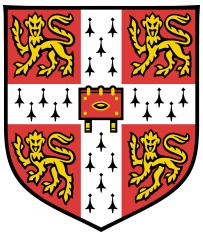
Augmented Statistical Models for Classifying Sequence Data

Martin Layton

Corpus Christi College
University of Cambridge



September 2006

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 [66, 69], two journal articles [36, 68], two workshop papers [35, 67] and a technical report [65]. The length of this thesis including appendices, bibliography, footnotes, tables and equations is approximately 60,000 words. This thesis contains 27 figures and 20 tables.

Summary

Many important applications, including speech recognition, document categorisation and bioinformatics, require the classification of variable-length sequences of observations. This is often performed using statistical techniques that learn the relationships between class labels and observation sequences using sets of labelled training data. A number of standard generative and discriminative statistical models are commonly used. Unfortunately, the assumptions associated with these models are often incorrect for many applications, potentially affecting classification accuracy. This thesis focuses upon systematic techniques for relaxing model assumptions and allowing additional dependencies to be represented.

The first approach discussed in this thesis are augmented statistical models. These are a powerful approach for extending standard generative (base) models using a set of additional sufficient statistics. Augmented model sufficient statistics are typically defined using a local exponential expansion of the base model, and allow a wide range of complex temporal and spatial dependencies to be modelled. In this work, the properties of both first-order and higher-order augmented models are examined. A maximum-margin distance-based training algorithm is then proposed. This directly optimises the decision boundary between pairs of augmented models, enabling model parameters to be estimated even when the normalisation terms cannot be calculated. Algorithms for base model parameter estimation are also proposed.

The second contribution of this thesis are continuous rational kernels. These are a new form of dynamic kernel that combines latent-variable generative models and weighted finite-state rational kernels to create flexible classifiers of continuous-observation sequences. By varying the generative models and rational kernels, continuous rational kernels allow a wide range of different kernel feature-spaces (and hence dependencies) to be modelled using only a small number of efficient and standardised algorithms. In addition to kernel feature-spaces, continuous rational kernels can also be used as a simple approach for generating the vectors of sufficient statistics used in augmented statistical models.

The final contribution of this thesis are conditional augmented (C-Aug) models. These use the same sufficient statistics as standard augmented models but are defined within a conditional-probability framework. The use of augmented model sufficient statistics allows C-Aug models to share many of the benefits of generative augmented models, including the ability to represent a wide range of temporal and spatial dependencies. Furthermore, by directly modelling the posterior probabilities of the correct class labels, C-Aug models are an inherently discriminatory approach that avoids many of the normalisation difficulties of standard augmented models. This allows C-Aug models to be applied to a wide range of classification tasks. Finally, a lattice-based framework is proposed that allows C-Aug models to be applied to simple continuous speech recognition tasks.

Acknowledgements

First, I would like to thank my supervisor Mark Gales. By providing the right balance of advice, suggestions and criticism, he helped to make this work possible. In particular, I am grateful for the time and support he has given me over the last three years, regardless of the direction I decided to pursue. I am also grateful to the Schiff Foundation for kindly funding my studies in Cambridge, and for enabling me to attend several international conferences and workshops. I would also like to thank Corpus Christi College for providing financial support for conference expenses.

Over the last three years, I have benefited greatly from interactions with other members of the Machine Intelligence Laboratory at Cambridge University. Although there are too many people to thank everyone individually, I would like to extend my thanks (in no particular order) to Xunying Liu, Khe Chai Sim, Kai Yu and Chris Longworth for many useful discussions. I also owe my thanks to Patrick Gosling and Anna Langley for maintaining the computing systems on which much of this work depended. I would like to thank Katy Jones and my father for proof-reading various parts of this thesis.

Special thanks go to Katy who has helped and supported me throughout much of my time in Cambridge. Finally, my biggest thanks go to my parents for their unwavering support and encouragement over the years.

Acronyms

A-GMM	Augmented Gaussian mixture model
A-HMM	Augmented hidden Markov model
BMM	Buried Markov model
C-Aug	Conditional augmented (model)
CG	Conjugate gradient
CML	Conditional maximum likelihood
CMN	Cepstral mean normalisation
CN	Confusion network
CRF	Conditional random field
CTS	Conversational telephone speech
CVN	Cepstral variance normalisation
EM	Expectation maximisation
GD	Gradient descent
GMM	Gaussian mixture model
HCRF	Hidden conditional random field
HLDA	Heteroscedastic linear discriminant analysis
HMM	Hidden Markov model
KKT	Karush-Kuhn-Tucker
LVCSR	Large vocabulary continuous speech recognition
MBR	Minimum Bayes risk
MCE	Minimum classification error
MFCC	Mel-frequency cepstral coefficient
ML	Maximum likelihood
MM	Maximum margin
MMI	Maximum mutual information
MPE	Minimum phone error
MWE	Minimum word error
PDF	Probability density function
PLP	Perceptual linear prediction
RBF	Radial basis function
RVM	Relevance vector machine
SCG	Scaled conjugate gradient
SLDS	Switching linear dynamical system
SSK	String subsequence kernel
SVM	Support vector machine
VTLN	Vocal-tract length-normalisation
WER	Word error rate

Mathematical Notation

$\boldsymbol{A}$	matrix
$\boldsymbol{A}^{\mathrm{T}}$	transpose of $\boldsymbol{A}$
$\boldsymbol{A}^{-1}$	inverse of $\boldsymbol{A}$
$\lvert\boldsymbol{A}\rvert$	determinant of $\boldsymbol{A}$
$\boldsymbol{x}$	vector
x_j	j-th element of $\boldsymbol{x}$
$\langle \boldsymbol{x}, \boldsymbol{y} \rangle$	inner-product of $\boldsymbol{x}$ and $\boldsymbol{y}$
$\boldsymbol{x} < \boldsymbol{y}$	$x_i < y_i$, $\forall i \in [1, d]$, where $\boldsymbol{x}$ and $\boldsymbol{y}$ are d-dimensional vectors
$\mathrm{vec}(\boldsymbol{A})$	vector form of matrix $\boldsymbol{A}$
$\nabla_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha})$	vector derivative of a function $f(\boldsymbol{\alpha})$ with respect to $\boldsymbol{\alpha}$
$\mathcal{E}_{\boldsymbol{O}}\{f(\boldsymbol{O})\}$	expected value of $f(\boldsymbol{O})$
$p(\boldsymbol{O}; \boldsymbol{\lambda})$	probability density function for an observation sequence $\boldsymbol{O}$
$\mathcal{N}(\boldsymbol{o}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$	Gaussian PDF with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$
$P(\theta \lvert \boldsymbol{O}; \boldsymbol{\lambda})$	posterior probability of θ given the observation sequence $\boldsymbol{O}$

Finite-state Transducers

$\boldsymbol{A}$, $\boldsymbol{L}$	finite-state acceptors
$\boldsymbol{U}$	finite-state transducer
$\boldsymbol{U}^{-1}$	transducer inverse
$\boldsymbol{U} \circ \boldsymbol{V}$	composition of transducers $\boldsymbol{U}$ and $\boldsymbol{V}$
$\boldsymbol{U} \otimes \boldsymbol{V}$	concatenation of transducers $\boldsymbol{U}$ and $\boldsymbol{V}$
$[\![\boldsymbol{U}]\!]$	transducer shortest-distance (lattice weight)

Base Model Notation

$\boldsymbol{O}$	sequence of observation vectors
$\boldsymbol{o}_t$	t-th observation vector in $\boldsymbol{O}$
$\boldsymbol{\theta}$	latent state sequence associated with an observation sequence $\boldsymbol{O}$
θ_t	t-th latent state in a sequence $\boldsymbol{\theta}$
$\theta_t = \{j, m\}$	observation $\boldsymbol{o}_t$ is associated with a latent state j and mixture-component m
θ_t^j, θ_t^{jm}	shorthand for $\theta_t = j$ and $\theta_t = \{j, m\}$
d	dimensionality of observation vectors
N	number of HMM latent states
M	number of GMM mixture-components
a_{ij}	probability of transitioning from an HMM state i to a new state j
c_{jm}	GMM mixture-component prior associated with state j and mixture-component m
$\boldsymbol{\mu}_{jm}$	GMM mean vector associated with state j and mixture-component m
$\boldsymbol{\Sigma}_{jm}$	GMM covariance matrix associated with state j and mixture-component m
$\boldsymbol{\lambda}$	Base model parameters
$\hat{p}(\boldsymbol{O}; \boldsymbol{\lambda})$	base statistical model with parameters $\boldsymbol{\lambda}$
$\boldsymbol{\Theta}$	set of all possible state sequences
$\mathcal{F}^{\texttt{ml}}$	Maximum likelihood objective function
$\mathcal{F}^{\texttt{mmi}}$	Maximum mutual information objective function
$\mathcal{F}^{\texttt{mpe}}$	Minimum phone error objective function

Augmented Model Notation

$\boldsymbol{\alpha}$	augmented parameters
$\boldsymbol{\alpha}_{\boldsymbol{\mu}_{jm}}$	augmented parameters associated with base model derivatives with respect to $\boldsymbol{\mu}_{jm}$
$p(\boldsymbol{O}; \boldsymbol{\lambda}, \boldsymbol{\alpha})$	augmented model with base model parameters $\boldsymbol{\lambda}$ and augmented parameters $\boldsymbol{\alpha}$
$\boldsymbol{\alpha}_j(t)$	forward vector associated with HMM state j at time t
$\boldsymbol{\beta}_j(t)$	backward vector associated with HMM state j at time t

Maximum Margin Estimation

n	number of training examples
$\mathcal{F}^{\mathtt{svm}}$	SVM objective function
$\boldsymbol{\alpha}^{\mathtt{svm}}$	SVM Lagrange multipliers
$\boldsymbol{\phi}^{\mathtt{LLR}}(\boldsymbol{O};\boldsymbol{\lambda})$	Log-likelihood-ratio score-space
$\boldsymbol{\phi}^{\mathtt{F}}(\boldsymbol{O};\boldsymbol{\lambda})$	Fisher score-space
$\boldsymbol{\phi}^{\mathtt{LL}}(\boldsymbol{O};\boldsymbol{\lambda})$	Generative/augmented model score-space
$\boldsymbol{\phi}^{\mathtt{CN}}(\boldsymbol{O};\boldsymbol{\lambda})$	Augmented model score-space with confusion network posteriors appended

Contents

Table of Contents viii

List of Figures xi

List of Tables xiii

1 Introduction 1

1.1 Organisation of thesis . . . 3

2 Statistical Classification 5

2.1 Generative models . . . 5

2.1.1 The exponential family . . . 6

2.1.2 Latent-variable models . . . 8

2.1.3 Parameter estimation for generative models . . . 15

2.1.4 Speech recognition . . . 17

2.2 Discriminative models . . . 22

2.2.1 Conditional random fields (CRFs) . . . 23

2.2.2 Hidden conditional random fields (HCRFs) . . . 24

2.2.3 Parameter estimation for conditional models . . . 25

2.2.4 Support vector machines . . . 26

2.3 Summary . . . 30

3 Dynamic Kernels 31

3.1 Discrete-observation dynamic kernels . . . 32

3.1.1 Bag-of-word kernels . . . 32

3.1.2 String kernels . . . 33

3.1.3 Marginalised count kernels . . . 34

3.2 Continuous-observation dynamic kernels . . . 35

3.2.1 Fisher kernels . . . 35

3.2.2 Generative kernels . . . 37

3.2.3 Sequence kernels . . . 37

3.3 Summary . . . 38

4 Augmented Statistical Models **40**
4.1 Local exponential expansion . . . 41
4.1.1 Taylor series expansion . . . 41
4.1.2 Manifolds and fibre-bundles . . . 44
4.2 Dependency modelling in augmented models . . . 45
4.2.1 Augmented exponential models . . . 47
4.2.2 Augmented latent-variable models . . . 48
4.2.3 Illustrative example: first-order A-GMM . . . 51
4.3 Augmented HMMs (A-HMMs) . . . 53
4.3.1 First-order statistics . . . 54
4.3.2 Second-order statistics . . . 55
4.3.3 Illustrative example: second-order A-HMM . . . 57
4.4 Kernelised augmented models . . . 58
4.5 Relationships with dynamic kernels . . . 60
4.5.1 Fisher kernels . . . 60
4.5.2 Marginalised count kernels . . . 61
4.5.3 Sequence kernels . . . 62
4.6 Summary . . . 62

5 Continuous Rational Kernels **64**
5.1 Weighted finite-state transducers . . . 65
5.1.1 Transducers . . . 65
5.1.2 Acceptors . . . 67
5.1.3 Operators . . . 67
5.2 Rational kernels . . . 68
5.3 Continuous rational kernels . . . 70
5.3.1 First-order GMM and HMM derivatives . . . 72
5.3.2 Second and higher-order derivatives . . . 77
5.3.3 Generative score-spaces and kernels . . . 79
5.4 Summary . . . 80

6 Training Generative Augmented Models **81**
6.1 Direct estimation of parameters . . . 82
6.2 Distance-based learning . . . 82
6.2.1 Estimating augmented parameters, $\boldsymbol{\alpha}$ . . . 84
6.2.2 Estimating base model parameters, $\boldsymbol{\lambda}$ . . . 87
6.3 Metrics . . . 93
6.4 Acoustic code-breaking . . . 95
6.5 Summary . . . 98

7 Conditional Augmented Models **99**
7.1 Conditional augmented models . . . 100
7.2 Parameter estimation . . . 102
7.3 Inference . . . 104
7.4 Speech recognition . . . 105
7.4.1 Sentence modelling . . . 105
7.4.2 Lattice-based C-Aug sentence models . . . 108
7.4.3 Conditional maximum likelihood estimation . . . 110
7.4.4 Minimum phone error estimation . . . 113
7.4.5 Recognition . . . 115
7.5 Language modelling . . . 115
7.6 Comparison with CRFs and HCRFs . . . 116
7.7 Summary . . . 118

8 Experimental Results **119**
8.1 Cross-validation experiments . . . 119
8.1.1 First-order augmented models . . . 120
8.1.2 Higher-order augmented models . . . 123
8.1.3 Varying the score-space dimensionality . . . 124
8.1.4 Kernelised augmented models . . . 125
8.1.5 MMI and MM base models . . . 126
8.2 LVCSR rescoring . . . 127
8.2.1 Development set: eval03 . . . 128
8.2.2 Evaluation sets: eval02 and eval04 . . . 130
8.2.3 Generative versus conditional augmented models . . . 132
8.2.4 Summary . . . 134
8.3 TIMIT . . . 134
8.3.1 Rescoring confusable phone pairs . . . 136
8.3.2 C-Aug classification . . . 138
8.3.3 C-Aug recognition . . . 140
8.4 Summary . . . 146

9 Conclusions and Future Work **148**
9.1 Augmented statistical models . . . 149
9.2 Continuous rational kernels . . . 150
9.3 Conditional augmented models . . . 151
9.4 Future work . . . 152

A HMM Derivatives **154**
A.1 Transition probability derivatives 154
A.2 Output-distribution derivatives 155
A.2.1 First-order derivatives 156
A.2.2 Second-order derivatives 156
A.2.3 Derivatives for GMM output distributions 157

B HMM Forward-Backward and Viterbi Algorithms **159**
B.1 The Forward-Backward algorithm 159
B.2 The Viterbi algorithm 160
B.3 The Double-Forward-Backward algorithm 161

C Variable-Margin Support Vector Machines **163**

D Code-Breaking Pairs **166**
D.1 LVCSR word pairs 166
D.2 TIMIT phone pairs 167

References **169**

List of Figures

2.1 A three-state left-to-right hidden Markov model 11
2.2 A typical speech recognition system 18
2.3 A typical monophone speech recognition system 19
2.4 The optimal SVM hyperplane for linearly separable data 27

4.1 Manifold representation of an augmented model 44
4.2 Modelling a 'symmetric' log-normal distribution 52

5.1 A finite-state transducer 65
5.2 An acceptor for the input sequence $\{a, a, b, a\}$ 67
5.3 An example trellis for a three-state left-to-right HMM 71

6.1 Explicit maximum-margin estimation of the base model parameters 88
6.2 Variation of the SVM dual objective function during maximum-margin estimation of the base models 89
6.3 Maximum margin estimation of augmented and base model parameters . . 92
6.4 Augmented model metrics 93
6.5 LVCSR conversion from multi-class to binary classification 96

8.1 Augmented model performance versus number of sufficient statistics for the pair CAN/CAN'T 125
8.2 Effects of β on performance of the `eval03` development set 128
8.3 Rescoring confusions in the `eval02` and `eval04` evaluation sets 130
8.4 Proportion of evaluation sets rescored as number of augmented models increases 131
8.5 Augmented model rescoring of the TIMIT phone classification task 137
8.6 Proportion of development and core test sets rescored as the number of augmented models is increased 138
8.7 CML estimation of C-Aug model augmented parameters 139
8.8 CML C-Aug models for TIMIT phone recognition 142
8.9 Analysing the poor performance of CML for C-Aug recognition 143
8.10 MPE training of C-Aug sentence models 145

B.1 A pictorial representation of the Forward-Backward algorithm 159
B.2 A pictorial representation of the Double-Forward-Backward algorithm . . . 161
B.3 Double-Forward-Backward algorithm for calculating HMM joint-posterior probabilities 162

List of Tables

2.1 A selection of popular kernels 29

4.1 Modelling symmetric log-normal data using GMMs and A-GMMs 52
4.2 Selected first- and second-order HMM mixture-component-prior derivatives 57

5.1 Example transducer transformations 66
5.2 Popular semirings for transducers 66
5.3 The vector semiring 76

8.1 Summary of HMM score-spaces 120
8.2 Cross validation results for HMM base models 121
8.3 Varying the number of mixture-components for CAN/CAN'T 122
8.4 Kernelised augmented models for the pair CAN/CAN'T 126
8.5 Baseline LVCSR results for the `eval02`, `eval03` and `eval04` data sets . . . 128
8.6 Augmented model rescoring of `eval03` development set 129
8.7 Comparison of MM augmented models and CML C-Aug models for the confusion pair CAN/CAN'T 132
8.8 Comparing optimisation algorithms for CML estimation of C-Aug models . 133
8.9 Confusable phone-pairs in the TIMIT classification core test set 136
8.10 Classification error on the TIMIT core test set 139
8.11 Baseline HMM recognition performance on TIMIT 141

D.1 LVCSR rescoring pairs for `eval02`, `eval03` and `eval04`. 166
D.2 TIMIT rescoring pairs for ML baseline. 167
D.3 TIMIT rescoring pairs for MMI baseline. 168

1

Introduction

Many important applications, ranging from text classification [22, 54] and speech recognition [6, 52, 100] to bioinformatics [47, 50, 61] and intrusion detection [76, 113], require the classification of variable-length sequences of observations. Although many different approaches for this classification have been proposed—including rule-based methods and syntactic approaches—statistical techniques have been found to yield the best performance in many cases [51]. Statistical classifiers, the focus of this thesis, use sets of training examples to estimate models that capture the relationships between class labels and observation sequences, taking into account the uncertainties that may arise from differences between examples. Class decision boundaries are then implied from these models.

Models for statistical classification are commonly categorised into two broad groups: generative models and discriminative models. As the name suggests, generative models approximate the probability density function associated with the observation sequences, allowing sequence likelihoods to be calculated. Using a combination of Bayes' rule and Bayes' decision rule, these likelihoods can then be used to calculate a set of class decision boundaries. Generative models are typically defined using a number of independence and conditional-independence assumptions that restrict the dependencies modelled. These assumptions enable model complexity to be controlled and allow robust parameter estimates to be obtained even when training data is limited. The second type of statistical model discussed in this thesis are discriminative models. Defined using a set of sufficient statistics (or features) extracted from the observation sequences, these directly model the decision boundaries between classes. Discriminative models therefore avoid the need to maintain valid generative and prior distributions. Furthermore, since the normalisation terms of

discriminative models are typically calculated as a summation over the set of class labels (instead of an integration over all possible observation sequences), discriminative models require few of the constraints normally present in generative models. This makes discriminative models extremely flexible, and enables them to represent a wide range of complex dependencies and distributions.

One of the difficulties often encountered when generative and discriminative models are applied to practical applications is the selection of appropriate modelling assumptions. These assumptions determine the range and types of dependencies that can be represented and are often specified using a combination of independence assumptions, conditional-independence assumptions and sufficient statistics. If too many assumptions are made, models may be a poor approximation to the underlying process, adversely affecting classification performance. Conversely, with too few assumptions, models may be too flexible, making robust parameter estimation difficult given the limited training data available. When modelling static data (single observations), empirical approaches are often used to identify which features to model. For sequence data, however, empirical feature selection approaches can be difficult to implement since the range of potential sequence features/dependencies is vast. Instead, a small number of standard models—such as hidden Markov models (HMMs)—are typically used. Unfortunately, for many applications, the independence and conditional-independence assumptions of these models are incorrect, potentially reducing classification accuracy. In recent years, researchers have therefore concentrated on developing statistical models that can represent more complex dependencies.

Many different approaches have been proposed for extending the range of dependencies modelled by both generative and discriminative statistical models. For generative models, popular approaches include: segmental models [29, 91], factor analysed HMMs [106, 107], switching linear dynamical systems [107], buried Markov models [8, 9] and mixed memory models [88, 109]. For discriminative models, latent-variable extensions such as hidden conditional random fields (HCRFs) [42,99], and kernel-based approaches such as Fisher [49,50] and generative kernels [117, 118], have been proposed. Unfortunately, although these techniques all allow additional dependencies to be modelled, many require that the dependencies are selected a priori using a combination of expert knowledge and empirical evidence. For sequence data, this can be difficult since sequences often contain a huge number of different dependencies. To avoid such problems, this thesis examines a number of systematic approaches for defining statistical models that include additional dependencies.

In this thesis, three approaches are presented for modelling complex dependencies in sequences of observations. The first, based upon work in [117], are augmented statistical models. These extend standard generative models—the base models—using a range of additional sufficient statistics that are defined by a local exponential approximation to

the base model. The additional statistics allow augmented models to represent a wide range of temporal and spatial dependencies between observations that were not present in the original, base, model. A novel variable-margin, maximum-margin training algorithm based upon support vector machines (SVMs) [19, 125] is proposed for estimating the augmented parameters associated with these sufficient statistics. Two algorithms for maximum-margin estimation of the base model parameters are also proposed.

The second contribution of this thesis are continuous rational kernels. These are a new form of dynamic kernel that combine latent-variable generative statistical models with weighted finite-state rational kernels to create highly flexible classifiers of continuous-observation sequences. By varying both the generative models and the rational kernels, continuous rational kernels allow a wide range of different kernel feature-spaces (and hence dependencies) to be modelled using only a small number of efficient and standardised algorithms. This work focuses upon continuous rational kernels as a method for calculating augmented model sufficient statistics. When augmented models are estimated using distance-based training algorithms, continuous rational kernels allow models to be trained without the use of derivative-specific dynamic programming algorithms. Instead, augmented model calculations are performed using a small number of simple yet efficient transducer operations.

The final contribution of this thesis are conditional augmented (C-Aug) models. These use the same sufficient statistics as generative augmented models but are defined within a discriminative framework. The use of augmented model sufficient statistics allows C-Aug models to share many of the benefits of augmented models, including being able to model a wide range of temporal and spatial dependencies. Additionally, by directly modelling the posterior probabilities of the correct class labels, C-Aug models avoid many of the normalisation difficulties faced by standard augmented models. This simplifies algorithms for both training and inference, and enables C-Aug models to be applied to a wide range of tasks. In this work, C-Aug model parameters are estimated using the discriminative conditional maximum likelihood (CML) criterion [63]. A lattice-based framework is proposed that allows C-Aug models to be applied to a simple continuous speech recognition task.

1.1 Organisation of thesis

This thesis is split into three parts. The first, consisting of chapters 2 and 3, introduces the reader to a range of standard statistical techniques for classifying sequence data. The second, in chapters 4–7, contains the main contributions of this thesis, namely: derivations, analyses, and implementation of generative augmented models, continuous rational kernels, and conditional augmented models. Chapters 8 and 9 then present detailed ex-

perimental results and offer a number of conclusions and suggestions for future work.

A more detailed chapter-by-chapter breakdown is given below.

Chapter 2 provides a brief introduction to statistical classification of sequence data. A range of generative and discriminative techniques are discussed, with examples given of each. Many common generative and discriminative training criteria are considered.

Chapter 3 introduces dynamic kernels as an alternative approach for sequence classification. Dynamic kernels map sequences of observations into high-dimensional (often implicit) feature-spaces, where kernel-based classifiers—such as support vector machines—can be used to determine the class decision boundaries.

Chapter 4 then introduces generative augmented models. After the initial derivations, this chapter focuses upon the additional temporal and spatial dependencies that can be represented within the augmented model framework. Detailed examples of augmented models are presented, along with dynamic programming algorithms for calculating augmented model sufficient statistics. Comparisons with a number of standard dynamic kernels are given.

Chapter 5 proposes continuous rational kernels as a powerful form of dynamic kernel that uses both latent-variable generative models and weighted finite-state transducers to calculate complex kernel feature-spaces for sequences of continuous observations. Calculation of augmented model sufficient statistics within this framework is discussed.

Chapter 6 combines ideas from the previous two chapters and describes a distance-based framework for estimating augmented model parameters. This allows model parameters to be estimated by positioning a linear decision boundary in a score-space. A variable-margin SVM is proposed for maximum-margin estimation of both the base model and the augmented model parameters.

Chapter 7 introduces conditional augmented (C-Aug) models. These are defined using the same complex sufficient statistics as generative augmented models, but within a discriminative framework. C-Aug models overcome many of the normalisation difficulties of augmented models, allowing them to be applied to a wider range of tasks. Training and inference are discussed, as are extensions for sentence-based speech recognition.

Chapter 8 discusses two sets of experiments. The first uses augmented models and C-Aug models to rescore confusable word-pairs extracted from a large vocabulary speech recognition task. The second, based upon the TIMIT database, evaluates augmented and C-Aug models using a range of phone classification and recognition experiments.

Chapter 9 concludes with a summary of the thesis and suggestions for future topics of research.

2

Statistical Classification

Statistical classification techniques are often categorised into two broad groups: generative models and discriminative models. This chapter considers both types. Generative models, discussed first, model the likelihood of observations given the class labels. Given a prior distribution over the class labels, decision boundaries are calculated using a combination of Bayes' rule and Bayes' decision rule. Although model parameters are often estimated using the maximum likelihood training criterion, discriminative training criteria—for example maximum mutual information (MMI)—have become increasingly popular in recent years.

A second, increasingly popular, group of models for statistical classification are discriminative models, such as conditional random fields (CRFs) and support vector machines (SVMs). These directly model the class decision boundaries avoiding the need to maintain valid generative and prior distributions. Parameter estimation for discriminative models is inherently discriminatory and is typically performed using training criteria such as conditional maximum likelihood and maximum margin estimation.

Both generative and discriminative models can be applied to either static (single observations) or dynamic (sequences of observations) data. Discussion in this chapter focuses upon the classification of sequence data.

2.1 Generative models

Generative models are one of the most popular forms of statistical model for classifying sequence data. As their name suggests, these model the probability density function associated with the observations, enabling observation sequences, $\boldsymbol{O} = \{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_T\}$, to be

randomly generated from distributions of the model. Alternatively, given an observation sequence, generative models allow the sequence likelihood to be calculated. This second use is the basis of many generative model classification techniques, and is the main focus of this section.

First, consider a generative model, $p(\boldsymbol{O}|\omega;\boldsymbol{\lambda})$, with observations $\boldsymbol{O}$, class label ω, and model parameters $\boldsymbol{\lambda}$. Given a prior distribution over the class labels, $P(\omega)$, Bayes' rule can be used to convert the generative model likelihood into a class posterior [30, 121],

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

where $p(\boldsymbol{O})$ is known as the evidence and is typically calculated as $\sum_{\omega\in\Omega} p(\boldsymbol{O}|\omega;\boldsymbol{\lambda})P(\omega)$, and Ω is the set of all class labels. Throughout this work, parameters of the prior distribution are assumed to be known and fixed a priori; they are therefore omitted from equations to simplify notation.

Once observation likelihoods have been converted into class posterior probabilities, decision boundaries can be calculated using Bayes' decision rule. This assigns each observation sequence a class label corresponding to the class with the highest posterior probability given that sequence [30]. For generative models, this can be written as,

$$\begin{aligned} y &= \arg\max_{\omega} P(\omega|\boldsymbol{O};\boldsymbol{\lambda}) \\ &= \arg\max_{\omega} \left(\frac{p(\boldsymbol{O}|\omega;\boldsymbol{\lambda})P(\omega)}{p(\boldsymbol{O})} \right) \end{aligned} \tag{2.2}$$

where y is the class label assigned to a sequence $\boldsymbol{O}$. When the correct generative and prior distributions are known, the class labels generated by equation (2.2) minimise the expected generalisation error. In practice, however, the true distributions are often not known and, instead, parametric approximations are typically used.[1] Since these approximations are normally incorrect, decision boundaries calculated using equation (2.2) may yield sub-optimal classification performance. However, with carefully chosen models and robust parameter estimation schemes, good classification performance can often be obtained [30].

2.1.1 The exponential family

Many generative statistical models are based upon members of the linear exponential family [131]. Popular examples are the Gaussian, Poisson and Bernoulli distributions. For an observation sequence, $\boldsymbol{O} = \{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_T\}$, the general form of a member of the exponential family (an exponential model) is given by,

$$p(\boldsymbol{O};\boldsymbol{\lambda}) = \frac{1}{\tau(\boldsymbol{\lambda})} h(\boldsymbol{O}) \exp\left(\left\langle \boldsymbol{\lambda}, \boldsymbol{T}(\boldsymbol{O}) \right\rangle \right) \tag{2.3}$$

[1] Non-parametric generative models can also be defined. These make few assumptions about the functional form of the probability distribution function and instead attempt to imply structures directly from the data. Non-parametric techniques are not considered further in this thesis.

where $\boldsymbol{\lambda}$ are the natural parameters, $h(\boldsymbol{O})$ is a reference distribution, $\boldsymbol{T}(\boldsymbol{O})$ are fixed-dimensional sufficient statistics extracted from the observation sequence, and $\langle \cdot, \cdot \rangle$ denotes an inner-product between two vectors. To simplify notation, the class-label dependence, ω, is omitted from the model likelihood, reference distribution and sufficient statistics. The normalisation term, $\tau(\boldsymbol{\lambda})$, is calculated as the expectation of the unnormalised distribution over all possible observation sequences,

$$\tau(\boldsymbol{\lambda}) = \int h(\boldsymbol{O}) \exp\Big(\big\langle \boldsymbol{\lambda}, \boldsymbol{T}(\boldsymbol{O}) \big\rangle\Big) d\boldsymbol{O} \tag{2.4}$$

where $\int \mathrm{d}\boldsymbol{O}$ denotes an integration over observation sequences. To ensure that the model in equation (2.3) is valid, the normalisation term must be real and bounded. This is typically achieved using a combination of carefully selected sufficient statistics and parameter constraints.[2]

One of the most important aspects of exponential models are the sufficient statistics. These control the temporal and spatial information extracted from observation sequences, and define the types of 'dependency' that can and cannot be modelled. The selection of sufficient statistics is an application-dependent problem that is typically approached using a combination of expert knowledge and empirical evidence. Unfortunately, for sequence data, this choice is especially difficult since the range of available statistics grows exponentially with sequence length. Therefore, to simplify the selection process, independence assumptions are often introduced. These reduce the number of potential statistics by eliminating statistics that measure the correlation between quantities that are assumed to be independent. A simple, though popular, example of an exponential model independence assumption is that observations within a sequence are independent and identically distributed (i.i.d.) according to the distribution, $p(\boldsymbol{o}; \boldsymbol{\lambda})$, with parameters $\boldsymbol{\lambda}$ and sufficient statistics $\boldsymbol{T}(\boldsymbol{o})$. Expressing the sequence likelihood as the product of independent observation likelihoods, $p(\boldsymbol{O}; \boldsymbol{\lambda})$ can be written as,

$$\begin{aligned} p(\boldsymbol{O}; \boldsymbol{\lambda}) &= \prod_{t=1}^{T} \frac{1}{\tau(\boldsymbol{\lambda})} h(\boldsymbol{o}_t) \exp\Big(\big\langle \boldsymbol{\lambda}, \boldsymbol{T}(\boldsymbol{o}_t) \big\rangle\Big) \\ &= \frac{1}{\tau(\boldsymbol{\lambda})^T} \Bigg(\prod_{t=1}^{T} h(\boldsymbol{o}_t) \Bigg) \exp\Bigg(\sum_{t=1}^{T} \big\langle \boldsymbol{\lambda}, \boldsymbol{T}(\boldsymbol{o}_t) \big\rangle \Bigg) \end{aligned} \tag{2.5}$$

where T is the number of observations in the sequence. From equation (2.5) it is clear that the exponential sequence model $p(\boldsymbol{O}; \boldsymbol{\lambda})$ has a reference distribution $h(\boldsymbol{O}) = \prod_{t=1}^{T} h(\boldsymbol{o}_t)$,

[2] Although there are relatively few restrictions on the types of sufficient statistics that are used for exponential models, some choices may result in unbounded normalisation terms for $\boldsymbol{\lambda} \neq \boldsymbol{0}$. A simple example of this is the vector of sufficient statistics $\boldsymbol{T}(\boldsymbol{o}) = \boldsymbol{o}$. As $\boldsymbol{o}$ varies, the exponential argument, $\langle \boldsymbol{\lambda}, \boldsymbol{T}(\boldsymbol{o}) \rangle$, can become infinitely positive or negative. When the argument is negative, the exponential term in equation (2.4) tends towards a lower bound of zero. When the argument is positive, however, the term may become unbounded, potentially resulting in an infinitely large normalisation term.

natural parameters $\boldsymbol{\lambda}$, and sufficient statistics $\boldsymbol{T}(\boldsymbol{O})$,

$$\boldsymbol{T}(\boldsymbol{O}) = \sum_{t=1}^{T} \boldsymbol{T}(\boldsymbol{o}_t) \tag{2.6}$$

where $\boldsymbol{T}(\boldsymbol{o}_t)$ are sufficient statistics for the distribution of individual observations, $p(\boldsymbol{o}_t; \boldsymbol{\lambda})$.

One of the most popular exponential distributions for $p(\boldsymbol{o}_t; \boldsymbol{\lambda})$ is the Gaussian distribution. This has easy-to-calculate sufficient statistics and a closed-form normalisation term that allows likelihood calculations and parameter estimation to be performed with minimal computational expense. For an observation $\boldsymbol{o}$, the Gaussian likelihood can be written as,

$$\mathcal{N}(\boldsymbol{o}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\tfrac{1}{2}(\boldsymbol{o}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{o}-\boldsymbol{\mu})\right) \tag{2.7}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are the distribution mean and variance respectively. Rewriting equation (2.7) in the canonical form of equation (2.3), allows the Gaussian sufficient statistics, $\boldsymbol{T}(\boldsymbol{o})$, and natural parameters, $\boldsymbol{\lambda} = \{\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2\}$, to be calculated,

$$\boldsymbol{T}(\boldsymbol{o}) = \begin{bmatrix} \boldsymbol{o} \\ \mathrm{vec}(\boldsymbol{o}\boldsymbol{o}^{\mathrm{T}}) \end{bmatrix} ; \qquad \boldsymbol{\lambda}_1 = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} ; \qquad \boldsymbol{\lambda}_2 = -\tfrac{1}{2}\mathrm{vec}(\boldsymbol{\Sigma}^{-1}) \tag{2.8}$$

Given these parameters and statistics, equations (2.5) and (2.6) can be used to define a simple model for sequences of independent Gaussian-distributed observations. Sufficient statistics for this model are given by,

$$\boldsymbol{T}(\boldsymbol{O}) = \sum_{t=1}^{T} \begin{bmatrix} \boldsymbol{o}_t \\ \mathrm{vec}(\boldsymbol{o}_t\boldsymbol{o}_t^{\mathrm{T}}) \end{bmatrix} \tag{2.9}$$

Unfortunately, for many 'real-world' tasks, the assumption that observations within a sequence are independent and identically distributed is unrealistic. Instead, it is often necessary for distributions to capture complex temporal or spatial dependencies within the sequence. Although it is possible to represent these dependencies using exponential models, a large number of sufficient statistics and model parameters are often required, making robust parameter estimation difficult. A popular alternative is to introduce latent variables and conditional-independence assumptions.

2.1.2 Latent-variable models

Latent-variable models are a popular approach for defining statistical models that incorporate complex spatial or temporal dependencies. Defined using a set of latent variables (representing some hidden state of the model), these allow observation sequence likelihoods to be written in terms of conditional distributions of the observations given the hidden states. When conditional-independence assumptions are introduced, this decomposition

often allows latent-variable models to offer a good compromise between model complexity and the ability to represent complex distributions.

For an observation sequence, $\boldsymbol{O} = \{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_T\}$, let $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_T\}$ be the sequence of latent-variables (hidden states) associated with the observations. Observation sequence likelihoods for latent-variable models are typically defined by marginalising the joint distribution of observation and latent-state sequences over all possible state sequences. In practice, however, the joint distribution of the states and observations is often decomposed into a prior distribution of the latent-states, $P(\boldsymbol{\theta}; \boldsymbol{\lambda})$, and a conditional distribution of the observations given the latent-state sequence, $p(\boldsymbol{O}|\boldsymbol{\theta}; \boldsymbol{\lambda})$,

$$p(\boldsymbol{O}; \boldsymbol{\lambda}) = \sum_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} p(\boldsymbol{O}, \boldsymbol{\theta}; \boldsymbol{\lambda}) = \sum_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} P(\boldsymbol{\theta}; \boldsymbol{\lambda}) p(\boldsymbol{O}|\boldsymbol{\theta}; \boldsymbol{\lambda}) \tag{2.10}$$

where $\boldsymbol{\lambda}$ are the model parameters, and $\boldsymbol{\Theta}$ is the set of all possible state sequences.[3] In practice, the state-dependent output distribution, $p(\boldsymbol{O}|\boldsymbol{\theta}; \boldsymbol{\lambda})$, often cannot be calculated directly. Instead, it is usually decomposed into a number of conditional distributions that relate the likelihood of an observation to the previous observations in the sequence,

$$\begin{aligned} p(\boldsymbol{O}|\boldsymbol{\theta}; \boldsymbol{\lambda}) = p(\boldsymbol{o}_T|\boldsymbol{o}_{T-1}, \boldsymbol{o}_{T-2}, \ldots, \boldsymbol{o}_1, \boldsymbol{\theta}; \boldsymbol{\lambda}) \times \\ p(\boldsymbol{o}_{T-1}|\boldsymbol{o}_{T-2}, \ldots, \boldsymbol{o}_1, \boldsymbol{\theta}; \boldsymbol{\lambda}) \times \\ \vdots \\ p(\boldsymbol{o}_1|\boldsymbol{\theta}; \boldsymbol{\lambda}) \end{aligned} \tag{2.11}$$

Unfortunately, even moderate-length observation sequences can result in a massive expansion of models and model parameters. To reduce the number of model parameters, the back-history $\{\boldsymbol{o}_{t-1}, \ldots, \boldsymbol{o}_1\}$ associated with each observation $\boldsymbol{o}_t$ is often truncated to include only the first l terms,

$$p(\boldsymbol{O}|\boldsymbol{\theta}; \boldsymbol{\lambda}) = \prod_{t=1}^{T} p(\boldsymbol{o}_t|\boldsymbol{o}_{t-1}, \ldots, \boldsymbol{o}_{t-l}, \boldsymbol{\theta}; \boldsymbol{\lambda}) \tag{2.12}$$

The simplified model in equation (2.12) typically requires fewer parameters than the full expansion in equation (2.11). Buried Markov models [8, 9] are one example where the current observation is assumed to depend on a small number of prior observations. More commonly, however, the back-history is truncated completely ($l = 0$), eliminating all

[3] In the general case, latent variables can be represented by a vector of continuous-valued elements. Decomposing the observation sequence likelihood into a conditional distribution with respect to the continuous latent variables therefore yields the expression,

$$p(\boldsymbol{O}; \boldsymbol{\lambda}) = \int P(\boldsymbol{\theta}; \boldsymbol{\lambda}) p(\boldsymbol{O}|\boldsymbol{\theta}; \boldsymbol{\lambda}) \mathrm{d}\boldsymbol{\theta}$$

An example of a continuous latent-variable model is the switching linear dynamical system (SLDS) [107]. See below for further details.

references to previous observations. This restricts the modelling of temporal dependencies to the latent-state prior, $P(\boldsymbol{\theta};\boldsymbol{\lambda})$, and allows the output distribution in equation (2.12) to be simplified,

$$p(\boldsymbol{O}|\boldsymbol{\theta};\boldsymbol{\lambda}) = \prod_{t=1}^{T} p(\boldsymbol{o}_t|\boldsymbol{\theta};\boldsymbol{\lambda}) \tag{2.13}$$

Although equation (2.13) is much simpler than the output distribution in equation (2.11), the dependence of observations on the complete state sequence can still make calculations difficult. To avoid these difficulties, many popular latent-variable models such as Gaussian mixture models (GMMs) and hidden Markov models (HMMs) introduce an additional assumption that the observations, $\boldsymbol{o}_t$, are dependent upon only the current state, θ_t. When this simplification is applied to equation (2.13), the latent-variable model likelihood in equation (2.10) can be rewritten as,

$$p(\boldsymbol{O};\boldsymbol{\lambda}) = \sum_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} P(\boldsymbol{\theta};\boldsymbol{\lambda}) \Big(\prod_{t=1}^{T} p(\boldsymbol{o}_t|\theta_t;\boldsymbol{\lambda}) \Big) \tag{2.14}$$

where $p(\boldsymbol{o}_t|\theta_t;\boldsymbol{\lambda})$ is a state-conditional output distribution. The ability of latent-variable models to break complex sequence distributions into a number of simpler distributions is important since it provides an effective method for separating different types of dependency. This separation makes it easier to control the complexity of different aspects of latent-variable models. For example, when modelling sequences with strong spatial dependencies but relatively weak temporal dependencies, a latent-variable model can be defined such that full-covariance Gaussian output distributions (with many parameters) are used to model the within-observation spatial dependencies, whereas relatively simple first-order Markov constraints (requiring relatively few parameters) are used when modelling temporal dependencies in the latent-state prior distribution.

In practice the choice of latent-variable prior and output distributions is often restricted by the need to sum over all possible latent-state sequences during likelihood calculations. This choice is especially important when long sequences are considered since the number of valid state-sequences grows exponentially with sequence length. To avoid these computational difficulties, state-prior and output distributions are typically selected from the small set of distributions that allow the summation to be simplified. Popular examples of distributions that allow this simplification are given in the sections below.

Gaussian mixture models

Mixture models are one of the simplest and most common forms of latent variable model. Constructed using a set of M mixture-components and a vector of model parameters $\boldsymbol{\lambda}$, mixture models allow observations to be generated using a two-stage process. First a mixture-component, m, is randomly selected according the prior distribution $P(m) = c_m$.

Then, given m, a single observation is sampled from a mixture-component-dependent output distribution, $p(\boldsymbol{o}|m;\boldsymbol{\lambda})$. This output distribution is often a member of the exponential family. The generation process repeats until the required number of observations have been generated.

The mixture-model likelihood of a single observation, $\boldsymbol{o}$, is calculated as a superposition of contributions from the mixture-component output distributions. For an M-mixture-component model, this allows the observation likelihood, $p(\boldsymbol{o};\boldsymbol{\lambda})$, to be written as,

$$p(\boldsymbol{o};\boldsymbol{\lambda}) = \sum_{m=1}^{M} c_m p(\boldsymbol{o}|m;\boldsymbol{\lambda}) \tag{2.15}$$

where $\boldsymbol{\lambda}$ are model parameters, and $c_m \in \boldsymbol{\lambda}$ are the mixture-component priors (or weights).

When observations within a sequence, $\boldsymbol{O} = \{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_T\}$, are assumed to be independent and identically distributed according to an M-mixture-component distribution, the likelihood, $p(\boldsymbol{O};\boldsymbol{\lambda})$, of a sequence $\boldsymbol{O}$ can be written as a product of the individual observation likelihoods,

$$p(\boldsymbol{O};\boldsymbol{\lambda}) = \prod_{t=1}^{T} p(\boldsymbol{o}_t;\boldsymbol{\lambda}) = \prod_{t=1}^{T} \sum_{m=1}^{M} c_m p(\boldsymbol{o}_t|m;\boldsymbol{\lambda}) \tag{2.16}$$

Finally, it is worth noting that although this section has presented mixture models in terms of the general mixture-component output distributions, $p(\boldsymbol{o}_t|m;\boldsymbol{\lambda})$, Gaussian distributions with mean $\boldsymbol{\mu}_m$ and variance $\boldsymbol{\Sigma}_m$ are used in many cases. The resulting models are known as Gaussian mixture models (GMMs).

Hidden Markov models

Hidden Markov models (HMMs) are a popular form of latent-variable model that allows simple temporal dependencies to be modelled according to a first-order Markov assumption. This allows observation sequences to incorporate simple continuity constraints, making HMMs popular for applications such as protein structure modelling in computational biology [50, 61] and acoustic modelling in speech recognition [6, 52, 100].

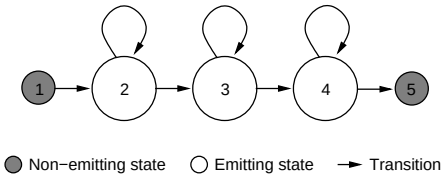


Figure 2.1: A three-state left-to-right hidden Markov model

Consider the example HMM in figure 2.1. This has two non-emitting states (states 1 and 5) and three emitting states (states 2–4). The generative process starts in the first non-emitting state, θ_0, at time $t = 0$. For times $t \in [1, T]$, the HMM then transitions

to a new state according to the state transition distribution, $P(\theta_t|\theta_{t-1};\boldsymbol{\lambda})$, where θ_t and θ_{t-1} denote the new and previous states respectively. These transitions follow a first-order Markov assumption and can be compactly represented by the transition probability matrix, $\boldsymbol{a}$, with elements, $a_{ij}=P(\theta_t=j|\theta_{t-1}=i)$.[4] HMM states are therefore conditionally independent given the previous state. Upon entry to a state $\theta_t=j$, a single observation, $\boldsymbol{o}_t$, is generated according to the state-conditional output distribution, $b_j(\boldsymbol{o}_t) = p(\boldsymbol{o}_t|\theta_t;\boldsymbol{\lambda})$, where $\boldsymbol{\lambda}$ is the set of all HMM parameters. Observations are conditionally independent given the current state.

When HMMs are used to classify observation sequences, $\boldsymbol{O} = \{\boldsymbol{o}_1,\ldots,\boldsymbol{o}_T\}$, the latent state sequence, $\boldsymbol{\theta} = \{\theta_1,\ldots,\theta_T\}$, is normally not known. It must therefore be marginalised out HMM likelihood calculations by summing over all possible state sequences. The HMM likelihood for an observation sequence $\boldsymbol{O}$, is thus given by,

$$p(\boldsymbol{O};\boldsymbol{\lambda}) = \sum_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \left(\prod_{t=1}^{T} P(\theta_t|\theta_{t-1};\boldsymbol{\lambda})\right)\left(\prod_{t=1}^{T} p(\boldsymbol{o}_t|\theta_t;\boldsymbol{\lambda})\right) \tag{2.17}$$

$$= \sum_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \prod_{t=1}^{T} a_{\theta_{t-1}\theta_t} p(\boldsymbol{o}_t|\theta_t;\boldsymbol{\lambda}) \tag{2.18}$$

where $\boldsymbol{\Theta}$ is the set of all possible state sequences, and $a_{\theta_0\theta_1}$ denotes the transition from the first non-emitting state θ_0 to an emitting state θ_1. It is worth noting that the first and second terms of equation (2.17) correspond directly to the first and second terms of the general latent variable model in equation (2.14). Since many data distributions have both temporal and spatial dependencies, GMM state-conditional output distributions, $b_j(\boldsymbol{o}_t) = \sum_{m=1}^{M} c_{jm}\mathcal{N}(\boldsymbol{o}_t;\boldsymbol{\mu}_{jm},\boldsymbol{\Sigma}_{jm})$, are often used. The resulting HMM likelihoods are given by,

$$p(\boldsymbol{O};\boldsymbol{\lambda}) = \sum_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \prod_{t=1}^{T} \left(a_{\theta_{t-1}\theta_t} \sum_{m=1}^{M} c_{\theta_t m}\mathcal{N}(\boldsymbol{o}_t;\boldsymbol{\mu}_{\theta_t m},\boldsymbol{\Sigma}_{\theta_t m})\right)$$

where $\theta_t=j$ is the current HMM state, and c_{jm}, $\boldsymbol{\mu}_{jm}$ and $\boldsymbol{\Sigma}_{jm}$ are state-conditional GMM mixture-component priors, means and variances. Defining a new latent-variable $\boldsymbol{\phi}$ that includes both the HMM latent-states, j, and the GMM mixture-components, m, the HMM likelihood can be rewritten as,

$$p(\boldsymbol{O};\boldsymbol{\lambda}) = \sum_{\boldsymbol{\phi}\in\boldsymbol{\Phi}} \prod_{t=1}^{T} a_{\phi_{t-1}\phi_t} c_{\phi_t}\mathcal{N}(\boldsymbol{o}_t;\boldsymbol{\mu}_{\phi_t},\boldsymbol{\Sigma}_{\phi_t}) \tag{2.19}$$

where $\mathcal{N}(\boldsymbol{o}_t;\boldsymbol{\mu}_{\phi_t},\boldsymbol{\Sigma}_{\phi_t})$ are the state/component Gaussian distributions, and $\phi_t = \{j,m\}$ is a latent-variable that specifies both the HMM state and the GMM mixture-component at time t.

[4]Different HMM topologies, for example left-to-right, are often implemented by placing constraints on the matrix of transition probabilities.

One of the main reasons for the popularity of HMMs within the machine-learning community is the Markov assumption that states are conditionally-independent given the previous state. This allows a number of efficient dynamic programming algorithms to be used for HMM parameter estimation and inference. The most common algorithms are the Forward-Backward algorithm [7,100] for HMM parameter estimation, and the Viterbi algorithm [32, 128] for inference. The computational complexity of both algorithms is linear with respect to both the observation sequence length and the number of HMM latent states. The Forward-Backward and Viterbi algorithms are discussed in more detail in appendix B.

The ability of HMMs to model temporal dependencies in sequences of observations, coupled with efficient algorithms for training and inference, means that HMMs are commonly used as acoustic models for speech recognition. Despite their popularity, however, HMMs are believed to be poor models of speech since the HMM conditional-independence assumptions severely limit the temporal dependencies that can be modelled. In recent years, many techniques have been proposed for overcoming these limitations. Two of the most popular approaches are mentioned here, with more detailed examples given below. The first is to introduce additional latent variables that allow more complex inter-state dependencies to be modelled. Recent examples, applied to speech recognition, include: factor analysed HMMs (FAHMMs) [106–108], switching linear dynamical systems (SLDSs) [107], and segmental models [29, 91, 92]. The second approach is to relax the assumption that observations are conditionally independent given the current state by introducing explicit dependencies between observations. Two recent examples of this approach are buried Markov models [8, 9] and mixed memory models [88, 109].

Switching linear dynamical systems

Switching linear dynamical systems (SLDSs) [107] are an extension of standard HMMs that utilise additional latent variables to relax the HMM conditional-independence assumptions. In addition to the HMM-style discrete latent variables, $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_T\}$, switching linear dynamical systems introduce a set of continuous latent variables, $\boldsymbol{x} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T\}$, that allows parameters of the state output distributions to vary with time. The first element of the continuous latent variable, $\boldsymbol{x}_1$, can be initialised in a number of ways. These range from re-initialising $\boldsymbol{x}$ at the start of every segment (phone or word boundaries), to initialising $\boldsymbol{x}$ once at the start of each utterance and propagating state information across segment boundaries (allowing longer-range dependencies to be modelled). The generative

model of an SLDS is given by,

$$\begin{aligned} \theta_t &\sim P(\theta_t|\theta_{t-1};\boldsymbol{\lambda}) \\ \boldsymbol{x}_t &= \boldsymbol{A}_{\theta_t}\boldsymbol{x}_{t-1} + \boldsymbol{w}_{\theta_t} \\ \boldsymbol{o}_t &= \boldsymbol{C}_{\theta_t}\boldsymbol{x}_t + \boldsymbol{v}_{\theta_t} \end{aligned} \tag{2.20}$$

where $\theta_t \sim P(\theta_t|\theta_{t-1};\boldsymbol{\lambda})$ denotes a first-order Markov process, $\boldsymbol{A}_{\theta_t}$ is a state-dependent transform matrix that controls the trajectory of the continuous latent variable $\boldsymbol{x}_t$, and $\boldsymbol{C}_{\theta_t}$ is a matrix that relates $\boldsymbol{x}_t$ to the observations. The terms, $\boldsymbol{w}_{\theta_t}$ and $\boldsymbol{v}_{\theta_t}$ are noise vectors—typically Gaussian distributed—associated with the continuous latent variables and the observation vectors respectively.

The continuous latent variable, $\boldsymbol{x}_t$, converts the piece-wise-constant HMM state-space to a more flexible piece-wise-linear state-space. This allows SLDSs to model observation sequences more accurately than would be possible with HMMs. Additionally, when $\boldsymbol{x}$ is allowed to propagate across segment boundaries (by only re-initialising at the beginning of utterances), SLDSs allow longer-range dependencies to be modelled. Unfortunately, the resulting dependence of observations on all previous states and observations can make calculation of latent-state posteriors (and hence parameter estimation) intractable in many cases. Model estimation and inference must therefore be performed using approximate methods, for example [104, 105].

When SLDSs with GMM noise vectors were applied to a standard speech recognition task, they were found to yield larger sequence likelihoods than comparable HMMs. This implies that SLDSs are a better model of speech than HMMs [104]. Unfortunately, when recognition performance was evaluated using a number of different speech recognition tasks [104], recognition performance was disappointing. SLDSs are therefore believed to be good models of speech, but poor classifiers.

Buried Markov models

Many attempts have been made at incorporating observational dependencies into latent-variable models such as HMMs [9, 39, 95, 132]. One of the most recent, applied to the task of speech recognition, are buried Markov models (BMM) [8, 9]. These relax the HMM assumption that observations are conditionally independent given the current state by introducing explicit dependencies between observations. The BMM likelihood for a sequence $\boldsymbol{O}$ is given by the extended HMM likelihood,

$$p(\boldsymbol{O};\boldsymbol{\lambda}) = \sum_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \prod_{t=1}^{T} a_{\theta_{t-1}\theta_t} p(\boldsymbol{o}_t|\boldsymbol{z}(\theta_t),\theta_t;\boldsymbol{\lambda}) \tag{2.21}$$

where $\boldsymbol{z}(\theta_t)$ is a state-conditional set of elements selected from previous observation vectors. An example of $\boldsymbol{z}(\theta_t)$ is the set of observation elements $\{o_{(t-1),4}, o_{(t-1),5}, o_{(t-2),10}, o_{(t-3),4}\}$

where $o_{(t),i}$ denotes the i-th dimension in the t-th observation. Note that only single dimensions of observations are selected to reduce the number of explicit dependencies. With different dependencies modelled in different states, BMMs concentrate their modelling power (parameters) on only the dependencies that are important for each state. This reduces the number of parameters required, and allows robust parameter estimates to be obtained without excessively large training sets.

BMM dependency selection is a complex process that is often approached in either an ad-hoc fashion (typically trial-and-error) or through the use of automated feature-selection methods, such as in [9]. One these dependencies have been determined, BMM parameters can be estimated in linear time using maximum likelihood estimation within the expectation maximisation (EM) framework [8, 9]. Alternatively, a number of discriminative algorithms—such as maximum mutual information—can be used. Inference is typically performed using a modified version of the Viterbi algorithm [8].

2.1.3 Parameter estimation for generative models

Many different algorithms have been proposed for estimating the parameters of generative models. Of these, maximum likelihood (ML) estimation [5, 100] is the most common. This updates model parameters in order to maximise the likelihood of the training data. Alternatively, generative parameters can be estimated using discriminative training algorithms. These typically update model parameters in order to minimise a differential approximation to the generalisation error [111]. Popular discriminative training criteria include: maximum mutual information (MMI) [5], minimum Bayes' risk (MBR) [57], and minimum classification error (MCE) [15,56]. Two additional discriminative training criteria that have been proposed for use in speech recognition systems are the minimum word error (MWE) [58, 85] and minimum phone error (MPE) [96, 97] criteria. In this thesis, discussion will focus upon ML, MMI and MPE estimation of generative models.

Maximum likelihood (ML)

Maximum likelihood (ML) estimation [7, 100] is a standard approach for estimating parameters of statistical models. Given a set of n independent and identically distributed training examples, $\mathcal{O} = \{\boldsymbol{O}_1, \ldots, \boldsymbol{O}_n\}$, with class labels, $\mathcal{Y} = \{y_1, \ldots, y_n\}$, maximum likelihood estimation adjusts model parameters in order to maximise the likelihood of the training set. For generative models of the form $p(\boldsymbol{O}|y;\boldsymbol{\lambda})$, where y is the class label, the ML objective function can be written as,

$$\mathcal{F}^{\texttt{ml}}(\boldsymbol{\lambda}) = p(\mathcal{O}|\mathcal{Y};\boldsymbol{\lambda}) = \prod_{i=1}^{n} p(\boldsymbol{O}_i|y_i;\boldsymbol{\lambda}) \tag{2.22}$$

where $\boldsymbol{\lambda}$ is a vector of all generative model parameters. Maximising equation (2.22) with respect to the parameters, $\boldsymbol{\lambda}$, allows ML parameter estimates to be calculated. With correct statistical models and a globally convergent optimisation algorithm, these estimates converge towards the true parameter values in the limit of infinite training data. In practice, however, the correct models are rarely known and optimisation algorithms are often only locally convergent. The optimality of maximum likelihood solutions is therefore not guaranteed. However, with careful initialisation, good performance can be obtained for a variety of tasks [100, 138].

For simple generative models such as the Gaussian distribution, equation (2.22) can be maximised analytically by calculating the partial derivatives of $\mathcal{F}^{\texttt{ml}}(\boldsymbol{\lambda})$ with respect to the model parameters, $\boldsymbol{\lambda}$, and equating them to zero. Solving the resulting set of simultaneous equations yields closed-form solutions for the model parameters. For more complex models, such as the latent-variable models discussed earlier, analytic parameter estimates are often not available. Instead, iterative algorithms such as expectation maximisation (EM) [27] and the Forward-Backward algorithm [7, 100] can be used. These are discussed in more detail in appendix B.

Maximum mutual information (MMI)

One of the most widely used discriminative training criteria for generative models is maximum mutual information (MMI) estimation [5]. Given independent and identically distributed training examples, $\mathcal{O} = \{\boldsymbol{O}_1, \ldots, \boldsymbol{O}_n\}$, with class labels, $\mathcal{Y} = \{y_1, \ldots, y_n\}$, maximum mutual information estimation updates model parameters in order to maximise the posterior probability of the correct transcriptions,

$$\mathcal{F}^{\texttt{mmi}}(\boldsymbol{\lambda}) = P(\mathcal{Y}|\mathcal{O};\boldsymbol{\lambda}) = \prod_{i=1}^{n} P(y_i|\boldsymbol{O}_i;\boldsymbol{\lambda}) \tag{2.23}$$

where $\boldsymbol{\lambda}$ is a vector containing all generative model parameters. Using Bayes' rule, the class-label posterior probabilities in equation (2.23) can be expressed in terms of the generative model likelihoods, $p(\boldsymbol{O}_i|y_i;\boldsymbol{\lambda})$, and class priors, $P(y_i)$,

$$\mathcal{F}^{\texttt{mmi}}(\boldsymbol{\lambda}) = \prod_{i=1}^{n} \left(\frac{p(\boldsymbol{O}_i|y_i;\boldsymbol{\lambda})P(y_i)}{\sum_{\omega\in\Omega} p(\boldsymbol{O}_i|\omega;\boldsymbol{\lambda})P(\omega)} \right) \tag{2.24}$$

where Ω is the set of all classes. Maximising equation (2.24) with respect to the parameters of both the generative models and the prior distributions allows MMI estimates of the model parameters to be calculated. Note that when the class priors are fixed during training, the MMI objective function is equivalent to the conditional maximum likelihood (CML) criterion [86].

For applications such as speech recognition, the set of all possible transcriptions, Ω, is often unfeasibly large, making calculation of the objective function denominator impossible. To avoid this, the denominator summation is often approximated using a small

number of 'likely' transcriptions. These transcriptions are usually calculated using Viterbi decoding of ML-estimated models [97, 136] and are typically stored as lattices or N-best lists. In these lattices and lists, acoustic model likelihoods are often scaled by a constant $0 < \kappa < 1$ in order to match the acoustic and language model dynamic ranges, thereby increasing the number of active hypotheses [136],

$$\mathcal{F}^{\texttt{mmi}}(\boldsymbol{\lambda}) = \prod_{i=1}^{n} \left(\frac{p(\boldsymbol{O}_i, y_i; \boldsymbol{\lambda})^{1/\kappa} P(y_i)}{\sum_{\omega \in \Omega_i} p(\boldsymbol{O}_i, \omega; \boldsymbol{\lambda})^{1/\kappa} P(\omega)} \right) \tag{2.25}$$

The acoustic model scale factor, κ, is typically set to the language model scale used during decoding. Note that scaling the acoustic models has a similar effect to the time-normalisation of conditional augmented model sufficient statistics (section 7.1). Both reduce the dynamic range of the acoustic models and help to prevent a single hypothesis from dominating the denominator score. This can increase the rate of convergence for MMI and CML training, reducing the computational cost associated with parameter estimation.

Minimum Bayes' risk (MBR)

A second form of discriminative training for generative models is the minimum Bayes' risk (MBR) criterion [57]. For a given loss function, MBR training attempts to minimise the expected loss over all possible class labels. For training examples $\mathcal{O} = \{\boldsymbol{O}_1, \ldots, \boldsymbol{O}_n\}$ and sentence transcriptions $\mathcal{Y} = \{y_1, \ldots, y_n\}$, the MBR training criterion is typically written as an expectation over the training data,

$$\begin{aligned} \mathcal{F}^{\texttt{mbr}}(\boldsymbol{\lambda}) &= \sum_{\mathcal{Y}'} P(\mathcal{Y}'|\mathcal{O}; \boldsymbol{\lambda}) l(\mathcal{Y}', \mathcal{Y}) \\ &= \sum_{i=1}^{n} \sum_{\omega \in \Omega} P(\omega|\boldsymbol{O}_i; \boldsymbol{\lambda}) l(\omega, y_i) \end{aligned} \tag{2.26}$$

where Ω is the set of all classes, and $l(\omega, y_i)$ is a task-specific loss function that calculates the 'loss' (or error) of the hypothesised label ω, given the true class label, y_i. When the correct class-label posterior distribution is known and the loss function measures the error rate, MBR training—minimising equation (2.26)—yields parameter estimates that minimise the expected generalisation error. When the correct posterior distribution is not known (as is often the case), the optimality of MBR estimation is not guaranteed.

An example loss function is given below for a speech recognition task.

2.1.4 Speech recognition

In the previous sections, generative statistical models were introduced. Although these have been applied to a wide variety of tasks, including speech recognition [6, 52, 100] and computational biology [61], this work will focus upon the task of speech recognition.

Automatic speech recognition (ASR) is the process of generating sentence transcriptions from recorded speech waveforms. The basic structure of a typical speech recognition system is shown in figure 2.2. This system consists of five main blocks: the front-end, acoustic models, language model, vocabulary and search algorithm.

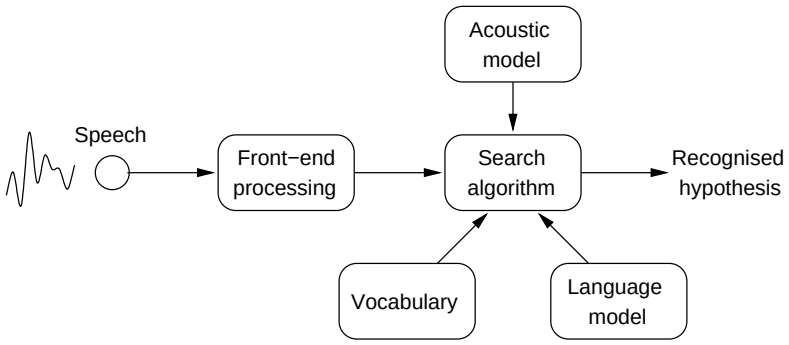


Figure 2.2: A typical speech recognition system

The first stage in a speech recognition system is the front-end. This converts recorded speech utterances into sequences of observation vectors (frames), $\boldsymbol{O} = \{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_T\}$. These observation sequences are then passed to the core of the speech recognition system: the search algorithm. This uses Bayes' decision rule to calculate the most likely word transcription associated with the observation sequence,

$$\hat{\boldsymbol{W}} = \arg\max_{\boldsymbol{W}} P(\boldsymbol{W}|\boldsymbol{O}) = \arg\max_{\boldsymbol{W}} \left(\frac{p(\boldsymbol{O}|\boldsymbol{W})P(\boldsymbol{W})}{p(\boldsymbol{O})} \right) \tag{2.27}$$

where $p(\boldsymbol{O}|\boldsymbol{W})$ is the acoustic model, $P(\boldsymbol{W})$ is the language model and $p(\boldsymbol{O})$ is the evidence. The different parts of a speech recognition system are discussed in more detail in the following sections.

Front-end processing

The first stage of a typical speech recognition system is the collection and processing of speech waveforms. Waveforms are typically recorded using a microphone with a sampling rate of 8kHz or 16kHz. The resulting sampled speech waveforms are rarely used directly since the variability between different speakers and acoustic conditions can make waveform manipulation difficult. Instead, useful information about the speech signal is usually extracted from the spectral shape of the waveforms [26]. The two most commonly used parameterisations for this information are Mel-frequency cepstral coefficients (MFCCs) [24] and perceptual linear prediction (PLP) [46]. Both assume that the speech signal is quasi-stationary, allowing it to be divided into short frames, often 10ms long. For each frame, an observation vector is generated by analysing a short segment of speech within a predefined window (typically 25ms). A Hamming windowing function is often used to reduce the boundary effects in the subsequent processing [26].

Given the short segment of speech within this window, an observation vector (a frame) is calculated as follows. First, a fast-Fourier transform is used to calculate the segment power spectrum in the frequency domain. This is then warped using either a Mel-frequency scale (MFCCs) or a Bark-frequency scale (PLPs). The resulting power spectrum is filtered and compressed. MFCC and PLP observation vectors are then generated using an inverse discrete cosine transform (IDCT) or a linear prediction (LP) analysis respectively. Thirteen-dimensional observation vectors are typically extracted.

In many speech recognition systems the MFCC and PLP features are extended using first-order (delta) and second-order (delta-delta) dynamic features [33]. These increase the feature-space dimensionality to 39, and help to overcome the HMM assumption that observations are conditionally independent given the state sequence. Furthermore, many state-of-the-art speech recognition systems use third-order dynamic features in order to generate 52-dimensional vectors. These are then projected into a smaller 39-dimensional space. One common projection scheme is heteroscedastic linear discriminant analysis (HLDA) [34, 62]. This performs both dimensionality reduction and feature-space decorrelation. In addition to HLDA, state-of-the-art speech recognition systems often include a range of feature-space normalisation techniques such as vocal-tract length-normalisation (VTLN) [70, 124], and cepstral mean and variance normalisation (CMN and CVN) [43].

Acoustic models

The most common form of acoustic model for speech recognition systems is the hidden Markov model [52]. These are used to estimate the observation sequence likelihoods, $p(\boldsymbol{O}|\boldsymbol{W})$, given the complete sentence transcription. For speech recognition tasks with a limited vocabulary it is often possible train HMMs for every possible word. However, as the number of words in the vocabulary increases, it becomes increasingly difficult to robustly estimate HMMs for all words. Instead, a pronunciation dictionary is typically used to split words into smaller sub-word units known as phones [90]. After estimating HMMs for each phone, word or sentence models can be generated by concatenating the phone HMMs.

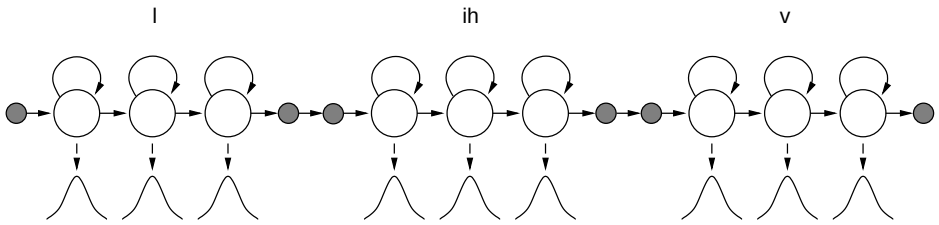


Figure 2.3: A typical monophone speech recognition system

When phone HMMs are estimated without taking phonetic context into account, they are known as context-independent or monophone models. The concatenation of the

context-independent phones 'l', 'ih' and 'v' to form the word 'live' is illustrated in figure 2.3. In practice this approach is rather limited since phone realisation can vary greatly depending upon the preceding and following phones. This effect is known as co-articulation. To model these variations, context-dependent phone models can be used: triphones are the most common. These depend upon both the preceding and the following phones, and may be either word-internal (do not cross word boundaries) or cross-word (word boundaries are ignored). In order to reduce the number of model parameters, triphone models often use parameter tying—typically using decision trees [139]. In this work, context-independent word and phone acoustic models are used. Model-based adaptation schemes [134] are not used.

As discussed in section 2.1.3, parameters of HMM acoustic models can be estimated using a number of different training criteria, including ML, MMI and MBR estimation. Although ML-estimated HMM acoustic models are still widely used, discriminatively trained models have become increasing popular due to their often superior generalisation performance. Unfortunately, for speech recognition, many standard discriminative criteria—such as MMI—minimise an approximation to the sentence error rate, whereas speech recognition systems are usually scored according to the word error rate. To reduce this mismatch, the minimum phone error criterion was proposed [96, 97].

The MPE training criterion is based upon the minimum Bayes' risk framework discussed in section 2.1.3, and uses a loss function that approximates the sentence phone error rate [96, 97]. Since minimising the phone error rate is equivalent to maximising the phone accuracy, MPE training involves maximising the objective function,

$$\mathcal{F}^{\texttt{mpe}}(\boldsymbol{\lambda}) = \sum_{i=1}^{n} \sum_{\boldsymbol{W}} P(\boldsymbol{W}|\boldsymbol{O}_i;\boldsymbol{\lambda}) A(\boldsymbol{W}, \boldsymbol{W}_i) \tag{2.28}$$

where $A(\boldsymbol{W}, \boldsymbol{W}_i)$ is the phone accuracy for a hypothesised sentence transcription $\boldsymbol{W}$, given the correct phone sequence $\boldsymbol{W}_i$. Unfortunately, for many speech recognition tasks, the set of all possible hypothesis sentences is unfeasibly large, making calculation of the objective function in equation (2.28) impossible. To avoid this, the summation over all possible sentence transcriptions is often approximated using a smaller number of 'likely' transcriptions (typically represented as a lattice).

For each hypothesised transcription, $\boldsymbol{W}$, the function $A(\boldsymbol{W}, \boldsymbol{W}_i)$ calculates the sentence phone accuracy. Ideally, this should be calculated after the hypothesis has been aligned with the reference transcription. In practice, however, this alignment can be computationally expensive and, instead, an approximation is often used [96]. This calculates the accuracy of individual phones in the hypothesised transcription using the function,

$$A(q) = \max_{z} \begin{cases} -1 + 2e(q,z) & \text{if } z = q \\ -1 + e(q,z) & \text{otherwise} \end{cases} \tag{2.29}$$

where $e(q, z)$ calculates the proportion of the reference phone duration, z, that is overlapped by the hypothesised phone, q. With perfect alignment between the reference and hypothesised phones, $A(q)$, outputs values of 1, 0 and -1 for a correct phone, a substitution or deletion, and an insertion.

Language modelling

In speech recognition systems, the prior distribution over sentence transcriptions, $P(\boldsymbol{W})$, is often estimated using a language model. Given a sequence of words, $\boldsymbol{W} = \{w_1, \ldots, w_L\}$, the language model probability is usually written as a product of the conditional probabilities of words given their history,

$$P(\boldsymbol{W}) = P(w_1, \ldots, w_L) = \prod_{l=1}^{L} P(w_l | w_{l-1}, \ldots, w_1) \tag{2.30}$$

For continuous speech recognition tasks, the vocabulary is often too large to allow robust estimation of $P(\boldsymbol{W})$. To improve robustness, the language model is often simplified using the assumption that word probabilities depend only upon the last $N-1$ words. This allows the word history to be truncated,

$$P(\boldsymbol{W}) = \prod_{l=1}^{L} P(w_l | w_{l-1}, \ldots, w_{l-N+1}) \tag{2.31}$$

Popular language models are the bigram, trigram and four-gram models, with N equal to 2, 3 and 4 respectively. Although the n-gram language model probabilities in equation (2.31) are easier to calculate than the probabilities in equation (2.30), robust estimation of the probabilities for all possible word combinations is often not possible. Instead, smoothing algorithms such as discounting and backing-off are commonly used [59].

In many speech recognition systems, the acoustic model and language model probabilities have different dynamic ranges. To compensate for this mismatch, the dynamic range of language model probabilities is often increased by scaling the language model log-probabilities by a factor κ (the language model scale factor). The value of κ is usually determined experimentally.

Search algorithm

The core of a speech recognition system is the search algorithm that determines the best sentence transcription for a given observation sequence. As discussed above, this search is usually expressed using Bayes' rule and Bayes' decision rule,

$$\begin{aligned} \hat{\boldsymbol{W}} &= \arg\max_{\boldsymbol{W}} P(\boldsymbol{W}|\boldsymbol{O}) \\ &= \arg\max_{\boldsymbol{W}} \left(\frac{p(\boldsymbol{O}|\boldsymbol{W}) P(\boldsymbol{W})}{p(\boldsymbol{O})} \right) \end{aligned} \tag{2.32}$$

where $p(\boldsymbol{O}|\boldsymbol{W})$ is the acoustic model and $P(\boldsymbol{W})$ is the language model. Note that the evidence, $p(\boldsymbol{O})$, acts as a normalisation term and is often ignored during recognition. As illustrated in equation (2.32), an ideal decoder should be able to search through all possible sentence transcriptions in order to find the one that has the largest posterior probability. In practice, however, direct evaluation of equation (2.32) is expensive and rapidly becomes impractical as sentence lengths increase. To overcome this problem, maximum likelihood HMM state-sequences are often used. These are typically calculated using the Viterbi algorithm, a computationally efficient approximation to the forward probability calculations used during training. Further details of the Viterbi algorithm are given in appendix B.

Scoring

Performance of speech recognition systems is typically evaluated by comparing hypothesised word transcriptions against known reference transcriptions. Scoring proceeds as follows. Hypothesised transcriptions are first aligned against the reference transcriptions using a dynamic programming string matching algorithm. Then, given the aligned hypotheses, the number of substitution (S), deletion (D) and insertion (I) errors is calculated by comparing the words in the reference and hypothesised transcriptions. The word error rate (WER) is then calculated using the expression,

$$\text{WER} = 100\left(1 - \frac{N - D - S - I}{N}\right) \tag{2.33}$$

where N is the total number of words in the reference transcriptions [138]. Word error rates are quoted as percentages.

2.2 Discriminative models

In recent years, discriminative models have become increasingly popular in the search to find alternatives to generative models for segmenting and labelling observation sequences [111]. Defined using features or sufficient statistics extracted from the observation sequence, discriminative models directly model the class decision boundaries, avoiding the need to maintain valid generative or prior distributions. Unlike generative models which are almost exclusively defined within a probabilistic framework, discriminative models can be defined in a variety of ways.

In the next sections, three forms of discriminative model are considered. The first two, conditional random fields (CRFs) [63] and hidden conditional random fields (HCRFs) [42, 99] are examples of probabilistic discriminative models. The third example, support vector machines (SVMs) [10, 19, 125] are a popular distance-based approach.

2.2.1 Conditional random fields (CRFs)

Conditional random fields (CRFs) [63] are one of the most popular forms of discriminative statistical model within the machine learning community. As a generalisation of maximum entropy Markov models (MEMMs) [78], CRFs use a single exponential distribution to model the conditional probability of the label sequence, $\boldsymbol{y} = \{y_1, \ldots, y_T\}$, given the observations, $\boldsymbol{O} = \{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_T\}$,

$$P(\boldsymbol{y}|\boldsymbol{O};\boldsymbol{\alpha}) = \frac{1}{Z(\boldsymbol{O};\boldsymbol{\alpha})} \exp\left(\sum_{t=1}^{T}\sum_{j} \alpha_j f_j(y_t, y_{t-1}, \boldsymbol{O})\right) \tag{2.34}$$

$$Z(\boldsymbol{O};\boldsymbol{\alpha}) = \sum_{\boldsymbol{y}} \exp\left(\sum_{t=1}^{T}\sum_{j} \alpha_j f_j(y_t, y_{t-1}, \boldsymbol{O})\right) \tag{2.35}$$

where $Z(\boldsymbol{O};\boldsymbol{\alpha})$ is a normalisation term, α_j are model parameters and $f_j(y_t, y_{t-1}, \boldsymbol{O})$ are features extracted from the complete observation sequence. Note that although CRF features may depend upon the complete observation sequence, dependencies on the label sequence are limited to only the current and previous labels (a first-order Markov assumption). This allows the normalisation summation in equation (2.35) to be calculated in linear time using a Forward-Backward-style algorithm [7,63,100].[5] It is worth noting that when class labels are drawn from a finite set, the CRF normalisation term is finite, irrespective of the model parameters. This means that, unlike generative models, parameter constraints on $\boldsymbol{\alpha}$ are not required.

As a form of discriminative exponential model, CRFs can be written in terms of sufficient statistics and natural parameters using the conditional version of the canonical exponential model in equation (2.3),

$$P(\boldsymbol{y}|\boldsymbol{O};\boldsymbol{\alpha}) = \frac{1}{Z(\boldsymbol{O};\boldsymbol{\alpha})} \exp\left(\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{T}(y_t, y_{t-1}, \boldsymbol{O})\right) \tag{2.36}$$

$$T_j(y_t, y_{t-1}, \boldsymbol{O}) = \sum_{t=1}^{T} f_j(y_t, y_{t-1}, \boldsymbol{O})$$

where $\boldsymbol{\alpha}$ are natural parameters and $\boldsymbol{T}(y_t, y_{t-1}, \boldsymbol{O})$ are sufficient statistics calculated from the CRF features. The exponential model reference distribution is given by $h(\boldsymbol{O}) = 1$.

One of the most important decisions that must be made when defining CRFs is the choice of features/sufficient statistics since these determine the spatial and temporal dependencies that can be modelled. Unfortunately, CRFs suffer from the same problem as generative exponential models (section 2.1.1) in that, for many applications, it is often not clear what statistics to use. For sequences of discrete observations, expert knowledge is often used to define CRF features using simple binary questions, such as 'is $\boldsymbol{o}_t$ capitalised and y_t a noun?' [63]. These features can either be used directly as CRF sufficient

[5] A detailed explanation of the Forward-Backward algorithm is given in appendix B.

statistics [63] or post-processed using a combinatorial algorithm to generate additional features that capture more complex dependencies [77, 93]. When sequences of continuous observations are considered, however, the choice of CRF features/sufficient statistics can be particularly difficult since there are few intuitive starting points for feature-selection. Some of the most popular features are based upon either Gaussian-style statistics [42] or a windowing approach [130].[6]

2.2.2 Hidden conditional random fields (HCRFs)

In the previous section conditional random fields were introduced as discriminative statistical models for applications where there is a one-to-one relationship between the observations and class labels. For applications such as speech recognition, however, this one-to-one relationship can be inappropriate since the alignment between class labels and observations is usually unknown. Hidden conditional random fields were introduced as one method of overcoming this problem.

Hidden conditional random fields (HCRFs) [99] are a latent-variable extension of standard CRFs that allows the one-to-one relationship between observations and labels to be relaxed. Similarly to CRFs, HCRFs directly model the posterior probability of a class label y, given an observation sequence $\boldsymbol{O} = \{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_T\}$,

$$\begin{aligned} P(y|\boldsymbol{O};\boldsymbol{\alpha}) &= \sum_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} P(y,\boldsymbol{\theta}|\boldsymbol{O};\boldsymbol{\alpha}) \\ &= \frac{1}{Z(\boldsymbol{O};\boldsymbol{\alpha})} \sum_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \exp\left(\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{T}(y,\boldsymbol{\theta},\boldsymbol{O})\right) \end{aligned} \tag{2.37}$$

where $\boldsymbol{\alpha}$ are the model parameters, $\boldsymbol{T}(y,\boldsymbol{\theta},\boldsymbol{O})$ are HCRF sufficient statistics,[7] and $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_T\}$ is a sequence of latent states, selected from the set of all possible latent state sequences, $\boldsymbol{\Theta}$. The normalisation term, $Z(\boldsymbol{O};\boldsymbol{\alpha})$, is calculated as the expectation of the unnormalised HCRF over all class labels and state sequences. When the HCRF latent states follow a Markov assumption, the normalisation term can be calculated using a Forward-Backward-style algorithm [42, 99].

As a latent-variable extension of conditional random fields, HCRFs suffer from the same problem as generative exponential models and CRFs in that, for many applications, it is often not clear which sufficient statistics to use. This choice is especially important for HCRFs since the marginalisation of HCRF posterior distributions over all possible latent-variable sequences may be infeasible for certain choices of latent variable and sufficient

[6] Note that in [42] and [130], experiments were performed using HCRFs (see below). However, the same features can also be used for CRFs.

[7] Note that these sufficient statistics differ from the CRF statistics discussed previously in two ways. First, there is the additional dependence on the latent state sequence. Second, since HCRFs have only a single class label per observation sequence, the CRF dependence on y_t and y_{t-1} can be replaced with a simple dependence on y.

statistics. One approach for defining HCRF latent variables and sufficient statistics is to emulate the structure and sufficient statistics of latent-variable generative models [42, 74]. For example, when latent variables are discrete and follow a first-order Markov assumption, HMM-style sufficient statistics can be defined using the features [42],

$$\begin{aligned}
T^{LM}_{y'}(y, \boldsymbol{\theta}, \boldsymbol{O}) &= \delta(y{=}y') && \forall y' \\
T^{Tr}_{\theta\theta'}(y, \boldsymbol{\theta}, \boldsymbol{O}) &= \sum_{t=1}^{T} \delta(\theta_{t-1}{=}\theta)\delta(\theta_t{=}\theta') && \forall \theta, \theta' \\
T^{Occ}_{\theta}(y, \boldsymbol{\theta}, \boldsymbol{O}) &= \sum_{t=1}^{T} \delta(\theta_t{=}\theta) && \forall \theta \\
\boldsymbol{T}^{G}_{\theta}(y, \boldsymbol{\theta}, \boldsymbol{O}) &= \sum_{t=1}^{T} \delta(\theta_t{=}\theta) \begin{bmatrix} \boldsymbol{o}_t \\ \mathrm{vec}(\boldsymbol{o}_t \boldsymbol{o}_t^{\mathrm{T}}) \end{bmatrix} && \forall \theta
\end{aligned} \tag{2.38}$$

where $\delta(y{=}y')$ is the Kronecker delta: a function that is equal to one when $y{=}y'$ and zero otherwise. As illustrated in equation (2.38), HMM-style HCRF sufficient statistics are composed of elements with four different functional forms. These correspond (in order) to: a unigram prior over the class labels, HMM-style transition probabilities, mixture-component priors, and mixture-component Gaussian sufficient statistics. The advantage of using these statistics within a discriminative framework is that fewer parameter constraints are required compared to equivalent generative models. For example, although generative model variances must be positive-definite, discriminative models such as HCRFs require no such constraints, making them very flexible. When HCRFs were applied to the TIMIT phone classification task [38], good performance was obtained, with HCRFs outperforming both ML and MMI estimated HMMs.

2.2.3 Parameter estimation for conditional models

Unlike generative statistical models where many different training criteria are used, probabilistic discriminative models, such as CRFs and HCRFs, are almost invariably estimated using the conditional maximum likelihood (CML) criterion. This is an inherently discriminatory approach that attempts to directly maximise the posterior probability of the training labels, $\mathcal{Y} = \{y_1, \ldots, y_n\}$, given the observation sequences, $\mathcal{O} = \{\boldsymbol{O}_1, \ldots, \boldsymbol{O}_n\}$. Parameter estimates are obtained by maximising the CML objective function [63],

$$\mathcal{F}^{\texttt{cml}}(\boldsymbol{\alpha}) = P(\mathcal{Y}|\mathcal{O}; \boldsymbol{\alpha}) = \prod_{i=1}^{n} P(y_i|\boldsymbol{O}_i; \boldsymbol{\alpha}) \tag{2.39}$$

where $\boldsymbol{\alpha}$ are the CRF/HCRF model parameters. It is worth noting that the CML objective function, above, is identical to the MMI objective function in equation (2.23). The difference between the two approaches lies in whether they update a generative model (MMI) or a discriminative model (CML).

For many discriminative models, such as CRFs and HCRFs, analytic closed-form solutions for the CML objective function in equation (2.39)—obtained by setting the gradient to zero and solving for $\boldsymbol{\alpha}$—do not exist. Instead, models are typically estimated using iterative algorithms: EM-style algorithms such as generalised iterative scaling [23, 63] and improved iterative scaling [63, 93] are commonly used for CRF parameter estimation. Alternatively, gradient-based techniques can be used. These often have simpler implementations and have been shown to have faster convergence in many cases [114, 129]. The CRF objective function is convex, causing the objective function to have a single, global, maximum. In contrast, the latent-variable structure of HCRFs yields a non-convex objective function that potentially has many local maxima. To minimise the effects of these local maxima, HCRF parameters are usually estimated using stochastic gradient-based methods [42, 74, 99, 130].

2.2.4 Support vector machines

Support vector machines (SVMs) [10, 19, 125] are based upon the intuitive concept of maximising the margin of separation—defined as the distance between the decision hyperplane and the closest training examples—between two competing classes. This has been shown to be related to minimising an upper bound on the generalisation error [125].

Consider a set of d-dimensional training examples, $\mathcal{O} = \{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_n\}$, $\boldsymbol{o}_i \in \mathbb{R}^d$, with class labels, $\mathcal{Y} = \{y_1, \ldots, y_n\}$, $y_i \in \{-1, +1\}$. When the training examples are linearly separable, a linear decision hyperplane can be located such that all examples in the training set are correctly classified. For a hyperplane with a gradient (weight) vector $\boldsymbol{w}$ and bias b, the hyperplane decision function is given by the equation,

$$y = \text{sign}(\boldsymbol{w}^{\text{T}}\boldsymbol{o} + b) \tag{2.40}$$

Since this function is invariant under a positive rescaling of the hyperplane parameters, parameter scaling must be fixed in order to obtain a unique solution. This is typically achieved by defining canonical hyperplanes on both sides of the decision hyperplane, $\boldsymbol{w}^{\text{T}}\boldsymbol{o} + b = 1$ and $\boldsymbol{w}^{\text{T}}\boldsymbol{o} + b = -1$. Training examples are then constrained to lie outside the region enclosed by the canonical hyperplanes. This arrangement is depicted in figure 2.4 for the simple case of two-dimensional data.

In this figure, the optimal and canonical hyperplanes are depicted by the solid and dashed lines respectively. As discussed above, the decision hyperplane is positioned such that no training examples lie in the shaded region between the canonical hyperplanes. Examples that lie on the canonical hyperplanes are known as support vectors and (as discussed below) play an important role in the calculation and optimisation of the decision hyperplane.

The perpendicular distance between the optimal/decision hyperplane and the canonical hyperplanes is known as the margin. Using the definition of the canonical hyperplanes,

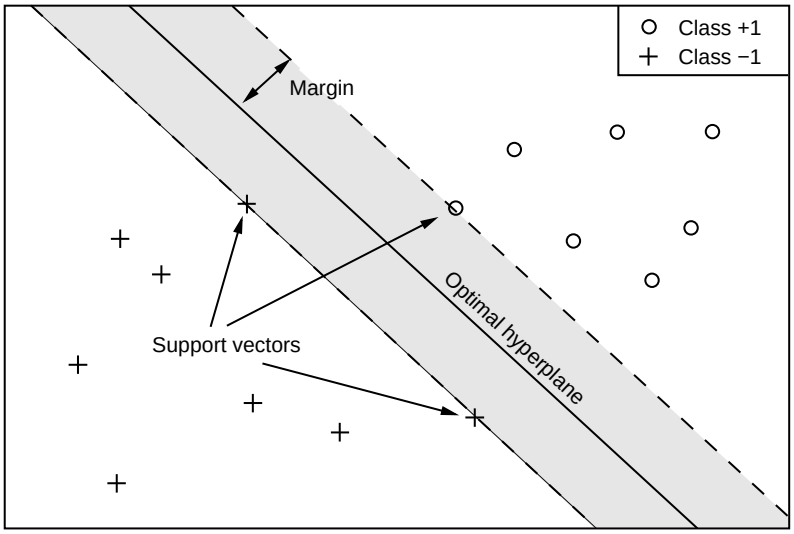


Figure 2.4: The optimal SVM hyperplane for linearly separable data

the size of this margin can be calculated using the expression [22],

$$\text{Margin} = \frac{1}{|\boldsymbol{w}|} \tag{2.41}$$

Statistical learning theory states that the decision hyperplane that minimises the probability of generalisation error is the one that maximises this margin [11, 22, 125]. Since SVMs are designed to minimise generalisation error, the SVM primal objective function is given by,

$$\begin{aligned} \text{minimise}_{\boldsymbol{w},b} \quad & \tfrac{1}{2}\langle \boldsymbol{w}, \boldsymbol{w} \rangle & & \\ \text{subject to} \quad & y_i\big(\langle \boldsymbol{w}, \boldsymbol{o}_i \rangle + b\big) \geq 1 & & \forall i \in [1, n] \end{aligned} \tag{2.42}$$

where $\langle \cdot, \cdot \rangle$ denotes an inner-product between two vectors, and $\boldsymbol{o}_i$ are training examples.

Unfortunately, many data sets are noisy and contain outliers. For these data, it may not be possible to find a linear decision boundary that correctly separates all training examples. To enable SVM training to converge for such data, the margin constraints, $y_i\big(\langle \boldsymbol{w}, \boldsymbol{o}_i \rangle + b\big) \geq 1$, are often relaxed to allow training examples to be misclassified. The resulting constraints are known as the soft-margin SVM constraints, and are given by $y_i(\langle \boldsymbol{w}, \boldsymbol{o}_i \rangle + b) \geq 1 - \epsilon_i$, where the slack variables, $\epsilon_i \geq 0$, measure the distance by which an example has failed to meet the original margin constraint. To ensure that the margin is not increased at the expense of unnecessary classification errors, the SVM objective function is altered such that incorrectly classified training examples are penalised. The

resulting SVM objective function and constraint are given by,

$$\begin{aligned} \text{minimise}_{\boldsymbol{w},b} \quad & \tfrac{1}{2}\langle \boldsymbol{w}, \boldsymbol{w} \rangle + C \sum_{i=1}^{n} \epsilon_i & \\ \text{subject to} \quad & y_i\big(\langle \boldsymbol{w}, \boldsymbol{o}_i \rangle + b\big) \geq 1 - \epsilon_i & \forall i \in [1, n] \\ \text{and} \quad & \epsilon_i \geq 0 & \forall i \in [1, n] \end{aligned} \tag{2.43}$$

where the regularisation parameter, C, controls the trade-off between maximising the margin and reducing the number of misclassified examples.

Although the soft-margin SVM objective function in equation (2.43) can be optimised in its primal form—for example [60]—it is often easier to consider the dual form of the objective [125]. This can be written as,

$$\begin{aligned} \text{maximise}_{\boldsymbol{\alpha}^{\texttt{svm}}} \quad & \sum_{i=1}^{n} \alpha_i^{\texttt{svm}} - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i^{\texttt{svm}} \alpha_j^{\texttt{svm}} y_i y_j \langle \boldsymbol{o}_i, \boldsymbol{o}_j \rangle \\ \text{subject to} \quad & \sum_{i=1}^{n} \alpha_i^{\texttt{svm}} y_i = 0 \\ \text{and} \quad & 0 \leq \alpha_i^{\texttt{svm}} \leq C \end{aligned} \tag{2.44}$$

where $\alpha_i^{\texttt{svm}}$ are Lagrange multipliers associated with training examples, $\boldsymbol{o}_i$. The upper bound, C, on the Lagrange multipliers limits the influence of individual examples, and is typically selected using either a development set or a data-dependent algorithm such as [55]. The dual objective function in equation (2.44) is a quadratic function of the Lagrange multipliers and is, by definition, convex. This allows the optimisation to converge to a single global solution. Although many algorithms have been proposed for training SVMs, two of the most popular are sequential minimal optimisation [94] and the decomposition and chunking algorithms in [55, 112].

At optimality, the Karush-Kuhn-Tucker (KKT) conditions ensure that only examples that lie either on the margin, $y_i(\langle \boldsymbol{w}, \boldsymbol{o}_i \rangle + b) = 1$, or on the wrong side of the margin, $y_i(\langle \boldsymbol{w}, \boldsymbol{o}_i \rangle + b) \leq 1$, have non-zero Lagrange multipliers, $\alpha_i^{\texttt{svm}} > 0$. These examples are known as the support vectors [22, 125]. Using Lagrangian theory, the weight vector, $\boldsymbol{w}$, and bias, b, of the optimal hyperplane can be calculated using only the support vectors,

$$\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i^{\texttt{svm}} y_i \boldsymbol{o}_i \tag{2.45}$$

$$b = -\frac{\max_{y_i=-1}(\langle \boldsymbol{w}, \boldsymbol{o}_i \rangle) + \min_{y_i=1}(\langle \boldsymbol{w}, \boldsymbol{o}_i \rangle)}{2} \tag{2.46}$$

where $\max_{y_i=-1}(\cdot)$ selects the training example from the class -1 that is closest to the decision hyperplane, and $\min_{y_i=1}(\cdot)$ selects the example in class $+1$ that is closest.

Thus far, only linearly separable data sets (with and without noise) have been considered. For many classification tasks, however, examples are not linearly separable, and a

more complex representation of the decision boundary is needed. To accommodate these tasks, Cover's theorem [20] can be used. This states that examples can be made linearly separable with a high probability given a non-linear transformation, $\boldsymbol{\phi}(\boldsymbol{o};\boldsymbol{\lambda})$, from the observation-space to a feature-space of sufficiently high dimensionality. Linear decision boundaries in this high dimensional feature-space correspond to non-linear decision boundaries in the original observation-space. By explicitly mapping all training examples into the feature-space, the SVM dual objective function can be rewritten as,

$$\text{maximise}_{\boldsymbol{\alpha}^{\texttt{svm}}} \quad \sum_{i=1}^{n} \alpha_i^{\texttt{svm}} - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i^{\texttt{svm}}\alpha_j^{\texttt{svm}} y_i y_j \big\langle \boldsymbol{\phi}(\boldsymbol{o}_i;\boldsymbol{\lambda}), \boldsymbol{\phi}(\boldsymbol{o}_j;\boldsymbol{\lambda})\big\rangle \tag{2.47}$$

subject to the same constraints as equation (2.44). In practice, however, for high-dimensional feature-spaces, this explicit mapping of examples into the feature-space may be impractical. Instead, by noting that equation (2.47) is a function of only the distance between feature-space points, and not the points themselves, SVMs can be written in terms of a kernel function [1, 10, 125],

$$K(\boldsymbol{o}_i, \boldsymbol{o}_j;\boldsymbol{\lambda}) = \big\langle \boldsymbol{\phi}(\boldsymbol{o}_i;\boldsymbol{\lambda}), \boldsymbol{\phi}(\boldsymbol{o}_j;\boldsymbol{\lambda})\big\rangle \tag{2.48}$$

In general, this function may be any symmetric non-linear function that satisfies Mercer's condition [79, 115, 125].[8] The advantage of expressing SVM training in terms of a kernel function is that, for some feature-spaces, the feature-space mapping and inner-product operations can be combined into a single efficient calculation. In such cases it is not necessary to explicitly calculate the high-dimensional feature-space. When the implicit feature-space has a much higher dimensionality than the observation-space, significant computational savings can be achieved through the use of kernel functions.

Table 2.1: A selection of popular kernels

Kernel	Functional form $K(\boldsymbol{o}_i, \boldsymbol{o}_j)$	Kernel parameters
Linear	$\langle \boldsymbol{o}_i, \boldsymbol{o}_j\rangle$	–
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	$\exp(-\|\|\boldsymbol{o}_i - \boldsymbol{o}_j\|\|/\sigma)$	Variance, σ
RBF	$\exp(-\|\|\boldsymbol{o}_i - \boldsymbol{o}_j\|\|^2/2\sigma^2)$	Variance, σ

Over the last decade, many different kernel functions have been proposed for mapping observations into high-dimensional feature-spaces. A small selection of popular kernels are given in table 2.1. With the exception of the linear kernel, these kernels all have an

[8] Mercer's condition simply states that the kernel matrix $\boldsymbol{K}(\boldsymbol{o}_i, \boldsymbol{o}_j;\boldsymbol{\lambda})\}_{(i,j)=1}^{n}$ must be positive semi-definite. This ensures that an expansion $\sum_{k=1}^{\infty} a_k z_k(\boldsymbol{o}_i) z_k(\boldsymbol{o}_j)$ exists where $\{z_1(\boldsymbol{o}), \ldots, z_k(\boldsymbol{o}), \ldots\}$ is some unknown feature-space. The kernel thus defines a inner-product in this implicit (possibly incalculable) feature-space.

implicit kernel feature-space with a dimensionality that varies with the kernel parameters. Although many kernels—for example, polynomial kernels—have a fixed dimensional feature-space, others—such as the Laplacian and RBF kernels—generate feature-spaces whose dimensionality varies with the number of training examples.

2.3 Summary

In this chapter, two different forms of statistical model were introduced: generative statistical models and discriminative statistical models.

Generative statistical models are probably the most prevalent form of statistical model currently in use for machine classification of sequence data. Using a sometimes complex parametric form, generative models approximate the probability distribution function associated with the observation sequence generation process, allowing sequence likelihoods to be calculated. Given these likelihoods, class label posterior probabilities can be calculated using Bayes' rule, with the final class label selected using Bayes' decision rule. Various forms of generative model were introduced, ranging from simple exponential models to complex latent-variable models that model long-range temporal dependencies (e.g. SLDSs). Maximum likelihood (ML), maximum mutual information (MMI), and minimum phone error (MPE) training criteria were briefly discussed.

In the second half of this chapter, discriminative statistical models were introduced. These directly model the class decision boundaries and avoid the need to maintain valid generative models. Two different approaches were discussed. The first used an exponential form to directly model the posterior probabilities of the class labels. The resulting discriminative models did not need the strong independence assumptions typically required by generative models. This allows them to model a much wider range of dependencies (particularly temporal). The most common form of conditional model, the CRF, was discussed, along with its latent-variable extension, the HCRF. Conditional maximum likelihood (CML) estimation was also briefly reviewed. Support vector machines, a second form of discriminative model, were then discussed. These are distance-based classifiers, trained using a maximum-margin training criterion.

3

Dynamic Kernels

Originally proposed in 1964 by Aizermann et al. [1], kernel-based classifiers have recently become popular with the invention and popularisation of support vector machines (SVMs) [10, 19, 125]. As discussed in the previous chapter, kernel-based techniques are based upon a kernel function that maps examples into an (often implicit) high-dimensional feature-space where linear classification can be performed. Over the last decade, many kernels have been developed for mapping vector-based examples into a wide range of different implicit feature-spaces [22]. However, when examples are best represented as sequences of observations, the options are rather more limited.

In this chapter, dynamic kernels—kernels that map variable-length observation sequences into a fixed-dimensional feature-space—are discussed as a powerful approach for applying standard kernel classification techniques to variable-length observation sequences. The dynamic kernels in this chapter are split into two overlapping types: those that operate on discrete observations, and those that operate on both discrete and continuous observations. To avoid confusion, the second category of kernels are described as continuous dynamic kernels in this work, reflecting the fact that kernels for continuous-observation sequences are often harder to define than kernels for discrete-observation sequences. Separate sections are dedicated to discrete and continuous dynamic kernels, with particular emphasis placed upon novel kernel mappings and concepts and approaches that will be useful in later chapters. The dynamic kernels discussed in this chapter are: the bag-of-words kernel, string kernels, marginalised count kernels, Fisher kernels, generative kernels and sequence kernels.

3.1 Discrete-observation dynamic kernels

There has recently been great interest in the use of dynamic kernels in the fields of computational biology [49, 123] and text processing [54, 73]. Characterised by their sequences of discrete observations, training examples in these fields are both easy to manipulate and amenable to the application of dynamic-programming algorithms. Using such techniques, researchers have developed a wide range of intuitive dynamic kernels, such as the bag-of-words kernel [54], string kernels [73], marginalised count kernels [123] and rational kernels [17, 18].

3.1.1 Bag-of-word kernels

One of the earliest applications of kernel-based techniques to sequence data was in text categorisation [22, 54]. This is a supervised classification problem where documents are classified into one or more predefined categories according to their content. For this task, each document is represented as a sequence of T words, $\boldsymbol{O} = \{o_1, \ldots, o_T\}$, along with a single class label, y (the document type).

The bag-of-words kernel [54] was proposed as a practical approach for mapping sequences of words into a fixed-dimensional feature-space. In its simplest form, the kernel has a feature-space defined by the vector of word counts (every element of the vector corresponds to a single word). For a set of k words, the bag-of-words feature-space can be written as,

$$\boldsymbol{\phi}^{\texttt{BoW}}(\boldsymbol{O}) = \begin{bmatrix} f(w_1|\boldsymbol{O}) \\ f(w_2|\boldsymbol{O}) \\ \vdots \\ f(w_k|\boldsymbol{O}) \end{bmatrix} \tag{3.1}$$

where w_1 and w_2 are specific words and $f(w_1|\boldsymbol{O})$ denotes the number of occurrences of the word w_1 in a sequence $\boldsymbol{O}$. The disadvantage of this simplistic approach is that words with similar meanings but different spellings, such as 'compute', 'computed' and 'computing', are mapped into different regions of the feature-space, making the feature-space mapping overly dependent on the tense and style used by the document's author. To avoid this, words can be replaced by their word-stems before the feature-space is constructed [54]. This reduces the number of features and ensures that similar words are treated equally, for example, 'compute', 'computed' and 'computing' would all be mapped to the single stem 'comput-'. The word-stem-based bag-of-word feature-space is thus given by,

$$\boldsymbol{\phi}^{\texttt{BoWS}}(\boldsymbol{O}) = \begin{bmatrix} f(ws_1|\boldsymbol{O}) \\ f(ws_2|\boldsymbol{O}) \\ \vdots \end{bmatrix} \tag{3.2}$$

where ws_1 and ws_2 are word stems. The word-stem bag-of-words kernel is defined as the dot-product in the feature-space,

$$\begin{aligned} K^{\texttt{BoWS}}(\boldsymbol{O}_i, \boldsymbol{O}_j) &= \boldsymbol{\phi}^{\texttt{BoWS}}(\boldsymbol{O}_i)^{\mathrm{T}} \boldsymbol{\phi}^{\texttt{BoWS}}(\boldsymbol{O}_j) \\ &= \sum_{ws \in \mathcal{W}_{\mathcal{S}}} f(ws|\boldsymbol{O}_i) f(ws|\boldsymbol{O}_j) \end{aligned} \tag{3.3}$$

where $\mathcal{W}_{\mathcal{S}}$ is a dictionary containing all known word-stems. For large dictionaries, the sparseness of the feature-space can make kernel calculations on the full feature-space inefficient. However, computational efficiency can be improved by replacing the summation over all possible words in equation (3.3) with the simpler summation over only the words that occur in $\boldsymbol{O}_i$ and $\boldsymbol{O}_j$. For large dictionaries, this can reduce computational cost by several orders of magnitude. Bag-of-words kernels can also be implemented efficiently using the rational kernel framework. This is discussed in more detail in chapter 5.

3.1.2 String kernels

A second form of dynamic kernel for sequences of discrete observations is the string kernel [73,110]. Although originally applied to the same document classification application as the bag-of-words kernel, string kernels take a fundamentally different approach. Whereas the bag-of-words kernel assumes that the smallest units within a document are words, string kernels operate directly on the sequence of characters that make up these words. Document similarity is based upon the number of shared substrings.

One of the simplest forms of string kernel is defined using a feature-space of n-gram counts. These counts are defined as the number of occurrences of a contiguous substring of n characters.[1] For example, the 3-gram (trigram) features for the text 'string kernel' are: str, tri, rin, ing, ng_, g_k, _ke, ker, ern, rne, nel, where '_' denotes a space. For long substrings (large n) the feature-space can grow rapidly, resulting in high-dimensional, sparse feature-spaces with dimensionality $|\Delta|^n$, where Δ is the set of input characters. Fortunately, for typical documents, a large number of trigrams are unlikely to appear in either the training or test set, for example: dqw or zzz. Such trigrams can be excluded from the feature-space, significantly reducing computational complexity. An example trigram feature-space is defined as,

$$\boldsymbol{\phi}^{\texttt{n-gram}}(\boldsymbol{O}) = \begin{bmatrix} f(aaa|\boldsymbol{O}) \\ f(aab|\boldsymbol{O}) \\ \vdots \end{bmatrix} \tag{3.4}$$

where $f(aaa|\boldsymbol{O})$ is the number of occurrences of the trigram aaa in the observation sequence $\boldsymbol{O}$. The definition can be extended to n-grams of any length.

[1] Note that when documents are tokenised into words rather than characters, the unigram kernel is equivalent to the bag-of-words kernel.

A second form of string kernel is the string subsequence kernel (SSK) [73], also known as the gappy-n-gram kernel [18,71]. This uses similar features to the n-gram feature-space above, except that characters in subsequences are no longer required to be consecutive. Instead, subsequences are assigned weights according to how contiguous their constituent characters are, with widely-spaced characters receiving a smaller weight than consecutive characters. Weights are calculated by first calculating the subsequence length that contains the feature-space characters. For example, for the word 'speech' and feature 'pc', the subsequence length is 4 (for the character sequence 'peec'). This subsequence length is then combined with a decay factor, $\boldsymbol{\lambda} \in (0, 1)$, in order to determine the feature-space weight. For the feature 'pc' the weight will be $\boldsymbol{\lambda}^4$. Next, using the subsequence feature-space, the SSK kernel can be defined. Consider, for example, the feature-spaces calculated for the three words: hat, mat and met.

	Feature-space dimensions						
	at	et	ha	ht	ma	me	mt
$\boldsymbol{\phi}^{\mathtt{ssk}}(\text{hat})$	$\boldsymbol{\lambda}^2$	-	$\boldsymbol{\lambda}^2$	$\boldsymbol{\lambda}^3$	-	-	-
$\boldsymbol{\phi}^{\mathtt{ssk}}(\text{mat})$	$\boldsymbol{\lambda}^2$	-	-	-	$\boldsymbol{\lambda}^2$	-	$\boldsymbol{\lambda}^3$
$\boldsymbol{\phi}^{\mathtt{ssk}}(\text{met})$	-	$\boldsymbol{\lambda}^2$	-	-	-	$\boldsymbol{\lambda}^2$	$\boldsymbol{\lambda}^3$

Given these feature-spaces, the SSK kernel is defined as the feature-space dot-product. The string pairs hat/mat and met/met therefore have kernel values of $K^{\mathtt{ssk}}(\text{hat}, \text{mat}) = \boldsymbol{\lambda}^4$ and $K^{\mathtt{ssk}}(\text{met}, \text{met}) = \boldsymbol{\lambda}^6 + 2\boldsymbol{\lambda}^4$ respectively. With many possible features, it is rarely possible to calculate SSK features directly (even for small documents with short features). Instead, the kernel can be calculated using efficient dynamic programming algorithms [73]. These algorithms scale linearly with both subsequence length and with document lengths T_i and T_j. Alternatively, as discussed in [18], string subsequence kernels can be calculated efficiently within the finite-state transducer framework of rational kernels. This is discussed in more detail in section 5.

3.1.3 Marginalised count kernels

In [123] marginalised kernels were proposed as a method of combining generative models with traditional kernel classification techniques. Given a set of training examples, $\mathcal{O} = \{\boldsymbol{O}_1, \ldots, \boldsymbol{O}_n\}$, parameters of a single latent variable generative model, $p(\boldsymbol{O}; \boldsymbol{\lambda})$ are first estimated, capturing the high-level structure of the data. Next, for every observation, $\boldsymbol{O}_i$, the posterior probabilities of all possible generative model state sequences, $P(\boldsymbol{\theta}_i | \boldsymbol{O}_i; \boldsymbol{\lambda})$, are calculated and used to weight examples in a feature-space $\boldsymbol{\phi}(\{\boldsymbol{O}_i, \boldsymbol{\theta}_i\})$. This feature-space maps points $\{\boldsymbol{O}_i, \boldsymbol{\theta}_i\}$ from the 'observation/state'-space to a high-dimensional space where examples are linearly separable. Marginalised kernels are defined by the inner-product in

this feature-space and are written as,

$$K^{\mathtt{mar}}(\boldsymbol{O}_i, \boldsymbol{O}_j; \boldsymbol{\lambda}) = \sum_{\boldsymbol{\theta}_i \in \boldsymbol{\Theta}} \sum_{\boldsymbol{\theta}_j \in \boldsymbol{\Theta}} P(\boldsymbol{\theta}_i|\boldsymbol{O}_i; \boldsymbol{\lambda}) P(\boldsymbol{\theta}_j|\boldsymbol{O}_j; \boldsymbol{\lambda}) K(\{\boldsymbol{O}_i, \boldsymbol{\theta}_i\}, \{\boldsymbol{O}_j, \boldsymbol{\theta}_j\}) \tag{3.5}$$

where $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_j$ denote state sequences associated with the observations $\boldsymbol{O}_i$ and $\boldsymbol{O}_j$, and $K(\{\boldsymbol{O}_i, \boldsymbol{\theta}_i\}, \{\boldsymbol{O}_j, \boldsymbol{\theta}_j\})$ is a kernel that calculates distances between the feature-space points $\{\boldsymbol{O}_i, \boldsymbol{\theta}_i\}$ and $\{\boldsymbol{O}_j, \boldsymbol{\theta}_j\}$. The choice of kernel is restricted by the requirement that it must allow distances between variable-length sequences of observations and states to be calculated.

Although, in theory, $K(\{\boldsymbol{O}_i, \boldsymbol{\theta}_i\}, \{\boldsymbol{O}_j, \boldsymbol{\theta}_j\})$ can be any kernel function that maps sequences of observations/latent-states into a fixed-dimensional space, in practice the choice is often more limited. This is because for many observation sequences, the computation cost of summing over all possible latent state sequences, $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_j$, can be prohibitive. To avoid this, the kernel must allow the summation to be simplified. One such kernel is the count, or n-gram, kernel (note that when count kernels are used, $K(\cdot, \cdot)$ is assumed to be independent of the latent-state sequences $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_j$). Using this kernel, first-order and second-order marginalised count kernels, with feature-spaces $\boldsymbol{\phi}^{\mathtt{mc1}}(\boldsymbol{O}; \boldsymbol{\lambda})$ and $\boldsymbol{\phi}^{\mathtt{mc2}}(\boldsymbol{O}; \boldsymbol{\lambda})$, can be defined [123],

$$\boldsymbol{\phi}_j^{\mathtt{mc1}}(\boldsymbol{O}; \boldsymbol{\lambda}) = \sum_{t=1}^{T} P(\theta_t = j|\boldsymbol{O}; \boldsymbol{\lambda}) \tag{3.6}$$

$$\boldsymbol{\phi}_{jk}^{\mathtt{mc2}}(\boldsymbol{O}; \boldsymbol{\lambda}) = \sum_{t=2}^{T} P(\theta_{t-1} = j, \theta_t = k|\boldsymbol{O}; \boldsymbol{\lambda}) \tag{3.7}$$

where $\boldsymbol{\phi}_j^{\mathtt{mc1}}(\boldsymbol{O}; \boldsymbol{\lambda})$ is the j-th element of the first-order marginalised count kernel feature-space and $\boldsymbol{\phi}_{jk}^{\mathtt{mc2}}(\boldsymbol{O}; \boldsymbol{\lambda})$ is the jk-th element of a second-order feature-space. By extending the marginalised count kernel to higher-order n-grams, more complex dependencies of the form, $\boldsymbol{\phi}_{ij\ldots k}^{\mathtt{mcn}}(\boldsymbol{O}) = \sum_{t=1}^{T-n} P(\theta_t^i, \theta_{t+1}^j, \ldots, \theta_{t+n}^k|\boldsymbol{O})$, can be modelled.

3.2 Continuous-observation dynamic kernels

In the following sections, a number of dynamic kernels for classifying sequences of continuous observations are presented.[2]

3.2.1 Fisher kernels

One of the most popular dynamic kernels for classifying sequences of continuous observations is the Fisher kernel [50, 89, 119, 127]. Defined using a generative model embedded

[2]Note that, in addition to classification of sequences of continuous observations, all of the kernels discussed in this section can be used to classify discrete-observation sequences. The reverse is not true for discrete dynamic kernels.

within a kernel-based framework, Fisher kernels combine the generative model's ability to process variable-length sequences with the flexibility and generalisation performance of discriminative kernel-based algorithms. Given a training set, $\mathcal{O} = \{\boldsymbol{O}_1, \ldots, \boldsymbol{O}_n\}$, parameters of a single generative model, $p(\boldsymbol{O}; \boldsymbol{\lambda})$, are first estimated using the maximum-likelihood criterion. The output of this model is a one-dimensional log-likelihood score. Since this score yields little information about how examples differ, it is of limited use when defining kernel feature-spaces. Instead, to capture the differences in the generative process between different examples, a feature-space defined by the gradients of the generative model can be used,

$$\boldsymbol{\phi}^{\mathrm{F}}(\boldsymbol{O}; \boldsymbol{\lambda}) = \nabla_{\boldsymbol{\lambda}} \ln p(\boldsymbol{O}; \boldsymbol{\lambda}) \tag{3.8}$$

Since the gradient of a statistical model is known as the Fisher score [25], feature-spaces defined using this gradient are known as Fisher score-spaces [50].

For applications where only a small proportion of the training data is labelled, Fisher score-spaces allow both labelled and unlabelled data to be utilised within a discriminative classification framework. This is possible since maximum-likelihood estimation of the generative model does not require knowledge of the class labels, allowing all training data (both labelled and unlabelled) to be used to estimate the generative model parameters. When labelled training data is limited, the use of unlabelled data allows more robust generative model parameter estimates to be obtained, resulting in a generative distribution that better reflects the dynamic properties of the complete training set. When this distribution is used to map labelled training examples into the Fisher score-space, the resulting Fisher scores are based upon information extracted from the whole training set (both labelled and unlabelled examples), potentially improving classification performance.

Given the Fisher score-space in equation (3.8), Fisher kernels are defined using the score-space inner-product [50],

$$K^{\mathrm{F}}(\boldsymbol{O}_i, \boldsymbol{O}_j; \boldsymbol{\lambda}) = \boldsymbol{\phi}^{\mathrm{F}}(\boldsymbol{O}_i; \boldsymbol{\lambda})^{\mathrm{T}} \boldsymbol{G}^{-1} \boldsymbol{\phi}^{\mathrm{F}}(\boldsymbol{O}_j; \boldsymbol{\lambda}) \tag{3.9}$$

where $\boldsymbol{G}^{-1}$ is the score-space metric, defined using a generalised form of the Fisher information matrix [50, 119],

$$\boldsymbol{G} = \mathcal{E}_{\boldsymbol{O}} \left\{ \Big(\boldsymbol{\phi}^{\mathrm{F}}(\boldsymbol{O}; \boldsymbol{\lambda}) - \boldsymbol{\mu}_{\boldsymbol{\phi}} \Big) \Big(\boldsymbol{\phi}^{\mathrm{F}}(\boldsymbol{O}; \boldsymbol{\lambda}) - \boldsymbol{\mu}_{\boldsymbol{\phi}} \Big)^{\mathrm{T}} \right\} \tag{3.10}$$

For many latent-variable generative models, the Fisher information matrix in equation (3.10) has no closed-form solution. Instead it is usually approximated using an expectation over the training examples [25]. Even with this approximation, the Fisher information matrix and its inverse can still be expensive to calculate for high-dimensional score-spaces. To avoid this, further approximations are often used to simplify calculation. The simplest approach, proposed by Jaakkola *et al.*, is to assume that the Fisher information matrix can be approximated by the identity matrix [50]. This assumes that score-space elements are

orthonormal and that the score-space is Euclidean. Alternatively, as used in this thesis, a diagonal approximation to the Fisher information matrix can be used [119]. This provides a reasonable approximation to the matrix whilst reducing the computational cost of the matrix inversion from $\mathcal{O}(n^3)$ to $\mathcal{O}(n)$.

3.2.2 Generative kernels

When Fisher kernels are applied to latent-variable models (in particular, mixture models), 'wrap-around' can occur [118]. This arises when multiple points in the observation-space map to the same score-space point, increasing the confusability of the classes. To illustrate this, consider the Fisher score-space defined by a GMM base model containing two widely-spaced mixture-components. If an observation is generated at the mean of the first mixture-component, the Fisher scores relative to both mixture-component means are zero (the first from a zero gradient, and the second from a zero posterior). An observation generated at the mean of the second mixture-component yields identical scores. It is therefore impossible to separate the two observations. A similar effect occurs with the mixture-component variances. By allowing different observations to be mapped into the same score-space region, confusion between the classes can occur, potentially affecting classification accuracy. This confusion is known as wrap-around [118].

For applications with fully labelled training sets, generative kernels offer a simple solution [117, 118]. Instead of defining score-spaces using a single shared model, generative kernels utilise separate generative models for each class, $p(\boldsymbol{O};\boldsymbol{\lambda}^{(1)})$ and $p(\boldsymbol{O};\boldsymbol{\lambda}^{(2)})$. Although both models are likely to suffer from wrap-around, the combination of scores from different models means that wrap-around in the final score-space is much less likely. The chances of different observations mapping to the same score-space point is further reduced by including the log-likelihood ratio of the two classes in the score-space. Generative score-spaces are defined by,

$$\boldsymbol{\phi}^{\mathrm{LL}}(\boldsymbol{O};\boldsymbol{\lambda}) = \begin{bmatrix} \ln p(\boldsymbol{O};\boldsymbol{\lambda}^{(1)}) - \ln p(\boldsymbol{O};\boldsymbol{\lambda}^{(2)}) \\ \nabla_{\boldsymbol{\lambda}^{(1)}} \ln p(\boldsymbol{O};\boldsymbol{\lambda}^{(1)}) \\ -\nabla_{\boldsymbol{\lambda}^{(2)}} \ln p(\boldsymbol{O};\boldsymbol{\lambda}^{(2)}) \end{bmatrix} \tag{3.11}$$

where $\boldsymbol{\lambda} = \{\boldsymbol{\lambda}^{(1)}, \boldsymbol{\lambda}^{(2)}\}$, and $\boldsymbol{\lambda}^{(1)}$ and $\boldsymbol{\lambda}^{(2)}$ are parameters for each of the generative models. It is worth noting that when the base models used to define a generative kernel are identical (i.e. the parameters are tied, $\boldsymbol{\lambda}^{(1)} = \boldsymbol{\lambda}^{(2)}$), a standard Fisher kernel is obtained.

3.2.3 Sequence kernels

In [12] sequence kernels were proposed. Unlike the kernels discussed thus far, sequence kernels separate the definition of the high-dimensional feature-space from the handling of

variable-length sequences. First, individual observations, $\boldsymbol{o}_t \in \boldsymbol{O}$, are mapped into a high-dimensional feature-space using a standard kernel feature-space mapping, $\boldsymbol{\phi}(\boldsymbol{o}_t)$; polynomial and RBF feature-space mappings are commonly used. This results in a variable-length sequence of implicitly-defined feature-space points, $\boldsymbol{\Phi}(\boldsymbol{O}) = \{\boldsymbol{\phi}(\boldsymbol{o}_1), \ldots, \boldsymbol{\phi}(\boldsymbol{o}_T)\}$. A second mapping function, $\boldsymbol{\phi}^{\mathtt{s}}$, then collapses this sequence of feature-space points into a single fixed-dimensional vector. The sequence kernel feature-space, $\boldsymbol{\phi}^{\mathtt{seq}}(\boldsymbol{O})$, and kernel, $K^{\mathtt{seq}}(\boldsymbol{O}_i, \boldsymbol{O}_j)$, are thus defined by,

$$\boldsymbol{\phi}^{\mathtt{seq}}(\boldsymbol{O}) = \boldsymbol{\phi}^{\mathtt{s}}(\boldsymbol{\Phi}(\boldsymbol{O})) = \boldsymbol{\phi}^{\mathtt{s}}\Big(\{\boldsymbol{\phi}(\boldsymbol{o}_1), \ldots, \boldsymbol{\phi}(\boldsymbol{o}_T)\}\Big) \tag{3.12}$$

$$K^{\mathtt{seq}}(\boldsymbol{O}_i, \boldsymbol{O}_j) = \boldsymbol{\phi}^{\mathtt{s}}(\boldsymbol{\Phi}(\boldsymbol{O}_i))^{\mathrm{T}} \boldsymbol{\phi}^{\mathtt{s}}(\boldsymbol{\Phi}(\boldsymbol{O}_j)) \tag{3.13}$$

Although the definition of sequence kernels as a combination of two different feature-space mappings appears to provide a wide range of options for defining application-specific dynamic kernels, in practice, this is not the case. This is because the functional form of $\boldsymbol{\phi}^{\mathtt{s}}$ is severely restricted by the need to compute both $\boldsymbol{\phi}^{\mathtt{s}}$ and $K^{\mathtt{seq}}(\boldsymbol{O}_i, \boldsymbol{O}_j)$ using only dot-products of the implicit feature-space $\boldsymbol{\Phi}(\boldsymbol{O})$. One such sequence mapping that meets this criteria is the averaging operation [12],

$$\boldsymbol{\phi}^{\mathtt{seq\text{-}av}}(\boldsymbol{O}) = \boldsymbol{\phi}^{\mathtt{s}}(\boldsymbol{\Phi}(\boldsymbol{O})) = \frac{1}{T} \sum_{t=1}^{T} \boldsymbol{\phi}(\boldsymbol{o}_t) \tag{3.14}$$

Dot-products in this feature-space can be expressed entirely in term of dot-products of the observation feature-spaces, $\boldsymbol{\phi}(\boldsymbol{o}_t)$,

$$\begin{aligned} K^{\mathtt{seq\text{-}av}}(\boldsymbol{O}_i, \boldsymbol{O}_j) &= \boldsymbol{\phi}^{\mathtt{s}}(\boldsymbol{\Phi}(\boldsymbol{O}_i))^{\mathrm{T}} \boldsymbol{\phi}^{\mathtt{s}}(\boldsymbol{\Phi}(\boldsymbol{O}_j)) \\ &= \frac{1}{T_i T_j} \sum_{t=1}^{T_i} \sum_{\tau=1}^{T_j} \boldsymbol{\phi}(\boldsymbol{o}_t)^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{o}_\tau) \end{aligned} \tag{3.15}$$

This allows sequence kernels to be calculated efficiently. Unfortunately, the averaging of observation features is equivalent to introducing an independence assumption between observations. This causes all temporal information to be lost, severely limiting the applicability of sequence kernels for classifying sequence data.

3.3 Summary

In this chapter, dynamic kernels were introduced as a powerful method for mapping sequences of discrete and continuous observations into high-dimensional feature-spaces. Standard distance-based classifiers such as the perceptron algorithm and SVMs can then be trained using these feature-spaces. The dynamic kernels in this chapter were grouped into two categories: those that operate on discrete-observation sequences, and those operate on continuous-observation sequences. The three discrete dynamic kernels discussed

were: the bag-of-words kernel, string kernels and marginalised count kernels. The three continuous kernels discussed were: Fisher kernels, generative kernels and sequence kernels.

4

Augmented Statistical Models

One of the most popular approaches for inferring class decision boundaries from statistical models is Bayes' decision rule. When the correct statistical models and distributions are known, this yields the set of decision boundaries that minimise the expected generalisation error. However, in practice, the correct models are often not known and, instead, approximations are usually selected from a small set of standard generative or discriminative statistical models. Unfortunately, the independence and conditional-independence assumptions embedded within these models may be incorrect, potentially resulting in suboptimal generalisation accuracy.

In recent years, several attempts have been made at extending standard generative models to incorporate additional temporal and spatial dependencies. As discussed in chapter 2, examples of these are: switching linear dynamical systems [107], segmental models [29, 91, 92], buried Markov models [8, 9] and mixed memory models [88]. Unfortunately, many of these models require the latent-variable and observational dependencies to be selected a priori using expert knowledge or empirical evidence. For sequence data, the vast number of possible dependencies can make this selection difficult.

In this chapter, augmented statistical models (augmented models) are introduced as a systematic approach for extending generative models to include additional temporal and spatial dependencies. Given a base generative model—typically an HMM—augmented models are defined using a local exponential expansion of the base model [117]. This expansion is used to define a vector of additional sufficient statistics that enable the base model conditional-independence assumptions to be 'broken', allowing augmented models to represent a wider range of distributions than the original, base, model. The forms of

augmentation, and hence sufficient statistics, discussed in this chapter relate directly to some of the dynamic kernel feature-spaces discussed in chapter 3.

4.1 Local exponential expansion

Augmented statistical models are a systematic and mathematically consistent approach for extending standard generative models to include additional, complex, dependencies. Given a base generative model, $\hat{p}(\boldsymbol{O};\boldsymbol{\lambda})$, augmented models are defined using a member of the exponential family with a reference distribution, $h(\boldsymbol{O}) = \hat{p}(\boldsymbol{O};\boldsymbol{\lambda})$ [117],

$$p(\boldsymbol{O};\boldsymbol{\lambda},\boldsymbol{\alpha}) = \frac{1}{\tau(\boldsymbol{\lambda},\boldsymbol{\alpha})}\hat{p}(\boldsymbol{O};\boldsymbol{\lambda})\exp\left(\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{T}(\boldsymbol{O};\boldsymbol{\lambda})\right) \tag{4.1}$$

where $\boldsymbol{\alpha}$ are known as the augmented parameters and $\boldsymbol{T}(\boldsymbol{O};\boldsymbol{\lambda})$ are sufficient statistics defined from a Taylor-series expansion of the base model. In information-geometric terms, augmented models can be viewed as defining a fibre-bundle that extends the base model statistical manifold.

4.1.1 Taylor series expansion

Let $\mathcal{O} = \{\boldsymbol{O}_1,\ldots,\boldsymbol{O}_n\}$ be a set of independent and identically distributed training examples, each generated according to some unknown data distribution, $p_T(\boldsymbol{O})$. Without prior knowledge of the underlying physical process, these examples are often assumed to originate from a standard generative model—typically a GMM or HMM. Estimating parameters of this model using (for example) the maximum likelihood training criterion, yields a generative distribution that most closely approximates the true data distribution.[1] Throughout this thesis, the generative model and distribution are known as the base model and distribution respectively, and are denoted by probability density functions of the form $\hat{p}(\boldsymbol{O};\boldsymbol{\lambda})$.

Given a base model $\hat{p}(\boldsymbol{O};\boldsymbol{\lambda})$ with ML-estimated parameters $\tilde{\boldsymbol{\lambda}}$, the base model distribution that most closely approximates the true distribution can be written as $\hat{p}(\boldsymbol{O};\tilde{\boldsymbol{\lambda}})$. Next, considering an infinite Taylor-series expansion of the base model about the base distribution [117],

$$\begin{aligned}\ln\hat{p}(\boldsymbol{O};\boldsymbol{\lambda}) = \ln\hat{p}(\boldsymbol{O};\tilde{\boldsymbol{\lambda}}) + (\boldsymbol{\lambda}-\tilde{\boldsymbol{\lambda}})^{\mathrm{T}}\nabla_{\boldsymbol{\lambda}}\ln\hat{p}(\boldsymbol{O};\boldsymbol{\lambda})\Big|_{\tilde{\boldsymbol{\lambda}}} \\ + \frac{1}{2}(\boldsymbol{\lambda}-\tilde{\boldsymbol{\lambda}})^{\mathrm{T}}\left(\nabla_{\boldsymbol{\lambda}}\nabla_{\boldsymbol{\lambda}}^{\mathrm{T}}\ln\hat{p}(\boldsymbol{O};\boldsymbol{\lambda})\right)\Big|_{\tilde{\boldsymbol{\lambda}}}(\boldsymbol{\lambda}-\tilde{\boldsymbol{\lambda}}) + \cdots\end{aligned} \tag{4.2}$$

[1] Throughout this derivation, it is assumed that (for clarity) the base model parameters are estimated using a maximum likelihood training criterion. This is not strictly necessary and, instead, other training criteria can be used. Note that the definition of 'most closely approximates' depends upon the choice of generative model training criterion.

For many base models the infinite expansion in equation (4.2) is valid for all values of $\boldsymbol{\lambda}$ and $\tilde{\boldsymbol{\lambda}}$.[2] In practice, however, the complete Taylor series often cannot be computed explicitly and must, therefore, be truncated. Consider, for example the truncated series that contains only the first-order and second-order derivatives of the base model (note that this assumption is for presentational purposes only: in general series of any length can be used),

$$\begin{aligned}\ln \hat{p}(\boldsymbol{O};\boldsymbol{\lambda}) \approx \ln \hat{p}(\boldsymbol{O};\tilde{\boldsymbol{\lambda}}) + (\boldsymbol{\lambda}-\tilde{\boldsymbol{\lambda}})^{\mathrm{T}} \nabla_{\boldsymbol{\lambda}} \ln \hat{p}(\boldsymbol{O};\boldsymbol{\lambda})\Big|_{\tilde{\boldsymbol{\lambda}}} \\ + \frac{1}{2}(\boldsymbol{\lambda}-\tilde{\boldsymbol{\lambda}})^{\mathrm{T}} \Big(\nabla_{\boldsymbol{\lambda}} \nabla_{\boldsymbol{\lambda}}^{\mathrm{T}} \ln \hat{p}(\boldsymbol{O};\boldsymbol{\lambda}) \Big)\Big|_{\tilde{\boldsymbol{\lambda}}} (\boldsymbol{\lambda}-\tilde{\boldsymbol{\lambda}})\end{aligned} \tag{4.3}$$

Although the truncated series in equation (4.3) is very similar to equation (4.2), the removal of higher-order terms means that good approximations to $\ln \hat{p}(\boldsymbol{O};\boldsymbol{\lambda})$ are only obtained when the model parameters, $\boldsymbol{\lambda}$, are sufficiently close to the base distribution parameters, $\tilde{\boldsymbol{\lambda}}$. This is a standard approximation for Taylor series expansions.

Additionally, since equation (4.3) is an expansion of a statistical model, two further issues must be considered. The first is that, in many cases, the removal of higher-order derivatives results in a model that is unnormalised (the total probability density is no longer equal to one); to rectify this, a normalisation term can be introduced. The second concern is that the dependencies modelled by the truncated series may differ from those in the original model. To illustrate this, consider a truncated Taylor-series expansion of an HMM, defined using only first-order and second-order derivatives of the HMM likelihood. As discussed later in section 4.3.1, first-order derivatives of an HMM depend upon the occupancies of single latent states, $P(\theta_t = j|\boldsymbol{O};\boldsymbol{\lambda})$, whilst second-order derivatives depend upon occupancies of pairs of states, $P(\theta_t = j, \theta_\tau = k|\boldsymbol{O};\boldsymbol{\lambda})$. This means that whilst estimation of the original HMM parameters requires calculation of only the state occupancies, estimation of the approximate model requires calculation of both state and state-pair occupancies. A range of different dependencies can therefore be modelled that could not be represented within the original model.[3]

By introducing a normalisation term, $\tau(\boldsymbol{\lambda}, \tilde{\boldsymbol{\lambda}})$, and re-arranging the truncated Taylor-series, equation (4.3) can be written as a log-linear generative model, $p(\boldsymbol{O};\tilde{\boldsymbol{\lambda}},\boldsymbol{\lambda})$, with

[2] There exist a small number of functions that do not equal the sum of their Taylor series [120]. Base models defined using these function therefore cannot be expanded in this way. Despite this, provided that derivatives of these models are well-defined, augmented statistical models can still be defined using the augmented model definition in equation (4.5).

[3] Note that when infinitely-long expansions are considered, all terms involving joint-posteriors of the latent-variables cancel, leaving a series that exactly mimics the original HMM.

parameters $\boldsymbol{\lambda}$ (note that $\tilde{\boldsymbol{\lambda}}$ are fixed parameters associated with base distribution),

$$\ln p(\boldsymbol{O};\tilde{\boldsymbol{\lambda}},\boldsymbol{\lambda}) = \ln \hat{p}(\boldsymbol{O};\tilde{\boldsymbol{\lambda}}) + \begin{bmatrix} \boldsymbol{\lambda}-\tilde{\boldsymbol{\lambda}} \\ (\boldsymbol{\lambda}-\tilde{\boldsymbol{\lambda}})\otimes(\boldsymbol{\lambda}-\tilde{\boldsymbol{\lambda}}) \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \nabla_{\boldsymbol{\lambda}} \ln \hat{p}(\boldsymbol{O};\boldsymbol{\lambda})\Big|_{\tilde{\boldsymbol{\lambda}}} \\ \frac{1}{2}\mathrm{vec}(\nabla^2_{\boldsymbol{\lambda}} \ln \hat{p}(\boldsymbol{O};\boldsymbol{\lambda}))\Big|_{\tilde{\boldsymbol{\lambda}}} \end{bmatrix} - \ln \tau(\boldsymbol{\lambda},\tilde{\boldsymbol{\lambda}}) \tag{4.4}$$

where $\nabla^2_{\boldsymbol{\lambda}}$ denotes $\nabla_{\boldsymbol{\lambda}}\nabla^{\mathrm{T}}_{\boldsymbol{\lambda}}$, $\otimes$ is the Kronecker product [41], and $\mathrm{vec}(\boldsymbol{A})$ converts the matrix $\boldsymbol{A}$ into a vector. When $\boldsymbol{\lambda}$ is a d-dimensional parameter vector, it is clear that the log-linear model, $p(\boldsymbol{O};\boldsymbol{\lambda})$, calculates weights (the Taylor series coefficients) for the $d(d+1)$ derivatives (the log-linear model sufficient statistics) using only d free parameters. This tying of weights severely restricts the range of distributions that can be modelled.

Augmented statistical models (often referred to simply as augmented models) [117] are defined by generalising the log-linear model in equation (4.4). The first generalisation is to remove the parameter tying of the weights associated with each derivative: this involves replacing the vector of $\boldsymbol{\lambda}-\tilde{\boldsymbol{\lambda}}$ and $(\boldsymbol{\lambda}-\tilde{\boldsymbol{\lambda}})\otimes(\boldsymbol{\lambda}-\tilde{\boldsymbol{\lambda}})$ by a new vector of augmented parameters, $\boldsymbol{\alpha}$, each of which can be varied independently. For second-order models, this increases the number of model parameters from d to $d(d+1)$. The second generalisation used when converting equation (4.4) to an augmented model is to replace the fixed base distribution parameters $\tilde{\boldsymbol{\lambda}}$ with a set of adjustable base model parameters, $\boldsymbol{\lambda}$. Using these generalisations, augmented statistical models can be defined in terms of base model parameters $\boldsymbol{\lambda}$ and augmented model parameters $\boldsymbol{\alpha}$, [117]

$$p(\boldsymbol{O};\boldsymbol{\lambda},\boldsymbol{\alpha}) = \frac{1}{\tau(\boldsymbol{\lambda},\boldsymbol{\alpha})}\hat{p}(\boldsymbol{O};\boldsymbol{\lambda}) \exp\left(\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{T}(\boldsymbol{O};\boldsymbol{\lambda})\right) \tag{4.5}$$

where $\tau(\boldsymbol{\lambda},\boldsymbol{\alpha})$ is a normalisation term. The augmented model in equation (4.5) is a member of the exponential family with the reference distribution given by a distribution of the base model. The model has natural parameters $\boldsymbol{\alpha}$ and sufficient statistics $\boldsymbol{T}(\boldsymbol{O};\boldsymbol{\lambda})$,

$$\boldsymbol{T}(\boldsymbol{O};\boldsymbol{\lambda}) = \begin{bmatrix} \nabla_{\boldsymbol{\lambda}} \ln \hat{p}(\boldsymbol{O};\boldsymbol{\lambda}) \\ \frac{1}{2!}\mathrm{vec}(\nabla^2_{\boldsymbol{\lambda}} \ln \hat{p}(\boldsymbol{O};\boldsymbol{\lambda})) \\ \vdots \\ \frac{1}{\rho!}\mathrm{vec}(\nabla^{\rho}_{\boldsymbol{\lambda}} \ln \hat{p}(\boldsymbol{O};\boldsymbol{\lambda})) \end{bmatrix} \tag{4.6}$$

where ρ denotes the order of the model and $\nabla^{\rho}_{\boldsymbol{\lambda}}$ is a ρ-th order derivative with respect to $\boldsymbol{\lambda}$. Note that since $\boldsymbol{\alpha}$ are the weights associated with the sufficient statistics, as $\boldsymbol{\alpha} \to \mathbf{0}$ the augmented model likelihood tends towards the base distribution likelihood. When $\boldsymbol{\alpha}$ is non-zero, however, the sufficient statistics in equation (4.6) combine with the base model sufficient statistics and allow augmented models to represent a wider range of data distributions than the base model. Examples of these distributions are augmented GMMs (section 4.2.3) and augmented HMMs (section 4.3).

In order to generate valid probability density functions, augmented models must satisfy the two axioms of probability [121]: they must be everywhere positive and the total probability mass must equal one. The first axiom is always satisfied since both the base model likelihood and the exponential extension are, by definition, non-negative. As discussed earlier, the second axiom must be enforced explicitly using the normalisation term, $\tau(\boldsymbol{\lambda}, \boldsymbol{\alpha})$, calculated as the expectation of the unnormalised model over all possible observation sequences,

$$\tau(\boldsymbol{\lambda}, \boldsymbol{\alpha}) = \int \hat{p}(\boldsymbol{O}; \boldsymbol{\lambda}) \exp\left(\boldsymbol{\alpha}^{\mathrm{T}} \boldsymbol{T}(\boldsymbol{O}; \boldsymbol{\lambda})\right) \mathrm{d}\boldsymbol{O} \tag{4.7}$$

In general, this integral has no closed-form solution and must, instead, be approximated (typically using numerical integration). This can make augmented model parameter estimation difficult since many generative model training criteria require explicit calculation of the normalisation term. An alternative form of training that avoids the normalisation calculation is discussed in chapter 6.

A further problem that can occur is that some choices of sufficient statistics (defined by the base model) can result in the exponential term in equation (4.7) having no upper bound, thereby causing the normalisation term $\tau(\boldsymbol{\lambda}, \boldsymbol{\alpha})$ to become infinitely large. For many augmented models, this can be avoided by placing constraints on the augmented parameters. These are often similar to the Gaussian constraint that variances must be positive-definite. Sections 4.2.1 and 4.2.3 contain examples of constraints for augmented Gaussians and augmented GMMs respectively.

4.1.2 Manifolds and fibre-bundles

In the previous section, augmented statistical models were motivated using a Taylor series expansion of a base statistical model about a single distribution of that model. This can be viewed from an information-geometric perspective in terms of fibre-bundles extending a manifold of statistical distributions [3, 101, 117].

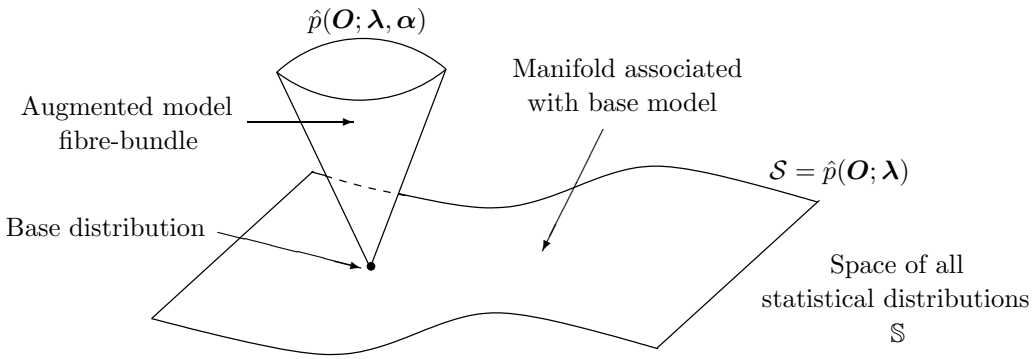


Figure 4.1: Manifold representation of an augmented model

First, consider a base model, $\hat{p}(\boldsymbol{O};\boldsymbol{\lambda})$. In the space of all statistical distributions, $\mathbb{S}$, this defines a statistical manifold, $\mathcal{S}$, parameterised by the base model parameters $\boldsymbol{\lambda}$. Points on the manifold represent specific distributions of the base model. Given a set of training examples, $\mathcal{O} = \{\boldsymbol{O}_1, \ldots, \boldsymbol{O}_n\}$, ML estimation of the base model parameters fixes a point on the manifold (the base distribution) that lies 'closest' to the true data distribution.[4] Augmenting this distribution with the additional sufficient statistics in equation (4.6) creates a fibre-bundle that extends the base manifold [3, 117] (figure 4.1). This fibre-bundle is defined by a combination of the base model parameters, $\boldsymbol{\lambda}$, and the augmented parameters, $\boldsymbol{\alpha}$ (which represent the 'distance' travelled within the fibre-bundle). With the addition of a normalisation term, points within this fibre-bundle represent probability distributions.

For augmented parameters, $\boldsymbol{\alpha} = \mathbf{0}$, augmented model distributions lie on the base model manifold and so yield no additional modelling power. However, as $\boldsymbol{\alpha}$ becomes non-zero, augmented models can represent a wide range of distributions within the fibre-bundle. Since many of these distributions lie away from the manifold, augmented models are able to represent many more distributions than is possible with just the base model. For data sampled from some unknown distribution, augmented models are therefore more likely to provide a better approximation to the true distribution than the original base model. This makes them a powerful and systematic method for extending standard generative statistical models.

4.2 Dependency modelling in augmented models

At this point, it is useful to contrast the dependencies embedded in augmented models, with those of the base models from which they are derived. These dependencies are determined by both the base model and the augmented model sufficient statistics and typically take two forms: independence assumptions and conditional-independence assumptions.

To determine how augmented models are affected by independence assumptions in the base statistical model, consider a simple base generative model $\hat{p}(\boldsymbol{o}_1, \boldsymbol{o}_2; \boldsymbol{\lambda})$, with observations $\boldsymbol{o}_1$ and $\boldsymbol{o}_2$, and model parameters $\boldsymbol{\lambda}$. Introducing the assumption that the observations $\boldsymbol{o}_1$ and $\boldsymbol{o}_2$ are independent allows the base model to be expanded in terms of two separate distributions, $p(\boldsymbol{o}_1;\boldsymbol{\lambda})$ and $p(\boldsymbol{o}_2;\boldsymbol{\lambda})$,

$$\hat{p}(\boldsymbol{o}_1, \boldsymbol{o}_2; \boldsymbol{\lambda}) = p(\boldsymbol{o}_1;\boldsymbol{\lambda})p(\boldsymbol{o}_2;\boldsymbol{\lambda}) \tag{4.8}$$

Next, augmenting this base model using the first-order (for clarity) sufficient statistics in

[4] The notions of 'closeness' and 'closest' are defined by the training criterion.

equation (4.6), a first-order augmented model, $p(\boldsymbol{o}_1, \boldsymbol{o}_2; \boldsymbol{\lambda}, \boldsymbol{\alpha})$, can be defined,

$$\begin{aligned} p(\boldsymbol{o}_1, \boldsymbol{o}_2; \boldsymbol{\lambda}, \boldsymbol{\alpha}) &= \frac{1}{\tau(\boldsymbol{\lambda}, \boldsymbol{\alpha})} \hat{p}(\boldsymbol{o}_1, \boldsymbol{o}_2; \boldsymbol{\lambda}) \exp\left(\boldsymbol{\alpha}^{\mathrm{T}} \nabla_{\boldsymbol{\lambda}} \ln \hat{p}(\boldsymbol{o}_1, \boldsymbol{o}_2; \boldsymbol{\lambda})\right) \\ &= \frac{1}{\tau(\boldsymbol{\lambda}, \boldsymbol{\alpha})} p(\boldsymbol{o}_1; \boldsymbol{\lambda}) p(\boldsymbol{o}_2; \boldsymbol{\lambda}) \exp\left(\boldsymbol{\alpha}^{\mathrm{T}} \nabla_{\boldsymbol{\lambda}} \Big[\ln p(\boldsymbol{o}_1; \boldsymbol{\lambda}) + \ln p(\boldsymbol{o}_2; \boldsymbol{\lambda}) \Big]\right) \\ &= \left(\frac{1}{\tau_1(\boldsymbol{\lambda}, \boldsymbol{\alpha})} p(\boldsymbol{o}_1; \boldsymbol{\lambda}) \exp\left(\boldsymbol{\alpha}^{\mathrm{T}} \nabla_{\boldsymbol{\lambda}} \ln p(\boldsymbol{o}_1; \boldsymbol{\lambda})\right)\right) \times \\ &\qquad \left(\frac{1}{\tau_2(\boldsymbol{\lambda}, \boldsymbol{\alpha})} p(\boldsymbol{o}_2; \boldsymbol{\lambda}) \exp\left(\boldsymbol{\alpha}^{\mathrm{T}} \nabla_{\boldsymbol{\lambda}} \ln p(\boldsymbol{o}_2; \boldsymbol{\lambda})\right)\right) \end{aligned} \tag{4.9}$$

where $\tau_1(\boldsymbol{\lambda}, \boldsymbol{\alpha})$ and $\tau_2(\boldsymbol{\lambda}, \boldsymbol{\alpha})$ are normalisation terms. From equation (4.9), it is clear that first-order augmented models constructed from the base model $\hat{p}(\boldsymbol{o}_1, \boldsymbol{o}_2; \boldsymbol{\lambda})$ cannot overcome the base model assumption that observations are independent. Similar results can also be shown for higher-order augmented models. Augmented models therefore retain base model independence assumptions.

The second form of assumption commonly used in generative models are conditional-independence assumptions. To illustrate the effect of augmenting a base model with conditional-independence assumptions, consider the simple mixture-model,

$$\hat{p}(\boldsymbol{o}; \boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2) = \tfrac{1}{2} p(\boldsymbol{o}; \boldsymbol{\lambda}_1) + \tfrac{1}{2} p(\boldsymbol{o}; \boldsymbol{\lambda}_2) = \sum_{i=1}^{2} \tfrac{1}{2} p(\boldsymbol{o}; \boldsymbol{\lambda}_i) \tag{4.10}$$

where $p(\boldsymbol{o}; \boldsymbol{\lambda}_1)$ and $p(\boldsymbol{o}; \boldsymbol{\lambda}_2)$ are the mixture-component output distributions. In this example, observations are conditionally independent given the mixture component that generated them. Augmenting the base mixture model using the first-order sufficient statistics in equation (4.6), yields the augmented model, $p(\boldsymbol{o}; \boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2)$,

$$\begin{aligned} &p(\boldsymbol{o}; \boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2) \\ &\quad = \frac{1}{\tau(\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2)} \hat{p}(\boldsymbol{o}; \boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2) \exp\left(\begin{bmatrix} \boldsymbol{\alpha}_1 \\ \boldsymbol{\alpha}_2 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \nabla_{\boldsymbol{\lambda}_1} \ln \hat{p}(\boldsymbol{o}; \boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2) \\ \nabla_{\boldsymbol{\lambda}_2} \ln \hat{p}(\boldsymbol{o}; \boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2) \end{bmatrix}\right) \\ &\quad = \frac{1}{\tau(\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2)} \left(\sum_{i=1}^{2} \tfrac{1}{2} p(\boldsymbol{o}; \boldsymbol{\lambda}_i)\right) \exp\left(\tfrac{1}{2} \boldsymbol{\alpha}_1^{\mathrm{T}} \nabla_{\boldsymbol{\lambda}_1} \ln p(\boldsymbol{o}; \boldsymbol{\lambda}_1) + \tfrac{1}{2} \boldsymbol{\alpha}_2^{\mathrm{T}} \nabla_{\boldsymbol{\lambda}_2} \ln p(\boldsymbol{o}; \boldsymbol{\lambda}_2)\right) \\ &\quad = \frac{1}{2\tau(\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2)} \sum_{i=1}^{2} p(\boldsymbol{o}; \boldsymbol{\lambda}_i) \exp\left(\tfrac{1}{2} \boldsymbol{\alpha}_1^{\mathrm{T}} \nabla_{\boldsymbol{\lambda}_1} \ln p(\boldsymbol{o}; \boldsymbol{\lambda}_1)\right) \exp\left(\tfrac{1}{2} \boldsymbol{\alpha}_2^{\mathrm{T}} \nabla_{\boldsymbol{\lambda}_2} \ln p(\boldsymbol{o}; \boldsymbol{\lambda}_2)\right) \end{aligned}$$

Since the augmented model observation likelihood is not a superposition of two independent distributions (unlike the original base mixture-model), the assumption that observations are conditionally independent given the current mixture-component is broken. This allows the augmented model to represent a wider range of data distributions than is possible with the base model. The relaxation and removal of base model conditional-independence assumptions in augmented models is discussed in more detail in the next sections.

4.2.1 Augmented exponential models

Consider a base statistical model selected from the exponential family with parameters $\boldsymbol{\lambda}$ and sufficient statistics $\boldsymbol{T}(\boldsymbol{o})$. When a Euclidean space is assumed, the exponential model likelihood, $\hat{p}(\boldsymbol{O};\boldsymbol{\lambda})$, is given by,

$$\hat{p}(\boldsymbol{O};\boldsymbol{\lambda}) = \prod_{t=1}^{T} \frac{1}{\tau(\boldsymbol{\lambda})} h(\boldsymbol{o}_t) \exp\Big(\boldsymbol{\lambda}^{\mathrm{T}} \boldsymbol{T}(\boldsymbol{o}_t)\Big) \tag{4.11}$$

where $\boldsymbol{O} = \{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_T\}$ is a sequence of independent observations, $h(\boldsymbol{o}_t)$ is the exponential model reference distribution, and $\tau(\boldsymbol{\lambda})$ is a normalisation term. Augmenting this model introduces additional sufficient statistics, defined by the first- and higher-order derivatives of the model log-likelihood with respect to $\boldsymbol{\lambda}$. First-order and second-order derivatives are calculated using the expressions,

$$\nabla_{\boldsymbol{\lambda}} \ln \hat{p}(\boldsymbol{O};\boldsymbol{\lambda}) = \sum_{t=1}^{T} \Big(\boldsymbol{T}(\boldsymbol{o}_t) - \nabla_{\boldsymbol{\lambda}} \ln \tau(\boldsymbol{\lambda})\Big) \tag{4.12}$$

$$\nabla_{\boldsymbol{\lambda}} \nabla_{\boldsymbol{\lambda}}^{\mathrm{T}} \ln \hat{p}(\boldsymbol{O};\boldsymbol{\lambda}) = -\sum_{t=1}^{T} \nabla_{\boldsymbol{\lambda}} \nabla_{\boldsymbol{\lambda}}^{\mathrm{T}} \ln \tau(\boldsymbol{\lambda}) \tag{4.13}$$

From equations (4.12) and (4.13), it is clear that second (and higher-order) derivatives of the base exponential model are dependent on only the base model normalisation term, $\tau(\boldsymbol{\lambda})$. Since, for a given set of base model parameters, this is constant with respect to the observations, contributions from these derivatives can be included in the augmented model normalisation term and need not be considered further. Similarly, equation (4.12) includes the first derivative of the normalisation term; this is also constant and so can included into the augmented model normalisation. Given these simplifications, a ρ-th order augmented exponential model can be written simply as,

$$\begin{aligned} p(\boldsymbol{O};\boldsymbol{\lambda},\boldsymbol{\alpha}) &= \frac{1}{\tau(\boldsymbol{\lambda},\boldsymbol{\alpha})} \Bigg\{ \prod_{t=1}^{T} \frac{1}{\tau(\boldsymbol{\lambda})} h(\boldsymbol{o}_t) \exp\Big(\boldsymbol{\lambda}^{\mathrm{T}} \boldsymbol{T}(\boldsymbol{o}_t)\Big) \Bigg\} \exp\Bigg(\sum_{\tau=1}^{T} \boldsymbol{\alpha}^{\mathrm{T}} \boldsymbol{T}(\boldsymbol{o}_\tau) \Bigg) \\ &= \frac{1}{\tau(\boldsymbol{\lambda},\boldsymbol{\alpha})} \Bigg\{ \prod_{t=1}^{T} \frac{1}{\tau(\boldsymbol{\lambda})} h(\boldsymbol{o}_t) \exp\Big(\boldsymbol{\lambda}^{\mathrm{T}} \boldsymbol{T}(\boldsymbol{o}_t)\Big) \Bigg\} \prod_{t=1}^{T} \exp\Big(\boldsymbol{\alpha}^{\mathrm{T}} \boldsymbol{T}(\boldsymbol{o}_t)\Big) \\ &= \prod_{t=1}^{T} \Bigg\{ \frac{1}{\tau(\boldsymbol{\lambda}+\boldsymbol{\alpha})} h(\boldsymbol{o}_t) \exp\Big((\boldsymbol{\lambda}+\boldsymbol{\alpha})^{\mathrm{T}} \boldsymbol{T}(\boldsymbol{o}_t)\Big) \Bigg\} \end{aligned} \tag{4.14}$$

where $\tau(\boldsymbol{\lambda},\boldsymbol{\alpha})$ denotes the augmented model normalisation and $\tau(\boldsymbol{\lambda})$ denotes the standard exponential base model normalisation term.

At this stage, it is interesting to contrast the dependencies of the augmented model, in equation (4.14), with the base exponential model in equation (4.11). As discussed earlier, the statistics obtained when augmenting a base model are functions of the base model sufficient statistics, causing independence assumptions to be retained. Since there are no

conditional-independence assumptions to break, the augmented model in equation (4.14) has an identical functional form to the original exponential model in equation (4.11), resulting in no additional modelling power.

The apparent simplicity of equation (4.14) can be deceptive since it suggests that valid augmented models are obtained for all values of $\boldsymbol{\lambda}$ and $\boldsymbol{\alpha}$. For many augmented models, however, this is not true and constraints must be introduced. Consider, for example, a Gaussian base distribution with parameters $\boldsymbol{\lambda} = \{\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2\}$ and sufficient statistics, $\boldsymbol{T}(\boldsymbol{o}) = [\boldsymbol{o}^{\mathrm{T}}, \mathrm{vec}(\boldsymbol{o}\boldsymbol{o}^{\mathrm{T}})^{\mathrm{T}}]^{\mathrm{T}}$,

$$p(\boldsymbol{o};\boldsymbol{\lambda}) = \frac{1}{\tau(\boldsymbol{\lambda})} \exp\left(\begin{bmatrix} \boldsymbol{\lambda}_1 \\ \boldsymbol{\lambda}_2 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \boldsymbol{o} \\ \mathrm{vec}(\boldsymbol{o}\boldsymbol{o}^{\mathrm{T}}) \end{bmatrix} \right) \tag{4.15}$$

As a member of the exponential family, this Gaussian distribution can be augmented using equation (4.14). An augmented Gaussian (A-Gaussian) with augmented parameters $\boldsymbol{\alpha} = \{\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2\}$ can therefore be written as,

$$p(\boldsymbol{o};\boldsymbol{\lambda},\boldsymbol{\alpha}) = \frac{1}{\tau(\boldsymbol{\lambda},\boldsymbol{\alpha})} \exp\left(\begin{bmatrix} \boldsymbol{\lambda}_1 + \boldsymbol{\alpha}_1 \\ \boldsymbol{\lambda}_2 + \boldsymbol{\alpha}_2 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \boldsymbol{o} \\ \mathrm{vec}(\boldsymbol{o}\boldsymbol{o}^{\mathrm{T}}) \end{bmatrix} \right) \tag{4.16}$$

For this A-Gaussian to be a valid statistical model, the normalisation term, $\tau(\boldsymbol{\lambda},\boldsymbol{\alpha})$, must have an upper bound. This is true only when all parameters associated with the $\mathrm{vec}(\boldsymbol{o}\boldsymbol{o}^{\mathrm{T}})$ term are negative (the Gaussian variance is positive definite). The following constraints are therefore required,

$$\text{Base model:} \qquad \boldsymbol{\lambda}_2 < \boldsymbol{0} \tag{4.17}$$

$$\text{Augmented model:} \qquad \boldsymbol{\lambda}_2 + \boldsymbol{\alpha}_2 < \boldsymbol{0} \tag{4.18}$$

where $\boldsymbol{a} < \boldsymbol{b}$ denotes the set of inequalities, $a_i < b_i$, $i \in [1, d]$, and $\boldsymbol{a}$ and $\boldsymbol{b}$ are d-dimensional vectors. The first constraint ensures that $p(\boldsymbol{o};\boldsymbol{\lambda})$ is a valid base model, whereas the second ensures that $p(\boldsymbol{o};\boldsymbol{\lambda},\boldsymbol{\alpha})$ is a valid augmented model. No constraints are required for $\boldsymbol{\lambda}_1$ and $\boldsymbol{\alpha}_1$.

4.2.2 Augmented latent-variable models

Latent-variable generative models, one of the standard types of model used for statistical pattern processing, can also be augmented. Consider, for example, an N-state latent-variable base model with a latent-state prior distribution, $P(\boldsymbol{\theta};\boldsymbol{\lambda})$, state-conditional output distributions, $p(\boldsymbol{O}|\boldsymbol{\theta};\boldsymbol{\lambda})$, and parameters $\boldsymbol{\lambda}$,

$$\hat{p}(\boldsymbol{O};\boldsymbol{\lambda}) = \sum_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} P(\boldsymbol{\theta};\boldsymbol{\lambda})\, p(\boldsymbol{O}|\boldsymbol{\theta};\boldsymbol{\lambda}) \tag{4.19}$$

where the latent state sequences, $\boldsymbol{\theta}$, are selected from the set of all possible sequences, $\boldsymbol{\Theta}$. As discussed in section 2.1.2, observations are typically assumed to be conditionally independent given the current state. For the purposes of this discussion, let the state-conditional output distributions, $p(\boldsymbol{o}_t|\theta_t = j;\boldsymbol{\lambda})$, be exponential distributions with parameters $\boldsymbol{\lambda}_j \subset \boldsymbol{\lambda}$ and sufficient statistics $\boldsymbol{T}(\boldsymbol{o}_t;\boldsymbol{\lambda}_j)$.[5] Differentiating the base model log-likelihood with respect to the output distribution parameters, $\boldsymbol{\lambda}_j$, therefore yields a set of augmented model statistics of the form (see appendix A for derivation),

$$\nabla_{\boldsymbol{\lambda}_j} \ln \hat{p}(\boldsymbol{O};\boldsymbol{\lambda}) = \sum_{t=1}^{T} P(\theta_t^j|\boldsymbol{O};\boldsymbol{\lambda})\boldsymbol{T}(\boldsymbol{o}_t;\boldsymbol{\lambda}_j) \tag{4.20}$$

where θ_t^j denotes $\theta_t = j$ and $P(\theta_t^j|\boldsymbol{O};\boldsymbol{\lambda})$ is the posterior probability of the base model being in state j at time t, given $\boldsymbol{O}$. Augmenting the base latent-variable model in equation (4.19) with the derivatives in equation (4.20), yields a first-order augmented model,

$$p(\boldsymbol{O};\boldsymbol{\lambda},\boldsymbol{\alpha}) = \frac{1}{\tau(\boldsymbol{\lambda},\boldsymbol{\alpha})}\hat{p}(\boldsymbol{O};\boldsymbol{\lambda}) \exp\left(\sum_{j=1}^{N}\sum_{t=1}^{T} P(\theta_t^j|\boldsymbol{O};\boldsymbol{\lambda})\boldsymbol{\alpha}_j^{\mathrm{T}}\boldsymbol{T}(\boldsymbol{o}_t;\boldsymbol{\lambda}_j)\right) \tag{4.21}$$

where $\boldsymbol{\alpha} = \{\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_N\}$ are augmented parameters associated with the base model sufficient statistics, $\boldsymbol{T}(\boldsymbol{o}_t;\boldsymbol{\lambda}_j)$, $j \in [1, N]$.

As for the augmented exponential model, it is interesting to compare the dependencies embedded in the augmented model in equation (4.21) with those in the original, base, model. Since the augmented model sufficient statistics in equation (4.20) are a function of only the base model statistics, independence assumptions of the base model are retained by the augmented model. Conversely, conditional-independence of observations given the state is broken since the augmented model sufficient statistics depend on the state posteriors, $P(\theta_t^j|\boldsymbol{O};\boldsymbol{\lambda})$, and so are a function of all observations and states.

More powerful augmented models can be constructed by introducing higher-order derivatives of the base model. Consider, for example, second-order derivatives of the base model log-likelihood with respect to the output distribution parameters, $\boldsymbol{\lambda}_j$ and $\boldsymbol{\lambda}_k$,[6]

$$\begin{aligned}\nabla_{\boldsymbol{\lambda}_j}\nabla_{\boldsymbol{\lambda}_k}^{\mathrm{T}} \ln \hat{p}(\boldsymbol{O};\boldsymbol{\lambda}) = \sum_{t=1}^{T}\sum_{\tau=1}^{T} D(\theta_t^j, \theta_\tau^k|\boldsymbol{O};\boldsymbol{\lambda}) \Big[\nabla_{\boldsymbol{\lambda}_j} \ln p(\boldsymbol{o}_t|\theta_t;\boldsymbol{\lambda})\Big] \Big[\nabla_{\boldsymbol{\lambda}_k} \ln p(\boldsymbol{o}_\tau|\theta_\tau;\boldsymbol{\lambda})\Big]^{\mathrm{T}} \\ + \sum_{t=1}^{T} P(\theta_t^j|\boldsymbol{O};\boldsymbol{\lambda}) \nabla_{\boldsymbol{\lambda}_j}\nabla_{\boldsymbol{\lambda}_k}^{\mathrm{T}} \ln p(\boldsymbol{o}_t|\theta_t;\boldsymbol{\lambda})\end{aligned}$$

where $D(\theta_t^j, \theta_\tau^k|\boldsymbol{O};\boldsymbol{\lambda})$ is a posterior-like term, defined by,

$$D(\theta_t^j, \theta_\tau^k|\boldsymbol{O};\boldsymbol{\lambda}) = P(\theta_t^j, \theta_\tau^k|\boldsymbol{O};\boldsymbol{\lambda}) - P(\theta_t^j|\boldsymbol{O};\boldsymbol{\lambda})P(\theta_\tau^k|\boldsymbol{O};\boldsymbol{\lambda}) \tag{4.22}$$

[5] Note that in general observations may be modelled using any valid distribution. In this section, exponential distributions are used to simplify the presentation.

[6] See appendix A for derivation.

For some base models, the second-order 'posterior', $D(\theta_t^j, \theta_\tau^k | \boldsymbol{O}; \boldsymbol{\lambda})$, allows dependencies between discontiguous observations to be represented even when the base model does not allow such dependencies to be modelled.[7] This is discussed in more detail in section 4.3 (A-HMMs).

With their additional complexity, both in terms of computational cost and memory usage, it is useful to examine the benefits of second- and higher-order augmented models compared to simpler first-order models. Second-order models offer two major benefits. The first is that they depend upon the posterior probabilities (occupancies) of pairs of discontiguous states, allowing second-order augmented models to capture temporal correlations between non-consecutive observations. This allows a wider range of dependencies to be modelled than is possible with first-order augmented models (which depend only upon the occupancies of individual states). The second advantage of higher-order augmented models is that they contain many more sufficient statistics than comparable first order models. For example, given a base model with a d-dimensional parameter vector, $\boldsymbol{\lambda}$, there are d first-order augmented model sufficient statistics, and $d(d+1)/2$ second-order sufficient statistics. Since each sufficient statistic acts as a degree of freedom in the space of all statistical models, the additional statistics of second-order models allows a wider range of distributions to be modelled than is possible with first-order augmented models.

Unfortunately, the additional modelling power of second-order augmented models comes at a significant computational cost. With many more parameters than their first-order equivalents, second-order models require significantly more training data for robust parameter estimation, thereby increasing the computational cost of training. A further cost arises from the complexity of the second- and higher-order augmented model sufficient statistics. Whereas the computational and storage requirements for first-order statistic calculations scale linearly with both sequence length and number of base model parameters, the cost of calculating second-order derivatives varies quadratically in both. This can make calculation and storage of higher-order augmented model sufficient statistics prohibitively expensive. As discussed in chapter 5, however, continuous rational kernels can be used to alleviate some of these difficulties. These use weighted finite-state transducer calculations to avoid explicit calculation and storage of all possible higher-order statistics. Instead, an efficient implicit representation is used that only calculates statistics as they are required.

[7] Note that this is only true when states of the base model are conditionally independent. When states are independent, $D(\theta_t^j, \theta_\tau^k | \boldsymbol{O}; \boldsymbol{\lambda})$ is always zero since the joint posterior for independent states is calculated as the product of the separate posteriors, $P(\theta_t^j, \theta_\tau^k | \boldsymbol{O}; \boldsymbol{\lambda}) = P(\theta_t^j | \boldsymbol{O}; \boldsymbol{\lambda}) P(\theta_\tau^n | \boldsymbol{O}; \boldsymbol{\lambda})$ causing the first and second terms of equation (4.22) to cancel. This result is consistent with the analysis in section 4.2 that augmented models can overcome only conditional-independence assumptions of the base model, not independence assumptions.

4.2.3 Illustrative example: first-order A-GMM

To illustrate the benefits of augmented models, consider a simple example: an augmented GMM. Let $\hat{p}(\boldsymbol{O};\boldsymbol{\lambda})$ be a GMM base model with parameters $\boldsymbol{\lambda} = \{c_m, \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m\}_{m=1}^{M}$. For observations, $\boldsymbol{O} = \{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_T\}$, the base model likelihood and first-order derivatives with respect to $\boldsymbol{\mu}_m$ and $\boldsymbol{\Sigma}_m$ are given by,

$$\hat{p}(\boldsymbol{O};\boldsymbol{\lambda}) = \prod_{t=1}^{T}\sum_{m=1}^{M} c_m \mathcal{N}(\boldsymbol{o}_t;\boldsymbol{\mu}_m,\boldsymbol{\Sigma}_m) \tag{4.23}$$

$$\nabla_{\boldsymbol{\mu}_m} \ln \hat{p}(\boldsymbol{O};\boldsymbol{\lambda}) = \sum_{t=1}^{T} P(\theta_t^m|\boldsymbol{o}_t;\boldsymbol{\lambda})\, \boldsymbol{\Sigma}_m^{-1}(\boldsymbol{o}_t - \boldsymbol{\mu}_m) \tag{4.24}$$

$$\nabla_{\boldsymbol{\Sigma}_m} \ln \hat{p}(\boldsymbol{O};\boldsymbol{\lambda}) = \sum_{t=1}^{T} \frac{P(\theta_t^m|\boldsymbol{o}_t;\boldsymbol{\lambda})}{2}\Big[-\boldsymbol{\Sigma}_m^{-1} + \boldsymbol{\Sigma}_m^{-1}(\boldsymbol{o}_t-\boldsymbol{\mu}_m)(\boldsymbol{o}_t-\boldsymbol{\mu}_m)^{\mathrm{T}}\boldsymbol{\Sigma}_m^{-1}\Big] \tag{4.25}$$

Augmenting the GMM in equation (4.23) with its derivatives with respect to the mixture-component means—equation (4.24)—yields a first-order A-GMM,[8]

$$p(\boldsymbol{O};\boldsymbol{\lambda},\boldsymbol{\alpha}) = \frac{1}{\tau(\boldsymbol{\lambda},\boldsymbol{\alpha})}\left\{\prod_{t=1}^{T}\sum_{m=1}^{M} c_m \mathcal{N}(\boldsymbol{o}_t;\boldsymbol{\mu}_m,\boldsymbol{\Sigma}_m)\right\}\times \exp\Big(\sum_{t=1}^{T}\sum_{n=1}^{M} P(\theta_t^n|\boldsymbol{o}_t;\boldsymbol{\lambda})\,\boldsymbol{\alpha}_{\boldsymbol{\mu}_n}^{\mathrm{T}}\boldsymbol{\Sigma}_n^{-1}(\boldsymbol{o}_t-\boldsymbol{\mu}_n)\Big) \tag{4.26}$$

where $\boldsymbol{\alpha}_{\boldsymbol{\mu}_n}$ denotes the augmented parameters associated with the mean derivatives. As an augmented latent-variable model, conditional-independence assumptions of the base GMM are broken (A-GMM observations are dependent on all mixture-components), whilst independence assumptions are retained.

To examine the effect that augmenting a GMM has on the distributions that can be modelled, consider the simple one-dimensional example in figure 4.2. This compares the performance of two models—a GMM and an A-GMM—on data generated using a symmetric log-normal distribution. Given the data, parameters of a two-component GMM were first estimated using ML estimation. As illustrated in figure 4.2, the resulting distribution modelled the data poorly, achieving an average log-likelihood of -1.59. Next, the GMM was augmented using equation (4.26) to form a two-component A-GMM with parameters estimated using ML estimation (the normalisation term was calculated using numerical integration). Figure 4.2 shows that the resulting A-GMM distribution approximates the data much more closely than the original GMM. This is reflected in an average A-GMM log-likelihood of the data of -1.45, compared to -1.59 for the GMM. Interestingly, the two-component A-GMM also outperformed a four-component ML-estimated GMM (not

[8]Note that, for clarity, the variance derivatives in equation (4.25) are not included in the augmented model.

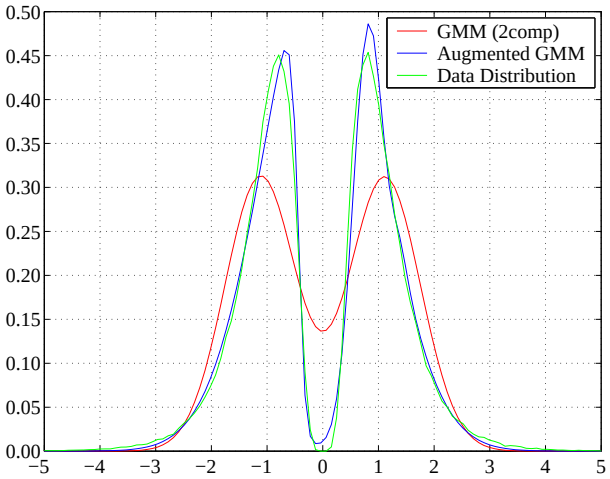


Figure 4.2: Modelling a 'symmetric' log-normal distribution

shown) that achieved a log-likelihood of -1.46, despite the augmented model having fewer parameters.

Table 4.1: Modelling symmetric log-normal data using two-component GMMs and A-GMMs

Classifier	Model parameters							
	c_1	μ_1	σ_1	α_1	c_2	μ_2	σ_2	α_2
GMM	0.50	-1.12	0.41	—	0.50	1.12	0.41	—
A-GMM	0.69	-2.21	1.03	2.20	0.31	0.35	0.07	0.46

Having compared the modelling ability of two-component GMMs and A-GMMs, it is instructive to briefly examine the parameters of each after ML estimation on the symmetric log-normal data. As shown in table 4.1, given a symmetric distribution of data, GMM parameters are symmetric across the components. In contrast, although the A-GMM generates an approximately (though not completely) symmetric distribution, the values of its parameters are highly asymmetric, illustrating the non-linear effect of the mixture-component posteriors on the A-GMM likelihood. It is this effect that gives augmented models much of their additional modelled power.

The simple A-GMM in equation (4.26) is an example of an augmented model that is valid regardless of the augmented parameter values. More complex A-GMMs, however, may require constraints on the augmented parameters to ensure that the normalisation term is bounded and that valid probability distributions are generated. One such model is the A-GMM constructed using the GMM variance derivatives in equation (4.25). Here, the augmented model sufficient statistics include terms of the form $\boldsymbol{o}_t\boldsymbol{o}_t^{\mathrm{T}}$ that interact with the base model mixture-component variances. To ensure that the resulting A-GMM variances

are positive-definite, constraints must be placed upon the augmented parameters; these constraints are similar to the A-Gaussian constraints given in equation (4.18). For a one-dimensional A-GMM with mean and variance derivatives, the constraints are given by,[9]

$$\frac{1}{\sigma_m^2} - \sum_{n=1}^{M} \frac{\alpha_{\sigma_n^2} P(\theta_t^n|o_t;\boldsymbol{\lambda})}{\sigma_n^4} \frac{(o_t-\mu_n)^2}{(o_t-\mu_m)^2} > 0 \qquad m \in [1,M],\ t \in [1,T] \tag{4.27}$$

where μ_m and σ_m^2 are the base GMM means and variances, and $\alpha_{\sigma_n^2}$ are the augmented parameters associated with the variance derivatives. Although more complex than the A-Gaussian constraints in equation (4.18), the A-GMM constraints in equation (4.27) perform a similar function.

4.3 Augmented HMMs (A-HMMs)

Hidden Markov models (HMMs) are one of the most popular statistical models for sequence modelling. As discussed in section 2.1.2, standard HMMs are based upon a number of conditional-independence assumptions that are believed to be incorrect for many tasks. In this section augmented HMMs (A-HMMs) are discussed as a systematic approach for breaking these conditional-independence assumptions, thereby allowing a wide range of additional dependencies to be modelled.

Consider an HMM base model, $\hat{p}(\boldsymbol{O};\boldsymbol{\lambda})$, with M-mixture-component GMM state-conditional output distributions and parameters $\boldsymbol{\lambda}$. The HMM likelihood for a sequence of observations, $\boldsymbol{O} = \{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_T\}$, is thus given by,

$$\hat{p}(\boldsymbol{O};\boldsymbol{\lambda}) = \sum_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \prod_{t=1}^{T} a_{\theta_{t-1}\theta_t} c_{\theta_t} \mathcal{N}(\boldsymbol{o}_t;\boldsymbol{\mu}_{\theta_t},\boldsymbol{\Sigma}_{\theta_t}) \tag{4.28}$$

where $a_{ij} \in \boldsymbol{\lambda}$ are state transition probabilities, $c_{jm} \in \boldsymbol{\lambda}$ are the GMM output distribution mixture-component priors, and $\boldsymbol{\mu}_{\theta_t} \in \boldsymbol{\lambda}$ and $\boldsymbol{\Sigma}_{\theta_t} \in \boldsymbol{\lambda}$ are the means and variances of the Gaussian mixture-component distributions. The latent-states, $\theta_t = \{j, m\}$, include both the HMM state, j, and the output-distribution mixture-component, m. This statistical model contains two conditional-independence assumptions: states are independent given the previous state, and observations are independent given the current state. For many applications, these assumptions are incorrect and may degrade performance. One method of overcoming these assumptions is to augmented the base HMM with additional sufficient statistics defined by the derivatives of the base model. These can take many forms

[9] The A-GMM constraint in (4.27) is a more general form of the A-Gaussian constraints in equation (4.18). For a one-component GMM, $P(\theta_t^n|o_t;\boldsymbol{\lambda}) = 1$, allowing equation (4.27) to be reduced to the form in equation (4.18).

depending on the derivatives used to generate them: first-order sufficient statistics are generated by first-order derivatives of the base model, second-order statistics are generated by second-order derivatives, etc.. In the following sections, first-order and second-order A-HMM sufficient statistics are discussed in detail. The advantages and disadvantages of each are discussed.

4.3.1 First-order statistics

Given an HMM base model, first-order A-HMM sufficient statistics are defined by the first-derivatives of the HMM with respect to its parameters. For an N-state, M-mixture-component HMM, log-likelihood derivatives with respect to the transition probabilities, a_{ij}, and the mixture-component priors, c_{jm}, are given by,[10]

$$\nabla_{a_{ij}} \ln \hat{p}(\boldsymbol{O};\boldsymbol{\lambda}) = \sum_{t=1}^{T} \frac{P(\theta_{t-1}^{i}, \theta_{t}^{j}|\boldsymbol{O};\boldsymbol{\lambda})}{a_{ij}} - P(\theta_{t-1}^{i}|\boldsymbol{O};\boldsymbol{\lambda}) \tag{4.29}$$

$$\nabla_{c_{jm}} \ln \hat{p}(\boldsymbol{O};\boldsymbol{\lambda}) = \sum_{t=1}^{T} \frac{P(\theta_{t}^{jm}|\boldsymbol{O};\boldsymbol{\lambda})}{c_{jm}} - P(\theta_{t}^{j}|\boldsymbol{O};\boldsymbol{\lambda}) \tag{4.30}$$

where θ_t^{jm} and θ_t^j are used to denote $\theta_t = \{j, m\}$ and $\theta_t = j$ respectively. As illustrated by the above equations, derivatives of the base model log-likelihood with respect to the parameters a_{ij} and c_{jm} contain two parts. The first is a summation over either transition or mixture-component occupancies, $P(\theta_{t-1}^i, \theta_t^j|\boldsymbol{O};\boldsymbol{\lambda})$ and $P(\theta_t^{jm}|\boldsymbol{O};\boldsymbol{\lambda})$, respectively. When normalised by a_{ij} and c_{jm}, these measure the ratio of actual-to-expected occupancies of transitions and mixture-components. The second term in each equation arises from the Lagrange multipliers used to maintain the sum-to-one constraints on the transition probabilities and mixture-component priors. These represent a centring of the score-space.

In addition to derivatives with respect to transition probabilities and mixture-component priors, log-likelihood derivatives with respect to the output distribution means and variances can be calculated. They are given by,

$$\nabla_{\boldsymbol{\mu}_{jm}} \ln \hat{p}(\boldsymbol{O};\boldsymbol{\lambda}) = \sum_{t=1}^{T} P(\theta_{t}^{jm}|\boldsymbol{O};\boldsymbol{\lambda})\, \boldsymbol{\Sigma}_{jm}^{-1}(\boldsymbol{o}_t - \boldsymbol{\mu}_{jm}) \tag{4.31}$$

$$\nabla_{\boldsymbol{\Sigma}_{jm}} \ln \hat{p}(\boldsymbol{O};\boldsymbol{\lambda}) = \sum_{t=1}^{T} \frac{P(\theta_{t}^{jm}|\boldsymbol{O};\boldsymbol{\lambda})}{2} \Big[-\boldsymbol{\Sigma}_{jm}^{-1} + \boldsymbol{\Sigma}_{jm}^{-1}(\boldsymbol{o}_t - \boldsymbol{\mu}_{jm})(\boldsymbol{o}_t - \boldsymbol{\mu}_{jm})^{\mathrm{T}} \boldsymbol{\Sigma}_{jm}^{-1} \Big] \tag{4.32}$$

From equations (4.31) and (4.32), it is clear that first-order HMM derivatives with respect to the mixture-component means and variances are calculated as posterior weighted sums of the observations.

Examining equations (4.29)–(4.32), it is clear that HMM derivatives are dependent on the posterior probabilities of all states $\theta_t \in \boldsymbol{\theta}$ associated with an observation sequence $\boldsymbol{O}$.

[10] Derivations are given in appendix A.

Since these posteriors are, themselves, dependent on the whole observation sequence, the derivatives are a function of all observations and latent states. When used as augmented model sufficient statistics, these first-order derivatives break the conditional-independence of observations given the current state. This can be clearly illustrated by an example. Consider the A-HMM generated by augmenting the HMM base model in equation (4.28) with the mean derivatives in equation (4.31),

$$p(\boldsymbol{O};\boldsymbol{\lambda},\boldsymbol{\alpha}) = \frac{1}{\tau(\boldsymbol{\lambda},\boldsymbol{\alpha})} \sum_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \left\{ \prod_{t=1}^{T} a_{\theta_{t-1}\theta_t} c_{\theta_t} \mathcal{N}(\boldsymbol{o}_t;\boldsymbol{\mu}_{\theta_t},\boldsymbol{\Sigma}_{\theta_t}) \right\} \times \exp\Big(\sum_{t=1}^{T}\sum_{k=1}^{N}\sum_{n=1}^{M} P(\theta_t^{kn}|\boldsymbol{O};\boldsymbol{\lambda})\, \boldsymbol{\alpha}_{\boldsymbol{\mu}_{kn}}^{\mathrm{T}} \boldsymbol{\Sigma}_{kn}^{-1}(\boldsymbol{o}_t - \boldsymbol{\mu}_{kn}) \Big) \quad (4.33)$$

where $\boldsymbol{\alpha}_{\boldsymbol{\mu}_{kn}}$ are the augmented parameters. Regrouping the terms allows equation (4.33) to be written similarly to a standard HMM, but with complex state-conditional output distributions, $p(\boldsymbol{o}_t|\theta_t;\boldsymbol{\lambda})$, instead of the standard Gaussian distributions. The new output distributions are defined by,

$$p(\boldsymbol{o}_t|\theta_t;\boldsymbol{\lambda}) = \mathcal{N}(\boldsymbol{o}_t|\theta_t;\boldsymbol{\lambda}) \exp\Big(\sum_{k=1}^{N}\sum_{n=1}^{M} P(\theta_t^{kn}|\boldsymbol{O};\boldsymbol{\lambda})\, \boldsymbol{\alpha}_{\boldsymbol{\mu}_{kn}}^{\mathrm{T}} \boldsymbol{\Sigma}_{kn}^{-1}(\boldsymbol{o}_t - \boldsymbol{\mu}_{kn}) \Big) \quad (4.34)$$

and are dependent on a posterior weighted sum of contributions from all states and mixture-components in the system. Since output distributions are dependent on all base model states, conditional-independent of observations given the current state is broken. With additional parameters and more complex output distributions, A-HMMs can therefore model a much wider range of distributions than normal HMMs (standard HMM distributions can be obtained from an A-HMM by setting $\boldsymbol{\alpha} = \mathbf{0}$). With an appropriate training criterion, A-HMMs can therefore generate better approximations to the true data distribution than HMMs in many cases (or at worst, the same).

4.3.2 Second-order statistics

In addition to the first-order statistics described above, augmented models can be defined using second-derivatives of the base HMM as sufficient statistics. These statistics are more complex than their first-order equivalents and allow many additional dependencies to be modelled. Example second-order derivatives with respect to the HMM state-conditional output distribution mixture-component priors and means are given in equations (4.35)

and (4.36),[11]

$$\nabla_{c_{kn}}\nabla^{\mathrm{T}}_{c_{jm}}\ln\hat{p}(\boldsymbol{O};\boldsymbol{\lambda}) = -\frac{2}{c_{jm}c_{kn}}\sum_{t=1}^{T}\delta_{jk}\delta_{mn}P(\theta_t^{jm}|\boldsymbol{O};\boldsymbol{\lambda}) + \sum_{t=1}^{T}\sum_{\tau=1}^{T}\left\{\frac{D(\theta_t^{jm},\theta_\tau^{kn}|\boldsymbol{O};\boldsymbol{\lambda})}{c_{jm}c_{kn}} - \frac{D(\theta_t^{j},\theta_\tau^{kn}|\boldsymbol{O};\boldsymbol{\lambda})}{c_{kn}} - \frac{D(\theta_t^{jm},\theta_\tau^{k}|\boldsymbol{O};\boldsymbol{\lambda})}{c_{jm}} + D(\theta_t^{j},\theta_\tau^{k}|\boldsymbol{O};\boldsymbol{\lambda})\right\} \tag{4.35}$$

$$\nabla_{\boldsymbol{\mu}_{kn}}\nabla^{\mathrm{T}}_{\boldsymbol{\mu}_{jm}}\ln\hat{p}(\boldsymbol{O};\boldsymbol{\lambda}) = -\sum_{t=1}^{T}\delta_{jk}\delta_{mn}P(\theta_t^{jm}|\boldsymbol{O};\boldsymbol{\lambda})\boldsymbol{\Sigma}_{jm}^{-1} + \sum_{t=1}^{T}\sum_{\tau=1}^{T}D(\theta_t^{jm},\theta_\tau^{kn}|\boldsymbol{O};\boldsymbol{\lambda})\boldsymbol{\Sigma}_{kn}^{-1}(\boldsymbol{o}_\tau-\boldsymbol{\mu}_{kn})(\boldsymbol{o}_t-\boldsymbol{\mu}_{jn})^{\mathrm{T}}\boldsymbol{\Sigma}_{jm}^{-1} \tag{4.36}$$

where δ_{jk} is the Kronecker delta and $D(\theta_t^{jm},\theta_\tau^{kn}|\boldsymbol{O};\boldsymbol{\lambda})$ is a second-order posterior term, defined by equation (4.22). For ease of reference it is reproduced below,

$$D(\theta_t^{jm},\theta_\tau^{kn}|\boldsymbol{O};\boldsymbol{\lambda}) = P(\theta_t^{jm},\theta_\tau^{kn}|\boldsymbol{O};\boldsymbol{\lambda}) - P(\theta_t^{jm}|\boldsymbol{O};\boldsymbol{\lambda})P(\theta_\tau^{jm}|\boldsymbol{O};\boldsymbol{\lambda})$$

Despite their complexity, second-order statistics have a roughly similar form to the first-order statistics discussed previously, with terms consisting of posterior-weighted functions of the observations and HMM variables. Unlike the simpler first-order statistics, however, equations (4.35) and (4.36), are based upon the second-order 'posterior', $D(\theta_t^{jm},\theta_\tau^{kn}|\boldsymbol{O};\boldsymbol{\lambda})$, allowing explicit dependencies between non-contiguous states to be modelled. In a similar fashion to the first-order posteriors discussed previously, the assumption that observations are conditionally independent given the current state is broken.

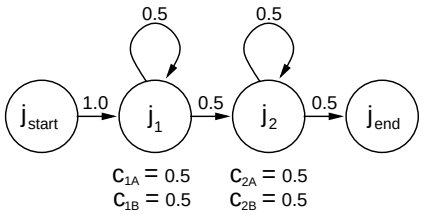
Second-order sufficient statistics offer three main advantages when combined with the first-order statistics discussed in the previous section. The first is that the combined set of first-order and second-order sufficient statistics is much larger than the set of just first-order statistics. Since these statistics act as degrees of freedom in the space of statistical models, second-order A-HMMs can model a much wider range of probability distributions than first-order A-HMMs. The second advantage is that the additional second-order statistics allow a wide selection of long-range dependencies to be modelled explicitly. This enables complex models to be constructed that better represent the original data distribution. The final advantage is that higher-order sufficient statistics contain more complex functions of the observations (for example, second-derivatives with respect to the mixture-component variances results in a fourth-degree tensor of the observations). This allows complex dependencies to be represented.

[11] Note that many of the terms in equation (4.35) arise from the Lagrange multipliers that enforce the sum-to-one constraint.

Unfortunately, second-order statistics are more complex to compute than their first-order relations since they require the calculation of joint posteriors of discontiguous states. In this work, an efficient extension of the standard Forward-Backward algorithm, the double-Forward-Backward algorithm, is used for calculating these posteriors. Details of this algorithm are given in appendix B. Computational complexity for these calculations varies quadratically with sequence length and with the number of HMM states and mixture-components.

4.3.3 Illustrative example: second-order A-HMM

To illustrate the benefits of second-order augmented models, consider a simple artificial example with an alphabet $\{A, B\}$ and training sequences: $AAAA$ (labelled ω_1), $BBBB$ (ω_1), $AABB$ (ω_2) and $BBAA$ (ω_2). Estimating two-state class-conditional discrete HMMs on this data using ML training yields identical models for ω_1 and ω_2,



Classifiers based upon HMMs are unable to utilise the temporal information available in the training sequences, resulting in models that are indistinguishable. The log-likelihood of obtaining any of the training examples from the HMMs is thus -1.11. To discriminate between examples, additional information is therefore required. A simple method of obtaining such information is to augment the HMMs using first-order and second-order derivatives with respect to the mixture-component priors. Example values that these take are shown in table 4.2.

Table 4.2: Selected first- and second-order HMM mixture-component-prior derivatives

Score-space element	Class ω_1		Class ω_2	
	$AAAA$	$BBBB$	$AABB$	$BBAA$
Log. Lik.	-1.11	-1.11	-1.11	-1.11
$\nabla_{c_{2A}}$	0.50	-0.50	0.33	-0.33
$\nabla_{c_{2A}}\nabla_{c_{2A}}^{\mathrm{T}}$	-3.83	0.17	-3.28	-0.61
$\nabla_{c_{2A}}\nabla_{c_{3A}}^{\mathrm{T}}$	-0.17	-0.17	-0.06	-0.06

As shown in table 4.2, first-order derivatives of the base HMMs introduce discriminatory information. Unfortunately, since A-HMMs are linear classifiers in the space of first-order derivatives, they suffer from the XOR problem [30] and so fail to correctly classify the training examples. Second derivatives with respect to the same state/component, $\nabla_{c_{2A}}\nabla_{c_{2A}}^{\mathrm{T}} \ln \hat{p}(\boldsymbol{O}; \boldsymbol{\lambda})$, suffered from the same problem. When second derivatives with respect

to different state/component pairs are considered, however, it becomes trivial to construct a linear decision boundary between the two classes. Second-order A-HMMs defined using cross derivatives can therefore correctly classify all the training examples.

Despite the simplicity of this example, it clearly illustrates the extra modelling ability of second-order A-HMMs, compared to both standard HMMs and first-order A-HMMs. In particular, it demonstrates the benefits of second-order statistics when temporal dependencies are important.

4.4 Kernelised augmented models

Given the computational cost of calculating higher-order augmented models, it is natural to ask whether similar forms of model can be obtained by kernelising the sufficient statistics of simpler models, such as first-order augmented models. The advantage of a kernel approach is that it allows a relatively small number of explicitly-calculated base model derivatives to be mapped into a higher-dimensional implicit feature-space, which can then be used to define the augmented model sufficient statistics. Augmented models defined in this way therefore benefit from large numbers of sufficient statistics without incurring the computational cost of explicit calculation.

Given a set of standard augmented model sufficient statistics, $\boldsymbol{T}(\boldsymbol{O};\boldsymbol{\lambda})$, let $\boldsymbol{f}(\boldsymbol{T}(\boldsymbol{O};\boldsymbol{\lambda});\psi)$ be a function that maps sufficient statistics into a high-dimensional feature-space, where ψ are the parameters of this mapping. The augmented model defined using these new sufficient statistics can be written as,

$$p(\boldsymbol{O};\boldsymbol{\lambda},\boldsymbol{\alpha},\psi) = \frac{1}{\tau(\boldsymbol{\lambda},\boldsymbol{\alpha},\psi)}\hat{p}(\boldsymbol{O};\boldsymbol{\lambda})\exp\Big(\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{f}(\boldsymbol{T}(\boldsymbol{O};\boldsymbol{\lambda});\psi)\Big) \tag{4.37}$$

Unfortunately, calculation of the augmented model likelihoods in equation (4.37) requires explicit calculation of the feature-space, $\boldsymbol{f}(\cdot\,;\psi)$. To avoid this, the augmented parameters, $\boldsymbol{\alpha}$, can be expressed as a linear combination of the sufficient statistics associated with each training example,

$$\boldsymbol{\alpha} = \sum_{i=1}^{n}\beta_i\boldsymbol{f}(\boldsymbol{T}(\boldsymbol{O}_i;\boldsymbol{\lambda});\psi) \tag{4.38}$$

where β_i is the weight associated with the i-th training example. Substituting equation (4.38) into the augmented model in equation (4.37) allows kernelised augmented models to be defined,

$$p(\boldsymbol{O};\boldsymbol{\lambda},\boldsymbol{\beta},\psi) = \frac{1}{\tau(\boldsymbol{\lambda},\boldsymbol{\beta},\psi)}\hat{p}(\boldsymbol{O};\boldsymbol{\lambda})\exp\Big(\sum_{i=1}^{n}\beta_i K\big(\boldsymbol{T}(\boldsymbol{O}_i;\boldsymbol{\lambda}),\boldsymbol{T}(\boldsymbol{O};\boldsymbol{\lambda});\psi\big)\Big) \tag{4.39}$$

where $K(\boldsymbol{T}(\boldsymbol{O}_i;\boldsymbol{\lambda}),\boldsymbol{T}(\boldsymbol{O};\boldsymbol{\lambda});\psi)$ is a kernel function that calculates the feature-space inner-product,

$$K(\boldsymbol{T}(\boldsymbol{O}_i;\boldsymbol{\lambda}),\boldsymbol{T}(\boldsymbol{O};\boldsymbol{\lambda});\psi) = \big\langle\boldsymbol{f}(\boldsymbol{T}(\boldsymbol{O}_i;\boldsymbol{\lambda});\psi),\boldsymbol{f}(\boldsymbol{T}(\boldsymbol{O};\boldsymbol{\lambda});\psi)\big\rangle \tag{4.40}$$

and $\langle \cdot, \cdot \rangle$ denotes the inner-product between two vectors.[12] The advantage of expressing kernelised augmented models in this form is that kernel functions can be defined that combine the the inner-product calculation and the feature-space mappings, $\boldsymbol{f}(\cdot, \psi)$, into a single operation. Examples of such kernels are the homogeneous polynomial kernel and the radial basis function (RBF) kernel. Note that although this thesis only considers the standard kernels discussed in section 2.2.4, a wide range of other kernels can be defined: the only restriction is that kernels must be non-linear symmetric functions that satisfy Mercer's condition [79, 115, 125].[13]

One kernel that is particularly useful for defining kernelised augmented models is the p-th order inhomogeneous polynomial kernel. This maps the explicitly-calculated sufficient statistics, $\boldsymbol{T}(\boldsymbol{O}; \boldsymbol{\lambda})$, into a feature-space containing the statistics raised to a power (up to and including p, including cross-terms). To examine the implicit sufficient statistics that this generates, consider the set of first-order augmented HMM statistics calculated using derivatives with respect to the mixture-component means,

$$\nabla_{\boldsymbol{\mu}_{jm}} \ln \hat{p}(\boldsymbol{O}; \boldsymbol{\lambda}) = \sum_{t=1}^{T} P(\theta_t^{jm} | \boldsymbol{O}; \boldsymbol{\lambda}) \boldsymbol{\Sigma}_{jm}^{-1} (\boldsymbol{o}_t - \boldsymbol{\mu}_{jm}) \tag{4.41}$$

where θ_t^{jm} denotes $\theta_t = \{j, m\}$. Kernelising this score-space using the inhomogeneous polynomial kernel yields a feature-space that contains both the original score-space (4.41) and higher-order powers. Consider, for example, the second-order powers: the score-space squared,

$$\Big(\nabla_{\boldsymbol{\mu}_{jm}} \ln \hat{p}(\boldsymbol{O}; \boldsymbol{\lambda})\Big)\Big(\nabla_{\boldsymbol{\mu}_{kn}} \ln \hat{p}(\boldsymbol{O}; \boldsymbol{\lambda})\Big)^{\mathrm{T}} = \tag{4.42}$$
$$\sum_{t=1}^{T} \sum_{\tau=1}^{T} P(\theta_t^{jm} | \boldsymbol{O}; \boldsymbol{\lambda}) P(\theta_\tau^{kn} | \boldsymbol{O}; \boldsymbol{\lambda}) \boldsymbol{\Sigma}_{jm}^{-1} (\boldsymbol{o}_t - \boldsymbol{\mu}_{jm})(\boldsymbol{o}_\tau - \boldsymbol{\mu}_{kn})^{\mathrm{T}} \boldsymbol{\Sigma}_{kn}^{-1}$$

Comparing the implicit second-order statistics in equation (4.42) with the explicit second-order HMM derivatives in equation (4.36), it is clear that the two approaches yield similar, though slightly different, results. In particular, the implicit statistics in equation (4.42) correspond to the second term of the second-order HMM derivative. Despite the differences between the two forms of statistics, kernelised augmented models allow the number of sufficient statistics to be increased without the computational cost associated with calculating higher-order statistics. Classification performance of kernelised augmented models is discussed in chapter 8.

[12] Note that in this example, the sufficient statistic feature-space metric is assumed to be Euclidean, resulting in a dot-product. For a more detailed discussion of metrics, see chapter 6.

[13] Mercer's condition simply states that the kernel matrix $K(\boldsymbol{o}_i, \boldsymbol{o}_j; \boldsymbol{\lambda})\}_{(i,j)=1}^{n}$ must be positive semi-definite. This ensures that an expansion $\sum_{k=1}^{\infty} a_k z_k(\boldsymbol{o}_i) z_k(\boldsymbol{o}_j)$ exists where $\{z_1(\boldsymbol{o}), \ldots, z_k(\boldsymbol{o}), \ldots\}$ is some unknown feature-space. The kernel thus defines a inner-product in this implicit (possibly incalculable) feature-space.

4.5 Relationships with dynamic kernels

In chapter 3, the use of dynamic kernels for classifying variable-length sequences of data was discussed in detail. Some of these kernels can be related to specific forms of augmented statistical models. Examples discussed in this section are: Fisher kernels [50], marginalised count kernels [123] and sequence kernels [12].

4.5.1 Fisher kernels

Fisher kernels [49,50] were introduced in chapter 3 are a form of dynamic kernel that allows sequences of continuous observations to be mapped into a fixed-dimensional feature-space. Defined using a generative model embedded within a discriminative framework, Fisher kernels combine the generative model's ability to process variable-length sequences with the flexibility and generalisation performance of the discriminative classification framework. Given a training set containing either labelled or unlabelled data, parameters of a single generative model, $\hat{p}(\boldsymbol{O};\boldsymbol{\lambda})$, are first estimated—typically using the ML criterion. Differences in the generative process between examples are then captured by mapping examples into the gradient-space of the generative model,

$$\boldsymbol{\phi}^{\mathrm{F}}(\boldsymbol{O};\boldsymbol{\lambda}) = \nabla_{\boldsymbol{\lambda}} \ln \hat{p}(\boldsymbol{O};\boldsymbol{\lambda}) \tag{4.43}$$

This gradient-space is known as the Fisher score-space. Augmented models are defined almost identically. First a base generative model, $\hat{p}(\boldsymbol{O};\boldsymbol{\lambda})$ is defined. Parameters of this model are typically estimated using the ML or MMI criteria. A set of augmented model sufficient statistics are then defined,

$$\boldsymbol{T}(\boldsymbol{O};\boldsymbol{\lambda}) = \begin{bmatrix} \nabla_{\boldsymbol{\lambda}} \ln \hat{p}(\boldsymbol{O};\boldsymbol{\lambda}) \\ \frac{1}{2!}\mathrm{vec}(\nabla^2_{\boldsymbol{\lambda}} \ln \hat{p}(\boldsymbol{O};\boldsymbol{\lambda})) \\ \vdots \\ \frac{1}{\rho!}\mathrm{vec}(\nabla^{\rho}_{\boldsymbol{\lambda}} \ln \hat{p}(\boldsymbol{O};\boldsymbol{\lambda})) \end{bmatrix} \tag{4.44}$$

Comparing the Fisher score-space in equation (4.43) with the augmented model sufficient statistics in equation (4.44), it is clear that first-order augmented model sufficient statistics and Fisher score-spaces are identical. There is, however, one important theoretical difference between the two approaches. Fisher score-spaces define a feature-space that allows distances between examples to be calculated (allowing distance-based classification approaches to be used). In contrast, augmented models define a set of sufficient statistics that are used to extend the base model: the result is a statistical model that (subject to calculation of the normalisation term) calculates observation sequence likelihoods. Classification is performed using a combination of Bayes' rule and Bayes' decision rule.

4.5.2 Marginalised count kernels

As discussed in section 3.1.3, marginalised kernels [123] are a form of dynamic kernel that combines generative models with more traditional kernel classification techniques. Defined in terms of a general kernel, $K(\{\boldsymbol{O}_i, \boldsymbol{\theta}_i\}, \{\boldsymbol{O}_j, \boldsymbol{\theta}_j\})$, that calculates distances between different observation/latent-state sequences, marginalised kernels are written as,

$$K^{\mathtt{mar}}(\boldsymbol{O}_i, \boldsymbol{O}_j; \boldsymbol{\lambda}) = \sum_{\boldsymbol{\theta}_i \in \boldsymbol{\Theta}} \sum_{\boldsymbol{\theta}_j \in \boldsymbol{\Theta}} P(\boldsymbol{\theta}_i | \boldsymbol{O}_i; \boldsymbol{\lambda}) P(\boldsymbol{\theta}_j | \boldsymbol{O}_j; \boldsymbol{\lambda}) K(\{\boldsymbol{O}_i, \boldsymbol{\theta}_i\}, \{\boldsymbol{O}_j, \boldsymbol{\theta}_j\}) \tag{4.45}$$

where $\boldsymbol{\theta}_i \in \boldsymbol{\Theta}$ is a latent-state sequence associated with the observations, $\boldsymbol{O}_i$. Unfortunately, as discussed in section 3.1.3, the summation over all possible generative model latent-state sequences, $\boldsymbol{\Theta}$, is often intractable and, instead, a count kernel [123] is often used. This simplifies the summation and allows first-order and second-order feature-spaces (corresponding to first-order and second-order count kernels) to be defined,

$$\boldsymbol{\phi}_j^{\mathtt{mc1}}(\boldsymbol{O}; \boldsymbol{\lambda}) = \sum_{t=1}^{T} P(\theta_t^j | \boldsymbol{O}; \boldsymbol{\lambda}) \tag{4.46}$$

$$\boldsymbol{\phi}_{jk}^{\mathtt{mc2}}(\boldsymbol{O}; \boldsymbol{\lambda}) = \sum_{t=2}^{T} P(\theta_{t-1}^j, \theta_t^k | \boldsymbol{O}; \boldsymbol{\lambda}) \tag{4.47}$$

These marginalised count kernel feature-spaces resemble some of the augmented model sufficient statistics defined in sections 4.3.1 and 4.3.2. Consider, for example, the augmented model sufficient statistics defined by the first-order derivatives of an HMM base model with respect to its mixture-component priors, c_{jm}, and transition probabilities, a_{ij},

$$\nabla_{c_{jm}} \ln \hat{p}(\boldsymbol{O}; \boldsymbol{\lambda}) = \sum_{t=1}^{T} \frac{P(\theta_t^{jm} | \boldsymbol{O}; \boldsymbol{\lambda})}{c_{jm}} - P(\theta_t^j | \boldsymbol{O}; \boldsymbol{\lambda}) \tag{4.48}$$

$$\nabla_{a_{ij}} \ln \hat{p}(\boldsymbol{O}; \boldsymbol{\lambda}) = \sum_{t=1}^{T} \frac{P(\theta_{t-1}^i, \theta_t^j | \boldsymbol{O}; \boldsymbol{\lambda})}{a_{ij}} - P(\theta_{t-1}^j | \boldsymbol{O}; \boldsymbol{\lambda}) \tag{4.49}$$

Comparing the feature-spaces in equations (4.46) and (4.47) with the augmented model sufficient statistics in equations (4.48) and (4.49), it is clear that the two are very similar. The biggest difference between the feature-spaces and the sufficient statistics are the additional, $P(\theta_t^j | \boldsymbol{O}; \boldsymbol{\lambda})$, terms in the sufficient statistics. These arise from the sum-to-one constraints on the HMM mixture-component priors and transition probabilities and represent a centring of the score-space. The second difference between the two approaches is the scaling of the latent-state posterior probabilities (by c_{jm} or a_{ij}) in the sufficient statistics. Although these scaling terms alter the dynamic range of the sufficient statistics, when distance-based training of augmented models is used (section 6), their effect is often negated by the use of a maximally-non-committal metric.

Despite their similarity, it is important to note that marginalised count kernels are restricted to modelling linear chain dependencies in sequences of discrete observations,

$\boldsymbol{\phi}^{\mathtt{mc}}_{ij\ldots k}(\boldsymbol{O}) = \sum_{t=1}^{T-n} P(\theta^i_t, \theta^j_{t+1}, \ldots, \theta^k_{t+n}|\boldsymbol{O})$. Conversely, augmented models can model complex dependencies between observations within a sequence of continuous observations (simply by defining higher-order sufficient statistics).

4.5.3 Sequence kernels

A third form of dynamic kernel, that bears many similarities to kernelised augmented models, are sequence kernels [12]. These define feature-spaces using a combination of two mapping functions. First, a standard kernel—such as an inhomogeneous polynomial or RBF kernel—is used to map all observations within a sequence into a high-dimensional implicit representation. The resulting sequence of high-dimensional implicit 'observations' is then converted into a fixed-dimensional feature-space using a second mapping function. Classification is performed in this fixed-dimensional space. Unfortunately, since the second mapping must be calculated using only dot-products of the original observations, the options for converting variable-length sequences into a fixed-dimensional feature-space are very limited. In practice, an averaging operation is used: this assumes that all observations are independent, preventing temporal dependencies from being captured.

Kernelised augmented models use the same types of mapping, but in the opposite order. Sequences of observations are first converted into a fixed-dimensional set of sufficient statistics. Since this is the first mapping, there are no restrictions on the functional form of this mapping, allowing flexibility to capture a wide variety of temporal dependencies within the sufficient statistics. These sufficient statistics are then mapped into a high-dimensional feature-space using a standard kernel mapping function. Temporal dependencies captured by the first mapping function are retained by the second. The advantage of the kernelised augmented model order for feature-space mappings is that it gives the greatest flexibility to the most complex mapping (the sequence to fixed-dimensional mapping), allowing temporal dependencies to be captured and used for classification. The overall result is that kernelised augmented models can capture and utilise complex temporal dependencies, whereas sequence kernels cannot.

4.6 Summary

In this chapter, generative augmented models were introduced as a systematic method for extending standard, base, statistical models using a set of additional sufficient statistics. These statistics are defined using a local exponential expansion of the base model and allow many complex dependencies to be modelled. In particular, when latent-variable base models are used, it was shown that the standard conditional-independence assumption—that observations are independent given the current state—is broken, allowing a wide range of additional distributions to be modelled. Detailed examples of augmented Gaussians,

GMMs and HMMs were presented to illustrate the benefits of augmented models. In each case, the nature of the additional statistics was discussed, as were methods of calculation. Kernelised augmented models were then introduced as a powerful method of increasing the number of augmented model sufficient statistics without incurring a proportional cost. Finally, augmented models and their sufficient statistics were compared to a number of popular dynamic kernels.

5

Continuous Rational Kernels

Rational kernels are a powerful form of discrete-observation dynamic kernel that allow distances between sequences of discrete observations to be calculated. Unlike the dynamic kernels discussed in chapter 3, rational kernels do not prescribe the kernel feature-spaces that are generated. Instead, they provide a general framework, based upon weighted finite-state transducers, that allows application-specific feature-spaces to be defined and calculated.

Unfortunately, rational kernels only handle sequences of discrete observations. To rectify this, this chapter proposes continuous rational kernels. These use a combination of standard rational kernels and latent-variable generative models to calculate distances between sequences of continuous observations. In addition to being a powerful form of dynamic kernel, continuous rational kernels can also be used to generate the additional sufficient statistics required by augmented models. For this task, continuous rational kernels offer significant advantages over the statistic-dependent algorithms discussed in chapter 4. In particular, the Forward-Backward and Double-Forward-Backward algorithms used for calculating first- and second-order augmented model sufficient statistics can be replaced by standardised algorithms that are calculated using only finite-state transducers. These transducers are, in general, easier to define than the dynamic programming algorithms that they replace.

This chapter is structured as follows. First an introduction to finite-state transducers and standard, discrete, rational kernels is given. Continuous rational kernels are then presented as an attractive method for classifying sequences of continuous observations. The remainder of the chapter is then dedicated to discussing how continuous rational kernels

can be used to calculate first- and second-order augmented HMM sufficient statistics.

5.1 Weighted finite-state transducers

In this section, weighted finite-state acceptors and transducers [80] are introduced as a natural approach for representing and manipulating sequences of discrete observations. Finite-state transducers, discussed first, are a flexible approach for representing sequence transformations for sequences of discrete observations. Acceptors, a special type of transducer, allow sequences and lattices of discrete observations to be represented within the finite-state transducer framework. Finally, a number of standard transducer and acceptor operations are discussed.

5.1.1 Transducers

Finite-state transducers consist of a set of states, labelled from one to N, that are joined by directed arcs. Transducers have a single start state—labelled as state one[1]—and one or more end states, denoted by a double circle. States are connected using directed arcs labelled in the form $\delta : \gamma$ where $\delta \in \{\Sigma \cup \epsilon\}$ and $\gamma \in \{\Delta \cup \epsilon\}$ are input and output symbols, selected from the input and output alphabets, Σ and Δ, respectively. The null symbol, ϵ, denotes a transition that either does not consume a symbol from the input sequence or does not output a symbol. In later sections, to simplify transducer definitions, transitions are grouped, with $\Delta : \Delta$ and $\Delta : \epsilon$ representing the sets of transitions $\{\delta : \delta, \forall \delta \in \Delta\}$ and $\{\delta : \epsilon, \forall \delta \in \Delta\}$ respectively. Here, input and output alphabets are assumed to be identical.

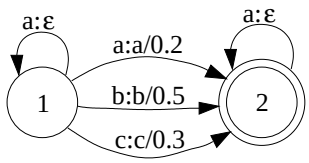


Figure 5.1: A finite-state transducer

With a combination of states and arcs, paths through transducers can be defined. Paths start at state one and terminate in one of the designated end-states. Each path represents a transformation of a single input sequence into a new, often different, output sequence. For non-cyclic transducers, where states are only visited once per path, the maximum length of an input sequence is determined by the number of arcs in the longest path. When self-transitions and other loops are introduced, arbitrary-length input or output sequences are possible.

[1]Since state numbering is arbitrary, transducer states can always be renumbered such that the start state is labelled as state one.

Table 5.1: Example transducer transformations

Input sequence	Output sequence / weight
a* a a*	a / 0.2
a* b a*	b / 0.5
a* c a*	c / 0.3
all others	– / –

where 'a*' denotes zero or more occurrences of the symbol 'a'

To illustrate some of the sequence transformations that are possible with transducers, consider the simple transducer in figure 5.1. This transforms input sequences of the form: zero or more a's, followed by an a, b or c, followed by zero or more a's, into an output sequence that contains a single character: a, b or c. These transformations are summarised in table 5.1. The use of self-transitions in the transducer in figure 5.1 means that the transducer has a many-to-one mapping, with multiple input sequences, such as $\{b\}$ and $\{a, a, b, a\}$, being mapped to the same output sequence, $\{b\}$. Other mappings, such as one-to-many and many-to-many, are also possible.

In addition to input and output symbols, each transducer arc may be assigned a scalar weight, $w \in \mathbb{K}$, selected from the set of valid weights, $\mathbb{K}$. These weights are typically specified using the notation $\delta{:}\gamma/w$ where δ, γ and w are the input symbol, output symbol and weight respectively. In the absence of a specified weight, arcs are assigned a default weight of $\bar{1}$ where $\bar{1}$ is defined by the transducer semiring. This semiring defines the minimum set of operations required for propagating arc weights through a transducer, and is written as $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$ [81]. The symbols $\oplus$ and $\otimes$ denote operations of addition and multiplication respectively. The zero value, $\bar{0}$, and the identity, $\bar{1}$, are selected to satisfy the identity axioms of addition, $x \oplus \bar{0} = x$, and multiplication, $x \otimes \bar{1} = x$. Some popular semirings, defined in table 5.2, are the real, log and tropical semirings.

Table 5.2: Popular semirings for transducers

Semiring	$\mathbb{K}$	$\oplus$	$\otimes$	$\bar{0}$	$\bar{1}$
Real	$\mathbb{R}^+$	$+$	$\times$	0	1
Log	$\mathbb{R} \cup \{\pm\infty\}$	$\oplus_{\log}$	$+$	$+\infty$	0
Tropical	$\mathbb{R} \cup \{\pm\infty\}$	min	$+$	$+\infty$	0

where $x \oplus_{\log} y = -\log(\exp(-x) + \exp(-y))$

Note that the log and tropical semirings are typically defined using negative log weights instead of the more conventional (in speech) log weights. A weight of zero is therefore specified in the log semiring as $+\infty$.

When weights are manipulated in the real semiring, they behave as probabilities and so are multiplied along paths and summed when paths merge. Unfortunately, although the real semiring is a very natural basis for defining weights, many practical tasks, such as speech, utilise small weights that underflow the limited floating-point precision of many

computers. To avoid this, weights are often specified using the log semiring, an isomorphism of the real semiring. For clarity, all transducers in this thesis are defined in the real semiring.

5.1.2 Acceptors

Similarly to transducers, acceptors are defined using a collection of finite states, joined by a set of arcs. In contrast to transducers, however, acceptor arcs are labelled with only a single symbol, δ. Acceptors therefore provide a natural representation for both sequences and lattices of discrete observations. Consider, for example, the sequence $\{a, a, b, a\}$ discussed previously. Figure 5.2 demonstrates how this can be written as an acceptor.

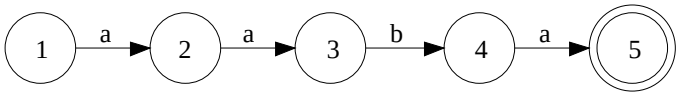


Figure 5.2: An acceptor for the input sequence $\{a, a, b, a\}$

As for transducers, arcs are each assigned a weight, $w \in \mathbb{K}$ (defaulting to $\bar{1}$), specified using the labelling convention, δ/w. With only a single input/output symbol, acceptors acts as either inputs or outputs depending upon the operation being performed.

5.1.3 Operators

One of the main advantages of using acceptors and transducers to represent sequences and sequence transformations is that many sequence operations can be performed using a small number of standardised and efficient algorithms. Examples of the transducer operations used in this thesis are: inversion, composition [82] and shortest-distance [81] (transducer weight).

Inversion is the simplest of the three operations. The inverse of a transducer $\boldsymbol{U}$ is calculated by swapping the input and output symbols on all transducer arcs [80]. The inverse transducer, $\boldsymbol{U}^{-1}$, therefore 'undoes' the transformation of the original transducer, $\boldsymbol{U}$.

Composition is the process of chaining together transducers so that the output of the first is used as the input for the second. This allows complex transformations to be constructed from a number of simpler transducer transformations. The composition of two transducers, $\boldsymbol{U}_1$ and $\boldsymbol{U}_2$, is defined as the transducer that, given any input sequence, generates an equivalent output sequence to that generated from passing the input through $\boldsymbol{U}_1$ and the output of that through $\boldsymbol{U}_2$. Mathematically, transducer composition is written as $\boldsymbol{U}_1 \circ \boldsymbol{U}_2$.

Shortest-distance is the final operator in this section. Given an acceptor $\boldsymbol{A}$ or transducer $\boldsymbol{U}$, the shortest distance operator calculates the sum of the weights of all possible paths through $\boldsymbol{A}$ or $\boldsymbol{U}$ [81]. The shortest distance of an acceptor or transducer is typically denoted by $[\![\boldsymbol{A}]\!]$ or $[\![\boldsymbol{U}]\!]$.

Using a combination of these transducer operators, rational kernels can be defined. These provide a powerful framework for calculating 'distances' between variable-length sequences of discrete observations using user-defined metrics.

5.2 Rational kernels

Using the weighted finite-state acceptors and transducers discussed in the previous section, rational kernels [17, 18] can be defined. These provide a powerful framework for defining and calculating high-dimensional kernel feature-spaces from sequences of discrete observations using only simple transducer operations. For data that can be represented using finite-state transducers, rational kernels offer two main advantages over other dynamic kernels. The first is that different kernel feature-spaces can be defined simply by selecting different transducers. Distances (defined by the dot-product in the feature-space) between observation sequences are then calculated using the standard algorithms of transducer inversion, composition and shortest-distance. The second advantage is that the feature-space need never be calculated explicitly. This is especially important for applications with high-dimensional, sparse feature-vectors. In these situations, rational kernels avoid the unnecessary calculation of feature-space elements with zero occupancy (i.e. those associated with symbols that do not appear in the input) since transducer composition is a lazy operation: paths are only instantiated as they are required. Paths for non-existent symbols are therefore never created and so contribute no computational cost.

The in-depth operation of rational kernels is best explained using a simple example. Let $\boldsymbol{O} = \{o_1, \ldots, o_T\}$ be a sequence of length T, with discrete observations o_t selected from an input alphabet Δ. Since sequence lengths may vary, and discriminative classification algorithms such as SVMs require fixed-dimensional feature-vectors, a mapping from $\boldsymbol{O}$ to a fixed-dimensional feature-space, $\boldsymbol{\phi}(\boldsymbol{O})$, is desired. One such mapping is the feature-space defined by the counts of the occurrences of each symbol in the input alphabet. This feature-space is known as the bag-of-words kernel [115], the count kernel, or the unigram kernel. In vector notation, the feature-space is written as,

$$\boldsymbol{\phi}(\boldsymbol{O}) = \begin{bmatrix} f(a|\boldsymbol{O}) \\ f(b|\boldsymbol{O}) \\ \vdots \end{bmatrix} \tag{5.1}$$

where a and b are elements of the input alphabet and $f(\delta|\boldsymbol{O})$ denotes the number of occurrences of the symbol δ in the sequence $\boldsymbol{O}$. The distance between examples in the

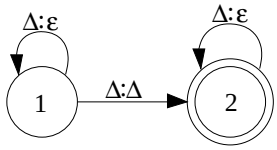
feature-space is given by the feature-space dot-product. This is known as the unigram kernel. For two sequences, $\boldsymbol{O}_i$ and $\boldsymbol{O}_j$, this distance is written as,

$$\begin{aligned} K(\boldsymbol{O}_i, \boldsymbol{O}_j) &= \boldsymbol{\phi}(\boldsymbol{O}_i)^T \boldsymbol{\phi}(\boldsymbol{O}_j) \\ &= \sum_{\delta \in \Delta} f(\delta|\boldsymbol{O}_i) f(\delta|\boldsymbol{O}_j) \end{aligned} \tag{5.2}$$

Calculation of the kernel in equation (5.2) requires that the feature-spaces for each example are calculated explicitly. The computational cost of the kernel calculation is therefore proportional to the dimensionality of the feature-space (the number of elements in the input alphabet) regardless of the data. When feature-spaces are sparse (such as when large alphabets are used), the above calculations can be inefficient since they require initialisation and calculation of a large number of empty features. Instead, an approach, based upon the finite-state transducer framework in section 5.1, can be used.

Using the above feature-space, consider two observation sequences $\boldsymbol{O}_i$ and $\boldsymbol{O}_j$ with lengths T_i and T_j respectively. Within the finite-state transducer framework these are represented as acceptors, denoted $\boldsymbol{A}_i$ and $\boldsymbol{A}_j$ respectively, with a structure similar to that of figure 5.2. The observation sequences can be mapped into the unigram feature-space by performing transducer composition of the acceptors, $\boldsymbol{A}_i$ and $\boldsymbol{A}_j$, with the unigram feature-space, $\boldsymbol{U}$,

$\boldsymbol{U}$ =



The composition of $\boldsymbol{A}_i$ with $\boldsymbol{U}$ yields an output acceptor, $\boldsymbol{A}_i \circ \boldsymbol{U}$. This contains T_i distinct paths, each of which has a weight of 1.0. Each path t, $t \in [1, T_i]$, contains T_i state transitions (arcs), each with an input symbol corresponding to o_t. All output symbols in the t-th path are null (ϵ) except for the t-th transition, which has an output symbol o_t. The number of distinct paths with an output label, δ, is thus equal to the number of occurrences of δ in the input sequence, $\boldsymbol{A}_i$. Since the weight of each path is 1.0, the total weight of paths with an output symbol δ is $f(\delta|\boldsymbol{O})$—the unigram count.

Although lattice-based unigram features offer a compact representation of the feature-space, they can be difficult to manipulate directly. Fortunately, kernel-based classification is based upon the feature-space dot-product which can be calculated within the transducer framework. This dot-product is calculated using a combination of transducer inversion, composition and shortest-distance, and is known as the unigram rational kernel [18]. The kernel is written as,

$$\begin{aligned} K(\boldsymbol{O}_i, \boldsymbol{O}_j) &= [\![(\boldsymbol{A}_i \circ \boldsymbol{U}) \circ (\boldsymbol{A}_j \circ \boldsymbol{U})^{-1}]\!] \\ &= [\![\boldsymbol{A}_i \circ \boldsymbol{U} \circ \boldsymbol{U}^{-1} \circ \boldsymbol{A}_j]\!] \end{aligned} \tag{5.3}$$

where $[\![\cdot]\!]$ denotes transducer shortest-distance (section 5.1.3). The inversion in the first line ensures that the feature-space elements of $\boldsymbol{O}_i$ match with the corresponding feature-space elements of $\boldsymbol{O}_j$: the acceptor $(\boldsymbol{A}_i \circ \boldsymbol{U})$ generates outputs symbols whereas the acceptor $(\boldsymbol{A}_j \circ \boldsymbol{U})^{-1}$ expects input symbols. When the two feature-space acceptors are composed together, the input and output symbols are matched, resulting in a single acceptor with paths representing products of feature-space elements. Shortest-distance over this acceptor performs a final summation over all dimensions, yielding the feature-space dot-product.

When variable-length observation sequences are considered, the value of the unigram kernel varies with both sequence similarity and length (penalising shorter sequences). To remove this bias, sequence-length normalisation can be performed. This converts the frequency counts in equation (5.1) into posterior probabilities by scaling the feature-space by $1/T$, resulting in a sequence-length normalised unigram kernel,

$$K(\boldsymbol{O}_i, \boldsymbol{O}_j) = \frac{1}{T_i T_j} [[\boldsymbol{A}_i \circ \boldsymbol{U} \circ \boldsymbol{U}^{-1} \circ \boldsymbol{A}_j]] \tag{5.4}$$

where T_i and T_j denote the sequence lengths for the observations $\boldsymbol{O}_i$ and $\boldsymbol{O}_j$ respectively.

Although this section has concentrated on the unigram feature-space, the generality of rational kernels allows many other feature-spaces to be constructed, for example, string kernels (section 3.1.2) and marginalised count kernels (section 3.1.3). Rational kernels have two major advantages over these other forms of kernel. The first is that regardless of the feature-space being used, the basic algorithms for computing the feature-space dot-product remain unchanged. This makes implementation of new feature-spaces significantly simpler than with other methods. The second advantage is that, in addition to basic sequences of observations, rational kernels can calculate distances between lattices. This is useful when there is uncertainty in the observation labels since lattices allows multiple label sequences to be represented. Rational kernels can therefore utilise all available information about these labelling uncertainties, potentially improving classification performance.

Despite these advantages, rational kernels have one major disadvantage: they require observations to have discrete labels. Many practical tasks, however, are based upon continuous observations. In the next section, a new form of rational kernel, the continuous rational kernel, is proposed. This combines the benefits of the generative models in chapter 2 with benefits of the rational kernels discussed above, to generate a flexible framework for calculating distances between sequences of continuous observations.

5.3 Continuous rational kernels

Continuous rational kernels, proposed in this section, are one approach for extending rational kernels to sequences of continuous observations. Defined using a combination

of generative models and transducer-based rational kernels, continuous rational kernels allow the benefits of rational kernels (standardised algorithms, ease of selecting different feature-spaces) to be applied to tasks with sequences of continuous observations.

Similarly to Fisher kernels, a latent-variable base model is first defined; parameters of this model are typically estimated using ML estimation.[2] Given this model, Viterbi or Forward-Backward alignment is then used to calculate the latent-states associated with each observation. Viterbi alignment yields the single most likely state sequence, whereas Forward-Backward alignment generates a lattice containing all possible state alignments weighted by their likelihoods. By recording the state alignments, an alignment acceptor, $\boldsymbol{L}$, is produced. This compactly represents the observation sequences as a sequence or lattice of latent states.

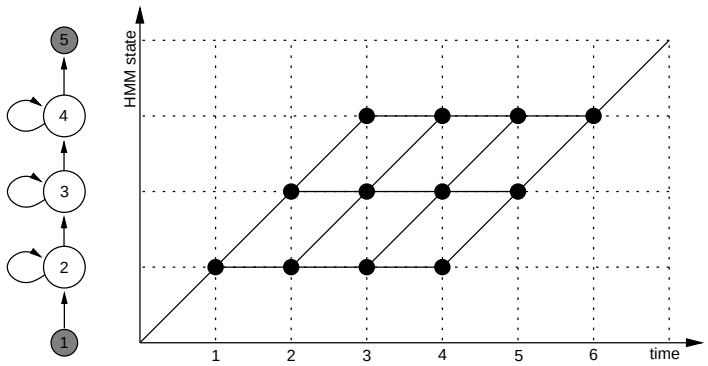


Figure 5.3: An example trellis for a three-state left-to-right HMM

When defining continuous rational kernels, Forward-Backward (FB) alignment is most useful since it retains more information about the original observations than Viterbi alignment. When HMM base models are used, lattice acceptors generated using FB alignment have an identical structure to the standard HMM 'trellis diagram' [13]—a simple trellis for a three-state left-to-right HMM is shown in figure 5.3. Alignment acceptors generated using these trellises have a set of sequentially numbered nodes[3] joined by arcs labelled with likelihoods and the HMM state/mixture-component associated with their target node. The weight of a particular path through the alignment acceptor is equal to the posterior probability of that path given the observations.

Given two alignment acceptors, $\boldsymbol{L}_i$ and $\boldsymbol{L}_j$, continuous rational kernels are defined as the distance between the acceptors. This distance is calculated using the standard (discrete) rational kernel framework discussed in section 5.2. Putting the pieces together,

[2]Note that continuous rational kernels can be extended to multiple base models in a similar fashion to that used when Fisher kernels are extended to form generative kernels. For clarity, this section will assume that a single base model is used for all sequences.

[3]Although node labels are often presented in transducer diagrams, they are a notational convenience, and are not used during calculation. For automatically generated acceptors, node labels are typically generated sequentially.

length-normalised continuous rational kernels are written as,

$$\begin{aligned} K(\boldsymbol{O}_i, \boldsymbol{O}_j) &= [\![(\boldsymbol{L}_i \circ \boldsymbol{U}) \circ (\boldsymbol{L}_j \circ \boldsymbol{U})^{-1}]\!] \\ &= [\![\boldsymbol{L}_i \circ \boldsymbol{U} \circ \boldsymbol{U}^{-1} \boldsymbol{L}_j]\!] \end{aligned} \tag{5.5}$$

where $\boldsymbol{U}$ is a transducer that maps the alignment acceptors $\boldsymbol{L}_i$ and $\boldsymbol{L}_j$ into a high-dimensional feature-space where distances can be calculated. Similarly to rational kernels, different feature-spaces can be obtained by changing $\boldsymbol{U}$. Additionally, by varying the latent-variable base model, different alignment acceptors can be generated which, in turn, alter the feature-space. Continuous rational kernels are therefore a highly flexible form of continuous dynamic kernel.

In the following sections, continuous rational kernels are applied to the task of calculating the first-order and second-order augmented model sufficient statistics discussed in chapter 4. By expressing the augmented model statistic calculations within a transducer framework, the derivative-specific algorithms discussed in sections 4.3.1 and 4.3.2 can be replaced by a single, standardised, transducer-based algorithm. Discussion in the next sections will initially focus upon the definition and calculation of base model derivatives for GMMs and HMMs. Then, given these derivatives, a method of combining derivative feature-spaces and kernels to form Fisher and generative score-spaces and kernels is presented. These kernels can either be substituted directly in a kernel-based classifier—for example a Perceptron or SVM—or used to train a pair of augmented models (see chapter 6 for a full discussion).

5.3.1 First-order GMM and HMM derivatives

In the next sections, techniques for calculating first-order derivatives of GMM and HMM base models within the continuous rational kernel framework are discussed. As discussed in chapter 4, these derivatives can be used to define augmented model sufficient statistics. Derivatives are presented in the form of score-spaces and kernels (many of which are based on unigram transducers) that can be substituted directly into the maximum-margin augmented model training algorithms discussed in the next chapter.

Mixture-component probability kernels

One of the simplest types of augmented model sufficient statistic is that defined by the first-order derivatives of an M-component GMM base model, $\hat{p}(\boldsymbol{O}; \boldsymbol{\lambda})$, with respect to its mixture-component priors, $c_m \in \boldsymbol{\lambda}$. Written as a score-space, $\boldsymbol{\phi}^{\mathsf{c}}(\boldsymbol{O}; \boldsymbol{\lambda})$, these derivatives are given by,

$$\phi^{\mathsf{c}}_m(\boldsymbol{O}; \boldsymbol{\lambda}) = \nabla_{c_m} \ln \hat{p}(\boldsymbol{O}; \boldsymbol{\lambda}) = \sum_{t=1}^{T} \left[\frac{P(m|\boldsymbol{o}_t; \boldsymbol{\lambda})}{c_m} - 1 \right] \tag{5.6}$$

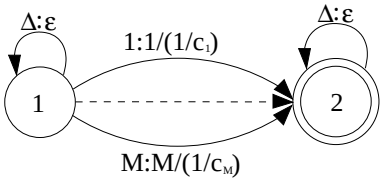
where $\boldsymbol{\phi}^{\mathtt{c}}_m(\boldsymbol{O};\boldsymbol{\lambda})$ denotes the m-th element of the score-space (the derivative with respect to c_m). Given this derivative/score-space, it is interesting to examine how equation (5.6) can be calculated within the continuous rational kernel framework.

First, a latent-state acceptor, $\boldsymbol{L}$, is constructed using Forward-Backward alignment of the base GMM using the continuous observation sequence, $\boldsymbol{O}$. This acceptor has a lattice structure containing all possible paths through the GMM. Each path is labelled with the sequence of mixture-components that generated it, and is assigned a weight equal to the probability of obtaining that sequence. Applying the unigram transducer in section 5.2 to this lattice yields a feature-space with elements given by the posterior probabilities of the GMM mixture-components,

$$\boldsymbol{\phi}^{\mathtt{uni}}(\boldsymbol{O};\boldsymbol{\lambda}) = \boldsymbol{L} \circ \boldsymbol{U} = \sum_{t=1}^{T} \begin{bmatrix} P(m_t = 1|\boldsymbol{O};\boldsymbol{\lambda}) \\ P(m_t = 2|\boldsymbol{O};\boldsymbol{\lambda}) \\ \cdots \\ P(m_t = M|\boldsymbol{O};\boldsymbol{\lambda}) \end{bmatrix} \tag{5.7}$$

Examining the similarities between equations (5.6) and (5.7), it is clear that the first-order derivatives of a GMM with respect to its mixture-components can be calculated using a unigram transducer with arcs scaled by the mixture-component priors, c_m. This scaled transducer is known as $\boldsymbol{U}^{/\mathtt{c}}_{\mathtt{gmm}}$, and is written as,[4]

$\boldsymbol{U}^{/\mathtt{c}}_{\mathtt{gmm}}$ =



Substituting $\boldsymbol{U}^{/\mathtt{c}}_{\mathtt{gmm}}$ into the general continuous rational kernel framework in equation (5.5), yields a continuous rational kernel that calculates distances using a first-order GMM mixture-component prior score-space,

$$K^{\mathtt{c}}(\boldsymbol{O}_i, \boldsymbol{O}_j;\boldsymbol{\lambda}) = \frac{1}{T_i T_j} [\![\boldsymbol{L}_i \circ \boldsymbol{U}^{/\mathtt{c}}_{\mathtt{gmm}} \circ \boldsymbol{U}^{/\mathtt{c}^{-1}}_{\mathtt{gmm}} \circ \boldsymbol{L}_j]\!] \tag{5.8}$$

where T_i and T_j are the lengths of the observation sequences $\boldsymbol{O}_i$ and $\boldsymbol{O}_j$, and provide sequence-length normalisation. Calculation of this kernel can be performed using only standard transducer algorithms—custom dynamic programming techniques are not required.

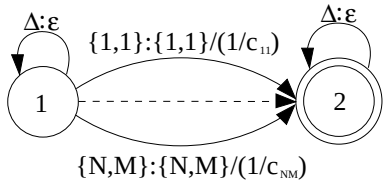
More complex derivatives can also be calculated. Consider, for example, the first-order derivatives of an N-state, M-mixture-component HMM with respect to the output distribution mixture-component priors, c_{jm},

$$\boldsymbol{\phi}^{\mathtt{c}}_{jm}(\boldsymbol{O};\boldsymbol{\lambda}) = \nabla_{c_{jm}} \ln \hat{p}(\boldsymbol{O};\boldsymbol{\lambda}) = \frac{1}{c_{jm}} \sum_{t=1}^{T} P(\theta_t^{jm}|\boldsymbol{O};\boldsymbol{\lambda}) - \sum_{t=1}^{T} P(\theta_t^{j}|\boldsymbol{O};\boldsymbol{\lambda}) \tag{5.9}$$

[4]Note that the additional constant in equation (5.6) does not affect classification and so can be ignored.

Similarly to the GMM derivatives in equation (5.6), feature-space elements consist of two terms. The first is the scaled posterior probabilities of state/mixture-component pairings and can be calculated using the scaled unigram transducer, $\boldsymbol{U}^{/\mathsf{c}}_{\mathtt{hmm}}$,[5]

$\boldsymbol{U}^{/\mathsf{c}}_{\mathtt{hmm}} =$

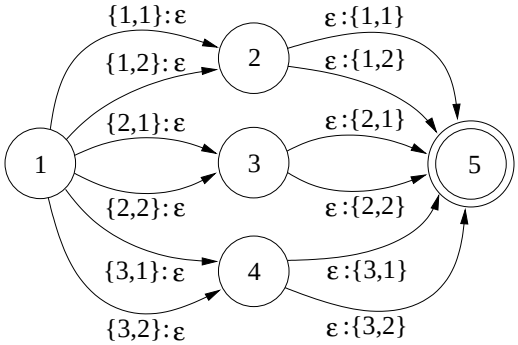


The second term in equation (5.9) is more complex since it is a function of the state posteriors, whereas the input acceptor $\boldsymbol{L}$ is labelled only in terms of state/mixture-component pairs. State posteriors, $P(\theta^j_t|\boldsymbol{O};\boldsymbol{\lambda})$, must therefore be decomposed into a summation over state/mixture-component posteriors,

$$P(\theta^j_t|\boldsymbol{O};\boldsymbol{\lambda}) = \sum_{m=1}^{M} P(\theta^{jm}_t|\boldsymbol{O};\boldsymbol{\lambda}) \tag{5.10}$$

As discussed above, the state/mixture-component posteriors can be calculated using the transducer $\boldsymbol{U}_{\mathtt{hmm}}$ (the unscaled version of $\boldsymbol{U}^{/\mathsf{c}}_{\mathtt{hmm}}$). Given these posteriors, the summation in equation (5.10) can be performed by composing $\boldsymbol{U}_{\mathtt{hmm}}$ with a special summation transducer, $\boldsymbol{S}$. For a three-state HMM with two mixture-components, this is defined as,

$\boldsymbol{S} =$



To calculate the HMM component-prior kernel, the component-prior derivatives are first expressed solely in terms of $P(\theta^{jm}_t|\boldsymbol{O};\boldsymbol{\lambda})$ by substituting equation (5.10) into equation (5.9),

$$\boldsymbol{\phi}^{\mathsf{c}}_{jm}(\boldsymbol{O};\boldsymbol{\lambda}) = \frac{1}{c_{jm}} \sum_{t=1}^{T} P(\theta^{jm}_t|\boldsymbol{O};\boldsymbol{\lambda}) - \sum_{t=1}^{T}\sum_{m=1}^{M} P(\theta^{jm}_t|\boldsymbol{O};\boldsymbol{\lambda}) \tag{5.11}$$

Using this expression for $\boldsymbol{\phi}^{\mathsf{c}}_{jm}(\boldsymbol{O};\boldsymbol{\lambda})$, the component-prior kernel can be calculated as a

[5]Note that the only difference between this transducer and $\boldsymbol{U}^{/\mathsf{c}}_{\mathtt{gmm}}$ is that arcs are labelled with both the state and mixture-component instead of just the mixture-component label; arc weights are adjusted accordingly. If all GMM mixture-components within the HMM were uniquely identified (instead to being referred to as mixture m of state j), then the GMM version of the scaled unigram transducer, $\boldsymbol{U}^{/\mathsf{c}}_{\mathtt{gmm}}$ could be used instead.

length-normalised dot-product in the $\boldsymbol{\phi}^{\mathsf{c}}_{jm}(\boldsymbol{O};\boldsymbol{\lambda})$ score-space,

$$\begin{aligned} K^{\mathsf{c}}_{\texttt{hmm}}(\boldsymbol{O}_i,\boldsymbol{O}_k;\boldsymbol{\lambda}) &= \frac{1}{T_iT_k}\sum_{j=1}^{N}\sum_{m=1}^{M}\boldsymbol{\phi}^{\mathsf{c}}_{jm}(\boldsymbol{O}_i;\boldsymbol{\lambda})\boldsymbol{\phi}^{\mathsf{c}}_{jm}(\boldsymbol{O}_k;\boldsymbol{\lambda}) \\ &= \frac{1}{T_iT_k}\sum_{j=1}^{N}\sum_{m=1}^{M}\left[\frac{1}{c_{jm}}\sum_{t=1}^{T}P(\theta_t^{jm}|\boldsymbol{O}_i;\boldsymbol{\lambda}) - \sum_{t=1}^{T}\sum_{m=1}^{M}P(\theta_t^{jm}|\boldsymbol{O}_i;\boldsymbol{\lambda})\right]\times \\ &\qquad \left[\frac{1}{c_{jm}}\sum_{t=1}^{T}P(\theta_t^{jm}|\boldsymbol{O}_k;\boldsymbol{\lambda}) - \sum_{t=1}^{T}\sum_{m=1}^{M}P(\theta_t^{jm}|\boldsymbol{O}_k;\boldsymbol{\lambda})\right] \end{aligned} \tag{5.12}$$

When the brackets are expanded, equation (5.12) can be written are the sum of three separate kernels, each of which is calculated using a different combination of the transducers $\boldsymbol{U}_{\texttt{hmm}}$, $\boldsymbol{U}^{/\mathsf{c}}_{\texttt{hmm}}$ and $\boldsymbol{S}$. Substituting these transducer combinations into the general continuous rational kernel framework, enables the HMM component-prior kernel to be expressed as a sum of continuous rational kernels,

$$\begin{aligned} K^{\mathsf{c}}_{\texttt{hmm}}(\boldsymbol{O}_i,\boldsymbol{O}_k;\boldsymbol{\lambda}) = \frac{1}{T_iT_k}\Big(&[\![\boldsymbol{L}_i\circ\boldsymbol{U}^{/\mathsf{c}}_{\texttt{hmm}}\circ\boldsymbol{U}^{/\mathsf{c}^{-1}}_{\texttt{hmm}}\circ\boldsymbol{L}_k]\!] \\ &- 2[\![\boldsymbol{L}_i\circ\boldsymbol{U}^{/\mathsf{c}}_{\texttt{hmm}}\circ\boldsymbol{S}^{-1}\circ\boldsymbol{U}^{-1}_{\texttt{hmm}}\circ\boldsymbol{L}_k]\!] \\ &+ [\![\boldsymbol{L}_i\circ\boldsymbol{U}_{\texttt{hmm}}\circ\boldsymbol{S}\circ\boldsymbol{S}^{-1}\circ\boldsymbol{U}^{-1}_{\texttt{hmm}}\circ\boldsymbol{L}_k]\!]\Big) \end{aligned} \tag{5.13}$$

Calculation of this kernel is performed using the same operations as used for calculating the first-order GMM-based component-prior kernel discussed previously. The only difference is the choice of transducers and the inclusion of the summation transducer, $\boldsymbol{S}$.

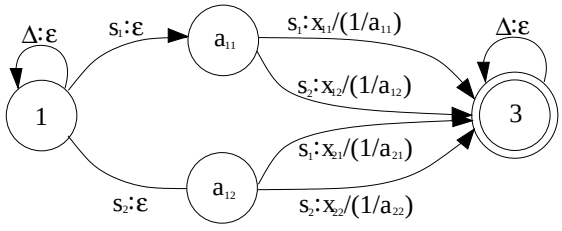
Transition probability kernels

In addition to derivatives with respect to the output distribution mixture-components, HMM derivatives with respect to the base model transition probabilities can be calculated. These are given by,

$$\nabla_{a_{ij}}\ln\hat{p}(\boldsymbol{O};\boldsymbol{\lambda}) = \sum_{t=1}^{T}\frac{P(\theta_{t-1}^{i},\theta_t^{j}|\boldsymbol{O};\boldsymbol{\lambda})}{a_{ij}} - P(\theta_{t-1}^{i}|\boldsymbol{O};\boldsymbol{\lambda}) \tag{5.14}$$

As for the mixture-component based derivatives, the score-space in equation (5.14) can be calculated within the continuous rational kernel framework. The first term is similar to a bigram feature-space and is calculated using a scaled version of the bigram transducer, $\boldsymbol{B}^{/a}$. For a two-state HMM, $\boldsymbol{B}^{/a}$ can be written as,

$\boldsymbol{B}^{/a}$ =



where s_i denotes the set of state labels (calculated from the state/mixture-component labels using the transducer $\boldsymbol{S}$). Elements in the feature-space are indexed by x_{ij}, which represents pairs of consecutive states. The second term, the state posterior, $P(\theta_t^j|\boldsymbol{O};\boldsymbol{\lambda})$, is calculated using the transducers $\boldsymbol{U}_{\texttt{hmm}}$ and $\boldsymbol{S}$, as discussed previously. The HMM transition probability kernel is thus calculated similarly to equation (5.13).

Fisher and generative score-spaces

In the previous sections, scaled unigram and bigram transducers were used for the calculating HMM derivatives with respect to the transition probabilities and mixture-component priors. However, more powerful score-spaces can be defined using derivatives with respect to the mixture-component means, $\boldsymbol{\mu}_{jm}$, and covariances, $\boldsymbol{\Sigma}_{jm}$. Consider the score-space of derivatives with respect to the means [118],[6]

$$\boldsymbol{\phi}^{\texttt{m}}_{jm}(\boldsymbol{O};\boldsymbol{\lambda}) = \nabla_{\boldsymbol{\mu}_{jm}} \ln \hat{p}(\boldsymbol{O};\boldsymbol{\lambda}) = \sum_{t=1}^{T} P(\theta_t^{jm}|\boldsymbol{O};\boldsymbol{\lambda})\boldsymbol{\Sigma}_{jm}^{-1}(\boldsymbol{o}_t - \boldsymbol{\mu}_{jm}) \tag{5.15}$$

Unfortunately, although the HMM-based latent-state acceptor—calculated using Forward-Backward alignment on the observation sequence—allows state/mixture-component posteriors to be calculated, it is not possible to weight the posteriors by the observations since these are vector quantities, whereas transducer weights are scalar.

To overcome this problem, a new semiring—the vector semiring—is proposed. Unlike standard semirings that assign scalar weights to arcs, the vector semiring enables transducer arcs to be labelled with vectors of weights. It is defined as the semiring $(\mathbb{R}^{d^+}, +, \otimes_{\text{vec}}, [\,0\,]^d, [\,1\,]^d)$ where $[\,x\,]^d$ represents a d-dimensional vector with all elements set to x. Semiring addition and multiplication are performed on a per-dimension basis, with multiplication defined as, $x \otimes_{\text{vec}} y = \{x_1y_1, x_2y_2, \ldots, x_dy_d\}$. For ease of comparison with the standard semirings in table 5.2, vector semirings are summarised in table 5.3.

Table 5.3: The vector semiring

Semiring	$\mathbb{K}$	$\oplus$	$\otimes$	$\bar{0}$	$\bar{1}$
Vector	$\mathbb{R}^{d^+}$	$+$	$\otimes_{\text{vec}}$	$[\,0\,]^d$	$[\,1\,]^d$

where $x \otimes_{\text{vec}} y = \{x_1y_1, x_2y_2, \ldots, x_dy_d\}$

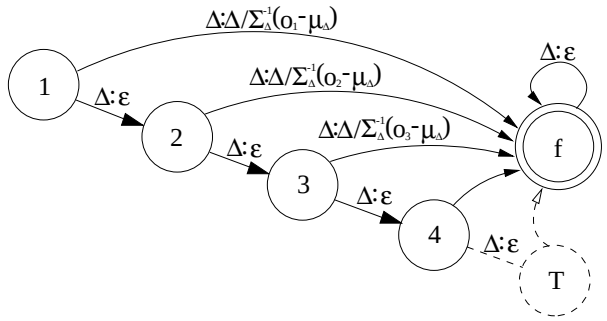
Although, at first glance, vector semirings look straightforward, there are a number of subtle issues that must be resolved. The first is that all acceptors and transducers within a calculation must be expressed using the same semiring. Composition of a real-semiring acceptor with a vector-semiring transducer therefore requires the acceptor weights, w, to be mapped to a vector weight, $[w]^d$. The second issue is that the shortest-distance of a

[6] Covariance derivatives have a similar functional form to derivatives with respect to the mean. For brevity, they are omitted.

vector semiring acceptor is a vector quantity instead of the scalar value required by the kernel. To convert from the shortest-distance vector to a standard dot-product, the sum of the vector elements must be computed.

The vector semiring allows the posterior probabilities generated by unigram transducers to be weighted by vector quantities. Unfortunately, for standard unigram transducers, these weights are time-independent whereas HMM derivatives with respect to the mixture-component means—equation (5.15)—require that posteriors are multiplied by time-dependent functions of the observations $\boldsymbol{o}_t$. To allow this, the unigram transducer self-transitions can be expanded to create a set of time-dependent paths that can be weighted by a function of the t-th observation, $\boldsymbol{\Sigma}_{jm}^{-1}(\boldsymbol{o}_t - \boldsymbol{\mu}_{jm})$. The resulting transducer is known as the Fisher transducer, and is thus given by,

$\boldsymbol{F} =$



The Fisher kernel with an HMM mean derivative score-space is given by,

$$K^{\mathtt{m}}(\boldsymbol{O}_i, \boldsymbol{O}_j) = \frac{1}{T_i T_j} [\![\boldsymbol{L}_i \circ \boldsymbol{F} \circ \boldsymbol{F}^{-1} \circ \boldsymbol{L}_j]\!] \tag{5.16}$$

Derivatives with respect to the state/mixture-component covariances can be calculated by replacing the path weights in the Fisher transducer with the weights: $-\frac{1}{2}\text{vec}\big[\boldsymbol{\Sigma}_{jm}^{-1} + \boldsymbol{\Sigma}_{jm}^{-1}(\boldsymbol{o}_t - \boldsymbol{\mu}_{jm})(\boldsymbol{o}_t - \boldsymbol{\mu}_{jm})^T\boldsymbol{\Sigma}_{jm}^{-1}\big]$.

5.3.2 Second and higher-order derivatives

In the previous sections continuous rational kernels were used to calculate first-order derivatives of GMMs and HMMs with respect to their parameters. For many of these derivatives, it was found that, with a few simple extensions, the standard unigram transducer, $\boldsymbol{U}$ (section 5.2), could be used within the continuous rational kernel framework. In this section, methods for calculating higher-order derivatives of the base models are considered.

As discussed in section 4.3.2, the reasons for considering higher-order derivatives of a base model are clear. From an augmented model point-of-view, higher-order derivatives represent additional sufficient statistics and more degrees of freedom. This allows higher-order augmented models to model a wider range of data distributions. Similarly, from a kernel point-of-view, second-order derivatives represent additional features that can be extracted from the observation sequence. These features expose the classifier to additional

information, potentially improving class discrimination. Unfortunately, as discussed in section 4.3.2, traditional dynamic programming approaches for calculating the joint posteriors required by second- and higher-order derivatives require custom algorithms and careful implementation. Continuous rational kernels, proposed in this work, allow such algorithms to be replaced by finite-state transducers.

Consider, for example, the second derivative of a base HMM with respect to its mixture-component priors,

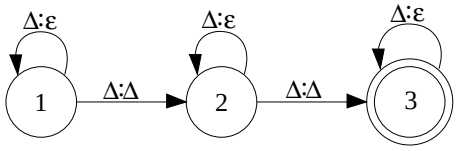
$$\nabla_{c_{kn}} \nabla^T_{c_{jm}} \ln \hat{p}(\boldsymbol{O}; \boldsymbol{\lambda}) = \frac{1}{c_{jm} c_{kn}} \sum_{t=1}^{T} \sum_{\tau=1}^{T} \Big(D(\theta_t^{jm}, \theta_\tau^{kn} | \boldsymbol{O}; \boldsymbol{\lambda}) - c_{jm} D(\theta_t^{j}, \theta_\tau^{kn} | \boldsymbol{O}; \boldsymbol{\lambda}) - c_{kn} D(\theta_t^{jm}, \theta_\tau^{k} | \boldsymbol{O}; \boldsymbol{\lambda}) + c_{jm} c_{kn} D(\theta_t^{j}, \theta_\tau^{k} | \boldsymbol{O}; \boldsymbol{\lambda}) \Big) - \frac{2}{c_{jm} c_{kn}} \sum_{t=1}^{T} P(\theta_t^{jm} | \boldsymbol{O}; \boldsymbol{\lambda}) \delta_{jk} \delta_{mn} \tag{5.17}$$

where

$$D(\theta_t^{jm}, \theta_\tau^{kn} | \boldsymbol{O}; \boldsymbol{\lambda}) = P(\theta_t^{jm}, \theta_\tau^{kn} | \boldsymbol{O}; \boldsymbol{\lambda}) - P(\theta_t^{jm} | \boldsymbol{O}; \boldsymbol{\lambda}) P(\theta_\tau^{kn} | \boldsymbol{O}; \boldsymbol{\lambda}) \tag{5.18}$$

Dynamic programming algorithms for calculating the joint posterior of being in state $\{j, m\}$ at time t and state $\{k, n\}$ at time τ are based upon the complex Double-Forward-Backward algorithm (appendix B), making direct calculation of second derivatives difficult. However, when continuous rational kernels are used, standard algorithms can be reused with new transducers. In particular, the transducer for calculating the joint posterior of the states, $P(\theta_t^{jm}, \theta_\tau^{kn} | \boldsymbol{O}; \boldsymbol{\lambda})$, is the gappy bigram transducer, $\boldsymbol{G}$,

$\boldsymbol{G}$ =



This generates a feature-space of the form, $\boldsymbol{\phi}^{\mathtt{gb}}_{jm,kn}(\boldsymbol{O}; \boldsymbol{\lambda})$,

$$\boldsymbol{\phi}^{\mathtt{gb}}_{jm,kn}(\boldsymbol{O}; \boldsymbol{\lambda}) = \sum_{t=1}^{T} \sum_{\tau=1}^{T} P(\theta_t = \{j, m\}, \theta_\tau = \{k, n\} | \boldsymbol{O}; \boldsymbol{\lambda}) \tag{5.19}$$

Scaling each arc in this transducer by $1/c_{jm}c_{kn}$—in a similar fashion to the scaled unigram transducers discussed earlier—yields a transducer that can be used to calculate the first part of the first term of equation (5.17). All other terms in this equation can then be generated using variants of the HMM gappy-bigram, $\boldsymbol{G}$, unigram, $\boldsymbol{U}_{\mathtt{hmm}}$, and summation, $\boldsymbol{S}$, transducers. For example, the term $P(\theta_t^{jm} | \boldsymbol{O}; \boldsymbol{\lambda}) P(\theta_\tau^{kn} | \boldsymbol{O}; \boldsymbol{\lambda})$ is calculated as the product of two unigram feature-spaces. This product can be calculated within the transducer framework as the concatenation, $(\boldsymbol{L} \circ \boldsymbol{U}_{\mathtt{hmm}}) \otimes (\boldsymbol{L} \circ \boldsymbol{U}_{\mathtt{hmm}})$, of two unigram feature-spaces,

$\boldsymbol{U}_{\texttt{hmm}}$, where $\otimes$ denotes transducer concatenation. Other terms can be calculated using similar techniques. By expanding the second-order derivative in equation (5.17) in a similar way to equation (5.12), the second-order HMM component-prior kernel can be written as the sum of continuous rational kernels.

Second derivatives of HMMs with respect to other model parameters can also be calculated. These take a similar form to equation (5.17) and are calculated using the transducers introduced in this chapter. Higher-order derivatives are calculated using other gappy-n-gram transducers, for example, the gappy-trigram transducer for third-derivatives. The benefit of calculating these higher derivatives using the continuous rational kernel framework is that many of the complexities of the alternative dynamic programming approach (such as allocation of multi-dimensional arrays, caching of results, etc.) are avoided. Instead, standard, efficient, transducer composition algorithms are used.

5.3.3 Generative score-spaces and kernels

Throughout this chapter, transducers have been presented as a method for calculating GMM and HMM derivatives. These derivatives can be used to define augmented model sufficient statistics (chapter 6) or kernel score-spaces (chapter 3). In practice, however, statistics and kernels are typically defined using a combination of different derivatives, allowing many different types of dependency to be modelled. Consider, for example, a typical first-order generative score-space,

$$\boldsymbol{\phi}^{\texttt{wmc}}(\boldsymbol{O};\boldsymbol{\lambda}) = \begin{bmatrix} \ln \hat{p}(\boldsymbol{O};\boldsymbol{\lambda}^{(1)}) - \ln \hat{p}(\boldsymbol{O};\boldsymbol{\lambda}^{(2)}) \\ \nabla_{\boldsymbol{c}^{(1)}} \ln \hat{p}(\boldsymbol{O};\boldsymbol{\lambda}^{(1)}) \\ \nabla_{\boldsymbol{\mu}^{(1)}} \ln \hat{p}(\boldsymbol{O};\boldsymbol{\lambda}^{(1)}) \\ \nabla_{\boldsymbol{\Sigma}^{(1)}} \ln \hat{p}(\boldsymbol{O};\boldsymbol{\lambda}^{(1)}) \\ -\nabla_{\boldsymbol{c}^{(2)}} \ln \hat{p}(\boldsymbol{O};\boldsymbol{\lambda}^{(2)}) \\ -\nabla_{\boldsymbol{\mu}^{(2)}} \ln \hat{p}(\boldsymbol{O};\boldsymbol{\lambda}^{(2)}) \\ -\nabla_{\boldsymbol{\Sigma}^{(2)}} \ln \hat{p}(\boldsymbol{O};\boldsymbol{\lambda}^{(2)}) \end{bmatrix} \tag{5.20}$$

where $\nabla_{\boldsymbol{c}^{(\omega)}}$, $\nabla_{\boldsymbol{\mu}^{(\omega)}}$ and $\nabla_{\boldsymbol{\Sigma}^{(\omega)}}$ denote the vectors of derivatives with respect to the mixture-component priors, means and variances of the base model $\hat{p}(\boldsymbol{O};\boldsymbol{\lambda}^{(\omega)})$. The corresponding kernel,

$$K^{\texttt{wmc}}(\boldsymbol{O}_i,\boldsymbol{O}_j;\boldsymbol{\lambda}) = \boldsymbol{\phi}^{\texttt{wmc}}(\boldsymbol{O}_i;\boldsymbol{\lambda})^T \boldsymbol{\phi}^{\texttt{wmc}}(\boldsymbol{O}_j;\boldsymbol{\lambda}) \tag{5.21}$$

can be written as a sum of the kernels of the individual parts,

$$\begin{aligned} K^{\texttt{wmc}}(\boldsymbol{O}_i,\boldsymbol{O}_j;\boldsymbol{\lambda}) = K^{\texttt{llr}}(\boldsymbol{O}_i,\boldsymbol{O}_j;\boldsymbol{\lambda}) + K^{\texttt{c}}(\boldsymbol{O}_i,\boldsymbol{O}_j;\boldsymbol{\lambda}^{(1)}) \\ + K^{\texttt{m}}(\boldsymbol{O}_i,\boldsymbol{O}_j;\boldsymbol{\lambda}^{(1)}) + \cdots \end{aligned} \tag{5.22}$$

where $K^{\texttt{llr}}(\boldsymbol{O}_i,\boldsymbol{O}_j;\boldsymbol{\lambda})$ is the dot-product of the log-likelihood ratios and $K^{\texttt{c}}(\boldsymbol{O}_i,\boldsymbol{O}_j;\boldsymbol{\lambda}^{(1)})$, $K^{\texttt{m}}(\boldsymbol{O}_i,\boldsymbol{O}_j;\boldsymbol{\lambda}^{(1)})$, etc., are the individual derivative kernels, calculated using the transducers discussed previously.

5.4 Summary

In this chapter finite-state transducers and rational kernels were introduced as a powerful and flexible approach for calculating kernels and kernel feature-spaces for sequences of discrete observations (such as those discussed in chapter 3). Many tasks, however, utilise continuous observations and so cannot benefit from the rational kernel framework. To rectify this, continuous rational kernels were proposed. These combine techniques from latent-variable generative models and standard rational kernels, allowing powerful kernel feature-spaces to be constructed from continuous observations in a systematic fashion.

Continuous rational kernels were shown to be a simple and attractive method for calculating first-, second- and higher-order derivatives of GMMs and HMMs, allowing augmented model sufficient statistics to be calculated. As discussed, calculation of these statistics in the continuous rational kernel framework has significant advantages over the dynamic programming techniques of sections 4.3.1 and 4.3.2 since differences between derivatives are captured by the transducer definitions, not the algorithms. This allows continuous rational kernels to be defined using only a small number of standard and efficient algorithms. As discussed in both chapters 4 and 6, these derivatives and their associated score-spaces can be used both to define augmented model sufficient statistics and to train the augmented model parameters.

6

Training Generative Augmented Models

In chapters 4 and 5, augmented statistical models and their sufficient statistics were introduced. Dynamic programming and continuous rational kernel approaches for calculating these statistics were also discussed. This chapter extends these ideas by considering training algorithms for both the base model and the augmented model parameters.

In general, the intractable nature of many augmented model normalisation terms means that standard generative model parameter estimation techniques such as ML and MMI estimation cannot be used. To avoid these normalisation problems, this chapter introduces a binary classification framework that uses distance-based algorithms to training pairs of augmented models 'against each other' in a discriminative fashion. This allows augmented model training to be expressed in terms of a linear decision boundary in a high-dimensional score-space. The advantage of this approach is that the intractable augmented model normalisation terms combine to form a bias that is inferred during training—the individual normalisation terms need not be calculated explicitly. Additionally, by performing inference within the same framework, calculation of the normalisation terms, is never required. In this work, a maximum margin criterion—SVMs—is used to estimate linear decision boundaries.

This chapter is structured as follows. First, standard generative model training criteria are reviewed. A distance-based binary classification framework is then proposed as an efficient method for training pairs of augmented models. As discussed above, this framework allows augmented model parameter estimation to be expressed in terms of a linear decision boundary in a high-dimensional score-space. A variable-margin SVM is then proposed as an attractive maximum-margin approach for estimating both the augmented model and

the base model parameters. Finally, a 'code-breaking' approach for applying pair-trained augmented models to large vocabulary speech recognition systems is introduced.

6.1 Direct estimation of parameters

As discussed in section 2.1.3, two of the most popular algorithms for parameter estimation in generative statistical models are maximum likelihood (ML) [7, 100] and maximum mutual information (MMI) [5] estimation. Maximum likelihood estimation updates model parameters in order to maximise the likelihood of the training set, whereas MMI attempts to maximise the posterior probability of the correct class labels (a differential approximation to the error rate). Unfortunately both ML and MMI parameter estimation require that the generative model normalisation term can be calculated. Since this is intractable for many augmented models an alternative training criterion must be used. In this thesis a distance-based learning criterion is proposed for training augmented model parameters within a binary-classification framework.

6.2 Distance-based learning

Consider the binary classification task where observation sequences, $\boldsymbol{O} = \{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_T\}$, are sampled from one of two class-conditional source distributions, ω_1 or ω_2. When the correct class posterior distributions, $P(\omega_1|\boldsymbol{O})$ and $P(\omega_2|\boldsymbol{O})$, are known, the decision boundary that minimises the probability of error is given by Bayes' decision rule,

$$\frac{P(\omega_1|\boldsymbol{O})}{P(\omega_2|\boldsymbol{O})} \begin{array}{c} \omega_1 \\ > \\ < \\ \omega_2 \end{array} 1 \tag{6.1}$$

Unfortunately, the correct posterior distributions are often unknown. Instead, they are normally approximated using class-conditional generative models and Bayes' rule. In this section it is assumed that the class-conditional generative models are defined by the augmented models $p(\boldsymbol{O}; \boldsymbol{\lambda}^{(1)}, \boldsymbol{\alpha}^{(1)})$ and $p(\boldsymbol{O}; \boldsymbol{\lambda}^{(2)}, \boldsymbol{\alpha}^{(2)})$. These allow sequence likelihoods to be calculated, which can then be converted into posterior probabilities using Bayes' rule. Substituting these posterior probabilities into equation (6.1) yields the pairwise augmented model decision function,

$$\frac{P(\omega_1)\, p(\boldsymbol{O}; \boldsymbol{\lambda}^{(1)}, \boldsymbol{\alpha}^{(1)})}{P(\omega_2)\, p(\boldsymbol{O}; \boldsymbol{\lambda}^{(2)}, \boldsymbol{\alpha}^{(2)})} \begin{array}{c} \omega_1 \\ > \\ < \\ \omega_2 \end{array} 1 \tag{6.2}$$

where $P(\omega_1)$ and $P(\omega_2)$ are the prior probability distributions for the classes ω_1 and ω_2 respectively. Expanding the augmented models using the definition in equation (4.5), and

taking the natural logarithm of both sides allows equation (6.2) to be written as,

$$\ln\left(\frac{\hat{p}(\boldsymbol{O};\boldsymbol{\lambda}^{(1)})}{\hat{p}(\boldsymbol{O};\boldsymbol{\lambda}^{(2)})}\right) + {\boldsymbol{\alpha}^{(1)}}^{\mathrm{T}}\boldsymbol{T}(\boldsymbol{O};\boldsymbol{\lambda}^{(1)}) - {\boldsymbol{\alpha}^{(2)}}^{\mathrm{T}}\boldsymbol{T}(\boldsymbol{O};\boldsymbol{\lambda}^{(2)}) + b \underset{\omega_2}{\overset{\omega_1}{\gtrless}} 0 \tag{6.3}$$

$$b = \ln\left(\frac{P(\omega_1)\,\tau(\boldsymbol{\lambda}^{(2)},\boldsymbol{\alpha}^{(2)})}{P(\omega_2)\,\tau(\boldsymbol{\lambda}^{(1)},\boldsymbol{\alpha}^{(1)})}\right) \tag{6.4}$$

where b is a bias that combines the class priors and the augmented model normalisation terms into a single value. Rearranging, the decision function can be expressed as a linear decision boundary in a high-dimensional score-space,

$$\begin{bmatrix} 1 \\ \boldsymbol{\alpha}^{(1)} \\ \boldsymbol{\alpha}^{(2)} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \ln\hat{p}(\boldsymbol{O};\boldsymbol{\lambda}^{(1)}) - \ln\hat{p}(\boldsymbol{O};\boldsymbol{\lambda}^{(2)}) \\ \boldsymbol{T}(\boldsymbol{O};\boldsymbol{\lambda}^{(1)}) \\ -\boldsymbol{T}(\boldsymbol{O};\boldsymbol{\lambda}^{(2)}) \end{bmatrix} + b \underset{\omega_2}{\overset{\omega_1}{\gtrless}} 0$$

$$\Rightarrow \quad \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}^{\mathrm{LL}}(\boldsymbol{O};\boldsymbol{\lambda}) + b \underset{\omega_2}{\overset{\omega_1}{\gtrless}} 0 \tag{6.5}$$

where $\boldsymbol{w}$ defines the gradient of the linear decision boundary in terms of the augmented model parameters, $\boldsymbol{\alpha}^{(1)}$ and $\boldsymbol{\alpha}^{(2)}$,

$$\boldsymbol{w} = \begin{bmatrix} 1 \\ \boldsymbol{\alpha}^{(1)} \\ \boldsymbol{\alpha}^{(2)} \end{bmatrix}; \qquad \boldsymbol{\phi}^{\mathrm{LL}}(\boldsymbol{O};\boldsymbol{\lambda}) = \begin{bmatrix} \ln\hat{p}(\boldsymbol{O};\boldsymbol{\lambda}^{(1)}) - \ln\hat{p}(\boldsymbol{O};\boldsymbol{\lambda}^{(2)}) \\ \boldsymbol{T}(\boldsymbol{O};\boldsymbol{\lambda}^{(1)}) \\ -\boldsymbol{T}(\boldsymbol{O};\boldsymbol{\lambda}^{(2)}) \end{bmatrix} \tag{6.6}$$

and $\boldsymbol{\phi}^{\mathrm{LL}}(\boldsymbol{O};\boldsymbol{\lambda})$ is a feature-space that is defined entirely in terms of the observations $\boldsymbol{O}$, the base model parameters $\boldsymbol{\lambda} = \{\boldsymbol{\lambda}^{(1)}, \boldsymbol{\lambda}^{(2)}\}$, and the augmented model sufficient statistics $\boldsymbol{T}(\boldsymbol{O};\boldsymbol{\lambda}^{(1)})$ and $\boldsymbol{T}(\boldsymbol{O};\boldsymbol{\lambda}^{(2)})$. This feature-space is known as the augmented model score-space,[1] and is identical to the generative score-spaces discussed in section 3.2.2 (a different name has been used to emphasise the difference between score-spaces used to train augmented models and score-spaces used to define features for kernel-based classifiers). To ensure that sequences with different lengths map to similar regions of the score-space, sequence-length normalisation is often used: this introduces an additional factor, $1/T$, into all score-space elements, where T is the sequence length.

By expressing the augmented model decision boundary in the score-space form given in equation (6.5), it is clear that augmented parameters can be estimated simply by positioning a linear decision boundary in a high-dimensional score-space. The advantage of this approach over more traditional generative model estimation techniques—such as ML and MMI estimation—is that it allows augmented parameters to be estimated without explicit

[1]The term 'score-space' is used in place of the more general term, 'feature-space', to emphasise the similarities between equation (6.6) and the Fisher score-spaces of Jaakkola and Haussler [50] and the generative score-spaces of Smith and Gales [118, 119].

calculation of the augmented model normalisation terms (which are often intractable). It is worth noting, however, that there are two side-effects to training augmented models in this fashion. The first is that the inclusion of the class-priors in the decision boundary bias means that they are estimated along with the augmented parameters (unlike standard training where the priors are fixed before training). The second side-effect is that including the normalisation term in the bias introduces an interaction between the base model parameters, $\boldsymbol{\lambda}$, and the augmented parameters, $\boldsymbol{\alpha} = \{\boldsymbol{\alpha}^{(1)}, \boldsymbol{\alpha}^{(2)}\}$.

Unfortunately, not all linear decision boundaries in the augmented model score-space correspond to valid augmented models. This is because a decision boundary requires only that its bias (the ratio of the normalisation terms and class priors) is finite. In contrast, for augmented models to be valid statistical models the normalisation terms for both must be finite. Since the augmented model requirements are stricter than the decision boundary requirements, some decision boundaries may not correspond to valid augmented models. This may occur, for example, when both augmented models have infinitely large normalisation terms whose ratio is fixed and finite. In this situation, valid decision boundaries between the models can be obtained since the bias (the ratio of the normalisation terms) is constant. In contrast, since both augmented models have infinite normalisation terms, neither are valid statistical models. To avoid this, additional constraints on the augmented parameters—such as those in sections 4.2.1 and 4.2.3—can be introduced. In practice, however, since performance of augmented models is often evaluated within a classification framework, these constraints can sometimes be omitted.

Examples of algorithms for positioning linear decision boundaries in high-dimensional feature-spaces/score-spaces are the Perceptron algorithm [37, 45, 102], support vector machines (SVMs) [10, 19, 125], and the relevance vector machine (RVM) [122]. As discussed in chapter 2, SVMs are based upon the intuitive concept of maximising the margin of separation between two competing classes, an approach that has been shown to be related to minimising an upper-bound on the generalisation error [125]. With good generalisation performance, both theoretically and empirically [125], the following sections use SVMs to estimate linear decision boundaries.

6.2.1 Estimating augmented parameters, $\boldsymbol{\alpha}$

As discussed above, estimating $\boldsymbol{\alpha}^{(1)}$ and $\boldsymbol{\alpha}^{(2)}$ for a pair of augmented models is equivalent to estimating a linear decision boundary in the augmented model score-space $\boldsymbol{\phi}^{\text{LL}}(\boldsymbol{O}; \boldsymbol{\lambda})$. One popular training algorithm for this decision boundary is the soft-margin SVM. This implements a form of maximum-margin estimation and is implemented using the objective

function and constraints given below [125],

$$\begin{aligned} \text{minimise}_{\boldsymbol{w},b} \quad & \tfrac{1}{2}\langle \boldsymbol{w}, \boldsymbol{w}\rangle + C\sum_{i=1}^{n}\epsilon_i & \\ \text{subject to} \quad & y_i\big(\langle \boldsymbol{w}, \boldsymbol{\phi}^{\text{LL}}(\boldsymbol{O}_i;\boldsymbol{\lambda})\rangle + b\big) \geq 1-\epsilon_i & \forall i \in [1,n] \\ \text{and} \quad & \epsilon_i \geq 0 & \forall i \in [1,n] \end{aligned} \tag{6.7}$$

where the slack variables, ϵ_i, allow SVM training to converge even when the training set contains examples that cannot be perfectly classified. The regularisation parameter, C, controls the trade-off between maximising the margin and minimising the number of misclassified training examples.

Given a linear hyperplane that satisfies the augmented model constraint $w_1 = 1$ (the weight associated with the log-likelihood ratio is one), augmented model parameter estimates, $\boldsymbol{\alpha}^{(1)}$ and $\boldsymbol{\alpha}^{(2)}$, can be calculated from $\boldsymbol{w}$ using the relationship in equation (6.6). However, since SVMs do not explicitly enforce the augmented model constraint, SVM-estimated hyperplanes often have gradients with $w_1 \neq 1$. To allow augmented parameter values to be extracted from these hyperplanes, the hyperplane gradient and bias must therefore be scaled by a factor $1/w_1$. Whilst this has no effect on the overall decision boundary, it transforms $\boldsymbol{w}$ into a form suitable for extracting $\boldsymbol{\alpha}^{(1)}$ and $\boldsymbol{\alpha}^{(2)}$ using equation (6.6). The relationship between SVM-estimated gradients, $\boldsymbol{w}$, and the augmented parameters, $\boldsymbol{\alpha}^{(1)}$ and $\boldsymbol{\alpha}^{(2)}$, is therefore given by,

$$\frac{\boldsymbol{w}}{w_1} = \begin{bmatrix} 1 \\ \boldsymbol{\alpha}^{(1)} \\ \boldsymbol{\alpha}^{(2)} \end{bmatrix} \tag{6.8}$$

where w_1 is the first-element of the vector $\boldsymbol{w}$ and is associated with the log-likelihood ratio of the base models. Unfortunately, this approach has the disadvantage that the final SVM margin (after scaling) is determined during training instead of being fixed a priori. This can make interpretation of distances between examples difficult, potentially limiting the usefulness of any results.

To rectify this, variable-margin SVMs are proposed. These explicitly enforce the augmented model constraint that $w_1 = 1$, thus avoiding the need to scale the final hyperplane. First consider the standard soft-margin SVM constraint,

$$y_i\big(\big\langle \boldsymbol{w}, \boldsymbol{\phi}^{\text{LL}}(\boldsymbol{O}_i;\boldsymbol{\lambda})\big\rangle + b\big) \geq 1-\epsilon_i \qquad \forall i \in [1,n] \tag{6.9}$$

Separating the terms that are associated with the log-likelihood ratio of the base models from the score-space, $\boldsymbol{\phi}^{\text{LL}}(\boldsymbol{O}_i;\boldsymbol{\lambda})$, and hyperplane gradient, $\boldsymbol{w}$, allows the score-space and gradient to be written as,

$$\boldsymbol{\phi}^{\text{LL}}(\boldsymbol{O}_i;\boldsymbol{\lambda}) = \begin{bmatrix} \phi^{\text{LLR}}(\boldsymbol{O};\boldsymbol{\lambda}) \\ \boldsymbol{\phi}^{\text{LL}'}(\boldsymbol{O};\boldsymbol{\lambda}) \end{bmatrix} \qquad \boldsymbol{w} = \begin{bmatrix} 1 \\ \boldsymbol{w}' \end{bmatrix} \tag{6.10}$$

where $\phi^{\texttt{LLR}}(\boldsymbol{O};\boldsymbol{\lambda})$ is a one-dimensional score-space containing only the log-likelihood ratio of the base models, and $\boldsymbol{\phi}^{\texttt{LL}'}(\boldsymbol{O};\boldsymbol{\lambda})$ is a score-space containing all score-space terms except for the log-likelihood ratio.[2] Substituting the expanded score-space and gradient into the constraint in equation (6.9) yields the following variable-margin constraint,

$$\begin{aligned} & y_i\big(\phi^{\texttt{LLR}}(\boldsymbol{O}_i;\boldsymbol{\lambda}) + \big\langle \boldsymbol{w}', \boldsymbol{\phi}^{\texttt{LL}'}(\boldsymbol{O}_i;\boldsymbol{\lambda})\big\rangle + b\big) \geq 1 - \epsilon_i \\ \Rightarrow \quad & y_i\big(\big\langle \boldsymbol{w}', \boldsymbol{\phi}^{\texttt{LL}'}(\boldsymbol{O}_i;\boldsymbol{\lambda})\big\rangle + b\big) \geq 1 - y_i\phi^{\texttt{LLR}}(\boldsymbol{O}_i;\boldsymbol{\lambda}) - \epsilon_i \end{aligned} \tag{6.11}$$

where the standard SVM margin is replaced by the variable margin, $1 - y_i\phi^{\texttt{LLR}}(\boldsymbol{O}_i;\boldsymbol{\lambda})$. For a given observation, $\boldsymbol{O}_i$, this margin is set according to how well the base models classify the example. For example, when $\boldsymbol{O}_i$ is incorrectly classified by the base models, $y_i\phi^{\texttt{LLR}}(\boldsymbol{O}_i;\boldsymbol{\lambda})$ is negative, resulting in an increased margin. Conversely, the margin is reduced for examples that are well classified by the base models.[3] The overall effect of varying the margin in this way is to increase the weight associated with examples that the base models struggle with, whilst sacrificing the margin of easily classified examples.

Substituting the variable-margin constraint in equation (6.11) into the primal SVM optimisation in equation (6.7), allows variable-margin SVMs to be trained. In practice, however, SVMs are often trained in their dual representation. For variable-margin SVMs, the dual objective is given by,[4]

$$\begin{aligned} \text{maximise}_{\boldsymbol{\alpha}^{\texttt{svm}}} \quad & \sum_{i=1}^{n} \alpha_i^{\texttt{svm}}\Big(1 - y_i\phi^{\texttt{LLR}}(\boldsymbol{O}_i;\boldsymbol{\lambda})\Big) - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i^{\texttt{svm}}\alpha_j^{\texttt{svm}} y_i y_j K^{\texttt{LL}'}(\boldsymbol{O}_i, \boldsymbol{O}_j;\boldsymbol{\lambda}) \\ \text{subject to} \quad & \sum_{i=1}^{n} \alpha_i^{\texttt{svm}} y_i = 0 \\ \text{and} \quad & 0 \leq \alpha_i^{\texttt{svm}} \leq C \end{aligned} \tag{6.12}$$

where $\boldsymbol{\alpha}^{\texttt{svm}}$ are Lagrange multipliers associated with each training example. The variable-margin augmented model kernel, $K^{\texttt{LL}'}(\boldsymbol{O}_i, \boldsymbol{O}_j;\boldsymbol{\lambda})$, is given by,

$$\begin{aligned} K^{\texttt{LL}'}(\boldsymbol{O}_i, \boldsymbol{O}_j;\boldsymbol{\lambda}) &= \big\langle \boldsymbol{\phi}^{\texttt{LL}'}(\boldsymbol{O}_i;\boldsymbol{\lambda}), \boldsymbol{\phi}^{\texttt{LL}'}(\boldsymbol{O}_j;\boldsymbol{\lambda})\big\rangle \\ &= \boldsymbol{\phi}^{\texttt{LL}'}(\boldsymbol{O}_i;\boldsymbol{\lambda})^{\mathrm{T}} \boldsymbol{G}_{\texttt{LL}'}^{-1} \boldsymbol{\phi}^{\texttt{LL}'}(\boldsymbol{O}_j;\boldsymbol{\lambda}) \end{aligned} \tag{6.13}$$

where $\boldsymbol{G}_{\texttt{LL}'}^{-1}$ is a metric for the variable-margin augmented model score-space, $\boldsymbol{\phi}^{\texttt{LL}'}(\boldsymbol{O};\boldsymbol{\lambda})$.

[2] Note that when sequence-length normalisation is used, both $\phi^{\texttt{LLR}}(\boldsymbol{O};\boldsymbol{\lambda})$ and $\boldsymbol{\phi}^{\texttt{LL}'}(\boldsymbol{O};\boldsymbol{\lambda})$ must be normalised by $1/T$, where T is the length of the observation sequence $\boldsymbol{O}$.

[3] In some cases, the base models may have sufficient confidence in their prediction that $y_i L_i > 1$, resulting in a negative margin. When this occurs, the variable-margin SVM algorithm may allow the example to be misclassified—based upon the score-space terms, $\boldsymbol{\phi}^{\texttt{LL}'}(\boldsymbol{O};\boldsymbol{\lambda})$—by a distance of $\big(1 - y_i\phi^{\texttt{LLR}}(\boldsymbol{O}_i;\boldsymbol{\lambda})\big)$ without incurring a penalty. The resulting classifier will still correctly classify the example since the negative margin, combined with the LLR, gives an overall positive margin.

[4] See appendix C for the derivation.

When variable-margin SVMs are used to estimate linear hyperplanes in the variable-margin augmented model score-space, $\boldsymbol{\phi}^{\mathtt{LL}'}(\boldsymbol{O};\boldsymbol{\lambda})$, augmented model parameters, $\boldsymbol{\alpha}^{(1)}$ and $\boldsymbol{\alpha}^{(2)}$, can be extracted directly from the hyperplane gradient (no scaling is necessary since the constraint $w_1 = 1$ is explicitly enforced by the training algorithm),

$$\left[\begin{array}{c} \boldsymbol{\alpha}^{(1)} \\ \boldsymbol{\alpha}^{(2)} \end{array} \right] = \boldsymbol{w}' = \sum_{i=1}^{n} \alpha_i^{\mathtt{svm}} y_i \boldsymbol{G}_{\mathtt{LL}'}^{-1} \boldsymbol{\phi}^{\mathtt{LL}'}(\boldsymbol{O}_i;\boldsymbol{\lambda}) \tag{6.14}$$

Given an SVM-estimated hyperplane, equation (6.14) allows maximum-margin estimated augmented parameters, $\boldsymbol{\alpha}$, to be calculated, thereby defining a pair of maximum-margin augmented models. Unfortunately, since the normalisation terms of these augmented models are often intractable, inference may not be possible within the standard probabilistic framework. Instead, inference must be performed within the SVM classification framework where the normalisation terms need not be calculated. This allows class label decisions to be determined using the distance of examples from the decision boundary. Since these distances represent the log-posterior ratio of the two augmented models—from the definition in equation (6.2)—the classification outputs are probabilistic in nature. This probabilistic interpretation allows SVM-trained augmented models to be used within standard system combination setups.

Note that, as discussed in the previous section, constraints on the augmented parameters may be required to ensure that decision boundaries correspond to valid augmented models. Depending upon the definition of the augmented model sufficient statistics (and hence score-space), these constraints represent either upper or lower bounds on elements of the hyperplane gradient. Unfortunately, there is no easy method of introducing these constraints into the dual SVM objective function. However, since augmented models are typically used for classification, these constraints can often be omitted.

6.2.2 Estimating base model parameters, $\boldsymbol{\lambda}$

In the previous section base model parameters were fixed to create a static score-space. This allowed the augmented parameters, $\boldsymbol{\alpha}^{(1)}$ and $\boldsymbol{\alpha}^{(2)}$, to be estimated using a maximum-margin training criterion. Experiments have shown that augmented models trained in this fashion often generalise well to unseen data despite the high-dimensionality of the score-space. However, there is a mismatch between the training criteria used for estimation of the base model parameters (ML/MMI) and that used for the augmented parameters (MM). To maintain consistency whilst achieving good generalisation performance, maximum-margin estimation of the base model parameters, $\boldsymbol{\lambda}$, is therefore preferred.

Explicit maximum-margin estimation of $\boldsymbol{\lambda}$

Maximum margin estimation of the base model parameters, $\boldsymbol{\lambda}$, is closely related to maximum margin estimation of the SVM kernel since changes in $\boldsymbol{\lambda}$ cause the augmented model score-space to vary. Previous approaches to kernel optimisation have relied upon minimising an estimate of an upper bound on the generalisation error [4,14,21]. Unfortunately, this approach requires two independent objective functions, calculated under different assumptions. Instead, in this work, a kernel optimisation approach is proposed that uses only a single objective function: a generalised version of the variable-margin SVM dual-objective function,[5]

$$\begin{aligned} \max_{\boldsymbol{\alpha}^{\mathtt{svm}}} \min_{\boldsymbol{\lambda}} \quad & W(\boldsymbol{\lambda}, \boldsymbol{\alpha}^{\mathtt{svm}}) = \sum_{i=1}^{n} \alpha_i^{\mathtt{svm}} \Big(1 - y_i \phi^{\mathtt{LLR}}(\boldsymbol{O}_i; \boldsymbol{\lambda})\Big) \\ & \qquad - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i^{\mathtt{svm}} \alpha_j^{\mathtt{svm}} y_i y_j K^{\mathtt{LL}'}(\boldsymbol{O}_i, \boldsymbol{O}_j; \boldsymbol{\lambda}) \\ \text{subject to} \quad & \sum_{i=1}^{n} \alpha_i^{\mathtt{svm}} y_i = 0 \\ \text{and} \quad & 0 \leq \alpha_i^{\mathtt{svm}} \leq C \end{aligned} \tag{6.15}$$

This is a non-standard optimisation problem with no closed-form solution. Instead, letting the k-th iteration estimates for $\boldsymbol{\lambda}$, $\boldsymbol{\alpha}^{\mathtt{svm}}$ and $W(\boldsymbol{\lambda}, \boldsymbol{\alpha}^{\mathtt{svm}})$ be written as $\boldsymbol{\lambda}^{(k)}$, $\boldsymbol{\alpha}^{\mathtt{svm}(k)}$ and $W^{(k)}$, the iterative algorithm in figure 6.1 can be used.

```
# Initialisation
Initialise base model parameters, λ^(0), using ML estimation
Determine initial support vectors, α^svm(0), using variable-margin SVM training
Calculate the objective function, W^(0) = W(λ^(0), α^svm(0))

# Iterate until convergence, W^(k-1) − W^k < threshold
Foreach k = 1 → MaxIter
    Update λ: λ^(k) = argmin_λ W(λ; α^svm)|_{λ^(k-1), α^svm(k-1)}
    Update α: α^svm(k) = argmax_{α^svm} W(α^svm; λ)|_{λ^(k), α^svm(k-1)}
    Calculate new objective function, W^(k) = W(λ^(k), α^svm(k))
End
```

Figure 6.1: Explicit maximum-margin estimation of the base model parameters, $\boldsymbol{\lambda}$

This algorithm operates as follows. First the base model parameters, $\boldsymbol{\lambda}$, and the SVM Lagrange multipliers, $\boldsymbol{\alpha}^{\mathtt{svm}}$, are initialised. Then, fixing $\boldsymbol{\alpha}^{\mathtt{svm}}$, gradient-descent is used to

[5] Maximising the margin is equivalent to minimising the norm of the weight vector, $|\boldsymbol{w}'|^2 = \boldsymbol{w}'^{\mathrm{T}}\boldsymbol{w}' = \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i^{\mathtt{svm}} \alpha_j^{\mathtt{svm}} y_i y_j K^{\mathtt{LL}'}(\boldsymbol{O}_i, \boldsymbol{O}_j; \boldsymbol{\lambda})$. Since $\boldsymbol{\alpha}^{\mathtt{svm}}$ is maximised (standard SVM training), the norm is minimised when the kernel is minimised with respect to $\boldsymbol{\lambda}$.

minimise the objective function with respect to $\boldsymbol{\lambda}$. Since this an unconstrained optimisation, the SVM Karush-Kuhn-Tucker (KKT) conditions (relating to the margin constraints) may be broken. To resolve this, the augmented parameters are re-estimated by re-training the SVM. This fixes the KKT conditions but may cause the objective function to be sub-optimal with respect to $\boldsymbol{\lambda}$. The process is therefore repeated until the objective function is optimal with respect to both $\boldsymbol{\lambda}$ and $\boldsymbol{\alpha}$ (with the KKT conditions satisfied).

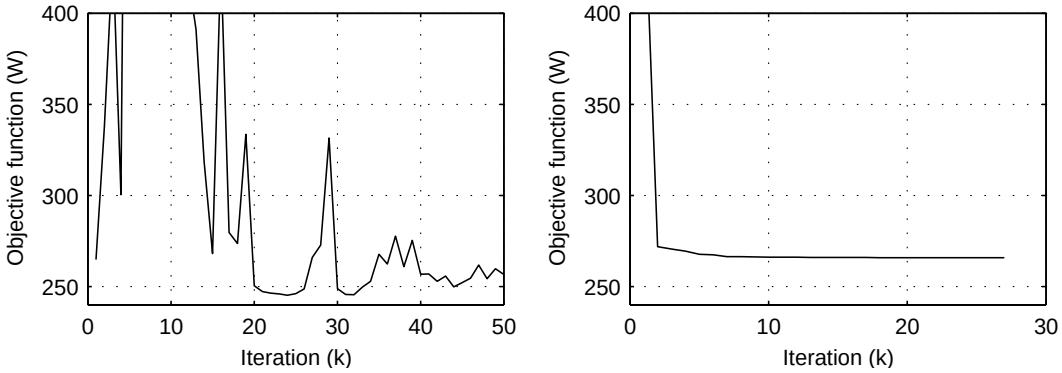


Figure 6.2: Variation of the SVM dual objective function during maximum-margin estimation of the base models: (a) constant step-size (b) using back-off algorithm

Unfortunately the two-step algorithm in figure 6.1 introduces discontinuities into the error surface, allowing small changes in the kernel parameters, $\boldsymbol{\lambda}$, to create large changes in the objective function. As illustrated in figure 6.2(a), this sensitivity to $\boldsymbol{\lambda}$ may cause the objective function to take higher (worse) values than in previous iterations. This instability can be alleviated by introducing a back-off algorithm: if the parameter update increases the objective function, the previous parameter values are restored and the learning rate is reduced. The effects of this approach are illustrated in figure 6.2(b). Without back-off, large fluctuations in the objective function are observed, with convergence occurring after approximately 50 iterations. With back-off, the objective function decreases monotonically, converging after just seven iterations.

Although the back-off algorithm stabilises the optimisation, it also biases the system against large changes in the base model parameters. This increases the likelihood of the optimisation converging to a local minimum. This is illustrated in figures 6.2(a) and 6.2(b) since without back-off the system converges to an objective function of 255, whereas with back-off the objective only reaches 265. Note, however, that without back-off, convergence is not guaranteed (and is often not attained) due to the two-stage nature of the optimisation process.

Approximate maximum-margin estimation for $\boldsymbol{\lambda}$

The maximum-margin base model parameter estimation algorithm discussed in the previous section is both computationally expensive and likely to converge to a local minimum. This makes it unsuitable for many applications. Instead, in this section, an approximate form of maximum-margin training for the base model parameters is proposed.

Given a ρ-th order augmented model, $p(\boldsymbol{O};\boldsymbol{\lambda},\boldsymbol{\alpha})$, let $\boldsymbol{\lambda}^{\texttt{ML}}$ be ML-estimated base model parameters,[6] and $\boldsymbol{\alpha}^{\texttt{MM}}$ be maximum-margin-estimated augmented parameters. Consider a υ-th order ($\upsilon \geq \rho$) Taylor series expansion of the augmented model about the fixed point $\{\boldsymbol{\lambda}^{\texttt{ML}}, \boldsymbol{\alpha}^{\texttt{MM}}\}$,[7]

$$\begin{aligned}\ln p(\boldsymbol{O};\boldsymbol{\lambda},\boldsymbol{\alpha}) \approx \ln p(\boldsymbol{O};\boldsymbol{\lambda}^{\texttt{ML}},\boldsymbol{\alpha}^{\texttt{MM}}) + \Big[&\boldsymbol{\beta}^{\mathrm{T}}\nabla_{\boldsymbol{\alpha}}\ln p(\boldsymbol{O};\boldsymbol{\lambda},\boldsymbol{\alpha}) \\ &+ \boldsymbol{\delta}_1^{\mathrm{T}}\mathrm{vec}\big(\nabla_{\boldsymbol{\alpha}}\nabla_{\boldsymbol{\lambda}}\ln p(\boldsymbol{O};\boldsymbol{\lambda},\boldsymbol{\alpha})\big) + \cdots + \boldsymbol{\delta}_{\upsilon-1}^{\mathrm{T}}\mathrm{vec}\big(\nabla_{\boldsymbol{\alpha}}\nabla_{\boldsymbol{\lambda}}^{\upsilon-1}\ln p(\boldsymbol{O};\boldsymbol{\lambda},\boldsymbol{\alpha})\big) \\ &+ \boldsymbol{\gamma}_1^{\mathrm{T}}\nabla_{\boldsymbol{\lambda}}\ln p(\boldsymbol{O};\boldsymbol{\lambda},\boldsymbol{\alpha}) + \cdots + \boldsymbol{\gamma}_{\upsilon}^{\mathrm{T}}\mathrm{vec}\big(\nabla_{\boldsymbol{\lambda}}^{\upsilon}\ln p(\boldsymbol{O};\boldsymbol{\lambda},\boldsymbol{\alpha})\big)\Big]_{\boldsymbol{\lambda}^{\texttt{ML}},\boldsymbol{\alpha}^{\texttt{MM}}}\end{aligned} \tag{6.16}$$

where $\nabla_{\boldsymbol{\lambda}}^{\upsilon} f(\cdot)$ denotes the υ-th order derivative of $f(\cdot)$ with respect to $\boldsymbol{\lambda}$. For clarity, the multiplicative constants in the Taylor series—1/2!, 1/3!, etc.—are factored into the Taylor-series coefficients $\boldsymbol{\beta}$, $\boldsymbol{\delta}$ and $\boldsymbol{\gamma}$. Second- and higher-order derivatives with respect to $\boldsymbol{\alpha}$ are zero since augmented models are log-linear in $\boldsymbol{\alpha}$.

It is important to note that although this section will concentrate upon base model parameter estimation, the definition of the augmented model sufficient statistics means that changes in $\boldsymbol{\lambda}$ have a knock-on effect on the augmented parameters $\boldsymbol{\alpha}$. Optimising $\boldsymbol{\lambda}$ may therefore require re-estimation of $\boldsymbol{\alpha}$. To take this link into account, the Taylor-series expansion in equation (6.16) is constructed using derivatives with respect to both $\boldsymbol{\lambda}$ and $\boldsymbol{\alpha}$, allowing interactions between the parameters to captured. A further point to note is that, by truncating the Taylor-series expansion, the approximation in equation (6.16) may allow additional dependencies to be modelled that were not present in the original model to be represented. For example, as discussed in chapter 4, first-order HMM-based augmented models depend upon only the occupancies of the latent-states. However, when these augmented models are differentiated, terms that depend upon the occupancies of pairs of states are generated. Since these were not present in the original model, additional modelling power is obtained.

Using the augmented model definition in equation (4.5), the Taylor-series expansion in equation (6.16) can be written in terms of the ML-estimated base statistical model, $\hat{p}(\boldsymbol{O},\boldsymbol{\lambda}^{\texttt{ML}})$. Consider, for example, a first-order Taylor-series expansion of a first-order

[6] As with many explanations in this thesis, ML-estimated base models are assumed for convenience. In practice, however, parameters of the base models may be estimated using a wide range of different criteria.

[7] For clarity, the normalisation term, $\ln \tau(\boldsymbol{\lambda},\boldsymbol{\alpha})$, is not shown in equations (6.16–6.18). Since derivatives of $\ln \tau(\boldsymbol{\lambda},\boldsymbol{\alpha})$ are independent of the observations, $\ln \tau(\boldsymbol{\lambda},\boldsymbol{\alpha})$ and its derivatives can be combined into a single term and included in the decision boundary bias, b.

augmented model about the point $\{\boldsymbol{\lambda}^{\texttt{ML}}, \boldsymbol{\alpha}^{\texttt{MM}}\}$,

$$\begin{aligned}\ln p(\boldsymbol{O};\boldsymbol{\lambda},\boldsymbol{\alpha}) \approx \ln \hat{p}(\boldsymbol{O};\boldsymbol{\lambda}^{\texttt{ML}}) \\ + \left[(\boldsymbol{\alpha}+\boldsymbol{\beta}+\boldsymbol{\gamma}_1)^{\mathrm{T}}\nabla_{\boldsymbol{\lambda}}\ln\hat{p}(\boldsymbol{O};\boldsymbol{\lambda}) + (\boldsymbol{\gamma}_1\otimes\boldsymbol{\alpha})^{\mathrm{T}}\mathrm{vec}\big(\nabla^2_{\boldsymbol{\lambda}}\ln\hat{p}(\boldsymbol{O};\boldsymbol{\lambda})\big)\right]_{\boldsymbol{\lambda}^{\texttt{ML}},\boldsymbol{\alpha}^{\texttt{MM}}}\end{aligned} \tag{6.17}$$

where $\otimes$ denotes tensor multiplication. Regrouping the terms allows equation (6.17) to be written as a second-order augmented model with base model parameters, $\boldsymbol{\lambda}^{\texttt{ML}}$, and augmented parameters, $\hat{\boldsymbol{\alpha}}$,

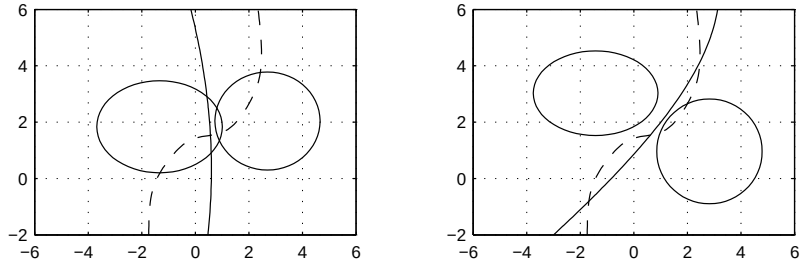
$$\begin{aligned}\ln p(\boldsymbol{O};\boldsymbol{\lambda},\boldsymbol{\alpha}) \approx \ln\hat{p}(\boldsymbol{O};\boldsymbol{\lambda}^{\texttt{ML}}) + \hat{\boldsymbol{\alpha}}^{\mathrm{T}}\left[\begin{array}{c}\nabla_{\boldsymbol{\lambda}}\ln\hat{p}(\boldsymbol{O};\boldsymbol{\lambda}) \\ \frac{1}{2!}\mathrm{vec}\big(\nabla^2_{\boldsymbol{\lambda}}\ln\hat{p}(\boldsymbol{O};\boldsymbol{\lambda})\big)\end{array}\right]_{\boldsymbol{\lambda}^{\texttt{ML}}} \\ \hat{\boldsymbol{\alpha}} = \left[\begin{array}{c}\boldsymbol{\alpha}^{\texttt{MM}}+\boldsymbol{\beta}+\boldsymbol{\gamma}_1 \\ 2\boldsymbol{\gamma}_1\otimes\boldsymbol{\alpha}^{\texttt{MM}}\end{array}\right]\end{aligned} \tag{6.18}$$

Equation (6.18) can therefore be considered to be an approximation to the original first-order augmented model. When the weights associated with cross-state derivatives are constrained to be zero, maximum margin estimation of $\hat{\boldsymbol{\alpha}}^{\mathrm{T}}$ is equivalent to maximum margin estimation of the base model parameters. The constraints on cross-state derivatives are used to ensure that the approximate model is not more powerful than the original model. This is particularly important for augmented models defined using latent-variable base models since first-order models are based upon the occupancy counts of individual states, $P(\theta_t^{jm}|\boldsymbol{O};\boldsymbol{\lambda})$, whereas the second-order model in (6.18) includes additional counts for pairs of states, $P(\theta_t^{jm},\theta_\tau^{kn}|\boldsymbol{O};\boldsymbol{\lambda})$. Since these counts are not present in the original first-order model, the weights associated them must be zero in order to avoid introducing additional dependencies. Despite these restrictions, provided that the ML-estimated base model parameters are sufficiently close to the true MM-estimated parameters, equation (6.18) allows maximum margin estimation of the base model parameters to be approximated simply by increasing the augmented model order by one. As the difference between the ML and MM base model parameters increases, the augmented model order must be increased by greater amounts.

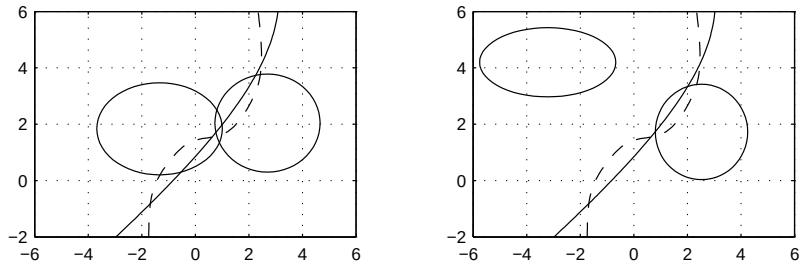
Approximate estimation of $\boldsymbol{\lambda}$ has two major advantages when compared to explicit optimisation. First, the time-consuming gradient-descent algorithm and repeated SVM training is avoided. Second, by fixing the sufficient statistics, the variable-margin SVM kernel remains fixed, allowing approximate MM estimation to converge to a unique solution, unlike the exact version. For tasks with complex error surfaces, approximate MM may outperform the exact alternative.

Illustrative example

Consider the two-dimensional, two-class artificial problem shown in figure 6.3. Observations for each class were generated using class-conditional GMMs (each with three mixture-



(a) $\boldsymbol{\phi}^{\mathtt{LLR}}(\boldsymbol{O};\boldsymbol{\lambda}^{\mathtt{ML}})$ (b) $\boldsymbol{\phi}^{\mathtt{LLR}}(\boldsymbol{O};\boldsymbol{\lambda}^{\mathtt{MM}})$



(c) $\boldsymbol{\phi}^{\mathtt{LL}}(\boldsymbol{O};\boldsymbol{\lambda}^{\mathtt{ML}})$ (d) $\boldsymbol{\phi}^{\mathtt{LL}}(\boldsymbol{O};\boldsymbol{\lambda}^{\mathtt{MM}})$

Figure 6.3: Maximum margin estimation of augmented and base model parameters (a) ML and (b) MM estimated Gaussians with log-likelihood-ratio feature-space $\boldsymbol{\phi}^{\mathtt{LLR}}(\boldsymbol{O};\boldsymbol{\lambda})$; (c) ML and (d) MM estimated Gaussians with augmented model score-space $\boldsymbol{\phi}^{\mathtt{LL}}(\boldsymbol{O};\boldsymbol{\lambda})$

components). To emulate the mismatch normally present between the true source distribution and the hypothesised models, Gaussian base models, $\hat{p}(\boldsymbol{O};\boldsymbol{\lambda}^{(1)})$ and $\hat{p}(\boldsymbol{O};\boldsymbol{\lambda}^{(2)})$, were estimated on the data. These are illustrated as ellipses in figure 6.3. The boundary of each ellipse represents a distance of one standard deviation from the mean.

Base model parameters were first estimated using ML training. Figure 6.3(a) shows the positions of the base model distributions, Bayes' decision boundary (dashed), and the SVM decision boundary for a one-dimensional feature-space, $\boldsymbol{\phi}^{\mathtt{LLR}}(\boldsymbol{O};\boldsymbol{\lambda})$ (solid line),

$$K^{\mathtt{LLR}}(\boldsymbol{O}_i,\boldsymbol{O}_j;\boldsymbol{\lambda}) = \left\langle \boldsymbol{\phi}^{\mathtt{LLR}}(\boldsymbol{O}_i;\boldsymbol{\lambda}), \boldsymbol{\phi}^{\mathtt{LLR}}(\boldsymbol{O}_j;\boldsymbol{\lambda}) \right\rangle \tag{6.19}$$

$$\boldsymbol{\phi}^{\mathtt{LLR}}(\boldsymbol{O};\boldsymbol{\lambda}) = \left[\ln \hat{p}(\boldsymbol{O};\boldsymbol{\lambda}^{(1)}) - \ln \hat{p}(\boldsymbol{O};\boldsymbol{\lambda}^{(2)}) \right] \tag{6.20}$$

From figure 6.3(a), it is clear that the log-likelihood ratio score-space derived from ML-estimated Gaussians is a poor approximation to the true (Bayes') decision boundary. Test error for the score-space was 22.8%, compared to 23.0% for Gaussian classification without the score-space, and 9.3% for the optimal minimum Bayes' error decision boundary.

Using equation (6.15), explicit MM estimation of the base model parameters was then performed. The resulting SVM decision boundary, shown in figure 6.3(b), is much closer to the Bayes' decision boundary and classification performance is significantly improved, with the test error falling from 22.8% (ML base model) to 10.4% (MM base model). Next, starting again with the ML base models, approximate MM estimation of $\boldsymbol{\lambda}$ was performed

by augmenting the base models with their first-derivatives. Decision boundaries were estimated using the $\boldsymbol{\phi}^{\mathtt{LL}}(\boldsymbol{O};\boldsymbol{\lambda}^{\mathtt{ML}})$ score-space and are shown in figure 6.3(c). Note that although the decision boundaries for both exact and approximate estimation of the base model parameters are almost identical, base model parameters estimated using approximate MM are unchanged. Finally, as shown in figure 6.3(d), both approximate and exact MM estimation were performed. This yielded no additional improvement in performance.

In this example, exact and approximate MM estimation yielded similar results. This is because, as discussed in section 4.2.1, the infinite Taylor series expansion for a member of the exponential family can be truncated to a first-order expansion with no loss of power. Approximate and exact MM estimation for Gaussian base models are therefore equivalent. Since the decision boundary is determined using only the sum of the base model and augmented model parameters, there are infinitely many different augmented models that generate the same decision boundary (each with a slightly different base model and augmented model parameters). When latent-variable base models are used, however, approximate and exact MM of model parameters are rarely equal. This is because truncating the Taylor series expansion alters the optimisation objective function.

6.3 Metrics

As discussed in the previous sections, parameters of augmented statistical models can be estimated using a maximum-margin training criterion. This uses the 'distances' between training examples to estimate augmented model parameters. These distances are measured using a kernel defined by a score-space and a score-space metric. This section examines the definition of the metric and proceeds as follows. First, a metric for calculating distances between probability distributions is defined. It is then shown that this metric can also be used when distances between examples are required (for maximum-margin training).

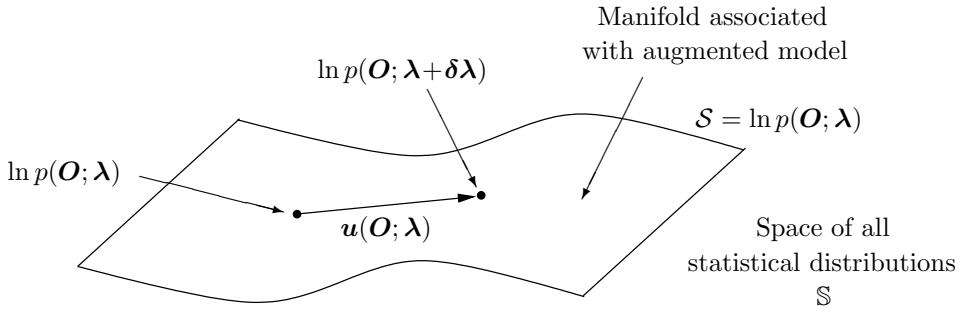


Figure 6.4: Augmented model metrics

Let $p(\boldsymbol{O};\boldsymbol{\lambda})$ be a generative statistical model with parameters $\boldsymbol{\lambda}$. As illustrated in figure 6.4, the set of all possible distributions of this model can be represented as a statistical manifold, $\mathcal{S}$, parameterised by $\boldsymbol{\lambda}$. The distance between two points (distributions) on this

manifold, $\ln p(\boldsymbol{O};\boldsymbol{\lambda})$ and $\ln p(\boldsymbol{O};\boldsymbol{\lambda}+\boldsymbol{\delta\lambda})$, can be calculated using an inner-product [3],

$$\begin{aligned} K^{\mathcal{S}} &= \big\langle \ln p(\boldsymbol{O};\boldsymbol{\lambda}), \ln p(\boldsymbol{O};\boldsymbol{\lambda}+\boldsymbol{\delta\lambda}) \big\rangle \\ &= \boldsymbol{u}(\boldsymbol{O};\boldsymbol{\lambda})^{\mathrm{T}}\, \boldsymbol{G}_{\mathcal{S}}\, \boldsymbol{u}(\boldsymbol{O};\boldsymbol{\lambda}) \end{aligned} \tag{6.21}$$

where the vector $\boldsymbol{u}$ denotes the direction of movement along the manifold. The matrix $\boldsymbol{G}_{\mathcal{S}}$ is the Riemannian metric tensor for $\mathcal{S}$. For statistical manifolds, this tensor is given by the Fisher Information matrix [3, 25, 101, 117],

$$\boldsymbol{G}_{\mathcal{S}} = \mathcal{E}_{\boldsymbol{O}}\Big\{ \Big[\nabla_{\boldsymbol{\lambda}} \ln p(\boldsymbol{O};\boldsymbol{\lambda})\Big] \Big[\nabla_{\boldsymbol{\lambda}} \ln p(\boldsymbol{O};\boldsymbol{\lambda})\Big]^{\mathrm{T}} \Big\} \tag{6.22}$$

Unfortunately, for many generative models, there is no closed-form solution for calculating $\boldsymbol{G}_{\mathcal{S}}$. Instead, it is usually approximated as either an identity matrix [50], or as an expectation over the training examples [25]. In this thesis, the latter approach is used.

Although equations (6.21) and (6.22) allows distances between distributions to be calculated, different directions of movement along the manifold, $\boldsymbol{u}(\boldsymbol{O};\boldsymbol{\lambda})$, generate different distances. To ensure uniqueness, the shortest distance is often used. As shown in [2], the shortest distance is obtained when $\boldsymbol{u}(\boldsymbol{O};\boldsymbol{\lambda})$ is in the direction of the natural gradient,

$$\boldsymbol{u}(\boldsymbol{O};\boldsymbol{\lambda}) = \boldsymbol{G}_{\mathcal{S}}^{-1} \nabla_{\boldsymbol{\lambda}} \ln p(\boldsymbol{O};\boldsymbol{\lambda}) \tag{6.23}$$

Substituting the natural gradient in equation (6.23) into the distance calculation in equation (6.21), allows the shortest distance across the manifold to be calculated. The shortest distance between the distributions $\ln p(\boldsymbol{O};\boldsymbol{\lambda})$ and $\ln p(\boldsymbol{O};\boldsymbol{\lambda}+\boldsymbol{\delta\lambda})$ is thus given by,

$$K^{\mathcal{S}} = \Big(\nabla_{\boldsymbol{\lambda}} \ln p(\boldsymbol{O};\boldsymbol{\lambda})\Big)^{\mathrm{T}} \boldsymbol{G}_{\mathcal{S}}^{-1} \Big(\nabla_{\boldsymbol{\lambda}} \ln p(\boldsymbol{O};\boldsymbol{\lambda})\Big) \tag{6.24}$$

Given this result, consider the manifold $\mathcal{T}$ of augmented model distributions parameterised by the augmented parameters $\boldsymbol{\alpha}$. Using equation (6.24), the distance between two different augmented model distributions can be written as,

$$K^{\mathcal{T}} = \Big[\boldsymbol{T}(\boldsymbol{O};\boldsymbol{\lambda}) - \boldsymbol{\mu}_{\boldsymbol{T}}\Big]^{\mathrm{T}} \boldsymbol{G}_{\mathcal{T}}^{-1} \Big[\boldsymbol{T}(\boldsymbol{O};\boldsymbol{\lambda}) - \boldsymbol{\mu}_{\boldsymbol{T}}\Big] \tag{6.25}$$

where the manifold metric, $\boldsymbol{G}_{\mathcal{T}}^{-1}$ is calculated using the Fisher information matrix,

$$\boldsymbol{G}_{\mathcal{T}} = \mathcal{E}_{\boldsymbol{O}}\Big\{ \Big[\boldsymbol{T}(\boldsymbol{O};\boldsymbol{\lambda}) - \boldsymbol{\mu}_{\boldsymbol{T}}\Big] \Big[\boldsymbol{T}(\boldsymbol{O};\boldsymbol{\lambda}) - \boldsymbol{\mu}_{\boldsymbol{T}}\Big]^{\mathrm{T}} \Big\} \tag{6.26}$$

and $\boldsymbol{T}(\boldsymbol{O};\boldsymbol{\lambda})$ are the augmented model sufficient statistics. The sufficient statistics mean $\boldsymbol{\mu}_{\boldsymbol{T}} = \mathcal{E}_{\boldsymbol{O}}\big\{\boldsymbol{T}(\boldsymbol{O};\boldsymbol{\lambda})\big\}$ arises from differentiating the generative model normalisation term.

Unfortunately, although equations (6.25) and (6.26) allow distances between different augmented model distributions to be calculated, parameter estimation for augmented models requires the calculation of distances between training examples, not probability

distributions. To determine how these distances relate to distances between distributions, consider a set of i.i.d. training examples, $\mathcal{O} = \{\boldsymbol{O}_1, \ldots, \boldsymbol{O}_n\}$, sampled from an unknown true distribution, and mapped into the space of augmented model sufficient statistics, $\{\boldsymbol{T}(\boldsymbol{O}_1;\boldsymbol{\lambda}), \ldots, \boldsymbol{T}(\boldsymbol{O}_n;\boldsymbol{\lambda})\}$. The probability density function for these examples can be written as [3],

$$p(\boldsymbol{T}(\boldsymbol{O}_i;\boldsymbol{\lambda});\boldsymbol{\xi}) = \begin{cases} \xi_i & 1 \leq i < n \\ 1 - \sum_{i=1}^{n-1} \xi_i & i = n \end{cases} \tag{6.27}$$

where $\boldsymbol{\xi}$ is an $n-1$ dimensional parameter vector that explicitly specifies the likelihood associated with each 'observation', $\boldsymbol{T}(\boldsymbol{O}_i;\boldsymbol{\lambda})$. The likelihood of obtaining $\boldsymbol{T}(\boldsymbol{O}_n;\boldsymbol{\lambda})$ is defined such that the total probability mass associated with $p(\boldsymbol{T}(\boldsymbol{O}_i;\boldsymbol{\lambda});\boldsymbol{\xi})$ is one.

As a probability distribution (albeit, an unusual one), $p(\boldsymbol{T}(\boldsymbol{O}_i;\boldsymbol{\lambda});\boldsymbol{\xi})$ defines a statistical manifold, $\mathcal{U}$, in the space of all possible probability distributions. Since different training examples represent different distributions on this manifold, distances between examples can be calculated using the results discussed above. By noting that the manifold of all parametric augmented statistical model distributions, $\mathcal{T}$, is a submanifold of the set of distributions defined by $p(\boldsymbol{T}(\boldsymbol{O}_i;\boldsymbol{\lambda});\boldsymbol{\xi})$, it is clear that the manifolds $\mathcal{T}$ and $\mathcal{U}$ must share the same metric [3], i.e.

$$\boldsymbol{G}_{\mathcal{U}} = \boldsymbol{G}_{\mathcal{T}}^{-1} \tag{6.28}$$

The metric for calculating distances between examples mapped into the space of augmented model sufficient statistics is therefore the inverse of the Fisher information matrix. This is identical to the inverse covariance of the sufficient statistics and corresponds to a maximally non-committal metric (a simple whitening of the data). For consistency, when examples are mapped into the augmented model score-space (containing the log-likelihood ratio), an equivalent metric is applied to the log-likelihood ratio.[8]

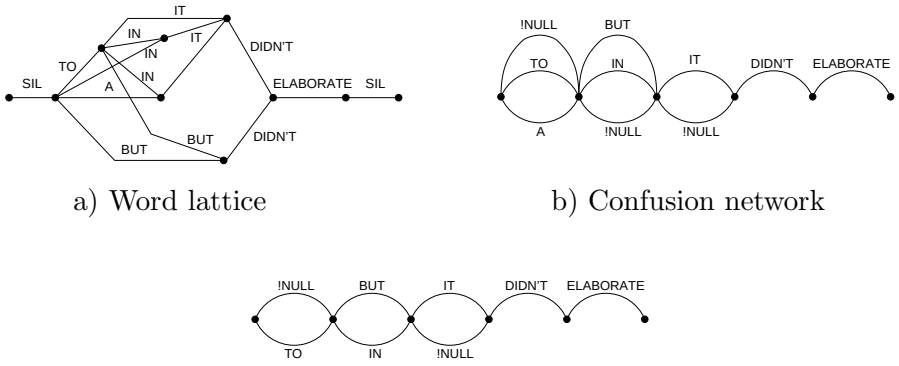
6.4 Acoustic code-breaking

As discussed in sections 6.2.1 and 6.2.2, maximum margin estimation of generative augmented model parameters is restricted to situations where utterances belong to one of two classes. Unfortunately, many practical speech recognition tasks require that utterance labels are chosen from a much wider range of class labels. It is therefore necessary

[8]Note: to ensure that Fisher information matrix has a full rank (allowing it to be inverted) the vector of sufficient statistics, $\boldsymbol{T}(\boldsymbol{O};\boldsymbol{\lambda})$, must have linearly-independent elements. For some forms of base model, this requires that linearly-dependent sufficient statistics are removed. Consider, for example, an M-mixture-component GMM. Sufficient statistics for the M-th mixture-component are linearly dependent on the other $M-1$ components due to the sum-to-one constraint on the component posterior probabilities, $P(\theta_t = M|\boldsymbol{O};\boldsymbol{\lambda}) = 1 - \sum_{m=1}^{M-1} P(\theta_t = m|\boldsymbol{O};\boldsymbol{\lambda})$. All statistics relating to the M-component must therefore be removed.

to examine how standard multiclass speech recognition tasks can be decomposed into a series of binary classification problems.

The machine learning literature includes many techniques for decomposing multiclass tasks into binary problems, for example [28] and [133]. Unfortunately, many of these scale badly with the number of classes—often $\mathcal{O}(C^2)$ or $\mathcal{O}(C^3)$, where C is the number of classes. Furthermore, for large vocabulary speech tasks, the number of potential classes (words) is usually too large for many traditional decomposition techniques. In this section, an acoustic code-breaking technique similar to that described in [126] is described.



a) Word lattice

b) Confusion network

c) Pruned confusion network

Figure 6.5: LVCSR conversion from multi-class to binary classification

Acoustic code-breaking is a three step process. Given the acoustic data for an utterance, a standard HMM-based LVCSR system is first used to determine the most likely utterance transcriptions. These transcriptions can be compactly represented using a word lattice (see figure 6.5(a)). Lattice nodes are labelled with time-stamps and arcs are labelled with words, language model probabilities and HMM acoustic model likelihoods. Next, using the algorithms in [75], the lattices are linearised to allow the competing words at each time instance to be identified. These linearised lattices are known as confusion networks (figure 6.5(b)). Each arc in the confusion network is labelled with a word, ω_i, a start and end time and the posterior probability of the word, $\mathcal{F}(\omega_i)$. The final step is to prune the confusion networks so that, between each pair of nodes, only the two most likely words (those with the highest word posteriors) remain. This pruning process is depicted in figure 6.5(c). The resulting pairs of words, for example BUT/IN in figure 6.5(c), are known as confusable pairs. When acoustic data associated with a confusable pair is extracted, data extraction must start at the earliest start time of the two words and finish at the latest end time. This ensures that the full acoustic data for both words is included and that neither word is truncated. The side effect of this process is that noise may be introduced at the start or end of each confusable pair.

Applying the above process to the training data allows commonly occurring confusions to be identified. Examples of each pair can then be located in the acoustic data and used

to train binary classifiers. Locating the same confusions in the test data then allows the confusable words in the LVCSR transcription to be rescored, potentially improving word and sentence accuracy. Maximum-margin generative augmented models are particularly suitable for this rescoring task since, with many complex sufficient statistics, augmented models can capture a much wider range of acoustic subtleties than standard HMM-based models. In addition, the use of a maximum margin training criterion allows augmented models to generalise well despite many model parameters and relatively little training data.

As discussed earlier in this chapter, maximum margin training of augmented model parameters involves the estimation of a unigram prior over the words.[9] For LVCSR tasks, however, trigram and higher-order language models are commonly used to capture longer-range dependencies between words. This information is reflected in the confusion network posteriors. To improve the discriminative ability of pairwise classifiers it is therefore useful to consider methods of incorporating this additional language model information (details of word context) into the binary classifier. One method of doing this is to consider a weighted combination of the confusion network classifiers and the length-normalised pairwise augmented model classifiers,

$$\frac{1}{T}\ln\left(\frac{p(\boldsymbol{O};\boldsymbol{\lambda}^{(1)},\boldsymbol{\alpha}^{(1)})P(\omega_1)}{p(\boldsymbol{O};\boldsymbol{\lambda}^{(2)},\boldsymbol{\alpha}^{(2)})P(\omega_2)}\right)+\beta\ln\left(\frac{\mathcal{F}(\omega_1)}{\mathcal{F}(\omega_2)}\right)\begin{array}{c}\omega_1\\ \gtrless\\ \omega_2\end{array}0 \tag{6.29}$$

where ω_1 and ω_2 are the confusable words. The relative weights of the augmented model likelihoods and the confusion network posteriors are specified by the scalar weight $\beta \geq 0$. This weight is often empirically set to maximise the performance over a development set. Alternatively, β can be determined automatically by appending the confusion network log-posterior-ratio to the augmented model score-space,

$$\boldsymbol{\phi}^{\mathtt{CN}}(\boldsymbol{O};\boldsymbol{\lambda})=\left[\begin{array}{c}\frac{1}{T}\boldsymbol{\phi}^{\mathtt{LL}}(\boldsymbol{O};\boldsymbol{\lambda})\\ \ln\mathcal{F}(\omega_1)-\ln\mathcal{F}(\omega_2)\end{array}\right] \tag{6.30}$$

When maximum margin training is performed using this score-space, a maximum margin estimate of β is obtained. The ability of MM-estimated augmented models to generalise well to unseen data means that, when training and test conditions are perfectly matched, MM-estimated values of β are expected to yield good generalisation performance. When conditions are mismatched, however, empirical estimates of β may yield better performance. For example, in cross validation experiments, confusion network lattices are often generated once using HMMs trained on all acoustic data. Posterior probabilities from these confusion networks are therefore biased since they are based upon both the training and test sets. Maximum margin estimated values of β can therefore be expected to be

[9] These unigram priors are part of the bias—equation (6.4)—that is estimated along with the decision hyperplane gradient.

larger than would otherwise be expected since the importance of the word posterior ratio is over-estimated.

6.5 Summary

In this chapter a binary distance-based classification framework was proposed for augmented model parameter estimation. This avoids the need to calculate the intractable normalisation terms by reducing augmented model parameter estimation to the task of estimating a linear decision boundary between pairs of augmented models in a high dimensional score-space. Variable-margin SVMs were proposed as a method of applying the maximum-margin training criterion to this task. These generate decision boundaries that generalise well to unseen data whilst maintaining the augmented model constraint that the weight associated with the base model is one. Using this criterion, parameter estimation algorithms for both the base model parameters and the augmented parameters were introduced. Score-space metrics were then derived using an information geometric argument. Finally a code-breaking approach was discussed for converting multi-class speech recognition problems into a series of binary classification tasks. Maximum-margin estimated augmented models can be applied to the resulting binary tasks.

7

Conditional Augmented Models

In previous chapters, generative augmented models were introduced as a powerful form of statistical model. Unfortunately, despite significant modelling benefits, the complexity of augmented model normalisation terms often makes likelihood calculations difficult and parameter estimation impractical. This severely limits the applicability of generative augmented models for many practical tasks.

In this chapter conditional augmented (C-Aug) models are proposed as an attractive alternative. These use the same sufficient statistics as generative augmented models but directly model class posteriors instead of utterance likelihoods. This simplifies the model normalisation by replacing the integration over all observation sequences with a simple summation over class labels. For many tasks this allows the normalisation to be calculated explicitly, making posterior probability calculations computationally feasible. Standard statistical model training algorithms can therefore be used for estimating C-Aug model parameters. In this work conditional maximum likelihood estimation—the discriminative model equivalent of maximum likelihood—is used. Finally, this chapter shows how C-Aug models can be extended to situations where observation sequences are transcribed with sequences of labels. One of the most common examples of this (and that discussed in this chapter) is continuous speech recognition.

This chapter is structured as follows. First, conditional augmented (C-Aug) models are introduced as a discriminative multi-class classification technique. Model definitions and properties are compared to those of generative augmented models. Parameter estimation and inference using the conditional maximum likelihood (CML) criterion are then discussed. Finally, a sentence-based classification framework (broadly equivalent to speech

recognition) is proposed. This uses the CML and MPE training criteria to estimate phone models within an efficient lattice-based framework.

7.1 Conditional augmented models

In this section conditional augmented (C-Aug) models are defined. These use the same sufficient statistics as the generative augmented models in chapter 4, but directly model the posterior probabilities of the correct class labels instead of utterance likelihoods. Defined using an exponential model with sufficient statistics given by the vectors $\boldsymbol{L}(\omega, \boldsymbol{O}; \boldsymbol{\lambda})$ and $\boldsymbol{T}(\omega, \boldsymbol{O}; \boldsymbol{\lambda})$, the C-Aug model posterior probability of the class $\omega \in \Omega = [1, C]$, given an observation sequence, $\boldsymbol{O} = \{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_T\}$, can be written as,

$$P(\omega|\boldsymbol{O}; \boldsymbol{\lambda}, \boldsymbol{\alpha}) = \frac{1}{Z(\boldsymbol{O}; \boldsymbol{\lambda}, \boldsymbol{\alpha})} \exp\left(\begin{bmatrix} \mathbf{1} \\ \boldsymbol{\alpha} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \boldsymbol{L}(\omega, \boldsymbol{O}; \boldsymbol{\lambda}) \\ \boldsymbol{T}(\omega, \boldsymbol{O}; \boldsymbol{\lambda}) \end{bmatrix} \right) \tag{7.1}$$

where elements of the augmented parameters, $\boldsymbol{\alpha}$, and sufficient statistics, $\boldsymbol{L}(\omega, \boldsymbol{O}; \boldsymbol{\lambda})$ and $\boldsymbol{T}(\omega, \boldsymbol{O}; \boldsymbol{\lambda})$, can be expanded in terms of the class-dependent base model parameters, $\boldsymbol{\lambda}^{(\omega)}$, and augmented parameters, $\boldsymbol{\alpha}^{(\omega)}$, using the expressions,

$$\begin{aligned} \boldsymbol{\alpha} &= [\boldsymbol{\alpha}^{(1)^{\mathrm{T}}}, \ldots, \boldsymbol{\alpha}^{(C)^{\mathrm{T}}}]^{\mathrm{T}} \\ L_{\omega'}(\omega, \boldsymbol{O}; \boldsymbol{\lambda}) &= \delta_{\omega=\omega'} \ln \hat{p}(\boldsymbol{O}; \boldsymbol{\lambda}^{(\omega')}) \qquad \forall \omega' \in \Omega \\ \boldsymbol{T}_{\omega'}(\omega, \boldsymbol{O}; \boldsymbol{\lambda}) &= \delta_{\omega=\omega'} \boldsymbol{T}(\boldsymbol{O}; \boldsymbol{\lambda}^{(\omega')}) \qquad \forall \omega' \in \Omega \end{aligned} \tag{7.2}$$

where $\boldsymbol{T}(\boldsymbol{O}; \boldsymbol{\lambda}^{(\omega)})$ are the standard augmented model sufficient statistics defined by equation (4.6). The C-Aug model normalisation term is denoted by $Z(\boldsymbol{O}; \boldsymbol{\lambda}, \boldsymbol{\alpha})$. For compactness, $\boldsymbol{\lambda}$ is used to denote the parameter vector $\left[\boldsymbol{\lambda}^{(1)^{\mathrm{T}}}, \ldots, \boldsymbol{\lambda}^{(C)^{\mathrm{T}}}\right]^{\mathrm{T}}$. The Kronecker-deltas, $\delta_{\omega=()}$, ensure that statistics from only a single base model are active at any one time, and allow the C-Aug model to be rewritten as,

$$P(\omega|\boldsymbol{O}; \boldsymbol{\lambda}, \boldsymbol{\alpha}) = \frac{1}{Z(\boldsymbol{O}; \boldsymbol{\lambda}, \boldsymbol{\alpha})} \hat{p}(\boldsymbol{O}; \boldsymbol{\lambda}^{(\omega)}) \exp\left(\boldsymbol{\alpha}^{\mathrm{T}} \boldsymbol{T}(\omega, \boldsymbol{O}; \boldsymbol{\lambda})\right) \tag{7.3}$$

The C-Aug model in equation (7.3) has a similar structure to the generative augmented models discussed in chapter 4. In practice, however, they are very different since the C-Aug model normalisation term, $Z(\boldsymbol{O}; \boldsymbol{\lambda}, \boldsymbol{\alpha})$,[1] is calculated as an expectation over all possible class labels, Ω,

$$Z(\boldsymbol{O}; \boldsymbol{\lambda}, \boldsymbol{\alpha}) = \sum_{\omega \in \Omega} \hat{p}(\boldsymbol{O}; \boldsymbol{\lambda}^{(\omega)}) \exp\left(\boldsymbol{\alpha}^{\mathrm{T}} \boldsymbol{T}(\omega, \boldsymbol{O}; \boldsymbol{\lambda})\right) \tag{7.4}$$

whereas generative augmented model normalisation terms are calculated as an expectation over all possible observation sequences. Since applications typically use only a small

[1] For clarity, conditional augmented model normalisation terms are denoted $Z(\cdot)$ instead of $\tau(\cdot)$ to emphasise that the expectation is calculated over all class labels instead of all possible observation sequences.

number of different classes, the summation in equation (7.4) can be calculated explicitly. Furthermore, the summation is bounded, thus avoiding the need for constraints on the augmented parameters.

Although it may seem strange to embed a generative base model within a discriminative model, this is a perfectly valid operation since the base model is used only to define a set of sufficient statistics. In contrast to other, ad-hoc, approaches to sufficient statistic selection, the base model allows the same systematic approach used for augmented models (chapter 4) to be applied when selecting C-Aug model sufficient statistics. Furthermore, since the statistics are based upon derivatives of latent-variable base models, they allow C-Aug models to overcome the conditional-independence assumptions of the base model (see section 4.2.2). This enables C-Aug models to represent a wider range of spatial and temporal dependencies than would otherwise be possible, allowing complex data distributions to be modelled.

The classification ability of C-Aug models can be further improved by calculating length-normalised base models and sufficient statistics that, when combined, form the C-Aug model sufficient statistics. This normalisation is useful since both the base model likelihoods and the augmented model sufficient statistics vary with sequence-length (from the summation over all observations in the sequence) whereas the augmented parameters associated with them do not. Length-normalisation prevents this mismatch by making all terms length-independent. Similarly to augmented models, length-normalisation is achieved by multiplying the base model likelihoods and sufficient statistics for observation sequences by a factor $1/T$, where T is the number of observations within a sequence. This allows C-Aug models to compare sequences of different lengths without classification results being biased in favour of either longer or shorter sequences. Experimentally, sequence-length normalisation has been found to improve classification performance.

A further advantage of sequence-length normalisation is that it compresses the dynamic range of the unnormalised—without $Z(\boldsymbol{O};\boldsymbol{\lambda},\boldsymbol{\alpha})$—C-Aug model scores. When $Z(\boldsymbol{O};\boldsymbol{\lambda},\boldsymbol{\alpha})$ is calculated using these length-normalised scores, a larger number of competing hypotheses contribute towards the overall model normalisation. In this respect, sequence-length normalisation is similar to the acoustic de-weighting used during MMI and MPE estimation of HMMs [96, 136] (see section 2.1.3 for more details). The main difference between the two approaches is that acoustic de-weighting scales by a fixed amount, whereas the scaling used by sequence-length normalisation varies according to sequence length.

As with all statistical models, the modelling power of C-Aug models is determined by the number of sufficient statistics used: the more sufficient statistics, the more degrees of freedom, and so the greater the power of the statistical model. Unfortunately, the benefits of additional statistics are obtained only at the expense of increased computational cost. Kernelised C-Aug models offer a potential solution. Defined similarly to kernelised

CRFs [64], these use standard Mercer kernel functions to map sufficient statistics into a higher-order feature-space where dot-products can be calculated efficiently. C-Aug models are then defined to use these high-dimensional features as sufficient statistics. The resulting models benefit from the additional statistics but do not pay the computational cost associated with calculating them. Kernelised C-Aug models are similar to the kernelised augmented models discussed in section 4.4.

7.2 Parameter estimation

As discussed in the previous section, C-Aug models have a significant advantage over standard generative augmented models since the normalisation term can be calculated simply. This makes direct training of the augmented model parameters using standard algorithms possible. One such algorithm is conditional maximum likelihood (CML) estimation.

Given a training set of i.i.d. observation sequences, $\mathcal{O} = \{\boldsymbol{O}_1, \ldots, \boldsymbol{O}_n\}$, with class labels $\mathcal{Y} = \{y_1, \ldots, y_n\}$, CML estimation adjusts model parameters in order to maximise the log-posterior probability of the correct class labels $\mathcal{Y}$, given the observations, $\mathcal{O}$. This can be written as,

$$\mathcal{F}^{\texttt{cml}}(\boldsymbol{\lambda}, \boldsymbol{\alpha}) = \ln P(\mathcal{Y}|\mathcal{O}; \boldsymbol{\lambda}, \boldsymbol{\alpha}) \tag{7.5}$$

where $\mathcal{F}^{\texttt{cml}}(\boldsymbol{\lambda}, \boldsymbol{\alpha})$ is the CML objective function. Since the observation sequences, $\boldsymbol{O} \in \mathcal{O}$, are independent and identically distributed, equation (7.5) can be expanded in terms of the log-posterior probabilities of the class labels for individual observation sequences. This enables the CML objective function, $\mathcal{F}^{\texttt{cml}}(\boldsymbol{\lambda}, \boldsymbol{\alpha})$, to be rewritten as,

$$\mathcal{F}^{\texttt{cml}}(\boldsymbol{\lambda}, \boldsymbol{\alpha}) = \sum_{i=1}^{n} \ln P(y_i|\boldsymbol{O}_i; \boldsymbol{\lambda}, \boldsymbol{\alpha}) \tag{7.6}$$

where $P(y_i|\boldsymbol{O}_i; \boldsymbol{\lambda}, \boldsymbol{\alpha})$ is the posterior probability of the class label y_i given the observation sequence, $\boldsymbol{O}_i$. From equations (7.5) and (7.6), it is clear that CML optimisation maximises the average log-posterior probability of the correct class labels—the same discriminative objective as MMI estimation. Despite this similarity, the two approaches are implemented differently since MMI training is a constrained optimisation problem that maintains the generative model parameter constraints, whereas, with no constraints on $\boldsymbol{\alpha}$, CML training is an unconstrained optimisation problem. Unfortunately, for many C-Aug models, equations (7.5) and (7.6) have no closed-form solution and instead iterative optimisation techniques are typically used. Although many algorithms have been proposed for training conditional models [23, 63, 93], this thesis will concentrate on gradient-based techniques (which have been shown to yield faster convergence in many cases [114, 129]).

Consider a C-Aug model with base model parameters $\boldsymbol{\lambda}$, augmented parameters $\boldsymbol{\alpha}$, and sufficient statistics $\boldsymbol{T}(\omega, \boldsymbol{O}; \boldsymbol{\lambda})$. Substituting this model into the CML objective function

in equation (7.6) yields the C-Aug CML objective,

$$\mathcal{F}^{\texttt{cml}}(\boldsymbol{\lambda}, \boldsymbol{\alpha}) = \sum_{i=1}^{n} \ln \left(\frac{1}{Z(\boldsymbol{O}_i; \boldsymbol{\lambda}, \boldsymbol{\alpha})} \hat{p}(\boldsymbol{O}_i; \boldsymbol{\lambda}^{(y_i)}) \exp \left(\boldsymbol{\alpha}^{\mathrm{T}} \boldsymbol{T}(y_i, \boldsymbol{O}_i; \boldsymbol{\lambda}) \right) \right) \tag{7.7}$$

This contains two inter-related sets of parameters—the base model parameters, $\boldsymbol{\lambda}$, and the augmented parameters, $\boldsymbol{\alpha}$—that can be optimised using gradient-based methods. Parameter updates are based upon the objective function gradients with respect to $\boldsymbol{\alpha}$ and $\boldsymbol{\lambda}$,

$$\nabla_{\boldsymbol{\alpha}} \mathcal{F}^{\texttt{cml}}(\boldsymbol{\lambda}, \boldsymbol{\alpha}) = \sum_{i=1}^{n} \left(\boldsymbol{T}(y_i, \boldsymbol{O}_i; \boldsymbol{\lambda}) - \sum_{\omega \in \Omega} P(\omega | \boldsymbol{O}_i; \boldsymbol{\lambda}, \boldsymbol{\alpha}) \boldsymbol{T}(\omega, \boldsymbol{O}_i; \boldsymbol{\lambda}) \right) \tag{7.8}$$

$$\nabla_{\boldsymbol{\lambda}} \mathcal{F}^{\texttt{cml}}(\boldsymbol{\lambda}, \boldsymbol{\alpha}) = \sum_{i=1}^{n} \left(\boldsymbol{T}(y_i, \boldsymbol{O}_i; \boldsymbol{\lambda}) - \sum_{\omega \in \Omega} P(\omega | \boldsymbol{O}_i; \boldsymbol{\lambda}, \boldsymbol{\alpha}) \boldsymbol{T}(\omega, \boldsymbol{O}_i; \boldsymbol{\lambda}) \right) \tag{7.9}$$

The first term in each derivative arises from differentiating the C-Aug model numerator; the second term arises from the denominator (normalisation) derivatives. Note that the gradient with respect to the base model parameters is derived using the simplifying assumption that second-order derivatives of the base model are zero. This assumption is motivated by the results in section 8.1.2.

Comparing equations (7.8) and (7.9) it is clear that the first-order derivatives of the CML objective function with respect to the augmented model and base model parameters are identical, allowing estimation algorithms for $\boldsymbol{\alpha}$ and $\boldsymbol{\lambda}$ to share the same update equations. Unfortunately, despite the similarity in parameter update equations, the error surfaces over which $\boldsymbol{\alpha}$ and $\boldsymbol{\lambda}$ are optimised can be very different since the second-order derivatives of the objective function with respect to $\boldsymbol{\alpha}$ and $\boldsymbol{\lambda}$ differ. This means that the optimisation of augmented model parameters and the optimisation of base model parameters may differ.

This difference is especially clear when the second derivatives of the objective function are calculated. Consider, for example, the second derivative with respect to the augmented parameters, $\boldsymbol{\alpha}$,

$$\begin{aligned}
\nabla_{\boldsymbol{\alpha}} \nabla_{\boldsymbol{\alpha}}^{\mathrm{T}} \mathcal{F}^{\texttt{cml}}(\boldsymbol{\lambda}, \boldsymbol{\alpha}) &= \sum_{i=1}^{n} \sum_{\omega \in \Omega} \left[\nabla_{\boldsymbol{\alpha}} P(\omega | \boldsymbol{O}_i; \boldsymbol{\lambda}, \boldsymbol{\alpha}) \right] \boldsymbol{T}(\omega, \boldsymbol{O}_i; \boldsymbol{\lambda})^{\mathrm{T}} \\
&= \sum_{i=1}^{n} \sum_{\omega \in \Omega} P(\omega | \boldsymbol{O}_i; \boldsymbol{\lambda}, \boldsymbol{\alpha}) \boldsymbol{T}(\omega, \boldsymbol{O}_i; \boldsymbol{\lambda}) \boldsymbol{T}(\omega, \boldsymbol{O}_i; \boldsymbol{\lambda})^{\mathrm{T}} \\
&\quad - \sum_{i=1}^{n} \sum_{\omega \in \Omega} \sum_{\omega' \in \Omega} P(\omega | \boldsymbol{O}_i; \boldsymbol{\lambda}, \boldsymbol{\alpha}) P(\omega' | \boldsymbol{O}_i; \boldsymbol{\lambda}, \boldsymbol{\alpha}) \boldsymbol{T}(\omega', \boldsymbol{O}_i; \boldsymbol{\lambda}) \boldsymbol{T}(\omega, \boldsymbol{O}_i; \boldsymbol{\lambda})^{\mathrm{T}}
\end{aligned} \tag{7.10}$$

Since the sufficient statistic outer-products, $\boldsymbol{T}(\cdot)\boldsymbol{T}(\cdot)^{\mathrm{T}}$, are positive semi-definite matrices, the Hessian matrix of the objective function, equation (7.10), is also positive semi-definite.

The CML objective function is therefore convex with respect to the augmented parameters, $\boldsymbol{\alpha}$. This means that the error function associated with the augmented parameters has only a single, global, maximum, making optimisation robust to changes in both the optimisation algorithm and the parameter initialisation. In contrast, the objective function Hessian matrix with respect to $\boldsymbol{\lambda}$ is not positive-definite, causing the optimisation with respect to the base model parameters to be non-convex. The error function used to update $\boldsymbol{\lambda}$ may therefore have many local maxima, making optimisation sensitive to both the optimisation algorithm and the parameter initialisation method.

In this work, parameter estimation was performed using four different gradient-based optimisation algorithms: gradient ascent, stochastic gradient ascent, conjugate gradient, and scaled conjugate gradient. The first three are well known algorithms and are discussed in detail in [87, 98, 116]. The fourth, scaled conjugate gradient is a less well-known, but powerful, optimisation algorithm that combines the benefits of conjugate gradient methods with a closed-form line-search algorithm that significantly reduces the computational cost of conjugate-gradient approaches [83, 84]. Experiments that compare the rates of convergence of these algorithms for CML estimation of C-Aug models are discussed in section 8.2.3.

7.3 Inference

Given a trained C-Aug model, $P(\omega|\boldsymbol{O};\boldsymbol{\lambda},\boldsymbol{\alpha})$, with parameters $\{\boldsymbol{\lambda},\boldsymbol{\alpha}\}$, consider the task of assigning a class label y to an unseen example $\boldsymbol{O}$. Applying Bayes' decision rule, the class label that minimises the probability of error is given by,

$$y = \arg\max_{\omega\in\Omega} P(\omega|\boldsymbol{O};\boldsymbol{\lambda},\boldsymbol{\alpha}) \tag{7.11}$$

Inference thus proceeds as follows: the C-Aug posterior probabilities of all possible class labels are first calculated. The class label with the largest posterior is then selected. This process can be simplified by noting that the C-Aug model normalisation term is class-independent and so identical for all posteriors associated with an observation sequence. The normalisation term thus scales all posteriors identically leaving the ordering unchanged. It can therefore be ignored during inference, allowing the C-Aug model decision rule to be rewritten as,

$$y = \arg\max_{\omega\in\Omega} \left[\hat{p}(\boldsymbol{O};\boldsymbol{\lambda}^{(\omega)}) \exp\left(\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{T}(\omega,\boldsymbol{O};\boldsymbol{\lambda})\right)\right] \tag{7.12}$$

Without the normalisation term, this decision function is easier to implement than equation (7.11). The disadvantage, however, is that it is often hard to apply system combination techniques when probabilities are unnormalised. Note that there is little difference in computational cost between the two approaches since the decision function for normalised

posteriors can be implemented by caching the unnormalised posteriors calculated in equation (7.12). Given these unnormalised posteriors, the normalisation term can be calculated at little cost, allowing C-Aug model normalisation to be performed using a simple division by $Z(\boldsymbol{O};\boldsymbol{\lambda},\boldsymbol{\alpha})$.

7.4 Speech recognition

In the previous sections, C-Aug models were defined as a discriminative approach for labelling observation sequences, $\boldsymbol{O}$, with a single class label, y. However, for more complex tasks such as speech recognition, the number of different class labels (one for each sentence transcription) is vast. This, coupled with the variability of sentences in terms of length and phone content, can make the definition and estimation of sentence-based statistical models difficult. Instead, sentences are typically decomposed into sequences of words, which may then be split into phones. Statistical models can then be defined for each of these phones. By modelling sentences using only a relatively small number of phone models, sentence variability can be captured in a robust fashion. In the following sections, a lattice-based approach is discussed for constructing sentence-based C-Aug models using a small number of simple phone-based models. These are known as C-Aug sentence models.

7.4.1 Sentence modelling

In this section, a discriminative framework that allows C-Aug models to be used for transcribing complete sentences, without knowledge of the word or phone boundaries, is discussed. In particular, a lattice-based approach for defining C-Aug sentence models is proposed. This allows the posterior probability of correct sentences to be calculated using only a relatively small number of phone models.

The first step when constructing a C-Aug sentence model is to define a base generative model of the sentence. Consider an observation sequence, $\boldsymbol{O} = \{\boldsymbol{o}_1,\ldots.\boldsymbol{o}_T\}$, with a phone transcription, $\boldsymbol{W} = \{w_1,\ldots,w_L\}$.[2] Since phones are often associated with multiple observations, let $\boldsymbol{A} = \{a_1,\ldots,a_L\}$ be an alignment that describes the mapping between the phone labels, w_i, and the observations in $\boldsymbol{O}$ that they relate to. The alignment, a_i, for a phone is typically specified using the phone start and end times. This alignment is rarely known in practice, and must instead be inferred during training and inference. Given an observation sequence, $\boldsymbol{O}$, and sentence transcription, $\boldsymbol{W}$, a generative base model of the observations given the transcription, $\hat{p}(\boldsymbol{O}|\boldsymbol{W};\boldsymbol{\lambda})$, can be defined by marginalising

[2]Note that, to simplify the derivations of C-Aug sentence models, this section assumes that $\boldsymbol{W}$ is a phone-based transcription instead of a more conventional word transcription.

the alignment-dependent observation likelihoods over all possible phone alignments,

$$\begin{aligned}\hat{p}(\boldsymbol{O}|\boldsymbol{W};\boldsymbol{\lambda}) &= \sum_{\boldsymbol{A}\in\mathcal{A}_{\boldsymbol{W}}} p(\boldsymbol{O},\boldsymbol{A}|\boldsymbol{W};\boldsymbol{\lambda}) \\ &= \sum_{\boldsymbol{A}\in\mathcal{A}_{\boldsymbol{W}}} p(\boldsymbol{O}|\boldsymbol{A},\boldsymbol{W};\boldsymbol{\lambda})P(\boldsymbol{A}|\boldsymbol{W};\boldsymbol{\lambda}) \end{aligned} \tag{7.13}$$

where $\mathcal{A}_{\boldsymbol{W}}$ is the set of all valid phone alignments for the transcription $\boldsymbol{W}$, $p(\boldsymbol{O}|\boldsymbol{A},\boldsymbol{W};\boldsymbol{\lambda})$ is a generative model of the observations given both the transcription and alignment, and $P(\boldsymbol{A}|\boldsymbol{W};\boldsymbol{\lambda})$ is a posterior distribution of the alignment given the phone sequence. To maintain consistency with augmented and C-Aug models discussed previously, the base generative distributions in this section are denoted by probability density functions of the form, $\hat{p}(\cdot)$. Other distributions are denoted using the more familiar notation, $p(\cdot)$ and $P(\cdot)$.

When the generative sentence model is constructed using HMM phone models—as is common in speech recognition—the phones $w \in \boldsymbol{W}$ are conditionally-independent given the phone alignment, $\boldsymbol{A}$. This allows the complete observation sequence, $\boldsymbol{O}$, to be segmented into shorter sequences, $\boldsymbol{O}^{(w)}$, associated with individual phones. By definition, these observation sequence segments are also conditionally-independent given the phone alignment, allowing the transcription- and alignment-dependent observation likelihood, $p(\boldsymbol{O}|\boldsymbol{A},\boldsymbol{W};\boldsymbol{\lambda})$ in equation (7.13), to be expanded in terms of alignment-dependent phone base models, $\hat{p}(\boldsymbol{O}^{(w)}|w,\boldsymbol{A};\boldsymbol{\lambda})$,

$$\hat{p}(\boldsymbol{O}|\boldsymbol{W};\boldsymbol{\lambda}) = \sum_{\boldsymbol{A}\in\mathcal{A}_{\boldsymbol{W}}} P(\boldsymbol{A}|\boldsymbol{W};\boldsymbol{\lambda}) \prod_{w\in\boldsymbol{W}} \hat{p}(\boldsymbol{O}^{(w)}|w,\boldsymbol{A};\boldsymbol{\lambda}) \tag{7.14}$$

where $\boldsymbol{O} = \{\boldsymbol{O}^{(w_1)},\ldots,\boldsymbol{O}^{(w_L)}\}$ is the complete observation sequence, and $\boldsymbol{O}^{(w)}$ are segments that contain the observations belonging to a particular phone $w \in \boldsymbol{W}$ in the transcription. The posterior probability of the alignment given the transcription, $p(\boldsymbol{A}|\boldsymbol{W};\boldsymbol{\lambda})$, can be difficult to calculate since it requires the marginalisation of the joint observation/alignment likelihood over all possible observation sequences. To avoid this often intractable calculation, this work assumes that $P(\boldsymbol{A}|\boldsymbol{W};\boldsymbol{\lambda})$ is a uniform distribution. This allows equation (7.14) to be written as,

$$\hat{p}(\boldsymbol{O}|\boldsymbol{W};\boldsymbol{\lambda}) = \frac{1}{|\mathcal{A}_{\boldsymbol{W}}|} \sum_{\boldsymbol{A}\in\mathcal{A}_{\boldsymbol{W}}} \prod_{w\in\boldsymbol{W}} \hat{p}(\boldsymbol{O}^{(w)}|w,\boldsymbol{A};\boldsymbol{\lambda}) \tag{7.15}$$

where $|\mathcal{A}_{\boldsymbol{W}}|$ is the cardinality of $\mathcal{A}_{\boldsymbol{W}}$. Equation (7.15) will be referred to as the sentence base generative model.

Given the sentence base model, $\hat{p}(\boldsymbol{O}|\boldsymbol{W};\boldsymbol{\lambda})$, augmented model sufficient statistics can be defined by considering first- and higher-order derivatives of the base model. In this section, to maintain clarity, only first-order derivatives are considered.[3] Using the definition

[3]To simplify discussions this chapter assumes that first-order C-Aug models are used. In practice,

in equation (4.6), the augmented model sufficient statistics, $\boldsymbol{T}(\boldsymbol{W}, \boldsymbol{O}; \boldsymbol{\lambda})$, for the sentence base model, $\hat{p}(\boldsymbol{O}|\boldsymbol{W}; \boldsymbol{\lambda})$, are given by,

$$\boldsymbol{T}(\boldsymbol{W}, \boldsymbol{O}; \boldsymbol{\lambda}) = \nabla_{\boldsymbol{\lambda}} \ln \left(\frac{1}{|\mathcal{A}_{\boldsymbol{W}}|} \sum_{\boldsymbol{A} \in \mathcal{A}_{\boldsymbol{W}}} \prod_{w \in \boldsymbol{W}} \hat{p}(\boldsymbol{O}^{(w)}|w, \boldsymbol{A}; \boldsymbol{\lambda}) \right) \tag{7.16}$$

Augmenting the sentence base model in equation (7.15) with these sufficient statistics breaks the conditional-independence of the observation segments, $\boldsymbol{O}^{(w)}$, and their phone transcriptions, w, on the base model alignment $\boldsymbol{A}$. This allows long-range temporal dependencies within the sentence to be modelled. Unfortunately, for many sentences, statistics of this form (a summation over all possible alignments) make C-Aug sentence model calculations computationally expensive.

To simplify the calculations, this work restricts the C-Aug sentence models considered to models that retain the original base model assumption that observation segments are conditionally independent given the base model phone-alignment. Although this limits the long-range temporal dependencies that can be modelled, it allows calculations to be performed at a phone-level instead of a sentence-level. This enables efficient algorithms to be derived for parameter estimation and inference. The base model conditional-independence assumptions are maintained by forcing the C-Aug sentence model sufficient statistic phone-alignments to match those of the base model. This allows the summation over all phone-alignments in equation (7.16) to be replaced by a simple dependence on the base model alignment, resulting in the set of alignment-dependent C-Aug sentence model sufficient statistics, $\boldsymbol{T}(\boldsymbol{W}, \boldsymbol{O}|\boldsymbol{A}; \boldsymbol{\lambda})$,

$$\begin{aligned} \boldsymbol{T}(\boldsymbol{W}, \boldsymbol{O}|\boldsymbol{A}; \boldsymbol{\lambda}) &= \nabla_{\boldsymbol{\lambda}} \ln \left(\prod_{w \in \boldsymbol{W}} \hat{p}(\boldsymbol{O}^{(w)}|w, \boldsymbol{A}; \boldsymbol{\lambda}) \right) \\ &= \sum_{w \in \boldsymbol{W}} \nabla_{\boldsymbol{\lambda}} \ln \hat{p}(\boldsymbol{O}^{(w)}|w, \boldsymbol{A}; \boldsymbol{\lambda}) \\ &= \sum_{w \in \boldsymbol{W}} \boldsymbol{T}(w, \boldsymbol{O}^{(w)}|\boldsymbol{A}; \boldsymbol{\lambda}) \end{aligned} \tag{7.17}$$

where $\boldsymbol{T}(w, \boldsymbol{O}^{(w)}|\boldsymbol{A}; \boldsymbol{\lambda})$ are the augmented model sufficient statistics associated with the phone base model, $\hat{p}(\boldsymbol{O}^{(w)}|w, \boldsymbol{A}; \boldsymbol{\lambda})$.

Given both the base generative model in equation (7.15), and the sufficient statistics

this assumption is not necessary and higher-order models can be defined by varying the definitions of $\boldsymbol{T}(\boldsymbol{W}, \boldsymbol{O}; \boldsymbol{\lambda})$, $\boldsymbol{T}(\boldsymbol{W}, \boldsymbol{O}|\boldsymbol{A}; \boldsymbol{\lambda})$ and $\boldsymbol{T}(w, \boldsymbol{O}^{(w)}|\boldsymbol{A}; \boldsymbol{\lambda})$.

in equation (7.17), a sentence-level C-Aug model can be defined,[4]

$$\begin{aligned}
&P(\boldsymbol{W}|\boldsymbol{O};\boldsymbol{\lambda},\boldsymbol{\alpha}) \\
&\quad= \frac{1}{Z(\boldsymbol{O};\boldsymbol{\lambda},\boldsymbol{\alpha})}\hat{p}(\boldsymbol{O}|\boldsymbol{W};\boldsymbol{\lambda})\exp\left(\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{T}(\boldsymbol{W},\boldsymbol{O}|\boldsymbol{A};\boldsymbol{\lambda})\right) \\
&\quad= \frac{1}{Z(\boldsymbol{O};\boldsymbol{\lambda},\boldsymbol{\alpha})}\frac{1}{|\mathcal{A}_{\boldsymbol{W}}|}\sum_{\boldsymbol{A}\in\mathcal{A}_{\boldsymbol{W}}}\left[\left(\prod_{w\in\boldsymbol{W}}\hat{p}(\boldsymbol{O}^{(w)}|w,\boldsymbol{A};\boldsymbol{\lambda})\right)\exp\left(\boldsymbol{\alpha}^{\mathrm{T}}\sum_{w\in\boldsymbol{W}}\boldsymbol{T}(w,\boldsymbol{O}^{(w)}|\boldsymbol{A};\boldsymbol{\lambda})\right)\right] \\
&\quad= \frac{1}{Z(\boldsymbol{O};\boldsymbol{\lambda},\boldsymbol{\alpha})}\sum_{\boldsymbol{A}\in\mathcal{A}_{\boldsymbol{W}}}\prod_{w\in\boldsymbol{W}}\frac{1}{|\mathcal{A}_{\boldsymbol{W}}|}\hat{p}(\boldsymbol{O}^{(w)}|w,\boldsymbol{A};\boldsymbol{\lambda})\exp\left(\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{T}(w,\boldsymbol{O}^{(w)}|\boldsymbol{A};\boldsymbol{\lambda})\right) \quad (7.18)
\end{aligned}$$

where $\boldsymbol{\alpha}$ are the C-Aug model augmented parameters, and $Z(\boldsymbol{O};\boldsymbol{\lambda},\boldsymbol{\alpha})$ is the normalisation term, calculated as the expectation of the unnormalised C-Aug model over all possible phone transcriptions,

$$Z(\boldsymbol{O};\boldsymbol{\lambda},\boldsymbol{\alpha}) = \sum_{\boldsymbol{W}\in\mathcal{W}}\sum_{\boldsymbol{A}\in\mathcal{A}_{\boldsymbol{W}}}\prod_{w\in\boldsymbol{W}}\frac{1}{|\mathcal{A}_{\boldsymbol{W}}|}\hat{p}(\boldsymbol{O}^{(w)}|w,\boldsymbol{A};\boldsymbol{\lambda})\exp\left(\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{T}(w,\boldsymbol{O}^{(w)}|\boldsymbol{A};\boldsymbol{\lambda})\right) \quad (7.19)$$

and $\boldsymbol{W}\in\mathcal{W}$ is a hypothesis from the set of all possible transcriptions for the observation sequence, $\boldsymbol{O}$. Together, equations (7.18) and (7.19) define C-Aug sentence models in terms of a small number of phone models.

7.4.2 Lattice-based C-Aug sentence models

Unlike the simple C-Aug models discussed in section 7.1, exact calculation of the C-Aug sentence model normalisation term is rarely possible since the calculation in equation (7.19) typically involves a summation over a huge number of different sentence transcriptions and alignments. Instead the summation over phone transcriptions, $\boldsymbol{W}\in\mathcal{W}$, and corresponding alignments, $\boldsymbol{A}\in\mathcal{A}_{\boldsymbol{W}}$, can be approximated by considering only the most 'likely' sequences and alignments. When HMM base models are used, a hypothesis lattice containing the most likely (according to the base model) sentence transcriptions and alignments is often generated using a standard speech recognition Viterbi decoder [48]. This lattice is similar to the training lattices used during MMI and MPE training of HMMs [96]. Furthermore, when hypothesis lattices are generated using Viterbi decoding, each hypothesised sentence transcription has only one alignment—the most likely (or Viterbi) alignment. This allows the $1/|\mathcal{A}_{\boldsymbol{W}}|$ term, and the summation over all possible alignments, to be omitted from calculations.

[4]Note that, in practice, C-Aug sentence models are often defined using length-normalised base models and sufficient statistics (normalised by the sentence length, not the phone length). This is similar to the acoustic de-weighting used when performing MMI and MPE training of HMMs [96,136]. A full discussion of the effects of C-Aug model sequence-length normalisation is given in section 7.1. To simplify the notation, all equations and derivations in this section use unnormalised models.

For a lattice of competing hypotheses, $\boldsymbol{L}$, the normalisation term in equation (7.19) can be approximated as,

$$Z(\boldsymbol{O};\boldsymbol{\lambda},\boldsymbol{\alpha}) = \sum_{\{\boldsymbol{W},\boldsymbol{A}\}\in\boldsymbol{L}} \prod_{w\in\boldsymbol{W}} \hat{p}(\boldsymbol{O}^{(w)}|w,\boldsymbol{A};\boldsymbol{\lambda}) \exp\Big(\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{T}(w,\boldsymbol{O}^{(w)}|\boldsymbol{A};\boldsymbol{\lambda})\Big) \tag{7.20}$$

where the summation $\sum_{\{\boldsymbol{W},\boldsymbol{A}\}\in\boldsymbol{L}}$ denotes a summation over all phone transcriptions (with their associated Viterbi alignments) in the lattice $\boldsymbol{L}$. The disadvantage of using the Viterbi approximation is that the number of confusions the C-Aug model is trained to resolve is reduced from all possible transcriptions to just the transcriptions in the lattice. Generalisation performance of the C-Aug model may therefore be affected. In practice, however, for any reasonably sized lattice the number of different sentence hypotheses is large enough that this is not believed to be a serious problem.

Unfortunately, the number of different sentence hypotheses in a typical lattice is often too large for the summation in equation (7.20) to be calculated in the form given. Instead, by taking advantage of the lattice-structure of the hypotheses, the summation can be simplified. Since calculations of the phone base models and sufficient statistics are dependent upon only the current phone label and alignment, calculations can be associated with arcs of the hypothesis lattice (which correspond to a single phone label and alignment). This means that the HMM likelihood score, $\hat{p}(\boldsymbol{O}^{(w)}|w,\boldsymbol{A};\boldsymbol{\lambda})$, in each arc of the lattice $\boldsymbol{L}$ can be replaced with the unnormalised C-Aug phone model score, $\hat{p}(\boldsymbol{O}^{(w)}|w,\boldsymbol{A};\boldsymbol{\lambda})\exp\big(\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{T}(w,\boldsymbol{O}^{(w)}|\boldsymbol{A};\boldsymbol{\lambda})\big)$, where w is the arc phone label and $\boldsymbol{A}$ is an alignment that reflects the arc start and end times. The resulting lattice is known as the denominator lattice and is denoted $\boldsymbol{L}^{\mathtt{den}}$.

Given the denominator lattice, the C-Aug model normalisation term in equation (7.20) can be calculated as a summation of the scores for all possible lattice paths, weighted by the path likelihoods. This summation can be calculated efficiently using the lattice weight operation.[5] Lattice weights are calculated by propagating arc weights (probabilities) from the start node, through to the end node. When probabilities are propagated along paths they are multiplied; when paths merge the probabilities are summed. This is similar to the forward-pass of the Forward-Backward algorithm used for calculating HMM likelihoods (appendix B). Given the denominator lattice, $\boldsymbol{L}^{\mathtt{den}}$, the C-Aug model normalisation term can be calculated as,

$$Z(\boldsymbol{O};\boldsymbol{\lambda},\boldsymbol{\alpha}) = [\![\boldsymbol{L}^{\mathtt{den}}]\!] \tag{7.21}$$

where $[\![\boldsymbol{L}^{\mathtt{den}}]\!]$ denotes the lattice-weight (also known as the shortest distance). Computational cost of this calculation is linear in the number of lattice arcs (unlike the explicit calculation in equation (7.20), which is linear in the number of hypotheses—a much larger value).

[5] In finite state transducer terminology this is known as the lattice/acceptor shortest distance.

Using a similar approach, the C-Aug model numerator term can be calculated. First, a simple lattice representing the correct transcription is constructed. Arcs of this lattice are then scored in a similar fashion to the denominator lattice discussed above. The resulting lattice is known as the numerator lattice and is denoted, $\boldsymbol{L}^{\mathtt{num}}$. By calculating the weight of this lattice, the C-Aug model posterior can be written as the ratio of two lattice weights,

$$P(\boldsymbol{W}|\boldsymbol{O};\boldsymbol{\lambda},\boldsymbol{\alpha}) = \frac{[\![\boldsymbol{L}^{\mathtt{num}}]\!]}{[\![\boldsymbol{L}^{\mathtt{den}}]\!]} \tag{7.22}$$

The use of lattices for calculating the numerator and denominator terms of C-Aug sentence models yields a computationally efficient method for calculating C-Aug sentence model posteriors. Furthermore, by representing competing hypotheses for the normalisation term as a lattice, C-Aug model normalisation can be performed using millions of competing hypotheses, instead of the thousands that could be used if N-best lists were used. This allows better approximations to the normalisation to be calculated.

7.4.3 Conditional maximum likelihood estimation

As discussed in section 7.2, parameters of C-Aug models can be trained using the conditional maximum likelihood (CML) criterion. Updates are typically performed using a gradient-based optimisation algorithm: this work uses scaled conjugate gradient (SCG) optimisation [83, 84]. In this section, a lattice-based implementation of CML estimation suitable for training C-Aug sentence models is presented.

Given a set of training examples, $\mathcal{O} = \{\boldsymbol{O}_1, \ldots, \boldsymbol{O}_n\}$, with phone transcriptions, $\mathcal{T} = \{\boldsymbol{W}_1, \ldots, \boldsymbol{W}_n\}$, the CML objective function is defined as the log-posterior probability of obtaining the correct sentence transcriptions given the observation sequences, written as,

$$\mathcal{F}^{\mathtt{cml}}(\boldsymbol{\lambda},\boldsymbol{\alpha}) = \ln P(\mathcal{T}|\mathcal{O};\boldsymbol{\lambda},\boldsymbol{\alpha}) = \sum_{i=1}^{n} \ln P(\boldsymbol{W}_i|\boldsymbol{O}_i;\boldsymbol{\lambda},\boldsymbol{\alpha}) \tag{7.23}$$

$$= \sum_{i=1}^{n} \ln \left(\frac{[\![\boldsymbol{L}_i^{\mathtt{num}}]\!]}{[\![\boldsymbol{L}_i^{\mathtt{den}}]\!]} \right) \tag{7.24}$$

where $P(\boldsymbol{W}_i|\boldsymbol{O}_i;\boldsymbol{\lambda},\boldsymbol{\alpha})$ is the C-Aug sentence model posterior probability of the transcription, $\boldsymbol{W}_i$, given the observation sequence $\boldsymbol{O}_i$. The lattices $\boldsymbol{L}_i^{\mathtt{num}}$ and $\boldsymbol{L}_i^{\mathtt{den}}$ are the numerator and denominator lattices defined in the previous section.

Before deriving the update equations for CML estimation of C-Aug sentence model augmented parameters, it is useful to define the unnormalised alignment-dependent C-Aug model sentence posterior, $S(\boldsymbol{O},\boldsymbol{W},\boldsymbol{A};\boldsymbol{\lambda},\boldsymbol{\alpha})$, and its gradient, $\nabla_{\boldsymbol{\alpha}} S(\boldsymbol{O},\boldsymbol{W},\boldsymbol{A};\boldsymbol{\lambda},\boldsymbol{\alpha})$,

$$S(\boldsymbol{O},\boldsymbol{W},\boldsymbol{A};\boldsymbol{\lambda},\boldsymbol{\alpha}) = \frac{1}{|\mathcal{A}_{\boldsymbol{W}}|} \prod_{w \in \boldsymbol{W}} \hat{p}(\boldsymbol{O}^{(w)}|w,\boldsymbol{A};\boldsymbol{\lambda}) \exp\left(\boldsymbol{\alpha}^{\mathrm{T}} \boldsymbol{T}(w,\boldsymbol{O}^{(w)}|\boldsymbol{A};\boldsymbol{\lambda})\right) \tag{7.25}$$

$$\nabla_{\boldsymbol{\alpha}} S(\boldsymbol{O},\boldsymbol{W},\boldsymbol{A};\boldsymbol{\lambda},\boldsymbol{\alpha}) = \sum_{w \in \boldsymbol{W}} S(\boldsymbol{O},\boldsymbol{W},\boldsymbol{A};\boldsymbol{\lambda},\boldsymbol{\alpha}) \boldsymbol{T}(w,\boldsymbol{O}^{(w)}|\boldsymbol{A};\boldsymbol{\lambda}) \tag{7.26}$$

Given these scores, the gradient of the C-Aug sentence model with respect to the augmented parameters, $\boldsymbol{\alpha}$, can be calculated,

$$\begin{aligned}
&\nabla_{\boldsymbol{\alpha}} \mathcal{F}^{\texttt{cml}}(\boldsymbol{\lambda}, \boldsymbol{\alpha}) \qquad (7.27)\\
&\quad = \sum_{i=1}^{n} \nabla_{\boldsymbol{\alpha}} \ln P(\boldsymbol{W}_i | \boldsymbol{O}_i; \boldsymbol{\lambda}, \boldsymbol{\alpha})\\
&\quad = \sum_{i=1}^{n} \left\{ \nabla_{\boldsymbol{\alpha}} \ln \left(\sum_{\boldsymbol{A} \in \mathcal{A}_{\boldsymbol{W}_i}} S(\boldsymbol{O}_i, \boldsymbol{W}_i, \boldsymbol{A}; \boldsymbol{\lambda}, \boldsymbol{\alpha}) \right) - \nabla_{\boldsymbol{\alpha}} \ln Z(\boldsymbol{O}_i; \boldsymbol{\lambda}, \boldsymbol{\alpha}) \right\}\\
&\quad = \sum_{i=1}^{n} \left\{ \frac{\sum_{\boldsymbol{A} \in \mathcal{A}_{\boldsymbol{W}_i}} \nabla_{\boldsymbol{\alpha}} S(\boldsymbol{O}_i, \boldsymbol{W}_i, \boldsymbol{A}; \boldsymbol{\lambda}, \boldsymbol{\alpha})}{\sum_{\boldsymbol{A} \in \mathcal{A}_{\boldsymbol{W}_i}} S(\boldsymbol{O}_i, \boldsymbol{W}_i, \boldsymbol{A}; \boldsymbol{\lambda}, \boldsymbol{\alpha})} - \nabla_{\boldsymbol{\alpha}} \ln Z(\boldsymbol{O}_i; \boldsymbol{\lambda}, \boldsymbol{\alpha}) \right\}\\
&\quad = \sum_{i=1}^{n} \left\{ \frac{\sum_{\boldsymbol{A} \in \mathcal{A}_{\boldsymbol{W}_i}} \sum_{w \in \boldsymbol{W}_i} S(\boldsymbol{O}_i, \boldsymbol{W}_i, \boldsymbol{A}; \boldsymbol{\lambda}, \boldsymbol{\alpha}) \boldsymbol{T}(w, \boldsymbol{O}_i^{(w)} | \boldsymbol{A}; \boldsymbol{\lambda}) \Big)}{\sum_{\boldsymbol{A} \in \mathcal{A}_{\boldsymbol{W}_i}} S(\boldsymbol{O}_i, \boldsymbol{W}_i, \boldsymbol{A}; \boldsymbol{\lambda}, \boldsymbol{\alpha})} - \nabla_{\boldsymbol{\alpha}} \ln Z(\boldsymbol{O}_i; \boldsymbol{\lambda}, \boldsymbol{\alpha}) \right\}\\
&\quad = \sum_{i=1}^{n} \sum_{\boldsymbol{A} \in \mathcal{A}_{\boldsymbol{W}_i}} \sum_{w \in \boldsymbol{W}_i} P(\boldsymbol{A} | \boldsymbol{W}_i, \boldsymbol{O}_i; \boldsymbol{\lambda}, \boldsymbol{\alpha}) \boldsymbol{T}(w, \boldsymbol{O}_i^{(w)} | \boldsymbol{A}; \boldsymbol{\lambda}) \Big) \qquad (7.28)\\
&\qquad\qquad - \sum_{i=1}^{n} \nabla_{\boldsymbol{\alpha}} \ln Z(\boldsymbol{O}_i; \boldsymbol{\lambda}, \boldsymbol{\alpha})
\end{aligned}$$

Using the definition of the normalisation term in equation (7.19), the normalisation derivative can be written as,

$$\begin{aligned}
\nabla_{\boldsymbol{\alpha}} \ln Z(\boldsymbol{O}_i; \boldsymbol{\lambda}, \boldsymbol{\alpha}) &= \frac{1}{Z(\boldsymbol{O}_i; \boldsymbol{\lambda}, \boldsymbol{\alpha})} \sum_{\boldsymbol{W} \in \mathcal{W}} \sum_{\boldsymbol{A} \in \mathcal{A}_{\boldsymbol{W}}} \nabla_{\boldsymbol{\alpha}} S(\boldsymbol{O}_i, \boldsymbol{W}, \boldsymbol{A}; \boldsymbol{\lambda}, \boldsymbol{\alpha})\\
&= \frac{1}{Z(\boldsymbol{O}_i; \boldsymbol{\lambda}, \boldsymbol{\alpha})} \sum_{\boldsymbol{W} \in \mathcal{W}} \sum_{\boldsymbol{A} \in \mathcal{A}_{\boldsymbol{W}}} \sum_{w \in \boldsymbol{W}} S(\boldsymbol{O}_i, \boldsymbol{W}, \boldsymbol{A}; \boldsymbol{\lambda}, \boldsymbol{\alpha}) \boldsymbol{T}(w, \boldsymbol{O}_i^{(w)} | \boldsymbol{A}; \boldsymbol{\lambda})\\
&= \sum_{\boldsymbol{W} \in \mathcal{W}} \sum_{\boldsymbol{A} \in \mathcal{A}_{\boldsymbol{W}}} \sum_{w \in \boldsymbol{W}} P(\boldsymbol{W}, \boldsymbol{A} | \boldsymbol{O}_i; \boldsymbol{\lambda}, \boldsymbol{\alpha}) \boldsymbol{T}(w, \boldsymbol{O}_i^{(w)} | \boldsymbol{A}; \boldsymbol{\lambda}) \qquad (7.29)
\end{aligned}$$

Finally, combining the results in equations (7.28) and (7.29), the CML derivative for C-Aug sentence models can be calculated,

$$\begin{aligned}
\nabla_{\boldsymbol{\alpha}} \mathcal{F}^{\texttt{cml}}(\boldsymbol{\lambda}, \boldsymbol{\alpha}) = \sum_{i=1}^{n} \Bigg[&\sum_{\boldsymbol{A} \in \mathcal{A}_{\boldsymbol{W}_i}} \sum_{w \in \boldsymbol{W}_i} P(\boldsymbol{A} | \boldsymbol{W}_i, \boldsymbol{O}_i; \boldsymbol{\lambda}, \boldsymbol{\alpha}) \boldsymbol{T}(w, \boldsymbol{O}_i^{(w)} | \boldsymbol{A}; \boldsymbol{\lambda}) \Big) \qquad (7.30)\\
&- \sum_{\boldsymbol{W} \in \mathcal{W}} \sum_{\boldsymbol{A} \in \mathcal{A}_{\boldsymbol{W}}} \sum_{w \in \boldsymbol{W}} P(\boldsymbol{W}, \boldsymbol{A} | \boldsymbol{O}_i; \boldsymbol{\lambda}, \boldsymbol{\alpha}) \boldsymbol{T}(w, \boldsymbol{O}_i^{(w)} | \boldsymbol{A}; \boldsymbol{\lambda}) \Bigg]
\end{aligned}$$

The CML gradient for C-Aug sentence models is almost identical to the gradient for the simpler C-Aug models discussed in section 7.1. The main difference between the two is that C-Aug sentence models consider the sufficient statistics for all possible model alignments (weighted by the posterior probabilities of those alignments).

Similarly to the sentence-based objective function calculations, the gradient is often intractable in the form given in equation (7.30). However, using the numerator and denominator lattices, $\boldsymbol{L}^{\texttt{num}}$ and $\boldsymbol{L}^{\texttt{den}}$, defined in the previous section, the efficiency of derivative calculations can be significantly improved. In particular, by noting that an arc's contribution towards the gradient is given by the arc's sufficient statistics weighted by the augmented model posterior probability of the arc given the observation sequence, the summations over hypothesised transcriptions and alignments can be simplified to a simple summation over lattice arcs. Equation (7.30) can thus be rewritten as,

$$\nabla_{\boldsymbol{\alpha}}\mathcal{F}^{\texttt{cml}}(\boldsymbol{\lambda},\boldsymbol{\alpha}) = \sum_{i=1}^{n}\left[\sum_{w,a\in\boldsymbol{L}_i^{\texttt{num}}} P^{\texttt{num}}(w,a|\boldsymbol{O}_i;\boldsymbol{\lambda},\boldsymbol{\alpha})\boldsymbol{T}(w,\boldsymbol{O}_i^{(w)}|a;\boldsymbol{\lambda}) - \sum_{w,a\in\boldsymbol{L}_i^{\texttt{den}}} P^{\texttt{den}}(w,a|\boldsymbol{O}_i;\boldsymbol{\lambda},\boldsymbol{\alpha})\boldsymbol{T}(w,\boldsymbol{O}_i^{(w)}|a;\boldsymbol{\lambda})\right] \quad (7.31)$$

where $w,a \in \boldsymbol{L}_i^{\texttt{den}}$ denotes a phone label/alignment pair selected from the lattice $\boldsymbol{L}_i^{\texttt{den}}$. The lattice-based numerator and denominator posterior probabilities, $P^{\texttt{num}}(w,a|\boldsymbol{O}_i;\boldsymbol{\lambda},\boldsymbol{\alpha})$ and $P^{\texttt{den}}(w,a|\boldsymbol{O}_i;\boldsymbol{\lambda},\boldsymbol{\alpha})$, are calculated using the C-Aug model scores in the numerator and denominator lattice arcs (section 7.4.2).[6] These scores are normalised using the hypotheses in the lattices $\boldsymbol{L}_i^{\texttt{num}}$ and $\boldsymbol{L}_i^{\texttt{den}}$.

Note that since, in many cases, the numerator lattice will contain only a single Viterbi-aligned phone sequence (the correct transcription), the phone/alignment posterior probabilities in the numerator term of equation (7.31) can be safely ignored since they are all one. However, explicit calculation of these posteriors has the advantage of providing a natural framework for handling multiple pronunciations and alignments of the correct sentence: simply define a numerator lattice that includes these alternative pronunciations and alignments, and use equation (7.31) to perform the calculations. A second advantage of retaining the phone/alignment posteriors is that the algorithms for calculating the gradient contributions from the numerator and denominator terms are identical and so can be shared.

To summarise, this section has described an efficient lattice-based method for calculating the CML objective function, $\mathcal{F}^{\texttt{cml}}(\boldsymbol{\lambda},\boldsymbol{\alpha})$, and its gradient, $\nabla_{\boldsymbol{\alpha}}\mathcal{F}^{\texttt{cml}}(\boldsymbol{\lambda},\boldsymbol{\alpha})$. Substituting these into a standard gradient-based maximisation algorithm allows the augmented parameters, $\boldsymbol{\alpha}$, to be estimated according to the conditional maximum likelihood criterion. In this work, scaled conjugate gradient (SCG) [83, 84] optimisation was used.

[6] Note that the numerator posterior probability of the phone alignments given the transcriptions, $P(\boldsymbol{A}|\boldsymbol{W},\boldsymbol{O};\boldsymbol{\lambda},\boldsymbol{\alpha})$ in equation (7.30), is written as a joint phone/alignment posterior, $P^{\texttt{num}}(w,a|\boldsymbol{O};\boldsymbol{\lambda},\boldsymbol{\alpha})$ in equation (7.31). This is possible because the numerator lattice used to calculate the arc posterior probabilities contains only the correct sentence transcription. This means that $P^{\texttt{num}}(\boldsymbol{W}|\boldsymbol{O};\boldsymbol{\lambda},\boldsymbol{\alpha})$ and $P^{\texttt{num}}(w|\boldsymbol{O};\boldsymbol{\lambda},\boldsymbol{\alpha})$ are both one.

7.4.4 Minimum phone error estimation

In the previous section conditional maximum likelihood estimation of C-Aug sentence models was discussed. This is similar to the MMI criterion used for training generative models and attempts to maximise the expected log-posterior probability of the correct sentence transcriptions. However, the most common performance measure for speech recognition is the word error rate (or, in the case of this work, the phone error rate).[7] This leads to a mismatch between the training criterion and the scoring function used to evaluate recognition performance. This mismatch may affect recognition accuracy.

To minimise the mismatch between training and scoring, training can be performed within the minimum Bayes' risk (MBR) framework. As discussed in section 2.1.3, given an application-dependent posterior distribution and loss function, MBR training attempts to minimise the expected loss of the training data. One common form of MBR training for speech recognition is the MPE training criterion [96,97], which minimises an approximation to the phone error rate. MPE training is usually expressed in terms of maximising an approximation to the phone accuracy of the training data,

$$\begin{aligned}\mathcal{F}^{\texttt{mpe}}(\boldsymbol{\lambda},\boldsymbol{\alpha}) &= \sum_{\mathcal{T}'} P(\mathcal{T}'|\mathcal{O};\boldsymbol{\lambda},\boldsymbol{\alpha})A(\mathcal{T}',\mathcal{T}) \\ &= \sum_{i=1}^{n}\sum_{\boldsymbol{W}\in\mathcal{W}} P(\boldsymbol{W}|\boldsymbol{O}_i;\boldsymbol{\lambda},\boldsymbol{\alpha})A(\boldsymbol{W},\boldsymbol{W}_i) \end{aligned} \tag{7.32}$$

where $A(\boldsymbol{W},\boldsymbol{W}_i)$ is the phone accuracy of the hypothesised transcriptions, $\boldsymbol{W}$, given the correct transcriptions, $\boldsymbol{W}_i$. Similarly to the conditional maximum likelihood criterion considered previously, the summation over all possible phone transcriptions is often simplified using a lattice that contains only the most likely phone transcriptions. This lattice is known as the hypothesis lattice.

In this work a gradient-based optimisation algorithm is used to maximise the C-Aug model MPE criterion. This uses the objective function derivatives with respect to the

[7] Recognition performance is usually calculated using a dynamic programming algorithm that calculates the number of deletions, substitutions and insertions in the hypothesised transcription, compared to the correct transcription. Phone recognition accuracy is then calculated using the formula,

$$\text{phone accuracy } (\%) = \frac{\#\text{ phones} - \#\text{ deletions} - \#\text{ substitutions} - \#\text{ insertions}}{\#\text{ phones}}$$

where # phones is the number of phones in the reference transcription. Word accuracy is calculated similarly, except that the number of word substitutions, insertions and deletions words are used.

augmented parameters, $\boldsymbol{\alpha}$, and is calculated using the expression,

$$
\begin{aligned}
&\nabla_{\boldsymbol{\alpha}} \mathcal{F}^{\texttt{mpe}}(\boldsymbol{\lambda}, \boldsymbol{\alpha}) \\
&\quad = \sum_{i=1}^{n} \sum_{\boldsymbol{W} \in \mathcal{W}} A(\boldsymbol{W}, \boldsymbol{W}_i) \nabla_{\boldsymbol{\alpha}} P(\boldsymbol{W}|\boldsymbol{O}_i; \boldsymbol{\lambda}, \boldsymbol{\alpha}) \\
&\quad = \sum_{i=1}^{n} \sum_{\boldsymbol{W} \in \mathcal{W}} A(\boldsymbol{W}, \boldsymbol{W}_i) P(\boldsymbol{W}|\boldsymbol{O}_i; \boldsymbol{\lambda}, \boldsymbol{\alpha}) \nabla_{\boldsymbol{\alpha}} \ln P(\boldsymbol{W}|\boldsymbol{O}_i; \boldsymbol{\lambda}, \boldsymbol{\alpha}) \\
&\quad = \sum_{i=1}^{n} \Bigg[\sum_{\boldsymbol{W} \in \mathcal{W}} \sum_{\boldsymbol{A} \in \mathcal{A}_{\boldsymbol{W}}} \sum_{w \in \boldsymbol{W}} P(\boldsymbol{W}|\boldsymbol{O}_i; \boldsymbol{\lambda}, \boldsymbol{\alpha}) A(\boldsymbol{W}, \boldsymbol{W}_i) P(\boldsymbol{A}|\boldsymbol{W}, \boldsymbol{O}_i; \boldsymbol{\lambda}, \boldsymbol{\alpha}) \boldsymbol{T}(w, \boldsymbol{O}_i^{(w)}|\boldsymbol{A}; \boldsymbol{\lambda}) \\
&\qquad - \Bigg[\sum_{\boldsymbol{W} \in \mathcal{W}} P(\boldsymbol{W}|\boldsymbol{O}_i; \boldsymbol{\lambda}, \boldsymbol{\alpha}) A(\boldsymbol{W}, \boldsymbol{W}_i) \Bigg] \times \\
&\qquad\qquad \Bigg[\sum_{\boldsymbol{W}' \in \mathcal{W}} \sum_{\boldsymbol{A} \in \mathcal{A}_{\boldsymbol{W}'}} \sum_{w \in \boldsymbol{W}'} P(\boldsymbol{W}', \boldsymbol{A}|\boldsymbol{O}_i; \boldsymbol{\lambda}, \boldsymbol{\alpha}) \boldsymbol{T}(w, \boldsymbol{O}_i^{(w)}|\boldsymbol{A}; \boldsymbol{\lambda}) \Bigg] \Bigg] \\
&\quad = \sum_{i=1}^{n} \sum_{\boldsymbol{W} \in \mathcal{W}} \sum_{\boldsymbol{A} \in \mathcal{A}_{\boldsymbol{W}}} \sum_{w \in \boldsymbol{W}} \Big[A(\boldsymbol{W}, \boldsymbol{W}_i) - A^{\texttt{avg}}(\boldsymbol{W}_i) \Big] \times \\
&\qquad\qquad P(\boldsymbol{W}, \boldsymbol{A}|\boldsymbol{O}_i; \boldsymbol{\lambda}, \boldsymbol{\alpha}) \boldsymbol{T}(w, \boldsymbol{O}_i^{(w)}|\boldsymbol{A}; \boldsymbol{\lambda}) \Big)
\end{aligned}
\tag{7.33}
$$

where $A^{\texttt{avg}}(\boldsymbol{W}_i)$ is the weighted-average phone accuracy of all (hypothesised) sentences. By approximating the sentence phone accuracy as the sum of the accuracies of the individual phones, both the objective function in equation (7.32) and the derivative in equation (7.33) can be expressed using efficient lattice-based algorithms [96, 97]. These are identical to the MPE estimation algorithms for HMMs in [96]. The difference between the two approaches lies in the hypothesis lattice arc scores: MPE estimation of HMMs uses arcs scored with HMM phone likelihoods, whereas C-Aug model estimation uses the unnormalised C-Aug model scores defined in section 7.4.2.

Before considering C-Aug sentence model recognition, it is worth mentioning the approximation used to calculate the sentence phone accuracies during MPE training. The sentence accuracy, $A(\boldsymbol{W}, \boldsymbol{W}_i)$ is typically approximated as a sum of the accuracies of the individual phones within that sentence. These 'phone accuracies' are usually calculated using the equation,

$$
A(w) = \max_{w' \in \boldsymbol{W}_i} \begin{cases} -1 + 2e(w, w') & \text{if } w' = w \\ -1 + e(w, w') & \text{otherwise} \end{cases}
\tag{7.34}
$$

where $e(w, w')$ calculates the proportion of the reference phone duration, w', that is overlapped by the hypothesised phone, w. With perfect alignment between the reference and hypothesised transcriptions, $A(w)$ outputs values of 1, 0 and -1 for a correct phone, a substitution/deletion, and an insertion. The advantage of calculating the individual phone accuracies instead of the sentence accuracy, $A(\boldsymbol{W}, \boldsymbol{W}_i)$, is that $A(w)$ is a local function

of only the current hypothesised phone. This allows calculations to be performed on a phone/arc basis instead of a sentence-basis, allowing lattice-based calculations to be used.

7.4.5 Recognition

Recognition with C-Aug sentence models is performed similarly to C-Aug model inference and proceeds according to Bayes' decision rule,

$$\hat{\boldsymbol{W}} = \arg\max_{\boldsymbol{W}\in\mathcal{W}} P(\boldsymbol{W}|\boldsymbol{O};\boldsymbol{\lambda},\boldsymbol{\alpha}) \tag{7.35}$$

where $\mathcal{W}$ is the set of all possible sentence transcriptions. To reduce the computational cost of C-Aug sentence model recognition, the posteriors in equation (7.35) need not be normalised since the same normalisation term is used by all posteriors. The C-Aug model decision function can therefore be written as,

$$\hat{\boldsymbol{W}} = \arg\max_{\boldsymbol{W}\in\mathcal{W}} \left(\frac{1}{|\mathcal{A}_{\boldsymbol{W}}|} \sum_{\boldsymbol{A}\in\mathcal{A}_{\boldsymbol{W}}} \prod_{w\in\boldsymbol{W}} \hat{p}(\boldsymbol{O}^{(w)}|w,\boldsymbol{A};\boldsymbol{\lambda}) \exp\left(\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{T}(w,\boldsymbol{O}^{(w)}|\boldsymbol{A};\boldsymbol{\lambda})\right) \right) \tag{7.36}$$

In practice, it is computationally expensive to evaluate the C-Aug sentence model posteriors across all possible phone transcriptions and alignments.[8] Instead a hypothesis lattice, $\boldsymbol{L}$, that represents the most likely transcriptions and alignments for the observation sequence, $\boldsymbol{O}$, is often used. For HMM base models, this lattice is typically generated using the Viterbi decoder of a standard speech recognition system.

Using the lattice-weight approach discussed in section 7.4.2, calculation of the most likely sentence transcription in equation (7.36) can be simplified further. Let $\boldsymbol{L}^{\texttt{recog}}$ be the lattice obtained by assigning all arcs in the hypothesis lattice, $\boldsymbol{L}$, the scores, $\hat{p}(\boldsymbol{O}^{(w)}|w,\boldsymbol{A};\boldsymbol{\lambda}) \exp\left(\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{T}(w,\boldsymbol{O}^{(w)}|\boldsymbol{A};\boldsymbol{\lambda})\right)$, where w is the arc phone label and $\boldsymbol{A}$ reflects the arc start and end times. This lattice provides a compact representation of the posterior probabilities of all possible sentences and alignments in the original lattice, $\boldsymbol{L}$. Finally, given the recognition lattice, $\boldsymbol{L}^{\texttt{recog}}$, the most likely phone transcription can be calculated using a Viterbi approximation to the lattice weight. If multiple hypotheses are required, an N-best list or lattice can be generated using multi-token Viterbi decoding [138].

7.5 Language modelling

Standard speech recognitions systems are typically constructed using a combination of generative acoustic models (often HMMs) and statistical language models (usually n-grams), which combine to form the sentence posterior probabilities used during recognition

[8] Note that the dependence of the C-Aug sufficient statistics on whole phones (instead of frames) means that frame-by-frame recognition and pruning (such as that used during HMM and HCRF recognition) cannot be used.

[53,103,137]. In many systems, the acoustic models and language models complement each other by modelling different aspects of the speech process. Acoustic models, trained using a relatively small amount of transcribed audio data, capture the short-term dependencies between observations within short speech segments (often corresponding to phones). In contrast, language models are typically trained using large corpora of written text and model the long-term relationships between words in phrases and sentences.

In previous sections, C-Aug sentence models were defined as a new form of discriminative acoustic model for speech recognition. Unfortunately, since they directly model the posterior probabilities of the sentence transcriptions, they cannot be combined with a language model in the same way as generative acoustic models. Instead, in this work a system combination approach is used. This proceeds as follows. First, the C-Aug acoustic model is used to generate a lattice that contains a number of hypothesised transcriptions. Then, for each sentence in this lattice, a language model score is calculated. This is combined with the C-Aug model acoustic score using a simple linear combination,[9]

$$\text{sentence score}(\boldsymbol{W}) = \ln P(\boldsymbol{W}|\boldsymbol{O};\boldsymbol{\lambda},\boldsymbol{\alpha}) + \beta \ln P(\boldsymbol{W}) \tag{7.37}$$

where $\boldsymbol{W}$ is a hypothesised transcription, $\ln P(\boldsymbol{W}|\boldsymbol{O};\boldsymbol{\lambda},\boldsymbol{\alpha})$ is the C-Aug acoustic model score, and $\ln P(\boldsymbol{W})$ is the language model score. The language model scale, β, is normally determined empirically using a development set. The hypothesis with the largest sentence score is used as the final sentence transcription. Similarly to model estimation and recognition, calculation of language model scores and sentence scores can be performed entirely within a lattice framework.

7.6 Comparison with CRFs and HCRFs

With many similarities between the C-Aug models discussed in this chapter and the CRFs [63] and HCRFs [42, 99] discussed in chapter 2, it is useful to examine the differences between the three approaches. To aid this comparison, the CRF, HCRF and C-Aug model posterior probabilities, $P_{\texttt{crf}}(\boldsymbol{y}|\boldsymbol{O};\boldsymbol{\alpha})$, $P_{\texttt{hcrf}}(y|\boldsymbol{O};\boldsymbol{\alpha})$ and $P_{\texttt{caug}}(y|\boldsymbol{O};\boldsymbol{\lambda},\boldsymbol{\alpha}')$,

[9] Note that the linear combination of C-Aug model and language model scores in equation (7.37) can also be interpreted as adding the language model score as an extra element in the C-Aug sentence model sufficient statistics. The augmented parameter associated with this element is equal to β. During parameter estimation, this may either be fixed (as described here) or estimated similarly to the other augmented parameters (c.f. the inclusion of the confusion network posterior ratio in section 6.4).

are restated below,

$$P_{\texttt{crf}}(\boldsymbol{y}|\boldsymbol{O};\boldsymbol{\alpha}) = \frac{1}{Z(\boldsymbol{O};\boldsymbol{\alpha})}\exp\Big(\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{T}(y_t, y_{t-1}, \boldsymbol{O})\Big) \tag{7.38}$$

$$P_{\texttt{hcrf}}(y|\boldsymbol{O};\boldsymbol{\alpha}) = \frac{1}{Z(\boldsymbol{O};\boldsymbol{\alpha})}\sum_{\boldsymbol{\theta}\in\boldsymbol{\Theta}}\exp\Big(\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{T}(y, \boldsymbol{\theta}, \boldsymbol{O})\Big) \tag{7.39}$$

$$P_{\texttt{caug}}(y|\boldsymbol{O};\boldsymbol{\lambda},\boldsymbol{\alpha}') = \frac{1}{Z(\boldsymbol{O};\boldsymbol{\lambda},\boldsymbol{\alpha}')}\exp\big(\boldsymbol{\alpha}'^{\mathrm{T}}\boldsymbol{T}'(y, \boldsymbol{O};\boldsymbol{\lambda})\big) \tag{7.40}$$

where the C-Aug model natural parameters and sufficient statistics are defined in terms of the base model likelihoods, $\boldsymbol{L}(y, \boldsymbol{O};\boldsymbol{\lambda})$, and the augmented model sufficient statistics, $\boldsymbol{T}(y, \boldsymbol{O};\boldsymbol{\lambda})$, using the expressions,

$$\boldsymbol{\alpha}' = \begin{bmatrix} \mathbf{1} \\ \boldsymbol{\alpha} \end{bmatrix} \qquad\qquad \boldsymbol{T}'(y, \boldsymbol{O};\boldsymbol{\lambda}) = \begin{bmatrix} \boldsymbol{L}(y, \boldsymbol{O};\boldsymbol{\lambda}) \\ \boldsymbol{T}(y, \boldsymbol{O};\boldsymbol{\lambda}) \end{bmatrix} \tag{7.41}$$

Examining equations (7.38)–(7.40), is is clear that CRFs, HCRFs and C-Aug models share many of the same features. In particular, comparing the C-Aug model in equation (7.40) with the CRF in equation (7.38), it is clear that C-Aug models are simply a special form of conditional random field with a particular form of sufficient statistics.[10] C-Aug models can therefore be described as a systematic method for defining CRF feature-vectors. More generally, however, C-Aug models are a systematic approach for defining sufficient statistics for discriminative exponential models. The choice of statistics in C-Aug models enables them to model a wide range of temporal and spatial dependencies. This modelling power arises from the use of latent-variable generative models to calculate complex statistics that capture a wide range of temporal and spatial dependencies. By confining the use of latent-variables to the calculation of the sufficient statistics, C-Aug models retain the convex structure of exponential models, allowing conditional maximum likelihood (CML) estimation of the augmented parameters to converge to a global maximum. Note that this is only true when the base model parameters of the C-Aug model are fixed (when optimisation of the base model parameters is performed, the C-Aug model sufficient statistics vary, resulting in a non-convex objective function).

The use of latent-variables in discriminative models is not restricted to C-Aug models. Consider, for example, hidden conditional random fields (HCRFs). As illustrated in equation (7.39), these explicitly introduce latent-variables into the posterior probability calculations using a similar approach to that used for generative models (section 2.1.2). The advantage of this approach is that, by introducing conditional-independence assumptions, HCRFs can model a wide range of a complex distributions using only a limited

[10] CRFs typically predict class labels for individual observations, $\boldsymbol{o}_t \in \boldsymbol{O}$. When C-Aug model sufficient statistics are used, however, the sufficient statistics collapse the whole observation sequence into a single feature-vector. The CRF views this feature-vector as a single 'compound-observation' and predicts a single class label.

number of sufficient statistics. Unfortunately, introducing latent-variables directly in the posterior probability calculations breaks the convexity of original exponential model. Conditional maximum likelihood estimation of HCRF parameters may therefore only converge to a local maximum of the objective function. This causes HCRF training to be sensitive both to the initialisation of parameters and to the update algorithms used during training (c.f. L-BFGS versus SGD versus R-PROP HCRF optimisation in [42, 74]). In contrast, by restricting the use of latent-variables to the calculation of sufficient statistics, C-Aug models retain the convex objective function of standard exponential models and so avoid these difficulties.

7.7 Summary

In this chapter, conditional augmented (C-Aug) models were proposed as a powerful form of discriminative statistical model. C-Aug models use the same sufficient statistics as standard generative augmented models but directly model class posteriors instead of utterance likelihoods. This simplifies the model normalisation calculations by replacing the integration over all observation sequences with a simple summation over class labels. Class label posterior probabilities can therefore be calculated efficiently. A method of C-Aug model parameter estimation, based upon the conditional maximum likelihood criterion, was presented. Finally, an extension was proposed that allows sequences of C-Aug models (representing, for example, phones) to be combined to form complex models of sentences. A number of lattice-based calculations for these extended models were derived. C-Aug sentence models can be applied to a number of different tasks, including continuous speech recognition.

8

Experimental Results

In this chapter, preliminary speech recognition experiments using augmented models (chapter 4) and conditional augmented models (chapter 7) are presented. To illustrate the properties of these models, data from two different databases was used. The first set of experiments uses the acoustic code-breaking approach of [126] to extract a set of confusable words from a 400 hour subset of the LDC Fisher corpus [31]. The resulting word pairs are used to train and evaluate augmented models, both within a cross-validation framework and as a method of rescoring transcriptions from a large vocabulary continuous speech recognition (LVCSR) task. The second database is the TIMIT phone database [38, 44]. This is used for both classification and recognition experiments.

8.1 Cross-validation experiments

The database used for initial LVCSR experiments is the `fsh2004sub` subset of the LDC Fisher data [31]. This consists of 400 hours of English-language conversational telephone speech (CTS), balanced for both gender and topics of conversation. The acoustic data was parameterised using 13 PLP features, along with their first, second and third derivatives. The resulting 52-dimension acoustic feature-spaces were then projected down to 39 dimensions using heteroscedastic linear discriminant analysis (HLDA) [34,62]. Vocal-tract length normalisation (VTLN) [70,124] and cepstral mean and variance normalisations (CMN and CVN) [43] were also applied. Next, using this data, a baseline LVCSR system was constructed. This uses ML-estimated, gender-independent, cross-word triphone HMMs as acoustic models. After decision-tree state-tying the model set contains 6,000 tied states,

each with 28 mixture-components. Given the ML-estimated HMMs, word lattices were generated for each training utterance using Viterbi decoding. A weakened language model was used to increase the number of confusions present in the lattices and prevent lattices from being dominated by a single hypothesis (see section 2.1.3 for more details). In this work, a heavily pruned bigram language model was used. Further details of the acoustic models, language models and lattice generation are given in [31, 72]. Confusion networks were generated using a trigram language model.

Next, using the acoustic code-breaking method described in the section 6.4, pairs of confusable words were identified and extracted from the data set. For each pair, the number of examples of each word were equalised by sampling to avoid bias (random selection yields an expected error rate of 50%). Binary classifiers were then trained for each pair using the acoustic data from examples of that confusable pair. Unless otherwise noted, classifiers used in the experiments are based upon ML-estimated HMMs with three emitting states and GMM output distributions. When MMI base models were trained, uniform class prior distributions were used (since there are an identical number of training examples in each class). The GMM output distributions typically have between one and sixteen Gaussian distributed mixture-components, each with a diagonal covariance matrix. Finally, it is worth noting that the Fisher data used in these experiments was transcribed using the 'quick transcription' process discussed in [16]. Transcriptions of training utterances are therefore good but not necessarily perfect. Although this affects many of the LVCSR experiments in this chapter, the overall effect on performance is believed to be minor.

8.1.1 First-order augmented models

Initial experiments with first-order augmented models were performed within an eight-fold cross-validation framework. Many score-spaces and word-pairs were evaluated. A summary of the score-spaces used is given in table 8.1. Results for a small selection of word pairs are shown in table 8.2.

Table 8.1: Summary of HMM score-spaces

Score-space	Elements	Description
$\boldsymbol{\phi}^{\mathtt{LLR}}(\boldsymbol{O};\boldsymbol{\lambda})$	LLR	Log-likelihood ratio score-space
$\boldsymbol{\phi}^{\mathtt{F}}(\boldsymbol{O};\boldsymbol{\lambda})$	∇	Fisher score-space
$\boldsymbol{\phi}^{\mathtt{LL}}(\boldsymbol{O};\boldsymbol{\lambda})$	LLR+∇	Augmented model score-space
$\boldsymbol{\phi}^{\mathtt{CN}}(\boldsymbol{O};\boldsymbol{\lambda})$	LLR+∇+CN	Augmented model score-space + confusion network

where: LLR = log-likelihood ratio of the base models
∇ = derivatives with respect to the c_{jm}, $\boldsymbol{\mu}_{jm}$ and $\boldsymbol{\Sigma}_{jm}$
CN = confusion network posterior ratio

For each example, multiple baselines are presented. The first is the error rate ob-

Table 8.2: Cross validation results for HMM base models (% error)

Classifier	Training criterion	Score-space	# Base model comp. 1	2	4
AND/IN (12,004 examples)					
CN	n/a	n/a	—22.7—		
HMM	ML	n/a	42.0	41.9	41.7
HMM	MMI	n/a	42.0	41.2	41.7
SVM	MM	$\boldsymbol{\phi}^{\mathtt{LLR}}(\boldsymbol{O};\boldsymbol{\lambda})$	41.5	41.8	41.5
SVM	MM	$\boldsymbol{\phi}^{\mathtt{F}}(\boldsymbol{O};\boldsymbol{\lambda})$	39.6	39.8	39.5
SVM	MM	$\boldsymbol{\phi}^{\mathtt{LL}}(\boldsymbol{O};\boldsymbol{\lambda})$	40.0	39.7	39.7
AUG	MM	$\boldsymbol{\phi}^{\mathtt{LL}}(\boldsymbol{O};\boldsymbol{\lambda})$	40.0	39.9	39.5
AUG	MM	$\boldsymbol{\phi}^{\mathtt{CN}}(\boldsymbol{O};\boldsymbol{\lambda})$	22.7	23.8	25.5
CAN/CAN'T (7,522 examples)					
CN	n/a	n/a	—21.5—		
HMM	ML	n/a	13.3	11.3	11.0
HMM	MMI	n/a	13.2	11.2	10.4
SVM	MM	$\boldsymbol{\phi}^{\mathtt{LLR}}(\boldsymbol{O};\boldsymbol{\lambda})$	13.2	11.4	10.8
SVM	MM	$\boldsymbol{\phi}^{\mathtt{F}}(\boldsymbol{O};\boldsymbol{\lambda})$	10.7	9.7	9.2
SVM	MM	$\boldsymbol{\phi}^{\mathtt{LL}}(\boldsymbol{O};\boldsymbol{\lambda})$	10.7	9.5	9.2
AUG	MM	$\boldsymbol{\phi}^{\mathtt{LL}}(\boldsymbol{O};\boldsymbol{\lambda})$	11.1	9.9	9.1
AUG	MM	$\boldsymbol{\phi}^{\mathtt{CN}}(\boldsymbol{O};\boldsymbol{\lambda})$	9.3	8.7	8.4
KNOW/NO (8,950 examples)					
CN	n/a	n/a	—16.9—		
HMM	ML	n/a	31.0	35.3	27.7
HMM	MMI	n/a	27.5	29.3	27.1
SVM	MM	$\boldsymbol{\phi}^{\mathtt{LLR}}(\boldsymbol{O};\boldsymbol{\lambda})$	31.5	32.9	27.3
SVM	MM	$\boldsymbol{\phi}^{\mathtt{F}}(\boldsymbol{O};\boldsymbol{\lambda})$	24.9	23.3	22.2
SVM	MM	$\boldsymbol{\phi}^{\mathtt{LL}}(\boldsymbol{O};\boldsymbol{\lambda})$	25.1	23.0	22.0
AUG	MM	$\boldsymbol{\phi}^{\mathtt{LL}}(\boldsymbol{O};\boldsymbol{\lambda})$	25.1	23.4	22.4
AUG	MM	$\boldsymbol{\phi}^{\mathtt{CN}}(\boldsymbol{O};\boldsymbol{\lambda})$	14.0	14.1	15.8

tained from standard confusion network (CN) decoding. This is based upon both acoustic information and word context (from the trigram language model) and measures the accuracy of the LVCSR system from which the word pairs were extracted. For the word pair CAN/CAN'T, 21.5% of the confusions were misclassified by the LVCSR system. The second and third baselines are pairwise ML and MMI HMMs, trained using the acoustic information from each pair. For CAN/CAN'T, these achieved an error rate significantly below that of CN decoding (10.4% for MMI versus 21.5% for CN decoding), with MMI HMMs generally outperforming ML models. Although, in general, pairwise classifiers outperformed CN decoding, this was not always the case since the pairwise HMM classifiers lacked the word context information of the confusion network language model. When words are acoustically similar, this information can be very important. Consider, for example, the homophone word-pair KNOW/NO. Standard phone-based speech recognition systems cannot separate the two words using acoustic information alone. Instead,

a language model is used. In contrast, when ML and MMI estimated word HMMs were trained using only the acoustic data, error rates of 27.7% for ML and 27.1% for MMI were achieved. This ability to separate 'identical' words is believed to arise from word classifiers basing decisions on the differences in the pronunciation of the start and end of words (arising from longer-range word context). When explicit word context information (in the form of a language model) is available, classification performance improves significantly. For KNOW/NO, confusion network decoding yields an error rate of just 16.9% (compared to 27.1% for MMI HMMs). For acoustically similar words, or other situations when word context is important, CN decoding was often found to outperform pairwise HMMs.

Next, SVMs were trained using score-spaces constructed from the pairwise ML HMMs. In all cases, the SVM regularisation parameter was determined automatically from the training data using the algorithm in [55]. The simplest classifier, with only two parameters, was the one-dimensional $\boldsymbol{\phi}^{\mathrm{LLR}}(\boldsymbol{O};\boldsymbol{\lambda})$ score-space. As expected, this yielded similar performance to the ML baseline in all cases. The second score-space was the Fisher score-space, $\boldsymbol{\phi}^{\mathrm{F}}(\boldsymbol{O};\boldsymbol{\lambda})$, defined using the gradient-space of a single HMM (trained using examples from both classes). Derivatives with respect to the HMM mixture-component priors, means and variances were used, with gains over the HMM baselines observed in all cases. The third and final SVM score-space was the generative score-space, $\boldsymbol{\phi}^{\mathrm{LL}}(\boldsymbol{O};\boldsymbol{\lambda})$.

Table 8.3: Varying the number of mixture-components for CAN/CAN'T

Mixture-components	Score-space & classifier		
	$\boldsymbol{\phi}^{\mathrm{F}}(\boldsymbol{O};\boldsymbol{\lambda})$ SVM	$\boldsymbol{\phi}^{\mathrm{LL}}(\boldsymbol{O};\boldsymbol{\lambda})$ SVM	$\boldsymbol{\phi}^{\mathrm{LL}}(\boldsymbol{O};\boldsymbol{\lambda})$ AUG
1	10.7	10.7	11.1
2	9.7	9.5	9.9
4	9.2	9.2	9.1
8	9.1	8.6	8.8
16	9.1	8.7	8.7

As shown in tables 8.2 and 8.3, when there are few mixture-components the generative score-space yielded similar error rates to the Fisher score-space. However, as the number of mixture-components was increased, wrap-around became important (section 3.2.2), and generative score-spaces started to outperform equivalent Fisher score-spaces. This is clearly illustrated by the results in table 8.3. Here, the inclusion of the log-likelihood-ratio and the additional derivatives in the generative score-space reduces wrap-around for large numbers of mixture-components, allowing generative score-spaces to outperform Fisher score-spaces. For example, for eight and sixteen mixture-components, generative score-spaces yielded error rates of 8.6% and 8.7%, compared to error rates of 9.1% for the Fisher score-space.

Next, augmented models were estimated using augmented model score-spaces and

variable-margin SVM training (chapter 6). With identical score-spaces to the SVM-trained generative score-spaces discussed previously, augmented models yielded similar, though slightly worse, results. The slight difference in performance arises from the different training criteria: augmented models are estimated using variable-margin SVM training whereas generative kernels utilise the standard SVM criterion. As shown in tables 8.2 and 8.3, variable-margin training, which ensures that the weight associated with the log-likelihood ratio is fixed, results in a small performance penalty in many cases. This is due to the reduced number of parameters trained and the poor discrimination and high confusability of the log-likelihood ratio for many pairs. Whereas standard SVM training can reduce the effect of the log-likelihood ratio feature by assigning it a small weight, augmented models, with their fixed weight of 1.0, are forced to compensate in other ways. This compromise leads to a reduction in performance.

Further experiments were performed to evaluate the benefits of combining augmented models with the original confusion network scores, as discussed in section 6.4. Since the training and test sets are perfectly matched in this cross-validation framework, maximum margin estimation of the system combination weight, β, was used. To illustrate the benefits of system combination, consider the confusable pair KNOW/NO in table 8.2. CN decoding resulted in an error rate of 16.9%; augmented model classification yielded 22.4% error. In this case, CN decoding outperforms pairwise augmented models since it utilises a combination of acoustic and language model information, whereas augmented models use only the acoustic data. When words are acoustically similar, the language model element of the confusion network is important. To realise the complementary benefits of both approaches, CN decoding and augmented model classification can be combined using the system combination approach discussed in section 6.4. For pair KNOW/NO, above, system combination yielded an error rate of 15.8%—an improvement of 1.1% over CN decoding and 6.6% better than the standard augmented model performance.

8.1.2 Higher-order augmented models

In addition to first-order augmented models, second- and higher-order models can be defined. These utilise large score-spaces that contain the log-likelihood ratio and the first- and second-order derivatives of the base models with respect to the means, variances and mixture-component weights. Higher-order derivatives may also be included.

Preliminary experiments were performed using augmented models with HMM base models and second-order score-spaces. In general these showed no reduction in error rate when compared to similar first-order models. In fact, in many cases, the score-space elements associated with second-order derivatives had little or no discriminative power, and instead increased the confusability of the training and test data. This caused a corresponding increase in classification error over both the training and test sets. For example,

the generalisation error for CAN/CAN'T increased from 9.1% to 21.8% and training error increased from 6.1% to 6.6% when the first-order score-space (1,897 dimensional) was augmented with second derivatives of the base model. The resulting second-order score-space was 14,221 dimensional. Smaller second-order score-spaces performed better, but still underperformed similarly sized first-order score-spaces.

At this stage, it is worth considering why augmented model sufficient statistics defined using second-order derivatives of an HMM may not be discriminatory for speech data. The problem is that the high dimensionality of the observation vectors used in speech recognition tasks (typically 39-dimensional) often causes the Gaussian distributions that model them to peak sharply near their means and then decay rapidly with distance. This results in highly polarised state/mixture-component posterior probabilities that are either approximately one or approximately zero. Derivatives of these posteriors are thus zero throughout much of the observation-space.[1] Since second-order derivatives of HMMs are a function of these posterior derivatives (section 4.3.2 and appendix A), they are also almost zero. With all second derivatives being approximately zero, very little discriminatory information is extracted.

For tasks that utilise low-dimensional or discrete observations—such as the simple example described in section 4.3—posterior probabilities (and hence their derivatives) take a much wider spectrum of values. In these circumstances, second-order augmented models may offer benefits when compared to standard first-order models.

8.1.3 Varying the score-space dimensionality

Although the augmented models discussed thus far have been defined using all possible mixture-component prior, mean and variance derivatives of the base model HMM, this is not a requirement. Instead, a subset of derivatives may be selected according to some predefined criterion. In this section, derivatives are selected according to their Fisher ratio.[2] Figure 8.1 illustrates the correlation between increasing numbers of augmented model sufficient statistics and decreasing classification error.

In addition to this correlation between the number of sufficient statistics and error rates, there is also a link between the base model complexity and the augmented model performance. For example, when augmented models are defined using 160 sufficient statistics, a four-mixture-component base model gives an error rate of 10.0%, whereas two-component and single-component base models yield poorer error rates of 10.6% and

[1] Since posteriors are effectively either one or zero, an alternative argument follows from the definition of the second-order 'posterior', $D(\theta_t^{jm}, \theta_\tau^{kn}|\boldsymbol{O};\boldsymbol{\lambda})$ (4.22). If either of the single posteriors is zero, then the joint posterior must also be zero, making the difference zero. However if both posteriors are one, the joint posterior will also be one, resulting in a difference of zero. $D(\theta_t^{jm}, \theta_\tau^{kn}|\boldsymbol{O};\boldsymbol{\lambda})$ is thus everywhere zero.

[2] The Fisher ratio is a measure of the signal to noise ratio, and is calculated using the ratio $\mathrm{tr}(\boldsymbol{W}^{-1}\boldsymbol{B})$, where $\boldsymbol{W}$ and $\boldsymbol{B}$ are diagonal approximations to the within-class and between-class covariances.

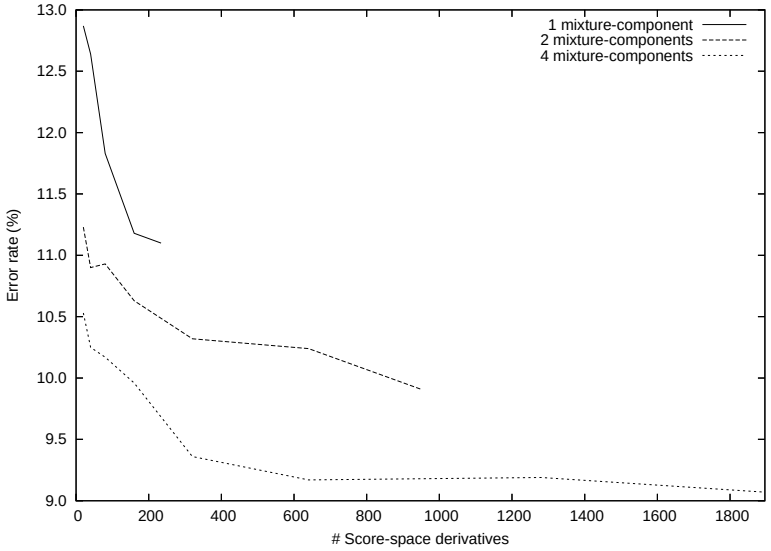


Figure 8.1: Augmented model performance versus number of sufficient statistics for the pair CAN/CAN'T

11.2% respectively. The link between base model complexity and performance arises since complex base models not only provide better statistics that are more closely tuned to the target data distribution, but also offer a wider selection of statistics to choose from. For example, for the four-component base model, 160 statistics are selected from a choice of 1,896 (8% of the total), whereas, the one-component model uses 160 statistics selected from just 234 (68% of the total). Non-discriminatory features can therefore be more easily avoided with complex base models.

8.1.4 Kernelised augmented models

The results in section 8.1.3 show that the performance of augmented models is closely related to the complexity of the base models and the number of augmented model sufficient statistics: the more complex the base models, and the more sufficient statistics, the better the augmented model performance. Unfortunately, calculation of complex base models and statistics can be computationally expensive. Kernelised augmented models (section 4.4) offer a solution. These make use of the standard inhomogeneous polynomial and RBF kernel functions (section 2.2.4) to significantly increase the effective number of sufficient statistics without incurring a proportional cost. This allows models to be defined using a small number of explicitly calculated statistics that are then kernelised to generate a much large set of implicit statistics. Augmented models defined using these implicit statistics are highly flexible and can be calculated without the computational cost associated with explicitly calculated sufficient statistics. The benefits (and disadvantages) of kernelising the sufficient statistics are illustrated in table 8.4.

In table 8.4, a four mixture-component HMM is augmented and kernelised to create

Table 8.4: Kernelised augmented models for the pair CAN/CAN'T (% test/training error)

Score-space elements	Linear	Polynomial		RBF
		$p = 2$	$p = 3$	$\sigma = 1$
40	10.3/9.6	9.9/7.8	10.4/7.2	11.1/5.1
160	10.0/8.5	9.5/5.9	10.4/6.2	11.0/5.1
640	9.2/7.1	9.4/4.6	10.4/5.8	11.0/5.1
1,896 (no selection)	9.1/6.1	10.1/4.8	10.8/6.0	11.0/5.1

a kernelised augmented model. Four different kernel functions are used: a linear kernel (standard augmented model), second- and third-order inhomogeneous polynomials, and a radial basis function (RBF). For small score-spaces with either 40 or 160 derivatives, applying the relatively simple second-order inhomogeneous polynomial kernel function yielded reductions in both the training and test errors. Application of more complex kernels—such as third-order polynomial or RBF kernels—was found to decrease training error whilst allowing test error to increase, a pattern suggestive of over-training. As table 8.4 shows, this over-training becomes increasingly severe as the number of explicitly calculated score-space elements increases, with the 1,896 statistic kernelised augmented model performing significantly worse than the standard model. This trend is unsurprising since the combination of large score-spaces and complex kernels results in extremely flexible statistical models that have many free parameters. When limited quantities of training data are available, such as in the experiments above, parameter estimation (even MM) is likely to lead to an over-fitting of the training data, impairing generalisation performance.

The tendency of kernelised augmented models to over-train limits their use to applications where, for some reason, it is not possible to explicitly calculate much more than 160 sufficient statistics. This may be due to a restrictions such as low power CPUs or limited storage, or due to time constraints, such as those encountered whilst performing real-time classification.

8.1.5 MMI and MM base models

All of the experiments in sections 8.1.1–8.1.4 were based upon augmented models with ML estimated base models. This is not necessary and, instead, MMI or MM (section 6.2.2) base models can be used. Note that the results in this section are preliminary, and were performed using augmented GMMs instead of the augmented HMMs used elsewhere in this thesis. Augmented models were defined using derivatives with respect to the mixture-component means and variances.

Using the confusion-pair cross-validation framework discussed above, augmented GMMs with either ML or MMI estimated base models were evaluated using a number of confusable word pairs. For the word pair KNOW/NO, augmented GMMs with MMI-estimated base models yielded an error rate of 26.3%, compared to 26.4% for ML-estimated base mod-

els, an improvement of just 0.1%. A similar experiment for the word pair CAN/CAN'T yielded no gains. When other word pairs were tested, similar results were obtained. Overall, augmented models with MMI-estimated base models were found to yield no significant performance benefits over augmented models with standard ML-estimated base models.

A second alternative to ML-estimated base models are MM-estimated base models, estimated using the gradient-based approach described in section 6.2.2. Augmented models with MM base models yielded mixed results. Maximum-margin estimation of single-component GMM (Gaussian) base models was found to yield the best results, with a 0.6% gain achieved for an MM-estimated Gaussian augmented with eight derivatives (selected using Fisher ratios). Unfortunately, such results were found to be anomalous, with no gains recorded for many other augmented Gaussians with MM base models. Two-, four- and eight-mixture-component augmented GMMs performed no better, with gradient-based MM model estimation immediately converging to a local maximum on the highly complex error surface (see section 6.2.2). Approximate MM estimation of the base model (extending the first-order augmented model to a second-order augmented model) was also found to yield no improvements in performance. This result is of no surprise given the previous results in section 8.1.2 (second-order augmented models).

8.2 LVCSR rescoring

In the previous section, experiments were conducted using a cross-validation framework and the `fsh2004sub` data set. Maximum-margin augmented models were shown to perform well at disambiguating confusable word pairs extracted from confusion networks. In this section, a second set of experiments are discussed that show that small benefits can potentially be obtained within a standard evaluation-based framework.

Similarly to the cross-validation experiments in section 8.1, pairwise augmented models were defined using 3-state, 4-mixture-component HMM base models and first-order derivatives with respect to the means, variances and mixture-component priors. These models were then estimated using examples extracted from the `fsh2004sub` data. Initially, twenty sets of augmented models were trained to disambiguate the twenty most common LVCSR confusions.[3] The scores from these models were combined with the confusion network scores using the methods in section 6.4. Next, the `eval03` development set was used to determine the order in which the models were used to rescore the baseline LVCSR transcription. The confusion network weight β was optimised empirically using the development set. Finally, performance was evaluated using the `eval02` and `eval04`

[3]Common confusions were located by analysing the LVCSR performance on the `eval03` development set. These confusions were then cross-referenced with the number of occurrences of the confusion in the training set to ensure that there were sufficient training examples for robust augmented model parameter estimation.

test sets. For easy reference, baseline results for the three data sets are given in table 8.5.

Table 8.5: Baseline LVCSR results for the `eval02`, `eval03` and `eval04` data sets

Decoding	`eval02`	`eval03`	`eval04`
Viterbi	32.7	30.8	29.9
Confusion network	32.1	30.1	29.2

8.2.1 Development set: eval03

In these experiments, the `eval03` corpus was used as a development set. This contains six hours of English CTS data, split into 36 conversations of Fisher data (`fsh`) and 36 conversations of Switchboard II phase 5 data (`s25`). Using the LVCSR system described in section 8.1, along with a trigram language model, baseline Viterbi and CN decoding transcriptions were generated. These were then scored against the correct transcriptions and yielded word error rates of 30.8% and 30.1% respectively.

Starting with the CN decoding transcription, augmented models were used to construct a rescored transcription by applying each new augmented model to the transcription obtained after rescoring with all previous augmented models. By retaining and scoring intermediate transcriptions, performances of individual augmented models were calculated. This allowed the classifiers to be sorted, best first, such that optimum performance was achieved for any number of augmented models. The ten best augmented models were then selected (details of the word pairs used are given in appendix D). To maximise system performance, pairwise augmented models were combined with the confusion network posterior ratios using the linear combination method discussed in section 6.4. The confusion network weight, β, was varied over four orders of magnitude. The resulting transcriptions were then scored, and percentage improvements plotted (figure 8.2).

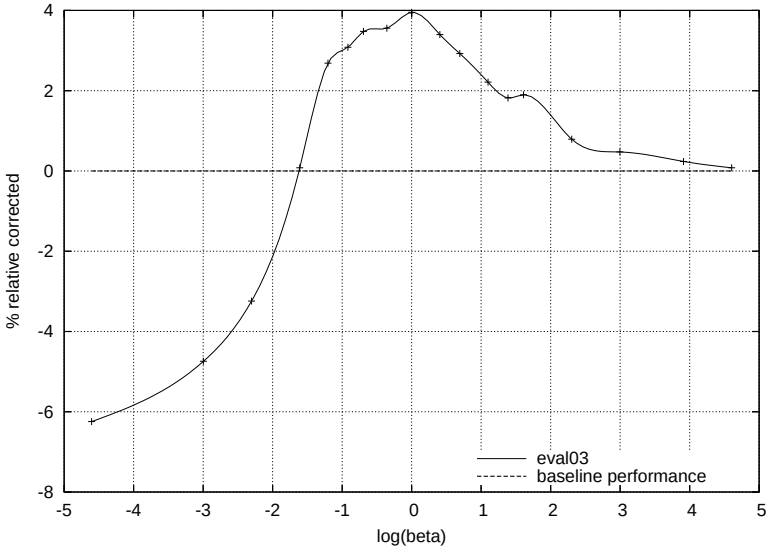


Figure 8.2: Effects of β on performance of the `eval03` development set

When β is small, the contribution from the confusion network posteriors is negligible, resulting in a transcription that is based solely upon the augmented model scores. The resulting performance is poor since language model information (an important aspect of this task) is ignored. At the opposite end of the spectrum, as β becomes large, augmented model scores are small in relation to the scaled confusion network scores. Performance therefore tends towards the accuracy of the baseline confusion networks. For β between 0.3 and 5.0, both augmented model scores and confusion network scores contributed towards the final classification result, yielding improvements over the baseline. Gains of up to 4.0% over standard confusion network decoding were observed. Best performance was achieved when $\beta = 1.0$.

Table 8.6: Augmented model rescoring of `eval03` development set; β=1.0

	% of rescored pairs corrected		
	fsh	s25	Overall
Corrections	5.1%	3.1%	4.0%
Scoring	4.7%	3.1%	3.9%
# Pairs rescored	591	674	1,265

When the baseline transcription was rescored using ten augmented models, relative gains of approximately 4% were observed. A breakdown of the results is shown in table 8.6. Although some pairwise classifiers performed well, such as CAN/CAN'T, the overall improvements in performance are small since improvements are averaged over ten different classifiers, each with very different performance. For example, the augmented models for disambiguating the pair CAN/CAN'T corrected 19 confusions out of 156 (12.2%) whereas models for the pair YEAH/YES corrected only 2 out of 213 (0.9%). When the corrections from these and eight other classifiers were combined, improvements in the accuracy of approximately 4.0% were observed.

It is interesting to note that the data collection protocol (Fisher versus Switchboard) [16] has a small effect on the number of confusions corrected. There are two possible reasons for this. The first is that augmented models were trained using only Fisher data. This means that the Fisher component of the development set has less mismatch between the training and test conditions than the Switchboard component, causing models to perform better. The second reason is the disparate baseline performances on the different types of data (CN decoding yields 33.7% WER for Switchboard and 26.2% for Fisher). The higher Switchboard word error rate suggests that Switchboard confusion network posterior scores are less reliable than Fisher data scores. Since the augmented model/confusion network combination weight β is estimated from both types of data, it is likely that the CN posterior contribution is over-estimated when Switchboard data is used, resulting in a degradation of performance. This suggests that β should varied for different data sources and collection protocols. Experiments to verify this have not been performed.

8.2.2 Evaluation sets: eval02 and eval04

In the previous section, the `eval03` development set was used to select ten augmented models for disambiguating confusable word pairs in Fisher and Switchboard data. In this section, pairwise augmented models are evaluated using two evaluation sets: `eval02` and `eval04`. The `eval02` corpus contains five hours of English CTS Switchboard data in the form of 60 conversations (split equally between Switchboard I, Switchboard II, and Switchboard Cellular data-types). The `eval04` corpus contains 3 hours of English CTS Fisher data in the form of 36 conversations. Baseline CN decoding on the ML-estimated LVCSR system yields performances on `eval02` and `eval04` of 32.1% and 29.2% respectively. The resulting CN transcriptions were used as baseline transcriptions for augmented model rescoring experiments.

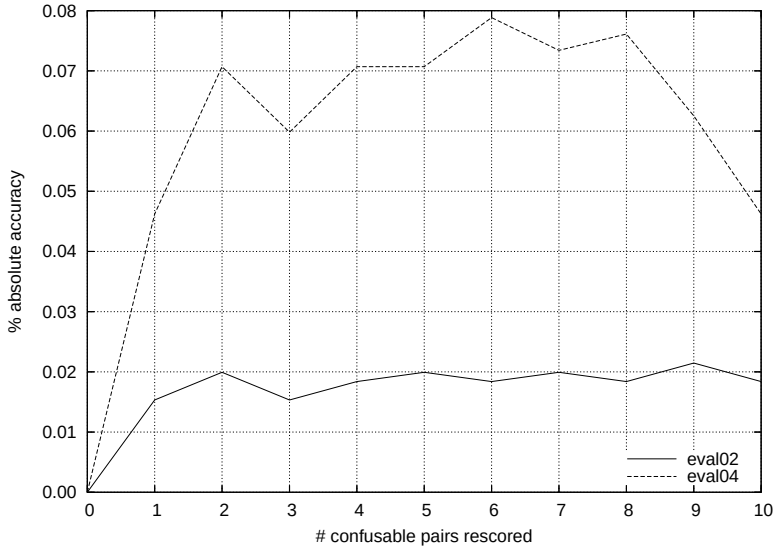


Figure 8.3: Rescoring confusions in the `eval02` and `eval04` evaluation sets

In figure 8.3, results from rescoring the `eval02` and `eval04` evaluation sets are presented. Absolute improvement over the baseline system is shown plotted against the number of confusions rescored. From the graph, it is clear that the performance improvements for both evaluation sets is due primarily to the corrections made by the first two word pairs, CAN/CAN'T and YEAH/YOU. Applying further augmented models yielded no significant change in accuracy.

A second feature of these experiments is that the two evaluation sets contain different types of data: `eval02` contains Switchboard data whereas `eval04` contains Fisher data. Since pairwise augmented models are trained using only Fisher data, performance gains on the `eval04` test set were expected to be larger than on the `eval02` data since the training and testing conditions were more closely matched. This difference is clearly visible in figure 8.3, with `eval04` showing performance gains 3–4 times larger than those of `eval02`.

At this stage, it is worth commenting on the small magnitude of the accuracy improvements for both evaluation sets. As illustrated in figure 8.3, rescoring `eval02` and `eval04` using up to ten augmented models yields absolute accuracy improvements of 0.02% and 0.07% respectively. When compared with the baseline accuracies of 32.1% and 29.2%, these improvements were found to statistically insignificant [40].[4] There are two main reasons why gains are so small. The first is that for any given confusion pair, the number of corrections that can be made is dependent upon the difference between the word error rates of the baseline CN decoding and the pairwise augmented models. Even for a good pair, such as CAN/CAN'T, this limits the potential gains. For example, for a representative 100 confusions, CN decoding of CAN/CAN'T will get 22 confusions incorrect whereas pairwise augmented models will get 9 incorrect (table 8.2). This means that only 13 pairs (13%) can possibly be corrected. Augmented models for many other word pairs will yield smaller improvements. This means that, on average, less than 10% of all rescored confusions will result in a correction (and an associated improvement in accuracy). The second reason for the small improvements is that the number of words being rescored is small in comparison to the total number of words in the evaluation set. For example, when ten augmented models are used, only 1,433 words of `eval02` are rescored out of a total of 65,236 (2.2%). Assuming an optimistic 10% improvement in accuracy for 2.2% of the data would yield a best-case performance improvement of only $\sim 0.2\%$—small in comparison to the baseline error.

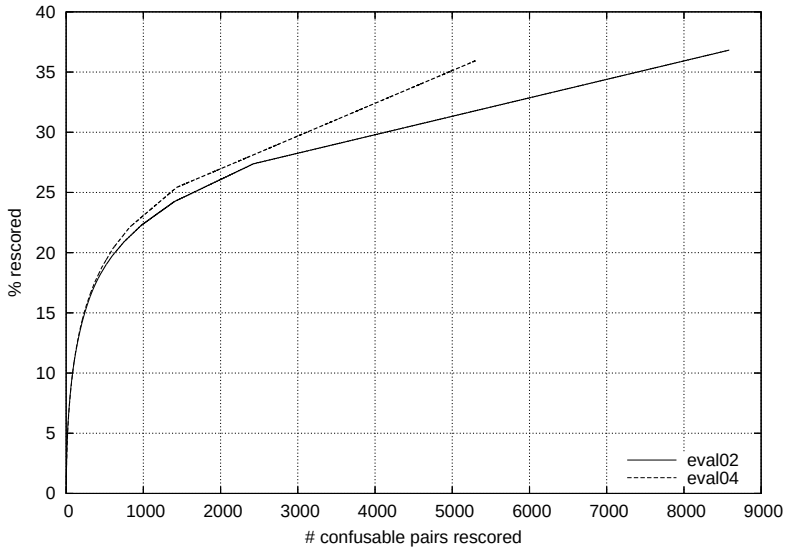


Figure 8.4: Proportion of evaluation sets rescored as number of augmented models increases

From figure 8.4, it is clear that as more augmented models are trained to disambiguate

[4]Statistical significance of experimental results was evaluated using the NIST scoring toolkit `sctk-1.2`. Tests were performed using the matched-pair sentence-segment word error rate (MAPSSWE) test and are quoted at a confidence level of 95%.

confusable pairs, the proportion of the evaluation sets being rescored increases. Unfortunately, the graph shows that a large number (c.1,500) of pairwise augmented models are required in order to rescore even 25% of the evaluation data. The computational cost of training this many models would outweigh the potential gains that could be achieved. Furthermore, when more accurate baselines are used (such as MMI-estimated systems), the potential rescoring gains may drop significantly since there are fewer errors to correct. The effect of better baselines on rescoring performance is discussed in more detail in section 8.3.1.

8.2.3 Generative versus conditional augmented models

In the experiments discussed thus far, maximum-margin estimated generative augmented models were used to disambiguate pairs of confusable words. In this section, classification performance of conditional augmented (C-Aug) models is compared against the performance of standard augmented models. For this comparison the pair CAN/CAN'T was used. Alongside baseline ML and MMI HMMs, augmented and conditional augmented models were trained using score-spaces containing either 640 or 1,896 derivatives with respect to the base model mixture-component priors, means and variances. All classifiers were based upon three-state, four-mixture-component HMMs. Results are shown in table 8.7.

Table 8.7: Comparison of MM augmented models and CML C-Aug models for the confusion pair CAN/CAN'T

Classifier	Training criterion		Score-space components	Training error (%)	Test error (%)
	$\boldsymbol{\lambda}$	$\boldsymbol{\alpha}$			
HMM	ML	–	–	10.4	11.0
HMM	MMI	–	–	9.0	10.4
Aug	ML	MM	640	7.1	9.2
C-Aug	ML	CML	640	7.3	9.1
Aug	ML	MM	1,896	6.1	9.1
C-Aug	ML	CML	1,896	5.0	10.5

As shown in table 8.7, for 640-dimensional score-spaces, both augmented models and conditional augmented models outperform the ML and MMI baseline HMMs. Both have a similar number of errors during training and yield test errors of approximately 9.1%. When more powerful augmented models (with 1,896 score-space components) are considered, the situation is different. Maximum-margin estimated augmented models, with their superior generalisation ability, reduce both the training and test errors. Conversely, although C-Aug models trained using the conditional maximum likelihood criterion also reduce training error, the improvement comes at a cost of over-training. Test error therefore increases from 9.1% to 10.5% (worse than the MMI-estimated HMM baseline). This

simple example demonstrates that although C-Aug models can perform similarly to MM augmented models given sufficient data, care must be taken to ensure that over-training does not occur. As expected, for high-dimensional score-spaces, the maximum-margin training criterion provides superior generalisation performance when compared to other training criteria such as CML.

In addition to the comparison between MM augmented models and C-Aug models, the CAN/CAN'T confusion pair, discussed above, was also used to compare the efficiency of different optimisation algorithms for CML training of C-Aug models. Like generative augmented models, C-Aug models are convex in their augmented parameters, allowing CML training to converge to a global solution regardless of the optimisation approach used. This means that the choice of optimisation algorithm can be based primarily upon computational efficiency. Results from four different optimisation algorithms are shown in table 8.8.[5] Where applicable, learning rates were tuned to maximise the rate of convergence. Note that many of the experiments were terminated early due to limited computational resources. Despite this, useful conclusions can still be drawn from the results.

Table 8.8: Comparing optimisation algorithms for CML estimation of C-Aug models

Algorithm	# Total iterations	Final objective	Training error (%)	Function evaluations	
				f	df
Gradient descent	5,000*	1,417.0	8.4	5,000	5,000
Stochastic gradient descent	5,000*	1,296.5	7.7	5,000	5,000
Conjugate gradient	200*	1,301.5	7.9	4,675	4,216
Scaled conjugate gradient	958	1,217.5	7.3	959	1,915

Note that (*) is used to denote that optimisation was terminated after 5000 or 200 iterations; convergence did not occur within this time

As illustrated in table 8.8, gradient descent (GD), the simplest algorithm, converged slowest, taking 5,000 iterations to achieve a training error of just 8.4%. Its online equivalent, stochastic gradient descent (SGD), performed significantly better, reaching the final GD objective of 1,417.0 after just 858 iterations. Optimisation continued until it was terminated at 5,000 iterations where is achieved a training error of 7.7%. Both algorithms required 5,000 evaluations of the objective function and a further 5,000 calculations of its derivative.

Next, to increase the rate of convergence, two conjugate gradient methods were considered. The first was a standard implementation of non-linear Polak-Ribière conjugate gradient (CG) [98,116]. As shown in table 8.8, this required far fewer iterations than SGD to obtain similar performance. Unfortunately, the average computational cost for each iteration of CG is much higher, with an average of 23 function and 21 derivative evalu-

[5] Although the optimisation algorithms used different stopping criteria (with some forcibly terminated), the results in table 8.8 are illustrative of the general trends in computational cost.

ations per iteration (c.f. 1 function and 1 derivative evaluation for GD and SGD). The overall computation cost of CG is therefore similar to that of GD and SGD.[6] The second conjugate gradient algorithm evaluated was scaled conjugate gradient. This was selected because it claimed to offer the benefits of standard CG algorithms without the expensive line minimisation. The downside is a significantly more complex implementation. As the table illustrates, SCG achieved the best objective function value and the lowest training error whilst requiring fewer than half the derivative evaluations of other techniques and less than a fifth of the function evaluations. The superior benefit to cost ratio of SCG optimisation for C-Aug models means that all C-Aug model experiments discussed in this thesis are trained using an SCG optimised CML or MPE objective function.

8.2.4 Summary

In this section, maximum-margin trained augmented models were used to rescore confusable word pairs in a large vocabulary speech recognition system. Although performance benefits were observed for selected pairs, these were found to be insignificant in the wider context of system performance. This was due to two main factors. The first is that the number of corrections that can be made for any given confusion pair is small. The second is that, within a large vocabulary speech recognition system, there are millions of potential word pairs, each of which requires its own augmented model classifier. Since limited training data restricts the number of augmented models that can be trained, only a small percentage of the evaluation data can be rescored. The combination of limited improvements per classifier, coupled with the small proportion of data rescored, means that maximum margin augmented model rescoring is not suitable for LVCSR tasks.

Instead, rescoring experiments—similar to those discussed in this section—were performed using the TIMIT phone recognition database [38, 44]. Results are discussed in section 8.3. Unlike the LVCSR task discussed here, which has millions of potential word pairs, the phone-labelled TIMIT database has only 1,176 possible phone pairs, allowing a much larger proportion of the evaluation data to be rescored. Significant improvements in performance were observed.

8.3 TIMIT

The second experimental database used in this thesis was the TIMIT phone-labelled data set. This contains broadband recordings of 630 speakers of the eight major dialects of

[6]Note that it may be possible to reduce the computation cost of conjugate gradient by reducing the accuracy of the built-in line minimisation. This will decrease the cost per iteration. Although the reduced accuracy is likely to increase the number of iterations required, the overall cost may still fall.

American English, each reading eight phonetically rich sentences.[7] Data is split into a training set (462 speakers; 3,696 sentences) and a testing set (168 speakers; 1,344 sentences), each balanced for phonetic and dialectical coverage. The testing set is further split into the NIST core test set and a development set. The core test set contains 24 speakers (two male and one female from each of the eight dialects) and 192 sentences, selected such that each phone appears in at least one of the sentences. The development set is constructed using all remaining test speakers and sentences. There is no overlap in speakers between the training, development and core test sets.

The acoustic data was parameterised using a standard Mel-frequency cepstral coefficient (MFCC) front-end. Cepstral analysis was performed using a frame-length of 10ms and a 25ms Hamming window; spectral analysis used a 40 channel Mel filter bank with frequencies ranging from 64Hz to 8kHz. A pre-emphasis coefficient of 0.97 was used. Thirteen MFCC features (including the zeroth) were then extracted. These were combined with their first and second derivatives to yield a set of 39-dimensional feature-vectors that were then used for training statistical models.

Given the coded data, the training and test data sets were prepared for use in phone classification and phone recognition experiments. Training and test data for the phone classification experiments was generated by extracting acoustic data for individual phones from the coded data using the hand-labelled phone time-stamps included as part of the TIMIT database. Phones with less than three frames of data (<30ms in length) were excluded since HMM base models with a minimum duration of 3 frames were used.[8] Although this differs from the standard TIMIT classification task, the proportion of data affected is small, with only a minimal effect on the classification performance when compared against other work, for example [42]. For phone recognition experiments, whole sentences were extracted, along with their phone transcriptions. Without fixed phone alignments, minimum duration constraints caused no problems.

Using the TIMIT data, three different experiments were performed. The first, in section 8.3.1, replicates the confusable pairs framework of section 6.4 in order to disambiguate confusable phone pairs. This uses the MM-estimated generative augmented models discussed in chapter 6. The second and third experiments, in sections 8.3.2 and 8.3.3, evaluate CML and MPE estimated C-Aug models within the TIMIT phone classification and recognition frameworks respectively.

[7] Speakers actually read ten sentences but, as is standard practice, the speaker adaptation sentences, 'sa1' and 'sa2', are excluded from both the training and test sets.

[8] A large number of these excluded phones were only 1 frame (10ms) long and are believed to be a by-product of the TIMIT transcription process. This process proceeded as follows. For each sentence, a word transcription was first generated. This was then converted into a phone transcription using a word-to-phone dictionary. The resulting string of phones was then manually fitted to the acoustic data irrespective of the acoustic sounds. Words with missing or partially pronounced phones may therefore contain short phone segments that fill the gaps between the articulated phones.

8.3.1 Rescoring confusable phone pairs

Similarly to the LVCSR confusable word-pair rescoring in section 8.2, MM-estimated augmented models were trained to disambiguate confusable phone-pairs within the TIMIT phone-classification framework. These experiments differ from the LVCSR results presented in section 8.2, with phone confusions used instead of word confusions. Since speech recognition systems typically have far fewer phones than words, there are many fewer confusions for the TIMIT data than for the LVCSR data discussed earlier. This makes it possible to rescore a much larger proportion of the test data, increasing the chance of statistically significant performance gains.

Using the TIMIT classification training set, baseline monophone HMMs with three states and ten mixture-components were estimated using ML training. A second MMI baseline using similar HMMs was also defined. Language models were not used. Phone error rates for the ML baseline system were 27.9% and 28.8% for the development and core test sets respectively; error rates for the MMI development and core test sets were 24.4% and 24.7% respectively. Next, for both baselines, and for each of the training, development and core test sets, lattices of confusable phones were generated using Viterbi decoding. These were used to identify the two most likely (and hence confusable) phone labels associated with each observation sequence. As for the LVCSR confusion-pair task, pairwise augmented models were trained to disambiguate the twenty most confusable phone-pairs.[9] Performance for the three most common phone pairs are shown in table 8.9. Further details of the augmented models used for rescoring are given in appendix D.

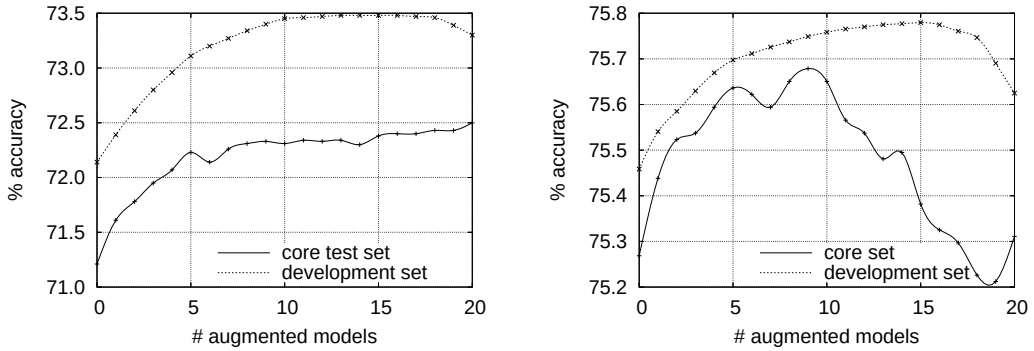
Table 8.9: Confusable phone-pairs in the TIMIT classification core test set (% error)

Classifier	Training criterion		Phone pair (# examples)		
	$\boldsymbol{\lambda}$	$\boldsymbol{\alpha}$	s/z (397)	er/r (317)	m/n (195)
HMM	ML	–	21.9	25.2	26.7
HMM	MMI	–	20.7	18.6	21.0
AUG	ML	MM	16.4	14.8	20.0

As illustrated by table 8.9, pairwise phone classifiers showed similar trends to the word classifiers described in section 8.1.1. In particular, MM-estimated augmented models outperformed MMI HMMs which, in turn, outperformed ML HMMs.

Next, MM augmented models were used to rescore the development set transcriptions, obtained from the ML-estimated HMM baseline system. Performance improvements for each phone-pair were recorded, allowing the phones to be sorted (best first). Using this ordering of pairs, the TIMIT core test set was cumulatively rescored. The process was repeated for the MMI HMM baseline. Rescoring performances for the development and core test sets, using both the ML and MMI baselines, are plotted in figure 8.5.

[9] As in section 8.1, the number of phones in each half of a confusion pair were equalised to reduce bias.



(a) ML baseline (b) MMI baseline

Figure 8.5: Augmented model rescoring of the TIMIT phone classification task

From the graphs it is clear that development set performance for the ML system peaks at approximately ten classifiers before tailing away at twenty classifiers. Selecting the ten best classifiers, and using them to rescore the core test set, allows 78 out of the 1,487 rescored phone pairs to be corrected, a relative improvement of 5.2%. This is comparable to the 4.0% improvement observed on the LVCSR development set. Where phone-pair rescoring experiments differ from the word-pair experiments discussed in the previous section, however, is the proportion of the test set rescored. Whereas ten word-pair classifiers allowed only 1–2% of the LVCSR task to be rescored, the ten phone-pairs used in this task allowed a little over 21% of the TIMIT core test data to be rescored, resulting in a reduction in the error rate from 28.8% to 27.7%, an absolute improvement of 1.1%. This is statistically significant to a 95% confidence [40].

As for the ML system, development set rescoring performance on the MMI baseline peaked at approximately ten classifiers before falling away sharply at twenty classifiers (figure 8.5). Gains on the MMI development set transcription were much less than those achieved on the ML transcription (0.3% versus 1.3%). This is believed to be due to the better performance of the baseline system (24.4% error for MMI system versus 27.9% for ML): with fewer errors to correct, augmented model classifiers have fewer chances to improve upon the baseline transcription, limiting the gains observed. This limitation is especially apparent when rescoring performance on the core test set is examined. Here, the limited performance gains and small size of the test set result in a graph in which classification noise is clearly visible. Overall, gains of only 0.3% were observed over the baseline MMI system despite rescoring a similar proportion of the data as the ML system. The improvement gained from rescoring the MMI baseline was too small to be statistically significant.

As a final contrast to the LVCSR system discussed earlier, the proportion of the development and core test sets rescored was plotted against the number of pairwise classifiers. This is shown in figure 8.6. From this, it is clear that relatively few pairwise classifiers are

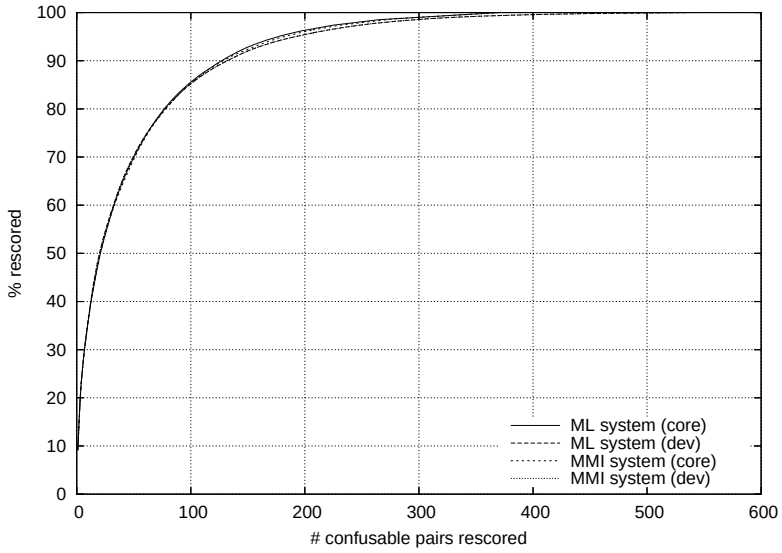


Figure 8.6: Proportion of development and core test sets rescored as the number of augmented models is increased

required in order to rescore a significant proportion of both the development and core test sets. For example, for all test sets, 100 phone-pair classifiers allows approximately 85% of the confusions in the core test set to be rescored, whereas 200 pairs allows approximately 95% to be rescored. In comparison, rescoring the LVCSR `eval02` and `eval04` evaluation sets using 200 word-pairs would only allow 25% and 14% of the data to be rescored, respectively. With far fewer classifiers required to get good data coverage, the TIMIT data benefits more from confusable pair rescoring than the LVCSR task discussed earlier. Other tasks, with few classes to disambiguate, such as number recognition and command and control are also expected to perform well within this confusable word/phone framework.

8.3.2 C-Aug classification

In all experiments discussed thus far, the acoustic code-breaking techniques in section 6.4 were used to convert multi-class speech classification problems into a series of binary tasks, for which binary classifiers were trained. However, although this is necessary for generative augmented models, C-Aug models, with their easy-to-calculate normalisation terms can be applied directly to multi-class classification problems.

In this section, the application of multi-class C-Aug models to the TIMIT phone classification data described previously is discussed. Using the standard practice of building acoustic models for 48 different phones, HMM base models were estimated using either an ML or an MMI training criterion. These HMMs contained three emitting states and either ten or twenty mixture-components. C-Aug models were then constructed using sufficient statistics defined by the first-order derivatives of the base models; derivatives with respect to the mixture-component priors, means and variances were used. Augmented parameters

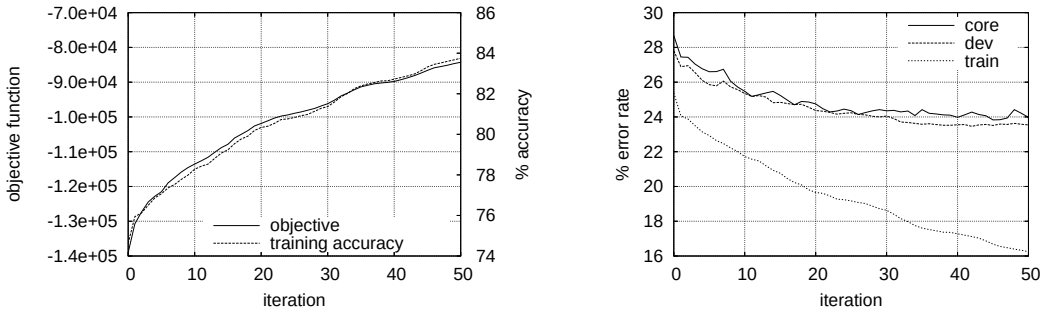


Figure 8.7: CML estimation of C-Aug model augmented parameters: (a) correlation between the CML objective function and training accuracy (b) reduction in development and core test set error.

were estimated using CML training (section 7.2); no language model was used.

In figure 8.7(a), results from conditional maximum likelihood (CML) estimation of a three-state, ten-mixture-component C-Aug model (ML base model parameters) are plotted. In the left graph—figure 8.7(a)—CML estimation of the augmented parameters is shown to yield a large increase in the CML objective function, from -139,000 to -84,300. This is matched by a corresponding reduction in training error from 25.4% to just 16.3%, an absolute improvement of 9.1%. The objective function (left axis) and the training set word accuracy (right axis) are highly correlated since the objective function—the posterior probability of the training set class labels—is a differential approximation to the training accuracy. Figure 8.7(b) illustrates the error reductions achieved on the development and core test sets. Gains of 4–5% were observed. Despite the large difference between the final training and test performances, over-training was not observed, even aften many training iterations. This illustrates the generalisation performance of CML training when sufficient training data is present. Further results on this task are presented in table 8.10.

Table 8.10: Classification error on the TIMIT core test set

Classifier	Criterion		Components	
	$\boldsymbol{\lambda}$	$\boldsymbol{\alpha}$	10	20
HMM	ML	—	29.4	27.3
C-Aug	ML	CML	24.0	—
HMM	MMI	—	25.3	24.8
C-Aug	MMI	CML	23.4	—

As illustrated in table 8.10, multi-class C-Aug model results show a similar pattern to the pairwise augmented model experiments discussed in sections 8.1 and 8.3.1. In particular, C-Aug models yielded statistically significant improvements over the base models from which they were derived. For example, C-Aug models with 10-mixture-component ML base models outperformed 10-mixture-component ML HMMs. Although this could be argued to be a consequence of having extra parameters (C-Aug models have almost

twice as many parameters as standard HMMs), this was found to account for only a small proportion of the gain, with 10-mixture-component C-Aug models also outperforming 20-mixture-component HMMs. This clearly illustrates that modelling power is dependent upon not only the number of parameters, but also the types of dependencies modelled.

In addition to C-Aug models with ML-estimated base models, experiments using MMI-estimated base models were performed. As shown in table 8.10, C-Aug models with MMI estimated base models outperformed equivalent C-Aug models with ML base models. This is believed to be due to the better performance and improved state segmentation of the MMI base models (in comparison to ML base models), resulting in more discriminative sufficient statistics. However, despite having poorer statistics than C-Aug models with MMI base models, ML-based C-Aug models managed to outperform the 20-mixture-component baseline MMI HMM by 0.8% (24.0% error versus 24.8%).

Although obtaining good performance compared to standard HMMs, the C-Aug models discussed in this section do not quite attain the test set performance of HCRFs [42] on the same task (HCRFs yielded a 21.8% core test error). This is believed to be due to two main factors: fixed state segmentation, and the lack of a language model. The first of these, fixed state segmentation, is believed to be the most important since it affects the quality of the C-Aug model sufficient statistics: the worse the base HMM, the more unreliable the latent-state alignments within that model are likely to be. Sufficient statistics derived from the base model are therefore likely to be less discriminatory than would otherwise be expected. Since CML estimation of the augmented parameters cannot alter this state segmentation, lower than expected C-Aug performance is obtained. This sensitivity is partially illustrated by the ML (worse) versus MMI (better) base model comparison discussed previously. The second reason that HCRFs outperform C-Aug models is that HCRFs use a unigram language model whereas C-Aug models contain no language model. Experiments on ML and MMI HMMs, with and without language models, suggest that this can yield an absolute performance improvement of between 0.5% and 1.3%. Introducing unigram language models into discriminatively trained C-Aug models is expected to yield gains in the lower end of this range.

8.3.3 C-Aug recognition

In addition to the classification experiments discussed above, the TIMIT database was used for evaluating C-Aug sentence model recognition performance. Phone-level recognition was performed (the standard approach for TIMIT) to ensure that system performance reflected the quality of the phone acoustic models, rather than the effects of external constraints such as phone-to-word mappings. The acoustic data was first parameterised using the TIMIT MFCC front-end discussed in section 8.3. Baseline phone HMMs (three latent states, each with a ten mixture-component GMM output distribution) were then

estimated using maximum likelihood estimation on the training set of 3,696 sentences. Recognition phone error rates on the training, development and core test sets for these models were found to be 37.9%, 41.5% and 42.9% respectively.

Using the correct transcriptions of the training sentences, a phone-level bigram language model was then estimated. The language model scale factor was optimised using the development set; best performance was achieved with a scale factor of 5. Recognition phone error rates for the ML-estimated HMMs with a bigram language model were 29.4%, 32.0% and 32.9% for the training, development and core test sets respectively. These results are summarised in table 8.11. Examining the large differences in phone error rate between recognition with and without the language model, it is clear that language models are an important component of HMM-based speech recognition systems.

Next, to allow efficient MMI and MPE training of HMMs, a set of training lattices were generated using Viterbi decoding on each training sentence [138]. To ensure that the probability mass of each lattice was distributed between a number of competing hypotheses, a weakened language model and lattice probability scaling were used. The training lattices used in this section were generated using a unigram language model, estimated from the correct training sentence transcriptions. This typically penalises incorrect sentences less than more powerful language models such as bigrams, thus preventing a single sentence from dominating the lattice. The second technique for increasing the number of competing hypotheses was to the scale the acoustic model likelihoods by a factor $0 < \kappa < 1$. This scaling reduces the dynamic range of acoustic model likelihoods, increasing the importance of less likely sentences [135]. As is standard practice, κ was set to the inverse of the language model scale, $\kappa = 0.2$.

Table 8.11: Baseline HMM recognition performance on TIMIT; A bigram language model was used with a language model scaling factor of 5.

Training criterion	Phone error (%)		
	Train	Dev	Core
ML	29.4	32.0	32.9
MMI	25.7	30.6	31.6
MPE	23.7	29.3	30.6

Given the training lattices, MMI and MPE trained HMMs were estimated. Recognition results for these are presented in table 8.11. As shown, the discriminative MMI and MPE training criteria outperform standard ML training, with MPE yielding the largest improvement in performance. Experiments showed that the ML system language model scale factor of 5 remained optimal for both the MMI and MPE systems.

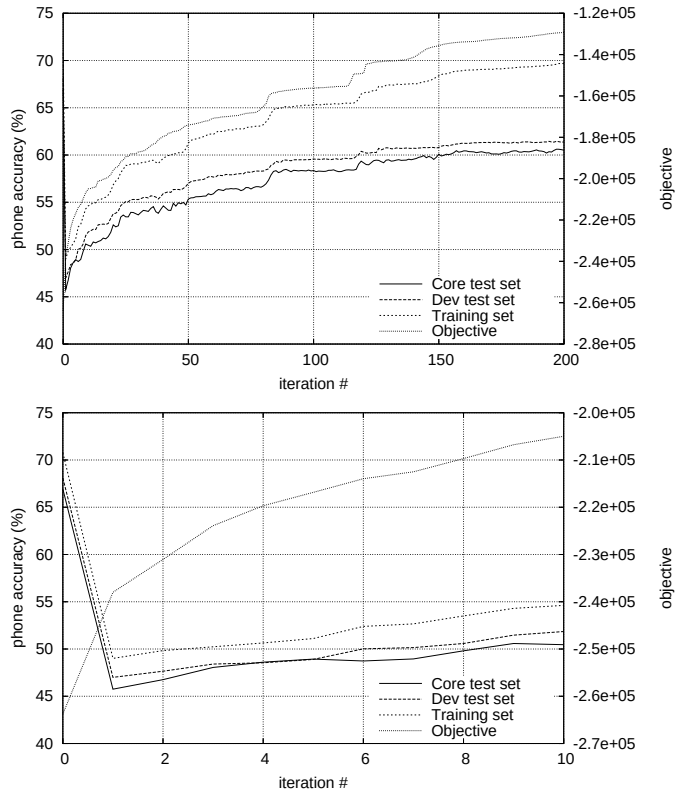


Figure 8.8: CML C-Aug models for TIMIT phone recognition: (a) CML objective function and training, development and core test set phone accuracies (b) close-up of the first 10 iterations.

CML C-Aug models

Given ML-estimated phone HMMs as base models, C-Aug models were constructed using sufficient statistics defined by the first-order derivatives of the phone HMM likelihoods: derivatives with respect to the HMM mixture-component priors, means and variances were used. C-Aug sentence models were then estimated using the lattice-based conditional maximum likelihood (CML) training criterion discussed in section 7.4. Phone accuracy results are shown in figure 8.8. Sentence accuracies and error rates are not quoted in this work since the long length of the TIMIT sentences, and the relative simplicity of the systems considered, means that sentence accuracy was 0% for all systems.

From figures 8.8(a) and 8.8(b), it is clear that the CML objective function—the sum of the log-posterior probabilities of the correct training sentence transcriptions—increases monotonically with every iteration. Since these posteriors approximate the sentence accuracy, improvements in the objective are expected to produce similar improvements in the recognition accuracy. As illustrated in figure 8.8(a), between iterations 2 and 200, there is a good correlation between the objective function and the training accuracy. A similar correlation exists between the recognition accuracy on the training data and on

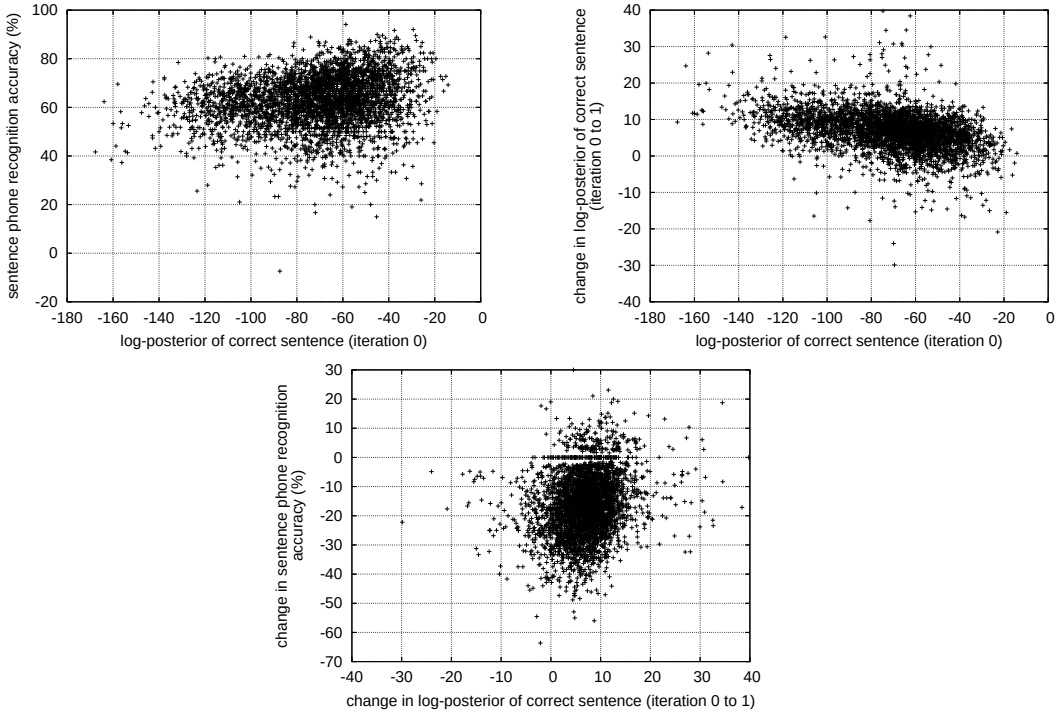


Figure 8.9: Analysing the poor performance of CML for C-Aug recognition: (a) Correlation between posterior probability of correct sentence and recognition accuracy for that sentence (b) negative correlation between sentence posteriors and the increase gained through CML training; and (c) overall effect of the first CML update on correct sentence posterior and recognition accuracy

the development and core test sets.

However, a closer examination of the first iterations of CML training, in figure 8.8(b), reveals that the first iteration of C-Aug model training causes a significant drop in recognition accuracy on both the training and test data, with training accuracy falling from 70.8% to 49.0%. The most likely reason for this reduction in performance is the mismatch between CML objective function (maximising the posterior probability of the correct sentence transcriptions) and the phone-based scoring function. In the first iteration, with many options for increasing the posterior probability of the correct sentences, the SCG optimisation selects an update that gives the largest increase in posterior, regardless of its effect on sentence accuracy. In practice, this tends to result in the scores associated with the correct sentence decreasing whilst incorrect sentence scores decrease even further. Overall, the correct sentence posterior increases even though the acoustic models are less representative of the data. Unfortunately, the resulting models perform poorly on both training and test sets.

The mismatch between the objective and scoring functions, and the effects of SCG training, are clearly illustrated by the graphs in figure 8.9. The first plot, in figure 8.9(a), shows that in the first iteration, phone recognition accuracy of individual training sen-

tences is only loosely correlated with the posterior probability of the correct sentence. This suggests that sentence posteriors are a poor estimate for the phone accuracy, and that maximising the posteriors will have, at best, only a limited effect on phone recognition accuracy. During later iterations of training, similar plots show that the correlation between the correct sentence posterior probability and the recognition accuracy is much stronger, explaining the correlation between the CML objective function and recognition accuracy for these iterations.

A second reason why increases in sentence posterior may not improve recognition accuracy is demonstrated in figure 8.9(b). Here, the negative correlation between the log-posterior of the correct sentences and the improvement in the posteriors suggests that much of the objective function improvement is achieved by increasing the posterior probabilities of the least likely correct sentences. Since these sentences are unlikely, the positive impact on these posterior improvements on phone recognition accuracy is minimal. In practice, accuracy was observed to deteriorate.

The final graph (c) in figure 8.9 depicts the combined effect of these two observations. This clearly shows that changes in sentence posteriors and phone recognition accuracy are centred on an increase in posterior of approximately 7 and a reduction in accuracy of approximately 18%. These figures correspond almost directly to the measured change in objective function (an increase of 25,756 for 3,696 sentences, yielding an average posterior improvement of 7.0 per sentence) and training accuracy (23.8% reduction). The actual reduction in accuracy of 23.8% is more than the 18% that would be expected from the centre-of-mass since the distribution is skewed towards lower phone accuracies (higher error rates).

At this stage, it is worth considering why CML estimation of C-Aug models suffers from the mismatch between the objective function and the scoring function more than MMI estimation of standard HMMs. The most likely reason why C-Aug models are more susceptible to this problem than HMMs is that they are extremely flexible, with few constraints. When used for recognition, the only constraint on C-Aug sentence models is that sentence posteriors must be properly normalised; individual phone scores may take any value. This means that CML training of C-Aug models does not need to consider the 'quality' or performance of individual phone models. In contrast, MMI estimation of HMMs typically requires that both sentence posteriors and individual phone posteriors are properly normalised—the additional normalisation of phone models is achieved through the use of constraints on the HMM parameters, such as sum-to-one constraints on the transition probabilities and the mixture-component priors, and the requirement that variance matrices in the state output distributions are positive-definite. MMI training therefore maximises sentence posteriors whilst maintaining the 'quality' of the individual phone models. MMI estimated HMMs can therefore be expected to be more robust to

recognition errors (if a single phone is incorrectly recognised it is less likely to disrupt the rest of the transcription) than CML training of C-Aug sentence models. It is also more likely to yield models that generalise well to unseen sentences. It is believed that if additional phone normalisation constraints were added to C-Aug sentence models, better performance would be obtained.

MPE C-Aug models

To improve the performance of C-Aug sentence models, the mismatch between training and scoring can be reduced by replacing the CML training criterion with the minimum phone error (MPE) training criterion. As discussed in section 7.4.4, this directly maximises an approximation to the sentence phone accuracy (equivalent to minimising an approximation to the phone error rate). Since the objective function is a good approximation to the true phone error rate, improvements in objective are expected to lead to similar improvements in accuracy. Preliminary results from SCG optimisation of the MPE objective support this hypothesis, and are shown in figure 8.10.[10]

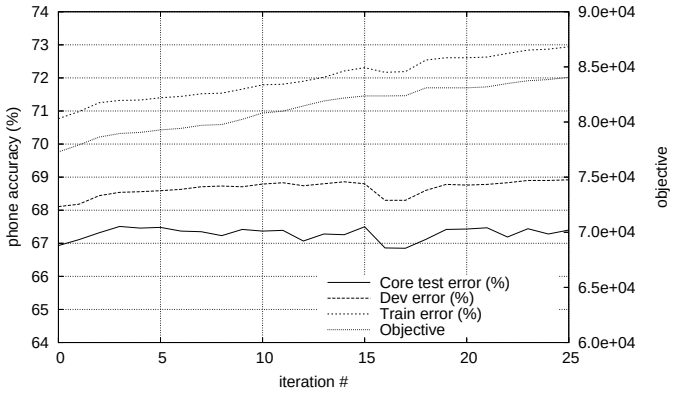


Figure 8.10: MPE training of C-Aug sentence models

As illustrated in the graph, the objective function and training set phone recognition accuracy are well correlated, indicating that the MPE is a good approximation to the scoring function. Between the zeroth and third iterations, recognition accuracy on the development and core test sets increased from 68.1% to 68.5% and from 66.9% to 67.5% respectively. In contrast to the CML training discussed earlier, there is no reduction in recognition accuracy. However, comparing the 0.6% improvement on the core test set with the 2.3% achieved for MPE training of the base HMM, it is clear that MPE training of C-Aug sentence models is limited. This underperformance is believed to arise for two reasons. First, the objective function converges towards a local maximum of the MPE

[10]Note that zeroth-iteration C-Aug sentence model accuracies differ slightly ($\sim 0.2\%$) from the HMM baseline results in table 8.11. This is because C-Aug model phone likelihoods are calculated using the Forward-Backward algorithm, whereas HMM likelihoods are calculating using the Viterbi approximation.

objective function (which, unlike the CML objective function, has many local maxima). Second, by fixing the base model parameters, the state alignments within each phone HMM/C-Aug model are fixed, limiting the gains that can be achieved (see section 8.3.2 for more details). Despite these limitations, MPE estimated C-Aug sentence models yielded statistically significant improvements in accuracy, compared to baseline ML HMMs.

Although experiments have not been performed to verify this, one approach that is likely to alleviate some of the problems of local maxima is to change the optimisation algorithm used for MPE estimation of C-Aug sentence models. Whilst the SCG algorithm has proven to be a very efficient algorithm for maximising convex objective functions such as the CML objective, the quasi-second-order nature of SCG increases the likelihood of it reaching a local maxima for non-convex tasks such as MPE training. For such tasks, better performance is often achieved using simpler algorithms where updates are only loosely based upon the objective function gradients. Examples of such algorithms are stochastic gradient ascent [83, 84] and R-Prop [87]. Both have been successfully used for training HCRFs [42, 74] which also suffer from local-maxima problems.

8.4 Summary

In this chapter, a simple code-breaking approach was used to decompose a multi-class speech recognition task into a series of binary classification tasks suitable for the application and training of augmented models. Initial experiments were based upon an eight-fold cross-validation framework and the `fsh2004sub` subset of the LDC Fisher data. In many cases, maximum-margin estimated augmented models were found to outperform baseline ML and MMI estimated HMMs. Augmented models with second-order sufficient statistics or MMI- or MM-estimated base models yielded no significant improvements in performance, compared with augmented models defined using the standard ML statistics. Kernelised augmented models were found to yield performance improvements only when the number of explicit sufficient statistics was reduced.

In addition to cross-validation experiments, augmented models were evaluated using the `eval02`, `eval03` and `eval04` test sets. For these experiments, models were trained using the same code-breaking approach as before, but with the whole `fsh2004sub` training set. Word pairs to be rescored and confusion network posterior-ratio weights were then selected using the `eval03` development set. Classification performance was evaluated using two test sets: `eval02` and `eval04`. Small gains were observed on both, with approximately 4% of the rescored pairs being corrected. However, with only a small proportion of the test sets being rescored, the overall improvement in word error rate was found to be insignificant.

The second database discussed in this chapter was the TIMIT phone-labelled data

set. Using a similar code-breaking approach to the LVCSR experiments, the TIMIT classification task was first converted into a series of binary classification tasks. Maximum-margin augmented models with ML base models were then trained. The ratio of phone corrections to rescored phones was found to be similar to the word rescoring rates for LVCSR. However, with far fewer phone confusions in comparison to word confusions, a much larger proportion of the TIMIT database was rescored. An improvement in the overall phone error rate of 1.1% was therefore observed. Augmented models with MMI base models yielded smaller performance gains since there were fewer errors to fix. Using the same data, C-Aug models were then evaluated. Since C-Aug models are inherently multi-class, code-breaking was not required. C-Aug models were found to outperform standard ML and MMI HMMs by a large margin. The improved sufficient statistics in C-Aug models with MMI base models provided performance gains over similar C-Aug models with ML HMM sufficient statistics.

Finally a lattice-based implementation of C-Aug model recognition was evaluated using the TIMIT phone recognition task. Both conditional maximum likelihood (CML) and minimum phone error (MPE) training criteria were used. Although CML-estimated C-Aug models yielded good performance on the simple classification tasks, CML-estimated C-Aug sentence models performed badly due to a mismatch between the training criterion (maximising the posterior probability of the correct sentence transcriptions) and the scoring function (phone error rate). The MPE criterion avoided this mismatch by directly optimising an approximation to the sentence phone accuracy. Small performance benefits were observed. The scale of these improvements is believed to have been limited by the tendency of the optimisation algorithm to converge to a local maximum.

Overall, this chapter has demonstrated some promising results, especially for C-Aug model classification. Results suggest that C-Aug models may be useful for speech recognition but further research is required before HMM-beating results are obtained.

9

Conclusions and Future Work

In this thesis, statistical classification of sequence data was investigated, with particular emphasis placed upon techniques that allow statistical models to represent complex temporal dependencies between observations. The thesis contains three major contributions, split across chapters 4, 5, 6 and 7, and summarised in sections 9.1, 9.2 and 9.3 below. The first extends the theory of augmented statistical models by examining topics such as dependency modelling in augmented models, higher-order augmented models, and kernelised augmented models. A maximum-margin framework for training both augmented model parameters and base model parameters was also introduced. The second contribution, continuous rational kernels, are an extension of rational kernels and allow sequences of continuous observations to be classified within a weighted finite-state transducer framework. In this work, continuous rational kernels were formulated both as a general framework for classifying sequences of continuous observations, and as an efficient method for calculating augmented model sufficient statistics. The final contribution of this thesis are conditional augmented (C-Aug) models. These use the same additional sufficient statistics as standard (generative) augmented models but are formulated within a discriminative framework. This allows the normalisation issues of standard augmented models to be resolved. Parameter estimation and inference for C-Aug models were both discussed. A lattice-based approach for applying C-Aug models to speech recognition was also proposed.

9.1 Augmented statistical models

In chapter 2, a range of generative and discriminative approaches for statistical classification of sequence data were introduced. These allow relationships between observation sequences and class labels to be modelled using a number of sufficient statistics, independence assumptions, and conditional-independence assumptions. A number of standard statistical models were described. Unfortunately, the modelling assumptions associated with these standard models are incorrect for many applications, potentially affecting classification accuracy. Although many different approaches have been used to relax these modelling assumptions and allow additional dependencies to be modelled [8, 29, 106, 108], this thesis concentrates on the systematic approach taken by augmented statistical models [117]. Given a standard (base) model, these introduce a set of additional sufficient statistics (derived from the base model), that allow a wide range of additional temporal and spatial dependencies to be modelled.

In this work, the theory of augmented statistical models was extended to include aspects such as augmented model dependency modelling, higher-order augmented models, and kernelised augmented models. In particular, by examining the effects of augmentation on the independence and conditional-independence assumptions of the base model, it was shown that although augmented models retain the base model independence assumptions, conditional-independence assumptions can be 'broken'. In particular, for latent-variable base models, augmented model sufficient statistics break the assumption that observations are conditionally independent given the current latent-state. This allows a wide range of additional dependencies to be modelled.

The second contribution that this thesis makes towards the theory of augmented models is the introduction of a distance-based training algorithm for augmented models. This directly optimises the decision boundary between pairs of augmented models, allowing model parameters to be estimated even when the normalisation terms are intractable (unlike ML and MMI estimation which require explicit calculation of the normalisation). When the decision boundary between two augmented models is defined using Bayes' decision rule, it was shown that the decision boundary can be written as a linear decision boundary in a high-dimensional score-space. Estimation of this decision boundary can be performed using standard algorithms—such as the Perceptron algorithm or SVMs. Estimates for the augmented parameters can then be extracted from the decision boundary gradient. The intractable normalisation terms are combined to form a bias, calculated during training. In this work, augmented parameters were estimated using a new variable-margin SVM. Algorithms for maximum-margin estimation of the base model parameters were also proposed.

Augmented models were evaluated using two simple speech recognition tasks. The first used pairs of confusable words extracted from a conversational telephone speech data set.

Augmented models (with HMM base models) trained to disambiguate these word pairs outperformed HMMs in many cases. Unfortunately, since few word pairs were rescored, the overall effect on system performance was insignificant. However, when augmented models were used to rescore confusable phones extracted from the TIMIT classification task, a much larger proportion of the data was rescored, resulting in greater performance improvements. Overall, augmented models were found to outperform their base models in many cases.

9.2 Continuous rational kernels

The second contribution of this thesis is a new dynamic kernel: the continuous rational kernel (an extension of standard rational kernels [17, 18]). Rational kernels are a powerful form of dynamic kernel that allow high-dimensional feature-spaces to be defined for sequences of discrete observations. Defined entirely within a weighted finite-state transducer framework, rational kernels allow a wide range of task-dependent feature-spaces to be defined using finite-state transducers. Kernel calculations are performed using a small number of standard and efficient transducer operations. Unfortunately, as a form of discrete-observation dynamic kernel, rational kernels are restricted to sequences of discrete observations. Many applications, however, require classification of sequences of continuous observations.

This work introduces continuous rational kernels as a flexible extension of rational kernels for sequences of continuous observations. Continuous rational kernels are defined using a combination of discrete-latent-variable generative models and standard, discrete, rational kernels. For any given continuous-observation sequence, the latent-variable model is first used to calculate the latent-state sequences associated with the observations. These state sequences are determined using either Viterbi or Forward-Backward state alignment. When Viterbi alignment is used, a single state sequence is generated, corresponding to the most likely path through the model; when Forward-Backward alignment is used, lattices containing all possible state alignments for a particular observation sequence are generated. Since these sequences and lattices of latent-variable states have discrete labels (the state), this process allows continuous observations to be converted into a discrete representation. Applying a rational kernel to these sequences/lattices allows distances to be calculated.

In addition to being a form of dynamic kernel, continuous rational kernels can be used to generate the vectors of sufficient statistics used in augmented models. In particular, this work showed that applying standard n-gram and gappy-n-gram transducers to Forward-Backward-defined alignment lattices allows the posterior probabilities of the latent states to be calculated. With appropriate weighting of the transducer arcs, derivatives of the latent-variable generative (base) models can be calculated. Derivatives with respect to

vector quantities—such as the means and variances—were calculated by introducing a vector semiring that allows transducer arcs to be associated with vector weights instead of the more usual scalar weights. Calculation of augmented model sufficient statistics using the continuous rational kernel framework was shown to offer benefits over direct calculation of sufficient statistics since statistic-dependent dynamic programming algorithms can be replaced by a single unified framework for calculation. One advantage of this framework is that it allows continuous rational kernels to calculate higher-order augmented model sufficient statistics using only a small number of additional transducer definitions. No new algorithms are required.

9.3 Conditional augmented models

The final contribution of this thesis are conditional augmented (C-Aug) models. These use the same additional sufficient statistics as standard augmented models but are formulated within a conditional-probability framework. Since C-Aug models directly model the class label posterior probabilities, they are inherently discriminatory. Furthermore, C-Aug models allow the intractable augmented model normalisation term (an expectation over all observation sequences) to be replaced by an easy-to-calculate summation over the class labels. This allows C-Aug models to be trained using standard training criteria such as conditional maximum likelihood (CML) and minimum phone error (MPE).

In this work, C-Aug models were introduced as a discriminative model for performing multi-class classification of sequence data. Defined using the same sufficient statistics as generative augmented models, C-Aug models allow a similar set of dependencies to be modelled. In particular, conditional-independence assumptions of the base models are broken. A gradient-based algorithm was presented to allow both the augmented parameters, $\boldsymbol{\alpha}$, and the base model parameters, $\boldsymbol{\lambda}$, to be estimated using the conditional maximum likelihood (CML) training criterion. Optimisation was performed using scaled conjugate gradient (SCG) [83, 84] maximisation. Experiments comparing C-Aug models with HMM base models, to standard HMMs, showed benefits in many cases. When compared with maximum-maximum augmented models, C-Aug models performed similarly in many cases. The major advantage of C-Aug models over standard augmented models, however, is that multi-class classification can be performed without use of binary decomposition techniques.

C-Aug models were then extended to allow classification of complete sentences, allowing C-Aug models to be applied to a simple speech recognition task. Using a similar approach to that used in HMM-based speech recognition systems, C-Aug sentence models were defined using a small number of phone base models and sufficient statistics. The phone models were then combined to form a single, complex, discriminative model of sentence transcriptions given the observations. With a vast number of possible sentence

transcriptions, the C-Aug model normalisation term was approximated using a lattice containing only the most likely sentences. This lattice was determined using a standard HMM-based recogniser. Model parameters for the C-Aug sentence model were estimated using either the CML criterion—maximising the posterior probabilities of the correct sentence transcriptions—or the MPE criterion—minimising an estimate of the phone error. When parameters of C-Aug models were estimated using CML estimation on the TIMIT data set, phone accuracy dropped significantly during the first iteration of training. This drop is believed to arise from a mismatch between the training criterion (minimising an approximation to the sentence error) and the scoring criterion (phone error). The smaller number of constraints in C-Aug models, compared to HMMs, make this mismatch especially significant for C-Aug models. When the MPE criterion was used, small performance gains were observed.

9.4 Future work

Several aspects of the work presented in this thesis may benefit from further investigation, either in terms of modifications to the approaches given, or in the form of application to different problem domains. A number of suggestions for these future avenues of work are given below.

- This thesis has concentrated upon methods of augmenting generative statistical models—in particular, hidden Markov models (HMMs)—using sufficient statistics derived from a Taylor-series expansion of the base model. In general, however, many other forms of base model and augmentation are possible. Research into these could allow fundamental questions such as 'what forms of base model yield the most powerful augmented models?' and 'what properties should a good form of augmentation have?' to be answered.

- As discussed in chapter 8, second-order augmented model sufficient statistics are approximately zero when high-dimensional observations (such as in speech) are used. It would therefore be interesting to apply second- and higher-order augmented models models to low-dimensional or discrete data where this problem does not occur. One such application is protein structure prediction [47].

- One of the fundamental problems with generative augmented models is the intractable normalisation term. In chapter 6 this difficulty was avoided by expressing the augmented model decision boundary in terms of a linear hyperplane in a high-dimensional score-space. Although this allows augmented models to be trained, many of the advantages of generative models are lost. However, until the normalisation issue can be resolved, C-Aug models appear to be the most attractive research

direction.

- In chapter 7, conditional augmented (C-Aug) models were introduced as a discriminative form of augmented model. When these are trained using the conditional maximum likelihood criterion with large quantities of training data, good generalisation performance is achieved. However, when training data is limited, C-Aug models (and, more generally, CRFs) have been found to be susceptible to over-training. To avoid this research into regularisation or other robustness approaches is required.

- This work presented a preliminary framework that allows C-Aug models to applied to the task of speech recognition. To take this framework beyond a proof-of-concept, further research is required into the optimal conditions for C-Aug sentence model recognition. Areas that could be examined are: additional normalisation constraints on the C-Aug sentence models (e.g. ensuring that individual phone models are properly normalised), how best to incorporate language model information, and alternative convex training criteria (like CML) that more closely match the scoring function (like MPE).

- Finally, as discussed in this work, one potential application for C-Aug models is speech recognition. However, there is currently a mismatch between the monophone base models discussed in this thesis and the triphone and quinphone acoustic models used in many practical speech recognition systems. Although these acoustic models can be augmented using the same Taylor-series expansion as the models discussed in this thesis, care must be taken to understand the implications and practical effects of base model state-tying on the augmented sufficient statistics that are produced. Extensions to allow training and recognition to use word transcriptions instead of the phone transcriptions discussed in this thesis are also required.

A

HMM Derivatives

Consider an N-emitting-state HMM, with model parameters, $\boldsymbol{\lambda}$. For an observation sequence, $\boldsymbol{O} = \{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_T\}$, the HMM likelihood, $p(\boldsymbol{O}; \boldsymbol{\lambda})$, is given by,

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

where $a_{\theta_{t-1}\theta_t} \in \boldsymbol{\lambda}$ are transition probabilities, $p(\boldsymbol{o}_t|\theta_t; \boldsymbol{\lambda})$ are state-conditional output distributions, $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_T\}$ is a sequence of latent-states, and $\boldsymbol{\Theta}$ is the set of all possible state sequences. Note that, to simplify notation in the following sections, $\theta_t = \{j, m\}$ and $\theta_t = s_j$ are denoted as θ_t^{jm} and θ_t^j respectively.

A.1 Transition probability derivatives

Given the HMM log-likelihood in equation (A.1), consider the log-likelihood derivatives with respect to the transition probabilities, a_{ij},

$$\begin{aligned}
\nabla_{a_{ij}} \ln p(\boldsymbol{O}; \boldsymbol{\lambda}) &= \frac{1}{p(\boldsymbol{O}; \boldsymbol{\lambda})} \nabla_{a_{ij}} \left\{ \sum_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \prod_{t=1}^{T} a_{\theta_{t-1}\theta_t} p(\boldsymbol{o}_t | \theta_t; \boldsymbol{\lambda}) \right\} \\
&= \frac{1}{p(\boldsymbol{O}; \boldsymbol{\lambda})} \sum_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \sum_{t=1}^{T} \left\{ \frac{\prod_{\tau=1}^{T} a_{\theta_{\tau-1}\theta_\tau} p(\boldsymbol{o}_\tau | \theta_\tau; \boldsymbol{\lambda})}{a_{\theta_{t-1}\theta_t}} \right\} \nabla_{a_{ij}} (a_{\theta_{t-1}\theta_t}) \\
&= \sum_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \sum_{t=1}^{T} \left[\frac{p(\boldsymbol{O}|\boldsymbol{\theta}; \boldsymbol{\lambda})}{p(\boldsymbol{O}; \boldsymbol{\lambda})} \right] \left[\frac{\nabla_{a_{ij}} (a_{\theta_{t-1}\theta_t})}{a_{\theta_{t-1}\theta_t}} \right]
\end{aligned} \tag{A.2}$$

Since $\nabla_{a_{ij}}(a_{\theta_{t-1}\theta_t})$ is zero for all state transitions apart from a_{ij}, let $\boldsymbol{\theta}' \in \boldsymbol{\Theta}'$ to be the set of all state sequences that contain the transition from $\theta_{t-1} = i$ to $\theta_t = j$. Re-expressing equation (A.2) in terms of these constrained state sequences yields the derivatives,

$$\begin{aligned}\nabla_{a_{ij}} \ln p(\boldsymbol{O};\boldsymbol{\lambda}) &= \frac{1}{a_{ij}} \sum_{t=1}^{T} \left[\sum_{\boldsymbol{\theta}' \in \boldsymbol{\Theta}'} \frac{p(\boldsymbol{O}|\boldsymbol{\theta}';\boldsymbol{\lambda})}{p(\boldsymbol{O};\boldsymbol{\lambda})} \right] \\ &= \frac{1}{a_{ij}} \sum_{t=1}^{T} P(\theta_{t-1}^{i}, \theta_{t}^{j}|\boldsymbol{O};\boldsymbol{\lambda}) \end{aligned} \tag{A.3}$$

Since HMMs contain a sum-to-one constraint on the transition probabilities, Lagrange multipliers must be introduced to enforce this. The Lagrangian is therefore given by,

$$L(\boldsymbol{O};\boldsymbol{\lambda},\boldsymbol{\gamma}) = \ln p(\boldsymbol{O};\boldsymbol{\lambda}) + \sum_{i=1}^{N} \gamma_i \left(\sum_{j=1}^{N} a_{ij} - 1 \right) \tag{A.4}$$

Differentiating this Lagrangian with respect to the HMM transition probabilities, a_{ij}, and with respect to the Lagrange multipliers, $\boldsymbol{\gamma}$, yields the Lagrangian derivatives,

$$\nabla_{a_{ij}} L(\boldsymbol{O};\boldsymbol{\lambda},\boldsymbol{\gamma}) = \left(\frac{1}{a_{ij}} \sum_{t=1}^{T} P(\theta_{t-1}^{i}, \theta_{t}^{j}|\boldsymbol{O};\boldsymbol{\lambda}) \right) + \gamma_i \tag{A.5}$$

$$\nabla_{\gamma_i} L(\boldsymbol{O};\boldsymbol{\lambda},\boldsymbol{\gamma}) = \sum_{j=1}^{N} a_{ij} - 1 \tag{A.6}$$

Equating these to zero and rearranging, allows equation (A.5) to be re-expressed as,

$$\begin{aligned}\gamma_i &= -\frac{1}{a_{ij}} \sum_{t=1}^{T} P(\theta_{t-1}^{i}, \theta_{t}^{j}|\boldsymbol{O};\boldsymbol{\lambda}) \\ \gamma_i \sum_{j=1}^{N} a_{ij} &= -\sum_{t=1}^{T} \sum_{j=1}^{N} P(\theta_{t-1}^{i}, \theta_{t}^{j}|\boldsymbol{O};\boldsymbol{\lambda}) \\ \gamma_i &= -\sum_{t=1}^{T} P(\theta_{t-1}^{i}|\boldsymbol{O};\boldsymbol{\lambda}) \end{aligned} \tag{A.7}$$

Substituting this expression for γ_i into the Lagrangian function in equation (A.4) yields the constrained HMM log-likelihood derivative,

$$\nabla_{a_{ij}} \ln p(\boldsymbol{O};\boldsymbol{\lambda}) = \sum_{t=1}^{T} \frac{P(\theta_{t-1}^{i}, \theta_{t}^{j}|\boldsymbol{O};\boldsymbol{\lambda})}{a_{ij}} - P(\theta_{t-1}^{i}|\boldsymbol{O};\boldsymbol{\lambda}) \tag{A.8}$$

A.2 Output-distribution derivatives

In this work, both first-order and second-order derivatives of the HMM log-likelihood with respect to the output distribution parameters are used. In the next sections, derivations of these derivatives are given. Mixture-model output distributions are assumed.

A.2.1 First-order derivatives

Consider the derivatives of equation (A.1) with respect to the parameters, $\boldsymbol{\lambda}_{jm}$, of the output distribution associated with state/mixture-component, $\{j, m\}$,

$$\begin{aligned}
\nabla_{\boldsymbol{\lambda}_{jm}} \ln p(\boldsymbol{O};\boldsymbol{\lambda}) &= \frac{1}{p(\boldsymbol{O};\boldsymbol{\lambda})} \nabla_{\boldsymbol{\lambda}_{jm}} \left\{ \sum_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \prod_{t=1}^{T} a_{\theta_{t-1}\theta_t} p(\boldsymbol{o}_t|\theta_t;\boldsymbol{\lambda}) \right\} \\
&= \frac{1}{p(\boldsymbol{O};\boldsymbol{\lambda})} \sum_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \sum_{t=1}^{T} \left\{ \frac{\prod_{\tau=1}^{T} a_{\theta_{\tau-1}\theta_\tau} p(\boldsymbol{o}_\tau|\theta_\tau;\boldsymbol{\lambda})}{p(\boldsymbol{o}_t|\theta_t;\boldsymbol{\lambda})} \right\} \nabla_{\boldsymbol{\lambda}_{jm}} p(\boldsymbol{o}_t|\theta_t;\boldsymbol{\lambda}) \\
&= \sum_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \sum_{t=1}^{T} \left[\frac{p(\boldsymbol{O},\boldsymbol{\theta};\boldsymbol{\lambda})}{p(\boldsymbol{O};\boldsymbol{\lambda})} \right] \left[\frac{\nabla_{\boldsymbol{\lambda}_{jm}} p(\boldsymbol{o}_t|\theta_t;\boldsymbol{\lambda})}{p(\boldsymbol{o}_t|\theta_t;\boldsymbol{\lambda})} \right] \qquad \text{(A.9)}
\end{aligned}$$

Since $\nabla_{\boldsymbol{\lambda}_{jm}} \ln p(\boldsymbol{o}_t|\theta_t;\boldsymbol{\lambda}_{\theta_t})$ is zero for all $\theta_t \neq \{j, m\}$, let $\boldsymbol{\theta}' \in \boldsymbol{\Theta}'$ to be the set of all state sequences that pass through the state/mixture-component $\theta_t = \{j, m\}$. Re-expressing equation (A.9) in terms of these constrained state sequences yields the derivatives,

$$\begin{aligned}
\nabla_{\boldsymbol{\lambda}_{jm}} \ln p(\boldsymbol{O};\boldsymbol{\lambda}) &= \sum_{t=1}^{T} \left[\sum_{\boldsymbol{\theta}'\in\boldsymbol{\Theta}'} \frac{p(\boldsymbol{O},\boldsymbol{\theta}';\boldsymbol{\lambda})}{p(\boldsymbol{O};\boldsymbol{\lambda})} \right] \left[\nabla_{\boldsymbol{\lambda}_{jm}} \ln p(\boldsymbol{o}_t|\theta_t^{jm};\boldsymbol{\lambda}) \right] \\
&= \sum_{t=1}^{T} P(\theta_t^{jm}|\boldsymbol{O};\boldsymbol{\lambda}) \, \nabla_{\boldsymbol{\lambda}_{jm}} \ln p(\boldsymbol{o}_t|\theta_t^{jm};\boldsymbol{\lambda}) \qquad \text{(A.10)}
\end{aligned}$$

A.2.2 Second-order derivatives

Consider the second-order derivatives of equation (A.1) with respect to the parameters, $\boldsymbol{\lambda}_{jm}$, of the output distribution associated with state $\{j, m\}$. Start by calculating the first-derivative of the latent-state posterior,

$$\begin{aligned}
\nabla_{\boldsymbol{\lambda}_{kn}} P(\theta_t^{jm}|\boldsymbol{O};\boldsymbol{\lambda}) &= \nabla_{\boldsymbol{\lambda}_{kn}} \left[\frac{p(\boldsymbol{O},\theta_t^{jm};\boldsymbol{\lambda})}{p(\boldsymbol{O};\boldsymbol{\lambda})} \right] \\
&= \frac{1}{p(\boldsymbol{O};\boldsymbol{\lambda})} \nabla_{\boldsymbol{\lambda}_{kn}} p(\boldsymbol{O},\theta_t^{jm};\boldsymbol{\lambda}) - \frac{p(\boldsymbol{O},\theta_t^{jm};\boldsymbol{\lambda})}{p(\boldsymbol{O};\boldsymbol{\lambda})^2} \nabla_{\boldsymbol{\lambda}_{kn}} p(\boldsymbol{O};\boldsymbol{\lambda}) \\
&= \frac{1}{p(\boldsymbol{O};\boldsymbol{\lambda})} \nabla_{\boldsymbol{\lambda}_{kn}} p(\boldsymbol{O},\theta_t^{jm};\boldsymbol{\lambda}) - \frac{p(\boldsymbol{O},\theta_t^{jm};\boldsymbol{\lambda})}{p(\boldsymbol{O};\boldsymbol{\lambda})} \nabla_{\boldsymbol{\lambda}_{kn}} \ln p(\boldsymbol{O};\boldsymbol{\lambda}) \\
&= \left[\sum_{\tau=1}^{T} P(\theta_t^{jm},\theta_\tau^{kn}|\boldsymbol{O};\boldsymbol{\lambda}) \, \nabla_{\boldsymbol{\lambda}_{kn}} \ln p(\boldsymbol{o}_\tau|\theta_\tau^{kn};\boldsymbol{\lambda}) \right] \\
&\qquad - \left[P(\theta_t^{jm}|\boldsymbol{O};\boldsymbol{\lambda}) \sum_{\tau=1}^{T} P(\theta_\tau^{kn}|\boldsymbol{O};\boldsymbol{\lambda}) \, \nabla_{\boldsymbol{\lambda}_{kn}} \ln p(\boldsymbol{o}_\tau|\theta_\tau^{kn};\boldsymbol{\lambda}) \right] \\
&= \sum_{\tau=1}^{T} D(\theta_t^{jm},\theta_\tau^{kn}|\boldsymbol{O};\boldsymbol{\lambda}) \nabla_{\boldsymbol{\lambda}_{kn}} \ln p(\boldsymbol{o}_\tau|\theta_\tau^{kn};\boldsymbol{\lambda}) \qquad \text{(A.11)}
\end{aligned}$$

where

$$D(\theta_t^{jm},\theta_\tau^{kn}|\boldsymbol{O};\boldsymbol{\lambda}) = P(\theta_t^{jm},\theta_\tau^{kn}|\boldsymbol{O};\boldsymbol{\lambda}) - P(\theta_t^{jm}|\boldsymbol{O};\boldsymbol{\lambda}) P(\theta_\tau^{kn}|\boldsymbol{O};\boldsymbol{\lambda}) \qquad \text{(A.12)}$$

Given this posterior derivative, second-order derivatives of the HMM log-likelihood with respect to the output distribution parameters, $\boldsymbol{\lambda}_{jm}$ and $\boldsymbol{\lambda}_{kn}$, can be written as,

$$\begin{aligned}
\nabla_{\boldsymbol{\lambda}_{kn}} \nabla^{\mathrm{T}}_{\boldsymbol{\lambda}_{jm}} \ln p(\boldsymbol{O};\boldsymbol{\lambda}) &= \nabla_{\boldsymbol{\lambda}_{kn}} \left[\sum_{t=1}^{T} P(\theta_t^{jm}|\boldsymbol{O};\boldsymbol{\lambda}) \, \nabla^{\mathrm{T}}_{\boldsymbol{\lambda}_{jm}} \ln p(\boldsymbol{o}_t|\theta_t^{jm};\boldsymbol{\lambda}) \right] \\
&= \sum_{t=1}^{T} \left[\Big(\nabla_{\boldsymbol{\lambda}_{kn}} P(\theta_t^{jm}|\boldsymbol{O};\boldsymbol{\lambda}) \Big) \nabla^{\mathrm{T}}_{\boldsymbol{\lambda}_{jm}} \ln p(\boldsymbol{o}_t|\theta_t^{jm};\boldsymbol{\lambda}) \right] \\
&\qquad + \sum_{t=1}^{T} \left[P(\theta_t^{jm}|\boldsymbol{O};\boldsymbol{\lambda}) \nabla_{\boldsymbol{\lambda}_{kn}} \nabla^{\mathrm{T}}_{\boldsymbol{\lambda}_{jm}} \ln p(\boldsymbol{o}_t|\theta_t^{jm};\boldsymbol{\lambda}) \right] \\
&= \sum_{t=1}^{T} \sum_{\tau=1}^{T} D(\theta_t^{jm}, \theta_\tau^{kn}|\boldsymbol{O};\boldsymbol{\lambda}) \Big(\nabla_{\boldsymbol{\lambda}_{kn}} \ln p(\boldsymbol{o}_\tau|\theta_\tau^{kn};\boldsymbol{\lambda}) \Big) \Big(\nabla_{\boldsymbol{\lambda}_{jm}} \ln p(\boldsymbol{o}_t|\theta_t^{jm};\boldsymbol{\lambda}) \Big)^{\mathrm{T}} \\
&\qquad + \sum_{t=1}^{T} \left[P(\theta_t^{jm}|\boldsymbol{O};\boldsymbol{\lambda}) \nabla_{\boldsymbol{\lambda}_{kn}} \nabla^{\mathrm{T}}_{\boldsymbol{\lambda}_{jm}} \ln p(\boldsymbol{o}_t|\theta_t^{jm};\boldsymbol{\lambda}) \right] \qquad \text{(A.13)}
\end{aligned}$$

A.2.3 Derivatives for GMM output distributions

When the HMM state-dependent output distributions are defined by Gaussian mixture models (the most common scenario), derivatives of the state/mixture-component Gaussians can be written as,

$$\nabla_{\boldsymbol{\mu}_{jm}} \ln \mathcal{N}(\boldsymbol{o}_t|\boldsymbol{\mu}_{jm}, \boldsymbol{\Sigma}_{jm}) = \boldsymbol{\Sigma}_{jm}^{-1}(\boldsymbol{o}_t - \boldsymbol{\mu}_{jm}) \qquad \text{(A.14)}$$

$$\nabla_{\boldsymbol{\Sigma}_{jm}} \ln \mathcal{N}(\boldsymbol{o}_t|\boldsymbol{\mu}_{jm}, \boldsymbol{\Sigma}_{jm}) = \frac{1}{2} \left[-\boldsymbol{\Sigma}_{jm}^{-1} + \boldsymbol{\Sigma}_{jm}^{-1}(\boldsymbol{o}_t - \boldsymbol{\mu}_{jm})(\boldsymbol{o}_t - \boldsymbol{\mu}_{jm})^{\mathrm{T}} \boldsymbol{\Sigma}_{jm}^{-1} \right] \qquad \text{(A.15)}$$

First-order derivatives of an HMM with GMM output distributions are therefore written,

$$\begin{aligned}
\nabla_{a_{ij}} \ln \hat{p}(\boldsymbol{O};\boldsymbol{\lambda}) &= \sum_{t=1}^{T} \frac{P(\theta_{t-1}^{i}, \theta_t^{j}|\boldsymbol{O};\boldsymbol{\lambda})}{a_{ij}} - P(\theta_{t-1}^{j}|\boldsymbol{O};\boldsymbol{\lambda}) \\
\nabla_{c_{jm}} \ln \hat{p}(\boldsymbol{O};\boldsymbol{\lambda}) &= \sum_{t=1}^{T} \frac{P(\theta_t^{jm}|\boldsymbol{O};\boldsymbol{\lambda})}{c_{jm}} - P(\theta_t^{j}|\boldsymbol{O};\boldsymbol{\lambda}) \\
\nabla_{\boldsymbol{\mu}_{jm}} \ln \hat{p}(\boldsymbol{O};\boldsymbol{\lambda}) &= \sum_{t=1}^{T} P(\theta_t^{jm}|\boldsymbol{O};\boldsymbol{\lambda}) \, \boldsymbol{\Sigma}_{jm}^{-1}(\boldsymbol{o}_t - \boldsymbol{\mu}_{jm}) \\
\nabla_{\boldsymbol{\Sigma}_{jm}} \ln \hat{p}(\boldsymbol{O};\boldsymbol{\lambda}) &= \frac{1}{2} \sum_{t=1}^{T} P(\theta_t^{jm}|\boldsymbol{O};\boldsymbol{\lambda}) \Big[-\boldsymbol{\Sigma}_{jm}^{-1} + \boldsymbol{\Sigma}_{jm}^{-1}(\boldsymbol{o}_t - \boldsymbol{\mu}_{jm})(\boldsymbol{o}_t - \boldsymbol{\mu}_{jm})^{\mathrm{T}} \boldsymbol{\Sigma}_{jm}^{-1} \Big]
\end{aligned}$$

Selected second-order derivatives are,

$$\begin{aligned}\nabla_{\boldsymbol{\mu}_{kn}}\nabla^{\mathrm{T}}_{\boldsymbol{\mu}_{jm}}\ln\hat{p}(\boldsymbol{O};\boldsymbol{\lambda}) &= -\delta_{jk}\delta_{mn}P(\theta_t^{jm}|\boldsymbol{O};\boldsymbol{\lambda})\boldsymbol{\Sigma}_{jm}^{-1} \\ &\quad + \sum_{t=1}^{T}\sum_{\tau=1}^{T}D(\theta_t^{jm},\theta_\tau^{kn}|\boldsymbol{O};\boldsymbol{\lambda})\boldsymbol{\Sigma}_{kn}^{-1}(\boldsymbol{o}_\tau-\boldsymbol{\mu}_{kn})(\boldsymbol{o}_t-\boldsymbol{\mu}_{jn})^{\mathrm{T}}\boldsymbol{\Sigma}_{jm}^{-1} \\ \nabla_{c_{kn}}\nabla^{\mathrm{T}}_{c_{jm}}\ln\hat{p}(\boldsymbol{O};\boldsymbol{\lambda}) &= -\sum_{t=1}^{T}\frac{2\delta_{jk}\delta_{mn}P(\theta_t^{jm}|\boldsymbol{O};\boldsymbol{\lambda})}{c_{jm}c_{kn}} \\ &\quad + \sum_{t=1}^{T}\sum_{\tau=1}^{T}\left[\frac{D(\theta_t^{jm},\theta_\tau^{kn}|\boldsymbol{O};\boldsymbol{\lambda})}{c_{jm}c_{kn}} - \frac{D(\theta_t^{j},\theta_\tau^{kn}|\boldsymbol{O};\boldsymbol{\lambda})}{c_{kn}}\right. \\ &\quad \left. - \frac{D(\theta_t^{jm},\theta_\tau^{k}|\boldsymbol{O};\boldsymbol{\lambda})}{c_{jm}} + D(\theta_t^{j},\theta_\tau^{k}|\boldsymbol{O};\boldsymbol{\lambda})\right]\end{aligned}$$

where,

$$D(\theta_t^{jm},\theta_\tau^{kn}|\boldsymbol{O};\boldsymbol{\lambda}) = P(\theta_t^{jm},\theta_\tau^{kn}|\boldsymbol{O};\boldsymbol{\lambda}) - P(\theta_t^{jm}|\boldsymbol{O};\boldsymbol{\lambda})P(\theta_\tau^{jm}|\boldsymbol{O};\boldsymbol{\lambda})$$

Derivatives with respect to the variances are similar to those for the means and so are omitted. Cross derivatives are omitted for brevity.

B

HMM Forward-Backward and Viterbi Algorithms

B.1 The Forward-Backward algorithm

The Forward-Backward algorithm [7, 100] is an efficient dynamic programming algorithm for calculating the posterior probabilities of the latent-states. The algorithm name refers to the two passes through the data that are required: the first starting at time 1 and working forwards in time, and the second starting a time T and working backwards in time.

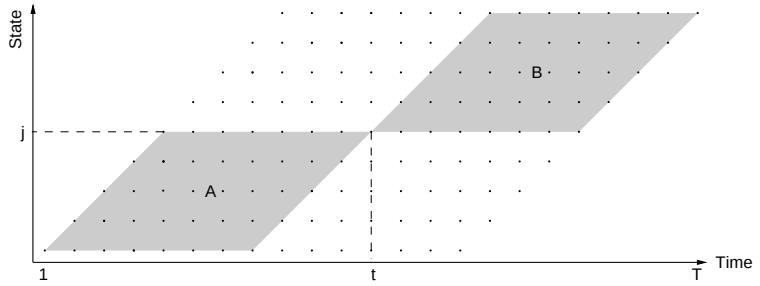


Figure B.1: A pictorial representation of the Forward-Backward algorithm

First consider the state/time diagram in figure B.1. As illustrated, the posterior probability, $P(\theta_t^j|\boldsymbol{O};\boldsymbol{\lambda})$, is calculated as the sum of the weights of the paths that pass through the point (t, j), divided by the total weight of all possible paths. This can be written as,

$$P(\theta_t^j|\boldsymbol{O};\boldsymbol{\lambda}) = \frac{p(\boldsymbol{O},\theta_t^j;\boldsymbol{\lambda})}{p(\boldsymbol{O};\boldsymbol{\lambda})} = \frac{\alpha_j(t)\beta_j(t)}{p(\boldsymbol{O};\boldsymbol{\lambda})} \tag{B.1}$$

where $\alpha_j(t)$ and $\beta_j(t)$ are known as the forward and backward probabilities, respectively.

Pictorially, the forward and backward probabilities represent the total weight of all paths in regions A and B in figure B.1. They are defined by the expressions,

$$\alpha_j(t) = p(\boldsymbol{o}_1, \ldots, \boldsymbol{o}_t, \theta_t^j; \boldsymbol{\lambda}) \tag{B.2}$$

$$\beta_i(t) = p(\boldsymbol{o}_{t+1}, \ldots, \boldsymbol{o}_T | \theta_t^i; \boldsymbol{\lambda}) \tag{B.3}$$

Using the HMM assumption that states are conditionally-independent given the previous state, both $\alpha_j(t)$ and $\beta_i(t)$ can be calculated recursively using the relations,

$$\alpha_j(t) = \left[\sum_{i=2}^{N-1} \alpha_i(t-1) a_{ij} \right] b_j(\boldsymbol{o}_t) \tag{B.4}$$

$$\beta_i(t) = \sum_{j=2}^{N-1} a_{ij} b_j(\boldsymbol{o}_{t+1}) \beta_j(t+1) \tag{B.5}$$

Initial conditions for these recursions are given by,

$$\alpha_j(1) = \begin{cases} 1 & \text{if } j = 1 \\ a_{1j} b_j(\boldsymbol{o}_1) & \text{otherwise} \end{cases} \tag{B.6}$$

$$\beta_i(T) = a_{iN} \tag{B.7}$$

where a_{ij} are the HMM transition probabilities and $b_j(\boldsymbol{o}_t)$ is the state-conditional output distribution for the HMM state j. The observation sequence likelihood—the normalisation term in equation (B.1)—can be calculated using the forward probability for time T.

B.2 The Viterbi algorithm

HMM inference is usually based upon the Viterbi algorithm. This is a computationally efficient algorithm that approximates the observation sequence likelihood using the maximum likelihood state sequence. This is calculated using a modified version of the forward probability in equations (B.4) and (B.6). The difference between the two algorithms is that the Viterbi algorithm replaces the summation by the maximum operation. Partial likelihoods are therefore calculated using the recursion,

$$\phi_j(t) = \max_i \Big[\alpha_i(t-1) a_{ij} \Big] b_j(\boldsymbol{o}_t) \tag{B.8}$$

with initial conditions,

$$\phi_j(1) = \begin{cases} 1 & \text{if } j = 1 \\ a_{1j} b_j(\boldsymbol{o}_1) & \text{otherwise} \end{cases} \tag{B.9}$$

for all $j \in (1, N)$. It is easy to extend the Viterbi algorithm to the task of continuous speech recognition using the token passing algorithm [140].

B.3 The Double-Forward-Backward algorithm

Second-order statistics are more complex to calculate than their first-order equivalents since they require joint posteriors of discontiguous states to be calculated. In this work, an efficient extension of the standard Forward-Backward algorithm is proposed for calculating these posteriors. This algorithm is discussed in more detail below.

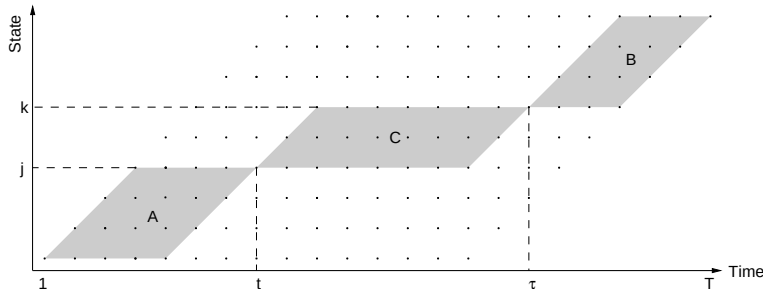


Figure B.2: A pictorial representation of the Double-Forward-Backward algorithm

Consider the state/time diagram in figure B.2. The joint posterior, $P(\theta_t^j, \theta_\tau^k|\boldsymbol{O};\boldsymbol{\lambda})$ is calculated as the sum of the weights of the paths (marked in grey) that pass through the points (t, j) and (τ, k), divided by the total weight of all possible paths. This calculation may be written as,

$$P(\theta_t^{jm}, \theta_\tau^{kn}|\boldsymbol{O};\boldsymbol{\lambda}) = \frac{\alpha_{jm}(t)\, p(\boldsymbol{o}_{t+1}, \ldots, \boldsymbol{o}_\tau, \theta_\tau^{kn}|\theta_t^{jm};\boldsymbol{\lambda})\, \beta_{kn}(\tau)}{p(\boldsymbol{O};\boldsymbol{\lambda})} \tag{B.10}$$

where $\alpha_{jm}(t)$ and $\beta_{kn}(\tau)$ are the standard forward and backward probabilities discussed in the previous section. They calculate the total weight of all paths in the regions A and B in figure B.2. The middle term, $p(\boldsymbol{o}_{t+1}, \ldots, \boldsymbol{o}_\tau, \theta_\tau = k|\theta_t = j;\boldsymbol{\lambda})$ corresponds to region C and represents the total weight of all paths that start in state θ_t^{jm} and end in state θ_τ^{kn}. With varying start and end points, computational cost of this term scales quadratically with the number of HMM states, mixture-components and sequence length.

Putting the pieces together, calculation of $P(\theta_t^{jm}, \theta_\tau^{kn}|\boldsymbol{O};\boldsymbol{\lambda})$ proceeds according to the algorithm in figure B.3. Computational complexity of this algorithm is quadratic in the number of HMM states, mixture-components and sequence length. Storage requirements for the intermediate calculations vary linearly with these factors.

```
# Initialisation
Calculate and cache forward and backward probabilities, α_jm(t) and β_jm(t)

# Perform Double-Forward-Backward
Foreach t = 1 → T
    Foreach j = 1 → N, m = 1 → M
        # Propagate paths that pass through state θ_jm(t)
        # all other paths are ignored (probabilities zeroed)
        Foreach k = 1 → N, n = 1 → M
            Let γ_kn(t) = 0
        End
        Let γ_jm(t) = α_jm(t)

        # Perform forward pass, starting at state θ_jm(t),
        ending in state θ_kn(τ)
        Foreach τ = t + 1 → T
            Foreach k = 1 → N, n = 1 → M
                Calculate new forward probability, γ_kn(τ) (section B.1)
                Let P(θ_t^jm, θ_τ^kn | O; λ) = γ_kn(τ)β_kn(τ)/p(O; λ)
            End
        End
    End
End
```

Figure B.3: Double-Forward-Backward algorithm for calculating HMM joint-posterior probabilities

C

Variable-Margin Support Vector Machines

Consider the standard soft-margin primal SVM objective function,

$$\begin{aligned} \text{minimise}_{\boldsymbol{w}} \quad & \tfrac{1}{2}\langle \boldsymbol{w}, \boldsymbol{w} \rangle + C \sum_{i=1}^{n} \epsilon_i & & \text{(C.1)} \\ \text{subject to} \quad & y_i\big(\langle \boldsymbol{w}, \boldsymbol{\phi}^{\text{LL}}(\boldsymbol{O}_i; \boldsymbol{\lambda}) \rangle + b\big) \geq 1 - \epsilon_i & \forall i \in [1, n] & \\ \text{and} \quad & \epsilon_i \geq 0 & \forall i \in [1, n] & \end{aligned}$$

with the additional constraint,

$$w_1 = 1 \tag{C.2}$$

Splitting the terms that relate to the log-likelihood ratio of the base models from the score-space, $\boldsymbol{\phi}^{\text{LL}}(\boldsymbol{O}_i; \boldsymbol{\lambda})$, and hyperplane gradient, $\boldsymbol{w}$, allows them to be written as,

$$\boldsymbol{\phi}^{\text{LL}}(\boldsymbol{O}_i; \boldsymbol{\lambda}) = \begin{bmatrix} \phi^{\text{LLR}}(\boldsymbol{O}; \boldsymbol{\lambda}) \\ \boldsymbol{\phi}^{\text{LL}'}(\boldsymbol{O}; \boldsymbol{\lambda}) \end{bmatrix} \qquad \boldsymbol{w} = \begin{bmatrix} 1 \\ \boldsymbol{w}' \end{bmatrix} \tag{C.3}$$

where $\phi^{\text{LLR}}(\boldsymbol{O}; \boldsymbol{\lambda})$ is a one-dimensional score-space containing only the log-likelihood ratio of the base models, and $\boldsymbol{\phi}^{\text{LL}'}(\boldsymbol{O}; \boldsymbol{\lambda})$ is a score-space containing all other terms from the augmented model score-space. Substituting these into the soft-margin constraint in equation (C.1) yields the following variable-margin constraint,

$$\begin{aligned} & y_i\Big(\phi^{\text{LLR}}(\boldsymbol{O}_i; \boldsymbol{\lambda}) + \big\langle \boldsymbol{w}', \boldsymbol{\phi}^{\text{LL}'}(\boldsymbol{O}_i; \boldsymbol{\lambda}) \big\rangle + b\Big) \geq 1 - \epsilon_i \\ \Rightarrow \quad & y_i\Big(\big\langle \boldsymbol{w}', \boldsymbol{\phi}^{\text{LL}'}(\boldsymbol{O}_i; \boldsymbol{\lambda}) \big\rangle + b\Big) \geq 1 - y_i \phi^{\text{LLR}}(\boldsymbol{O}_i; \boldsymbol{\lambda}) - \epsilon_i \end{aligned} \tag{C.4}$$

where the standard SVM margin is replaced by the variable-margin, $1 - y_i\phi^{\texttt{LLR}}(\boldsymbol{O}_i;\boldsymbol{\lambda})$. The primal objective function for the variable-margin SVM is thus given by,

$$\begin{aligned} \text{minimise}_{\boldsymbol{w}'} \quad & \tfrac{1}{2}\langle \boldsymbol{w}', \boldsymbol{w}'\rangle + C\sum_{i=1}^{n}\epsilon_i & \text{(C.5)} \\ \text{subject to} \quad & y_i\big(\langle \boldsymbol{w}', \boldsymbol{\phi}^{\texttt{LL}'}(\boldsymbol{O}_i;\boldsymbol{\lambda})\rangle + b\big) \geq 1 - y_i\phi^{\texttt{LLR}}(\boldsymbol{O}_i;\boldsymbol{\lambda}) - \epsilon_i \qquad & \forall i \in [1,n] \\ \text{and} \quad & \epsilon_i \geq 0 & \forall i \in [1,n] \end{aligned}$$

This is a convex function of $\boldsymbol{w}'$ with linear constraints. Using the method of Lagrange multipliers, the optimisation in equation (C.5) can be expressed into its dual form. First, construct the primal Lagrangian function,

$$\begin{aligned} L^{\texttt{primal}}(\boldsymbol{w}', b, \boldsymbol{\epsilon}, \boldsymbol{\alpha}^{\texttt{svm}}, \boldsymbol{\beta}^{\texttt{svm}}) = {} & \tfrac{1}{2}\langle \boldsymbol{w}', \boldsymbol{w}'\rangle + C\sum_{i=1}^{n}\epsilon_i \\ & - \sum_{i=1}^{n}\alpha_i^{\texttt{svm}}\Big[y_i\big(\langle \boldsymbol{w}', \boldsymbol{\phi}^{\texttt{LL}'}(\boldsymbol{O};\boldsymbol{\lambda})\rangle + b\big) - 1 + y_i\phi^{\texttt{LLR}}(\boldsymbol{O}_i;\boldsymbol{\lambda}) + \epsilon_i\Big] \\ & - \sum_{i=1}^{n}\beta_i^{\texttt{svm}}\epsilon_i \end{aligned} \quad \text{(C.6)}$$

where $\alpha_i^{\texttt{svm}} \geq 0$ and $\beta_i^{\texttt{svm}} \geq 0$ are known as the Lagrange multipliers. The corresponding dual objective is found by differentiating $L(\boldsymbol{w}', b, \boldsymbol{\epsilon}, \boldsymbol{\alpha}^{\texttt{svm}}, \boldsymbol{\beta}^{\texttt{svm}})$ with respect to W', b and ϵ, and equating the resulting equations to zero [22],

$$\frac{\partial L(\boldsymbol{w}', b, \boldsymbol{\epsilon}, \boldsymbol{\alpha}^{\texttt{svm}}, \boldsymbol{\beta}^{\texttt{svm}})}{\partial \boldsymbol{w}'} = \boldsymbol{w}' - \sum_{i=1}^{n}\alpha_i^{\texttt{svm}} y_i \boldsymbol{\phi}^{\texttt{LL}'}(\boldsymbol{O};\boldsymbol{\lambda}) = \mathbf{0} \quad \text{(C.7)}$$

$$\frac{\partial L(\boldsymbol{w}', b, \boldsymbol{\epsilon}, \boldsymbol{\alpha}^{\texttt{svm}}, \boldsymbol{\beta}^{\texttt{svm}})}{\partial \epsilon_i} = C - \alpha_i^{\texttt{svm}} - \beta_i^{\texttt{svm}} = 0 \quad \text{(C.8)}$$

$$\frac{\partial L(\boldsymbol{w}', b, \boldsymbol{\epsilon}, \boldsymbol{\alpha}^{\texttt{svm}}, \boldsymbol{\beta}^{\texttt{svm}})}{\partial b} = \sum_{i=1}^{n}\alpha_i^{\texttt{svm}} y_i = 0 \quad \text{(C.9)}$$

Rearranging equations (C.7)–(C.9) and substituting them into the Lagrangian in equation (C.6) allows the dual Lagrangian to be written as,

$$\begin{aligned} L^{\texttt{dual}}(\boldsymbol{\alpha}^{\texttt{svm}}) = {} & \\ & \sum_{i=1}^{n}\alpha_i^{\texttt{svm}}\Big(1 - y_i\phi^{\texttt{LLR}}(\boldsymbol{O}_i;\boldsymbol{\lambda})\Big) - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i^{\texttt{svm}}\alpha_j^{\texttt{svm}} y_i y_j K^{\texttt{LL}'}(\boldsymbol{O}_i, \boldsymbol{O}_j;\boldsymbol{\lambda}) \end{aligned} \quad \text{(C.10)}$$

where

$$K^{\texttt{LL}'}(\boldsymbol{O}_i, \boldsymbol{O}_j;\boldsymbol{\lambda}) = \big\langle \boldsymbol{\phi}^{\texttt{LL}'}(\boldsymbol{O}_i;\boldsymbol{\lambda}), \boldsymbol{\phi}^{\texttt{LL}'}(\boldsymbol{O}_j;\boldsymbol{\lambda})\big\rangle \quad \text{(C.11)}$$

Since the primal problem required minimisation of the primal Lagrangian with respect to $\boldsymbol{w}'$ and b, the dual problem requires maximisation of the dual Lagrangian with respect to the parameters $\boldsymbol{\alpha}^{\texttt{svm}}$. There are two constraints associated with this maximisation.

The first, from equation (C.9), is that $\sum_{i=1} \alpha_i^{\texttt{svm}} y_i = 0$. The second, arising from the constraints $\alpha_i^{\texttt{svm}} \geq 0$ and $\beta_i^{\texttt{svm}} \geq 0$, coupled with the relationship in equation (C.8), is that $0 \leq \alpha_i^{\texttt{svm}} \leq C$. The dual objective function and constraints for variable-margin SVM training are thus given by,

$$\begin{aligned} \text{maximise}_{\boldsymbol{\alpha}^{\texttt{svm}}} \quad & \sum_{i=1}^{n} \alpha_i^{\texttt{svm}} \Big(1 - y_i \phi^{\texttt{LLR}}(\boldsymbol{O}_i; \boldsymbol{\lambda})\Big) - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i^{\texttt{svm}} \alpha_j^{\texttt{svm}} y_i y_j K^{\texttt{LL}'}(\boldsymbol{O}_i, \boldsymbol{O}_j; \boldsymbol{\lambda}) \\ \text{subject to} \quad & \sum_{i=1}^{n} \alpha_i^{\texttt{svm}} y_i = 0 \\ \text{and} \quad & 0 \leq \alpha_i^{\texttt{svm}} \leq C \end{aligned} \tag{C.12}$$

D

Code-Breaking Pairs

In chapter 8, code-breaking (section 6.4) was used for a number of LVCSR and TIMIT experiments. This appendix gives information about the individual word/phone classifiers.

D.1 LVCSR word pairs

Table D.1 provides a breakdown of the pairs used to rescore the LVCSR task in section 8.2. The order of the word pairs is identical to the order in which they were used for rescoring. For each data set, the number of occurrences of each word pair are given.

Table D.1: LVCSR rescoring pairs for `eval02`, `eval03` and `eval04`.

Pair #	Pair	# occurrences of word pair			
		`fsh2004sub`	`eval02`	`eval03`	`eval04`
1.	CAN/CAN'T	7,522	159	156	75
2.	YEAH/YOU	4,670	143	159	85
3.	THE/TO	4,144	127	81	75
4.	THEIR/THERE	1,318	59	72	49
5.	YEAH/YES	3,848	170	213	107
6.	KNOW/NOW	2,424	59	72	34
7.	I/THAT	1,074	41	28	20
8.	IS/WAS	1,498	56	42	25
9.	BUT/THAT	3,308	135	114	71
10.	A/THE	17,066	484	328	227
	Total words rescored	60,941	1,433	1,265	768
	Total words in data set	–	65,236	76,157	36,781
	Fraction of data rescored	–	2.2%	1.7%	2.0%

D.2 TIMIT phone pairs

Tables D.2 and D.3 give individual results for the ten phone pairs used to rescore the ML and MMI baselines in section 8.3.1. The order of the phones in each table is identical to the order in which the classifiers were used for rescoring. For each of the training, development and core test sets, the following information is given:

- number of occurrences of phone-pair in data set;
- percentage error rate for a three-state, four-mixture-component HMM base model;
- percentage error rate for an augmented model with a three-state, four-mixture-component HMM base model and sufficient statistics defined using derivatives with respect to the means, variances and mixture-component priors.

Note that the rescoring results in section 8.3.1 cannot be derived from these numbers since the baseline system being rescored is a three-state, ten-mixture-component system.

Table D.2: TIMIT rescoring pairs for ML baseline.

Pair		Training set			Development set			Core test set		
		Pairs (#)	HMM (%)	Aug (%)	Pairs (#)	HMM (%)	Aug (%)	Pairs (#)	HMM (%)	Aug (%)
1.	er/r	5,332	21.5	10.2	1,943	24.5	17.3	317	25.2	14.8
2.	m/n	3,230	20.1	9.4	1,198	20.9	14.8	195	26.7	20.0
3.	s/z	5,398	20.1	10.1	2,329	24.1	17.1	397	21.9	16.4
4.	n/ng	1,634	20.1	9.2	737	26.5	22.7	123	26.0	23.6
5.	dx/n	1,072	13.4	6.1	317	17.7	14.2	54	25.9	24.1
6.	l/w	2,068	15.4	6.7	830	18.8	13.4	128	25.0	16.4
7.	iy/y	1,220	12.1	5.6	528	16.1	14.8	84	16.7	15.5
8.	b/p	1,034	18.2	12.8	335	22.7	19.4	62	17.7	19.4
9.	d/t	1,118	21.3	13.0	499	29.5	29.7	83	31.3	28.9
10.	eh/ih	1,032	25.9	11.9	282	36.9	33.3	44	29.6	34.1

Table D.3: TIMIT rescoring pairs for MMI baseline.

Pair		Training set			Development set			Core test set		
		Pairs (#)	HMM (%)	Aug (%)	Pairs (#)	HMM (%)	Aug (%)	Pairs (#)	HMM (%)	Aug (%)
1.	m/n	3,264	20.9	9.1	1,206	23.0	15.7	204	22.1	15.7
2.	ae/eh	1,558	25.4	11.6	516	33.3	27.1	70	41.4	32.9
3.	b/p	994	17.5	11.0	333	23.7	21.9	60	18.3	23.3
4.	s/z	5,298	19.9	10.0	2,323	24.0	17.6	391	20.7	15.4
5.	er/r	4,964	22.1	10.3	1,807	25.0	18.5	298	26.5	17.8
6.	ch/jh	702	16.4	6.6	240	22.9	25.0	44	34.1	34.1
7.	d/t	1,262	22.2	13.0	513	29.6	28.3	86	27.9	25.6
8.	eh/ih	956	26.8	13.3	280	36.4	33.9	55	34.6	29.1
9.	ch/sh	522	9.0	3.6	267	12.0	9.0	50	14.0	12.0
10.	n/ng	1,684	19.0	9.0	762	24.3	21.8	125	24.0	25.6

References

[1] M. Aizerman, E. Braverman, and L. Rozonoer. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.

[2] S. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, February 1998.

[3] S. Amari and S. Wu. *Methods of Information Geometry*. Oxford University Press, 2000. (Translations of Mathematical Monographs, Volume 191, American Mathematical Society).

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

[5] 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*, pages I:49–52, Tokyo, Japan, April 1986.

[6] J.K. Baker. The DRAGON system—an overview. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 23(1):24–29, February 1975.

[7] 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.

[8] J.A. Bilmes. Buried Markov models for speech recognition. In *Proc. ICASSP*, pages II:713–716, Phoenix, AZ, March 1999.

[9] J.A. Bilmes. Buried Markov models: A graphical-modelling approach to automatic speech recognition. *Computer Speech and Language*, 17(2–3):213–231, 2003.

[10] B.E. Boser, I.M. Guyon, and V.N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory (COLT)*, pages 144–152. ACM Press, 1992.

[11] C. Campbell. Kernel methods: A survey of current techniques. *Neurocomputing*, 48:63–84, 2002.

[12] 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 & Language*, 20:210–229, 2006.

[13] O. Cappé, E. Moulines, and T. Rydén. *Inference in Hidden Markov Models*. Springer, 2005. Springer Series in Statistics.

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

[15] W. Chou, C.H. Lee, and B.H. Juang. Minimum error rate training based on N-best string models. In *Proc. ICASSP*, pages II:652–655, Minneapolis, USA, April 2005.

[16] C. Cieri, D. Miller, and K. Walker. From Switchboard to Fisher: Telephone collection protocols, their uses and yields. In *Eurospeech*, 2003.

[17] C. Cortes, P. Haffner, and M. Mohri. Positive definite rational kernels. In *16th Annual Conference on Computational Learning Theory (COLT 2003)*, pages 656–670, Washington D.C., August 2003.

[18] C. Cortes, P. Haffner, and M. Mohri. Rational kernels: Theory and algorithms. *Journal of Machine Learning Research*, 5:1035–1062, 2004.

[19] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.

[20] 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.

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

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

[23] J.N. Darroch and D. Ratcliff. Generalised iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 43(5):1470–1480, October 1972.

[24] S.B. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28:357–366, August 1980.

[25] M.H. DeGroot and M.J. Schervish. *Probability and Statistics*. Addison-Wesley, 3rd edition, 2002.

[26] J.R. Deller, J.G. Proakis, and J.H.L. Hansen. *Discrete-Time Processing of Speech Signals*. Prentice Hall, 1993.

[27] 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.

[28] T.G. Dietterich and G. Bakiri. Error-correcting output codes: A general method for improving multiclass inductive learning programs. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, pages 572–577. AAAI Press, 1991.

[29] V.V. Digalakis. *Segment-based Stochastic Models of Spectral Dynamics for Continuous Speech Recognition*. PhD thesis, Boston University Graduate School, 1992.

[30] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. John Wiley & Sons, 2nd edition, 2001.

[31] G. Evermann, H.Y. Chan, M.J.F. Gales, B. Jia, D. Mrva, P.C. Woodland, and K. Yu. Training LVCSR systems on thousands of hours of data. In *Proc. ICASSP*, pages 209–212, Philadelphia, USA, March 2005.

[32] G.D. Forney. The Viterbi algorithm. *Proceedings IEEE*, 61:268–278, March 1973.

[33] 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, February 1986.

[34] M.J.F. Gales. Maximum likelihood multiple subspace projection schemes for hidden Markov models. *IEEE Transactions on Speech and Audio Processing*, 10(2):37–47, February 2002.

[35] M.J.F. Gales and M.I. Layton. SVMs, score-spaces and maximum margin statistical models. In *Beyond HMM workshop, ATR*, 2004.

[36] M.J.F. Gales and M.I. Layton. Training augmented models using SVMs. *IEICE Transactions on Information and Systems*, E89-D(3):892–899, March 2006.

[37] S.I. Gallant. Perceptron-based learning algorithms. *IEEE Transactions on Neural Networks*, 1(2):179–191, 1990.

[38] J.S. Garofolo. *DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus CD-ROM*. National Institute of Standards and Technology (NIST), 1993.

[39] Z. Ghahramani and M.I. Jordan. Factorial hidden Markov models. *Machine Learning*, 29(2–3):245–273, November/December 1998.

[40] L. Gillick and S. Cox. Some statistical issues in the comparison of speech recognition algorithms. In *Proc. ICASSP*, volume 1, pages 532–535, Glasgow, UK, May 1989.

[41] A. Graham. *Kronecker Products and Matrix Calculus With Applications*. Ellis Horwood, Chichester, 1st edition, 1981.

[42] A. Gunawardana, M. Mahajan, A. Acero, and J.C. Platt. Hidden conditional random fields for phone classification. In *Interspeech*, 2005.

[43] T. Hain, P.C. Woodland, G. Evermann, M.J.F. Gales, A. Liu, G. Moore, D. Povey, and L. Wang. Automatic transcription of conversational telephone speech – development of the CU-HTK 2002 system. Technical Report CUED/F-INFENG/TR.465, Cambridge University Engineering Department, 2003.

[44] A.K. Halberstadt and J.R. Glass. Heterogeneous acoustic measurements for phonetic classification. In *Eurospeech*, pages 401–404, 1997.

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

[46] H. Hermansky. Perceptual linear prediction (PLP) of speech. *Journal of the Acoustic Society of America*, 87(4):1738–1752, April 1990.

[47] S. Hua and Z. Sun. A novel method of protein secondary structure prediction with high segment overlap measure: Support vector machine approach. *Journal of Molecular Biology*, 308(2):397–407, 2001.

[48] X. Huang, A. Acero, and H. Hon. *Spoken Language Processing*. Prentice Hall, 2001.

[49] T. Jaakkola, M. Diekhans, and D. Hausser. Using the Fisher kernel to detect remote protein homologies. In *Proceedings of the 7th International Conference on Intelligent Systems for Molecular Biology*, pages 149–158. AAAI Press, August 1999.

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

[51] 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, January 2000.

[52] F. Jelinek. Continuous speech recognition by statistical methods. *IEEE Proceedings*, 64(4):532–556, April 1976.

[53] F. Jelinek. *Statistical Methods for Speech Recognition*. MIT Press, Cambridge, Massachusetts, 1997.

[54] T. Joachims. Text categorisation with support vector machines: Learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, pages 137–142, Chemnitz, DE, 1998. Springer Verlag, Heidelburg.

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

[56] B.H. Juang, W. Chou, and C.H. Lee. Minimum classification error rate methods for speech recognition. *IEEE Transactions on Speech and Audio Processing*, 5(3):257–265, May 1997.

[57] Z. Kaiser, B. Horvat, and Z. Kacic. A novel loss function for the overall risk criterion based discriminative training of HMM models. In *International Conference on Spoken Language Processing*, Beijing, China, October 2000.

[58] Z. Kaiser, B. Horvat, and Z. Kacic. Overall risk criterion estimation of hidden Markov model parameters. *Speech Communication*, 38(3–4):383–398, 2002.

[59] S.M. Katz. Estimation of probabilities from sparse data for the language-model component of a speech recogniser. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 35(3):400–411, March 1987.

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

[61] A. Krogh, M. Brown, I.S. Mian, K. Sjölander, and D. Haussler. Hidden Markov models in computational biology: Applications to protein modelling. *Journal of Molecular Biology*, 235:1501–1531, February 1994.

[62] N. Kumar. *Investigation of Silicon-Auditory Models and Generalisation of Linear Discriminant Analysis for Improved Speech Recognition*. PhD thesis, Johns Hopkins University, Baltimore, 1997.

[63] J. Lafferty, A. McCallum, and F. Pereira. Condition random fields: Probabilistic models for segmenting and labelling sequence data. In *ICML*, pages 591–598, 2001.

[64] J. Lafferty, X. Zhu, and Y. Liu. Kernel conditional random fields: Representation and clique selection. In *ICML*, 2004.

[65] M.I. Layton and M.J.F. Gales. Maximum margin training of generative kernels. Technical Report CUED/F-INFENG/TR.484, Cambridge University Engineering Department, June 2004.

[66] M.I. Layton and M.J.F. Gales. Acoustic modelling using continuous rational kernels. In *IEEE International Workshop on Machine Learning for Signal Processing*, pages 165–170, September 2005.

[67] M.I. Layton and M.J.F. Gales. Augmented statistical models: Exploiting generative models in discriminative classifiers. In *NIPS workshops*, Whistler, Canada, 2005.

[68] M.I. Layton and M.J.F. Gales. Acoustic modelling using continuous rational kernels. *Journal of VLSI Signal Processing*, 2006. To appear.

[69] M.I. Layton and M.J.F. Gales. Augmented statistical models for speech recognition. In *Proc. ICASSP*, Toulouse, France, May 2006.

[70] L. Lee and R.C. Rose. Speaker normalisation using efficient frequency warping procedures. In *Proc. ICASSP*, pages I:353–356, Atlanta, USA, May 1996.

[71] C. Leslie and R. Kuang. Fast string kernels using inexact matching for protein sequences. *Journal of Machine Learning Research*, 5:1435–1455, December 2004.

[72] X. Liu, M.J.F. Gales, K.C. Sim, and K. Yu. Investigation of acoustic modelling techniques for LVCSR systems. In *Proc. ICASSP*, pages 849–852, Philadelphia, USA, March 2005.

[73] 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.

[74] M. Mahajan, A. Gunawardana, and A. Acero. Training algorithms for hidden conditional random fields. In *Proc. ICASSP*, pages 273–276, Toulouse, France, May 2006.

[75] L. Mangu, E. Brill, and A. Stolcke. Finding consensus among words: Lattice-based word error minimisation. In *Proc. Eurospeech*, pages 495–498, 1999.

[76] R.A. Maxion and T.N Townsend. Masquerade detection using truncated command lines. In *Proc. of the International Conference on Dependable Systems and Networks (DSN02)*, pages 219–228, Washington, DC, June 2002. IEEE Computer Society Press.

[77] A. McCallum. Efficiently inducing features of conditional random fields. In *19th Conference on Uncertainty in Artificial Intelligence (UAI03)*, 2003.

[78] A. McCallum, D. Freitag, and F. Pereira. Maximum entropy Markov models for information extraction and segmentation. In *Proc. 17th International Conference on Machine Learning*, pages 591–598, San Francisco, CA, 2000. Morgan Kaufman.

[79] 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.

[80] M. Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311, 1997.

[81] M. Mohri. Semiring frameworks and algorithms for shortest-distance problems. *Journal of Automata, Languages and Combinatorics*, 7:321–350, 2002.

[82] M. Mohri, F. Pereira, and M. Riley. Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, 16:69–88, January 2002.

[83] M. Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6:525–533, June 1993.

[84] M. Møller. *Efficient Training of Feed-Forward Neural Networks*. PhD thesis, Computer Science Department, Aarhus University, Denmark, November 1997.

[85] 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 *Eurospeech*, pages 97–100, 1995.

[86] 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 on Acoustics, Speech, and Signal Processing*, 31(4):814–817, August 1993.

[87] J. Nocedal and S.J. Wright. *Numerical Optimisation*. Springer Series on Operations Research. Springer, 1999.

[88] H. Nock. *Techniques for modelling Phonological Processes in Automatic Speech Recognition*. PhD thesis, Cambridge University, 2001.

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

[90] M. Ostendorf. Moving beyond the 'beads-on-the-string' model of speech. In *Proceeding ASRU*, volume 1, pages 79–84, Colorado, USA, December 1999.

[91] M. Ostendorf, V.V. Digalakis, and O.A. Kimball. From HMMs to segment models: A unified view of stochastic modelling for speech recognition. *IEEE Transactions Speech and Audio Processing*, 4:360–378, 1996.

[92] M. Ostendorf and S. Roukos. A stochastic segment model for phoneme-based continuous continuous speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(12):1857–1869, 1989.

[93] 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, April 1997.

[94] J. Platt. Fast training of support vector machines using sequential minimal optimisation. In B. Scholkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*. MIT Press, 1999.

[95] A.B. Poritz. Linear predictive hidden Markov models and the speech signal. In *Proc. ICASSP*, pages 1291–1294, Paris, France, May 1982.

[96] D. Povey. *Discriminative Training for Large Vocabulary Speech Recognition*. PhD thesis, University of Cambridge, July 2004.

[97] D. Povey and P.C. Woodland. Minimum phone error and I-smoothing for improved discriminative training. In *Proc. ICASSP*, pages I:105–108, Florida, USA, May 2002.

[98] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C++: The Art of Scientific Computing*. Cambridge University Press, 2nd edition, 2002.

[99] A. Quattoni, M. Collins, and T. Darrell. Conditional random fields for object recognition. In *Advances in Neural Information Processing Systems 17*, pages 1097–1104, Cambridge, MA, 2005. MIT Press.

[100] L.A. Rabiner. A tutorial on hidden Markov models and selective applications in speech recognition. In *Proc. of the IEEE*, volume 77, pages 257–286, February 1989.

[101] C.R. Rao. Information and the accuracy attainable in the estimation of statistical parameters. *Bull. Calcutta Math. Soc.*, 37:81–91, 1945.

[102] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organisation in the brain. *Psychological Review*, 65(6):386–408, 1958.

[103] R. Rosenfeld. Two decades of statistical language modelling: where do we go from here? *Proceedings of the IEEE*, 88(8):1270–1278, August 2000.

[104] A-V.I. Rosti. *Linear Gaussian Models for Speech Recognition*. PhD thesis, University of Cambridge, May 2004.

[105] A-V.I. Rosti. Rao-Blackwellised Gibbs sampling for switching linear dynamical systems. In *Proc. ICASSP*, pages I:809–812, Montreal, May 2004.

[106] A-V.I. Rosti and M.J.F. Gales. Factor analysed hidden Markov models. In *Proc. ICASSP*, pages I:949–952, Florida, USA, May 2002.

[107] 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. Available from: mi.eng.cam.ac.uk/∼mjfg.

[108] D.B. Rubin and D.T. Thayer. EM algorithms for ML factor analysis. *Psychometrika*, 47(1):69–76, 1982.

[109] L.K. Saul and M.I. Jordan. Mixed memory Markov models. *Machine Learning*, 37:75–87, 1999.

[110] C. Saunders, J. Shawe-Taylor, and A. Vinokourov. String kernels, Fisher kernels and finite state automata. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 633–640. MIT Press, 2003.

[111] R. Schlüter, W. Macherey, B. Müller, and H. Ney. Comparison of discriminative training criteria and optimisation methods for speech recognition. *Speech Communication*, 34(3):287–310, June 2001.

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

[113] M. Schonlau, W. DuMouchel, W. Ju, A.F. Karr, M. Theus, and Y. Vardi. Computer intrusion: Detecting masquerades. *Statistical Science*, 16(1):59–74, February 2002.

[114] F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proceedings of Human Language Technology, HLT/NAACL-03*, 2003.

[115] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[116] J.R. Shewchuk. An introduction to the conjugate gradient method without the agonising pain. Available from http://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.ps, August 1994.

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

[118] N.D. Smith and M.J.F. Gales. Speech recognition using SVMs. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 1197–1204. MIT Press, 2002.

[119] 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.

[120] J. Stewart. *Calculus: Early Transcendentals*. Brooks/Cole Publishing, 4th edition, 1999.

[121] P.T. Strait. *A First Course in Probability and Statistics with Applications*. Harcourt Brace Jovanovich, 2nd edition, 1989.

[122] M.E. Tipping. Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, pages 211–244, 2001.

[123] K. Tsuda, T. Kin, and K. Asai. Marginalised kernels for biological sequences. *Bioinformatics*, 18:S268–S275, 2002.

[124] L.F. Uebel and P.C. Woodland. An investigation into vocal tract length normalisation. In *Eurospeech*, volume 6, pages 2527–2530, Budapest, Hungary, September 1999.

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

[126] V. Venkataramani, S. Chakrabartty, and W. Byrne. Support vector machines for segmental minimum Bayes risk decoding of continuous speech. In *ASRU 2003*, pages 13–18, 2003.

[127] A. Vinokourov and M. Girolami. A probabilistic framework for the hierarchic organisation and classification of document collections. *Journal of Intelligent Information Systems*, 18(2–3):153–172, 2002. Special Issue on Automated Text Categorisation.

[128] A.J. Viterbi. Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, April 1967.

[129] H. Wallach. *Efficient Training of Conditional Random Fields*. M.Sc. thesis, Division of Informatics, University of Edinburgh, 2002.

[130] S.B. Wang, A. Quattoni, L.-P. Morency, D. Demirdjian, and T. Darrell. Hidden conditional random fields for gesture recognition. In *Proc. CVPR*, 2006.

[131] B.-C. Wei. *Exponential family nonlinear models*. Springer-Verlag, 1998. (Lecture notes in statistics, Volume 130).

[132] C.J. Wellekens. Explicit time correlation in hidden Markov models for speech recognition. In *Proc. ICASSP*, pages 384–386, Dallas, USA, April 1987.

[133] J. Weston and C. Watkins. Multi-class support vector machines. Technical Report CSD-TR-98-04, Royal Holloway, University of London, May 1998.

[134] P.C. Woodland. Speaker adaptation for continuous density HMMs: A review. In *Proc. ISCA ITR-Workshop on Adaptation Methods for Speech Recognition*, 2001.

[135] P.C. Woodland and D. Povey. Large scale discriminative training for speech recognition. In *Proc. ISCA ITRW ASR2000*, pages 7–16, 2000.

[136] P.C. Woodland and D. Povey. Large scale discriminative training of hidden Markov models for speech recognition. *Computer Speech and Language*, 16(1):25–47, 2002.

[137] S. Young. Large vocabulary continuous speech recognition: A review. In *IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 3–28, December 1995.

[138] S. Young, G. Evermann, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland. *The HTK Book (for HTK Version 3.2)*. Cambridge University, 2002.

[139] S.J. Young, J.J. Odell, and P.C. Woodland. Tree-based state tying for high accuracy acoustic modelling. In *ARPA Human Language Technology Workshop*, pages 307–312, Plainsboro, USA, March 1994.

[140] S.J. Young, N.H. Russell, and J.H.S. Thornton. Token passing: a simple conceptual model for connected speech recognition systems. Technical Report CUED/F-INFENG/TR.38, Cambridge University Engineering Department, July 1989.