usually achieved by combining a diverse range of classifiers.

6.1.1.1 Logistic regression

Various forms of post-classifier may be used to train an appropriate score-weighting. For example, an additional SVM may be trained using the individual scores obtained from classifying an auxiliary dataset. A commonly used post-classifier for score-fusion is logistic regression [158]. Logistic regression is normally used to train a set of scores to approximate a *log-odds ratio* between two classes. For the case of speaker-verification the prior likelihood of a particular test utterance belonging to a true speaker $P(\omega^{\mathtt{tar}})$ is often known. As this may differ from the training distribution, the fused-scores are instead usually trained to resemble a *log-likelihood ratio*.

$$S(\boldsymbol{O}, s; \boldsymbol{\beta}) \approx \log \frac{p(S(\boldsymbol{O}, s; \boldsymbol{\beta}); \mathcal{H}^{\mathtt{tar}})}{p(S(\boldsymbol{O}, s; \boldsymbol{\beta}); \mathcal{H}^{\mathtt{non}})} \tag{6.3}$$

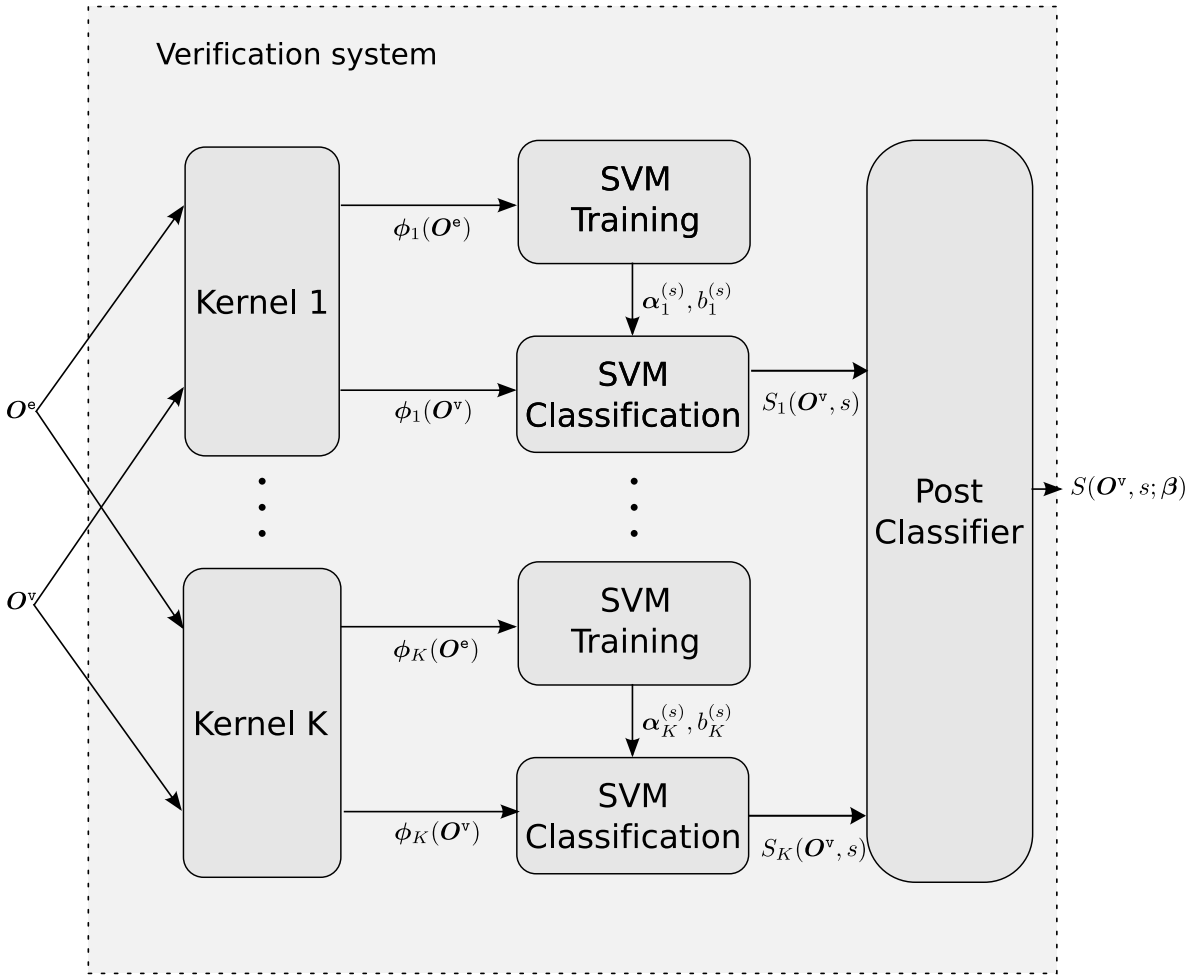


Figure 6.1: Score-fusion: a distinct classifier is trained using each kernel. The final score is then determined by combining the classifier outputs using a post-classifier.

where $\mathcal{H}^{\mathtt{tar}}$ is the hypothesis that the score was emitted by the target speaker and $\mathcal{H}^{\mathtt{non}}$ is the hypothesis that the score was actually emitted by an imposter. A suitable set of score-weights may be obtained by maximising the following optimisation function [22].

$$\begin{aligned}\mathcal{F}^{\mathtt{LR}}(\boldsymbol{\beta}, P(\omega^{\mathtt{tar}})) \quad = \quad & \frac{P(\omega^{\mathtt{tar}})}{N^{\mathtt{tar}}} \sum_{i=1}^{N^{\mathtt{tar}}} \log(1 + e^{-S(\boldsymbol{O}_i, s; \boldsymbol{\beta}) - \mathtt{logitP}(\omega^{\mathtt{tar}})}) \\ & + \frac{1 - P(\omega^{\mathtt{tar}})}{N^{\mathtt{non}}} \sum_{i=1}^{N^{\mathtt{non}}} \log(1 + e^{S(\boldsymbol{O}_i, s; \boldsymbol{\beta}) + \mathtt{logitP}(\omega^{\mathtt{tar}})})\end{aligned} \tag{6.4}$$

where $N^{\mathtt{tar}}$ and $N^{\mathtt{non}}$ are the number of target and imposter trials in the training set respectively. $\mathtt{logitP}(\omega^{\mathtt{tar}})$ is the prior log odds given by

$$\mathtt{logitP}(\omega^{\mathtt{tar}}) = \log \frac{\mathtt{P}(\omega^{\mathtt{tar}})}{1 - \mathtt{P}(\omega^{\mathtt{tar}})} + \log \frac{\mathtt{C}^{\mathtt{miss}}}{\mathtt{C}^{\mathtt{false-alarm}}} \tag{6.5}$$

where $C^{\mathtt{miss}}$ and $C^{\mathtt{false-alarm}}$ are the costs associated with a miss and false-alarm respectively. $\mathcal{F}^{\mathtt{LR}}(\boldsymbol{\beta}, P(\omega^{\mathtt{tar}}))$ can be efficiently minimised using projected-gradient schemes. A benefit of logistic regression compared to other schemes is that the scores obtained are well-calibrated. For a given set of costs an optimal threshold $b^{\mathtt{LR}}$ may be determined. This is simply the negative of equation 6.5.

$$b^{\mathtt{LR}} = -\log \frac{P(\omega^{\mathtt{tar}})}{1 - P(\omega^{\mathtt{tar}})} - \log \frac{C^{\mathtt{miss}}}{C^{\mathtt{false-alarm}}} \tag{6.6}$$

Thus, when $\boldsymbol{\beta}$ is obtained using logistic regression, the optimal classifier for a particular set of costs, $C^{\mathtt{miss}}$ $C^{\mathtt{false-alarm}}$, is given by equation 6.7.

$$S(\boldsymbol{O}^{\mathtt{v}}, s; \boldsymbol{\beta}) - b^{\mathtt{LR}} \quad \begin{matrix} \text{accept} \\ > \\ < \\ \text{reject} \end{matrix} \quad 0 \tag{6.7}$$

6.1.2 Kernel combination

For kernel-based classifiers, an alternative combination scheme is to to combine systems at the kernel level. Given a set of K kernels, $\{K_1(\boldsymbol{O}_i, \boldsymbol{O}_j), \ldots, K_K(\boldsymbol{O}_i, \boldsymbol{O}_j)\}$, a combined kernel may be defined as the weighted sum of the individual kernels. The combined kernel has this form.

$$K(\boldsymbol{O}_i, \boldsymbol{O}_j; \boldsymbol{\beta}) = \sum_{k=1}^{K} \beta_k K_k(\boldsymbol{O}_i, \boldsymbol{O}_j) \tag{6.8}$$

To ensure that the resultant kernel function is positive semi-definite, the weights are constrained such that $\beta_k \geq 0 \ \forall k$. By applying the operations described in section 2.2.6 this approach can be shown to yield valid kernel functions. An additional constraint, $\sum_{k=1}^{K} \beta_k = 1$,

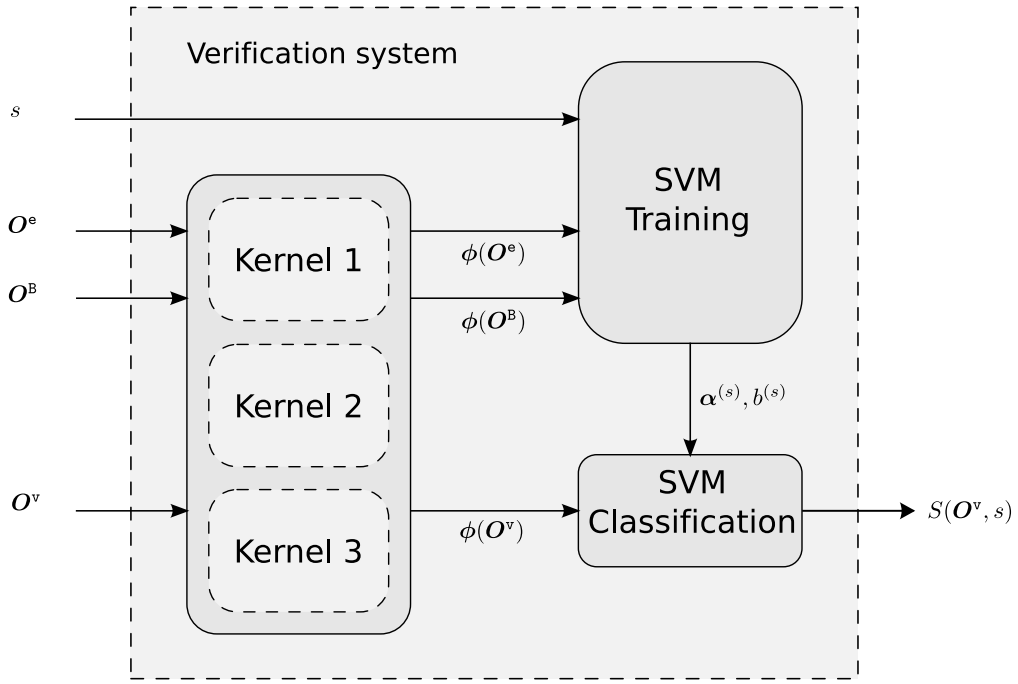


Figure 6.2: For kernel combination a single classifier is trained using a kernel formed by the weighted sum of K distinct kernels. Kernel combination using three kernels is depicted in the figure.

is also typically applied. Combining kernels using equation 6.8 is equivalent to a weighted concatenation of the associated feature spaces.

$$\boldsymbol{\phi}(\boldsymbol{O};\boldsymbol{\beta}) = \begin{bmatrix} \sqrt{\beta_1}\boldsymbol{\phi}_1(\boldsymbol{O}) \\ \vdots \\ \sqrt{\beta_K}\boldsymbol{\phi}_K(\boldsymbol{O}) \end{bmatrix} \tag{6.9}$$

A single SVM classifier is then trained using the kernel defined in equation 6.8. This process is illustrated in figure 6.2. The score $S(\boldsymbol{O}^{\mathrm{v}}, s; \boldsymbol{\beta})$ associated with an utterance $\boldsymbol{O}^{\mathrm{v}}$ is given by

$$S(\boldsymbol{O}^{\mathrm{v}}, s; \boldsymbol{\beta}) = \sum_{i=1}^{N} \alpha_i^{(s)} y_i^{(s)} \sum_{k=1}^{K} \beta_k K_k(\boldsymbol{O}_i, \boldsymbol{O}^{\mathrm{v}}) + b^{(s)} \tag{6.10}$$

Like score-fusion, kernel combination will provide gains only when complementary kernels are used. Schemes for learning kernel weights from data are discussed in section 6.2

6.1.3 Relationship between score-fusion and kernel combination

Kernel combination and score-fusion are closely related. This can be observed by comparing equations 6.2 and 6.10. Both equations have a similar form and will yield identical scores when $\boldsymbol{\alpha}_k$ and b_k are tied over all classifiers during score-fusion. In practice, $\boldsymbol{\alpha}_k$ and b_k will vary between classifiers. Since there is little comparison of these schemes in the literature, it is interesting to briefly compare the optimisation functions associated with score-fusion and kernel combination. Dropping the reference to the target speaker s, the combined optimisation for score-fusion in the unweighted case may be expressed as

$$\begin{aligned} \min \quad & \sum_{k=1}^{K} \left(\frac{1}{2}\langle \boldsymbol{w}_k, \boldsymbol{w}_k \rangle + C \sum_{i=1}^{N} \xi_{ik} \right) \\ \text{w.r.t.} \quad & \boldsymbol{w}_1, \ldots, \boldsymbol{w}_K, b_1, \ldots, b_K, \boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_K \\ \text{s.t.} \quad & y_i \left(\langle \boldsymbol{w}_k, \boldsymbol{\phi}_k(\boldsymbol{O}_i; \boldsymbol{\lambda}) \rangle + b_k \right) \geq 1 - \xi_{ik} \ \ \forall i \ \ \forall k \\ & \xi_{ik} \geq 0 \ \ \forall i \ \ \forall k \end{aligned} \tag{6.11}$$

where $\boldsymbol{\phi}_k(\boldsymbol{O}; \boldsymbol{\lambda})$ is the feature space and $\boldsymbol{\xi}_k$ is the training error associated with classifier k. For kernel combination, the combined kernel in the unweighted case is defined as

$$K(\boldsymbol{O}_i, \boldsymbol{O}_j) = \sum_{k=1}^{K} K_k(\boldsymbol{O}_i, \boldsymbol{O}_j) \tag{6.12}$$

The primal form optimisation problem that corresponds to an SVM with this kernel is given by

$$\begin{aligned} \min \quad & \frac{1}{2}\sum_{k=1}^{K}\langle \boldsymbol{w}_k, \boldsymbol{w}_k\rangle + C\sum_{i=1}^{N}\xi_i \\ \text{w.r.t.} \quad & \boldsymbol{w}_1, \ldots, \boldsymbol{w}_K, b, \boldsymbol{\xi} \\ \text{s.t.} \quad & y_i \sum_{k=1}^{K}\langle \boldsymbol{w}_k, \boldsymbol{\phi}_k(\boldsymbol{O}_i; \boldsymbol{\lambda})\rangle + b \geq 1 - \xi_i \ \ \forall i \\ & \xi_i \geq 0 \ \ \forall i \end{aligned} \tag{6.13}$$

The relationship between the optimisation functions associated with score-fusion and kernel combination can be expressed more clearly through a substitution of variables. By introducing new variables b_k and $\xi_{1k}, \ldots, \xi_{Nk}$ for each kernel k such that $b = \sum_{k=1}^{K} b_k$, $\xi_i = \sum_{k=1}^{K} \xi_{ik}$ and $\xi_{ik} \geq 0 \ \ \forall i \forall k$, the optimisation problem defined by equation 6.13 can be expressed in a similar form to equation 6.11

$$\begin{aligned} \min \quad & \sum_{k=1}^{K}\left(\frac{1}{2}\langle \boldsymbol{w}_k, \boldsymbol{w}_k\rangle + C\sum_{i=1}^{N}\xi_{ik}\right) \\ \text{w.r.t.} \quad & \boldsymbol{w}_1, \ldots, \boldsymbol{w}_K, b_1, \ldots, b_K, \boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_K \\ \text{s.t.} \quad & y_i \sum_{k=1}^{K}\left(\langle \boldsymbol{w}_k, \boldsymbol{\phi}_k(\boldsymbol{O}_i; \boldsymbol{\lambda})\rangle + b_k\right) \geq 1 - \sum_{k=1}^{K}\xi_{ik} \ \ \forall i \\ & \sum_{k=1}^{K}\xi_{ik} \geq 0 \ \ \forall i \end{aligned} \tag{6.14}$$

The optimisation functions 6.11 and 6.14 differ only in the constraints. The constraints for score-fusion are more restrictive since each example must satisfy a separate margin constraint for each kernel. For kernel combination, individual terms associated with each kernel in the constraint may be violated if on average the example lies outside the margin. One consequence of this is that kernel combination may generalise more effectively than score-fusion schemes when training errors are made by individual classifiers.

For case when each score/kernel is individually weighted, and the weights are fixed during training, the relationship between score-fusion and kernel-combination is similar. Since each classifier is trained independently, the score-fusion problem in the weighted case has the same form as equation 6.11. Similarly, by redefining b as $b = \sum_{k=1}^{K} \sqrt{\beta_k} b_k$, the weighted primal optimisation problem for kernel combination can be expressed in a form that has the same objective function as equation 6.14 and a modified margin constraint given by

$$y_i \sum_{k=1}^{K}\sqrt{\beta_k}\left(\langle \boldsymbol{w}_k, \boldsymbol{\phi}_k(\boldsymbol{O}_i; \boldsymbol{\lambda})\rangle + b_k\right) \geq 1 - \sum_{k=1}^{K}\xi_{ik} \tag{6.15}$$

Thus, the effect of the kernel weights is to adjust the relative importance of each margin term in determining whether the constraint is satisfied. However, as in the unweighted case, for score-fusion a separate margin constraint must be satisfied for each kernel.

6.2 Multiple kernel learning

Learning a suitable set of kernel weights $\{\beta_1, \ldots, \beta_K\}$ for kernel combination is known as the multiple kernel learning (MKL) problem [8]. One approach to solving the MKL problem is to select kernel weights that reduce the cross-validation error on some development dataset. This could be achieved by conducting a grid search over all possible weightings and selecting the weights that minimise the Equal Error Rate. In this thesis, this criterion is termed `minEER`. Unfortunately this approach is generally impractical for anything other than pairwise kernel combination. However, in cases where calculating the `minEER` kernel weights is feasible this metric can provide an upper bound for the gains that can be achieved using other criteria to select an appropriate set of kernel weights.

6.2.1 Maximum-margin MKL

An efficient approach to MKL was developed in [188] and extended in [168]. Here the kernel weights are incorporated into the standard SVM objective function. For a set of N utterances $\mathcal{O} = \{\boldsymbol{O}_1, \ldots, \boldsymbol{O}_N\}$ each with associated label $y_i \in \{-1, 1\}$, the optimal set of weights are those that maximise the margin.

$$\begin{aligned} \min \quad & \frac{1}{2}\sum_{k=1}^{K}\frac{1}{\beta_k}\langle \boldsymbol{w}_k, \boldsymbol{w}_k\rangle + C\sum_{i=1}^{N}\xi_i \\ \text{w.r.t.} \quad & \boldsymbol{\beta}, \boldsymbol{w}_k, \boldsymbol{b}, \boldsymbol{\xi} \\ \text{s.t.} \quad & y_i\left(\sum_{k=1}^{K}\langle \boldsymbol{w}_k, \boldsymbol{\phi}_k(\boldsymbol{O}_i;\boldsymbol{\lambda})\rangle + b\right) \geq 1-\xi_i \ \ \forall i \\ & \xi_i \geq 0 \ \ \forall i, \quad \beta_k \geq 0 \ \ \forall k, \quad \sum_{k=1}^{K}\beta_k = 1 \end{aligned} \tag{6.16}$$

where $\boldsymbol{w}_k$ are the primal SVM weights associated with kernel k and b, $\boldsymbol{\xi}$ and C are the standard SVM bias, slack vector and regularisation term. In this formulation β_k is subsumed into the definition of the primal weights and hence does not directly appear in the margin constraint[1]. The primal weight vector can be expressed in terms of the dual variables as

$$\boldsymbol{w}_k = \sum_{i=1}^{N}\alpha_i y_i \beta_k \boldsymbol{\phi}_k(\boldsymbol{O}_i;\boldsymbol{\lambda}) \tag{6.17}$$

There are a number of issues to address when applying this form of MKL directly to speaker verification.

[1] By defining $\hat{\boldsymbol{w}}_k = \boldsymbol{w}_k/\sqrt{\beta_k}$, an equivalent primal form may be obtained that includes the kernel weights in the constraints only. Here the objective function becomes $\min \frac{1}{2}\sum_{k=1}^{K}\langle \hat{\boldsymbol{w}}_k, \hat{\boldsymbol{w}}_k\rangle + C\sum_{i=1}^{N}\xi_i$ and the margin constraints become $y_i\left(\sum_{k=1}^{K}\sqrt{\beta_k}\langle \hat{\boldsymbol{w}}_k, \boldsymbol{\phi}_k(\boldsymbol{O}_i;\boldsymbol{\lambda})\rangle + b\right) \geq 1-\xi_i \ \ \forall i$. This is the form used in section 6.1.3.

6.2.1.1 Regularisation term

In equation 6.16, an l_1-norm constraint is applied to the kernel weights via the sum-to-one constraint. A known consequence of this is to introduce a tendency towards sparse solutions [168]. For a given set of kernels, there is no guarantee that the level of sparsity will be optimal. One solution is to incorporate a regularisation term $\mathcal{R}$ into the objective function to allow the user to control the level of sparsity. A suitable form of regularisation is

$$\mathcal{R} = \zeta \sum_{k=1}^{K} \left(\beta_k - \frac{1}{K} \right)^2 \tag{6.18}$$

$$= \zeta \left(\sum_{k=1}^{K} \beta_k^2 - \frac{2}{K} \left[\sum_{k=1}^{K} \beta_k \right] + \frac{1}{K^2} \left[\sum_{k=1}^{K} 1 \right] \right) \tag{6.19}$$

$$= \zeta \left(\sum_{k=1}^{K} \beta_k^2 - \frac{1}{K} \right) \tag{6.20}$$

due to the sum-to-one constraint on the kernel weights. Since the optimal solution is independent of any constant terms in the objective function, $\mathcal{R} = \zeta \sum_{k=1}^{K} \beta_k^2$ may be used instead. This allows the same form of regularisation term irrespective of K. Here ζ is an empirically set constant. For positive values of ζ the effect of this form of regularisation is to drive towards a uniform set of weights. When ζ is negative the solution will tend to be sparse and the objective function will perform kernel selection. Although an additional parameter has been introduced, an appropriate value for ζ may be obtained through cross-validation even when the number of kernels is large. Note that only a single value needs to be estimated, irrespective of the number of kernels.

6.2.1.2 Cross-speaker tying

In most SVM-based speaker verification systems, such as [22, 30], a distinct set of SVM parameters is trained for each speaker. However, the amount of enrollment data available per speaker is typically limited. Hence, learning a set of speaker-dependent kernel weights in addition to the SVM parameters may lead to over-training. One way to obtain a more robust set of weights is to tie $\boldsymbol{\beta}$ over all enrolled speakers. This can be achieved by redefining the MKL objective function to sum over all speakers, while maintaining a separate set of margin constraints for the enrollment data associated with each speaker.

6.2.1.3 Dynamic range normalisation

The form of objective function given in equation 6.16 is biased towards those kernels for which the average magnitude of the associated feature vectors is greatest. Under a maximally non-committal kernel metric, this corresponds to the kernels for which the associated score space has the greatest dimensionality. It is therefore important that the kernel function includes some form of dynamic range normalisation. One option is spherical normalisation, described in section 2.2.6, where each feature vector is mapped onto the surface of a unit sphere. An alternative approach is to perform normalisation at the kernel level. This may be achieved by replacing

$$K_k(\boldsymbol{O}_i, \boldsymbol{O}_j) \rightarrow \frac{1}{F_k} K_k(\boldsymbol{O}_i, \boldsymbol{O}_j) \tag{6.21}$$

where F_k is the dimensionality of $\boldsymbol{\phi}_k(\boldsymbol{O};\boldsymbol{\lambda})$.

6.2.1.4 Objective function

The `maxMargin` MKL criterion used in this thesis includes a regularisation term and cross-speaker tying of kernel weights. It is defined by the following objective function.

$$\begin{aligned}
\min \quad & \sum_{s=1}^{S}\left(\frac{1}{2}\sum_{k=1}^{K}\frac{1}{\beta_k}\left\langle \boldsymbol{w}_k^{(s)},\boldsymbol{w}_k^{(s)}\right\rangle + C\sum_{i=1}^{N^{(s)}}\xi_i^{(s)}\right) + \zeta\sum_{k=1}^{K}\beta_k^2 \\
\text{w.r.t.} \quad & \boldsymbol{\beta},\boldsymbol{w}_k,\boldsymbol{b},\boldsymbol{\xi} \\
\text{s.t.} \quad & y_i^{(s)}\left(\sum_{k=1}^{K}\left\langle \boldsymbol{w}_k{}^{(s)},\boldsymbol{\phi}_k(\boldsymbol{O}_i^{(s)};\boldsymbol{\lambda})\right\rangle + b^{(s)}\right) \geq 1-\xi_i^{(s)} \;\; \forall i \;\; \forall s \\
& \xi_i^{(s)} \geq 0 \;\; \forall i \;\; \forall s, \quad \beta_k \geq 0 \;\; \forall k, \quad \sum_{k=1}^{K}\beta_k = 1
\end{aligned} \tag{6.22}$$

where $\boldsymbol{w}_k = \{\boldsymbol{w}_k^{(1)},\ldots,\boldsymbol{w}_k^{(S)}\}$, $\boldsymbol{b} = \{b^{(1)},\ldots,b^{(S)}\}$ and $\boldsymbol{\xi} = \{\xi^{(1)},\ldots,\xi^{(s)}\}$.

Equation 6.22 may be efficiently optimised by a similar approach to that used in [168]. First, an equivalent constrained optimisation problem is defined.

$$\min_{\boldsymbol{\beta}} \quad \textstyle\sum_{s=1}^{S} J(s,\boldsymbol{\beta}) + \zeta\sum_{k=1}^{K}\beta_k^2 \tag{6.23}$$

$$\text{s.t.} \quad \beta_k \geq 0 \;\; \forall k, \quad \textstyle\sum_{k=1}^{K}\beta_k = 1 \tag{6.24}$$

where $J(s,\boldsymbol{\beta})$ is the optimal value of the objective function associated with an SVM with the kernel defined by equation 6.8 and fixed kernel weights $\boldsymbol{\beta}$ after training on data associated with speaker s.

$$\begin{aligned}
J(s,\boldsymbol{\beta}) = \quad & \max \sum_{i=1}^{N}\alpha_i^{(s)} - \frac{1}{2}\sum_{i,j=1}^{N}\alpha_i^{(s)}\alpha_j^{(s)}y_i^{(s)}y_j^{(s)}K(\boldsymbol{O}_i,\boldsymbol{O}_j,\boldsymbol{\beta}) \\
\text{w.r.t.} \quad & \boldsymbol{\alpha}^{(s)} \\
\text{s.t.} \quad & \sum_{i=1}^{N}\alpha_i^{(s)}y_i^{(s)} = 0 \\
& 0 \leq \alpha_i^{(s)} \leq C \;\; \forall i
\end{aligned} \tag{6.25, 6.26}$$

where $K(\boldsymbol{O}_i,\boldsymbol{O}_j,\boldsymbol{\beta})$ is the combined kernel function, defined by equation 6.8, for a fixed set of kernel weights $\boldsymbol{\beta}$. A projected-gradient scheme can then be used to optimise equation 6.23. At each iteration $J(s,\boldsymbol{\beta})$ can be estimated using a standard efficient SVM implementation such as [94]. The derivatives of $J(s,\boldsymbol{\beta})$ evaluated at $\boldsymbol{\beta}$ are given by [168].

$$\frac{\partial J(s,\boldsymbol{\beta})}{\partial\beta_k} = -\frac{1}{2}\sum_{i,j=1}^{N}\alpha_i\alpha_j y_i^{(s)}y_j^{(s)}K_k(\boldsymbol{O}_i,\boldsymbol{O}_j) \tag{6.27}$$

6.3 Derivative and parametric kernel combination

In order to achieve gains by combining kernels, it is necessary to select kernels whose associated feature spaces express complementary features. In this following sections two general classes of dynamic kernel are defined, parametric kernels and derivative kernels. Many of the dynamic kernels described in chapter 3 can be placed within one of these two classes. Derivative and parametric kernels are closely related and under certain conditions, described here, the features obtained can be shown to be identical. When these conditions are avoided, kernel combination may be applied to improve system performance.

6.3.1 Parametric kernels

Parametric kernels are a form of dynamic kernel where the feature space consists of a set of parameters $\boldsymbol{\lambda}$ associated with a generative model. A variable-length utterance is mapped to a fixed-dimensional feature representation by training a generative model to represent the utterance and then concatenating the model parameters into a feature vector. Hence the location of an utterance within the feature space is determined, for example, by maximum likelihood parameter estimates given the verification utterance $\boldsymbol{O}^{(\mathrm{v})}$. Thus

$$\boldsymbol{\phi}_{\boldsymbol{\lambda}}(\boldsymbol{O}^{(\mathrm{v})};\boldsymbol{\lambda}) = [\boldsymbol{\lambda}^*], \quad \boldsymbol{\lambda}^* = \underset{\boldsymbol{\lambda}}{\mathrm{argmax}}\{\log p(\boldsymbol{O}^{(\mathrm{v})};\boldsymbol{\lambda})\} \tag{6.28}$$

where the notation $\boldsymbol{\phi}_{\boldsymbol{\lambda}}(\boldsymbol{O};\boldsymbol{\lambda})$ indicates the score space associated with a parametric kernel. The notation $\boldsymbol{\phi}_{\boldsymbol{\lambda}}(\boldsymbol{O};\boldsymbol{\lambda})$ implies that the parametric feature vector is dependent on the current value of the generative model parameters. This is not true when ML estimates are used. However, for other parameter estimation schemes, such as MAP adaptation, this may be the case. One property of this form of kernel is that the derivative with respect to the parameters of the generative model is zero when differentiated at the ML estimate, i.e.

$$\nabla_{\boldsymbol{\lambda}} \log p(\boldsymbol{O}^{(\mathrm{v})};\boldsymbol{\lambda})\Big|_{\boldsymbol{\lambda}^*} = \mathbf{0} \tag{6.29}$$

The precise nature of the parametric kernel is determined by the generative model used to represent the speaker. One parametric kernel that has been successfully used for speaker verification is the GMM-supervector kernel [28], described in section 3.3.1. In this kernel, the feature space is formed from the concatenated means of an utterance-dependent GMM. As there are typically not enough observations per component to robustly estimate the parameters, MAP adaptation, using the UBM as a prior, is used instead. Here

$$\boldsymbol{\lambda}^* = \underset{\boldsymbol{\lambda}}{\mathrm{argmax}}\{\log p(\boldsymbol{O}^{(\mathrm{v})};\boldsymbol{\lambda}) + \log p(\boldsymbol{\lambda})\} \tag{6.30}$$

where $p(\boldsymbol{\lambda})$ is based on the UBM parameters. In this case the property in equation 6.29 will not be satisfied. For a GMM the ML or MAP estimate has no closed form solution. Instead, as described in chapter 4, iterative approaches based on EM are commonly used. For component m the MAP-adapted mean at iteration k is given by

$$\boldsymbol{\mu}_m^{(k)} = \frac{\sum_{t=1}^{T} P(\theta_t = m|\boldsymbol{o}_t;\boldsymbol{\lambda}^{(k-1)})\boldsymbol{o}_t^{(\mathrm{v})} + \tau^{\mathtt{map}}\boldsymbol{\mu}_m^{\mathtt{UBM}}}{\sum_{t=1}^{T} P(\theta_t = m|\boldsymbol{o}_t;\boldsymbol{\lambda}^{(k-1)}) + \tau^{\mathtt{map}}} \tag{6.31}$$

where $\boldsymbol{\mu}_m^{\texttt{UBM}}$ is the UBM mean vector associated with component m (which is also used as the initial parameters $\boldsymbol{\mu}_m^{(0)}$). If k iterations of mean-only MAP adaptation are performed the feature space for the GMM-supervector kernel is

$$\phi_{\boldsymbol{\lambda}}(\boldsymbol{O}^{(\mathrm{v})};\boldsymbol{\lambda}^{(k)}) = \begin{bmatrix} \boldsymbol{\mu}_1^{(k)} \\ \vdots \\ \boldsymbol{\mu}_M^{(k)} \end{bmatrix} \tag{6.32}$$

In section 3.3.1, a metric is defined such that the kernel function is related to an upper bound on the KL divergence between the two utterance-dependent models. This normalises each component mean by the associated mixture weight and the inverse of the covariance matrix. However, when a maximally non-committal metric is applied, these terms will be normalised yielding the form in equation 6.32.

Parametric kernels may be based around alternative forms of generative model. The MLLR [189] and CMLLR [58] kernels, described in section 3.2.6, are examples of parametric kernels where the generative model includes an utterance-dependent linear transform. In both cases the feature space consists of the concatenated transform parameters only. Other forms of parametric kernel include the CAT kernel [211], described in section 3.2.7, and the factor-analysis based kernel used in [47].

6.3.2 Derivative kernels

Derivative kernels provide an interesting contrast to parametric kernels. Rather than using model parameters as the feature space, the partial derivatives of the utterance log-likelihood with respect to individual model parameters are used instead. The set of partial derivatives form a sufficient statistic for the utterance log-likelihood. They are therefore a natural choice for an utterance-dependent fixed-dimensional feature set. For a set of model parameters, $\boldsymbol{\lambda}$, the derivative feature space generated from a verification utterance $\boldsymbol{O}^{(\mathrm{v})}$ has the form

$$\phi_{\nabla}(\boldsymbol{O}^{(\mathrm{v})};\hat{\boldsymbol{\lambda}}) = \left[\nabla_{\boldsymbol{\lambda}} \log p(\boldsymbol{O}^{(\mathrm{v})};\boldsymbol{\lambda})\Big|_{\hat{\boldsymbol{\lambda}}}\right] \tag{6.33}$$

where $\phi_{\nabla}(\boldsymbol{O}^{(\mathrm{v})};\hat{\boldsymbol{\lambda}})$ denotes the score space associated with a derivative kernel and $\hat{\boldsymbol{\lambda}}$ is the model parameter value at which the derivative is evaluated. Derivative kernels may also include higher order derivative terms in the feature space. This is not possible for parametric kernels. However, generally only first order derivatives have been found to contain useful discriminative information [118]. Some derivative kernels, such as the log-likelihood ratio kernel [185], described in section 3.2.4, also include a log-likelihood ratio term in the score space. This is effectively a form of equally weighted kernel combination. The probabilistic sequence kernel [120], described in section 3.2.5, can be interpreted as a form of derivative kernel. Here the features consist of derivatives of the utterance log-likelihood with respect to GMM component priors.

When using derivative kernels it is necessary to define the point around which the derivative kernel feature space will be evaluated. Various points are possible. For example, the point may be based on the UBM parameters. This is similar to using the Fisher kernel described in section 3.2.3. Another possibility is to use speaker-specific parameters associated with a speaker-dependent model. As a GMM is typically used, iterative approaches are used

to obtain the speaker-specific parameters. To more clearly specify the iteration at which the derivative is evaluated, $\log p(\boldsymbol{O}^{(\mathrm{v})};\boldsymbol{\lambda}^{(k)})$, will be used for the feature space evaluated at the k^{th} iteration.

The nature of derivative kernels is again determined by the generative model used to represent a speaker. Derivatives with respect to the means of the GMM can be used [205]. Here elements of the feature space have the form

$$\frac{1}{\rho_m}\nabla_{\boldsymbol{\mu}_m}\log p(\boldsymbol{O}^{(\mathrm{v})};\boldsymbol{\lambda})\Big|_{\boldsymbol{\lambda}^{(k)}} = \frac{1}{\rho_m}\sum_{t=1}^{T} P(\theta_t = m|\boldsymbol{o}_t^{(\mathrm{v})};\boldsymbol{\lambda}^{(k)})\boldsymbol{\Sigma}_m^{-1}(\mathbf{o}_t^{(\mathrm{v})} - \boldsymbol{\mu}_m^{(k)}) \tag{6.34}$$

Equation 6.33 includes an optional duration-normalisation term ρ_m. This is important if the utterances in the dataset vary greatly in duration. ρ_m may be set to the number of frames T in $\boldsymbol{O}^{(\mathrm{v})}$. Alternatively, the derivatives may be normalised by the component occupancy, $\rho_m = \sum_{t=1}^{T} P(\theta_t = m|\boldsymbol{o}_t^{(\mathrm{v})};\boldsymbol{\lambda})$. This approach is only suitable when $\boldsymbol{O}^{(\mathrm{v})}$ is of sufficient duration to ensure that all components are observed in the data. If $\rho_m = 0$ for some component m, the associated features will be undefined.

Given a generative model and associated subset of model parameters both a parametric kernel and derivative kernel can be simply defined. This can be used to motivate new kernels to apply to the SV task. For example, derivative kernels could be defined based around the parameters of a CAT, MLLR or factor analysis system.

6.3.3 Relationship between parametric and derivative kernels

It is interesting to briefly contrast derivative kernels with parametric kernels. From equation 6.29, the derivative of the parametric kernel features at the ML-estimate of the model parameters will be zero for the verification data $\boldsymbol{O}^{(\mathrm{v})}$. In general this will not be the case for the derivative kernel. Instead the features of the derivative kernel will be zero for the enrollment data if the ML-estimate is used as the point to specify the derivative.

$$\hat{\boldsymbol{\lambda}}_e = \underset{\boldsymbol{\lambda}}{\operatorname{argmax}}\{\log p(\boldsymbol{O}^{(\mathrm{e})};\boldsymbol{\lambda})\}, \quad \boldsymbol{\phi}_{\nabla}(\boldsymbol{O}^{(\mathrm{e})};\hat{\boldsymbol{\lambda}}_e) = \mathbf{0} \tag{6.35}$$

In addition derivative kernels commonly use a length normalisation term, $\rho_m = T$. This is not necessary for parametric kernels, where there is an implicit normalisation for the lengths, for example the normalisation term in equation 6.31. A consequence of this is that when the posterior probability of a component is zero, ML-based parametric kernels are undefined, whereas derivative kernels tend to zero.

Both parametric and derivative kernels have been used successfully for speaker-verification [28, 204]. The respective feature spaces can express different types of speaker-discriminant information and thus may be complementary. It is useful to establish under what conditions the two forms of kernel are the same, as this yields information as to how to make the features complementary to one another. The parametric kernel feature space at the kth iteration of training can be expressed in the form of a gradient ascent update.

$$\boldsymbol{\phi}_{\boldsymbol{\lambda}}(\boldsymbol{O}^{(\mathrm{v})};\boldsymbol{\lambda}^{(k+1)}) = \left[\boldsymbol{\lambda}^{(k)} + \boldsymbol{\eta}\nabla_{\boldsymbol{\lambda}}\log p(\boldsymbol{O}^{(\mathrm{v})};\boldsymbol{\lambda})\Big|_{\boldsymbol{\lambda}^{(k)}}\right] \tag{6.36}$$

where $\boldsymbol{\eta}$ is the learning rate[1]. This may then be expressed as a function of a derivative feature space evaluated at $\boldsymbol{\lambda}^{(k)}$

$$\boldsymbol{\phi}_{\boldsymbol{\lambda}}(\boldsymbol{O}^{\mathrm{v}};\boldsymbol{\lambda}^{(k+1)}) = \left[\boldsymbol{\lambda}^{(k)} + \boldsymbol{\eta}\boldsymbol{\phi}_{\nabla}(\boldsymbol{O}^{\mathrm{v}};\boldsymbol{\lambda}^{(k)})\right] \tag{6.37}$$

The two classes of dynamic kernel are thus related to each other. Compared to the derivative kernel feature space, the parametric kernel features includes a term $\boldsymbol{\lambda}^{(k)}$ which introduces a translation of the feature space. If a kernel that is invariant to translation is used, such as a stationary kernel [72], this will have no effect. Note that stationary kernels, such as the Kullback-Leibler divergence kernel [150], have the general form $K(\boldsymbol{O}_i, \boldsymbol{O}_j) = \mathcal{Q}\left(\boldsymbol{\phi}(\boldsymbol{O}_i) - \boldsymbol{\phi}(\boldsymbol{O}_j)\right)$ where $\mathcal{Q}(\cdot)$ is the function that defines the kernel.

Even if a stationary kernel is used, it is not sufficient to ensure that the two sets of features will be identical. Equation 6.37 contains a learning rate $\boldsymbol{\eta}$. Using an appropriate metric, the kernels will not depend on the learning rate if the learning rate is independent of the observation sequence since this dependency is removed by the metric (A maximally non-committal metric has this property but is not stationary). However this is not generally the case. To illustrate this consider the situation where the parametric kernel is obtained using an EM-based ML-estimation of the mean. At iteration $k+1$ the mean parametric feature space for component m can be expressed as

$$\boldsymbol{\mu}_m^{(k+1)} = \boldsymbol{\mu}_m^{(k)} + \left(\frac{\rho_m \boldsymbol{\Sigma}_m}{\sum_{t=1}^{T} P(\theta_t = m|\boldsymbol{o}_t^{\mathrm{v}};\boldsymbol{\lambda}^{(k)})}\right)\left[\frac{1}{\rho_m}\nabla_{\boldsymbol{\mu}_m}\log p(\boldsymbol{O}^{\mathrm{v}};\boldsymbol{\lambda})\Big|_{\boldsymbol{\lambda}^{(k)}}\right] \tag{6.38}$$

EM is thus equivalent to gradient ascent using the derivative features with a learning rate that depends on the total component occupancy for that observation sequence as well as the length of the observation sequence (when length normalisation is being used)[2]. If the derivative features are normalised by the component occupancy rather than the total duration then $\boldsymbol{\eta} = \boldsymbol{\Sigma}_m$, which is independent of the observation sequence[3].

It is more common to use MAP adaptation to successively update models. When the MAP prior $\boldsymbol{\mu}^{\mathtt{prior}}$ equals $\boldsymbol{\mu}^{(k)}$, this update is equivalent to gradient ascent with the following learning rate.

$$\boldsymbol{\eta} = \left(\frac{\rho_m \boldsymbol{\Sigma}_m}{\sum_{t=1}^{T} P(\theta_t = m|\boldsymbol{o}_t^{\mathrm{v}};\boldsymbol{\lambda}^{(k)}) + \tau^{\mathtt{map}}}\right) \tag{6.39}$$

If both parametric and derivative features are to be used, it is important that the features differ. This can be achieved using a non-stationary kernel, evaluating the derivative terms at a different point to the parametric features (effectively using a different number of iterations), or, when $\rho_m = T$, simply using either EM updates or MAP adaptation with a low value of $\tau^{\mathtt{map}}$. Combinations of these may make the features more complementary.

[1]The learning rate in equation 6.36 has the form of a diagonal matrix. This allows the learning rate associated with each element of the feature vector to be set independently.

[2]The fact that EM always has a component in the right direction, and hence can be expressed in the form of a gradient ascent update, is a well known property of the EM algorithm.

[3]Since $\boldsymbol{\Sigma}_m$ is typically diagonal, the associated learning rate $\boldsymbol{\eta}$ will also be diagonal.

6.3.4 Combination of generative model structures

For dynamic kernels that incorporate a generative model, such as parametric or derivative kernels, an appropriate form of model must be selected. If a GMM is used, the number of Gaussian components must be chosen. This is a trade-off between improving the ability of the model to approximate the distribution over the acoustic space and ensuring that the model parameters can be robustly estimated with the available data. A suitable model size is typically chosen by selecting a value that reduces the error rate on some development dataset. As the trade-off is data-dependent this strategy may not be optimal.

If a suitable scheme for combining classifiers is available, then other strategies may be used. Rather than selecting a single form of model, a series of dynamic kernels can instead be defined, each based on different model structures. The associated classifiers can then be combined. Although this approach is more computationally expensive it has two advantages. Firstly, there is no need for prior knowledge about the task in order to select a suitable model size. Secondly, rather than making a single trade-off, the combined classifier can make use of features extracted from a range of different model structures, potentially leading to gains.

6.4 Summary

This chapter has examined the combination of multiple dynamic kernels to improve the performance of an SV system. The standard approach for classifier combination is to fuse the output scores. Here an alternative approach was considered, suitable for SVMs, where classifiers are combined at the kernel level. In this chapter, a number of modifications were proposed for a recently developed maximum-margin based scheme for learning a suitable kernel weighting. The scheme has a known tendency towards sparse weightings, which may not be optimal for speaker verification. A regularisation term was proposed allowing the user to tune the sparsity by adjusting a single parameter. Tying of kernel weights over all speakers was also applied to increase the robustness of the parameter estimates.

In order to obtain gains using kernel-combination it is important that the kernels used are complementary. In this chapter it was shown that many existing dynamic kernels can be placed into one of these two classes, *parametric kernels*, where the feature space consists of parameters from the utterance-dependent model, and *derivative kernels*, where the derivatives of the utterance log-likelihood with respect to parameters of a generative model are used. The two sets of features produced have different properties and may be complementary. However, under certain conditions, discussed in section 6.3.3, the feature spaces produced can be shown to be identical. By avoiding these conditions a complementary set of kernels may be obtained.

Chapter 7

Static and Dynamic Kernel Combination

One method of improving the performance of an SVM-based speaker verification system is to combine complementary kernels. In chapter 6, new kernels were formed by combining the features of multiple dynamic kernels. This chapter examines an alternative approach where dynamic kernels are combined with traditional static kernels designed for fixed-dimensional data. Two combination schemes are considered in this chapter. In the first approach, feature-level combination, fixed-dimensional feature vectors are obtained from each utterance using a dynamic kernel. Then, rather than defining the kernel function using an inner product, a static kernel function is applied instead. This approach has previously been found to yield gains, for example in [18].

For the second approach, observation-level combination, instead of calculating dynamic features in the original observation space, these features are calculated in the space associated with a static kernel defined between pairs of observations. Verification may then be based on higher order observation level features while exploiting a dynamic kernel to obtain a fixed set of features. This combination strategy is implicitly used by the GLDS kernel [24] to combine static kernels with simple 'averaging' dynamic kernels. In this chapter, this approach is extended to more complex forms of dynamic kernel based on generative models.

In general, it is not possible to explicitly train a generative model in the feature space associated with a static kernel. However, this chapter shows how such a model can be approximated allowing generalised versions of both derivative and parametric kernels to be computed. Static and dynamic kernel combination can potentially yield extremely high-dimensional feature spaces, which may lead to overtraining. In the final section of this chapter two schemes

are proposed to handle these conditions. The first scheme involves splitting training utterances and in the second scheme the classifier parameters are tied over multiple target speakers.

7.1 Static and dynamic kernels

The use of kernel functions to allow linear classifiers to yield non-linear decision boundaries was described in chapter 2. Many kernels, including the polynomial and Gaussian kernels [181], introduced in section 2.2.6, are examples of static kernels. They can only handle data of fixed dimensionality. In contrast, speech utterances are typically parameterised as variable-length sequences of observation vectors $\boldsymbol{O} = \{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_T\}$. This has led to the use of dynamic kernels, introduced in chapter 3. These kernels map variable-length sequences into a fixed dimension feature space in which the inner product can be computed.

As well as mapping variable-length sequences to a fixed-dimensional representation, dynamic kernels also share an important requirement with static kernels. The kernel should map the input into a feature space where the classes may easily be separated. It is therefore of interest to examine whether combining dynamic kernels with static kernels can improve system performance. In chapter 6, new kernels were defined by combining the features expressed by multiple dynamic kernels. Since static kernels cannot be applied directly to variable-length data an alternative approach is used in this chapter. Static and dynamic expansions are applied alternatively to map each utterance into a fixed, high-dimensional feature space. As in previous chapters, the notation $k(\boldsymbol{o}_i, \boldsymbol{o}_j)$ and $\boldsymbol{\psi}(\boldsymbol{o})$ is used to indicate the kernel function and score operator associated with a static kernel. $K(\boldsymbol{O}_i, \boldsymbol{O}_j)$ and $\boldsymbol{\phi}(\boldsymbol{O})$ is reserved for dynamic kernels.

Static and dynamic kernels may be combined in several ways. Since static kernels can only handle fixed-dimensional data, two natural points to apply a static expansion are at the level of individual observations, $\boldsymbol{\psi}(\boldsymbol{o})$, and to the fixed-dimensional set of features obtained from a dynamic kernel, $\boldsymbol{\psi}(\boldsymbol{\phi}(\boldsymbol{O}))$. These approaches, shown in figure 7.2, are discussed in the following two sections. While it is also possible to combine these approaches and perform multiple static expansions within the same kernel, this approach is likely to suffer from overtraining due to the extremely large feature spaces obtained and hence is not considered here.

7.2 Feature-level combination

Many forms of dynamic kernel, including the families of derivative and parametric kernels discussed in chapter 6, are explicitly defined in terms of the associated feature space. A kernel function between two utterances $\boldsymbol{O}_i$ and $\boldsymbol{O}_j$ is then evaluated by explicitly mapping each utterance $\boldsymbol{O}$ into the feature space using a dynamic score operator $\boldsymbol{\phi}(\boldsymbol{O})$, and calculating the inner product between the mapped feature vectors. This approach yields dynamic kernels of the form.

$$K(\boldsymbol{O}_i, \boldsymbol{O}_j) = \langle \boldsymbol{\phi}(\boldsymbol{O}_i), \boldsymbol{\phi}(\boldsymbol{O}_j) \rangle \tag{7.1}$$

Rather than using the inner product, it is possible to instead apply a suitable static kernel function, $k(\boldsymbol{o}_i, \boldsymbol{o}_j)$, between the mapped feature vectors $\boldsymbol{\phi}(\boldsymbol{O})$. By applying the kernel operations introduced in section 2.2.6 this approach can be shown to yield valid kernel functions.

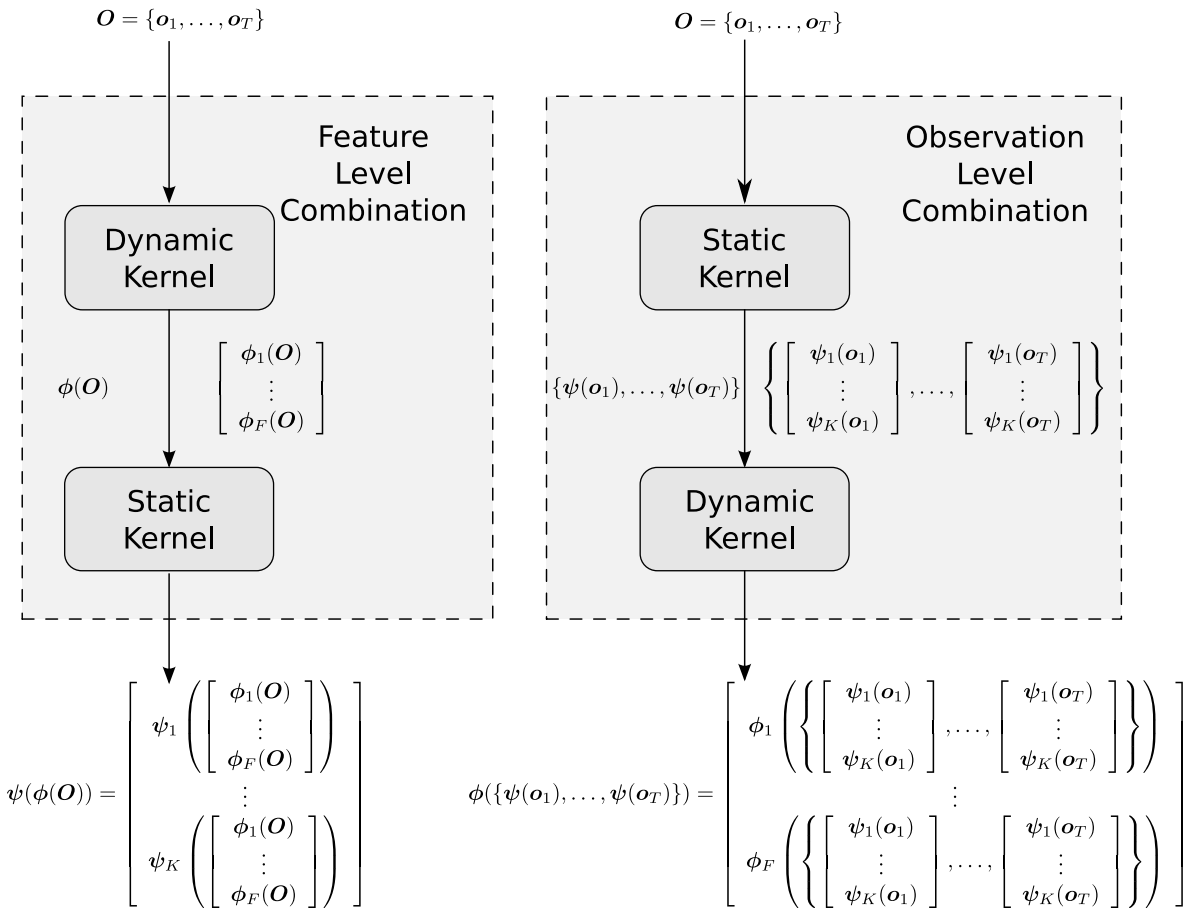


Figure 7.2: Observation and feature-level combination of dynamic and static kernels

Thus a new dynamic kernel may be defined that has the form

$$K(\boldsymbol{O}_i, \boldsymbol{O}_j) = k(\boldsymbol{\phi}(\boldsymbol{O}_i), \boldsymbol{\phi}(\boldsymbol{O}_j)) \tag{7.2}$$

This approach yields more complex feature spaces, which may express useful information for discriminating between speakers. For example, when a polynomial static kernel is applied to a parametric or derivative dynamic kernel based on GMMs, the resulting features will combine information obtained from multiple Gaussian components.

For dynamic kernels it is common to explicitly define a suitable metric, for example one that is maximally non-committal, between feature vectors. This yields dynamic kernels of the form.

$$K(\boldsymbol{O}_i, \boldsymbol{O}_j) = \boldsymbol{\phi}(\boldsymbol{O}_i)^{\mathsf{T}} \boldsymbol{G}^{-1} \boldsymbol{\phi}(\boldsymbol{O}_j) \tag{7.3}$$

where $\boldsymbol{G}^{-1}$ defines the metric. However, for many forms of static kernel, including polynomial and Gaussian kernels, the nature of the kernel function implicitly defines the metric used. Thus, when static and dynamic kernels are combined using equation 7.2 it is difficult to explicitly define a suitable metric in the combined feature space. When an explicit metric is required, for example to normalise the dynamic features, this may be approximated using a metric defined in the original dynamic feature space by applying a projection to each feature vector. Here the combined kernel has the form.

$$K(\boldsymbol{O}_i, \boldsymbol{O}_j) = k(\boldsymbol{P}\boldsymbol{\phi}(\boldsymbol{O}_i), \boldsymbol{P}\boldsymbol{\phi}(\boldsymbol{O}_j)) \tag{7.4}$$

where $\boldsymbol{P}$ is a projection defined by $\boldsymbol{G}^{-1} = \boldsymbol{P}^{\mathsf{T}}\boldsymbol{P}$. Many forms of derivative and parametric kernel, such as the Fisher [90] or GMM-supervector kernels [28], are typically implemented using diagonal metrics. In this case equation 7.4 can be calculated efficiently.

Feature-level static and dynamic combination is relatively simple to implement and has been applied in systems such as [18]. Dynamic kernels may also implicitly include static approaches. For example, the non-linear GMM-supervector kernel [45], introduced in section 3.3.2, is motivated as an exponentiated KL divergence between distributions. However, the form of kernel obtained is equivalent to applying a Gaussian static kernel to the features obtained from a standard GMM-supervector kernel.

7.3 Observation-level combination

An alternative combination strategy for static and dynamic kernels is to apply a static kernel at the frame level. Here, instead of computing an inner product between pairs of frames, a static kernel function is applied instead. The GLDS kernel [24], introduced in section 3.2.1, is an example of this approach. Here each observation $\boldsymbol{o}_t$ is initially mapped into a static feature space $\boldsymbol{\psi}(\boldsymbol{o}_t)$. A duration-independent, fixed-dimensional vector is then obtained by taking the mean of the expanded observations $\boldsymbol{\phi}(\boldsymbol{O}) = \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{\psi}(\boldsymbol{o}_t)$. The kernel function is defined by taking the inner product of the means and is equivalent to

$$K(\boldsymbol{O}_i, \boldsymbol{O}_j) = \frac{1}{T_i T_j} \sum_{t=1}^{T_i} \sum_{\tau=1}^{T_j} k(\boldsymbol{o}_{it}, \boldsymbol{o}_{j\tau}) \tag{7.5}$$

where $k(\boldsymbol{o}_t, \boldsymbol{o}_s) = \langle \boldsymbol{\psi}(\boldsymbol{o}_t), \boldsymbol{\psi}(\boldsymbol{o}_\tau) \rangle$. Thus the GLDS kernel may be interpreted as combining a static kernel at the frame level with a simple dynamic kernel that averages over all frames. Standard forms of static kernel such as polynomial or Gaussian kernels may be applied. When $k(\boldsymbol{o}_i, \boldsymbol{o}_j)$ is linear the GLDS kernel may be simplified to

$$K(\boldsymbol{O}_i, \boldsymbol{O}_j) = \langle \boldsymbol{\mu}_i, \boldsymbol{\mu}_j \rangle \tag{7.6}$$

where $\boldsymbol{\mu}_i$ is the mean of the observations in $\boldsymbol{O}_i$. Thus, in the linear case the GLDS kernel can be interpreted as a form of mean-based parametric kernel, using single component GMMs. One disadvantage to using this form of kernel is that for static kernels that do not have an explicitly defined feature space, such as the Gaussian kernel, the kernel function must be calculated between all pairs of observations. For longer utterances this can be computationally expensive. In the following sections higher order kernels are derived based on derivative and parametric kernels. Unlike the GLDS kernel these can be efficiently computed for long utterances.

7.3.1 Generalised derivative kernel

The GLDS kernel exploits a static kernel to explicitly map each observation into a more separable feature space. However, by taking a direct sum over all observations useful information may be 'averaged out'. Also, the resultant features may lack robustness to intermittent noise or long regions of silence. In contrast, generative kernels, such as the derivative kernel, provide a well-motivated mapping to a fixed-dimensional set of features. However typically only a simple inner product is calculated in the dynamic feature space.

The *generalised derivative kernel* (GDK) combines these two approaches. Here, the features are derivatives with respect to the parameters of a model, $p(\tilde{\boldsymbol{O}}; \tilde{\boldsymbol{\lambda}})$, defined in the feature space associated with a static kernel. Here $p(\tilde{\boldsymbol{O}}; \tilde{\boldsymbol{\lambda}})$ models the likelihood of a sequence of expanded observations, $\tilde{\boldsymbol{O}} = \{\boldsymbol{\psi}(\boldsymbol{o}_1), \ldots, \boldsymbol{\psi}(\boldsymbol{o}_T)\}$. For GMMs, derivatives with respect to each feature space component mean are defined by

$$\nabla_{\tilde{\boldsymbol{\mu}}_m} \log p(\tilde{\boldsymbol{O}}; \tilde{\boldsymbol{\lambda}}) \Big|_{\tilde{\boldsymbol{\lambda}}} = \sum_{t=1}^{T} \tilde{\boldsymbol{\Sigma}}_m^{-1} P(\theta_t = m | \boldsymbol{\psi}(\boldsymbol{o}_t); \tilde{\boldsymbol{\lambda}}) (\boldsymbol{\psi}(\boldsymbol{o}_t) - \tilde{\boldsymbol{\mu}}_m) \tag{7.7}$$

where $\tilde{\boldsymbol{\mu}}_m$ and $\tilde{\boldsymbol{\Sigma}}$ are the mean and covariance matrix respectively associated with component m of the feature space GMM. $P(\theta_t = m | \boldsymbol{\psi}(\boldsymbol{o}_t); \tilde{\boldsymbol{\lambda}})$ is the posterior probability that $\boldsymbol{\psi}(\boldsymbol{o}_t)$ was emitted by component m of this GMM. When the feature space consists of only mean derivatives the kernel function $\tilde{K}_\nabla(\boldsymbol{O}_i, \boldsymbol{O}_j)$ associated with the generalised derivative kernel has the form

$$\tilde{K}_\nabla(\boldsymbol{O}_i, \boldsymbol{O}_j) = \sum_{m=1}^{M} \frac{1}{\rho_{im} \rho_{jm}} \sum_{t=1}^{T_i} \sum_{\tau=1}^{T_j} P(\theta_t = m | \boldsymbol{\psi}(\boldsymbol{o}_t); \tilde{\boldsymbol{\lambda}}) P(\theta_\tau = m | \boldsymbol{\psi}(\boldsymbol{o}_\tau); \tilde{\boldsymbol{\lambda}}) f_m(\boldsymbol{o}_{it}, \boldsymbol{o}_{j\tau}) \tag{7.8}$$

$$f_m(\boldsymbol{o}_i, \boldsymbol{o}_j) = [\boldsymbol{\psi}(\boldsymbol{o}_i) - \tilde{\boldsymbol{\mu}}_m]^{\mathsf{T}} \tilde{\boldsymbol{\Sigma}}_m^{-1} \tilde{\boldsymbol{G}}_m^{-1} \tilde{\boldsymbol{\Sigma}}_m^{-1} [\boldsymbol{\psi}(\boldsymbol{o}_j) - \tilde{\boldsymbol{\mu}}_m] \tag{7.9}$$

where T_i and T_j are respectively the length of utterances $\boldsymbol{O}_i$ and $\boldsymbol{O}_j$, and $\tilde{\boldsymbol{G}}_m$ is the mth block of a block-diagonal metric $\tilde{\boldsymbol{G}}$ defined in the combined feature space. Here ρ_{im} is a duration normalisation term. This may be equal to the number of frames $\rho_{im} = T_i$ or the total component occupancy $\rho_{im} = \sum_{t=1}^{T_i} P(\theta_t = m | \boldsymbol{\psi}(\boldsymbol{o}_t); \tilde{\boldsymbol{\lambda}})$. When the static kernel function is linear, the GDK has the form of a standard GMM mean-based derivative kernel. Alternatively,

when the model is a single-component GMM with zero mean and unit variance, the GDK has the form of the GLDS kernel in equation 7.5. Thus both the GLDS and derivative kernels are special cases of the GDK.

Evaluating equation 7.8 requires training a generative model in the static feature space. In general, this is not possible and approximations must be used. Two key issues are how to obtain suitable component posteriors and how to estimate $f_m(\boldsymbol{o}_i, \boldsymbol{o}_j)$. These are discussed in the following subsections.

7.3.1.1 Component posterior estimation

To calculate equation 7.8, a kernel function must be computed between each pair of observations. This requires $\mathcal{O}(T^2)$ static kernel evaluations and may not be feasible when the utterances are long. A more efficient approach is to perform a Viterbi alignment of each observation to a component. Here the kernel function is approximated by

$$K(\boldsymbol{O}_i, \boldsymbol{O}_j) \approx \sum_{m=1}^{M} \frac{1}{\rho_{im}\rho_{jm}} \sum_{t \in \tilde{\mathcal{S}}_{im}} \sum_{\tau \in \tilde{\mathcal{S}}_{jm}} f_m(\boldsymbol{o}_{it}, \boldsymbol{o}_{j\tau}) \tag{7.10}$$

where

$$t \in \tilde{\mathcal{S}}_{i\hat{m}} \quad \text{if} \quad \hat{m} = \underset{m}{\text{argmax}} \left\{ P(\theta_t = m | \boldsymbol{\psi}(\boldsymbol{o}_t); \tilde{\boldsymbol{\lambda}}) \right\} \tag{7.11}$$

When the number of frames assigned to each component is roughly equal, this requires only $\mathcal{O}(T^2/M)$ static kernel evaluations. The use of a Viterbi alignments can therefore yield considerable computational savings. Although counter-intuitive, it is also more efficient to evaluate equation 7.10 for GMMs with a greater number of components.

Doubling the dimension of each observation, e.g. by duplicating each element and model parameter, results in a squared decrease in the likelihood of it being generated by each GMM component. Hence, the posterior probability of the most likely component will increase. A consequence of this is that as the dimensionality of the feature space increases, GMMs tend towards hard component alignments. Therefore the approximation used in equation 7.10 will be more robust when the dimensionality of $\boldsymbol{\psi}(\boldsymbol{o})$ is large. This is related to an approach used in [142], where a kernel function is only computed between a frame and its 'closest' frame in the other sequence. Here the generative model is used to identify sets of close frames.

An important issue is how to obtain $P(\theta_t = m | \boldsymbol{\psi}(\boldsymbol{o}_t); \tilde{\boldsymbol{\lambda}})$ without training a generative model in the feature space. One approach is to use the posteriors or Viterbi alignments from a model $p(\boldsymbol{O}; \boldsymbol{\lambda})$ trained in the observation space. Thus

$$\tilde{\mathcal{S}}_{im} \approx \mathcal{S}_{im} \text{ where } t \in \mathcal{S}_{i\hat{m}} \text{ if } \hat{m} = \underset{m}{\text{argmax}} \left\{ P(\theta_t = m | \boldsymbol{o}_t; \boldsymbol{\lambda}) \right\} \tag{7.12}$$

This may yield poor estimates when relative distances between observations differ greatly between the feature and input spaces.

7.3.1.2 Static kernel estimation

In equation 7.9, $f_m(\boldsymbol{o}_i, \boldsymbol{o}_j)$ is a function of the feature space variance $\tilde{\boldsymbol{\Sigma}}_m$ and the metric $\tilde{\boldsymbol{G}}_m$ applied to the derivatives associated with component m. Generally a generative model cannot be trained in the static feature space, nor the feature space $\boldsymbol{\psi}(\boldsymbol{o})$ explicitly generated. Thus it is not possible to obtain these parameters directly, as in the linear case. When $\boldsymbol{\psi}(\boldsymbol{o})$ does not have an explicit representation one option is to apply an Incomplete Cholesky Decomposition to obtain a low-rank approximation to the kernel matrix, as in [135]. In this work an alternative approach is used. Noting that $\tilde{\boldsymbol{\Sigma}}_m^{-1}$ is independent of $\boldsymbol{O}$ and thus is normalised under a maximally non-committal metric, here $\tilde{\boldsymbol{\Sigma}}_m^{-1}\tilde{\boldsymbol{G}}_m^{-1}\tilde{\boldsymbol{\Sigma}}_m^{-1}$ is approximated by the identity matrix. This allows standard forms of static kernel to be used and avoids explicitly estimating $\tilde{\boldsymbol{\Sigma}}_m^{-1}$ and $\tilde{\boldsymbol{G}}_m$. This approximation is likely to be more robust when observations are globally whitened. Here $f_m(\boldsymbol{o}_i, \boldsymbol{o}_j)$ is approximated by

$$f_m(\boldsymbol{o}_i, \boldsymbol{o}_j) \approx (\boldsymbol{\psi}(\boldsymbol{o}_i) - \tilde{\boldsymbol{\mu}}_m)^{\mathsf{T}}(\boldsymbol{\psi}(\boldsymbol{o}_i) - \tilde{\boldsymbol{\mu}}_m) \tag{7.13}$$

For the case when $\tilde{\boldsymbol{\mu}}_m$ represents the ML estimate over a set of background observations $\boldsymbol{O}^{\mathsf{B}} = \{\boldsymbol{o}^{\mathsf{B}1}, \ldots, \boldsymbol{o}^{\mathsf{B}}_{T^{\mathsf{B}}}\}$, equation 7.13 can be expressed in terms of the static kernel function.

$$f_m(\boldsymbol{o}_i, \boldsymbol{o}_j) \approx k(\boldsymbol{o}_i, \boldsymbol{o}_j) - k^{\mu}_m(\boldsymbol{o}_i) - k^{\mu}_m(\boldsymbol{o}_j) + k^{\mu\mu}_m \tag{7.14}$$

where $k^{\mu}_m(\boldsymbol{o})$ and $k^{\mu\mu}_m$ are defined as

$$k^{\mu}_m(\boldsymbol{o}) = \frac{1}{C^{\mathsf{B}}_m} \sum_{t \in \mathcal{S}^{\mathsf{B}}_m} k(\boldsymbol{o}, \boldsymbol{o}^{\mathsf{B}}_t) \tag{7.15}$$

$$k^{\mu\mu}_m = \frac{1}{(C^{\mathsf{B}}_m)^2} \sum_{t,\tau \in \mathcal{S}^{\mathsf{B}}_m} k(\boldsymbol{o}^{\mathsf{B}}_t, \boldsymbol{o}^{\mathsf{B}}_\tau) \tag{7.16}$$

where C^{B}_m is the number of frames in $\boldsymbol{O}^{\mathsf{B}}$ aligned with component m. Calculating $f_m(\boldsymbol{o}_i, \boldsymbol{o}_j)$ requires evaluating a static kernel function between each observation and the entire background dataset. When the number of background observations is large, this may not be feasible. Instead, the mean normalisation in the feature space, $\boldsymbol{\psi}(\boldsymbol{o}) - \tilde{\boldsymbol{\mu}}_m$, can be approximated by a normalisation evaluated in the observation space $\boldsymbol{\psi}(\boldsymbol{o} - \boldsymbol{\mu}_m)$. Again, when distances vary greatly between the observation and feature space, this approximation may not be robust. In this thesis, $f_m(\boldsymbol{o}_i, \boldsymbol{o}_j)$ is approximated by

$$f_m(\boldsymbol{o}_i, \boldsymbol{o}_j) \approx k([\boldsymbol{o}_i - \boldsymbol{\mu}_m], [\boldsymbol{o}_j - \boldsymbol{\mu}_m]) \tag{7.17}$$

7.3.2 Generalised parametric kernel

Derivative and parametric kernels are closely related and under certain conditions, discussed in chapter 6, the functions obtained can be shown to be identical. A similar approach may therefore be used to define a *generalised parametric kernel* (GPK) where the generative model is defined in the feature space associated with a static kernel.

If the feature space consists of GMM means only, trained using ML-based EM, then for each utterance $\boldsymbol{O}_i$ the elements of the generalised parametric feature space associated with component m is defined by

$$\tilde{\boldsymbol{\phi}}^{\mathtt{gpk}}_m(\boldsymbol{O}_i) = \left[\frac{\sum_{t=1}^{T} P(\theta_t = m | \boldsymbol{\psi}(\boldsymbol{o}_{it}); \tilde{\boldsymbol{\lambda}}_i) \boldsymbol{\psi}(\boldsymbol{o}_t)}{\sum_{t=1}^{T} P(\theta_t = m | \boldsymbol{\psi}(\boldsymbol{o}_{it}); \tilde{\boldsymbol{\lambda}}_i)} \right] \tag{7.18}$$

where $\tilde{\boldsymbol{\lambda}}_i$ are the parameters of a generative model $p(\boldsymbol{\psi}(\boldsymbol{o}); \tilde{\boldsymbol{\lambda}}_i)$, trained in the feature space, associated with utterance $\boldsymbol{O}_i$.

As in section 7.3.1, hard component alignments can be obtained for each utterance using a linear model $p(\boldsymbol{O}; \boldsymbol{\lambda}_i)$, trained to represent $\boldsymbol{O}_i$, and a maximally non-committal metric again approximated using an identity metric. In this case, an approximation to the GPK kernel function, $\tilde{K}_{\boldsymbol{\lambda}}(\boldsymbol{O}_i, \boldsymbol{O}_j)$, is obtained that has the form

$$\tilde{K}_{\boldsymbol{\lambda}}(\boldsymbol{O}_i, \boldsymbol{O}_j) \approx \sum_{m=1}^{M} \frac{\sum_{t \in \mathcal{S}_{im}} \sum_{\tau \in \mathcal{S}_{jm}} k(\boldsymbol{o}_{it}, \boldsymbol{o}_{j\tau})}{C_{im} C_{jm}} \tag{7.19}$$

where C_{im} is number of frames in the set $\mathcal{S}_{im}$ associated with component m of $p(\boldsymbol{O}; \boldsymbol{\lambda}_i)$. This is similar to equation 7.10 when component-occupancy normalisation is used. If the GDK is computed using the approximation in equation 7.17, all component alignments estimated using the same model, and $k(\boldsymbol{o}_i, \boldsymbol{o}_j)$ is stationary, the two forms of kernel will be identical.

In practice, instead of using ML-based features, parametric features are often obtained by adapting the parameters of a robust background model $p(\tilde{\boldsymbol{O}}; \tilde{\boldsymbol{\lambda}}^{\texttt{UBM}})$ using MAP. Typically the parameter estimates $\tilde{\boldsymbol{\lambda}}^{\texttt{UBM}}$ are also used to define the prior distribution. Computing the MAP estimate requires evaluating a kernel function between each background observation used to train the prior model, which will usually be infeasible. Alternatively, the parameters $\tilde{\boldsymbol{\lambda}}^{\texttt{UBM}}$ of a prior defined in the static feature space may be approximated by statically mapped parameters $\boldsymbol{\lambda}^{\texttt{UBM}}$ defined in the observation space. In this case the GPK has the following form

$$K_{\boldsymbol{\lambda}}(\boldsymbol{O}_i, \boldsymbol{O}_j) \approx \sum_{m=1}^{M} \frac{\left(k_m^{\texttt{map}}\right)^2 + k_{im}^{\texttt{map}} + k_{jm}^{\texttt{map}} + \sum_{t \in \mathcal{S}_{im}} \sum_{s \in \mathcal{S}_{jm}} k(\boldsymbol{o}_{it}, \boldsymbol{o}_{js})}{(C_{im} + \tau^{\texttt{map}})(C_{jm} + \tau^{\texttt{map}})} \tag{7.20}$$

$$k_{im}^{\texttt{map}} = \tau^{\texttt{map}} \sum_{t \in \mathcal{S}_{im}} k(\boldsymbol{o}_{it}, \boldsymbol{\mu}_m^{\texttt{UBM}}) \tag{7.21}$$

$$k_m^{\texttt{map}} = \tau^{\texttt{map}} \boldsymbol{\mu}_m^{\texttt{UBM}} \tag{7.22}$$

7.4 Dealing with limited data

Static and dynamic kernel combination may yield gains if the combined feature space allows more effective discrimination between speakers. However, when static and dynamic feature spaces are combined, the resultant feature spaces are often extremely high dimensional. For speaker verification tasks, utterances typically contain tens of thousands of frames. This is usually sufficient to allow each feature vector to be robustly estimated. However, each enrollment utterance yields only a single example for SVM training. For many speaker verification tasks, such as the NIST SRE 2002 task used to evaluated these methods in chapter 8, only one training utterance is available per speaker. Thus, over-training will often become a significant issue. For dynamic kernels where the size of the feature vector can be set by the user, the optimal number of dynamic features may be smaller when combining with complex static kernels. For generative kernels, including parametric and derivative kernels, this may be achieved by using models with fewer Gaussian components. Alternatively, various approaches may be used to increase the number of SVM training examples per classifier. Two strategies are described in the following subsections.

7.4.1 Data partitioning

Before the development of dynamic kernels, several alternative strategies were proposed to handle the dynamic nature of speech. One approach, used in [203], is to treat each speech frame as a distinct training utterance. An SVM classifier may then be trained, using a static kernel, to distinguish between frames extracted from the speech associated with target and imposter speakers. For test utterances, each frame is then classified independently and the average of the scores obtained.

$$S(\boldsymbol{O}^{\mathrm{v}}, s) = \frac{1}{T} \sum_{t=1}^{T} \left\langle \boldsymbol{w}, \psi(\boldsymbol{o}_t^{\mathrm{v}}, \boldsymbol{\lambda}^{(s)}) \right\rangle + b \tag{7.23}$$

As described in chapter 4, state-of-the-art SVM-based SV systems typically use dynamic kernels to handle the variable-length-nature of the speech. Here each utterance is mapped into a single feature vector, located within the feature space defined by the dynamic kernel. This approach offers a number of advantages over the frame-level method described above. For example, generative models may be used to extract high-level structure from each utterance. For SV, the utterance is also a more natural unit to use, since most causes of inter-session variability will be constant during a single recording session.

When the number of enrollment utterances per speaker is limited, and high dimensional feature spaces are used, the SVM classifier parameters may not be robust. Thus, it may be worth considering approaches that combine these schemes. One approach is to extract multiple feature vectors from each utterance using a sliding window, applied for static kernels in [57]. Here a simplified scheme is considered, where each utterance $\boldsymbol{O}_i$ is simply divided into K partitions of equal length. An SVM training example $\phi(\boldsymbol{O}_{ik})$ is then obtained from each partition using a dynamic kernel. This process is illustrated in figure 7.3.

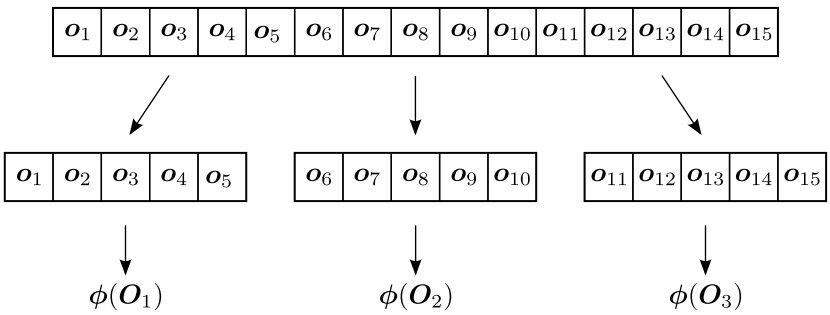


Figure 7.3: Data partitioning

During training, a distinct SVM parameter α_{ik} is associated with each feature vector $\phi(\boldsymbol{O}_{ik})$ extracted from utterance $\boldsymbol{O}_i$. Thus the SVM is able to independently weight features extracted from different regions of the utterance, potentially yielding a more discriminative classifier. During test, the effect of data partitioning is dependent on the form of dynamic and static kernels used. For many dynamic kernels, including the linear GLDS kernel and GMM mean-based derivative kernels using duration normalisation, the mean of the obtained feature vectors is equal to the feature vector $\phi(\boldsymbol{O})$ associated with the original utterance.

$$\phi(\boldsymbol{O}_i) = \frac{1}{K} \sum_{k=1}^{K} \phi(\boldsymbol{O}_{ik}) \tag{7.24}$$

For dynamic kernels where equation 7.24 holds, the mean of the scores obtained from classifying a partitioned test utterance $\boldsymbol{O}^{\mathrm{v}}$ using a linear kernel can be expressed as

$$S(\boldsymbol{O}^{\mathrm{v}}, s) = \frac{1}{K}\sum_{k=1}^{K}\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{O}_k^{\mathrm{v}}, s)\rangle + b \tag{7.25}$$

$$= \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{O}^{\mathrm{v}}, s)\rangle + b \tag{7.26}$$

Thus, no gain is obtained from applying data partitioning to the test data. Note that this is not the case when using non-linear static kernels, HMM-based dynamic kernels, or component-occupancy based dynamic kernels such as parametric kernels.

A disadvantage to data partitioning is that as the number of partitions is increased, less frames are used to generate each feature-vector, potentially yielding less robust estimates. Data partitioning is therefore a trade-off between increasing the number of SVM training examples and ensuring that for each training example the parameters are robust. Note that for feature-level combination using generative kernels, it is primarily the number of observations per model component that determines the robustness of the parameter estimates rather than the dimension of the combined feature-space. Thus data partitioning will be most appropriate when dynamic kernels based on small generative models are to be combined with high dimensional static kernels.

7.4.2 Cross-speaker tying

The dominant approach for SVM-based speaker verification is to learn a distinct SVM classifier for each target speaker. Each speaker-dependent classifier is then trained using positive data obtained from the target-speaker and negative data obtained from competing imposter speakers. One drawback to this approach is that the amount of available enrollment data per target speaker is typically limited. Thus the available training set is heavily biased in favour of imposter speakers, potentially causing generalisation issues.

An alternative approach, that has not previously been applied in the literature, is to train a single speaker-independent classifier, tied over all target speakers, and use a dynamic kernel, defined by $\boldsymbol{\phi}(\boldsymbol{O}; \boldsymbol{\lambda}^{(s)})$, to handle changes in the target speaker. Here the SVM objective function is given by,

$$\begin{aligned}
\min \quad & \frac{1}{2}\langle \boldsymbol{w}, \boldsymbol{w}\rangle + C\sum_{s=1}^{S}\sum_{i=1}^{N^{(s)}}\xi_i^{(s)} \\
\text{w.r.t.} \quad & \boldsymbol{w}, b, \boldsymbol{\xi} \\
\text{s.t.} \quad & y_i^{(s)}\left(\left\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{O}_i^{(s)}; \boldsymbol{\lambda}^{(s)})\right\rangle + b\right) \geq 1 - \xi_i^{(s)} \ \ \forall i\, \forall s \\
& \xi_i^{(s)} \geq 0 \ \ \forall i\, \forall s
\end{aligned} \tag{7.27}$$

where S is the number of target-speakers, $N^{(s)}$ is the number of (positive and negative) training examples for speaker s and $y_i^{(s)} = 1$ if $\boldsymbol{O}_i^{(s)}$ is from target-speaker s, otherwise $y_i^{(s)} = -1$. To allow a speaker-independent decision boundary to be trained, it is important that the dynamic features are dependent on the target speaker identity s. Ideally, the kernel function will map all utterances associated with the current target speaker to one region of the

feature space and all imposters to a different region, allowing a speaker-independent decision boundary to be learned. This is not the case for parametric kernels, where each utterance is mapped to a consistent point in the feature space, irrespective of the target identity. However, it is possible to obtain suitable features using generative models. A simple example of such a feature is the LLR between a model representing the current target-speaker and the UBM.

In this thesis, speaker-independent SVM classifiers are applied using derivative kernels. Given an enrollment utterance $\boldsymbol{O}^{(s)}$ associated with target speaker s, the derivatives of $\log p(\boldsymbol{O}^{(s)};\boldsymbol{\lambda})$ evaluated at the ML estimate $\boldsymbol{\lambda}^{(s)}$ over the enrollment data will be zero.

$$\nabla_{\boldsymbol{\lambda}} \log p(\boldsymbol{O}^{(s)};\boldsymbol{\lambda})\Big|_{\boldsymbol{\lambda}^{(s)}} = \mathbf{0} \tag{7.28}$$

Thus, by taking derivatives around a target-speaker dependent model $\boldsymbol{\lambda}^{(s)}$, the enrollment data associated with the current target speaker will be mapped to a consistent point (the origin). Note that equation 7.28 is only approximately true for test data, when multiple enrollment utterances are available per target speaker or when MAP adaptation is used to obtain $\boldsymbol{\lambda}^{(s)}$. When the feature space consists of (component-occupancy normalised) derivatives with respect to the mean-parameters of the target-speaker model and diagonal covariances are used, the features can be expressed as follows

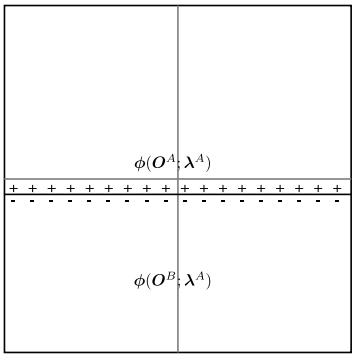
$$\begin{aligned}
\boldsymbol{\phi}(\boldsymbol{O}_i;\boldsymbol{\lambda}^{(s)}) &= \frac{1}{\sum_{t=1}^{T} P(\theta_t{=}m|\boldsymbol{o}_{it};\boldsymbol{\lambda}^{(s)})} \sum_{t=1}^{T} P(\theta_t{=}m|\boldsymbol{o}_{it};\boldsymbol{\lambda}^{(s)})\boldsymbol{\Sigma}_m^{(s)-1}(\boldsymbol{o}_{it}-\boldsymbol{\mu}_m^{(s)}) && (7.29)\\
&= \boldsymbol{\Sigma}_m^{(s)-1}\left[\boldsymbol{\mu}_m^{(i)} - \boldsymbol{\mu}_m^{(s)}\right] && (7.30)
\end{aligned}$$

where $\boldsymbol{\mu}_m^{(i)}$ is the utterance-dependent mean obtained by updating $\boldsymbol{\lambda}^{(s)}$ using a single iteration of EM to maximise the likelihood of $\boldsymbol{O}_i$.

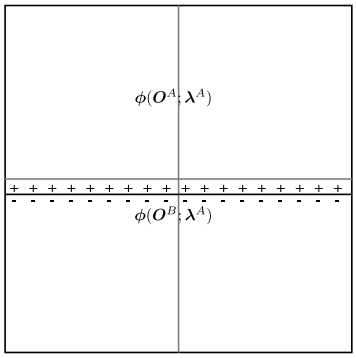
$$\boldsymbol{\mu}_m^{(i)} = \frac{\sum_{t=1}^{T} P(\theta_t = m|\boldsymbol{o}_{it};\boldsymbol{\lambda}^{(s)})\boldsymbol{o}_t}{\sum_{t=1}^{T} P(\theta_t = m|\boldsymbol{o}_{it};\boldsymbol{\lambda}^{(s)})} \tag{7.31}$$

The utterance-dependent mean $\boldsymbol{\mu}_m^{(i)}$ is an approximation to the speaker-independent features associated with a parametric kernel. Thus, derivative features associated with equation 7.30 can be interpreted as the (scaled) difference between an utterance-dependent term and a target-speaker dependent term. Hence the effect of using a target speaker-dependent generative model, is only to perform fixed translations, or positive scalings, of the feature space.

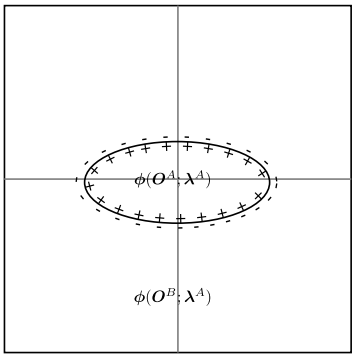
This presents an issue. When the only speaker-dependent effect of the kernel is to perform translations and positive scalings, it is not possible to learn a speaker-independent decision boundary using a linear static kernel. To explain this, consider the simplified case of two speakers A and B each with a single utterance of enrollment data. This is shown in figure 7.4 for two dimensional features. In the linear case, it is necessary to place a fixed decision boundary that classifies both utterances correctly when either A is the target speaker (figure 7.4(a)) or B is the target speaker (figure 7.4(b)). From the diagram it can be seen that it is not possible to place a decision boundary such that the utterances are separated and that the target speaker utterance lies on the positive side of the boundary in both cases. For example, in figure 7.4(b) both training examples are misclassified. Note that this is not an



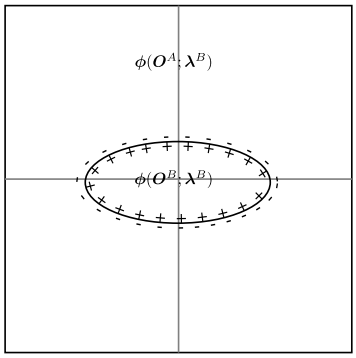
(a) Target speaker A (linear boundary) Correct classification



(b) Target speaker B (linear boundary) Incorrect classification



(c) Target speaker A (non-linear boundary) Correct classification



(d) Target speaker B (non-linear boundary) Correct classification

Figure 7.4: Speaker tying using a derivative kernel with a linear static kernel (a)(b) and non-linear static kernel (c)(d) for two speakers A and B. It is not possible to place a speaker-independent linear decision boundary that correctly classifies both (a) and (b). In (b) both examples are misclassified

issue when tying SVM parameters over multiple noise-conditions for ASR, as in [66, 67], since the label associated with each utterance is fixed irrespective of the noise-dependent kernel used. By using a non-linear decision boundary, obtained using a complex static kernel, a speaker-independent boundary can be used for both target speaker A (figure 7.4(c)) and target speaker B (figure 7.4(d)). This approach is dependent on the speaker-dependent models being a good representation of the speech of the target-speaker. When this is not the case, the positive training data will not be sufficiently aligned to allow a robust decision boundary to be trained.

An alternative approach to enable a linear speaker-independent classifier to be trained, is to modify the derivative features to introduce speaker-dependent rotations into the feature space. This may be achieved by using full-covariance matrices, however these are difficult to train for large models. An option that requires fewer speaker-dependent parameters is to introduce an explicit binary normalisation. Here each derivative is multiplied by either 1 or -1 dependent on the direction that the corresponding mean parameter of the target speaker-dependent model was adapted from the UBM.

$$\phi_m(\boldsymbol{O};\boldsymbol{\lambda}^{(s)}) = \text{sign}\left\{\boldsymbol{\mu}_m^{(s)} - \boldsymbol{\mu}_m^{\texttt{UBM}}\right\}\left(\nabla_{\boldsymbol{\mu}_m}\log p(\boldsymbol{O}^{(s)};\boldsymbol{\lambda})\Big|_{\boldsymbol{\lambda}^{(s)}}\right) \tag{7.32}$$

The effect of this to reflect all positive training examples into the same corner of the feature space enabling a linear decision boundary to be trained. An advantage of this approach over using non-linear static kernels is that, due to the linear decision boundary, it is less dependent on the speaker-dependent models being a good representation for the true speaker-dependent distributions. This may be useful, for example when MAP is used to obtain the speaker-dependent models.

7.5 Summary

In this chapter, the combination of dynamic and static kernels was examined. Since static kernels can not be applied directly to speech utterances, the combination approaches discussed in chapter 6 are unsuitable. Instead, alternative combination schemes must be applied. One such scheme, described in this chapter, is feature-level combination. Here, a fixed-dimensional feature vector is obtained from each utterance using a dynamic kernel. Then, instead of taking an inner product, a static kernel function is calculated between pairs of dynamic feature vectors.

An alternative combination scheme, observation-level combination, was also used in this chapter. Here a dynamic kernel function is evaluated on observations mapped into the feature space associated with a static kernel. It is not usually possible to train generative models in this space. Hence, many standard forms of dynamic kernel, including parametric and derivative kernels, can not easily be applied. However, by choosing a suitable metric, and using approximate component posteriors, these forms of dynamic kernel may be calculated. In this chapter, two forms of kernel were proposed based on observation-level combination. These respectively generalise derivative and parametric kernels when a linear static kernel is applied. Also, both generalise the GLDS kernel when a single component Gaussian is used as the generative model

The combination of static and dynamic kernels can potentially yield extremely high-dimensional feature spaces. Thus, when the amount of enrollment data available is limited, the classifiers may fail to generalise. This issue may be handled in several ways. For example, the dimension of the feature space may be constrained by using smaller generative models when applying high dimensional static kernels. Two alternative schemes were also presented, the first based on partitioning the enrollment utterances and the second based on tying the classifier parameters over multiple target speakers.

Chapter 8
Speaker Verification Experiments

This chapter presents experimental results for a standard speaker verification task. In chapters 5, 6 and 7 a number of kernel-based schemes were proposed for improving the performance of an SVM-based speaker verification system. In this chapter, these schemes are evaluated and results are presented. The chapter is organised as follows: First, the NIST 2002 SRE one-speaker detection task used to evaluated these approaches is described. Next, initial results are presented comparing the GMM and SVM-based approaches for speaker verification introduced in chapter 4. For the SVM-based system both parametric and derivative-based kernels, defined in chapter 6, are evaluated. These include several of the standard dynamic kernels for SV introduced in chapter 3.

In section 8.3, the variational dynamic kernels proposed in chapter 5 are evaluated. Two forms of variational kernel are implemented. The first is derived from the variational approximation to the KL divergence and the second is derived from the variational upper bound. These are evaluated under different training conditions and compared with standard forms of dynamic kernel derived from the matched-pair bound. Next, in section 8.4 the combination of multiple dynamic kernels is examined. Here, multiple SVM-based systems are combined at the kernel level, using the adapted maximum-margin based scheme for SV proposed in chapter 6. This kernel-based approach is then compared with a standard approach for combining systems, score-fusion, which uses a post-classifier to combine the output scores from multiple SV systems.

Finally, in section 8.5 two approaches are evaluated for combining dynamic kernels with traditional static kernels, such as the polynomial or Gaussian kernel. In the first approach,

dynamic feature vectors are obtained as usual. Then, rather than taking an inner product, a static kernel function is applied to the dynamic features. Data partitioning and speaker-tying schemes, proposed in chapter 7 to prevent over-training, are also evaluated. The second approach evaluated here involves applying a static kernel at the observation level. Although it is typically not possible to train a generative model in the feature space associated with a static kernel, such a model may be approximated. In chapter 7 this approach was used to derive two forms of kernel, the generalised derivative and the generalised parametric kernels. Both are evaluated in section 8.5.2.

8.1 NIST SRE 02 task

System performance is evaluated using the NIST SRE one-speaker detection task [143]. This task requires classifying utterances of conversational speech recorded over a cellular telephone channel and is the subject of annual NIST speaker recognition evaluations. The experiments described in the chapter were evaluated using data compiled for the 2002 evaluation. This consists of speech data taken from the switchboard-cellular phase 2 corpus. The 2002 dataset was chosen as it is one of the most recent evaluation datasets to be made generally available through the LDC. Later datasets are currently only available to SRE participants. The 2002 data has a number of disadvantages compared with more recent datasets. Most importantly, it only contains a single enrollment utterance per speaker. It is therefore not possible to apply standard approaches for reducing inter-session variability, such as NAP or WCCN, as described in chapter 4.

The one speaker detection task is the core condition in the 2002 SRE. Here each utterance only contains speech from a single speaker. The dataset consists of 330 target speakers (139 male and 191 female) each with a single utterance of enrollment data of around 120 seconds in duration. There are 3570 test utterances in total, each of known gender. The duration of the test utterances is more variable than that of the enrollment data. The mean duration of the test utterances is 32 seconds and 90% of the utterances are between 10 and 50 seconds in duration. The shortest test utterance is only 0.5 seconds long. Each of these test utterances is then scored against 11 potential speaker identities of the same gender, one of which is usually the true speaker. Thus in total the task consists of 39270 trials. Trials may contain speech uttered by speakers not present in the enrollment set. The correct response for these cases is to reject the utterance.

Each utterance of speech data provided by NIST consists of a single conversation side and is preprocessed using a silence detector to remove gaps. The utterances were then parameterised using a frame rate of 10ms and a window size of 30ms. A 31 dimensional feature vector was extracted from each frame using a bandwidth of 0-3.8 KHz. Low frequencies were retained as preliminary experiments suggested that they contained information useful for discrimination. The feature vector consisted of 15 static mel-PLP coefficients, 15 delta coefficients and the delta energy. Static energy or acceleration coefficients were not included since previous work [12] has shown that they contain little speaker-discriminant information. To introduce additional robustness to noise, cepstral feature warping [156], a form of short-term Gaussianisation described in chapter 4, was then applied to each utterance using a three second sliding window. PCA was also applied to decorrelate the observations.

The experimental results presented in this chapter are not intended to demonstrate a state-of-the-art SV system but to instead allow comparison between SV systems based on different

forms of dynamic kernel. For example, no attempt is made to select a suitable operating threshold. Instead performance figures are quoted using threshold-independent metrics. The primary performance metrics used in this chapter are equal error rate (EER) and detection error trade-off (DET) curves. To aid comparison with other work some minDCF scores are also quoted. These metrics are described in detail in section 4.7. Similarly, normalisation techniques, such as Z-norm or T-norm, described in section 4.6 were not applied. This was primarily due to a lack of suitable development data.

Statistical measures of significance are also quoted when comparing two similarly performing systems. These are calculated using McNemars's test, described in section 4.7.5, and represent the probability of obtaining a difference in performance at least as extreme as that observed given that the systems perform equally well. As discussed in section 4.7.5, the value of the measure associated with a given pair of systems is extremely dependent on the choice of threshold and will usually not hold for other points on the operating range. It is therefore necessary to explicitly specify the threshold used for each system. Since the EER is the main performance metric used to compare systems, all significance measures are quoted using the associated EER thresholds. These are calculated independently for each system. The normal approximation was used to calculate any P-values where the number of test examples misclassified by a single system was greater than 50.

8.2 Results for single dynamic kernels

In chapter 6 it was shown that many commonly used dynamic kernels can be placed into one of two broad families, derivative and parametric kernels. The exact nature of a parametric or derivative kernel is dependent on a number of factors including the form of the generative model, the choice of score space features, and the form of metric used. Initially, various forms of parametric and derivative kernel were evaluated and the effect of varying each of these factors examined.

8.2.1 Initial classifiers

An initial set of SVM and GMM-LLR based classifiers were trained as follows: gender-dependent UBMs were trained with a maximum-likelihood criterion using all SRE 2002 enrollment data of the appropriate gender. Each UBM consisted of a diagonal covariance GMM with a range of Gaussian components. As noted in chapter 4, GMMs have become the dominant form of generative model for SV and hence are used throughout these experiments. For each enrolled speaker, a speaker-dependent GMM was constructed by MAP adapting the means of the appropriate gender-dependent UBM using two iterations of static-prior MAP. Performing additional iterations was not found to yield gains in preliminary experiments. The appropriate UBM was chosen using gender information provided with the corpus. GMM training and adaptation was implemented using the HTK toolkit [216].

An initial baseline classifier (`GMM-LLR`) was formed by taking the log-likelihood ratio between the target speaker model and the UBM of the appropriate gender. The speaker-dependent models were also used as the generative models for a derivative kernel. Initially, the score space defined in equation 6.33 was used. This consisted of first order derivatives with respect to the GMM means only. Derivative kernels using more complex forms of score space are evaluated in section 8.2.3. This form of score space is equivalent to the standard

Fisher kernel [90], described in section 3.2.3, using a speaker-dependent model. Initially, each derivative feature was normalised by the total number of frames in the utterance. As in [67], standard deviation normalisation, rather than variance normalisation was used keep the dynamic range of each set of features consistent.

System	Description
$\texttt{GMM-LLR}_M$	GMM-based log-likelihood ratio classifier
∇_M	SVM classifier using derivative kernel
λ_M	SVM classifier using parametric kernel

M indicates #components of associated generative model (when present)

Table 8.1: Summary of initial classifiers.

To construct the parametric kernels, utterance-dependent GMMs were obtained by adapting the appropriate UBM means using two iterations of static-prior MAP. The task does not permit cross-gender trials so the gender of the utterance was known in all cases. Finally, for each utterance a parametric feature vector was constructed by concatenating the GMM means to give the score space defined in equation 6.32. This implementation of a parametric kernel is equivalent, under a maximally non-committal metric, to the standard GMM-supervector kernel [28] described in section 3.3.1.

SVM classifiers based on derivative (∇) and parametric (λ) kernels were trained using SVM^{light} [94]. The SVM regularisation term C was left at the SVM^{light} default, described in section 2.2.3. For each target speaker, an SVM classifier was trained using imposter examples obtained from the enrollment data associated with all other speakers of the same gender[1]. To reduce classifier bias each true utterance was duplicated until the two training sets were equal. For each kernel, a diagonal approximation to the maximally non-committal distance metric shown in equation 2.107 was implemented by normalising the global variance of each feature calculated using the training data for all target speakers. Spherical normalisation [205], described in section 2.2.6, was not applied since it was not found to yield performance gains during preliminary experiments. This differs from results previously reported in [205]. This difference may be related to the value of C used here. At the SVM^{light} default value, a (linear) SVM classifier is invariant to global scalings of the feature vectors. As described in section 8.1, the 2002 SRE task contains only a single enrollment session per speaker. Thus it was not possible to implement techniques such as NAP or WCCN to compensate for inter-session variability.

8.2.2 Effect of the generative model

Initially, the effect of varying the value of $\tau^{\texttt{map}}$ used during model adaptation was evaluated. Table 8.2 shows the performance of the GMM-LLR and derivative systems when 1024 component generative models were used. For all values of $\tau^{\texttt{map}}$ the SVM classifier outperformed the baseline GMM-LLR classifier. These results were statistically significant at a level of 99%. However, as the value of $\tau^{\texttt{map}}$ increased the performance gain of the SVM classifier over

[1]The setup used did not conform to the NIST SRE protocol, since enrollment data was used for both UBM training and imposter modelling. This was necessary due to a lack of suitable development data. Typically imposter data is obtained from an auxiliary dataset, recorded under similar conditions.

the baseline gradually increased (note the performance of the GMM-LRR classifier initially improved and then got worse). This can be explained as the SVM decision boundary is estimated in the score space defined by the MAP-adapted generative model. If the distribution associated with the model is too "close" to the training example, a "biased" score space, and scores, compared to the test data will be obtained. This bias is present for both the GMM-LLR and derivative systems, but it affects the performance more in the SVM case because of the large dimension of the score space. Thus the best value of $\tau^{\texttt{map}}$ is expected to be smaller for the GMM-LLR system than the derivative system, as seen in table 8.2. When $\tau^{\texttt{map}} = 0$ the parameter estimates for the speaker models are based only on the maximum-likelihood estimate given the adaptation data, this is similar to the approach used in [205].

$\tau^{\texttt{map}}$	$\texttt{GMM-LLR}_{1024}$		∇_{1024}	
	EER(%)	minDCF	EER(%)	minDCF
0	11.67	0.4664	9.42	0.4236
10	10.13	0.4289	7.88	0.3600
25	10.86	0.4514	7.60	0.3398
50	11.83	0.4947	7.65	0.3311
∞	-	-	7.98	0.3433

Table 8.2: Performance for 1024-component log-likelihood ratio ($\texttt{GMM-LLR}_{1024}$) and derivative kernel (∇_{1024}) systems using target speaker models adapted with varying $\tau^{\texttt{map}}$.

$\tau^{\texttt{map}}$	λ_{128}		λ_{1024}	
	EER(%)	minDCF	EER(%)	minDCF
0	8.54	0.3561	8.00	0.3413
1	8.32	0.3455	8.47	0.3488
5	8.58	0.3545	9.42	0.3976
10	9.05	0.3755	10.11	0.4597

Table 8.3: Performance of parametric kernel systems using 128 (λ_{128}) and 1024 (λ_{1024}) component GMMs for different values of $\tau^{\texttt{map}}$.

Results for the parametric SVM-classifier are detailed in table 8.3. Initially, classifiers based on 128 and 1024-component models were evaluated. The best performance was 8.00% EER, achieved using 1024-components when $\tau^{\texttt{map}} = 0$. When $\tau^{\texttt{map}}$ was set to 1 or greater, the 128-component system performed best. Additionally, for all model sizes evaluated, the optimal value of $\tau^{\texttt{map}}$ was lower than for the derivative kernel and the log-likelihood ratio classifier. This is because for the parametric kernel, when $\tau^{\texttt{map}}$ is large (or the number of observations per component is low) the parameters will not be adapted far from the UBM. Hence the dynamic features will be similar for all utterances. When a maximally non-committal metric is used, the resultant features will not be robust since small differences caused by the numerical accuracy will be magnified by the metric. This is not the case for the derivative kernel. Here as $\tau^{\texttt{map}}$ approaches infinity the features tend to resemble derivatives with respect to the parameters of the UBM. This is why, unlike the GMM-LLR or parametric systems, it is possible to evaluate a derivative kernel-based system at the point $\tau^{\texttt{map}} = \infty$. For the

parametric kernel, best performance at 1024 components was achieved with $\tau^{\texttt{map}} = 0$, equivalent to estimating the parameters of the utterance-dependent models using an ML criterion. However, as discussed in section 6.3.1, when large models are used there will generally not be enough observations per component to robustly estimate the parameters. Thus, unless indicated otherwise, for the remainder of the experiments the adaptation constant was fixed at 1 for parametric kernels. This avoided the issue of undefined feature values when components were unobserved in the adaptation data.

# Components	GMM-LLR			Derivative (∇)		Parametric (λ)	
	$\tau^{\texttt{map}}$	EER(%)	minDCF	EER(%)	minDCF	EER(%)	minDCF
128	25	12.04	0.4911	8.62	0.3775	8.32	0.3455
256	10	10.87	0.4503	8.08	0.3457	8.18	0.3302
512	10	10.23	0.4345	7.92	0.3375	8.17	0.3256
1024	10	10.13	0.4290	7.60	0.3398	8.47	0.3488

Table 8.4: Comparison of log-likelihood ratio, derivative and parametric system performance at various model sizes. For the log-likelihood ratio system, the optimal $\tau^{\texttt{map}}$ used is indicated. For derivative and parametric kernel systems optimal $\tau^{\texttt{map}}$ for all model sizes was 25 and 1 respectively.

Table 8.4 compares the performance of the baseline GMM-LLR classifier with the derivative and parametric SVM-based systems over a range of model sizes. The value of $\tau^{\texttt{map}}$ was fixed independently for each system to minimise the equal error rate. For parametric systems, only values of $\tau^{\texttt{map}} > 0$ were considered to ensure the features were robust. The two SVM-based systems showed large gains over the baseline GMM-LLR system for all sizes of model. This is consistent with previous work such as [28] and [205]. The performance of the GMM-LLR system only improved marginally as the number of components was increased from 256 up to 1024. Similarly, the performance of the parametric system did not improve significantly for larger model sizes. However the derivative kernel continued to show gains as the number of components increased. At 1024 components, the derivative kernel even outperformed the best parametric kernel obtained by setting $\tau^{\texttt{map}} = 0$. This result was statistically significant at a level of %. This discrepancy is a consequence of the mismatch between train and test utterance duration present in the 2002 SRE data. As described in section 6.3, only the parametric kernel requires adapting the UBMs using test data. Thus the parametric kernel is more sensitive to the shorter duration of the test utterances. Although the results in Table 8.4 suggest for the derivative system, the use of model sizes greater than 1024 might yield further gains, this was not evaluated due to the computational cost of evaluating this system for large numbers of components. The optimal number of components for the parametric system was 512. This was significantly less than reported for equivalent GMM-supervector kernels, for example in [27], which typically use up to 2000 Gaussian components. This difference is likely to be related to the limited amount of data used for UBM training.

Figure 8.1 compares the overall performance of the best GMM-LLR system with the best parametric and derivative kernel-based SVM systems obtained by varying the generative model structure and parameter estimation schemes. Both SVM systems significantly outperformed the GMM-LLR over the entire operating range. Overall, the best performing system was the 1024 component derivative system which gave an EER of 7.60%. It can be seem

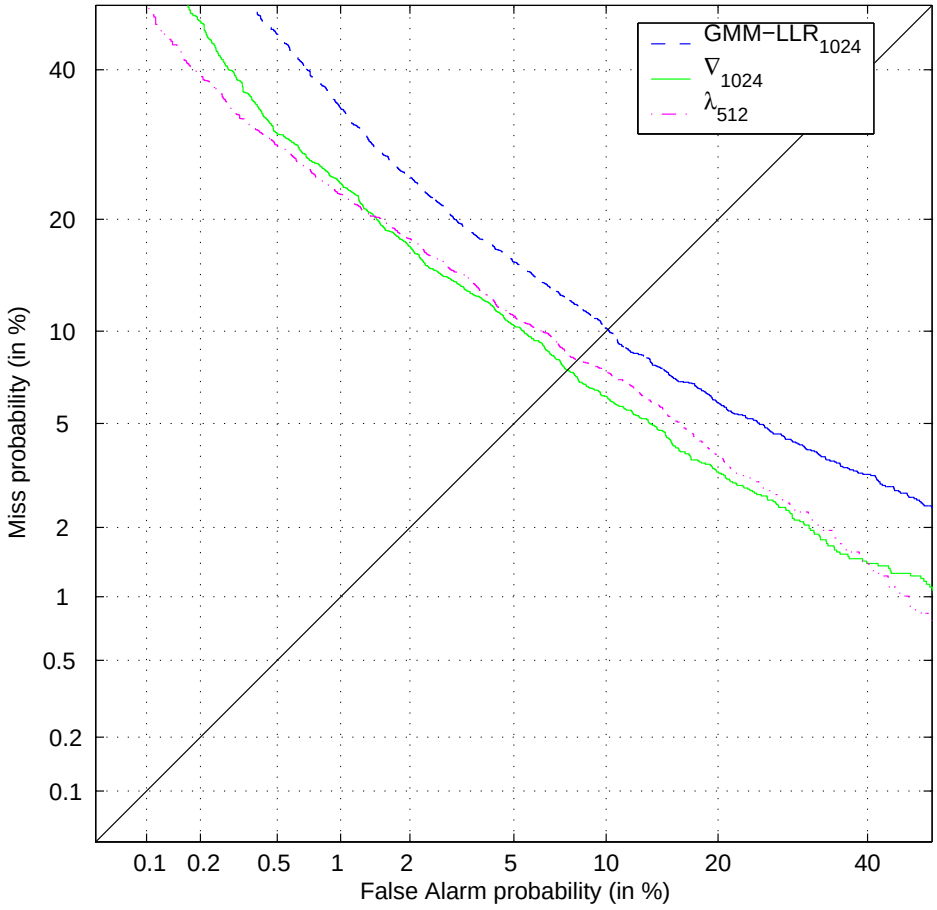


Figure 8.1: Comparison of best performing LLR (`GMM-LLR`), derivative (∇_{1024}) and parametric (λ_{512}) SVM systems.

from figure 8.1 that this kernel outperformed the 512-component parametric kernel over the majority of the operating range. However at the limits, the parametric kernel performed best.

8.2.3 Effect of the score space

The score space associated with a derivative kernel may contain more complex features. Table 8.5 shows the effect of including in the score space derivatives with respect to the means ($\boldsymbol{\mu}$), variances ($\boldsymbol{\Sigma}$) and mixture weights ($\boldsymbol{c}$) of the target speaker model and of the UBM. 128 and 1024 component GMMs were used and in each case the speaker model was trained by adapting the means only with $\tau^{\text{map}} = 25$. The final score space was obtained by concatenating the set of derivatives. This approach is closely related to the equal weight dynamic kernel combination described in chapter 6. Note that although the target speaker model and UBM variances and mixture weights were identical, the derivative features associated with each differ due to the varying component occupancies caused by the adaptation of the mean parameters.

# Components	Score space	#Features	EER(%)	minDCF
128	$\boldsymbol{\mu}$	3968	8.62	0.3775
128	$\boldsymbol{\mu} + \boldsymbol{\Sigma}$	7936	8.94	0.3544
128	$\boldsymbol{\mu} + \boldsymbol{\Sigma} + \boldsymbol{c}$	8064	9.04	0.3547
128	$\boldsymbol{\mu}^{\text{ubm}}$	3968	9.46	0.3894
128	$\boldsymbol{\mu}^{\text{ubm}} + \boldsymbol{\Sigma}^{\text{ubm}}$	7936	9.53	0.3972
128	$\boldsymbol{\mu}^{\text{ubm}} + \boldsymbol{\Sigma}^{\text{ubm}} + \boldsymbol{c}^{\text{ubm}}$	8064	9.48	0.3980
128	$\boldsymbol{\mu} + \boldsymbol{\mu}^{\text{ubm}}$	7936	8.65	0.3749
128	$\boldsymbol{\mu} + \boldsymbol{\mu}^{\text{ubm}} + \boldsymbol{\Sigma} + \boldsymbol{\Sigma}^{\text{ubm}}$	15872	8.86	0.3688
128	$\boldsymbol{\mu} + \boldsymbol{\mu}^{\text{ubm}} + \boldsymbol{\Sigma} + \boldsymbol{\Sigma}^{\text{ubm}} + \boldsymbol{c} + \boldsymbol{c}^{\text{ubm}}$	16128	8.94	0.3704
1024	$\boldsymbol{\mu}$	31744	7.60	0.3398
1024	$\boldsymbol{\mu} + \boldsymbol{\Sigma}$	63488	8.40	0.3444
1024	$\boldsymbol{\mu} + \boldsymbol{\Sigma} + \boldsymbol{c}$	64512	8.38	0.3429
1024	$\boldsymbol{\mu}^{\text{ubm}}$	31744	7.98	0.3433
1024	$\boldsymbol{\mu}^{\text{ubm}} + \boldsymbol{\Sigma}^{\text{ubm}}$	63488	8.48	0.3782
1024	$\boldsymbol{\mu}^{\text{ubm}} + \boldsymbol{\Sigma}^{\text{ubm}} + \boldsymbol{c}^{\text{ubm}}$	64512	8.51	0.3747
1024	$\boldsymbol{\mu} + \boldsymbol{\mu}^{\text{ubm}}$	63488	7.75	0.3325
1024	$\boldsymbol{\mu} + \boldsymbol{\mu}^{\text{ubm}} + \boldsymbol{\Sigma} + \boldsymbol{\Sigma}^{\text{ubm}}$	126976	8.38	0.3559
1024	$\boldsymbol{\mu} + \boldsymbol{\mu}^{\text{ubm}} + \boldsymbol{\Sigma} + \boldsymbol{\Sigma}^{\text{ubm}} + \boldsymbol{c} + \boldsymbol{c}^{\text{ubm}}$	131072	8.35	0.3569

Table 8.5: Performance for 128 and 1024 component derivative kernels using different score space features. Features include derivatives with respect to target speaker model means ($\boldsymbol{\mu}$), variances ($\boldsymbol{\Sigma}$) and component priors ($\boldsymbol{c}$) and UBM means ($\boldsymbol{\mu}^{\text{ubm}}$), variances ($\boldsymbol{\Sigma}^{\text{ubm}}$) and component priors ($\boldsymbol{c}^{\text{ubm}}$).

The inclusion of either mixture weight or variance features was found to degrade performance. This was also the case when 128-component models were used, indicating that this degradation was not simply due to the increased complexity of the feature space. Derivatives with respect to the UBM means ($\boldsymbol{\mu}^{\text{ubm}}$), variances ($\boldsymbol{\Sigma}^{\text{ubm}}$) and mixture weights ($\boldsymbol{c}^{\text{ubm}}$) were also used. For both 128 and 1024 component models performance was not significantly worse than

derivatives with respect to the target speaker model for all combinations of parameters, indicating that derivatives with respect to the UBM still contained useful speaker-discriminative information. When derivatives with respect to the target speaker model and the UBM were combined, gains were only observed when combining derivatives with respect to variances and mixture-weights. In chapter 7, it was shown that the generative model induces an approximately linear translation of the mean derivatives. Since (linear) SVMs are invariant to translations of the feature space, this may explain why combination of mean derivatives (associated with the same model structure) did not yield gains.

Parametric kernels with different score spaces were also evaluated. The inclusion of variance features in the score space has previously been shown to lead to gains on tasks such as language identification [26]. Unlike derivative kernels, for parametric kernels it is necessary to adapt all parameters to be included in the score space. For 512-component models, including both means and variance parameters in the parametric score space lead to an EER of 10.60%. This represented an absolute reduction in performance of 2.43% compared to the mean-only score space. This performance loss is likely to due to the limited amount of data available for variance adaptation in the 2002 SRE task, especially for the test set. Parametric kernels based on component-prior features were not evaluated. This was because there was insufficient adaptation data to robustly re-estimate the priors and maintain the sum-to-one constraint.

Normalisation	# Components			
	128		1024	
	EER (%)	minDCF	EER (%)	minDCF
Total frames	8.62	0.3775	7.60	0.3398
Component occupancy	8.35	0.3704	7.81	0.3297

Table 8.6: Duration normalisation for 128 and 1024-component derivative kernels. Derivatives were normalised by either the total utterance duration or the associated component occupancy.

For derivative kernels it is important that the dynamic features include some form of duration normalisation, particularly when the dataset consists of utterances that vary greatly in duration. This is not necessary for parametric kernels since each feature is implicitly normalised by the component occupancy. In table 8.6 results are given for 128 and 1024-component derivative kernels using different forms of duration normalisation. All models were adapted with $\tau^{map} = 25$ and derivatives were taken with respect to the speaker-means only.

For 128 components best performance was achieved using component occupancy normalisation. This was due to the more accurate duration normalisation yielding a more consistent set of features. However, this was not the case when 1024-components were used. Here, the large number of components meant that many components were rarely observed in the data. Thus when component occupancy normalisation was used the resultant dynamic features were not robust. For both evaluated model sizes test performance was poor ($> 30\%$ EER) using unnormalised features. This was due to both the variation in the duration of the test utterances and the disparity in duration between the training and test sets in the NIST 2002 task. Test utterances are on average only a quarter the average length of the training utterances.

For consistency, in the remainder of this chapter derivative kernels are primarily normalised by the number of frames. The exception is in section 8.5, in which 16 and 128-component derivative kernels are combined with static kernels. Due to the smaller generative models used, there component occupancy normalisation is used instead.

8.2.4 Effect of the metric

The decision function associated with an SVM classifier is not invariant to scaling of the input features. Thus the selection of a suitable metric is important when using dynamic kernels. In table 8.7 results are presented for derivative and parametric kernels using a variety of metrics. For both parametric and derivative kernels, the score space consisted of only features based on the means. For the derivative kernels, these features were normalised by the standard deviation rather than the variance.

System	Metric	EER(%)	minDCF
∇_{1024}	Identity	8.23	0.3927
	Speaker-dependent	7.88	0.4061
	Speaker-independent	7.60	0.3398
λ_{512}	Identity	8.22	0.3264
	Speaker-dependent	8.23	0.3241
	Speaker-independent	8.17	0.3256
	GMM-supervector	8.33	0.3280

Table 8.7: Derivative and parametric system performance using (a) an identity metric, (b) a diagonal approximation to a maximally non-committal metric estimated independently for each speaker and (c) the same metric estimated globally over all speakers. Parametric kernel performance using a GMM-supervector equivalent metric is also included.

The performance of the derivative and parametric kernels was similar when an identity metric was used Applying a speaker-dependent maximally non-committal metric lead to an 0.35% reduction in EER for the derivative kernel. However, for the parametric kernel no gains were observed. This difference may be related to the fact that, for the derivative kernel, the maximally non-committal metric normalises each feature by the Fisher information associated with the parameter. This is not the case for parametric features.

Since the complete dataset is unknown, an approximation to a maximally non-committal metric must be estimated from the training data, as described in chapter 3. This may be estimated either globally or independently for each target speaker. For both parametric and derivative kernels best performance was obtained using a global metric, due to the more robust parameter estimates. It is possible to incorporate a-priori knowledge through the use of a suitable metric, this is the case for the GMM-supervector kernel. In contrast to results described in [103], here the use of a GMM-supervector metric was not found to provide gains.

GMM mean-based derivative features are typically normalised by the component variance. As discussed in section 3.2.3, standard deviation normalisation may instead be used to yield a more consistent dynamic range for each dimension of the feature-space. When a maximally non-committal metric is used, the form of variance normalisation chosen does not affect the decision since the effects are removed by the metric. However, for other forms of metric this

Normalisation	EER(%)	minDCF
Variance	10.56	0.4341
Standard deviation	8.23	0.3927
Maximally non-committal	7.60	0.3398

Table 8.8: Variance normalisation for 1024-component mean-based derivative features using an identity metric. Derivatives were normalised using either the component variance or standard deviation. System performance using a maximally non-committal metric is also shown.

is not the case. Table 8.8 compares the performance of derivative kernels using variance and standard deviation normalised features. Both forms of kernel were defined using an identity metric and 1024-component GMMs were used. For the standard deviation normalised features, the variance of the training feature vectors was generally consistent in each dimension. Greater variation was observed when using variance normalised features. This was reflected in the performance of the kernels. Here standard deviation-based normalisation yielded an absolute improvement of 2.33% EER compared to variance-based normalisation. However, best overall performance was obtained when a maximally non-committal metric was used.

8.3 Results for variational kernels

An alternative to parametric and derivative kernels are the variational dynamic kernels proposed in chapter 5. Many standard forms of dynamic kernel, such as the linear and non-linear GMM-supervector kernels introduced in chapter 3, are derived using a matched-pair bound to the KL-divergence between two distributions. For variational kernels, a variational approximation is used instead. This has a number of advantages. First, a variational approximation will typically be closer to the true KL-divergence. Thus the kernel function obtained will more accurately reflect the underlying KL-divergence between distributions, potentially yielding gains. A further disadvantage to using parametric and derivative kernels is that they require that model components are coordinated in order to obtain a consistent feature space. This places restrictions on the parameter estimation schemes that may be used. For example, to apply the GMM-supervector kernel all utterance-dependent models must be adapted from a single background model. When speakers cluster, for example within gender, this may not be optimal.

Variational kernels do not suffer from this restriction. Moreover, the distributions are not even restricted to have the same form. For example the number of mixture components may be varied depending on the utterance duration. Variational kernels do have a number of disadvantages. Unlike many parametric and derivative kernels, the kernel function does not have an explicit feature space. Thus it is not possible to compact the speaker models to improve test efficiency. Additionally, in the worst case evaluating a variational kernel function between two M-component GMM distributions requires $\mathcal{O}(M^2)$ operations compared to $\mathcal{O}(M)$ for the parametric and derivative kernels evaluated in the previous section.

8.3.1 KL divergence approximations

Initially the different KL divergence approximations were compared. For each training and test speech utterance, a GMM distribution was adapted from a 512-component gender-dependent UBM. Two iterations of static-prior mean-only MAP were applied with $\tau^{\texttt{map}} = 1$. Thus, the utterance-dependent distributions were identical to those used in 8.2 for the 512-component parametric kernel. Figure 8.2 shows the distribution of the log-relative deviation between various approximation schemes and a reference estimate of the true KL divergence. This estimate was obtained using the Monte-Carlo approach described in [87] using 10,000 samples[1]. The log-relative divergence is defined by

$$R(f_i, f_j) = \log \frac{|KL(f_i||f_j) - KL^{\texttt{monte-carlo}}(f_i||f_j)|}{KL^{\texttt{monte-carlo}}(f_i||f_j)} \tag{8.1}$$

The distribution of R was estimated by calculating the divergence between all 108,570 pairwise combinations of models estimated from the enrollment utterances, including cross-gender pairs. Figure 8.2 does not include divergences of the form $R(f_i, f_i)$. For both the variational approximations and the matched-pair bound, the true KL-divergence (0) is obtained in these cases.

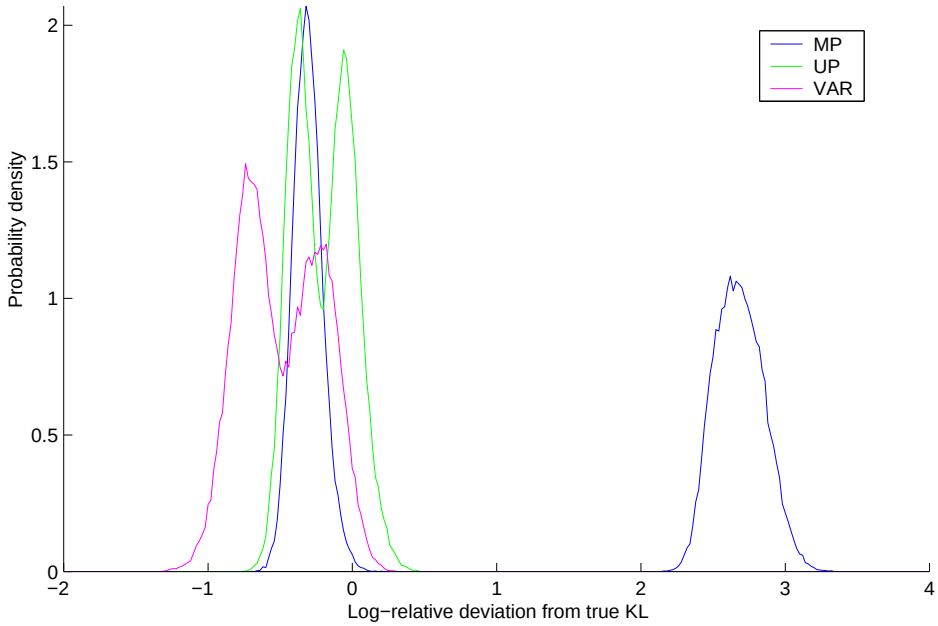


Figure 8.2: Log-relative deviation between KL-approximations and 'true' divergence over all pairs of enrollment utterances for matched-pair bound (`MP`),variational upper bound (`UP`) and variational approximation (`VAR`).

It is important to establish whether an estimate of the KL-divergence obtained using Monte-Carlo sampling with 10,000 samples is sufficiently accurate to be considered a true representation of the underlying KL-divergence. Figure 8.3 shows the log-relative deviation

[1] An alternative sampling approach that can be used to estimate the true KL divergence is the unscented transform [96]. This was not used here since previous work [87] has indicated that, for a given number of samples, Monte-Carlo sampling can yield a more accurate estimate of the true divergence.

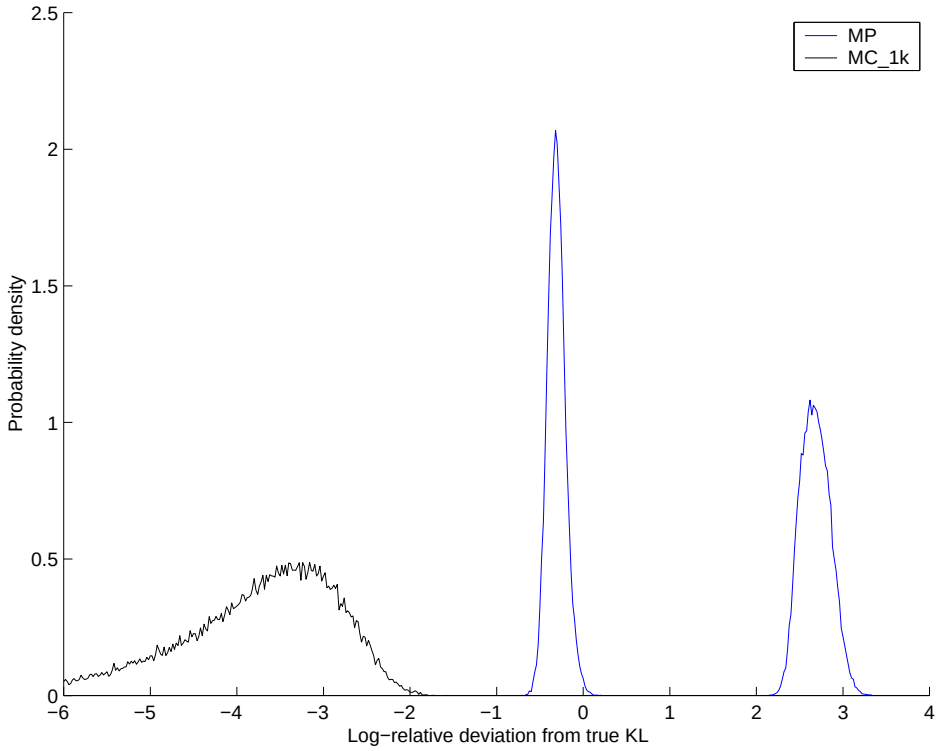


Figure 8.3: Log-relative deviation between Monte-Carlo estimate of divergence using 1,000 samples (`MC_1k`) and 'true' KL divergence (10,000 samples) compared with matched-pair bound (`MP`).

of a Monte-Carlo estimate using 1,000 samples (MC_1k) from the 'true' (10,000 samples) divergence. When 1000 samples are used, each component of $f_i(\boldsymbol{o})$ will only be sampled twice on average. Despite this, the divergence between the two sampling approaches was small, particularly compared to the matched-pair bound. Since the samples were drawn independently in each case, this suggests that 10,000 samples is sufficient to provide an extremely close estimate of the true KL-divergence.

The accuracy of the matched-pair bound (MP), variational approximation (VAR), and variational upper bound (UP) is shown in figure 8.2. Each approximation was calculated as described in chapter 5. For the variational upper bound, all variational parameters were initialised to the product of the respective component priors, $q_{m|n} = v_{n|m} = c_n c_m$. 15 iterations of re-estimation were then applied, which was sufficient for the variational parameters to converge.

The true KL-divergence associated with cross-gender comparisons was approximately twice that of within-gender comparisons. This is because each utterance was adapted from a gender-dependent UBM, which in turn were trained independently. For within-gender utterances this lead to a strong coordination between pairs of Gaussian components. For the variational upper bound the optimal variational parameters were sparse and consistently approached the matched-pair solution. This was true to a lesser extent for the variational approximation which generally provided a closer approximation to the true KL-divergence.

For cross-gender utterances there was greater variation in the accuracy of the approximations. This difference is reflected in figure 8.2. The distribution of $R(f_i, f_j)$ using the matched-pair bound and the two variational approximations is clearly bimodal. The matched-pair bound was worst, typically exceeding the true KL-divergence by an order of magnitude. In contrast, for the two variational approximations the accuracies of the cross-gender-and within-gender comparisons were more similar. This indicates that the two variational approximations were able to learn the structure of any coordination between components more effectively than the matched-pair bound. The remaining discrepancy between cross and within-gender accuracy is likely to be due to the error in the variational parameter estimates. This is smaller for the within-gender case since most of the variational parameters lie on the boundary of the ≥ 0 constraint. Hence the 'true' value of the parameter cannot be less than the estimate. This is not an issue for the Monte-Carlo estimate since this approach does not require the estimation of any distribution parameters.

8.3.2 Variational kernels using a single background model

Next, the variational kernels were evaluated. For each pair of utterances, $\boldsymbol{O}_i$ and $\boldsymbol{O}_j$, with corresponding GMM-distributions, f_i and f_j, the variational kernel functions defined in equations 5.28 and 5.29 were applied. The constant α was fixed at $\alpha = 1$ for all kernels. During preliminary experiments this was found to optimise performance when the matched-pair bound was used. Kernels were evaluated based on the matched-pair bound (MP), variational (VAR) and variational upper bound (UP) approximations. For the variational upper bound, evaluating the kernel function requires independently optimising four sets of variational parameters. The same initialisation and re-estimation schemes were used to obtain the variational parameters as in figure 8.2. For comparison, a 512-component mean-based parametric system (λ_{512}) was also evaluated, trained as described in section 8.2. Since the variational kernels do not have explicit feature spaces, it is not possible to combine them with a maximally non-committal metric. To determine whether differences in performance were due to the lack

of a maximally non-committal metric, a final parametric kernel (GMM-SV) was also evaluated, based on the GMM-supervector metric described in chapter 3. As shown in chapter 5, this kernel only differs from (MP) by the method used to derive the kernel from the matched-pair bound, polarisation for GMM-SV and exponentiation for MP. Performance figures for these kernels are presented in table 8.9. The corresponding DET curves are shown in figure 8.4. As in section 8.2 SVM imposter examples were obtained from all enrollment utterances of the same gender as the target speaker. Since the 2002 SRE task contains no cross-gender trials this experimental setup does not require cross-gender kernel evaluations.

Kernel	EER(%)	minDCF
λ_{512}	8.17	0.3256
GMM-SV	8.35	0.3286
MP	9.91	0.3989
UP	9.89	0.3984
VAR	9.68	0.3849

Table 8.9: Performance of variational kernels against reference GMM-supervector (GMM-SV) and parametric system (λ_{512}). All systems were gender-dependent and used 512 component GMMs.

The performance of the parametric kernel was significantly better than that for the exponential kernels, despite using the same GMM models in all systems. From figure 8.4 it can be seen that this gain extended over the entire operating range. This difference was not simply due to the use of a maximally non-committal metric, since the GMM-supervector kernel also yielded substantially better performance than the exponential kernels. This agrees with results reported in [48]. There, gains were achieved by normalising the matched-pair estimate of the divergence between the background model and each GMM, a process described in section 3.3.2. However, this is non-trivial for the variational kernels.

Overall the three kernels based on exponential KL approximations performed at a roughly similar level. The best performing kernel was based on the variational approximation with an EER of 9.68%. The improvement in performance of this kernel relative to the matched-pair kernel was statistically significant at a level of 99%. Since the experimental setup did not require cross-gender kernel evaluations, these results are in line with the accuracies in figure 8.2.

8.3.3 Variational kernels using multiple background models

One advantage of using variational kernels is that they do not require all distributions to be adapted from the same background model. In section 8.2 it was shown that the optimal τ^{map} for the parametric kernel is extremely small, indicating that it is more important that the distributions closely reflect the underlying speech than that the parameter estimates are particularly robust. Similarly, if an utterance-dependent distribution is adapted from a background model that lies 'closer' to the true distribution of speech, the resultant distributions is also likely to provide a more accurate representation of the underlying speech. Thus, learning clusters of 'close' speakers, and adapting a cluster-dependent background model may potentially leading to gains. Here a simplified version of this approach is evaluated, where utterances are adapted from one of two background models dependent on speaker gender.

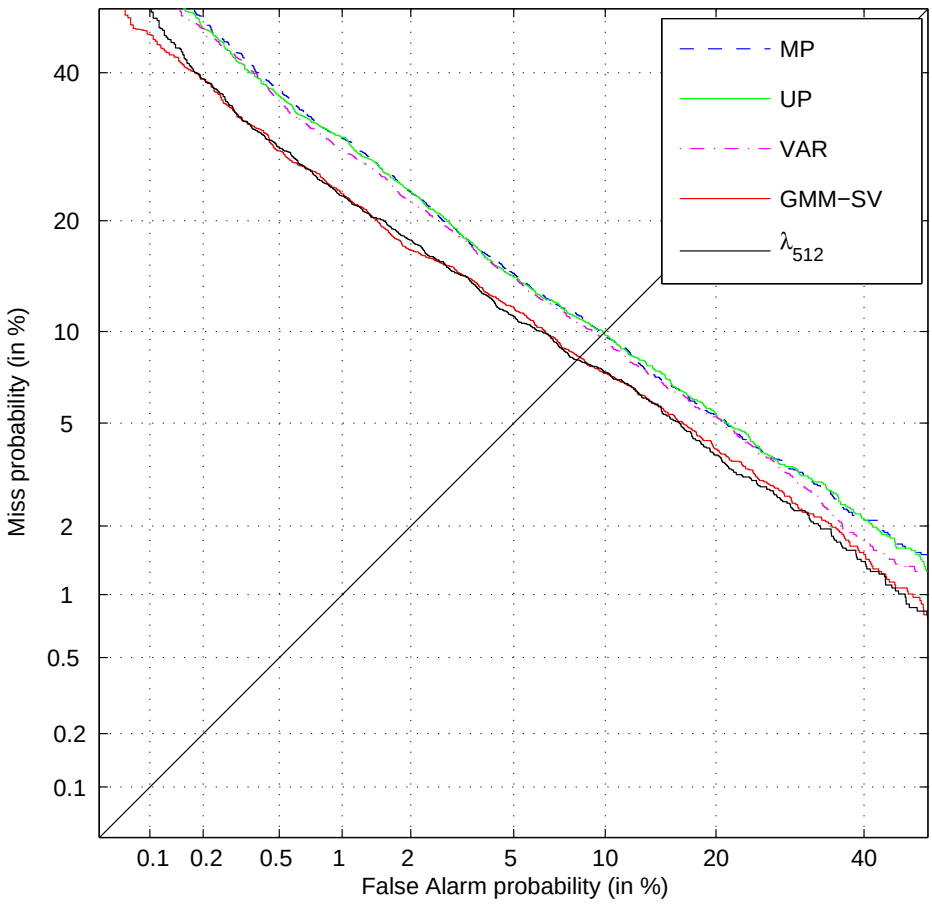


Figure 8.4: DET performance for the kernels in table 8.9. All systems were gender-dependent and used 512 component GMMs.

Unlike the results in section 8.3.2, during training both cross and within-gender enrollment utterances are used as SVM imposter examples.

System	Background Model	SVM Imposters	Equal Error Rate (%)			
			λ_{512}	MP	VAR	UP
GD	Gender-dependent	Gender-dependent	8.17	9.91	9.68	9.89
GD-UBM		All	8.17	10.29	9.59	9.69
GI	Single	All	8.99	10.11	9.91	10.05

Table 8.10: Kernel performance using gender-dependent UBMs and imposter data (GD), gender-dependent UBMs and gender-independent imposter data (GD-UBM) and gender-independent UBM and imposter data (GI).

Results for this experimental setup, (GD-UBM), are shown in table 8.10 for various forms of kernel function. Results for the strictly gender-dependent setup (GD) used in table 8.10 are also provided. For all systems minDCF results were in line with reported EERs. For the matched-pair kernel, including cross-speaker imposter data degraded performance by 0.42% EER, indicating that the inclusion of additional imposter data simply introduces noise into the training set. The fact that this loss is relatively small, and the performance of the parametric kernel did not degrade, is due to the ability of the SVM to select appropriate support vectors. For the matched-pair kernel, only 7% of imposter support vectors came from cross-gender speakers. Similarly, for the parametric kernel only 3% of imposter support vectors were from cross-gender speakers. In both cases, the cross-gender support vectors were generally among the lowest weighted.

For both variational kernels, small performance gains were observed. To establish whether this gain was simply due to the additional imposter data, a third experimental setup (GI) was also evaluated. Here all distributions were adapted from a single gender-independent background model. Again, imposter data from both genders was used for each target speaker. For all kernels evaluated this system performed worst. This degradation was primarily because the utterance-dependent distributions were less well adapted to the speech. Unfortunately, the lack of cross-gender trials in the 2002 NIST SRE meant that the gains that may be obtained by including cross-gender imposter data are limited. The use of a larger number of background models, for example trained via speaker clustering, may potentially yield additional gains over those reported here.

Figure 8.5 compares the best performances obtained for each form of kernel. Best overall performance was 8.17% EER using a parametric kernel with either GD or GD-UBM setups. This gain extended over most of the operating range. Although the performance of the exponential kernels was comparatively poor, the variational kernels generally outperformed the matched-pair kernel, particularly as the operating threshold is decreased.

8.4 Results for dynamic kernel combination

There has been a recent trend towards combining multiple classifiers to improve overall system performance. This is typically implemented by scoring each utterance using a range of classifiers and then combining the output scores. For SVM classifiers, an alternative approach,

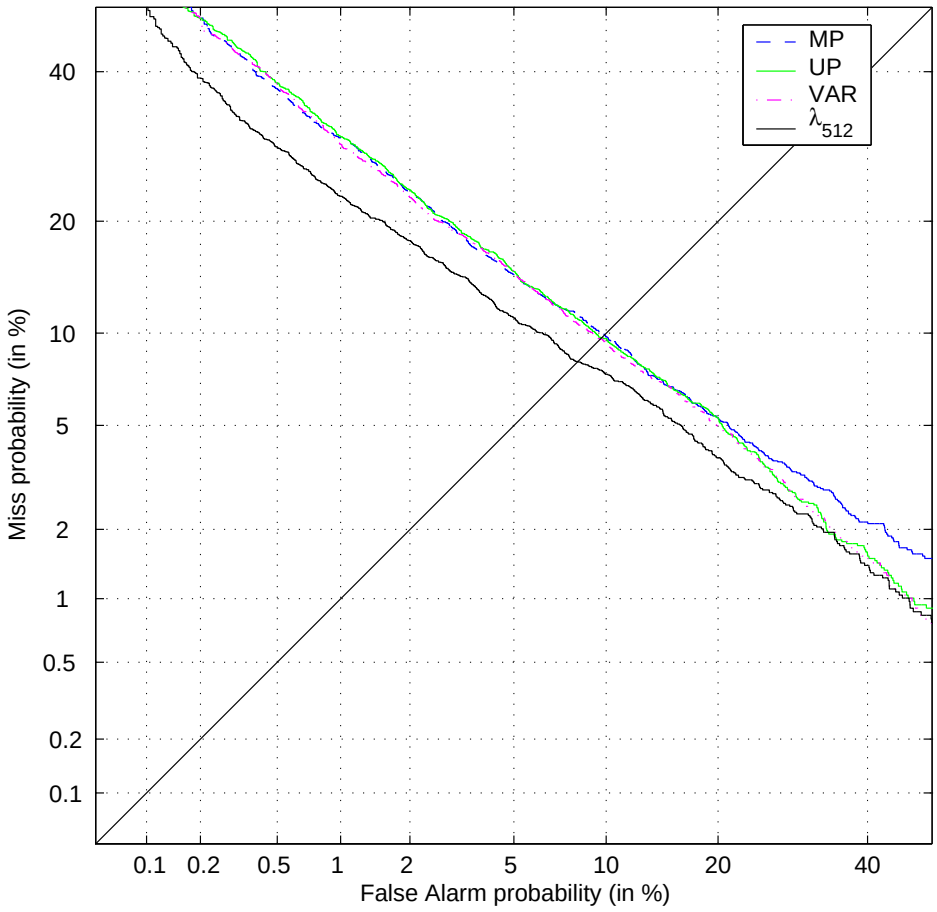


Figure 8.5: Comparison of best performing kernels. Parametric (λ_{512}) and matched-pair bound kernels used gender-dependent imposters. The variational (`VAR`) and variational upper bound(`UP`) kernel used imposters of both genders. All systems used gender-dependent UBMs.

described in chapter 6, is to combine at the kernel level. Here this approach is evaluated and compared with a traditional score-fusion approach.

The primary forms of kernel combined in this section are derivative and parametric kernels. Due to their relatively poor performance and less efficient training schemes, combinations of variational kernels are not considered here. As in section 8.2, GMM-based models were used based on a range of model sizes. Each derivative kernel consisted of derivatives with respect to the mean parameters of a speaker-dependent model. This model was adapted from a gender-dependent background model with $\tau^{\texttt{map}}$ set at 25. For parametric kernels, utterance-dependent distributions were also adapted from gender-dependent background models with $\tau^{\texttt{map}}$ fixed at 1. Like the derivative kernels, the parametric feature-space consisted of mean parameters only, since, in section 8.2, the inclusion of additional parameters was not found to yield gains. For all kernels, a maximally non-committal metric was applied, estimated globally over all target speakers. The form of derivative and parametric kernels used in this section were chosen both due to their performance in section 8.2 and to avoid conditions where the features-spaces will be identical. For consistency, all derivatives were normalised by the total number of frames in the utterance. Table 8.11 summarises the performance of SVM-classifiers using these kernels for a range of model sizes.

# Components	Derivative (∇)		Parametric (λ)	
	EER (%)	minDCF	EER (%)	minDCF
128	8.62	0.3775	8.32	0.3455
256	8.08	0.3457	8.18	0.3302
512	7.92	0.3375	8.17	0.3256
1024	7.60	0.3398	8.47	0.3488

Table 8.11: Summary of individual kernel performance.

8.4.1 Effect of kernel combination

Initially, pairwise kernel combination of a 1024-component derivative kernel and a 512-component parametric kernel was examined. From table 8.11 it can be seen that these were the best performing derivative and parametric kernels respectively. When equal weights were used the error associated with this classifier was 7.40% EER, representing a small 0.20% gain compared to the 1024-component derivative kernel[1]. Experiments were then performed to identify whether individually weighting each kernel could yield gains compared to equal weight combination. Initially, combination using a `minEER` criterion was evaluated. A line-search was performed and the kernel weights selected that gave the lowest EER. Although this approach is not practical for larger number of kernels, this criterion forms an upper bound on the gains obtainable by combining kernels. The minimum error rate obtained in this manner was 7.31% using kernel weights of 0.53 and 0.47 for the derivative and parametric kernels respectively. This suggests that the equal weight combination scheme is not optimal.

[1]The results presented here differ from those previously reported in [130] and [131]. Since publication of these papers a flaw was discovered in the evaluation methodology that caused the reported gains from kernel combination to be severely optimistic. Updated versions of these papers are available from mi.eng.cam.ac.uk/∼cl336

$\log \zeta$	Kernel Weights		EER (%)	minDCF
	∇_{1024}	λ_{512}		
$-\infty$	1.00	0.00	7.60	0.3397
-7	1.00	0.00	7.60	0.3397
-6	0.73	0.27	7.41	0.3187
-5	0.59	0.41	7.38	0.3105
-4	0.53	0.47	7.31	0.3068
-3	0.51	0.49	7.38	0.3060
-2	0.50	0.50	7.38	0.3047
∞	0.50	0.50	7.40	0.3053
`minEER`	0.53	0.47	7.31	0.3068

Table 8.12: Performance of `maxMargin` MKL combination as ζ varies compared to optimal `minEER` weighting.

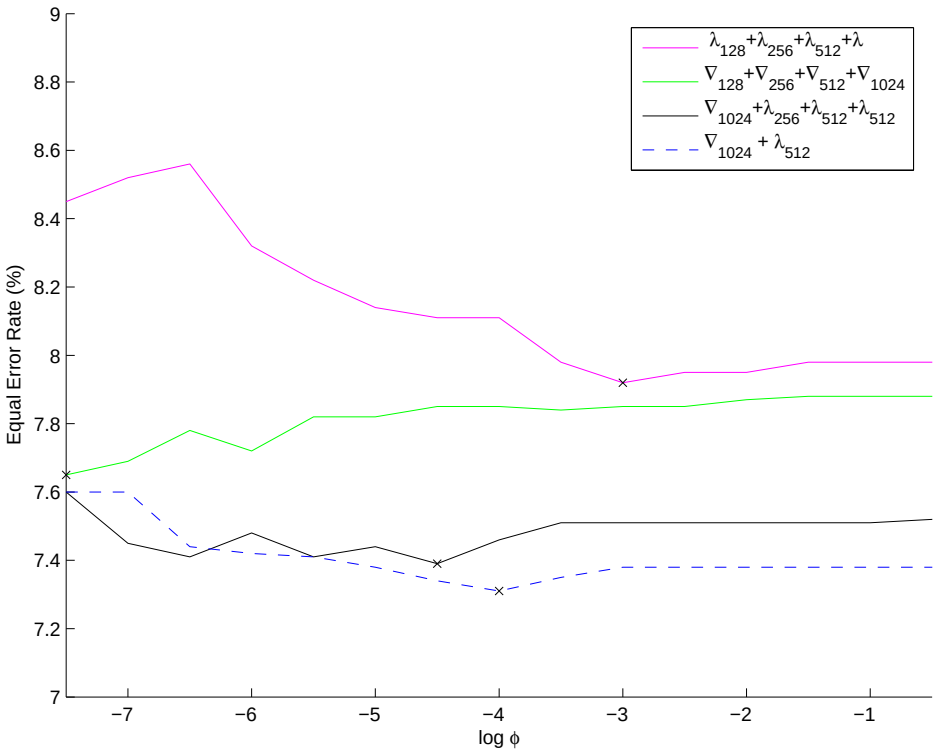


Figure 8.6: EER versus $\log(\zeta)$ for different systems combined using the `maxMargin` MKL scheme. Low values of $\log(\zeta)$ correspond to the unregularised MKL solution. High values correspond to equal weightings. For each system, the minimum EER that may be obtained by adjusting ζ is marked.

Next system combination was performed using the adapted `maxMargin` criterion for multiple kernel learning proposed in chapter 6 for SV. Here the kernel weights $\boldsymbol{\beta}$ were tied over all speakers to obtain more robust parameter estimates. In preliminary experiments where weights were trained independently for each speaker, a large variation in the weightings were observed for different speakers. However the effect on system performance was about the same. Table 8.12 shows the performances obtained using `maxMargin` with a range of values of the constant ζ associated with the sparsity regularisation term proposed in section 6.2.1.1. The optimal `minEER` is also shown for comparison. The case when $\log \zeta = \infty$ is equivalent to an equal-weighted combination. When $\zeta = 0$ ($\log \zeta = -\infty$) a sparse weighting is obtained that performs poorly compared to the baseline. This indicates that the standard level of sparsity associated with the originally proposed MKL scheme [168] is not appropriate for this task. By increasing ζ small gains are observed until the `minEER` solution is reached.

The `maxMargin` MKL scheme was then applied to other combinations of kernels where `minEER` is not always possible. Combinations of derivative and parametric kernels based on different generative model structures were examined, using GMMs ranging from 128 to 1024 components. As discussed in section 6.3.4, using features extracted from a range of model structures may improve performance. Combining kernels based on different generative models also allows the user to avoid explicitly choosing an appropriate model structure. Figure 8.6 shows how the EER varies with $\log \zeta$ for a selection of these systems. The minimum EER obtained in each case is also indicated. For most systems the optimal values of $\log \zeta$ did not correspond to either the equal weight or the sparse solutions indicating that the adapted `maxMargin` MKL scheme can lead to improved results over both equal weight combination and the non-regularised `maxMargin` scheme for more general combinations of kernels.

If a value for ζ is selected that minimises the EER, MKL is guaranteed to not perform worse than equal-weight combination. Unlike using the `minEER` criterion, this is feasible for large numbers of kernels as in all cases only a single parameter must be optimised. For the modified `maxMargin` MKL objective function defined by equation 6.22, the value of the objective function increases monotonically with ζ. This means that an appropriate regularisation factor cannot be automatically selected by maximising the objective function. In the remainder of this section, the results quoted for `maxMargin` MKL were obtained by adjusting ζ a-posteriori to reduce the EER. Selecting a value for ζ that optimises EER on the test set, rather than development data, can introduce bias. However as only a single parameter is tuned this bias is expected to be small. In preliminary experiments ζ was optimised independently over different subsets of speakers. The optimal ζ was found to be independent of the subset chosen and, as shown in figure 8.6, the minimal EER is typically achieved over a wide range of $\log \zeta$.

Table 8.13 shows the results of combining various forms of derivative and parametric kernels. For each combination the `maxMargin` scheme is compared against equal weight combination (`Equal`) and, for pairwise combinations, the `minEER` result. The performance of the best individual kernel is also indicated (`Single`). Although minDCF results are not stated, these were in line with the EER results. For parametric kernels, equal weight combination of 128 and 256 component models yielded small gains compared to the individual kernels. By comparison, the performance of a 512-component system was 8.17% indicating that these gains were not simply due to the increased complexity of the combined classifier. For combination of larger models, the gains were smaller and when 512 and 1024 component models were combined the performance of the equal-weight combination was worse than the 512-component parametric kernel alone. Out of the pairwise combinations examined, this

Table 8.13: Kernel-level classifier combination. Systems were combined using both equal kernel weights (`equal`) and kernel weights trained using MKL using either a minimum EER (`minEER`) or a maximum-margin (`maxMargin`) criterion. Performance of the best individual kernel (`single`) is also quoted for comparison.

System	EER (%)			
	`Single`	`Equal`	`minEER`	`maxMargin`
$\lambda_{128}+\lambda_{256}$	8.18	8.08	8.08	8.08
$\lambda_{256}+\lambda_{512}$	8.17	8.14	8.01	8.01
$\lambda_{128}+\lambda_{256}+\lambda_{512}$	8.17	8.09	-	8.07
$\lambda_{512}+\lambda_{1024}$	8.17	8.48	8.16	8.42
$\lambda_{256}+\lambda_{512}+\lambda_{1024}$	8.17	8.15	-	8.15
$\lambda_{128}+\lambda_{256}+\lambda_{512}+\lambda_{1024}$	8.17	7.98	-	7.95
$\nabla_{128}+\nabla_{256}$	8.08	8.26	8.08	8.08
$\nabla_{256}+\nabla_{512}$	7.92	7.92	7.89	7.89
$\nabla_{512}+\nabla_{1024}$	7.60	7.81	7.59	7.59
$\nabla_{256}+\nabla_{512}+\nabla_{1024}$	7.60	7.81	-	7.60
$\nabla_{128}+\nabla_{256}+\nabla_{512}+\nabla_{1024}$	7.60	7.88	-	7.60
$\lambda_{128}+\nabla_{128}$	8.32	8.18	8.11	8.11
$\lambda_{256}+\nabla_{256}$	8.08	7.61	7.57	7.57
$\lambda_{512}+\nabla_{512}$	7.92	7.64	7.59	7.59
$\lambda_{512}+\nabla_{1024}$	7.60	7.40	7.31	7.31
$\lambda_{256}+\lambda_{512}+\nabla_{1024}$	7.60	7.40	-	7.31
$\lambda_{128}+\lambda_{256}+\lambda_{512}+\nabla_{1024}$	7.60	7.71	-	7.38
$\lambda_{1024}+\nabla_{1024}$	7.60	7.45	7.38	7.38
$\lambda_{512}+\lambda_{1024}+\nabla_{1024}$	7.60	7.68	-	7.39
$\lambda_{256}+\lambda_{512}+\lambda_{1024}+\nabla_{1024}$	7.60	7.51	-	7.38

combination was unique in that no weighted combination was found to yield significant gains over the 512-component parametric kernel alone. This is reflected in the `minEER` result for this combination. For `maxMargin` MKL, in most cases gains were observed over the equal-weight combination, however these were usually small. By contrast, for derivative kernels all `maxMargin` combinations performed worse than the best individual kernel. The comparatively high `minEER` results associated with the pairwise combinations suggest that this was due to the features not being complementary, rather than a poor choice of kernel weights.

Finally combinations of derivative and parametric kernels were examined. Due to the poor performance obtained when combining multiple derivative kernels, only a single derivative kernel was included in each combination. A maximum of four kernels were combined in any one system. This was due to the memory requirements of the MKL scheme, which scale linearly with the number of kernels. For all combinations except $\lambda_{512} + \lambda_{1024} + \nabla_{1024}$ equal weight combination yielded gains over the best single kernel. When the `maxMargin` scheme was applied, further gains were achieved in all cases. The best overall performance was 7.31% EER (0.3068 minDCF) obtained from pairwise combination of the 512-component parametric kernel and 1024-component derivative kernel using the maxMargin scheme. This was a reduction in EER of 0.09% compared to equal weight combination and a reduction of 0.29% over the best performing single kernel. Both results were statistically significant at a level of 99%.

8.4.2 Comparison with score-fusion

Finally, the maximum-margin kernel combination scheme was compared with standard score-fusion approaches. Combined scores were initially obtained by either equally weighting scores (`Equal`) or by selecting the score a-posteriori that resulted in the lowest EER (`minEER`). Logistic regression, described in chapter 6, was also used. In the absence of a suitable development dataset two logistic regression schemes were applied. In `LR-trn`, weights were obtained using the scores obtained by classifying the training set. In `LR-tst`, logistic regression was applied directly to the test scores using the correct label information. Like the `minEER` criterion, this breaks experimental protocol, however it does provide an upper bound on the gains that can be achieved using logistic regression. An alternative `SVM` fusion scheme was also applied. Here an SVM post-classifier was trained using training scores as input features.

Results for these experiments are given in table 8.14. Overall, the performance obtained using score-fusion was similar to that obtained using kernel-combination. When the scores from parametric systems were combined, almost all combination schemes showed gains. Only pairwise combination using 512 and 1024 component showed a loss, similar to kernel combination. Similarly, the combination of scores from derivative systems did not yield gains. When the scores of parametric and derivative systems were combined almost all equal-weight combination schemes yielded gains, aside from $\lambda_{512} + \lambda_{1024} + \nabla_{1024}$ and $\lambda_{128} + \lambda_{256} + \lambda_{256} + \nabla_{1024}$. The relative performance of `SVM` and the logistic regression schemes varied depending on the type of system combined. Neither scheme gave consistently better performance. Despite the potential for overtraining, the `LR-trn` scheme generally gave similar performance to the 'optimal' `LR-tst` scheme and in several cases performed better. The best overall performance achieved using score-fusion was 7.38% (0.3126 minDCF) obtained when combining the 1024-component parametric and derivative kernels using the `SVM` scheme.

Table 8.14: Score-level classifier combination. Individual classifier scores were combined using either equal weights (`Equal`), weights trained using an SVM classifier (`SVM`) or weights trained using logistic-regression optimised on either the training set (`LR-trn`) or test set (`LR`). Performance of optimal minimum EER pairwise weighting (`minEER`) and the best individual classifier (`Single`) is provided for comparison.

System	EER (%)					
	`Single`	`Equal`	`minEER`	`SVM`	`LR-trn`	`LR-tst`
$\lambda_{128}+\lambda_{256}$	8.18	8.08	8.08	8.08	8.11	8.12
$\lambda_{256}+\lambda_{512}$	8.17	8.15	8.01	8.11	8.12	8.15
$\lambda_{128}+\lambda_{256}+\lambda_{512}$	8.17	8.10	-	8.08	8.12	8.15
$\lambda_{512}+\lambda_{1024}$	8.17	8.52	8.15	8.52	8.52	8.25
$\lambda_{256}+\lambda_{512}+\lambda_{1024}$	8.17	8.05	-	8.06	8.08	8.08
$\lambda_{128}+\lambda_{256}+\lambda_{512}+\lambda_{1024}$	8.17	8.05	-	8.01	8.01	8.14
$\nabla_{128}+\nabla_{256}$	8.08	8.25	8.08	8.25	8.25	8.17
$\nabla_{256}+\nabla_{512}$	7.92	7.93	7.88	7.91	7.91	7.86
$\nabla_{512}+\nabla_{1024}$	7.60	7.80	7.60	7.78	7.79	7.80
$\nabla_{256}+\nabla_{512}+\nabla_{1024}$	7.60	7.85	-	7.85	7.81	7.79
$\nabla_{128}+\nabla_{256}+\nabla_{512}+\nabla_{1024}$	7.60	7.92	-	7.93	7.95	7.75
$\lambda_{128}+\nabla_{128}$	8.32	8.21	8.14	8.24	8.23	8.15
$\lambda_{256}+\nabla_{256}$	8.08	7.73	7.68	7.73	7.72	7.73
$\lambda_{512}+\nabla_{512}$	7.92	7.65	7.65	7.75	7.67	7.70
$\lambda_{512}+\nabla_{1024}$	7.60	7.51	7.38	7.42	7.52	7.59
$\lambda_{256}+\lambda_{512}+\nabla_{1024}$	7.60	7.61	-	7.51	7.60	7.44
$\lambda_{128}+\lambda_{256}+\lambda_{512}+\nabla_{1024}$	7.60	7.81	-	7.76	7.84	7.41
$\lambda_{1024}+\nabla_{1024}$	7.60	7.44	7.35	7.38	7.44	7.65
$\lambda_{512}+\lambda_{1024}+\nabla_{1024}$	7.60	7.66	-	7.68	7.74	7.58
$\lambda_{256}+\lambda_{512}+\lambda_{1024}+\nabla_{1024}$	7.60	7.58	-	7.58	7.58	7.44
$\lambda_{512}+\lambda_{1024}+\nabla_{512}+\nabla_{1024}$	7.60	7.54	-	7.55	7.55	7.58

Combination	Criterion	EER (%)			
		$\nabla_{1024}+\lambda_{512}$	$\nabla_{1024}+\lambda_{1024}$	$\nabla_{1024}+\lambda_{256}+$ $+\lambda_{512}$	$\nabla_{1024}+\lambda_{128}$ $+\lambda_{256}+\lambda_{512}$
Score	`Equal`	7.51	7.44	7.61	7.81
	`minEER`	7.38	7.35	-	-
	`SVM`	7.41	7.38	7.51	7.76
	`LR-trn`	7.52	7.44	7.60	7.84
	`LR-tst`	7.59	7.65	7.44	7.41
Kernel	`Equal`	7.40	7.45	7.40	7.71
	`minEER`	7.31	7.38	-	-
	`maxMargin`	7.31	7.38	7.31	7.38

Table 8.15: Comparison of kernel-level and score-level approaches for classifier combination for best performing combinations.

In table 8.15, kernel-combination and score-fusion schemes are compared over four different sets of systems. These are the two combinations that yielded the best performance for kernel-combination and the two that performed best under score-fusion. For three of the four systems compared, equal weight kernel combination yielded at least a 0.10% improvement over equal-weight score-fusion indicating that this combination strategy is able to generalise more effectively. This may be related to the more relaxed margin constraint associated with kernel combination. For the fourth system the performance of equal weight combination was roughly equal under both combination schemes. Unlike `LR-tst` and `minEER` score-fusion, where a weight for each classifier was trained using the test data, for `maxMargin` MKL only ζ, a single parameter, was optimised on the test set. Despite this, the best performance was obtained using the `maxMargin` MKL scheme for all system combinations. Best overall performance was obtained using the `maxMargin` MKL scheme to combine a 1024 component derivative kernel with a 512-component parametric kernel. From figure 8.7 it can be seen that this system outperformed the best score-fusion system over most of the operating range.

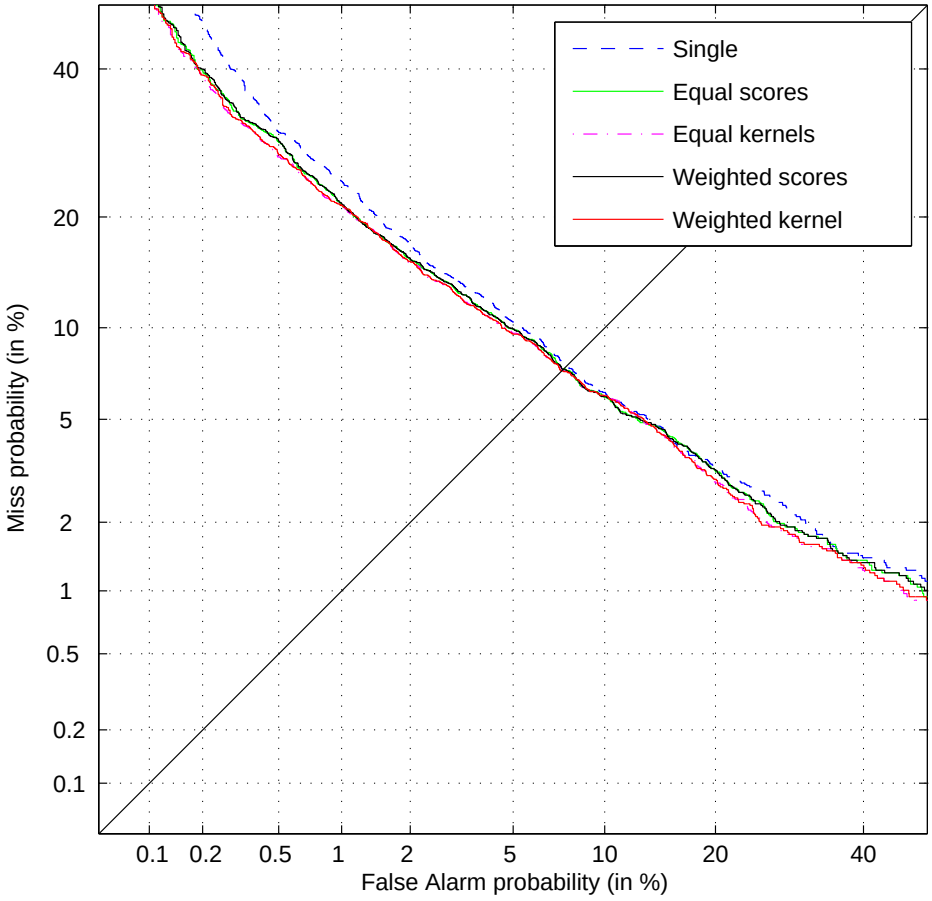


Figure 8.7: Performance of (a) best individual system (b) best equal weighted score-fusion (b) best equal weighted kernel combination (c) best overall score-fusion (d) best overall kernel combination.

8.5 Results for static and dynamic kernel combination

In the previous section, an MKL scheme was applied to improve the performance of an SV system by combining multiple dynamic kernels. An alternative approach is to combine dynamic kernels with static kernels, which operate on fixed-dimensional data. In chapter 7 two general approaches were identified for combining static and dynamic kernels. These are feature-level combination and observation-level combination. Both of these approaches are evaluated in the following subsections. For both feature and observation-level combination, static and dynamic kernel combination can yield extremely large feature spaces. Thus, to avoid overtraining it is necessary to use generative models containing comparatively few Gaussian components. Alternatively, the data partitioning or speaker-tying schemes proposed in chapter 7 may be applied. In the following section these approaches are evaluated for a feature-level combination scheme.

8.5.1 Feature-level static kernels

Feature-level combination of static and dynamic kernels was introduced in chapter 7. Here, each utterance is initially mapped into the feature space associated with a dynamic kernel. Then, instead of taking the inner product, a static kernel function is calculated. Thus, this approach is only suitable for dynamic kernels that have an explicit associated feature space.

Feature-level combination was evaluated using derivative and parametric dynamic kernels of various sizes. Each kernel was trained as described in section 8.2. However, unlike the previous experiments, here derivatives were normalised by the component occupancy instead of the total number of frames. This was because the results in section 8.2 indicated that component occupancy normalisation was more effective for the smaller generative models used in these experiments. This resulted in a gain of 0.20% EER for 16-component models and 0.27% EER for 128-component models. The dynamic kernels were then combined with a range of static kernels, including linear, inhomogeneous polynomial kernels of various orders and a Gaussian kernel. The functional form for each of these kernels is given in section 2.2.6. For linear and polynomial kernels, the dimension of the resulting feature space is determined by the kernel parameters. For the linear kernel, this was equal to $31M$, where M is the number of model components. For the polynomial kernels, the number of features was equal to $(31M + p)!/(31M)!p!$, where p is the order of the kernel.

In these experiments, a dynamic feature-level, maximally non-committal metric was applied using a projection as described in section 7.2. Thus for each dynamic kernel, the metric had the same form irrespective of the static kernel used. To allow a consistent parameterisation to be used for each static kernel, the dynamic range of each feature vector was initially normalised before applying a static kernel. This was implemented by dividing each feature by the square root of the feature vector dimension, since when a maximally non-committal metric is used, the expected value of the kernel is equal to the dimension of the feature vector. This approach was found to perform slightly better than spherical normalisation. For the Gaussian kernel, the variance was then fixed at $\sigma^2 = 1$. In preliminary experiments, further adjustments of σ^2 were not found to improve performance. Note that normalising the dynamic range does not affect the performance when linear static kernels are used and C is

kept fixed at the SVM^{light} default. In this case, the decision boundary is invariant to fixed scalings of the kernel function.

Dynamic kernel	Static kernel	16		128	
		Features	EER (%)	Features	EER (%)
Derivative (∇)	Linear	496	12.14	3968	8.35
	Polynomial (order 2)	123,753	11.94	7,878,465	8.28
	Polynomial (order 3)	20,584,249	12.07	31,285,384,515	8.68
	Gaussian	-	15.73	-	14.97
Parametric (λ)	Linear	496	12.54	3968	8.32
	Polynomial (order 2)	123,753	12.74	7,878,465	8.52
	Polynomial (order 3)	20,584,249	12.61	31,285,384,515	8.68
	Gaussian	-	16.43	-	13.85

Table 8.16: Feature level combination of static kernels with parametric and derivative dynamic kernels.

The initial results obtained for feature-level combination of static and dynamic kernels are shown in table 8.16. Here 16 and 128 component GMMs were used. For parametric kernels, applying complex static kernels did not yield gains over the linear case. However, for derivative kernels small gains were obtained from using second order polynomial kernels. The reason that the use of more complex static kernels did not yield gains for all systems was primarily due to the extremely large feature spaces generated. Even the simplest combination considered, 16-component models combined with a second order polynomial kernel, yields a feature space that is larger than that obtained when using a linear kernel with 3,000 component models. For the non-linear kernels almost all training examples lay within or near the margin, indicating that, due to the large feature spaces, the classifiers were overfitting the training data.

Several strategies for handling conditions of limited data were discussed in chapter 7. Since each training utterance in the 2002 NIST SRE typically contains in the order of ten thousand frames, one approach is to partition each utterance into multiple sections, and treat each as a distinct training example. This partitioning was applied to both negative and positive enrollment utterances. The duration of the training utterances is fairly consistent in the 2002 NIST SRE dataset therefore all partitions contained approximately the same number of frames.

The results of applying data partitioning to 16 and 128 component derivative kernels are shown in table 8.17. These were combined with both linear and second order polynomial kernels. Each utterance was split into up to 6 sections, resulting in a minimum duration of approximately 20 seconds per utterance. For both linear and polynomial kernels using 16 components, gains were observed when the enrollment utterances were split. This indicated that the typical approach of extracting a single SVM training example per utterance is not optimal for all experimental configurations. However, for 128-component models the performance of both kernels degraded as the number of training utterances increased. This is expected since, due to the shorter duration of each utterance, there are fewer observations per component and the parameter estimates for the feature vectors will be less robust.

In chapter 7, speaker-tying was proposed as an alternative approach to handle the lack of enrollment data per speaker. This was implemented by training a single SVM classifier

#Partitions	Duration (s)	EER(%)			
		16		128	
		Linear	Polynomial	Linear	Polynomial
1	120	12.14	11.94	8.35	8.28
2	60	11.80	11.44	8.38	8.43
3	40	11.29	11.24	8.35	8.45
4	30	11.28	11.14	8.45	8.38
5	24	11.12	11.00	8.58	8.46
6	20	11.10	10.79	8.62	8.58

Table 8.17: Feature-level combination using 16 and 128 component derivative kernels and varying numbers of training utterance partitions. Multiple training utterances were created by partitioning 120s training utterances into equal duration sequences.

per gender. A speaker-dependent derivative kernel was then applied to map each utterance into a speaker-independent feature space. As in the previous sections, the features of the kernel were derivatives with respect to the mean parameters of a speaker-dependent GMM. This was adapted from a gender-dependent UBM using two iterations of static-prior MAP. Kernels based on 16 and 128 component models was evaluated. Tied systems (`Tied`) were trained using both linear and second order inhomogeneous polynomial kernels. Additional systems (`Tied + normalised`) were also trained for both forms of static kernel. For these systems an explicit feature space normalisation was applied, as described in chapter 7.

System	EER(%)			
	16		128	
	Linear	Polynomial	Linear	Polynomial
`Untied`	12.14	11.94	8.35	8.28
`Tied`	35.14	23.41	30.74	22.93
`Tied + normalised`	18.09	17.64	11.73	11.58

Table 8.18: Cross-speaker SVM parameter tying. 16 and 128 component derivative kernels were combined with linear and polynomial static kernels using (a) speaker-dependent SVM parameters (`Untied`) (b) gender-dependent SVM parameters (`Tied`) or (c) gender-dependent SVM parameters with additional speaker-dependent feature normalisation (`Tied + normalised`).

The performance of each of these systems is shown in table 8.18. For comparison, the performance obtained using speaker-dependent SVM classifiers (`Untied`) is also shown. For both model sizes evaluated, the performance of the tied systems was extremely poor. This was also the case when polynomial static kernels were applied, suggesting that this performance degradation was not simply due to the inability of the linear classifier to model a speaker-independent decision boundary, as discussed in section 7.4.2. In the tied system, the majority of the dynamic features associated with positive training data were not centered around the origin. Thus, the speaker-dependent models were not sufficiently representative of the true speech distributions to provide an effective normalisation. When an explicit normalisation

was performed all tied systems yielded gains. However, in no instances did the resultant system outperform the untied systems.

The relatively poor performance of classifier-tying is in contrast to results reported in [66, 67] for ASR. There, the parameters of the SVM classifier were tied over a range of noise conditions. Noise condition-dependent dynamic kernels were then used for each utterance, adapted using either VTS compensation or single pass retraining. The difference in the performance of classifier-tying for these two tasks is likely to be due to several factors. First, in the SV case the classifier was tied over a much larger number of classes. Thus it may not be possible to find a single decision boundary that is appropriate for all target speakers. Second, the different adaptation scheme used for the ASR task may yield more accurate class-dependent models. In this case the dynamic kernels will provide a more effective normalisation of the feature space. Finally, in the ASR task classifiers were trained to distinguish between confusable pairs of words. For this task the positive and negative examples will typically occupy roughly equal areas of the feature-space. In contrast, for SV the positive examples associated with each target speaker will typically occupy a much smaller region of the feature space than the negative examples, which contain data from many speakers. Thus, the tied-classifier for SV will be more sensitive to conditions where the positive data associated with each target-speaker/noise-condition is misaligned.

8.5.2 Observation-level static kernels

In chapter 7 an alternative method of combining static and dynamic kernels was described. Here static kernels are applied at the observation level. In general, it is not possible to train a generative model in the feature space associated with a static kernel. However, as described in chapter 7, such a model can be approximated. Two forms of combined dynamic kernel were then proposed, the generalised derivative and generalised parametric kernels. When linear static kernels are used, these respectively generalise the standard derivative and parametric kernels evaluated in section 8.2. Also, when a single component generative model is used, both kernels can be shown to generalise the GLDS kernel [24], described in section 3.2.1.

Initially SVM classifiers were evaluated using the GLDS kernel. For this experiment, spectral-based normalisation techniques were not used since they typically normalise each utterance mean to zero. A maximally non-committal metric was also not used since over long utterances the mean of the ΔPLP coefficients tends to zero. Using a linear static kernel, the Equal Error Rate (EER) of the GLDS kernel was 23.74%. This poor performance is expected since, as shown in chapter 7, the linear GLDS kernel has the form of a parametric kernel defined by a single-component GMM. When a second order polynomial kernel was used the performance was 22.40%. This small gain is partly due to the fact that the SVM is able to use information from the delta coefficients.

Next, the generalised derivative kernel (GDK) was evaluated. For certain forms of static kernel, such as linear or low order polynomial kernels, it is possible to evaluate the GDK kernel by explicitly training a generative model in the static feature space. As in section 8.2, cepstral feature warping was initially applied to all utterances to reduce the effects of noise. Each speech observation was then explicitly mapped into the corresponding feature space. For each form of static kernel, two gender-dependent UBMs were then trained in the static feature space using all enrollment utterances of the appropriate gender. Finally speaker-dependent GMMs, defined in the static feature space, were obtained by adapting the mean parameters of the appropriate gender-dependent UBM. A generalised derivative kernel (`Explicit`) was

then applied by taking derivatives with respect to the means of the speaker-dependent models. As in the previous section, due to the use of smaller generative models the derivatives were normalised using the component occupancies rather than the total number of frames. A diagonal approximation to a maximally non-committal metric was also used. Unlike feature-level combination, where the metric was defined in the dynamic feature space only, here the metric was applied in the feature space associated with the combined static and dynamic features.

System	EER(%)			
	Linear		Polynomial	
	16	128	16	128
`GMM-LLR`	17.40	12.04	16.40	14.52
`Explicit`	12.14	8.38	10.32	10.94
`Viterbi`	12.30	9.15	10.38	11.11
`Approx`	12.21	10.20	9.25	10.29

Table 8.19: Performance of GDK systems based on explicit models (`Explicit`), explicit models using Viterbi alignments (`Viterbi`) and using approximated models (`Approx`).

Table 8.19 shows the performance for linear and second order inhomogeneous polynomial kernels, using an GDK-based SVM classifier for 16 and 128-component models. Baseline results, using a GMM-LLR classifier with the same models, are also presented. In all cases, the SVM results outperformed the GMM-LLR classifier. In the linear case, GMM-LLR and SVM performance improved for larger model sizes. By contrast, for the polynomial SVM classifier best performance was obtained using 16-component models. This difference is due to the significantly larger feature space associated with the polynomial kernel, 527 versus 31 features per component. Although the best polynomial system did not outperform the 128-component linear system, further small gains were achieved when these two systems were combined. Applying the maximum-margin based kernel combination scheme evaluated in section 8.4 gave a performance of 8.08%.

To examine the effect on performance of using Viterbi component alignments, GDK systems (`Viterbi`) were trained as defined in equation 7.10. Using hard alignments degraded the performance of all systems. The effect was less severe for the polynomial kernels, since for GMMs trained in a high-dimensional space, the component posteriors already tend towards hard alignments. Lastly, GDK systems (`Approx`) using approximated models were trained for these two forms of kernel. Here the component posteriors were obtained using the linear models and the static kernel function between observation was approximated as described in section 7.3.1.2. For the linear kernels, performance was worse for this system, due to the approximation used for the metric. However, when polynomial static kernels were used the approximated models actually outperformed the `Explicit` system. This gain suggests that the component-posteriors associated with the linear models were more robust than those of the explicitly trained polynomial models.

Finally, the generalised derivative kernel was evaluated using other forms of static kernel. A third order inhomogeneous polynomial kernel and a Gaussian kernel were used. For the Gaussian kernel σ^2 was set to 31, the dimension of the observations. For 128-components, no gains were observed using non-linear kernels due to the limited amount of available training

Static kernel	EER(%)	
	16	128
Linear	12.21	10.20
Polynomial (order 2)	9.25	10.29
Polynomial (order 3)	9.96	13.17
Gaussian	14.29	12.94

Table 8.20: Combination of derivative kernel with various static kernels. Viterbi statistics were used and component posteriors were obtained from GMMs trained using the original observations.

data. When 16-component models were used both polynomial kernels gave gains over the linear case and also outperformed the 128-component linear system. For the model sizes evaluated, the best performance was obtained using a second order polynomial kernel with a 16-component model. However, as shown in section 8.2, the use of linear static kernels with larger generative models yields better overall performance.

The generalised parametric kernel (GPK), proposed in chapter 7, was also evaluated. For these experiments, approximated models were used with a range of static kernels. The training scheme used for the GPK was based on the training scheme used for the standard parametric kernels evaluated in section 8.2. Component-occupancies were obtained using linear utterance-dependent models, adapted using a single iteration of MAP adaptation. The utterance-dependent models associated with the GPK were then approximated as described in section 7.3.2 resulting in two MAP iterations in total, one in the linear space, and one in the space associated with a static kernel. The adaptation constant was fixed at $\tau^{\texttt{map}} = 1$ for all systems. Preliminary experiments using a range of values of $\tau^{\texttt{map}}$ and alternative MAP adaptation schemes did not yield gains over the results described here.

Static kernel	EER(%)	
	16	128
Linear	13.31	9.29
Polynomial (order 2)	11.10	9.18
Polynomial (order 3)	11.12	9.64
Gaussian	14.16	11.24

Table 8.21: Combination of parametric kernel with various static kernels. Viterbi statistics were used and component posteriors were obtained from linear GMM-Models

Results obtained using the generalised parametric kernel are given in table 8.21. As with the generalised derivative kernel, the use of Viterbi statistics and an approximated metric was found to degrade the performance of the linear systems, relative to the standard parametric kernels evaluated in section 8.2. For the 128-component system this resulted in an absolute performance degradation of 0.97% EER. However, for both 16 and 128 component models, small gains were then obtained when second order polynomial kernels were applied. The performance of the standard parametric kernel was 12.54% for 16-component models, compared with 11.10% for the GPK using a second order polynomial kernel. However for

128-component models, the GPK did not yield gains over the standard parametric kernel for any of the static kernels evaluated. For third order polynomial kernels, gains were only obtained with 16-components. When larger models were used the classifiers failed to generalise due to the high feature space dimension. Gaussian static kernels were also evaluated with $\sigma^2 = 31$, however at both model sizes, these kernels failed to yield gains over the linear case.

8.6 Summary

In this chapter, three kernel-based schemes, proposed in this thesis to improve SV systems, were evaluated using the 2002 NIST SRE task. Initially, variational dynamic kernels, proposed in chapter 5, were evaluated. Unlike many standard forms of dynamic kernel, which are derived from a matched-pair bound to the KL divergence, these kernels are derived from more accurate variational approximations to the divergence. However, the two variational kernels evaluated were found to only yield small gains over the related non-linear GMM-supervector kernel and performed worse than the GMM-supervector kernel, both derived from the matched-pair bound. Unlike standard derivative or parametric kernels, variational kernels do not require that model component indices are coordinated. This allows the use of a more flexible modelling framework. For example, a range of model structures or background models may be used. Results were then obtained using a more complex framework, where the imposter training set contained examples that were adapted from two independently trained background models. In this setup, the variational kernels yielded further small gains relative to the non-linear GMM-supervector kernel.

One strategy to improve the performance of an SV system is to combine multiple SVM classifiers. This combination may be applied either at the score level or at the level of the dynamic kernels. In section 8.4 a maximum-margin based kernel combination scheme, adapted for SV in chapter 6, was evaluated on combinations of parametric and derivative kernels. For this scheme, the use of regularisation to control sparsity was found to be important. When the regularisation term was fixed at zero, a sparse weighting was obtained that performed poorly compared to equal-weight combination. Combinations of parametric kernels based on different model structures was found to yield gains, but this was not the case when combining derivative kernels. Classifier combination using score-fusion was also evaluated using either a logistic-regression scheme or an SVM post-classifier. In general, the systems that yielded gains when combined were similar for kernel combination and score-fusion. The best overall performance achieved was 7.31% EER when 1024-component derivative and 512-component parametric kernels were combined. This represented a small absolute gain of 0.09% compared to equal weight combination. In comparison, the best score-fusion result was 7.38%, obtained by combining 1024-component parametric and derivative kernels using an SVM post-classifier.

The combination of dynamic kernels with traditional static kernels may also yield gains. Two combination strategies, feature-level and observation-level combination, were described in chapter 7. Both of these approaches were evaluated in section 8.5. For feature-level combination, gains were obtained when derivative kernels were combined with second order polynomial kernels. However, the use of more complex static kernels or combination of static and parametric kernels did not yield gains. This was due to a combination of the large feature spaces obtained, and the limited amount of enrollment utterances available per target-speaker. When a data partitioning scheme was applied, further gains were observed from using non-linear static kernels with small generative models. Cross-speaker tying of SVM classifiers was

also evaluated, however this was found to significantly degrade overall performance. This was due to the sensitivity of the tied system to the accuracy of the target-speaker models. Finally observation-level combination was evaluated. For both derivative and parametric dynamic kernels, the use of Viterbi statistics and an approximated metric degraded performance relative to standard derivative and parametric systems. However, for both kernels, combination with simple non-linear static kernels was found to yield small gains. Neither generalised kernel outperformed the best results obtained using standard derivative and parametric kernels with larger models.

Chapter 9
Conclusions

This thesis has investigated statistical approaches for classifying utterances of speech, with particular emphasis placed on applying kernel-based discriminative classifiers to the task of speaker verification. This thesis contains three main contributions, described in chapters 5, 6 and 7 and summarised in the following sections. The first contribution of this thesis, summarised in section 9.1, was to propose alternative forms of dynamic kernel, derived from variational approximations to the KL divergence. Unlike many standard forms of dynamic kernel, these can be combined with generative models that vary in model structure, and that are adapted from a range of background models. The second contribution of this thesis, summarised in section 9.2, was to adapt a maximum-margin based kernel combination scheme for the SV task, and to propose a general framework in which many forms of dynamic kernel are placed into one of two general categories, parametric and derivative kernels. This framework can be used to motivate new forms of dynamic kernel as well as establish the conditions under which dynamic kernels will be complementary when combined. The final contribution of this thesis, summarised in section 9.3, was to propose a scheme for combining dynamic kernels with static kernels, defined between pairs of observations, to potentially yield more discriminative features. Based on this scheme, higher order versions of standard derivative and parametric kernels were proposed. Finally, in section 9.4 several directions for future work are discussed.

9.1 Variational dynamic kernels

The first contribution of this thesis was to derive alternative forms of dynamic kernel, based on variational approximations to the KL divergence. Many dynamic kernels used for SV

are examples of distributional kernels. For these kernels, described in chapter 3, a distinct GMM is trained to represent each utterance in the dataset. A dynamic kernel function between two utterances is then defined based on a divergence measure between the associated distributions. Two commonly used types of distributional kernel are the linear and non-linear GMM-supervector kernels, both derived from the Kullback-Leibler divergence between two GMMs. Since it is not possible to calculate this in closed form, a standard approach is to use a matched-pair bound on the divergence instead. However, the use of this bound requires that all GMMs must contain the same number of components, and components with the same index must be coordinated. In practice this requires that all distributions are adapted from a single background model. These restrictions may limit the performance of an SV system.

In chapter 5, two recently proposed alternative KL approximations were described. These are the variational approximation and the variational upper bound, both derived by introducing additional variational distributions between Gaussian components. The two variational approximations are generally more accurate than the matched-pair bound, however they both yield the matched-pair solution in the worst case. Unlike the matched-pair bound, these two variational approximations do not place restrictions of the forms of GMM that can be used. Two variational dynamic kernels were then proposed based on these approximations, derived in a similar manner to the non-linear GMM-supervector kernel. However, unlike the linear and non-linear GMM-supervector kernels these variational kernels do not restrict all GMMs to have the same structure. This allows more complex training schemes. For example, GMMs may be adapted from a range of gender or noise condition-dependent background models. Additionally, the use of a kernel that more accurately reflects the true KL divergence between GMMs may lead to gains.

In chapter 8, the variational kernels were evaluated on the 2002 NIST SRE task. Initially, utterance-dependent generative models were obtained by MAP adapting an appropriate gender-dependent background model using the provided gender information. When the various approximations to the KL divergence were compared, both variational approximations were found to be more accurate than the matched-pair bound. For cross-gender comparisons, where there was no strong coordination between components, the matched-pair bound typically exceeded the variational approximations by an order of magnitude. However when the kernels were compared, the performance of the variational dynamic kernels was found to yield only small gains over the non-linear GMM-supervector kernel and perform worse than the standard GMM-supervector kernel. This was due to the lack of cross-gender kernel evaluations in the experimental setup. Including cross-gender SVM imposters gave a small improvement relative to the non-linear GMM-supervector kernel. This was not observed in a gender-independent system indicating that this was not simply due to the larger amount of training data used. Best performance was 9.59% EER using a kernel based on the variational approximation and 9.69% for the variational upper bound, compared to 8.35% for the GMM-supervector kernel.

9.2 Dynamic kernel combination

The second contribution of this thesis was to investigate the combination of multiple classifiers to improve the performance of a speaker verification system. In many recent systems, classifier combination is performed by fusing the output scores. This may be achieved using either an

SVM post-classifier or a scheme such as logistic regression to train a suitable weighting for each score. For SVM-based systems, an alternative approach is to combine at the kernel level. This requires choosing a suitable weighting for each kernel. One approach is to select the weighting that minimises the cross-validation error by performing a grid search over all possible weightings. Unfortunately this is only feasible when the number of kernels is small and is generally unsuitable for anything other than pairwise combination.

A recently proposed alternative scheme was described in chapter 6. Here the kernel weights are trained in conjunction with the SVM parameters to optimise a maximum-margin criterion. This scheme can be efficiently implemented using standard SVM implementations. A number of modifications were proposed to allow this scheme to be applied for SV. The standard scheme has a known tendency towards sparse weightings, which may not be optimal for speaker verification. A regularisation term was introduced allowing the user to tune the sparsity by adjusting a single parameter. Unlike grid-search based schemes, this parameter may be efficiently optimised using development data even when a large number of kernels are combined. The kernel weights were also tied over all target-speaker classifiers to increase the robustness of the parameter estimates.

Kernel combination will only yield gains when the features associated with the kernels are complementary. In this thesis it was shown that many existing dynamic kernels can be placed into one of these two classes, *parametric kernels*, where the feature space consists of parameters from the utterance-dependent model, and *derivative kernels*, where the derivatives of the utterance log-likelihood with respect to parameters of a generative model are used. The two sets of features produced have different properties and may be complementary. However, under certain conditions, described in chapter 6, the feature spaces produced may be shown to be identical. By avoiding these conditions a complementary set of kernels may be obtained.

Various combinations of derivative and parametric kernels were evaluated using the NIST evaluation data. Initially, the best individually performing derivative and parametric kernels were combined. When the regularisation term was fixed at zero, a sparse weighting was obtained that performed poorly compared to the optimal weight obtained through grid search. By adjusting the regularisation term, the optimal weighting was obtained using the maximum-margin scheme. When combining multiple parametric kernels based on different model structures further gains were observed. However, this was not the case for derivative kernels. The best overall performance was 7.31% EER achieved by combining 1024-component derivative and 512-component parametric kernels using the maximum-margin scheme. In comparison, the best score-fusion result was 7.38%, obtained by combining the same kernels using an SVM post-classifier.

9.3 Static and dynamic kernel combination

The final contribution of this thesis was to investigate the combination of dynamic kernels with traditional static kernels. This will potentially yield higher order features that are more useful for discriminating between speakers. Two general combination schemes were considered. In the first approach, feature-level combination, fixed dimensional feature vectors are obtained from each utterance as usual. Then instead of taking the inner product, a static kernel is evaluated. For the second approach, observation-level combination, instead of calculating

dynamic features in the original observation space, these features are calculated in the feature space associated with a static kernel defined between pairs of observations.

In chapter 7, two new forms of dynamic kernel were proposed, based on combining an observation-level static kernel with the derivative and parametric dynamic kernels defined in chapter 6. In general, it is not possible to explicitly train a generative model in the feature space associated with a static kernel. However, such a model can be approximated. In this thesis, a kernel metric was selected that normalises the variance term from the features, avoiding explicit calculation of these parameters. The component posteriors in the feature space were then approximated by posteriors derived in the original observation space.

Combining static and dynamic kernels can yield extremely high-dimensional feature spaces. For speaker verification, typically only limited numbers of enrollment utterances are available. Hence, the classifier may fail to generalise. Two approaches were proposed for handling these conditions. In the first approach, each enrollment utterance is partitioned into multiple sections. These are then used as distinct examples during SVM training. By varying the number of training examples extracted from each utterance, a trade-off can be made between robustly estimating the dynamic features and obtaining a robust decision boundary. An alternative scheme, proposed here, is cross-speaker parameter tying. Instead of training a distinct classifier for each target speaker, a single speaker-independent classifier is trained. A speaker-dependent kernel function is then applied, determined by the target identity. In chapter 7 it was shown that this approach requires the use of either a non-linear SVM classifier or a suitable speaker-dependent normalisation scheme.

In chapter 8, static and dynamic kernel combination schemes were evaluated. Initially feature-level combination was examined. For derivative kernels, combination with second order polynomial kernels yielded gains. However this was not the case for higher order static kernels or when parametric kernels were combined. This was due to the large feature spaces obtained and the limited amount of enrollment data per speaker. When the data partitioning scheme was applied, gains were obtained using both linear and polynomial static kernels when combined with a derivative kernel using small generative models. However, for larger models there were not enough observations per utterance to robustly estimate the feature vectors and the scheme was found to degrade performance. Cross-speaker tying of SVM classifiers was also evaluated using derivative and second order polynomial kernels. However, this was also found to degrade performance due to the sensitivity of the tied system to the accuracy of the speaker models. Finally, observation-level combination was evaluated. For both derivative and parametric dynamic kernels, combination with non-linear static kernels was found to yield gains, but typically only for small generative models and low order polynomial kernels. Better overall performance was obtained using larger generative models in combination with linear static kernels.

9.4 Future work

A number of the refinements proposed in this thesis may benefit from further investigation, either in terms of modifications to the approaches described, evaluation within a state-of-the-art system or by application to different tasks. Some suggestions for future work are given below:

- As discussed in chapter 5, variational dynamic kernels may be applied with more complex training schemes. For example, GMM structure may be allowed to vary, or clusters of speakers could be adapted from a wider range of background models. These schemes are likely to yield improvements on datasets where there is large variation in utterance duration, or where utterances are strongly clustered, for example by gender, accent, or environmental conditions.

- In this thesis, only variational dynamic kernels derived through exponentiation were evaluated. Similar kernels derived using the polarisation identity, such as the GMM-supervector kernel, have been found to perform more effectively, perhaps due to the lack of non-linearities in the distance induced between utterances [48]. Future work might examine the derivation of variational kernels from the polarisation identity, and compare the resulting features with the kernels used in this work. Alternatively, applying KL divergence based model normalisation, as used in [45], may improve performance of the variational dynamic kernels. For each utterance, the KL divergence between the associated generative model and the UBM is normalised, where the divergence is approximated using a matched-pair bound. Development of a related scheme based on a variational approximation to the divergence may yield further gains.

- The maximum-margin MKL scheme, described in chapter 6, was used to effectively combine multiple dynamic kernels. This thesis primarily examined combination of mean-based parametric and derivative kernels. Combination of more general forms of kernel, such as MLLR or CAT kernels, or kernels based on different speech parameterisations such as SNERF N-gram, or term-frequency LLR, kernels, may improve SV performance. The implementation of the scheme used in this thesis required a parameter to be set to control the level of sparsity. Although a suitable value for this parameter can be efficiently selected using a development dataset, further research on factors that affect the sparsity of the kernel weights may allow a suitable value to be selected without the need for development data.

- In chapter 6, the categorisation of dynamic kernels into derivative and parametric kernels, suggested new forms of dynamic kernel that have not previously been applied. These included kernels where the dynamic feature space is defined by derivatives with respect to MLLR transform parameters or to CAT cluster weights. Future work might examine the performance of SVM-based systems using these kernels on the SV task.

- Although the combination of static and dynamic kernels was not found to lead to significant performance gains, this was primarily due to the lack of available training data combined with the extremely high-dimensional feature spaces obtained. Recent NIST evaluations [144] have included an extended data task where up to eight minutes of enrollment data is available per speaker. Under these conditions, the static and and dynamic kernel combination approaches proposed in chapter 7 are more likely to yield gains.

- The refinements described in this thesis may be applied to any speech processing task where dynamic kernel-based algorithms can be applied. Possible avenues for future work include applying these techniques to tasks such as language recognition [183], speaker clustering [82], or to confusable pair discrimination for large vocabulary continuous speech recognition [199].

References

[1] T. Anastasakos, J. McDonaough, R. Schwarz, and J. Makhoul. A compact model for speaker-adaptive training. In *Proc. ICSLP*, 1998. 2.1.4.3

[2] W. Andrews, M. Kohler, J.P. Campbell, and J. Godfrey. Phonetic, idiolectal, and acoustic speaker recognition. In *Proceedings of the IEEE workshop on speaker and language recognition (Odyssey)*, 2001. 4.2.3

[3] W. Andrews, M. Kohler, J.P. Campbell, J. Godfrey, and J. Hernandez-Cordero. Gender-dependent phonetic refraction for speaker recognition. In *Proc. ICASSP*, 2002. 4.2.3, 4.3

[4] B.S. Atal. Effectiveness of linear prediction characteristics of the speech wave for automatic speaker identification and verification. *Journal of the Acoustical Society of America*, 55:1304–1312, 1974. 4.2.1

[5] B.S. Atal and S.L. Hanauer. Speech analysis and synthesis by linear prediction. *Journal of the Acoustical Society of America*, 50:637–655, 1971. 4.2.1

[6] R. Auckenthaler, M. Carey, and H. Lloyd-Thomas. Score normalisation for text-independent speaker verification. *Digital Signal Processing*, 10:42–54, 2000. 4, 4.6.1, 4.6.2

[7] S. Axelrod, R.A. Gopinath, and P.A. Olsen. Modeling with a subspace constraint on inverse covariance matrices. In *Proc. ICSLP*, 2002. 2.1.1

[8] F.R. Bach, G.R.G. Lanckriet, and M.I. Jordan. Multiple kernel learning, conic duality and the SMO algorithm. In *Proc. ICML*, 2004. 6.2

[9] L.R. Bahl, P. Brown, P. de Souza, and R. Mercer. Maximum mutual information estimation of hidden Markov model parameters for speech recognition. In *Proc. ICASSP*, 1986. 2.1.3, 2.1.3.2

[10] C. Bahlman, B. Haasdonk, and H. Burkhaadt. On-line handwriting recognition using support vector machines - a kernel approach. In *Proc. International Workshop on Frontiers in Handwriting Recognition*, 2002. 1, 2.2.3

[11] B. Baker, R. Vogt, M. Mason, and S. Sridharan. Improved phonetic and lexical speaker recognition through MAP adaptation. In *Proc. Odyssey*, 2004. 4.3

[12] C. Barras and J.L. Gauvain. Feature and score normalization for speaker verification of cellular data. In *Proc. ICASSP*, 2003. 4.2.1, 4.2.1, 4.2.1, 4.2.2.1, 8.1

[13] L.E. Baum and J.A. Eagon. An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology. *Bull. Amer. Math. Soc.*, 73:360–363, 1967. 2.1.3.1, 2.1.3.1, 2.2.1

[14] A. Beygelzimer, J. Langford, and B. Zadrozny. Multiclass classification with filter trees. Available from http://hunch.net/∼beygel/filter-tree.pdf, 2007. 2.2.5.3, 2.2.5.3

[15] A. Bhattacharyya. On a measure of divergence between two statistical populations defined by their probability distributions. *Bull. Calcutta Math. Soc.*, 35:99–109, 1943. 3.3, 3.3.3

[16] J.A. Bilmes. Buried Markov models: A graphical-modelling approach to automatic speech recognition. *Computer Speech and Language*, 17(2-3):213–231, 2003. 2.1.2

[17] C. Bishop. *Pattern recognition and machine learning*. Springer, 2006. 1, 4.2.2, 4.2.2.3

[18] T. Bocklet, A. Maier, J.G. Bauer, F. Burkhardt, and E. Noth. Age and gender recognition for telephone applications based on GMM supervectors and support vector machines. In *Proc. ICASSP*, 2008. 7, 7.2

[19] B.E. Boser, L.M. Guyon, and V.N. Vapnik. A training algorithm for optimal margin classifiers. In *Proc. 5th Annual Conference on Computational Learning Theory (COLT 1992)*, 1992. 2.2.3

[20] H. Bourlard and C.J. Wellekens. Links between Markov models and multilayer perceptrons. *IEEE Transactions on pattern analysis and machine intelligence*, 12(12): 1167–1178, 1990. 2.2.1

[21] M.P.S. Brown, W. Grundy, W.N. Lin, N. Cristianini, C.W. Sugnet, T.S. Furrey, M. Ares, and D. Haussler. Knowledge-based analysis of microarray gene expression data by using support vector machines. *Proceedings of the National Academy of Sciences USA*, 97: 262–267, 2000. 1, 2.2.3

[22] N. Brummer, L. Burget, J. Cernocky, O. Glembek, F. Grezl, M. Karafiat, D.A. Van Leeuwen, P. Matejka, P. Schwarz, and A. Strasheim. Fusion of heterogeneous speaker recognition systems in the STBU submission for the NIST speaker recognition evaluation 2006. *IEEE Transactions on Audio, Speech and Language Processing*, 15(7): 2072–2084, 2007. 4.2.1, 4.5.1, 6.1.1, 6.1.1.1, 6.2.1.2

[23] J.P. Campbell. Speaker recognition: a tutorial. *Proceedings of the IEEE*, 85(9):1437–1462, 1997. 1

[24] W.M. Campbell. Generalized linear discriminant sequence kernels for speaker recognition. In *Proc. ICASSP*, 2002. 1.1, 1.2, 3.2, 3.2.1, 4.5.1, 5.1, 7, 7.3, 8.5.2

[25] W.M. Campbell. High-level speaker verification with support vector machines. In *Proc. ICASSP*, 2004. 4.5.1

[26] W.M. Campbell. A covariance kernel for SVM language recognition. In *Proc. ICASSP*, 2008. 8.2.3

[27] W.M. Campbell, J.P. Campbell, D.A. Reynolds, E. Singer, and P.A. Torres-Carrasquillo. Support vector machines for speaker and language recognition. *Computer Speech and Language*, 20:210–229, 2005. 8.2.2

[28] W.M. Campbell, D. Sturim, D.A. Reynolds, and A. Solomonoff. SVM based speaker verification using a GMM supervector kernel and NAP variability compensation. In *Proc. ICASSP*, 2006. 1.1, 1.2, 3.3.1, 3.3.1, 4.2.3, 4.5, 4.5.1, 4.5.2.1, 4.5.2.1, 5, 5.1, 6.3.1, 6.3.3, 7.2, 8.2.1, 8.2.2

[29] W.M. Campbell, J.P. Campbell, D.A. Reynolds, and W. Shen. Speaker verification using support vector machines and high-level features. *IEEE Transactions on Audio, Speech and Language Processing*, 15(7):2085–2094, 2007. 3.1.4, 3.1.4

[30] W.M. Campbell, D. Sturim, W. Shen, D.A. Reynolds, and J. Navrátil. The MIT-LL/IBM 2006 speaker recognition system: High-performance reduced-complexity recognition. In *Proc. ICASSP*, 2007. 6.1.1, 6.2.1.2

[31] C. Champod and D. Meuwly. The inference of identity in forensic speaker recognition. *Speech Communication*, 31:193–203, 2000. 1.2

[32] Y-H. Chao, H-M. Wang, and R-C Chang. GMM-based Bhattacharyya kernel Fisher discriminant analysis for speaker recognition. In *Proc. ICASSP*, 2005. 3.3.3

[33] O. Chapelle, P. Haffner, and V. Vapnik. Support vector machines for histogram-based image classification. *IEEE Transactions on Neural Networks*, 10(5):1055–1064, 1999. 1, 2.2.3

[34] S.S. Chen and R.A. Gopinath. Gaussianization. In *Proc. Advances in NIPS*, 2000. 4.2.2.2

[35] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995. 2.2.3

[36] C. Cortes, P. Haffner, and M. Mohri. Weighted automata kernels - general framework and algorithms. In *Proc. Eurospeech*, 2003. 3.1.3

[37] C. Cortes, P. Haffner, and M. Mohri. Rational kernels: Theory and algorithms. *Journal of Machine Learning Research*, 5:1035–1062, 2004. 3.1.3

[38] T.M. Cover. Geometric and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Transactions on Electronic Computers, EC14(3)*, pages 326–334, June 1965. 2.2.6

[39] K. Crammer and Y. Singer. On the learnability and design of output codes for multiclass problems. *Journal of Machine Learning Research*, 47:35–46, 2002. 2.2.5.4

[40] K. Crammer, Y. Singer, N. Cristianini, J. Shawe-Taylor, and B. Williamson. On the algorithmic interpretation of multi-class kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2001. 2.2.5

[41] N. Cristianini and J. Shawe-Taylor. *Support vector machines and other kernel-based learning methods*. Cambridge University Press, 2000. 2.2.3, 2.2.3

[42] J.N. Darroch and D. Ratcliff. Generalised iterative scaling for log-linear models. *The Annals of Mathematical statistics*, 43(5):1470–1480, 1972. 2.2.1

[43] S.B. Davis and P. Mermelstein. Comparison of parametric representations for monosyllable word recognition in continuously spoken sentences. *IEEE Transactions on Speech and Audio Processing*, 28:357–366, 1980. 4.2, 4.2.1

[44] M.H. DeGroot and M.J Schervish. *Probability and Statistics (3rd Edition)*. Addison-Wesley, 2002. 2.2.6, 3.2.3

[45] N. Dehak and G. Chollet. Support vector GMMs for speaker verification. In *IEEE Odyssey*, 2006. 3.3.2, 3.3.2, 5, 5.1, 7.2, 9.4

[46] N. Dehak, P. Dumouchel, and P. Kenny. Modelling prosodic features with joint factor analysis for speaker verification. *IEEE transactions on Audio, Speech and language processing*, 15(7):2095–2103, 2007. 4.4

[47] N. Dehak, P. Kenny, R. Dehak, O. Glembek, P. Dumouchel, L. Burget, V. Hubeika, and F. Castaldo. Support vector machines and joint factor analysis for speaker verification. In *Proc. ICASSP*, 2009. 4.5.1, 6.3.1

[48] R. Dehak, N. Dehak, P. Kenny, and P. Dumouchel. Linear and non linear GMM supervector machines for speaker verification. In *Proc. ICSLP*, 2007. 1.2, 3.3.2, 4.5, 4.5.1, 8.3.2, 9.4

[49] R. Dehak, N. Dehak, P. Kenny, and P. Dumouchel. Kernel combination for SVM speaker verification. In *Proc. Odyssey*, 2009. 1

[50] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977. 2.1.3.1

[51] T.G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995. 2.2.5.4, 2.2.5.4

[52] V. Digilakis and L. Neumeyer. Speaker adaptation using combined transformation and bayesian methods. In *Proc. ICASSP*, 1995. 2.1.4.2

[53] M.N. Do. Fast approximation of Kullback-Leibler distance for dependence trees and hidden Markov models. *IEEE Signal Processing Letters*, 10:115–118, 2003. 3.3.1, 5.2

[54] G. Doddington. Speaker recognition based on idiolectal differences between speakers. In *Proc. Eurospeech*, 2001. 4.2.3

[55] G. Doddington, M.A. Przybocki, A. Martin, and D.A. Reynolds. The NIST speaker recognition evaluation- overview, methodology, systems, results, perspective. *Speech Communication*, 2:263–286, 2000. 4.2, 4.2.1

[56] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. John Wiley & Sons, 2nd edition, 2001. 2.1, 2.1

[57] R. Fernández-Lorenzana, F. Pérez-Cruz, J.M. García-Cabellos, C. Pelaez-Moreno, A. Gallardo-Antolín, and D. Diaz de Maria. Some experiments on speaker-independent isolated digit recognition using SVM classifiers. In *Proc. ISCA Tutorial workshop on non-linear speech processing*, 2003. 7.4.1

[58] M. Ferras, C. Leung, C. Barras, and J.L. Gauvain. Constrained MLLR for speaker recognition. In *Proc. ICASSP*, 2007. 3.2.6, 6.3.1

[59] J. Friedman. Another approach to polychotomous classification. Technical report, Department of Statistics, Seaford University, 1996. 2.2.5.1

[60] S. Furui. Cepstral analysis technique for automatic speaker verification. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 29(2):254–272, 1981. 4, 4.2.1, 4.2.2, 4.2.2.1

[61] S. Furui. Speaker-independent isolated word recognition using dynamic features of speech spectrum. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 34 (1):52–59, 1986. 1

[62] M.J.F. Gales. The generation and use of regression class trees for MLLR adaptation. Technical Report CUED/F-INFENG/TR.263, Cambridge University Engineering Department, 1996. 2.1.4.2

[63] M.J.F. Gales. Maximum likelihood linear transformations for HMM-based speech recognition. *Computer Speech and Language*, 12:75–98, 1998. 2.1.4.2, 2.1.4.2

[64] M.J.F. Gales. Semi-tied covariance matrices for hidden Markov models. *IEEE Transactions on Speech and Audio Processing*, 7:272–281, 1999. 2.1.1

[65] M.J.F. Gales. Cluster adaptive training of hidden Markov models. *IEEE Transactions on Speech and Audio Processing*, 8(4):417–428, 2000. 2.1.4.3, 2.1.4.3, 2.1.4.3, 2.1.4.3

[66] M.J.F. Gales and F. Flego. Discriminative classifiers and generative kernels for noise robust speech recognition. Technical Report CUED/F-INFENG/TR.605, Cambridge University Engineering Department, August 2008. 7.4.2, 8.5.1

[67] M.J.F. Gales and C. Longworth. Discriminative classifiers with generative kernels for noise robust ASR. In *Proc. Interspeech*, 2008. 3.2.3, 7.4.2, 8.2.1, 8.5.1

[68] M.J.F. Gales and P.C. Woodland. Mean and variance adaptation within the MLLR framework. *Computer Speech and Language*, 10:249–264, 1996. 2.1.4.2

[69] M.J.F Gales and S.J. Young. The application of hidden markov models in speech recognition. *Foundation and Trends in Signal Processing*, 1(3):195–304, 2008. 2.1.2

[70] C. Gautier. Filter trees for combining binary classifiers. Master's thesis, University of Cambridge, 2008. 2.2.5.3

[71] J.L. Gauvain and C-H. Lee. Maximum A-Posteriori estimation for multivariate Gaussian mixture observations of Markov chains. *IEEE Transactions on Speech and Audio Processing*, 2:291–298, 1994. 2.1.4.1, 2.1.4.1, 2.1.4.1, 2.1.4.1

[72] M. Genton. Classes of kernels for machine learning: A statistics perspective. *Journal of Machine Learning Research*, 2:299–312, 2000. 6.3.3

[73] L. Gillick and S. Cox. Some statistical issues in the comparison of speech recognition algorithms. In *Proc. ICASSP*, 1989. 4.7.5

[74] A. Gunawardana, M. Mahajan, A. Acero, and J.C. Platt. Hidden conditional random fields for phone classification. In *Proc. Interspeech*, 2005. 2.2.1, 1, 2.2.2, 2.2.2, 2.2.2

[75] W. Guo, Y. Long, Y. Li, L. Pan, E. Wang, and L. Dai. IFLY system for the NIST 2008 speaker recognition evaluation. In *Proc. ICASSP*, 2009. 4.5, 4.5.1

[76] J. Hamaker and J. Picone. A sparse modeling approach to speech recognition based on relevance vector machines. In *Proc. ICSLP*, 2002. 2.2.4, 2.2.4

[77] A. Hannani and D. Petrovska-Delacretaz. Comparing data-driven and phonetic n-gram systems for text-independent speaker verification. In *Proc. First IEEE International Conference on Biometrics: Theory, Applications, and Systems*, 2007. 4.3

[78] A. Hannani and D. Petrovska-Delacretaz. Data-driven high-level information for text-independent speaker verification. In *Proc. IEEE workshop on Automatic identification advanced technologies*, 2007. 4.5.1

[79] A.O. Hatch, B. Peskin, and A. Stolcke. Improved phonetic speaker recognition using lattice decoding. In *Proc. ICASSP*, 2005. 4.3

[80] A.O. Hatch, S. Kajarekar, and A. Stolcke. Generalised linear kernels for one-versus-all classification: application to speaker recognition. In *Proc. ICASSP*, 2006. 4.5.2.2, 4.5.2.2

[81] A.O. Hatch, S. Kajarekar, and A. Stolcke. Within-class covariance normalisation for SVM-based speaker recognition. In *Proc. Interspeech*, 2006. 4.5.2.2

[82] O. Hazrati, S.M. Ahmadi, and O. Sadjadi. Kernel-based speaker clustering for rapid speaker adaptation. In *Proc. Fifth internation conference on information technology: New Generations*, 2008. 9.4

[83] L. Heck and M. Weintraub. Handset-dependent background models for robust text-independent speaker verification. In *Proc. ICASSP*, 1997. 4.6

[84] G. Heigold, R. Schluter, and H. Ney. On the equivalence of Gaussian HMM and Gaussian HMM-like hiddeen conditional random fields. In *Proc. Interspeech*, 2007. 2.2.2

[85] G. Heigold, P. Lehnen, R. Schluter, and H. Ney. On the equivalence of Gaussian and log-linear HMMs. In *Proc. Interspeech*, 2008. 2.2.2

[86] H. Hermansky. Perceptual linear predictive (PLP) analysis of speech. *Journal of the Acoustical Society of America*, 87:1738–1752, 1990. 4.2, 4.2.1

[87] J.R. Hershey and P.A. Olsen. Approximating the Kullback Leibler divergence between Gaussian mixture models. In *Proc. ICASSP*, 2007. 5, 5.1, 5.2.1, 5.2.2, 5.2.2, 8.3.1, 1

[88] C. Hildreth. A quadratic programming procedure. *Naval Research Logistics Quarterly*, 4:79–85, 1957. 2.2.3

[89] C-J. Hsu and C-W. Lin. A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, 13:415–425, 2002. 2.2.5.1

[90] T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *Proc. NIPS*, 1999. 1.1, 3.2, 3.2.3, 3.2.3, 5.1, 7.2, 8.2.1

[91] A.K. Jain, R.P.W Duin, and J. Mao. Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37, 2000. 1

[92] T. Jebara and R. Kondor. Bhattacharyya and expected likelihood kernels. In *Proc. 16th Annual Conference on Learning Theory (COLT 2003)*, 2003. 3.3.3, 3.3.3

[93] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proc. ECML*, 1998. 3.1, 3.1.1, 3.1.1

[94] T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges and A. Smola, editor, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1999. 2.2.3, 2.2.3, 6.2.1.4, 8.2.1

[95] B-H. Juang, W. Chou, and C-H. Lee. Minimum classification error rate methods for speech recognition. *IEEE Transactions on Speech and Language Processing*, 5(3):257–265, 1997. 2.1.3

[96] S. Julier and J.K Uhlmann. A general method for approximating nonlinear transformations of probability distributions. Technical report, Dept. of Engineering Science, University of Oxford, 1996. 1

[97] D. Jurafsky and J.H. Martin. *Speech and Language Processing*. Prentice Hall, 2000. 2.1.2

[98] T. Kailath. The divergence and Bhattacharyya distance measures in signal selection. *IEEE Transactions on Communication Technologies*, 15:49–52, 1967. 3.3.3

[99] Z. Kaiser, B. Horvat, and Z. Kacic. A novel loss function for the overall risk criterion based discriminative training of HMM models. In *Proc. ICSLP*, 2000. 2.1.3

[100] S. Kajarekar. Four weightings and a fusion: a cepstral-SVM system for speaker recognition. In *Proc. ASRU*, 2005. 4.5.2.2

[101] S. Kajarekar and A. Stolcke. NAP and WCCN: comparison of approaches using MLLR-SVM speaker verification system. In *Proc. ICASSP*, 2007. 4.5.2.2

[102] S. Kajarekar, N. Scheffer, M. Graciarena, E. Shriberg, A. Stolcke, L. Ferrer, and T. Bocklet. The SRI NIST 2008 speaker recognition system evaluation system. In *Proc. ICASSP*, 2009. 4.2.1, 4.2.3, 4.5, 4.5.1

[103] Z.N. Karam and W.M. Campbell. A new kernel for SVM MLLR based speaker recognition. In *Proc. Interspeech*, 2007. 3.2.6, 8.2.4

[104] Z.N. Karam and W.M. Campbell. A multi-class MLLR kernel for SVM speaker recognition. In *Proc. ICASSP*, 2008. 3.2.6, 3.2.6

[105] T. Kemp and T. Schaaf. Estimating confidence using word lattices. In *Proc. Eurospeech*, 1997. 2.1.3

[106] P. Kenny. Joint factor analysis of speaker and session variability: theory and algorithms. Technical report, CRIM, 2005. Available from http://www.crim.ca/perso/patrick.kenny/. 4.4.1, 4.4.1, 4.4.2

[107] P. Kenny, G. Boulianne, P. Ouellet, and P. Dumouchel. Speaker adaptation using an eigenphone basis. *IEEE transactions on speech and audio processing*, 12(6):579–589, 2004. 4.4.2

[108] P. Kenny, G. Boulianne, and P. Dumouchel. Eigenvoice modelling with sparse training data. *IEEE transactions on speech and audio processing*, 13(2):345–354, 2005. 4.4.2

[109] P. Kenny, G. Boulianne, P. Ouellet, and P. Dumouchel. Joint factor analysis versus eigenchannels in speaker recognition. *IEEE transactions on Audio, Speech and language processing*, 15(4):1435–1447, 2007. 4, 4.4

[110] P. Kenny, G. Boulianne, P. Ouellet, and P. Dumouchel. Speaker and session variability in GMM-based speaker verification. *IEEE Transactions on Audio, Speech and Language Processing*, 15(4):1448–1460, 2007. 4.4, 4.4.1

[111] P. Kenny, N. Dehak, P. Ouellet, V. Gupta, and P. Dumouchel. Development of the primary CRIM system for the NIST 2008 speaker recognition evaluation. In *Proc. Interspeech*, 2008. 4.4, 4.6.1

[112] R. Kondor and T. Jebara. A kernel between sets of vectors. In *Proc. ICML*, 2003. 3.3.3

[113] M.H. Kuhn. Access control by means of automatric speaker verification. *Journal of Physics E: Scientific instruments.*, 13(1):85–86, 1980. 1.2

[114] R. Kuhn, L. Nguyen, J-C. Junqua, L. Goldwasser, N. Niedzielski, S. Finke, K. Field, and M. Contolini. Eigenvoices for speaker adaptation. In *Proc. ICSLP*, 1998. 2.1.4.3

[115] S. Kullback and R.A. Leibler. On information and sufficiency. *The Annals of Mathematical statistics*, 22(1):79–86, 1951. 3.3, 3.3.1, 5.1

[116] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML*, 2001. 2, 2.1.3.2, 2.2.1, 2.2.1, 2.2.1, 2.2.1

[117] J. Langford and A. Beygelzimer. Sensitive error correcting output codes. In *Proc. 18th Annual Conference on Learning Theory (COLT 2005)*, 2005. 2.2.5.4

[118] M.I. Layton. *Kernel Methods for Classifying Variable Length Data*. PhD thesis, Cambridge University, 2006. 6.3.2

[119] M.I. Layton and M.J.F. Gales. Acoustic modelling using continuous rational kernels. In *Proc. MLSP*, 2005. 3.2.3

[120] K-A. Lee, C.H. You, H. Li, and T. Kinnunen. A probabilistic sequence kernel for speaker verification. In *Proc. ICSLP*, 2007. 3.2, 3.2.5, 3.2.5, 4.5.1, 6.3.2

[121] C.J. Leggetter and P.C. Woodland. Speaker adaptation of continuous density HMMs using linear regression. In *Proc. ICSLP*, 1994. 2.1.4.2, 2.1.4.2

[122] C.J. Leggetter and P.C. Woodland. Maximum likelihood linear regression speaker adaptation of contiuous density HMMs. *Computer speech and language*, 9(2):171–185, 1995. 2.1.4.2, 3.2.6

[123] C. Leslie and R. Kuang. Fast string kernels using inexact matching for protein sequences. *Journal of Machine Learning Research*, 5:1435–1455, 2004. 3.1, 3.1.2, 3.1.3, 3.1.3

[124] C. Leslie, E. Eskin, and W.S. Noble. The spectrum kernel: A string kernel for SVM protein classification. In *Proceedings of the Pacific Symposium on Biocomputing*, August 2002. 3.1.2, 3.1.2

[125] C. Leslie, E. Eskin, J. Weston, and W.S. Noble. Mismatch string kernels for SVM protein classification. In *Proc. NIPS*, 2002. 3.1.2

[126] H. Li, B. Ma, K-A. Lee, H. Sun, D. Zu, K.C. Sim, C.H. You, R. Tong, I. Kärkkäinen, C-L. Huang, V. Pervouchine, W. Guo, Y. Li, L. Dai, M. Nosratighods, T. Tharmarajah, J.Epps, E. Ambikairajah, E-S. Chng, T. Schultz, and Q. Jin. The I4U system in NIST 2008 speaker recognition evaluation. In *Proc. ICASSP*, 2009. 4.5, 4.5.1, 4.6.1, 6.1.1

[127] A. Ljolje. The importance of cepstral parameter correlations in speech recognitions. *Computer speech and language*, 8:223–232, 1994. 4.2.2, 4.2.2.3

[128] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444, 2002. 3.1, 3.1.3, 3.1.3

[129] C. Longworth and M.J.F. Gales. Discriminative adaptation for speaker verification. In *Proc. ICSLP*, 2006. (document), 4.3.1

[130] C. Longworth and M.J.F. Gales. Derivative and parametric kernels for speaker verification. In *Proc. ICSLP*, 2007. 1

[131] C. Longworth and M.J.F. Gales. Multiple kernel learning for speaker verification. In *Proc. ICASSP*, 2008. 1

[132] C. Longworth and M.J.F. Gales. A generalised derivative kernel for speaker verification. In *Proc. ICSLP*, 2008. (document)

[133] C. Longworth and M.J.F. Gales. Combining derivative and parametric kernels for speaker verification. *IEEE Transactions on Audio, Speech and Language Processing*, 17 (4):748–757, 2009. (document)

[134] C. Longworth, R.C. van Dalen, and M.J.F. Gales. Variational dynamic kernels for speaker verification. In *Proc. ICSLP*, 2009. (document)

[135] J. Louradour, K. Daoudi, and F. Bach. Feature space Mahalanobis sequence kernels: Application to SVM speaker verification. *IEEE Transactions on Audio, Speech and Language Processing*, 15:2465–2475, 2007. 7.3.1.2

[136] W. Macherey, L. Haferkamp, R. Schluter, and H. Ney. Investigations on Error Minimizing Training Criteria for Discriminative Training in Automatic Speech Recognition. In *Proc. Interspeech*, 2005. 2.1.3

[137] D.J.C. MacKay. Probable networks and plausible predictions — a review of practical Bayesian methods for supervised neural networks. *Network: Computation in Neural Systems*, 6:469–505, 1995. 2.2.4, 2.2.4

[138] M. Mahajan, A. Gunawardana, and A. Acero. Training algorithms for hidden conditional random fields. In *Proc. ICASSP*, 2006. 2.2.2

[139] M.W. Mak and S.Y. Kung. Estimation of elliptical basis function parameters by the EM algorithm with application to speaker verification. *IEEE Transactions Neural Networks*, 11(4):961–969, 2000. 3.2.5

[140] J. Makhoul. Linear prediction: A tutorial review. *Proceedings of the IEEE*, 63(4): 561–580, 1975. 4.2.1

[141] A. Malegaonkar, A. Ariyaeeinia, P. Sivakumaran, and S. Pillay. On the discrimination capabilities of speech cepstral features. In *Proc. European workshop on biometrics and identity management*, 2008. 4.2.1

[142] J. Mariéthoz and S. Bengio. A kernel trick for sequences applied to text-independent speaker verification systems. *Pattern Recognition*, 40:2315–2324, 2007. 1.1, 7.3.1.1

[143] A. Martin. The NIST year 2002 speaker recognition evaluation plan. Available from http://www.itl.nist.gov/iad/mig/tests/sre/2002, 2002. 4.7.2, 8.1

[144] A. Martin. The NIST year 2008 speaker recognition evaluation plan. Available from http://www.itl.nist.gov/iad/mig/tests/sre/2008, 2008. 1.2, 4.1, 4.7.2, 9.4

[145] A. Martin and A.N. Le. The current state of language recognition: NIST 2005 evaluation results. In *Proc. Odyssey*, 2006. 1

[146] A. Martin, G. Doddington, T. Kamm, M. Ordowski, and M.A. Przyboci. The DET curve in assessment of detection task performance. In *Proc. Eurospeech*, 1997. 4.7.4

[147] A. McCallum. Efficiently inducing features of conditional random fields. In *Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence*, 2003. 2.2.1

[148] A. McCallum, D. Freitag, and F. Pereira. Maximum entropy Markov models for information extraction and segmentation. In *Proceedings of the 17th International Conference on Machine Learning*, 2000. 2.2.1

[149] J. Mercer. Functions of positive and negative type and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society London. Series A*, 209:415–446, 1909. 2.2.6

[150] P. Moreno, P. Ho, and B. Vasconcelos. A Kullback-Leibler divergence based kernel for SVM classification in multimedia applications. In *Proc. Advances in NIPS*, 2004. 3.3.2, 6.3.3

[151] K. Na, B. Jeon, D. Chang, S. Chae, and S. Ann. Discriminative training of hidden Markov models using overall risk criterion and reduced gradient method. In *Proc. Eurospeech*, pages 97–100, 1995. 2.1.3

[152] A. Nadas. A decision theoretic formulation of a training problem in speech recognition and a comparison of training by unconditional versus conditional maximum likelihood. *IEEE Transactions Acoustics, Speech and Signal Processing*, 31(4):812–817, 1983. 2.1.3, 2.1.3.2, 2.2.1

[153] Y. Normandin and S.D. Morgera. An improved MMIE training algorithm for speaker-independent, small vocabulary, continuous speech recognition. In *Proc. ICASSP*, 1991. 2.1.3.2

[154] M. Nosratighods, T. Thiruvaran, J. Epps, E. Ambikairajah, B. Ma, and H.Li. Evaluation of a fused FM and cepstral-based speaker recogniton system on the NIST 2008 SRE. In *Proc. ICASSP*, 2009. 6.1.1

[155] P.A Olsen and S. Dharanipragada. An efficient integrated gender detection scheme and time-mediated averaging of gender dependent acoustic models. In *Proc. Eurospeech*, 2003. 1

[156] J. Pelecanos and S. Sridharan. Feature warping for robust speaker verification. In *Proc. ISCA Workshop on Speaker Recognition - 2001: A Speaker Odyssey*, 2001. 4, 4.2.2, 4.2.2.1, 4.2.2.2, 4.2.2.2, 8.1

[157] S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393, 1997. 2.2.1, 2.2.1

[158] S. Pigeon, P. Druyts, and P. Verlinde. Applying logistic regression to the fusion of the NIST'99 1-speaker submissions. *Digital Signal Processing*, pages 237–248, 2000. 4.5.1, 6.1.1.1

[159] R.L. Plackett. Karl Pearson and the chi-squared test. *International statistical review*, pages 59–72, 1983. 4.7.5

[160] J.C. Platt. Fast training of support vector machines using Sequential Minimal Optimization. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1999. 2.2.3

[161] J.C. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin DAGs for multiclass classification. In *Proc. Advances in NIPS*, 2000. 2.2.5.2, 2.2.5.2

[162] D. Povey. *Discriminative Training for Large Vocabulary Speech Recognition*. PhD thesis, University of Cambridge, 2003. 2.1.3

[163] D. Povey and P.C. Woodland. Minimum phone error and I-smoothing for improved discriminative training. In *Proc ICASSP*, 2002. 2.1.3

[164] D. Povey, M.J.F. Gales, D.Y. Kim, and P.C. Woodland. MMI-MAP and MPE-MAP for acoustic model adaptation. In *Proc. Eurospeech*, 2003. 2.1.3.2, 2.1.3.2, 2.1.4.1

[165] A. Quattoni, M. Collins, and T. Darrell. Conditional random fields for object recognition. In *Proc. Advances in NIPS*, 2005. 2, 2.2.2, 2.2.2, 2.2.2

[166] A. Quattoni, S. Wang, L-P. Morency, M. Collins, and T. Darrell. Hidden conditional random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29 (10):1848–1852, 2007. 2.2.2

[167] L.A. Rabiner. A tutorial on hidden Markov models and selective applications in speech recognition. *Proceedings of the IEEE*, 77:257–289, 1989. 2.1.2, 2.1.3.1

[168] A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet. More efficiency in multiple kernel learning. In *Proc. ICML*, pages 775–782, 2007. 1.2, 6, 6.2.1, 6.2.1.1, 6.2.1.4, 6.2.1.4, 8.4.1

[169] D.A. Reynolds. Speaker verification using adapted Gaussian mixture models. *Digital Signal Processing*, 10:19–41, 2000. 1.2, 4.3.1

[170] D.A. Reynolds. Speaker identification and verification using Gaussian mixture models. *Speech Communication*, 17:91–108, 1995. 1.2, 4.2.2.1, 4.3

[171] D.A. Reynolds. Comparison of background normalisation methods for text-independent speaker verification. In *Proc. Eurospeech*, 1997. 4, 4.6.1, 4.6.1

[172] D.A. Reynolds. An overview of automatic speaker recognition technology. In *Proc. ICASSP*, 2002. 1

[173] H. Robbins. An empirical bayes approach to statistics. In *Proc. Third Berkeley symposium on Math. Statist. and Prob.*, pages 157–164, 1955. 2.1.4

[174] A.E. Rosenberg, J. Delong, C-H. Lee, B-H. Juang, and F.K. Soong. The use of cohort normalised scores for speaker verification. In *Proc. ICSLP*, 1992. 4.3.1

[175] A-V.I. Rosti and M.J.F. Gales. Factor analysed hidden Markov models. In *Proc. ICASSP*, 2002. 2.1.2

[176] A-V.I. Rosti and M.J.F. Gales. Switching linear dynamical systems for speech recognition. Technical Report CUED/F-INFENG/TR461, Cambridge University, 2003. 2.1.2

[177] R. Schluter and W. Macherey. Comparison of discriminative training criteria. In *Proc. ICASSP*, 1998. 2.1.3.2

[178] B. Schölkopf and A.J. Smola. *Learning with kernels*. MIT Press, 2002. 2.2.3

[179] B. Schölkopf, A.J. Smola, and K-R. Muller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998. 4.5.2.1

[180] F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proc. Human Language Technology, HLT/NAACL-03*, 2003. 2.2.1, 2.2.1

[181] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004. 1.1, 2.2.6, 2.2.6, 3.3.2, 7.1

[182] E. Shriberg, L. Ferrer, A. Venkataraman, and S. Kajarekar. SVM modelling of "SNERF-grams" for speaker recognition. In *Proc. Interspeech*, 2004. 4.2.3, 4.5.1

[183] E. Singer, P.A. Torres-Carrasquillo, T.P. Gleason, W.M. Campbell, and D.A. Reynolds. Acoustic, phonetic, and discriminative approaches to automatic language identification. In *Proc. Eurospeech*, 2003. 9.4

[184] N.D. Smith. *Using Augmented Statistical Models and Score Spaces for Classification*. PhD thesis, University of Cambridge, September 2003. 3.2.3, 3.2.4

[185] 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. 3.2, 3.2.4, 6.3.2

[186] A. Solomonoff, W.M. Campbell, and C. Quillen. Channel compensation for SVM speaker recognition. In *Proc. Odyssey*, 2004. 4, 4.5.2.1

[187] A. Solomonoff, W.M. Campbell, and I. Boardman. Advances in channel compensation for SVM speaker recognition. In *Proc. ICASSP*, 2005. 4.5.2.1, 4.5.2.1, 4.5.2.1

[188] S. Sonnenburg, G. Rätsch, and C. Schäfer. A general and efficient multiple kernel learning algorithm. In *Proc. Advances in NIPS*, 2005. 1.2, 6.2.1

[189] A. Stolcke, L. Ferrer, S. Kajarekar, E. Shriberg, and A. Venkataraman. MLLR transforms as features in speaker recognition. In *Proc. Interspeech*, 2005. 1.1, 1.2, 3.2, 3.2.6, 3.2.6, 4.2.3, 4.5.1, 6.3.1

[190] A. Stolcke, L. Ferrer, and S. Kajarekar. Improvements in MLLR-transform based speaker recognition. In *Proc. Odyssey*, 2006. 1.2, 3.2.6, 3.2.6, 4.5

[191] D. Sturim and D.A. Reynolds. Speaker adaptive cohort selection for tnorm in text-independent speaker recognition. In *Proc. ICASSP*, 2005. 4.6.3

[192] D. Sturim, D.A. Reynolds, E. Singer, and J.P. Campbell. Speaker indexing in large audio databases using anchor models. In *Proc. ICASSP*, 2001. 3.2.2, 3.2.2

[193] J. Swets. *Signal detection theory and Roc analysis in psychology and diagnostics: collected papers*. Lawrence Erlbaum Associates, 1996. 4.7.3

[194] M.E. Tipping. The relevance vector machine. In *Proc. Advances in NIPS*, 2000. 2.2.4

[195] M.E. Tipping. Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, pages 211–244, 2001. 2.2.4, 2.2.4, 2.2.4, 2.2.4, 2.2.5

[196] F. Topsoe. Some inequalities for information divergence and related measures of discrimination. *Journal of Inequalities in Pure and Applied Mathematics*, 46(4):1602–1609, 1999. 3.3.3

[197] K. Tsuda, T. Kin, and K. Asai. Marginalized kernels for biological sequences. *Bioinformatics*, 18(90001):S268–S275, 2002. 3.1, 3.1.5, 3.1.5

[198] V. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, 1998. 1, 2, 2.2.3, 2.2.3, 2.2.5, 2.2.6

[199] V. Venkataramani, S. Chakrabartty, and W. Byrne. Support vector machines for segmental minimum Bayes risk decoding of continuous speech. In *Proc. ASRU*, 2003. 9.4

[200] R. Vogt, B. Baker, and S. Sridharan. Modelling session variability in text-independent speaker verification. In *Proc. Interspeech*, 2005. 4.4, 4.6.2

[201] V. Vural and J. Dy. A hierarchical method for multi-class support vector machines. In *Proc. ICML*, 2004. 2.2.5.2, 2.2.5.2

[202] H. Wallach. Efficient training of conditional random fields. Master's thesis, University of Edinburgh, 2002. 2.2.1

[203] V. Wan and W.M. Campbell. Support vector machines for speaker verification and identification. In *Proc. Neural Networks for Signal Processing X*, 2000. 7.4.1

[204] V. Wan and S. Renals. Evaluation of kernel methods for speaker verification and identification. In *Proc. ICASSP*, 2002. 1.1, 3, 6.3.3

[205] V. Wan and S. Renals. Speaker verification using sequence discriminant support vector machines. *IEEE Transactions Speech and Audio Processing*, 13(2):203–210, 2004. 1.1, 1.2, 2.2.6, 4.5.1, 6.3.2, 8.2.1, 8.2.2, 8.2.2

[206] S. Wang, A. Quattoni, L-P. Morency, D. Demirdjian, and T. Darrell. Hidden conditional random fields for gesture recognition. In *Proc. CVPR*, 2006. 2.2.2

[207] S. Watanabe, Y. Minami, A. Nakamura, and N. Ueda. Application of variational Bayesian approach to speech recognition. In *Proc. Advances in NIPS*, 2003. 2.1.4

[208] J. Weston and C. Watkins. Multi-class support vector machines. Technical Report CSD-TR-98-04, Royal Holloway, University of London, May 1998. 2.2.5

[209] P.C. Woodland and D. Povey. Automatic speech recognition: challenges for the new millenium. In *Proc. ASR2000*, 2000. 2.1.3.2

[210] P.C. Woodland and D. Povey. Large scale discriminative training of hidden Markov models for speech recognition. *Computer Speech and Language*, 16:25–48, 2002. 2.1.3.2, 2.1.3.2

[211] H. Yang, Y. Dong, X. Zhao, L. Lu, and H. Wang. Cluster adaptive training weights as features in SVM-based speaker verification. In *Proc. Interspeech*, 2007. 1.1, 1.2, 3.2, 3.2.7, 4.5.1, 6.3.1

[212] F. Yates. Contingency tables involving small numbers and the χ^2 test. *Journal of the royal statistical society*, Supplement 1:217–235, 1934. 4.7.5

[213] C.H. You, K-A. Lee, and H. Li. An SVM kernel with GMM-supervector based on the Bhattacharyya distance for speaker recognition. *IEEE Signal processing letters*, 16: 49–52, 2009. 3.3.3, 4.5.1

[214] C.H. You, K-A. Lee, and H. Li. A GMM-supervector kernel with the Bhattacharyya distance for SVM-based speaker recognition. In *Proc. Interspeech*, 2009. 3.3.3

[215] S.J. Young. A review of large-vocabulary continuous-speech recognition. *IEEE Signal processing magazine*, 13:45–57, 1996. 1.2

[216] S.J. Young, G. Evermann, M.J.F. Gales, T. Hain, D. Kershaw, X. Liu, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. C. Woodland. *The HTK Book, version 3.4*. Cambridge, UK, 2006. 8.2.1

[217] K. Yu. *Adaptive training for large vocabulary continuous speech recognition*. PhD thesis, University of Cambridge, July 2006. 2.1.3.2, 2.1.4.3, 5, 5.2.2

[218] X. Zhao, Y. Dong, H. Yang, J. Zhao, and H. Wang. SVM-based speaker verification by location in the space of reference speakers. In *Proc. ICASSP*, 2007. 3.2.2

[219] X. Zhao, Y. Dong, H. Yang, L. Lu, and H. Wang. Nonlinear kernel nuisance attribute projection for speaker verification. In *Proc. ICASSP*, 2008. 4.5.2.1

[220] X. Zhu, Y. Chen, J. Liu, and R. Liu. Feature selection in mandarin large vocabulary continuous speech recognition. In *Proc. ICOST*, 2002. 4.2.1

[221] E. Zwicker. Subdivision of the audible frequency range into critical bands. *Journal of the Acoustical Society of America*, 33, 1961. 4.2.1