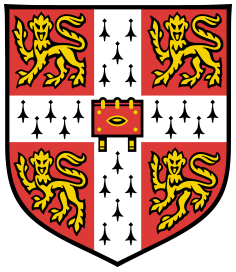
Structured Support Vector Machines for Speech Recognition

SHI-XIONG ZHANG

DARWIN COLLEGE



Department of Engineering

University of Cambridge

This dissertation is submitted for the degree of Doctor of Philosophy

March 2014

Structured Support Vector Machines for Speech Recognition

Shi-Xiong ZHANG

Abstract

Discriminative training criteria and discriminative models are two effective improvements for HMM-based speech recognition. This thesis proposed a structured support vector machine (SSVM) framework suitable for medium to large vocabulary continuous speech recognition. An important aspect of structured SVMs is the form of features. Several previously proposed features in the field are summarized in this framework. Since some of these features can be extracted based on generative models, this provides an elegant way of combine generative and discriminative models. To apply the structured SVMs to continuous speech recognition, a number of issues need to be addressed. First, features require a segmentation to be specified. To incorporate the optimal segmentation into the training process, the training algorithm is modified making use of the concave-convex optimisation procedure. A Viterbi-style algorithm is described for inferring the optimal segmentation based on discriminative parameters. Second, structured SVMs can be viewed as large margin log linear models using a zero mean Gaussian prior of the discriminative parameter. However this form of prior is not appropriate for all features. An extended training algorithm is proposed that allows general Gaussian priors to be incorporated into the large margin criterion. Third, to speed up the training process, strategies of parameter tying, 1-slack optimisation, caching competing hypotheses, lattice constrained search and parallelization, are also described. Finally, to avoid explicitly computing in the high dimensional feature space and to achieve the nonlinear decision boundaries, kernel based training and decoding algorithms are also proposed. The performance of structured SVMs is evaluated on small and medium to large speech recognition tasks: AURORA 2 and 4.

Declaration

This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration except where specifically indicated in the text. It has not been submitted in whole or part for a degree at any other university.

Most of the work in this thesis has been published, as technical reports (van Dalen *et al.* 2012; Zhang and Gales 2010), in conference proceedings (Knill *et al.* 2013; Zhang and Gales 2011*a*;*b*; 2013*a*), and as journal articles (Zhang and Gales 2013*b*; Zhang *et al.* 2010).

The length of this thesis including appendices, bibliography, footnotes, tables and equations is 42 590 words. It contains 58 figures and 22 tables.

Acknowledgements

I would like to express sincere gratitude to various bodies from the University of Cambridge, where I have the opportunity to study with. My major debt is to my supervisor, Prof. Mark Gales, for his mentorship and friendship over these years. Looking back, his wise and guidance over these four years transformed me from an inexperienced students to an independent researcher ready to tackle problems on his own. His taste and choice of problems, discipline and commitment to hard work, all had a great influence on me. I admire his vast knowledge in many areas, e.g., noise robust speech recognition, LVCSR, speech synthesis, machine learning, coding and presentation skills. His encouragement during my ups and downs, his advice to me to solve problems, all made my PhD journey much smoother and at the same time much more exciting.

Special thanks go to the people who have funded me to do a Ph.D. at Cambridge University and to attend many international conferences and workshops. Toshiba Research Europe Ltd. partly funded my Ph.D., and after specifying the subject area—structured discriminative models, did not constrain me at all. I have much appreciated that. I would like to thank the BABEL program (IARPA funded) for providing financial support for my fourth-year study. This provided me an excellent opportunity to be involved in high-standard research and building state-of-the-art speech recognition systems. I must also thank Prof. Mark Gales and Prof. Phil Woodland for helping me in the project. As the principal and co- investigators, they successfully leads the team in the competitive evaluations of the program. Their insight, experience and wide-range knowledge on speech and language processing have also benefited me a lot. It was a privilege to study with them.

I owe my thanks to my colleagues in the Machine Intelligence Lab for the help and encouragements they have given to me. Particular thanks must go to Dr. Anton Ragni, Dr. Rogier van Dalen, Dr. Kate Knill, Eric Wang, Justin Yang, Chao Zhang, Jeff Chen, Dr. Andrew Liu, Dr. Pierre Lanchantin and Dr. Penny Karanasou for scintillating discussions, whether it be speech recognition, machine learning, or subjects less directly related to our research. All have enriched my time here. I also would like to thank Patrick Gosling and Anna Langley for their excellent work in maintaining the computing facilities. These intelligent and kind people make the group an interesting place to work in. The friendship I made in Cambridge will be a treasure forever.

Finally, the biggest thanks go to my parents to whom I always owe everything! For many years, they have offered everything possible to support me. This thesis is dedicated to them.

Contents

List of Figures

Notation

Matrixes and vectors

s	a scalar is denoted by a plain lowercase letter
$\boldsymbol{v}$	a vector is denoted by a bold lowercase letter
$\mathbf{A}$	a matrix is denoted by a bold uppercase letter
$\{\cdot\}^\mathsf{T}$	transpose of a vector or a matrix
$\{\cdot\}^{-1}$	inverse of a square matrix
$\|\cdot\|$	Norm of a vector or a matrix
$\mathrm{diag}(\cdot)$	Matrix diagonalisation
$\mathrm{tr}(\cdot)$	trace of a square matrix
$\langle\cdot,\cdot\rangle$	Inner product of two vectors
$\otimes$	Kronecker tensor product of two vectors
$a_{ij} = [\mathbf{A}]_{ij}$	Element (i, j) of $\mathbf{A}$
$\mathbf{I}$	Identity matrix
$\mathbf{0}$	Vector/matrix with all entries 0
$\mathbf{1}$	Vector/matrix with all entries 1
$\mathbf{exp}(\cdot)\,, \mathbf{log}(\cdot)\,, \circ$	Element-wise exponentiation, logarithm, multiplication

Distributions

$p(\cdot)$	probability density function
$p(\cdot\|\cdot)$	conditional probability density
$P(\cdot)$	probability mass distribution
$P(\cdot\|\cdot)$	conditional probability mass distribution
$\mathcal{E}\{\cdot\}$	Expected value
$\mathrm{Var}\{\cdot\}$	Variance
$\mathcal{KL}(p\|q)$	Kullback-Leibler divergence to p from q
$\mathcal{H}(p\|q)$	Cross-entropy of p and q
$\delta(\cdot)$	Kronecker delta function: evaluates to 1 if the argument is zero and to 0 otherwise.
$\mathbf{a} \sim \mathcal{N}(\boldsymbol{\mu}_a, \boldsymbol{\Sigma}_a)$	$\mathbf{a}$ is Gaussian-distributed with mean $\boldsymbol{\mu}_a$ and covariance $\boldsymbol{\Sigma}_a$
$\mathcal{N}(\mathbf{a};\, \boldsymbol{\mu}_a,\, \boldsymbol{\Sigma}_a)$	Gaussian density evaluated at $\mathbf{a}$

Speech data

$\boldsymbol{o}_t$	observation column vector at time t
$\mathbf{O}_{1:T}$	observation sequence $\mathbf{O}_{1:T} = \{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_T\}$
w	isolated word or sub-word
$\mathbf{w}$	word sequence $\mathbf{w} = \{w_1, \ldots, w_{\lvert\mathbf{w}\rvert}\}$
$\boldsymbol{\theta}$	alignment that splits the observation sequence $\mathbf{O}_{1:T}$ into segments, $\mathbf{O}_{1:T} = \{\mathbf{O}_{t(w_1,\boldsymbol{\theta})}, \ldots, \mathbf{O}_{t(w_{\lvert\mathbf{w}\rvert},\boldsymbol{\theta})}\}$
$\boldsymbol{\phi}(\mathbf{O}, \mathbf{w})$	joint feature space for the utterance $(\mathbf{O}, \mathbf{w})$
$\boldsymbol{\psi}(\mathbf{O}_{i\lvert\boldsymbol{\theta}})$	feature space for the i-th segment $\mathbf{O}_{i\lvert\boldsymbol{\theta}}$
$\mathbf{n}$	additive noise
$\mathbf{h}$	convolutional noise
$\mathbb{L}$	lattices

Speech model

$\boldsymbol{\lambda}$	parameter set of HMMs
$\boldsymbol{\alpha}$	parameter set of discriminative models
$\mathcal{Z}$	normalisation term
$\boldsymbol{s}$	state sequence, $\boldsymbol{s} = [s_1, \cdots, s_T]$, s_t is the state at time t
a_{ij}	discrete state transition probability from state i to j
$b_j(\mathbf{o})$	state output distribution given state j
m	index of a distinct Gaussian component
$\boldsymbol{\mu}^{(m)}$	Mean of component m
$\boldsymbol{\Sigma}^{(m)}$	Covariance matrix of component m
$\boldsymbol{\Lambda}^{(m)}$	Precision matrix of component m
$\gamma_t^{(m)}$	Occupancy for component m at time t
$\pi_m^{(\theta)}$	Weight of component m in the mixture for θ
$\mathbf{A}$	Linear transformation

Optimization

$\mathcal{F}(\cdot)$	training criterion. e.g., $\mathcal{F}_{\texttt{cml}}, \mathcal{F}_{\texttt{mbr}}, \mathcal{F}_{\texttt{lm}}$ represent the conditional maximum likelihood, minimum bayes's risk and large margin criterion, respectively.
$\arg\max_x \mathcal{F}(x)$	the value of x that maximises $\mathcal{F}(x)$
$\mathcal{L}(\cdot)$	loss function
$\mathcal{Q}(\cdot;\cdot)$	auxiliary function of the parameters to estimate given the current estimates
$\boldsymbol{\alpha}^{\texttt{dual}}$	dual variables
ξ	slack variables
$k(\mathbf{O}^{(i)}, \mathbf{O}^{(j)})$	kernel function for $\mathbf{O}^{(i)}$ and $\mathbf{O}^{(j)}$
$K\left((\mathbf{O}^{(i)}, \mathbf{w}^{(i)}), (\mathbf{O}^{(j)}, \mathbf{w}^{(j)})\right)$	joint kernel function for $(\mathbf{O}^{(i)}, \mathbf{w}^{(i)})$ and $(\mathbf{O}^{(j)}, \mathbf{w}^{(j)})$
$\mathbf{G}$	Gram Matrix (Kernel Matrix)

Indices	
t	Time
τ	Iteration
l	Window size
r	Training utterance
d	Feature vector length
e	Node in a lattice $\mathbb{L}$

Chapter 1

Introduction

1.1 Structured Data

One goal of supervised machine learning is to develop mathematic models that are able to learn and generalize from previously observed data, $(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_n, \boldsymbol{y}_n)$, where $\boldsymbol{x}$ is the observation sequence and $\boldsymbol{y}$ is the corresponding class. In many applications, for example face recognition, speaker verification and isolated word recognition, machine learning has been involved with predicting a class label from observations, as in the case of *classification*; or predicting a scalar value, as in the case of *regression* (Bishop 2006). In these cases, the class $\boldsymbol{y}$ has been mainly considered as a single, atomic (unstructured) label y.[1] These $(\boldsymbol{x}, y)$ are referred as *unstructured* data (Bakir *et al.* 2007).

However, in some other tasks for example natural language parsing and continuous speech recognition, the output class $\boldsymbol{y}$ is not a single label but a *structured* object, such as a tree or a word sequence with some grammars. There are dependencies between these classes $\boldsymbol{y}$. Classification for *structured* data $(\boldsymbol{x}, \boldsymbol{y})$ is far from trivial (Lacoste-Julien 2010; Rabiner 1989; Taskar *et al.* 2005). One reason is that the space of all possible classes, $\boldsymbol{y}$, is enormous, usually exponential in the number of individual

[1] In this work, unstructured labels are denoted by unbold symbols, e.g., y and structured labels are denoted by bold symbols, e.g., $\boldsymbol{y}$.

unstructured y. For example, in isolated digit recognition, the space of, unstructured, output y is only 10. However, in continuous digit recognition, for a 6-digit length utterance, the space of all possible $\boldsymbol{y}$ is 10^6.

1.2 Classification Models

Statistical models for classification commonly fall into one of two broad categories: *generative* and *discriminative* models.

- The generative models, e.g., Hidden Markov Models (HMMs), approximate the joint distribution, $p(\boldsymbol{x}, \boldsymbol{y})$, over observation sequences and classes. In many cases it is usually convenient to decompose the joint probability into a prior probability $P(\boldsymbol{y})$ and a likelihood probability $p(\boldsymbol{x}|\boldsymbol{y})$ (Bishop 2006). A classification decision can be made by computing posterior probabilities based on Bayes' rule, $P(\boldsymbol{y}|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\boldsymbol{y})P(\boldsymbol{y})}{p(\boldsymbol{x})}$. The parameters of the generative model were originally estimated by using the maximum likelihood criterion (Rabiner 1989). Generative models can also be trained using discriminative criteria, e.g., maximum mutual information (MMI) (Gauvain and Lee 1994), minimum Bayes risk (MBR) (Na *et al.* 1995), minimum classification error (MCE) (Juang and Katagiri 1992) and maximum margin (MM) training (Sha and Saul 2007).

- By contrast, discriminative models, directly model the mapping from observation sequences to label sequences, either as a posterior distribution $P(\boldsymbol{y}|\boldsymbol{x})$ or as a discriminant function $f_{\boldsymbol{y}}(\boldsymbol{x})$. Discriminative models therefore avoid the need to maintain valid likelihood and prior distributions (Minka 2005). For example, logistic regression machines (Ng and Jordan 2001) and Conditional Random Fields (CRF) (Lafferty *et al.* 2001) directly model the posterior of the label sequence given the observations. Support vector machines (SVMs) (Vapnik 1995) directly models the discriminant function (or decision boundaries) between classes. The parameters of these discriminative models are typically estimated by optimizing various objectives related to the classification loss,

such as conditional maximum likelihood (CML) (Heigold 2010), MCE (Smith and Eisner 2006), MBR (Venkataramani *et al.* 2003) or a maximum margin criterion (Vapnik 1995; Zhang *et al.* 2010).

Discriminative approaches usually show better performance when there is sufficient training data, as they are better "tuned" to the classification task (Jebara 2001). On the other hand, generative modeling approaches provide a more natural way to incorporate complex structural information about the data. To handle structured data (with enormous numbers of classes), the parameters of generative/discriminative models are typically broken down into a common set of basic structure units. These shared basic parameters can then be combined together in different sequences to model all possible classes. In this work, these generative and discriminative models are referred as *structured models*. Several commonly used unstructured and structured models are summarized in Table 1.1.

1.3 Organisation of Thesis

Continuous speech recognition (CSR), also known as speech to text transcription, systems are typically trained using a large (comparing to many machine learning tasks) amount of training data, millions of words of language model training data and millions of frames of acoustic model training data (Gales *et al.* 2006). This thesis views speech recognition as a structured classification problem (Taskar 2005; Zhang *et al.* 2010) in which class labels (sentences) have meaningful internal structure (e.g., words). Thus, although the number of possible class labels may be unlimited, these class labels can be decomposed into a common set of basic structures, e.g., words/phones. Deriving an appropriate set of basic structures is often as important as modelling speech-text dependencies (Odell 1995; Ragni 2013).

Chapter 2 discusses generative approaches to speech recognition. Most speech recognition systems are based on structured generative models, employing hidden Markov models (HMM), as the acoustic models. Likelihoods from these phone-based

HMMs are combined together with the prior, usually an n-gram language model, to yield the sentence posterior based on Bayes' rule (Gales and Young 2007). This enables posteriors of all possible sentences to be captured. Although discriminative training of HMMs has been shown to yield performance gains (Byrne 2006; Juang and Katagiri 1992; Keshet *et al.* 2011; Sha and Saul 2007; Woodland and Povey 2002), the underlying acoustic models are still generative, with the standard HMM conditional independence assumptions and the form of posteriors may be restricted by Bayes' rule. This has led to interest in discriminative models for speech recognition in Chapter 3 and 4.

For discriminative models three important decisions need to be made: the form of the features to use; the appropriate training criterion; and how to handle structure for continuous speech. Chapter 3 introduces *unstructured* discriminative models, e.g., logistic regression models (Birkenes *et al.* 2006) and SVMs (Venkataramani *et al.* 2003) for isolated word recognition. Chapter 4 discusses *structured* discriminative models, e.g., conditional random fields (CRFs) (Abdel-Haleem 2006), hidden CRFs (HCRFs) (Gunawardana *et al.* 2005) and segmental CRFs (SCRF) (Zweig and Nguyen 2009), where the sentence posterior given the observation are directly modelled. Many common discriminative training criteria are considered.

A central aspect of discriminative models is the set of features extracted from the observation and hypothesized word sequences. Features for discriminative models are summarized in Chapter 5. An overview of feature functions proposed for handling variable-length sequence is given. A number of features at the frame, model and word level (Gunawardana *et al.* 2005; Morris and Fosler-Lussier 2008; Ragni and Gales 2011*b*; Zhang *et al.* 2010) are summarized in an correlated format. In particular, features based on generative models are discussed. They are an attractive option as they allow state-of-the-art speaker adaptation and noise robustness approaches for generative models to be used (Gales and Flego 2010).

Chapter 6 introduces structured SVMs for speech recognition which is the focus of this thesis. The relationship between several commonly used unstructured and structured models for speech recognition are summarized in Table 1.1. The motivation

of using structured SVMs for speech recognition can be seen from two perspectives in this table.

Training	**Unstructured** y	$\longrightarrow$	**Structured** $\boldsymbol{y}$
Max Likelihood	Naive Bayes $p(\boldsymbol{x}\|y)$	$\longrightarrow$	HMM $p(\boldsymbol{x}\|\boldsymbol{y})$
$\downarrow$	$\downarrow$		$\downarrow$
Max Conditional Likelihood	Logistic regression $P(y\|\boldsymbol{x})$	$\longrightarrow$	CRF $P(\boldsymbol{y}\|\boldsymbol{x})$
$\downarrow$	$\downarrow$		$\downarrow$
Max Margin	SVM $\boldsymbol{\alpha}_y^\mathsf{T}\boldsymbol{\phi}(\boldsymbol{x})$	$\longrightarrow$	**?**

Table 1.1 *Summary of commonly used models for sequence classification, where $\boldsymbol{x}$ is the observation sequence and $\boldsymbol{y}$ (or y) is the corresponding label sequence (or an unstructured label). Here CRFs means linear chain CRFs. The gray row represents generative models and the yellow rows represent discriminative models. Note that the logistic regression and CRF can be related to the discriminatively trained naive bayes and HMMs (Heigold* et al. *2007; Roos* et al. *2005), respectively; SVMs can be related to the maximum margin trained logistic regression models (for details see Section 3.2.3). The question mark is the focus of this thesis (It is discussed and related to the SVMs and CRFs in Chapter 6).*

First, from the perspective of structured extension. Unstructured discriminative models, e.g. logistic regression models and SVMs, assume that the class labels have no structure. When applying these models to a complete utterance in continuous speech recognition, the space of possible classes becomes very large, e.g., a 6-digit length utterance yields 10^6 classes. One solution to deal with this is to segment the continuous speech into words/sub-words observation sequences (Venkataramani *et al.* 2003). For each segment, SVMs or logistic regression models can be applied in the same fashion as in isolated classification tasks (Birkenes *et al.* 2006; Gales and Flego 2010; Zhang *et al.* 2010). However, there are two problems with this approach. First, the classification is based on one, fixed, segmentation. Second, each segment is treated independently. An alternative solution is to incorporate structure into the model. This transforms the unstructured discriminative model into a structured model. For naive Bayesian classifiers, this structured extension can lead to HMMs. For logistic regressions,

this can lead to CRFs. For SVMs, this yields structured SVMs which was originally proposed by (Joachims *et al.* 2009) in the machine learning field.

Second, from the perspective of maximum margin training. An appropriate training criterion is very important for speech recognition (Schlüter *et al.* 2001). Maximum margin learning has been widely studied and applied by machine learning researchers in many application domains and often has achieved superior results (Keshet and Bengio 2008; Roark 2009; Sha and Saul 2007; Smola 2000). Interestingly, it was shown by (Zhang *et al.* 2003) that the SVMs can be related to logistic regression models trained by the maximum margin criteria. Table 1.1 illustrates that structured SVMs may also related to CRFs with maximum margin learning. This maximum margin learning of discriminative models may offer new opportunities to improve the performance of state-of-the-art speech recognition systems.

The major contribution of this thesis is we complete the Table 1.1 by structured SVM and extend its training and inference algorithm for speech recognition. We show that the proposed structured SVMs can be related to many existing models in the speech field. A brief chapter-by-chapter breakdown is given as follows.

Chapter 2 provides an overview of generative approaches to speech recognition. In particular, it describes state-of-the-art HMM-based systems.

Chapter 3 describes unstructured discriminative models for isolated word recognition. To apply these models to continuous speech recognition, an acoustic code-breaking scheme is also introduced.

Chapter 4 discusses structured discriminative models for continuous speech recognition. Various forms of discriminative training criteria are also discussed.

Chapter 5 provides an overview of features for discriminative models. The features can map the variable-length sequences into a fixed-dimension vector.

Chapter 6 describes structured SVMs for continuous speech recognition and implementation issues.

Chapter 7 describes kernel methods for structured SVMs that can avoid computing the high-dimension feature space explicitly.

Chapter 8 provides the experimental setup and results on two speech recognition tasks where vocabulary ranges from small to medium-to-large.

Chapter 9 concludes with a summary of the thesis and outlines possible directions for future work.

Chapter 2

Generative Models

As was discussed in Chapter 1, Statistical classification approaches are often categorised into two broad groups *generative* models and *discriminative* models. Discussion in this chapter focuses on the generative models.

Generative models are one of the most important forms of statistical model for classifying sequence data such as speech. Generative models model the probability density function associated with the observations, enabling observation sequences to be randomly generated from distributions of the model. For example in speech recognition, given an observation sequence, $\mathbf{O} = \{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_T\}$, and a word sequence (class), $\mathbf{w} = \{w_1, \ldots, w_L\}$, generative models allow the sequence likelihood, $p_{\boldsymbol{\lambda}}(\mathbf{O}|\mathbf{w})$, to be calculated,[1] where $\boldsymbol{\lambda}$ is model parameters. Given a prior distribution over the class, $P(\mathbf{w})$, Bayes' rule can be used to convert the likelihood into a class posterior (Bishop 2006; Duda *et al.* 2001),

$$P_{\boldsymbol{\lambda}}(\mathbf{w}|\mathbf{O}) = \frac{p_{\boldsymbol{\lambda}}(\mathbf{O}|\mathbf{w})P(\mathbf{w})}{p(\mathbf{O})} \tag{2.1}$$

where $p(\mathbf{O})$ is the class-independent probability of $\mathbf{O}$, known as the *evidence* (Bishop 2006). It is typically calculated by marginalising over all classes,

$$p(\mathbf{O}) = \sum_{\mathbf{w} \in \mathcal{W}} p_{\boldsymbol{\lambda}}(\mathbf{O}|\mathbf{w})P(\mathbf{w}) \tag{2.2}$$

[1] Strictly generative classifiers use models of the joint distribution, typically of the form $p_{\boldsymbol{\lambda}}(\mathbf{O}, \mathbf{w}) = p_{\boldsymbol{\lambda}}(\mathbf{O}|\mathbf{w})P(\mathbf{w})$.

where $\mathcal{W}$ is the set of all classes. A class $\hat{\mathbf{w}} \in \mathcal{W}$ can then be assigned to unlabelled $\mathbf{O}$ using Bayes' decision rule (Bishop 2006),

$$\begin{aligned}\hat{\mathbf{w}} &= \arg\max_{\mathbf{w}\in\mathcal{W}} P_{\boldsymbol{\lambda}}(\mathbf{w}|\mathbf{O}) \\ &= \arg\max_{\mathbf{w}\in\mathcal{W}} p_{\boldsymbol{\lambda}}(\mathbf{O}|\mathbf{w})P(\mathbf{w}) \end{aligned} \tag{2.3}$$

The evidence, $p(\mathbf{O})$, is omitted in equation 2.3 since it does not depend on the $\mathbf{w}$.

As discussed in Chapter 1, generative models can be divided into unstructured and structured approaches. The typical example of *unstructured* generative models is the naive Bayes, where the likelihood $p_{\boldsymbol{\lambda}}(\mathbf{O}|w)$ given a single unstructured label w is modelled. One famous form of *structured* generative model is the HMM, where the likelihood $p_{\boldsymbol{\lambda}}(\mathbf{O}|\mathbf{w})$ given a label sequence $\mathbf{w}$ can be modelled. The details of naive Bayes and HMMs are described in the following sections.

2.1 Naive Bayes

The naive Bayes model is the simplest example of an unstructured generative model. It assumes that there is no structure in the class (i.e., $\mathbf{w} \rightarrow w$) and each feature $\boldsymbol{o}_i$ is conditionally independent of every other feature $\boldsymbol{o}_j, \forall\, i \neq j$. Thus the likelihood of observation sequence can be calculated as

$$p_{\boldsymbol{\lambda}}(\mathbf{O}|w) = \prod_{t=1}^{\mathsf{T}} p_{\boldsymbol{\lambda}}(\boldsymbol{o}_t|w) \tag{2.4}$$

The graphical representation of naive Bayesian model is shown in Figure 2.1. The naive Bayes assumption dramatically reduces the number of parameters to be estimated when modeling likelihoods (Friedman *et al.* 1997). These parameters (i.e., observation probability distributions) can be estimated using the maximum likelihood criterion (Bishop 2006). It has been shown that the classification function of naive Bayes model is a linear function when the observation is discrete (McCallum *et al.* 1998). Note that there are no hidden states and parameter sharing (between classes) in these models. Although naive Bayes works well in practice for many classification tasks (McCallum

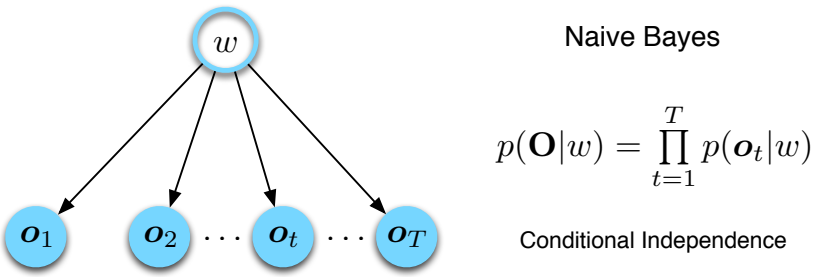


Figure 2.1 *The structure of naive Bayesian model (Friedman* et al. *1997).*

et al. 1998; Tóth *et al.* 2005; Ye *et al.* 2002), it is not easy to apply this model for recognizing structured data. This leads to interest in structured generative models.

2.2 Hidden Markov Models

Structured generative models consider the structures in the classes $\mathbf{w}$ when modelling likelihoods $p_{\boldsymbol{\lambda}}(\mathbf{O}|\mathbf{w})$. However this is normally a difficult problem to tackle directly since the numbers of classes is enormous when $\mathbf{w}$ is a structured object. The key idea to handle this is to break down the parameters of structured models, for all observed classes (e.g., sentences) in the training data, into a common set of basic structure units (e.g., words/subwords). These shared basic parameters can then be combined together in different sequences to model all possible classes in the test data. Latent variables $\boldsymbol{\theta}$ are normally introduced in these models to allow the data $(\mathbf{O}, \mathbf{w})$ to be segmented into the structure units.

The Hidden Markov Model (HMM) (Rabiner 1989) is one example of structured generative models. It was very widely applied to the tasks of speech recognition (Baker 1975; Gales and Young 2007). The Hidden Markov Model is a finite state machine composed of a fixed number of discrete latent states including non-emitting initial and accept latent states. The HMM starts in the initial state at time $t = 0$. At each subsequent time instance t the HMM transitions into a new state $\theta_t = j$ with transition probability a_{ij} from state $\theta_{t-1} = i$. An observation is then generated based

on the output distribution on current state, $b_j(\boldsymbol{o}_t) = p(\boldsymbol{o}_t|\theta_t)$.[2] In HMMs, only the observation sequence, $\{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_T\}$ is observable. The corresponding state sequence $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_T\}$ is unobserved. The underlining assumption for a HMM are:

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

Neither of these two assumptions is true for speech. Much research has been carried out to compensate the effect of the poor assumptions or to find alternative models for speech. However, the standard HMM is still a successful acoustic modelling technique and is widely used in many speech recognition systems. Figure 2.2 shows the topology and directed graphical model (Bilmes 2003) associated with an example HMM. The left diagram illustrates a strict left-to-right topology with three emitting states for a phone HMM in speech recognition. The right diagram illustrates the directed graphical model associated with HMMs for a word sequence.

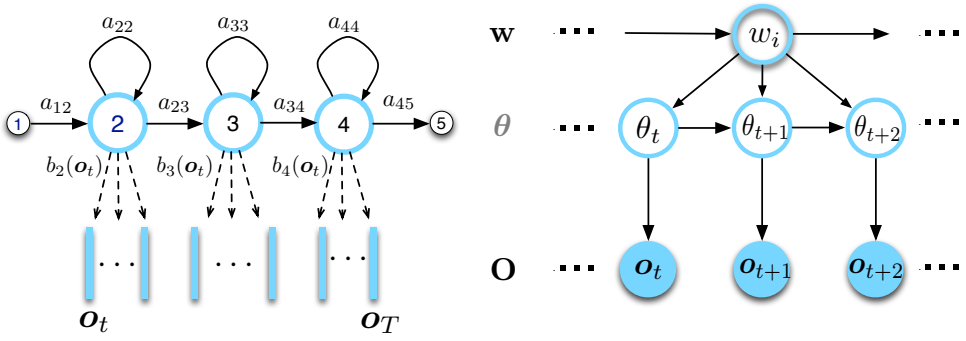


Figure 2.2 *Hidden Markov models. (a) An example of left-to-right topology with three emitting states. (b) A directed graphical model for HMMs with a specific state sequence $\boldsymbol{\theta}$, where $\mathbf{w}$ is a word sequence and w_i is a word/subword.*

[2] The output distribution, $b_j(\boldsymbol{o}_t)$, can also be a discrete probability, $P(\boldsymbol{o}_t|\theta_t)$.

The parameter set, $\boldsymbol{\lambda} = \{\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B}\}$, associated with a HMM is:

- $\boldsymbol{\pi} = \{\pi_i\}$ — *Initial state distribution*

  The initial state distribution of state i is expressed as

  $$\pi_i = P(\theta_1 = i), \ \sum_{i=1}^{N} \pi_i = 1 \tag{2.5}$$

  where N is the total number of states. From equation (2.5), by introducing a non-emitting entry state and having a standard left-to-right topology, the initial state distribution of the first emitting state is always 1.

- $\boldsymbol{A} = \{a_{i,j}\}$ — State *transition probability* matrix

  Letting θ_t denote the state at time t, the element of state transition probability matrix $\boldsymbol{A}$ is defined as

  $$a_{ij} = P(\theta_t = j | \theta_{t-1} = i), \ \sum_{j=1}^{N} a_{ij} = 1. \tag{2.6}$$

  As the HMMs used in speech recognition are normally constrained to be left-to-right, the matrix contains zeros.

- $\boldsymbol{B} = \{b_j(\cdot)\}$ — *State output probability* distributions

  Each emitting state j is associated with one probability distribution which generates an observation at each time instance.

  $$b_j(\boldsymbol{o}_t) = p(\boldsymbol{o}_t | \theta_t = j) \tag{2.7}$$

  There are two forms of state output distribution that are usually adopted in speech recognition. One is to compute $b_j(\boldsymbol{o}_t)$ using *Gaussian Mixture Models* (GMMs). The resulting model is usually referred as GMM-HMMs. Alternatively, $b_j(\boldsymbol{o}_t)$ can adopt the probability density function derived from *Deep Neural Networks* (DNNs). The resulting framework is known as DNN-HMM hybrid systems (Bourlard and Morgan 1998; Seide *et al.* 2011). The details of these forms will be discussed in Sections 2.2.1.2 and 2.2.1.3.

The use of HMMs in practical applications requires solutions to the following three standard problems (Rabiner 1989):

- **Likelihood Caculation**: Given the observation sequence $\mathbf{O}$, the corresponding label sequence $\mathbf{w}$ and HMM parameters $\boldsymbol{\lambda}$, how to compute the likelihood $p_{\boldsymbol{\lambda}}(\mathbf{O}|\mathbf{w})$ efficiently?

- **Infering the state sequence**[3]: Given the observation sequence $\mathbf{O}$ and HMM parameters $\boldsymbol{\lambda}$, how to find the "most likely" state sequence $\boldsymbol{\theta}$?

- **Parameter estimation**: How to estimate HMM parameters $\boldsymbol{\lambda}$?

Solutions to these problems are considered in Sections 2.2.1, 2.2.2 and 2.3.

2.2.1 *Likelihood Calculation*

Likelihood calculation is a basic issue to be addressed when using HMMs. As states $\boldsymbol{\theta}$ are hidden, the probability density of observations $\mathbf{O}$ given $\mathbf{w}$ is computed by marginalising over all possible latent state sequences

$$\begin{aligned} p_{\boldsymbol{\lambda}}(\mathbf{O}|\mathbf{w}) &= \sum_{\boldsymbol{\theta}} p_{\boldsymbol{\lambda}}(\mathbf{O}, \boldsymbol{\theta}|\mathbf{w}) = \sum_{\boldsymbol{\theta}} P(\boldsymbol{\theta}|\mathbf{w}) p_{\boldsymbol{\lambda}}(\mathbf{O}|\boldsymbol{\theta}; \mathbf{w}) \\ &= \sum_{\boldsymbol{\theta} \in \Theta_{\mathbf{w}}^{T}} a_{\theta_0 \theta_1} \prod_{t=1}^{T} a_{\theta_{t-1}\theta_t} b_{\theta_t}(\boldsymbol{o}_t) \end{aligned} \tag{2.8}$$

where $\Theta_{\mathbf{w}}^{T}$ indicates all valid state sequences of $\mathbf{w}$ with length T. Note that even for small numbers of states and observations, the use of direct summation becomes computationally impractical due to a large number of possible state sequences. However, the conditional independence and first order Markov assumptions enable the likelihood $p_{\boldsymbol{\lambda}}(\mathbf{O}|\mathbf{w})$ to be calculated efficiently using dynamic programming approaches, for example the *forward-backward* algorithm (Gales and Young 2007; Rabiner 1989).

[3] Infering the word sequence $\mathbf{w}$ will be discussed in Section 2.4.

2.2.1.1 *Forward-backward algorithm*

Let the *forward probability*, $\alpha_t(j)$, denote the sum of the likelihoods of all partial paths ending in state j at time t. For a N-state HMM, the forward probability for emitting states can be calculated recursively using

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

Note that the first and the last states are "non-emitting" states. The initial and final conditions for the above recursion are

$$\alpha_j(t) = \begin{cases} 1 & j = 1 \quad t = 0 \\ a_{1j} b_j(\boldsymbol{o}_t) & 1 < j < N \quad t = 1 \\ \sum_{i=2}^{N-1} \alpha_i(T) a_{iN} & j = N \quad t = T \end{cases} \tag{2.10}$$

As a result, the likelihood is given by the forward probability of the state N at time T

$$p_{\boldsymbol{\lambda}}(\mathbf{O}|\mathbf{w}) = \alpha_N(T) \tag{2.11}$$

In addition to forward probabilities, the *backward probability* $\beta_i(t)$ is introduced as the conditional probability that the model will generate the rest of the sequence from time t given $\theta_t = i$,

$$\begin{aligned}\beta_i(t) &= p_{\boldsymbol{\lambda}}(\boldsymbol{o}_{t+1}, \ldots, \boldsymbol{o}_T | \theta_t = i) \\ &= \sum_{i=2}^{N-1} a_{ij} b_j(\boldsymbol{o}_{t+1}) \beta_j(t+1).\end{aligned} \tag{2.12}$$

The initial and final conditions are

$$\beta_i(t) = \begin{cases} a_{iN} & 1 < i < N \quad t = T \\ \sum_{i=2}^{N-1} a_{1j} b_j(\boldsymbol{o}_1) \beta_i(1) & i = 1 \quad t = 1 \end{cases} \tag{2.13}$$

Note that differently to forward probabilities computation is performed starting at time $t = T$ and terminating at time $t = 1$. As a results, the likelihood can also be given by the backward probability of the initial state at time $t = 1$.

$$p_{\boldsymbol{\lambda}}(\mathbf{O}|\mathbf{w}) = \beta_1(1)$$

The forward-backward algorithm can be also used to compute a posterior probability $\gamma_j(t)$ of occupying state j at time t

$$\gamma_j(t) = P_{\boldsymbol{\lambda}}(\theta_t = j|\mathbf{O};\mathbf{w}) = \frac{\alpha_j(t)\beta_j(t)}{p_{\boldsymbol{\lambda}}(\mathbf{O}|\mathbf{w})} \tag{2.14}$$

where $\alpha_j(t)$ is the forward probability, $\beta_j(t)$ is the backward probability and $p_{\boldsymbol{\lambda}}(\mathbf{O}|\mathbf{w})$ is the likelihood. The posterior probability $\gamma_j(t)$ plays a important role in estimating HMM parameters (for details see Section 2.3).

Note that both forward and backward probabilities require a specific form of state likelihood probabilities, $b_j(\boldsymbol{o}_t)$. Two forms of $b_j(\boldsymbol{o}_t)$ that are commonly used in speech recognition are described in the following sections.

2.2.1.2 *GMM Likelihoods*

The state output distributions $b_j(\boldsymbol{o}_t)$ usually adopt probability density functions in the form of Gaussian mixture models (GMM)

$$b_j(\boldsymbol{o}_t) = p(\boldsymbol{o}_t|\theta_t = j) = \sum_{m=1}^{M} c_{jm}\mathcal{N}\left(\boldsymbol{o}_t;\boldsymbol{\mu}_{jm},\boldsymbol{\Sigma}_{jm}\right) \tag{2.15}$$

where M is the number of mixture components and c_{jm} is the weight of component m of state j. Each component is a multivariate Gaussian distribution

$$\mathcal{N}\left(\boldsymbol{o};\boldsymbol{\mu},\boldsymbol{\Sigma}\right) = (2\pi)^{-\frac{D}{2}}\left|\boldsymbol{\Sigma}\right|^{-\frac{1}{2}}\exp\left\{-\frac{1}{2}\left(\boldsymbol{o}-\boldsymbol{\mu}\right)^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}\left(\boldsymbol{o}-\boldsymbol{\mu}\right)\right\}. \tag{2.16}$$

where D is the dimension of feature vector, $\boldsymbol{\mu}$ is the mean vector, and the covariance matrix $\boldsymbol{\Sigma}$ is normally assumed to be diagonal. In order to ensure that the state output distributions are valid probability density functions, the mixture component weights must satisfy

$$\forall\, j,\ \forall\, m \quad c_{jm} \geq 0; \qquad \forall\, j \quad \sum_{m=1}^{M} c_{jm} = 1 \tag{2.17}$$

The number of mixture components, M, can be set using simple approaches such as mixture splitting (Young *et al.* 2009) or using more refined approaches such as those described in (Chen and Gopinath 1997; Gales *et al.* 2006; Liu and Gales 2007). Thus, the resulting HMMs are normally called GMM-HMMs (Gales and Young 2007).

2.2.1.3 *DNN "Likelihoods"*

Alternatively, the state output distributions $b_j(\boldsymbol{o}_t)$ can adopt the probability density functions derived from neural networks (or multilayer perceptrons–MLPs) (Bourlard and Morgan 1994; 1998). This approach provides an elegant way of combining neural networks and HMMs. Especially, when those neural networks with "deep architecture" are used, the resulting framework is known as DNN-HMM hybrid systems (Hinton *et al.* 2012) [4]. Promising results based on these hybrid systems have been reported (Hinton *et al.* 2012; Seide *et al.* 2011).

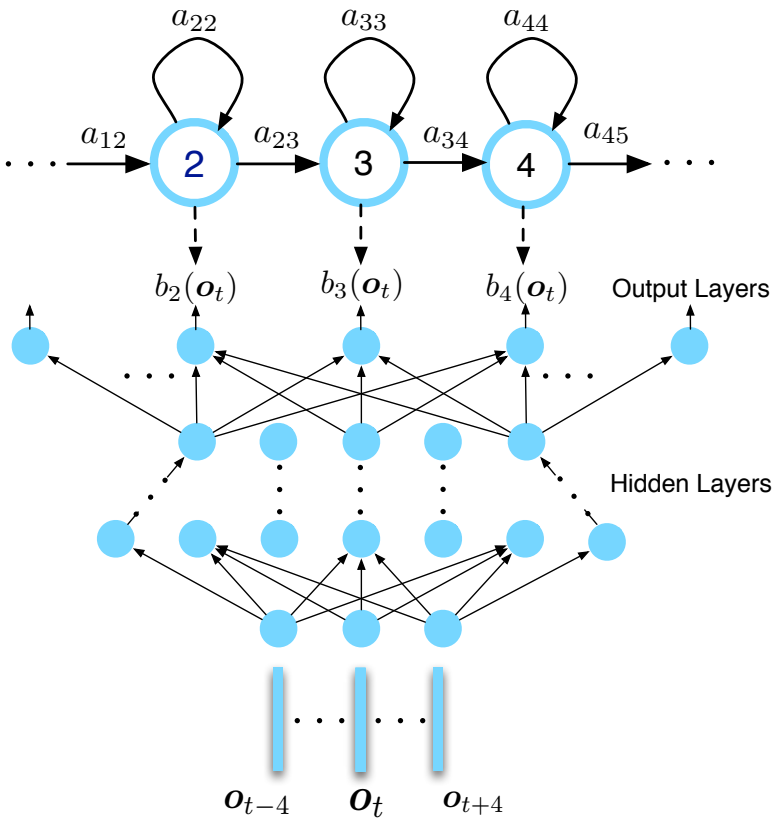


Figure 2.3 *DNN-HMM hybrid architectures.*

[4] There are two ways of incorporating neural networks into HMM systems in literatures of CSR: Hybrid (Bourlard and Morgan 1998) and Tandem systems (Hermansky *et al.* 2000). In contrast to DNN-HMM hybrid systems, Tandem system still belongs to GMM-HMM framework. The neural network is used for extracting the features. These features are described in Section 5.1.1.2.

These neural networks aim to model the posterior probability of each state $p(\theta_t = j|o_t)$ directly. The state output distributions can then be obtained by applying the Bayes' rule,

$$b_j(\boldsymbol{o}_t) = p(\boldsymbol{o}_t|\theta_t = j) = \frac{P(\theta_t = j|\boldsymbol{o}_t)p(\boldsymbol{o}_t)}{P(\theta_t = j)} \tag{2.18}$$

where θ_t is the state label of observation $\boldsymbol{o}_t$, $P(\theta_t = j|o_t)$ is the state posterior probability estimated from the DNN, $P(\theta_t = j)$ is the prior probability of each state estimated from the training set. All of the likelihoods produced in this way are scaled by the same unknown factor of $p(\boldsymbol{o}_t)$, which can be ignored since it is independent of label sequences (Seide *et al.* 2011). Dividing the state posterior probability by prior probability $P(\theta_t = j)$ has been found to be very useful to overcome the label bias problem (Dahl *et al.* 2012), especially in speech recognition when the training utterances contain many long silence segments.

The framework of DNN-HMM is shown in Figure 2.3. The input feature is the concatenation of several consecutive frames, and the state labels are used for training the DNN. The training process can be split into two stages, pre-training and fine-tuning. Pre-training aims to find a good initialization. There are mainly two types of pre-training. The first approach is to use the Restrict Boltzman Machine, stacking layer by layer using greedy unsupervised training (Hinton *et al.* 2006). The other approach is to train a shallow neural network (e.g. only 1 hidden layer) through supervised training first. The hidden layers are added one by one and followed by sweeping the training data with a few epochs, until required number of hidden layers are achieved. The later approach was found converged faster in the fine-tuning stage (Hinton *et al.* 2012). In the fine-tuning stage, the standard error back-propagation algorithm can be applied.

Although good performances are achieved in systems based on DNNs, there are still some challenges for these hybrid systems. For example, the training speech with state labels is a issue for DNNs. It is not easy to parallelize the back propagation with stochastic gradient descent. In addition, the adaptation methods for DNN-HMMs still need to be explored.

2.2.2 *Viterbi Decoding*

Given observation sequence $\mathbf{O}$ and model parameters $\boldsymbol{\lambda}$, the corresponding state sequence $\boldsymbol{\theta}$ is not known. There are several possible criteria that can be used to infer an "optimal" state sequence [5], in some meaningful sense (Rabiner 1989). In speech recognition, the criterion based on maximising the likelihood is most commonly used. The state sequence that satisfies this criterion is called the most likely state sequence. The problem of inferring the most likely state sequence can be expressed as

$$\begin{aligned}\hat{\boldsymbol{\theta}}_{1:T} &= \arg\max_{\boldsymbol{\theta}} p_{\boldsymbol{\lambda}}(\mathbf{O}, \boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \left\{ p_{\boldsymbol{\lambda}}(\mathbf{O}|\boldsymbol{\theta}) p(\boldsymbol{\theta}) \right\} \\ &= \arg\max_{\boldsymbol{\theta}_{1:T}} \left\{ a_{\theta_0,\theta_1} \prod_{t=1}^{T} b_{\theta_t}(\mathbf{O}_t) a_{\theta_t,\theta_{t+1}} \right\} \end{aligned} \tag{2.19}$$

The technique for inferring the most likely state sequence known as *Viterbi algorithm* (Viterbi 1982). To find the most likely state sequence, the Viterbi algorithm introduces the following term,

$$\rho_t^{(j)} = \max_{\boldsymbol{\theta}_{1:t-1}} \left\{ p_{\boldsymbol{\lambda}}(\mathbf{O}_{1:t}, \boldsymbol{\theta}_{1:t-1}, \theta_t = j) \right\} \tag{2.20}$$

which is the maximum likelihood of observing the partial observation sequence $\mathbf{O}_{1:t}$ and then being in state j at time t. The Viterbi algorithm can be visualised in trellises (Rabiner 1989) such as the one given in Figure 2.4.

The Viterbi algorithm computes equation (2.20) recursively based on (Young *et al.* 2009)

$$\rho_t^{(j)} = \max_i \left\{ \rho_{t-1}^{(i)} \cdot a_{i,j} \right\} b_j(\boldsymbol{o}_t) \tag{2.21}$$

with the initial conditions given by (Young *et al.* 2009)

$$\rho_0^{(1)} = 1, \quad \rho_1^{(2)} = a_{12} b_2(\boldsymbol{o}_1), \quad \dots \tag{2.22}$$

where $b_j(\boldsymbol{o}_t)$ may from the GMM likelihood (2.15) or DNN likelihood in (2.18). Upon termination at time $t = T$, the likelihood of $\hat{\boldsymbol{\theta}}$ obtained by (Young *et al.* 2009)

$$p_{\boldsymbol{\lambda}}(\mathbf{O}, \hat{\boldsymbol{\theta}}) = \max_i \left\{ \rho_T^{(i)} \cdot a_{iN} \right\} \tag{2.23}$$

[5] Decoding the word sequence $\mathbf{w}$ for speech recognition will be discussed in Section 2.4.

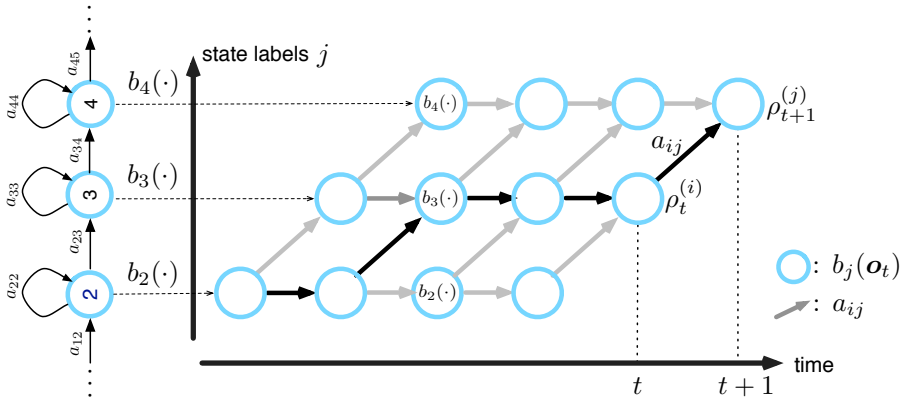


Figure 2.4 *The viterbi decoding of HMMs. Each blue circle represents an emission probability $b_j(\boldsymbol{o}_t)$ for state j at time t. Each arrow represents a state transition probability a_{ij}, and $\rho_t^{(i)}$ is the maximum likelihood of observing the partial observation sequence $\mathbf{O}_{1:t}$ and then being in state i at time t. The black arrows indicate the most likely state sequence which can be retrieved using equation* (2.24).

The most likely state sequence can be retrieved through the following recursion (Rabiner 1989)

$$\hat{\theta}_t = \arg\max_i \left\{ \rho_t^{(i)} a_{i\hat{\theta}_{t+1}} \right\} \tag{2.24}$$

given the optimal state $\hat{\theta}_{t+1}$ from previous stage. The initial state at time T is given by

$$\hat{\theta}_T = \arg\max_i \left\{ \rho_T^{(i)} \cdot a_{iN} \right\} \tag{2.25}$$

The computational complexity of the Viterbi algorithm is $\mathcal{O}(N^2 T)$.

2.3 Parameter Estimation for Generative Models

As discussed in Section 2.2, the use of generative models, e.g. HMMs, in real applications, requires knowing how to estimate the parameters $\hat{\boldsymbol{\lambda}}$. A standard approach is to select values $\hat{\boldsymbol{\lambda}}$ for the model parameters that maximise some training criterion $\mathcal{F}(\boldsymbol{\lambda})$ given a training dataset $\mathcal{D}$,

$$\hat{\boldsymbol{\lambda}} = \arg\max_{\boldsymbol{\lambda}} \{\mathcal{F}(\boldsymbol{\lambda}|\mathcal{D})\} . \tag{2.26}$$

For supervised training in speech recognition, the datasets are usually consist of utterances with transcriptions,

$$\mathcal{D} = \left\{ \left(\mathbf{O}^{(1)}, \mathbf{w}_{\mathtt{ref}}^{(1)} \right), \ldots, \left(\mathbf{O}^{(R)}, \mathbf{w}_{\mathtt{ref}}^{(R)} \right) \right\} \tag{2.27}$$

where $\mathbf{O}^{(r)}$ is the r^{th} observation sequence and $\mathbf{w}_{\mathtt{ref}}^{(r)}$ is the r^{th} reference transcription (word sequence).

Many different criteria, $\mathcal{F}(\boldsymbol{\lambda})$, have been proposed for estimating the parameters of generative models. Among all these algorithms, the most common training criterion is maximum likelihood (ML) estimation, where the parameters are optimised to maximise the likelihood of the training data. Although model parameters are often estimated using the ML training criterion, discriminative training criteria have become increasingly popular. These attempt to optimise the model parameters with respect to an objective function that is directly related to the classification performance. Proposed criteria include: maximum mutual information (MMI) (Bahl *et al.* 1986), minimum Bayes risk (MBR) (Byrne 2006; Kaiser *et al.* 2000) and minimum classification error (MCE) (Juang *et al.* 1995). These popular training criteria are described in the following subsections.

2.3.1 Maximum Likelihood (ML)

Maximum likelihood estimation is one of the standard approach for training the parameters of generative models. It adjusts model parameters to maximise the likelihood of a given training set. The ML training criterion can be expressed as maximising

$$\mathcal{F}_{\mathtt{ml}}(\boldsymbol{\lambda}|\mathcal{D}) = \sum_{r=1}^{R} \log p_{\boldsymbol{\lambda}}(\mathbf{O}^{(r)}|\mathbf{w}_{\mathtt{ref}}^{(r)}), \tag{2.28}$$

where $\boldsymbol{\lambda}$ is the generative model parameters and $p_{\boldsymbol{\lambda}}(\mathbf{O}^{(r)}|\mathbf{w}_{\mathtt{ref}}^{(r)})$ is the likelihood of the parameter $\boldsymbol{\lambda}$ for r-th observation sequence given the r-th label sequence. The likelihood can be calculated using the forward-backward algorithm described in Section 2.2.1.1. Maximising equation (2.28) with respect to $\boldsymbol{\lambda}$ allows ML parameter estimates

to be calculated. In practice the global optimal solution of maximum likelihood estimation is not guaranteed. However, good performance can be obtained for a variety of tasks with appropriate initialisation (Rabiner 1989; Young *et al.* 2006).

For some simple generative models such as a single Gaussian distribution, equation (2.28) can be maximised analytically by computing the partial derivatives of $\mathcal{F}_{\texttt{ml}}(\boldsymbol{\lambda}|\mathcal{D})$ with respect to the model parameter $\boldsymbol{\lambda}$, and setting them to zero. For latent variable models, such as HMMs, differentiating equation 2.28 with respect to $\boldsymbol{\lambda}$ often does not yield simple closed form estimates for the optimal model parameters. Instead expectation maximisation (EM) algorithm (Baum *et al.* 1970; Dempster *et al.* 1977; Rabiner 1989) can be used to iteratively update the model parameters.

2.3.2 *Maximum Mutual Information (MMI)*

Discriminative training criteria provide a method of optimising the model parameters with respect to an objective function that is directly related to the classification performance. One of the most widely used discriminative training criteria for generative models is maximum mutual information (MMI) estimation (Normandin 1991). In the MMI training, the following form is *maximised*

$$\mathcal{F}_{\texttt{mmi}}(\boldsymbol{\lambda}|\mathcal{D}) = \sum_{r=1}^{R} \log(P(\mathbf{w}_{\texttt{ref}}^{(r)}|\mathbf{O}^{(r)};\boldsymbol{\lambda})), \tag{2.29}$$

where $(\mathbf{O}^{(r)}, \mathbf{w}_{\texttt{ref}}^{(r)})$ is the r^{th} training pair. Using Bayes' rule, the posterior probabilities a label sequence in equation (2.29) can be expressed in terms of the generative model likelihoods $p_{\boldsymbol{\lambda}}(\mathbf{O}|\mathbf{w})$, and class priors $P(\mathbf{w})$,

$$P(\mathbf{w}_{\texttt{ref}}^{(r)}|\mathbf{O}^{(r)};\boldsymbol{\lambda}) = \frac{p_{\boldsymbol{\lambda}}(\mathbf{O}^{(r)}|\mathbf{w}_{\texttt{ref}}^{(r)})P(\mathbf{w}_{\texttt{ref}}^{(r)})}{\sum_{\mathbf{w}'} p_{\boldsymbol{\lambda}}(\mathbf{O}^{(r)}|\mathbf{w}')P(\mathbf{w}')}. \tag{2.30}$$

where $\mathbf{w}'$ denotes all possible hypothesis (label sequences) including both the correct and competing ones.

In speech recognition, the space of all possible $\mathbf{w}'$ is usually very large, therefore calculating the denominator of equation (2.30) is impractical. To avoid this, the

denominator summation is often approximated using a relatively small number of 'likely' hypotheses. These hypotheses are usually generated using Viterbi decoding of ML-estimated models (Povey 2003) and are normally stored as N-Best lists or lattices (see Section 2.4.4). With the above approximation, maximising equation (2.29) with respect to the parameters of both the generative models and the prior distributions allows MMI estimates of the model parameters to be calculated. For speech recognition, the prior distributions $P(\mathbf{w}_{\mathtt{ref}}^{(r)})$ is the language model probability (see Section 2.4.3) which is typically not estimated in conjunction with the likelihood $p_{\boldsymbol{\lambda}}(\mathbf{O}^{(r)}|\mathbf{w}_{\mathtt{ref}}^{(r)})$ so will be assumed fixed in this section.

2.3.3 *Minimum Classification Error (MCE)*

The parameter estimation of generative models based on minimum classification error (MCE) criterion (Chou *et al.* 1993; Juang and Katagiri 1992; McDermott *et al.* 2007) can be performed by *minimising*

$$\mathcal{F}_{\mathtt{mce}}(\boldsymbol{\lambda}|\mathcal{D}) = -\frac{1}{R}\sum_{r=1}^{R}\left(1+\left[\frac{P(\mathbf{w}_{\mathtt{ref}}^{(r)}|\mathbf{O}^{(r)};\boldsymbol{\lambda})}{\sum\limits_{\mathbf{w}'\neq\mathbf{w}_{\mathtt{ref}}^{(r)}} P(\mathbf{w}'|\mathbf{O}^{(r)};\boldsymbol{\lambda})}\right]^{\xi}\right)^{-1} \tag{2.31}$$

where $P(\mathbf{w}_{\mathtt{ref}}^{(r)}|\mathbf{O}^{(r)};\boldsymbol{\lambda})$ is defined in equation (2.30) and ξ is an additional free parameter. MCE is a smooth measure of the difference between the log-likelihood of correct reference sequence and all other competing word sequences (Juang and Katagiri 1992). There are some important differences between MCE and MMI. The first is that the denominator term does not include the correct word sequence. Second the log-probabilities are smoothed with a sigmoid function, which introduces an additional smoothing term ξ. When $\xi = 1$ it is possible to show that (Gales 2007)

$$\mathcal{F}_{\mathtt{mce}}(\boldsymbol{\lambda}|\mathcal{D}) = 1 - \frac{1}{R}\sum_{r=1}^{R} P(\mathbf{w}_{\mathtt{ref}}^{(r)}|\mathbf{O}^{(r)};\boldsymbol{\lambda}). \tag{2.32}$$

The second term in equation (2.32) is actually the objective function in equation (2.29).

2.3.4 Minimum Bayes's Risk (MBR)

Alternatively, maximizing the posterior probability of training data in the MMI criterion, MBR *minimising* the expected loss,

$$\mathcal{F}_{\texttt{mbr}}(\boldsymbol{\lambda}|\mathcal{D}) = \frac{1}{R}\sum_{r=1}^{R}\sum_{\mathbf{w}'} P(\mathbf{w}'|\mathbf{O}^{(r)};\boldsymbol{\lambda})\mathcal{L}(\mathbf{w}', \mathbf{w}_{\texttt{ref}}^{(r)}) \tag{2.33}$$

where $\mathbf{w}'$ denotes all possible hypothesis, $P(\mathbf{w}'|\mathbf{O}^{(r)};\boldsymbol{\lambda})$ is defined in equation (2.30) and $\mathcal{L}(\mathbf{w}', \mathbf{w}_{\texttt{ref}}^{(r)})$ is a loss functions that calculates the error between the hypothesis $\mathbf{w}'$ and the reference $\mathbf{w}_{\texttt{ref}}^{(r)}$. Designing a suitable loss function is very important and leads to several MBR-style criteria in speech recognition (Gales 2007).

0/1 loss For continuous speech recognition 0/1 loss is equivalent to a sentence-level loss function.

$$\mathcal{L}(\mathbf{w}', \mathbf{w}_{\texttt{ref}}^{(r)}) = \begin{cases} 0, & \text{if} \quad \mathbf{w}' = \mathbf{w}_{\texttt{ref}}^{(r)} \\ 1, & \text{if} \quad \mathbf{w}' \neq \mathbf{w}_{\texttt{ref}}^{(r)} \end{cases} \tag{2.34}$$

When $\xi = 1$ MCE and MBR training with this loss function are the same.

Word-level loss This loss function directly related to minimising the expected Word Error Rate (WER). It is normally computed by word-level Levenshtein distance between $\mathbf{w}'$ and $\mathbf{w}_{\texttt{ref}}^{(r)}$ (Gales 2007). Using this loss function in equation (2.33) yields the *minimum word error* (MWE) criterion (Mangu *et al.* 1999).

Phone-level loss For large vocabulary speech recognition not all word sequences will be observed. To help the generalization the loss function is often computed between the phone sequences, rather than word sequences (Gales 2007). In the literature this is known as *Minimum Phone Error* (MPE) training (Povey 2003).

Frame-level loss This loss is the Hamming distance used in (Taskar 2005). It measures the number of observations having incorrect phone labels. Using this loss function in equation (2.33) yields the *minimum phone frame error* (MPFE) criterion (Zheng and Stolcke 2005).

Some of the above criteria have been compared on the Wall Street Journal (WSJ) task in (Macherey *et al.* 2005; Schlüter *et al.* 2001). Both MCE and MPE were found to outperform MMI on this task. MPFE yielded small, but consistents, gains over MPE in (Zheng and Stolcke 2005).

2.3.5 *Maximum Margin (MM)*

In many applications, since the underlying models are usually not known and the training data is always limited, MMI and MBR-style training criteria may have over-training problems. In order to train a robust classifier that generalizes better on high-dimension space, maximum margin based approaches become popular (Gales 2007; M. Layton and M.J.F. Gales 2004; Sha and Saul 2007). The simplest form of maximum margin training criterion can be expressed as *maximising*

$$\mathcal{F}_{\texttt{mm}}(\boldsymbol{\lambda}|\mathcal{D}) = \frac{1}{R}\sum_{r=1}^{R}\left(\min_{\mathbf{w}'\neq\mathbf{w}^r}\left\{\log\left(\frac{P(\mathbf{w}^{(r)}|\mathbf{O}^{(r)};\boldsymbol{\lambda})}{P(\mathbf{w}'|\mathbf{O}^{(r)};\boldsymbol{\lambda})}\right)\right\}\right). \tag{2.35}$$

This objective function aims to maximise the distance between the log-posterior of the correct label and the "closest" incorrect labels. Note that the posteriors $P(\mathbf{w}^{(r)}|\mathbf{O}^{(r)};\boldsymbol{\lambda})$ and $P(\mathbf{w}'|\mathbf{O}^{(r)};\boldsymbol{\lambda})$ are defined in equation (2.30), however the normalization terms of these posteriors (denominator in equation (2.30)) can be cancelled out.

Many variants of maximum margin training have also been used. In (Sha and Saul 2007), the size of the margin is forced to be not smaller than a loss function. This leads to *minimising* the following objective function

$$\mathcal{F}_{\texttt{mm}}(\boldsymbol{\lambda}|\mathcal{D}) = \frac{1}{R}\sum_{r=1}^{R}\left[\max_{\mathbf{w}'\neq\mathbf{w}^r}\left\{\mathcal{L}(\mathbf{w}',\mathbf{w}_{\texttt{ref}}^{(r)}) - \log\left(\frac{P(\mathbf{w}^{(r)}|\mathbf{O}^{(r)};\boldsymbol{\lambda})}{P(\mathbf{w}'|\mathbf{O}^{(r)};\boldsymbol{\lambda})}\right)\right\}\right]_{+} \tag{2.36}$$

where $\mathcal{L}(\mathbf{w}',\mathbf{w}_{\texttt{ref}}^{(r)})$ is the Hamming distance (Sha and Saul 2007) between the two label sequences. In continuous speech recognition, this is the frame-level loss described in Section 2.3.4. To ignore the data that already classified correctly and beyond the

margin, the hinge function $[\,\cdot\,]_+$ is introduced

$$[f(x)]_+ = \begin{cases} f(x), & \text{if } f(x) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.37}$$

Note that the objective function in (2.36) is not differentiable because of the $\max\{\cdot\}$ function. In order to simplify optimisation, in (Sha and Saul 2007) the following soft-max inequality is used to approximate the non-differentiable objective function,

$$\max_i x_i \leq \log\left(\sum_i \exp(x_i)\right) \tag{2.38}$$

This yields the following upper bound of the objective function in equation (2.36),

$$\mathcal{F}_{\texttt{mm}}(\boldsymbol{\lambda}|\mathcal{D}) \leq \frac{1}{R}\sum_{r=1}^{R}\Big[-\log\Big(P(\mathbf{w}_{\texttt{ref}}^{(r)}|\mathbf{O}^{(r)};\boldsymbol{\lambda})\Big) + \log\left(\sum_{\mathbf{w}'} P(\mathbf{w}'|\mathbf{O}^{(r)};\boldsymbol{\lambda})\mathcal{L}_{\texttt{exp}}(\mathbf{w}',\mathbf{w}_{\texttt{ref}}^{(r)})\right)\Big]_+ \tag{2.39}$$

where the new loss function is

$$\mathcal{L}_{\texttt{exp}}(\mathbf{w}',\mathbf{w}_{\texttt{ref}}^{(r)}) = \begin{cases} 0, & \text{if} \quad \mathbf{w}' = \mathbf{w}_{\texttt{ref}}^{(r)} \\ \exp\left\{\mathcal{L}(\mathbf{w}',\mathbf{w}_{\texttt{ref}}^{(r)})\right\}, & \text{if} \quad \mathbf{w}' \neq \mathbf{w}_{\texttt{ref}}^{(r)} \end{cases} \tag{2.40}$$

This lower bound has properties related to both the MMI and MBR criterion (Gales 2007). The first term within the hinge-loss function is the negated log-posterior, the same as the MMI objective function in (2.29). The second term is the logarithm of MBR objective function in (2.33) with a new loss function given by (2.40).

Furthermore, the lower bound in equation (2.39) can also be related to the *boosted MMI* (bMMI) criterion (Povey *et al.* 2008). In bMMI criterion the following objective function is *maximised* (Saon and Povey 2008)

$$\mathcal{F}_{\texttt{bmmi}}(\boldsymbol{\lambda}|\mathcal{D}) = \frac{1}{R}\sum_{r=1}^{R}\log\left(\frac{p_{\boldsymbol{\lambda}}(\mathbf{O}^{(r)}|\mathbf{w}_{\texttt{ref}}^{(r)})P(\mathbf{w}_{\texttt{ref}}^{(r)})}{\sum_{\mathbf{w}'} p_{\boldsymbol{\lambda}}(\mathbf{O}^{(r)}|\mathbf{w}')P(\mathbf{w}')\; e^{-\epsilon\mathcal{A}(\mathbf{w}',\mathbf{w}_{\texttt{ref}}^{(r)})}}\right) \tag{2.41}$$

where $\mathcal{A}(\mathbf{w}',\mathbf{w}_{\texttt{ref}}^{(r)})$ is the accuracy function of label sequence $\mathbf{w}'$ against the reference $\mathbf{w}_{\texttt{ref}}^{(r)}$, and ϵ is a boosting factor. There are two differences between the maximum

margin criterion in (2.39) and bMMI criterion in (2.41). First, in the former, the hinge-loss function is used to prevent the log-posterior ratio to grow arbitrary large. Second, in the latter, a scaled phone-level accuracy $e^{-\epsilon\mathcal{A}(\mathbf{w}',\mathbf{w}_{\mathrm{ref}}^{(r)})}$ is used instead of the loss function in equation (2.40). Note that the criterion in (2.39) is an approximation of the "exact" maximum margin in (2.36).

2.4 Speech Recognition

In the previous sections, generative models and their training criteria were introduced. Although these models have been applied to a wide variety of tasks, including speech recognition (Rabiner 1989), machine translation (Somers 1992) and computational biology (Krogh *et al.* 1994), this work focuses on the task of speech recognition. The aim of an automatic speech recognition (ASR) system is to produce a word sequence (transcription) given a speech waveform. The basic structure of an ASR system is shown in Figure 2.5. This system consists of five principal components: the Front-end processing, acoustic model, language model, dictionary and decoding algorithm.

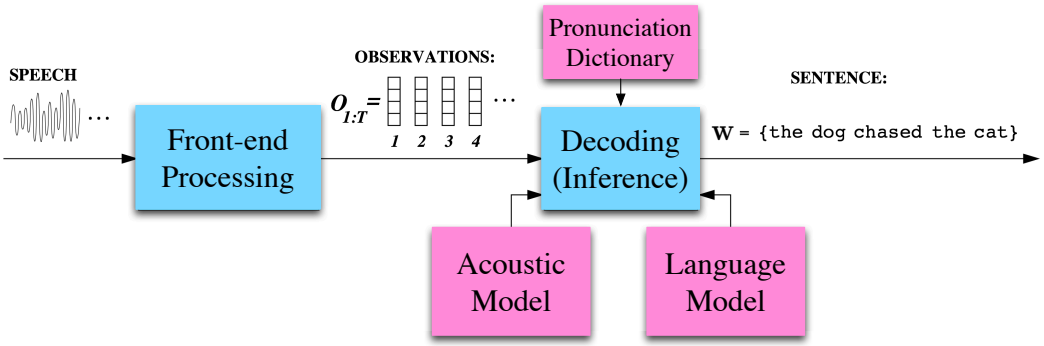


Figure 2.5 *A typical structure of automatic speech recognition system based on generative models.*

The first stage of speech recognition is to compress the speech signals into a sequence of acoustic feature vectors, referred to as *observations*, denoted as $\mathbf{O} = \{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_T\}$. This process is known as the *feature extraction* or *front-end processing*. Given the ob-

servation sequence, generally two main sources of information are required to decode the most likely word sequence: the *language model* and *acoustic model*. The decoder uses Bayes' decision rule to calculate the most likely word transcription associated with the observation sequence,

$$\hat{\mathbf{w}} = \arg\max_{\mathbf{w}} P(\mathbf{w}|\mathbf{O}) = \arg\max_{\mathbf{w}} \left(p_{\boldsymbol{\lambda}}(\mathbf{O}|\mathbf{w})P(\mathbf{w})\right), \tag{2.42}$$

where $P(\mathbf{w})$ is the language model and $p_{\boldsymbol{\lambda}}(\mathbf{O}|\mathbf{w})$ is the generative acoustic model. Note that $P(\mathbf{O})$ is ignored in equation (2.42) as it is independent of the word sequence $\mathbf{w}$. These principal parts of a speech recognition system are discussed in more details in the following sections.

2.4.1 *Front-end processing*

The raw form of speech is a continuous speech waveform. This waveform is normally recorded using a microphone and sampled at 8kHz or 16kHz depends on the channel. To effectively perform speech recognition, the sampled waveform is usually converted into a sequence of time-discrete parametric vectors. These parametric vectors are assumed to contain sufficient information and be compact enough for efficient decoding, referred to as *feature vectors* or *observations*, $\mathbf{O} = \{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_T\}$. There are two widely used feature extraction schemes: Mel-frequency Cepstral coefficients (MFCCs) (Davis and Mermelstein 1980) and perceptual linear prediction (PLP) (Hermansky 1990). Both are based on Cepstral analysis.

The speech signal is assumed to be quasi-stationary. It is then split into discrete segments normally with 10ms shifting rate and 25ms window length. These discrete segments are often referred to as *frames*. A pre-emphasising approach is sometimes applied during the feature extraction, where overlapping window functions, such as Hamming windows are commonly used to smooth the signals and reduce the boundary effect in signal processing (Deller *et al.* 2000). A fast Fourier transform (FFT) is then performed on the time-domain speech signals of each frame, generating the segment frequency-domain power spectrum. This is then warped using either a Mel-

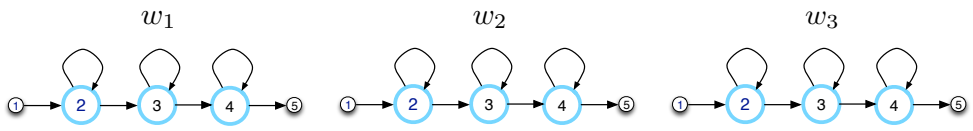
frequency scale (MFCCs) or a Bark-frequency scale (PLPs). The resulting power spectrum is filtered and compressed. MFCC and PLP vectors are then generated using an inverse discrete cosine transform (IDCT) or a linear prediction (LP) analysis respectively. 13-dimensional vectors are typically extracted. In many speech recognition systems, the MFCC and PLP vectors are extended using first-order (delta) and second-order (delta-delta) dynamic features. These increase the observation-space to 39 dimension, and help to overcome the HMM assumption that observations are conditionally independent given the state sequence (Forney 1973).

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) (Goel and Andreou 1998; Kumar 1997). 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) (Lee and Rose 1996; Uebel and Woodland 1999), and Cepstral mean and variance normalisation (CMN and CVN) (Furui 1981; Viikki and Laurila 1998).

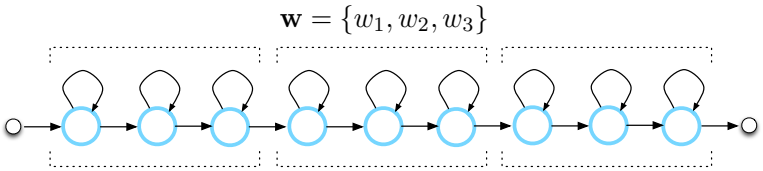
2.4.2 *Acoustic models*

Acoustic models are used to estimate the observation sequence probabilities, $p_{\boldsymbol{\lambda}}(\mathbf{O}|\mathbf{w})$, given the complete sentence transcription. For medium and large vocabulary, the number of possible sentences (classes) is very large. It is impractical to associate the parameters to each class at sentence level. In order to address this issue, structure can be introduced into the model, where sentences are broken down into words or phone units, and modelled by combining units into a composite sentence model. This leads to the use of structured generative models for speech recognition.

As discussed earlier, hidden Markov models are the most common form of structured generative models. For speech recognition tasks with a limited vocabulary it is often possible train HMMs for every possible word. However, as the number of



(a) Three HMMs $\boldsymbol{\lambda}^{(w_1)}, \boldsymbol{\lambda}^{(w_2)}, \boldsymbol{\lambda}^{(w_3)}$



(b) A composite HMM $\boldsymbol{\lambda}^{(\mathbf{w})}$

Figure 2.6 *A composite HMM constructed from three individual HMMs.*

words in the vocabulary increases, it becomes increasingly difficult to robustly estimate HMMs for all words. Instead, a *pronunciation dictionary* used to split words into smaller sub-word units known as phones. After estimating HMMs for each phone (or word), sentence models can be generated by concatenating the phone (or word) HMMs as shown in Figure 2.6. For example, concatenating HMMs for phones (or words) w_1, w_2 and w_3 could form a sentence model called *composite HMM* with parameters $\boldsymbol{\lambda}^{(\mathbf{w})} = \{\boldsymbol{\lambda}^{(w_1)}, \boldsymbol{\lambda}^{(w_2)}, \boldsymbol{\lambda}^{(w_3)}\}$. The parameters of the HMM for each phone (or word) can be estimated using any of training criteria discussed in Section 2.3, e.g., ML, MMI and MBR estimation.

There are two main types of phone model sets used, *mono-phones* which are context-independent phones, and *context-dependent phones*. The *mono-phone* set uses individual phones as the sub-word unit and does not take into account the context information. However, due to the co-articulatory effect, the pronunciation of the current phone is highly dependent on the preceding and following phones. Thus, for many speech recognition tasks, especially for LVCSR, the use of mono-phones does not yield good performance. Thus, context-dependent phone models are often used. One form is triphones which 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). One issue with using triphones is that the number of pos-

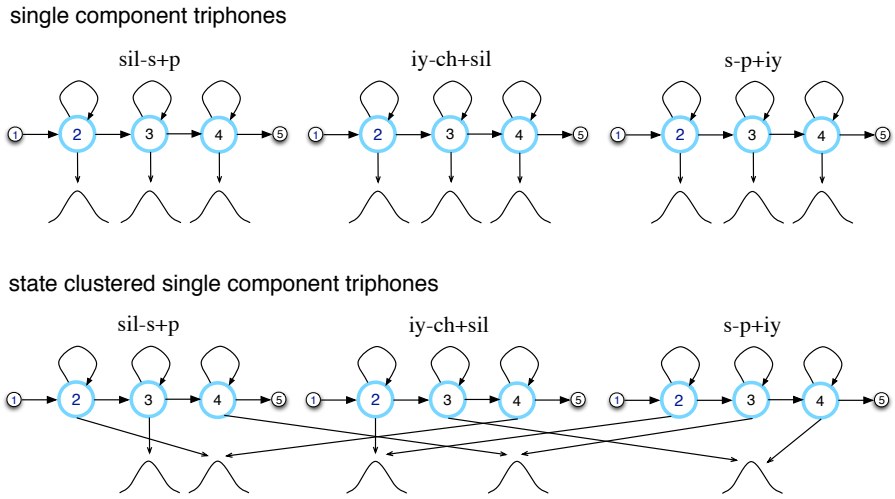


Figure 2.7 *State tying for single Gaussian triphones. The triphone symbol "s-p+iy" denotes the context-dependent version of the phone "p" which is to be used when the left neighbour is the phone "s" and the right neighbour is the phone "iy".*

sible acoustic units is significantly increased. For example, for a mono-phone set with 46 phones, the number of possible cross-word triphones is about 100,000. It is hard to collect sufficient training data to robustly train all triphones. To solve this problem, *parameter tying* or clustering techniques are often used (Young *et al.* 1994). The basic idea of the technique is to consider a group of parameters as sharing the same set of values. In training, statistics of the whole group is used to estimate the shared parameter. Tying can be performed at various levels, such as phones, states or Gaussian component. The most widely used approach is to do state level parameter tying, referred to as *state clustering* (Young 1995). In state clustering, an output distribution is shared among a group of states as illustrated in Figure 2.7.

2.4.3 *Language models*

In speech recognition systems, the prior distribution of sentence transcriptions, $P(\mathbf{w})$, is usually estimated using a language model. Given a word sequence, $\mathbf{w} = \{w_1, \ldots, w_L\}$, the language model probability is usually written as a product of the conditional prob-

abilities of words given their history,

$$P(\mathbf{w}) = P(w_1, \ldots, w_L) = \prod_{l=1}^{L} P(w_l | w_{l-1}, \ldots, w_1). \tag{2.43}$$

For continuous speech recognition, the vocabulary is often too large to allow robust estimation of $P(\mathbf{w})$. To improve robustness, the language model is usually simplified by assuming that word probabilities only depend on the last $n-1$ words. This allows the word history to be truncated,[6]

$$P(\mathbf{w}) = P(w_1, \ldots, w_L) \approx \prod_{l=1}^{L} P(w_l | w_{l-1}, \ldots, w_{l-n+1}). \tag{2.44}$$

The conditional probabilities $P(w_l | w_{l-1}, \ldots, w_{l-n+1})$ are estimated from training texts. Let $C(w_{l-n+1}, \ldots, w_{l-1}, w_l)$ be the number of times the underlying n-gram occurs in the training texts. A maximum likelihood (ML) estimate is then given by (Chen and Goodman 1998; Gales and Young 2007)

$$P(w_l | w_{l-1}, \ldots, w_{l-n+1}) = \frac{C(w_{l-n+1}, \ldots, w_{l-1}, w_l)}{C(w_{l-n+1}, \ldots, w_{l-1})} \tag{2.45}$$

Popular language models are the bigram and trigram models, with n equal to 2 and 3 respectively (Moore 2001; Shannon 1948). Although the n-gram language model probabilities in equation (2.44) are relatively easier to calculate than the probabilities in equation (2.43), robust estimation of the probabilities for all word combinations is usually impossible. Instead, smoothing algorithms such as discounting and backing-off are normally used (Chen and Goodman 1998; Jelinek 1998; Katz 1987). For instance, the Katz smoothing scheme (Katz 1987) sets conditional probabilities by

$$P(w_l | w_{l-1}, \ldots, w_{l-n+1}) = \tag{2.46}$$
$$\begin{cases} D\dfrac{C(w_{l-n+1}, \ldots, w_{l-1}, w_i)}{C(w_{l-n+1}, \ldots, w_{l-2}, w_{l-1})}, & \text{if} \quad 0 \leq C(w_{l-n+1}, \ldots, w_{l-1}, w_l) \leq C^{\texttt{min}} \\ \dfrac{C(w_{l-n+1}, \ldots, w_{l-1}, w_l)}{C(w_{l-n+1}, \ldots, w_{l-2}, w_{l-1})}, & \text{if} \quad C(w_{l-n+1}, \ldots, w_{l-1}, w_l) > C^{\texttt{min}} \\ \dfrac{P(w_l | w_{l-1}, \ldots, w_{l-n+2})}{Z(w_{l-n+1}, \ldots, w_{l-2}, w_{l-1})}, & \text{otherwise} \end{cases}$$

[6] Note that the context of the first $n-1$ N-grams can be filled with start-of-sentence symbols, $<s>$. The end-of-sentence symbol, $</s>$, are also appended to every sentence as additional words.

where D is a discounting coefficient for n-grams observed less than $C^{\texttt{min}}$ times in the training texts and $Z(w_{l-n+1}, \ldots, w_{l-2}, w_{l-1})$ is a normalisation constant to ensure a valid probability mass function. The goal of discounting is to reserve probability mass for the unseen n-grams (Moore 2001). There are several options how the discounting coefficient can be set (Chen and Goodman 1998). For instance, in Good-Turing discounting (Good 1953) the n-grams occurring exactly c times in the training texts are discounted by (Gales and Young 2007)

$$D = \frac{(c+1)C_{c+1}}{c\,C_c} \tag{2.47}$$

where C_c is the number of n-grams occurring exactly c times in the training texts. If $\{w_{l-n+1}, \ldots, w_{l-1}, w_l\}$ has not been observed in the training texts then an estimate of its conditional probability is obtained from the third case in equation (2.46), which uses the estimate of conditional probability associated with the $(n-1)$-gram scaled by the normalisation constant. In practice, the acoustic model and language model probabilities may have different dynamic ranges. To compensate for this mismatch, the dynamic range of language model probabilities is scaled by a language model scale factor κ. The value of κ is normally determined experimentally (Young *et al.* 2006).

In addition to n-gram models, a range of other language models has been investigated for speech classification tasks: class n-gram models (Brown *et al.* 1992; Moore 2001), maximum entropy language models (Rosenfeld 1994), neural network language models (Bengio *et al.* 2003) and recurrent neural network language models (Mikolov *et al.* 2010).

2.4.4 Decoding and Lattices

Decoding is at the core of speech recognition systems. The decoding algorithm determines the best sentence transcription for a given observation sequence. 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 equation (2.42). Section 2.2 has described the form of likelihood $p_{\boldsymbol{\lambda}}(\mathbf{O}|\mathbf{w})$. As in equation (2.8), to find the most

likely word sequence, the state sequence $\boldsymbol{\theta}$ should be marginalised out,

$$\hat{\mathbf{w}} = \arg\max_{\mathbf{w}} P(\mathbf{w}) \sum_{\boldsymbol{\theta}} P(\boldsymbol{\theta}|\mathbf{w}) p_{\boldsymbol{\lambda}}(\mathbf{O}|\boldsymbol{\theta}; \mathbf{w}) \tag{2.48}$$

However, summing over all possible $\boldsymbol{\theta}$ is computationally infeasible. Instead, the sum in (2.48) can be replaced by a max operator. Thus, rather than finding the best word sequence, decoders find the word sequence corresponding to the best state sequence,

$$\hat{\mathbf{w}} \approx \arg\max_{\mathbf{w}} P(\mathbf{w}) \max_{\boldsymbol{\theta}} P(\boldsymbol{\theta}|\mathbf{w}) p_{\boldsymbol{\lambda}}(\mathbf{O}|\boldsymbol{\theta}; \mathbf{w}) \tag{2.49}$$

If all possible word sequences can be compactly encoded into a single composite HMM (Section 2.4.2) then the solution of decoding problem equation (2.49) can be found efficiently by using the Viterbi algorithm described in Section 2.2.2.

In practice, the use of the Viterbi algorithm for decoding becomes very complex due to the topology, the n-gram language model constraints, the use of cross-word context-dependent units and the size of memory required to hold the composite HMM (Gales and Young 2007). A range of methods have been proposed to handle these problems such as dynamic decoding (Odell *et al.* 1994; Ortmanns *et al.* 1997), stack decoding (Jelinek 1969), and static decoding based on weighted finite-state transducer (WFST) technology (Mohri *et al.* 2002; Watanabe *et al.* 2010).

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,

$$\text{WER} = 100 \left(\frac{D + S + I}{N} \right) \tag{2.50}$$

where N denotes the total number of words in the reference transcription. Word error rate are quoted as percentages.

Lattices Although the main purpose of decoder is to find the most likely hypothesised word sequence, it is also usually possible to output N most likely candidates or the N-best list (Young *et al.* 2009). As the number of candidates increases, the use of N-best lists becomes computationally and memory inefficient. In order to store N-best lists in a compact and efficient manner, the use of word lattices can be adopted (Odell *et al.* 1994; Thompson 1990). The use of lattices is useful as it allows multiple passes over the observation sequence without the computational expense of repeatedly solving the decoding problem from scratch (Gales and Young 2007).

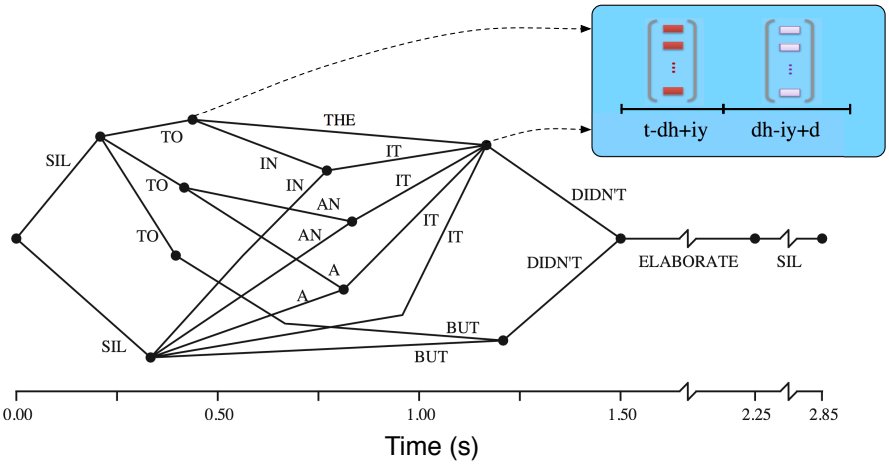


Figure 2.8 *An example lattice with phone-marked information. The colourful vectors are the segmental feature space described in Section 5.1.2.*

A word lattice consists of a set of nodes representing points in time and a set of spanning arcs representing word hypotheses (Gales and Young 2007). Figure 2.8 shows an example lattice. Lattices can also be converted into phone-marked lattices (Young *et al.* 2009). A phone-marked lattice is an extension to a word lattice where each word arc is split into phone arcs corresponding to the underlying sequence of phones. In addition to label information, each arc can also carry additional information such as acoustic, pronunciation and language model scores. For example, each phone arc can contain acoustic model scores, such as the HMM likelihood associated with the phone. Figure 2.8 gives an example of phone-marked lattice with segmental

feature space. This feature space will be described in details in Section 5.1.2 and used in Chapters 6 and 7.

2.4.5 Adaptation to Speaker and Noise Condition

In speech recognition, the acoustic conditions during training and testing are seldom matched. There are many sources of variability in the speech signal, such as inter-speaker variability, intra-speaker variability, background noise conditions, channel distortion and reverberant noise (longer term channel distortions) (Gales 2011). A number of approaches have been developed to reduce the level of variability: some are based on general linear transformations (Gales 1998; Leggetter and Woodland 1995), e.g., the *Maximum likelihood linear regression* (MLLR) approach in (Gales 1998; Leggetter and Woodland 1995); others are based on a model of how the mismatch impacts the acoustic models or observations (Acero 1993; Acero *et al.* 2000; Gales 1995; Lee and Rose 1996), e.g., the *Vector Taylor series* (VTS) compensation approach in (Moreno 1996).

The rest of this section adopts bar notation to denote unmodified, canonical, acoustic models and unmodified, "clean", observations. For instance, $\overline{\boldsymbol{\lambda}}$ denotes the canonical set of HMM parameters, whilst $\boldsymbol{\lambda}$ denotes the *adapted* set of HMM parameters. Similarly, $\overline{\boldsymbol{o}}$ denotes the "clean" observation, whilst $\boldsymbol{o}$ denotes the noise-corrupted observation.

2.4.5.1 Maximum Likelihood Linear Regresssion

Various configurations of linear transforms have been proposed. In the simplest case, a global *maximum likelihood linear regression* (MLLR) transform may be applied to mean vectors (Leggetter and Woodland 1995)

$$\boldsymbol{\mu}_{j,m} = \mathbf{A}\overline{\boldsymbol{\mu}}_{j,m} + \boldsymbol{b} \tag{2.51}$$

where $\mathbf{A}$, $\boldsymbol{b}$ are transform parameters associated with mean vectors. This configuration is usually called *mean MLLR* (Young *et al.* 2009). In addition to mean vectors, it

is also possible to adapt covariance matrices in which case (Gales 1998)

$$\boldsymbol{\Sigma}_{j,m} = \mathbf{H}\overline{\boldsymbol{\Sigma}}_{j,m}\mathbf{H}^{\mathsf{T}} \tag{2.52}$$

where $\mathbf{H}$ are transform parameters associated with covariance matrices. This configuration is usually called *variance MLLR* (Young *et al.* 2009). When both mean vectors and covariance matrices are adapted then the state-component $s^{j,m}$ output density may be computed by transforming observations and mean vectors whilst keeping covariance matrices unchanged as follows (Gales and Young 2007)

$$\begin{aligned} p(\boldsymbol{o}|s^{j,m},\mathcal{T}) &= \mathcal{N}(\boldsymbol{o};\mathbf{A}\overline{\boldsymbol{\mu}}_{j,m}+\boldsymbol{b},\mathbf{H}\overline{\boldsymbol{\Sigma}}_{j,m}\mathbf{H}^{\mathsf{T}}) && (2.53)\\ &= |\mathbf{H}|^{-1}\mathcal{N}(\mathbf{H}^{-1}\boldsymbol{o};\mathbf{H}^{-1}(\mathbf{A}\overline{\boldsymbol{\mu}}_{j,m}+\boldsymbol{b}),\overline{\boldsymbol{\Sigma}}_{j,m}) && (2.54)\end{aligned}$$

where $\mathcal{T}$ are transform parameters $\mathbf{A}$, $\boldsymbol{b}$ and $\mathbf{H}$. Using this form it is possible to efficiently apply full transformations, especially in situations when covariance matrices are diagonal (Young *et al.* 2009). In addition, when the transformation matrices $\mathbf{A}$ and $\mathbf{H}$ are constrained to be the same, then

$$\begin{aligned} \boldsymbol{\mu}_{j,m} &= \mathbf{A}\overline{\boldsymbol{\mu}}_{j,m}+\boldsymbol{b} && (2.55)\\ \boldsymbol{\Sigma}_{j,m} &= \mathbf{A}\overline{\boldsymbol{\Sigma}}_{j,m}\mathbf{A}^{\mathsf{T}} && (2.56)\end{aligned}$$

Thus the state-component $s^{j,m}$ output density can be expressed as (Gales and Young 2007)

$$\begin{aligned} p(\boldsymbol{o}|s^{j,m},\mathcal{T}) &= \mathcal{N}(\boldsymbol{o};\mathbf{A}\overline{\boldsymbol{\mu}}_{j,m}+\boldsymbol{b},\mathbf{A}\overline{\boldsymbol{\Sigma}}_{j,m}\mathbf{A}^{\mathsf{T}}) && (2.57)\\ &= |\mathbf{A}^{-1}|\mathcal{N}(\mathbf{A}^{-1}\boldsymbol{o}-\mathbf{A}^{-1}\boldsymbol{b};\overline{\boldsymbol{\mu}}_{j,m},\overline{\boldsymbol{\Sigma}}_{j,m}) && (2.58)\\ &= |\mathbf{A}^{-1}|\mathcal{N}(\overline{\boldsymbol{o}};\overline{\boldsymbol{\mu}}_{j,m},\overline{\boldsymbol{\Sigma}}_{j,m}) && (2.59)\end{aligned}$$

where $\overline{\boldsymbol{o}}$ is the transformed observation vector given by

$$\overline{\boldsymbol{o}} = \mathbf{A}^{-1}\boldsymbol{o}-\mathbf{A}^{-1}\boldsymbol{b} \tag{2.60}$$

This configuration is usually called *constrained MLLR* (CMLLR) (Gales 1998). Compared to the mean and variance MLLR, the CMLLR does not require transforming

means and covariances which makes this configuration efficient if the speaker (or environment) rapidly changes (Gales and Young 2007). The details of estimating the transform parameters $\mathcal{T}$ can be found in (Gales and Young 2007) .

2.4.5.2 *Vector Taylor Series*

To compute the effect of the acoustic noise on the observations of a speech recogniser, an expression for the mismatch between clean and corrupted speech is needed. The VTS adopts a simplified model of the noisy acoustic environment Acero *et al.* (2000) or noise model, which combines various additive and convolutional noise sources into single additive and linear channel or convolutional noises, as shown in Figure 2.9. In

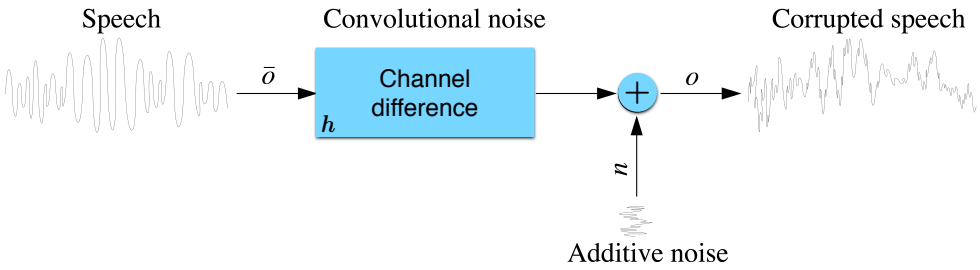


Figure 2.9 *A simplified model of noisy acoustic environment.*

the time domain, the additive noise $\boldsymbol{n}$ and the convolutional noise $\boldsymbol{h}$ transform the clean speech $\bar{\boldsymbol{o}}$, resulting in noise-corrupted speech $\boldsymbol{o}$ (Acero 1993):

$$\boldsymbol{o} = \boldsymbol{h} * \bar{\boldsymbol{o}} + \boldsymbol{n} \tag{2.61}$$

where $*$ denotes convolution. In the mel-cepstral domain, the *mismatch function* between the *static* clean speech $\bar{\boldsymbol{o}}^s$ and noise corrupted speech observations $\boldsymbol{o}^s$ can be expressed as (Acero *et al.* 2000)

$$\boldsymbol{o}^s = \bar{\boldsymbol{o}}^s + \boldsymbol{h}^s + \mathbf{C}\log\left(\mathbf{1} + \exp\left(\mathbf{C}^{-1}(\boldsymbol{n}^s - \bar{\boldsymbol{o}}^s - \boldsymbol{h}^s)\right)\right), \tag{2.62}$$

where $\mathbf{C}$ is the Discrete Cosine Transformation (DCT) matrix, and $\log(\cdot)$ and $\exp(\cdot)$ indicate the element-wise logarithm and exponent. The superscript s denote the static coefficients and parameters. The observation vector $\boldsymbol{o}_t$ is often formed of the static parameters appended by the delta and delta-delta parameters: $\boldsymbol{o}_t = [\boldsymbol{o}_t^{s\mathsf{T}}\ \Delta\boldsymbol{o}_t^{s\mathsf{T}}\ \Delta^2\boldsymbol{o}_t^{s\mathsf{T}}]^{\mathsf{T}}$.

The target of model-based compensation methods is to obtain the parameters of the noise-corrupted speech model from the clean speech and noise models. Most model-based compensation schemes assume that if the speech and noise models are Gaussian, $\mathcal{N}(\bar{\boldsymbol{\mu}}^s, \bar{\boldsymbol{\Sigma}}^s)$, $\mathcal{N}(\boldsymbol{\mu}^{s,n}, \boldsymbol{\Sigma}^{s,n})$ and $\mathcal{N}(\boldsymbol{\mu}^{s,h}, \boldsymbol{\Sigma}^{s,h})$,[7] then the combined noisy model will also be Gaussian, $\mathcal{N}(\boldsymbol{\mu}^s, \boldsymbol{\Sigma}^s)$. Thus to compute the expected value of the observation for each clean speech component, the following need to be computed

$$\boldsymbol{\mu}_m^s = \mathcal{E}\{\boldsymbol{o}^s|m\}; \quad \boldsymbol{\Sigma}_m^s = \text{diag}\left(\mathcal{E}\{\boldsymbol{o}^s\boldsymbol{o}^{s\mathsf{T}}|m\} - \boldsymbol{\mu}_m^s\boldsymbol{\mu}_m^{s\mathsf{T}}\right). \tag{2.63}$$

where the expectations are over the distribution of component m of the clean speech model and the noise distribution combined in equation (2.62). There is no simple closed-form solution to these equations so various approximations such as Parallel Model Combination (Gales and Young 1996) and Vector Taylor Series (Acero *et al.* 2000) have been proposed.

Vector Taylor series model-based compensation is a popular model-based compensation approach. There are a number of possible forms that have been examined. The first-order VTS scheme described comprehensively in (Liao and Gales 2006) is frequently used. In this scheme the static noise-corrupted mean $\boldsymbol{\mu}_m^s$ and covariance $\boldsymbol{\Sigma}_m^s$ are given by

$$\boldsymbol{\mu}_m^s = \bar{\boldsymbol{\mu}}_m^s + \boldsymbol{\mu}^{s,h} + \mathbf{C}\log\left(\mathbf{1} + \exp\left(\mathbf{C}^{-1}(\boldsymbol{\mu}^{s,n} - \bar{\boldsymbol{\mu}}_m^s - \boldsymbol{\mu}^{s,h})\right)\right) \tag{2.64}$$

$$\boldsymbol{\Sigma}_m^s = \text{diag}\left(\mathbf{A}_m\bar{\boldsymbol{\Sigma}}_m^s\mathbf{A}_m^{\mathsf{T}} + (\mathbf{I} - \mathbf{A}_m)\boldsymbol{\Sigma}^{s,n}(\mathbf{I} - \mathbf{A}_m)^{\mathsf{T}}\right) \tag{2.65}$$

The matrix $\mathbf{A}_m$ is the partial derivative, $\partial\boldsymbol{o}^s/\partial\bar{\boldsymbol{o}}^s$. It can be expressed as

$$\mathbf{A}_m = \left.\frac{\partial\boldsymbol{o}^s}{\partial\bar{\boldsymbol{o}}^s}\right|_{\bar{\boldsymbol{\mu}}_m^s, \boldsymbol{\mu}^{s,n}, \boldsymbol{\mu}^{s,h}} = \mathbf{C}\mathbf{F}_m\mathbf{C}^{-1} \tag{2.66}$$

where $\mathbf{C}$ is a DCT matrix and $\mathbf{F}_m$ is a diagonal matrix with elements given by $\mathbf{1} + \exp(\mathbf{C}^{-1}(\boldsymbol{\mu}^{s,n} - \bar{\boldsymbol{\mu}}_m^s - \boldsymbol{\mu}^{s,h}))$. The noise model parameters, $\boldsymbol{\mu}^{s,n}$, $\boldsymbol{\Sigma}^{s,n}$ and $\boldsymbol{\mu}^{s,h}$, are seldom known in advance and must be estimated from the test data. Normally, the noise parameters are estimated in a ML fashion from the silence available immediately before the test utterance.

[7] The convolutional noise is usually assumed to be constant, in which case $\boldsymbol{\Sigma}^{s,h} = \mathbf{0}$ (Acero *et al.* 2000).

2.5 Summary

Generative techniques model the probability density function of observations given the class labels. Given a prior distribution of the class labels, decision boundaries can be calculated using a combination of Bayes' rule and Bayes' decision rule. A range of training criteria, such as maximum likelihood (ML), maximum mutual information (MMI), minimum phone error (MPE) and maximum margin (MM), were introduced and compared. The choice of statistical model is important and the appropriate form of model to use is normally task and data dependent. This chapter introduced the most important model suitable for speech recognition, hidden Markov models (HMMs). The various of techniques for building the state-of-the-art HMM-based systems, e.g., DNN likelihood calculation, discriminative training and speaker and noise adaptation, were discussed.

Chapter 3

Unstructured Discriminative Models

In the previous chapter, generative models were introduced for speech recognition. Most continuous speech recognition systems use structured generative models, in the form of hidden Markov models (HMMs) as the acoustic models. Likelihoods from these HMMs are combined with a prior, usually an n-gram language model, to yield the sentence posterior based on Bayes' rule. This enables posteriors of all possible sentences to be obtained. Although discriminative training (Byrne 2006; Juang and Katagiri 1992; Keshet *et al.* 2011; Sha and Saul 2007; Woodland and Povey 2002) of HMMs has been shown to yield performance gains, the underlying acoustic models are still generative, with the standard HMM conditional independence assumptions, and the form of posteriors are found by Bayes' rule. This has led to interest in discriminative models.

Discriminative schemes are an alternative approach to sequential data classification. Unlike generative approaches, these directly model the mapping of observations $\mathbf{O}$ to the class $\mathbf{w}$, either as a *conditional distribution* or as a *function*. Depending on whether the internal structure in the class $\mathbf{w}$ is considered, discriminative models can

be divided into unstructured and structured approaches (see examples in Table 1.1). This chapter focuses on the unstructured discriminative models, including logistic regression models, support vector machines (SVMs) and multi-class SVMs (MSVMs). These models assume class labels are independent and have no structure, thus the symbol w (instead of $\mathbf{w}$) is used as the output class of models in this chapter. These models can be directly applied to isolated word recognition (Birkenes 2007) or frame-level phone classification (Salomon *et al.* 2002). However, when applying these unstructured models to complete utterances in continuous speech recognition, the space of possible classes becomes very large. One option to handle this issue are acoustic code-breaking based schemes discussed at the end of this chapter.

3.1 Logistic Regression Model

Logistic Regression (LR) is an approach to learning functions of the form $f : \mathbf{O} \rightarrow w$, or $P(w|\mathbf{O})$ in the case where w is discrete-valued, and $\mathbf{O} = \{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_T\}$ is any vector sequence containing discrete or continuous variables. Classification is accomplished by selecting the class label $\hat{w}$ that giving the largest conditional probability,

$$\hat{w} = \arg\max_{w} P(w|\mathbf{O}) \tag{3.1}$$

Before presenting the general form of the logistic regression model $P(w|\mathbf{O})$, first consider the simple case of two classes ($C = 2$). A popular model for the conditional probability of class $w = 1$ given $\mathbf{O}$ is

$$P(w = 1|\mathbf{O}) = \frac{e^f}{1 + e^f} \tag{3.2}$$

and the conditional probability for class $w = -1$ is

$$P(w = -1|\mathbf{O}) = \frac{1}{1 + e^f} \tag{3.3}$$

where the discriminant function $f = \boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\psi}(\mathbf{O})$ is a dot product of feature vector $\boldsymbol{\psi}(\mathbf{O})$ and parameters $\boldsymbol{\alpha}$. The feature vectors are used to capture the long-term information by mapping the observation sequence $\mathbf{O}$ into an D-dimensional Euclidean

space. This feature mapping $\psi(\cdot)$ can transform variable length sequences into a fixed-dimensional vector. A simple example of the feature vector is

$$\boldsymbol{\psi}(\mathbf{O}) = \left[\sum_{t=1}^{T} \boldsymbol{o}_t\right] \tag{3.4}$$

Notice that equation (3.3) follows directly from equation (3.2), because the sum of these two probabilities must equal 1. The function in (3.2) is known as the *logistic function* and apart from being just a squashing function that maps f into the interval $[0, 1]$, it also has good probabilistic properties in the context of classification (Jordan *et al.* 1995). The graphic representation of the logistic function is shown in Figure 3.1.

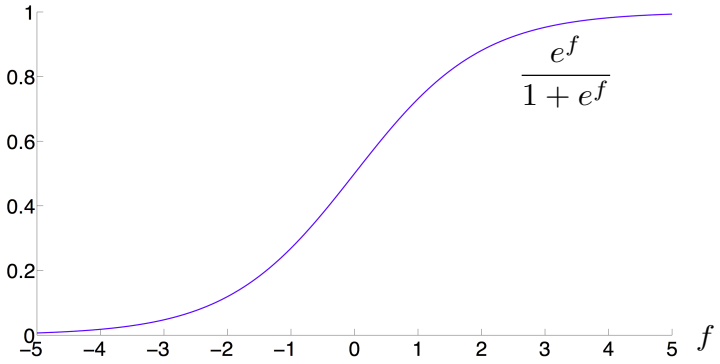


Figure 3.1 *A logistic function is a common sigmoid curve.*

The extension to classification problems with more than two classes is to model the conditional probabilities with the *softmax function* or *multinomial logistic regression* defined by

$$P(w|\mathbf{O}) = \frac{e^{f_w}}{\sum_{w=1}^{C} e^{f_w}}, \qquad w \in \{1, \ldots, C\} \tag{3.5}$$

where f_w is the discriminant function for class w parameterized by the weight vector $\boldsymbol{\alpha}_w$

$$f_w = \boldsymbol{\alpha}_w^{\mathsf{T}} \boldsymbol{\psi}(\mathbf{O})$$

Figure 3.2 illustrates the logistic regression model with feature space. Due to the probability constraint $\sum_{w=1}^{C} P(w|\mathbf{O}) = 1$, the weight vector for one of the classes, e.g. $\boldsymbol{\alpha}_w$,

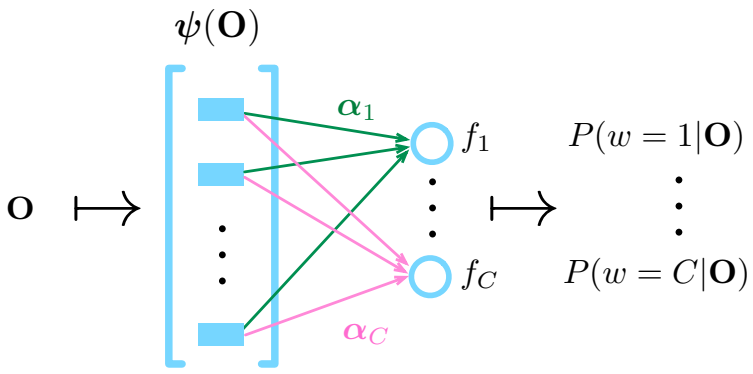


Figure 3.2 *The model of the posterior probabilities for a class, $P(w|\mathbf{O})$, with feature space $\boldsymbol{\psi}(\mathbf{O})$.*

does not need to be estimated and can be set to all zeros (Tanabe 2001). However, in this section we follow the convention in (Birkenes 2007) and keep the redundant parameters. As explained in (Tanabe 2001), this is done for numerical stability reasons, and enables all classes to be treated equally. The parameters of model in (3.5) can be combined in a single column vector

$$\boldsymbol{\alpha} = \left[\boldsymbol{\alpha}_1^{\mathsf{T}}, \ldots, \boldsymbol{\alpha}_w^{\mathsf{T}}, \ldots, \boldsymbol{\alpha}_C^{\mathsf{T}}\right]^{\mathsf{T}} \tag{3.6}$$

Thus the discriminative function $f_w = \boldsymbol{\alpha}_w^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O})$ can also be expressed in the following form

$$\boldsymbol{\alpha}_w^{\mathsf{T}}\boldsymbol{\psi}(\mathbf{O}) = \boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O}, w) \tag{3.7}$$

where $\boldsymbol{\phi}(\mathbf{O}, w)$ is a sparse *joint* feature vector described the relationship between observations and word labels,

$$\boldsymbol{\phi}(\mathbf{O}, w) = \begin{bmatrix} \vdots \\ \delta(w, \texttt{dog})\ \boldsymbol{\psi}(\mathbf{O}) \\ \delta(w, \texttt{cat})\ \boldsymbol{\psi}(\mathbf{O}) \\ \vdots \end{bmatrix}. \tag{3.8}$$

The graphical representation of logistic regression models is shown in Figure 3.3.

As discussed earlier, to apply discriminative models for speech recognition three important decisions need to be made: the form of the features to use; the appropriate

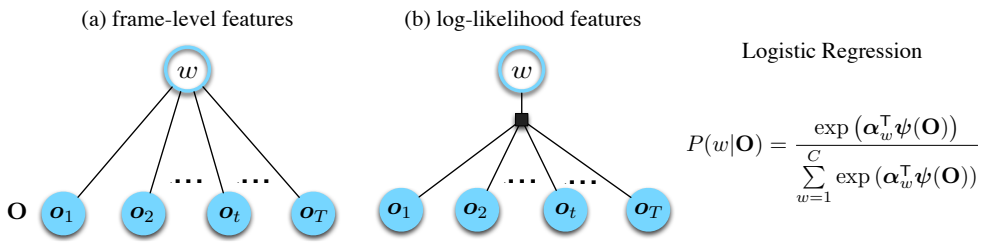


Figure 3.3 *The graphical model for logistic regression with frame-level feature and log-likelihood feature. The form of these features are discussed in Section 3.1.1.*

training criterion; and how to handle the structure in continuous speech. Two examples of features $\boldsymbol{\psi}(\cdot)$ are introduced in the next Section 3.1.1. The standard way to estimate the parameters of logistic regression model, $\boldsymbol{\alpha}$, is to *maximise* the probability of the observed labels,

$$\mathcal{F}_{\texttt{cml}}(\boldsymbol{\alpha}|\mathcal{D}) = \sum_{i=1}^{R}\log(P(w_i|\mathbf{O}_i;\boldsymbol{\alpha})) \tag{3.9}$$

where the probability $P(w_i|\mathbf{O}_i;\boldsymbol{\alpha})$ is defined in equation (3.5). $\mathbf{O}_i$ is the i-th observation sequence in the training set $\mathcal{D}$ and w_i is the corresponding class label. This objective function is related to the maximum mutual information (MMI) objective function for HMMs discussed in Section 2.3.2. In the context of discriminative models it is usually called *conditional maximum likelihood* (CML) criterion. More options for discriminative parameter estimation will be discussed in Section 4.4. The logistic regression model assumes no structure in the class label, thus it can be directly applied to the tasks such as isolated word classification (Birkenes *et al.* 2006), where f_w can be viewed as a discriminant function for word w.

3.1.1 Feature Functions

The form of the feature-function is central to the performance of discriminative models. For speech recognition, a wide range of feature-functions at the frame, model and word level (Gunawardana *et al.* 2005; Morris and Fosler-Lussier 2008; Ragni and Gales 2011*b*; Zhang *et al.* 2010) have been proposed. A fundamental requirement of feature

function is that they can map variable length sequences $\mathbf{O}$ into a fixed-dimensional vector $\psi(\mathbf{O})$. The nonlinear map $\boldsymbol{\psi}(\cdot)$ should preserve the discriminative information embedded in the speech signals. A simple example of frame-level feature-function is based on equation (3.4) in the last section,

$$\boldsymbol{\psi}(\mathbf{O}) = \begin{bmatrix} 1 \\ \sum_{t=1}^{T} \boldsymbol{o}_t \end{bmatrix} \tag{3.10}$$

where 1 is included to allow for bias. (Birkenes *et al.* 2006) proposed a mapping involving a set of HMMs for the words,

$$\boldsymbol{\psi}(\mathbf{O}) = \begin{bmatrix} \vdots \\ \log p_{\boldsymbol{\lambda}}(\mathbf{O}|w = \texttt{dog}) \\ \log p_{\boldsymbol{\lambda}}(\mathbf{O}|w = \texttt{cat}) \\ \vdots \end{bmatrix}. \tag{3.11}$$

where $p_{\boldsymbol{\lambda}}(\mathbf{O}|w = \texttt{dog})$ is the log-likelihood probability of HMM for word "dog", and $\boldsymbol{\lambda}$ is the HMM parameters. This feature is also known as *log-likelihood feature* (for details see Chapter 5). The log-likelihoods from all models are concatenated, including the correct model and competing ones, to yield additional information from the observations. More examples of features for speech recognition will be described in Chapter 5.

3.2 Support Vector Machines

Support Vector Machines (SVMs) (Cortes and Vapnik 1995; Vapnik 1995) are a popular discriminative classifier described extensively in the literature. They have been successfully applied to many different applications, such as text categorization (Burges 1998), speaker verification (Campbell *et al.* 2006), image classification (Chapelle *et al.* 1999), and bioinformatics (Bahlmann *et al.* 2002). SVMs are based on the intuitive concept of maximising the margin of separation between two competing classes, where the margin is defined as the distance between the decision hyperplane and the

closest training examples. This has been shown to be related to minimising an upper bound on the generalisation error (Vapnik 1995).

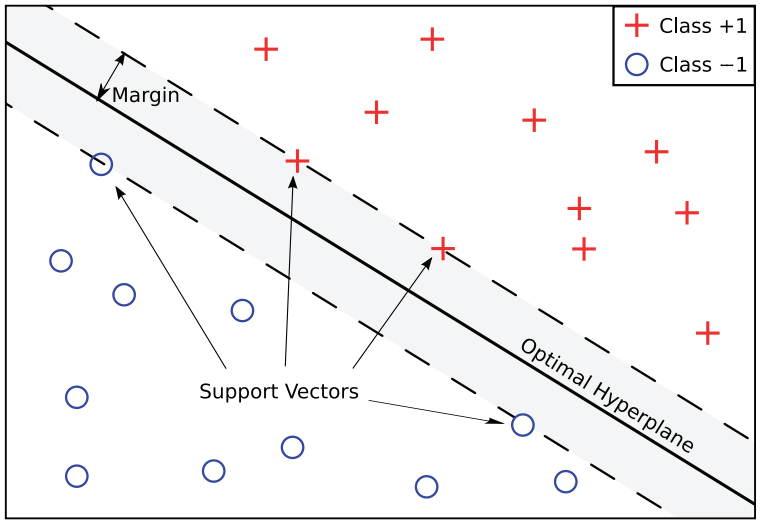


Figure 3.4 *Maximum margin separation for simple classification task*

Consider supervised training data $\mathcal{O} = \{\boldsymbol{o}_1, \boldsymbol{o}_2, \ldots, \boldsymbol{o}_R\}$, $\boldsymbol{o}_i \in \mathbb{R}^d$ with class labels $\{y_1, y_2, \ldots, y_R\}$, where $y_i \in \{-1, 1\}$. When the training data is linearly separable, SVMs can estimate a linear decision boundary such that all examples in the training set are correctly classified. Given this boundary (hyperplane), defined by weight (direction) vector $\boldsymbol{\alpha}$ and bias b, a test example $\boldsymbol{o}$ may be classified according to

$$y = \text{sign}\left(\boldsymbol{\alpha}^\mathsf{T}\boldsymbol{o} + b\right) \tag{3.12}$$

Note that this function is invariant under a positive rescaling of the parameters $\boldsymbol{\alpha}$ and b, parameter scaling must be fixed in order to obtain a unique solution from training. This is typically achieved by defining two canonical hyperplanes on both sides of the decision boundary,

$$\boldsymbol{\alpha}^\mathsf{T}\boldsymbol{o} + b = +1$$
$$\text{and } \boldsymbol{\alpha}^\mathsf{T}\boldsymbol{o} + b = -1$$

Training examples are then constrained to lie outside the region enclosed by the margin hyperplanes. This is shown in Figure 3.4 for the simple case of two-dimensional

data. The decision boundary is shown by a solid line. The margin hyperplanes are shown by two dashed lines. Training samples 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 boundary.

The shortest distance between the optimal/decision boundary and the margin hyperplanes is known as the margin. Using the definition of the canonical hyperplanes, the size of this margin can be calculated using the following expression

$$\text{margin} = \frac{1}{||\boldsymbol{\alpha}||} \tag{3.13}$$

Statistical learning theory states that the decision boundary that minimises the probability of generalisation error[1] is the one that *maximises this margin* (N. Cristianini and J. Shawe-Taylor 2000; Vapnik 1995). Since SVMs are designed to minimise the generalisation error, the SVM objective function for linearly separable data is given by

$$\begin{aligned} \min_{\boldsymbol{\alpha},b} \quad & \frac{1}{2}||\boldsymbol{\alpha}||^2 \\ \text{s.t.} \quad & y_i \left(\boldsymbol{\alpha}^\mathsf{T}\boldsymbol{o}_i + b\right) \geq 1 \qquad \forall i \end{aligned} \tag{3.14}$$

Unfortunately, in practice it is often not possible to find a linear boundary that correctly separates all training examples. To enable SVM training to converge for such data, the margin constraints, $y_i \left(\boldsymbol{\alpha}^\mathsf{T}\boldsymbol{o}_i + b\right) \geq 1$ 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 $y_i \left(\boldsymbol{\alpha}^\mathsf{T}\boldsymbol{o}_i + b\right) \geq 1 - \xi_i$, where the slack variables, $\xi_i \geq 0$, 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 penalized. The resulting objective function and

[1] This is also known as structural risk minimisation (Vapnik 1995).

constraints become

$$\begin{aligned} \min_{\boldsymbol{\alpha},b} \quad & \frac{1}{2}||\boldsymbol{\alpha}||^2 + C\sum_{i=1}^{R}\xi_i & \\ \text{s.t.} \quad & y_i\left(\boldsymbol{\alpha}^\mathsf{T}\boldsymbol{o}_i + b\right) \geq 1 - \xi_i & \forall\, i \\ & \xi_i \geq 0 & \forall\, i \end{aligned} \tag{3.15}$$

where ξ_i is the slack variables that accounts for linearly inseparable training examples. The term $\sum_{i=1}^{R}\xi_i$ in equation (3.15) gives the upper bound on the training classification error. The constant C allows to trade-off margin maximisation to training classification error.

Although this soft-margin SVM can be optimised directly in its primal form (equation (3.15)), it is often easier to consider the dual form of the objective (Vapnik 1995)[2]. The dual objective function is defined by

$$\begin{aligned} \max_{\boldsymbol{\alpha}^{\texttt{dual}}} \quad & \sum_{i=1}^{R}\alpha_i^{\texttt{dual}} - \frac{1}{2}\sum_{i=1}^{R}\sum_{j=1}^{R}\alpha_i^{\texttt{dual}}\alpha_j^{\texttt{dual}}y_iy_j\boldsymbol{o}_i^\mathsf{T}\boldsymbol{o}_j & \\ \text{s.t.} \quad & \sum_{i=1}^{R}\alpha_i^{\texttt{dual}}y_i = 0 & \\ & 0 \leq \alpha_i^{\texttt{dual}} \leq C & \forall\, i \end{aligned} \tag{3.16}$$

where $\boldsymbol{\alpha}^{\texttt{dual}}$ are the dual variables (Lagrange multipliers) of $\boldsymbol{\alpha}$. The upper bound, C, on the dual variables limits the impact of individual examples, and is typically selected using either a development set or a data-dependent algorithm such as (Joachims 1999). The dual objective function in (3.16) is convex (quadratic for the dual variables). Thus the optimisation can converge to a single global solution. Many algorithms have been proposed for training SVMs, two of the most popular are sequential minimal optimisation (Platt 1999) and the decomposition and chunking algorithms in (Joachims 1999).

[2] If the dimension of the feature space is larger than the number of training examples, it becomes more efficient to solve equation (3.16) rather than the primal equation. This will be explained in next section.

One important property of SVM is that the dual variables $\alpha_i^{\texttt{dual}}$ are only non-zero for a limited set of training examples. During optimization, the Karush-Kuhn-Tucker (KKT) conditions (Vapnik 1995) ensure that only examples that lie either on the margin, $y_i\left(\boldsymbol{\alpha}^\mathsf{T}\boldsymbol{o}_i + b\right) = 1$, or on the wrong side of the margin, $y_i\left(\boldsymbol{\alpha}^\mathsf{T}\boldsymbol{o}_i + b\right) \leq 1$, have non-zero dual variables, $\alpha_i^{\texttt{dual}} > 0$. These examples are known as the *support vectors* (Vapnik 1995). This gives a sparse representation to SVMs. Using Lagrangian theory, the weight vector, $\boldsymbol{\alpha}$, and bias, b, in the primal form (3.15) can be obtained using only the support vectors,

$$\boldsymbol{\alpha} = \sum_{i=1}^{R} \alpha_i^{\texttt{dual}} y_i \boldsymbol{o}_i \tag{3.17}$$

The weight vector $\boldsymbol{\alpha}$ is a linear combination of the support vectors. To determine the value of bias b, select a correctly classified training example that lie on the margin. In practice a more accurate estimate is found from averaging all such values.

3.2.1 Kernelization

In the previous section describes SVMs, only linear decision hyperplane have been considered. In order to extend the above approach to nonlinear decision boundary, a nonlinear mapping $\boldsymbol{\psi}(\boldsymbol{o})$ is introduced. This mapping transforms the data from the observation-space to a feature-space of higher dimensionality. Linear decision boundaries are found in this high dimensional feature-space, which correspond to non-linear decision boundaries in the original observation-space as illustrated in Figure 3.5. Thus the SVM dual objective function in (3.16) becomes

$$\max_{\boldsymbol{\alpha}^{\texttt{dual}}} \sum_{i=1}^{R} \alpha_i^{\texttt{dual}} - \frac{1}{2}\sum_{i=1}^{R}\sum_{j=1}^{R} \alpha_i^{\texttt{dual}}\alpha_j^{\texttt{dual}} y_i y_j \boldsymbol{\psi}(\boldsymbol{o}_i)^\mathsf{T}\boldsymbol{\psi}(\boldsymbol{o}_j) \tag{3.18}$$

subject to the same constraints in equation (3.16). Note that equation (3.18) is only a function of the distance between feature-space points. In practice, computing the high-dimensional feature-space explicitly may be inefficient. Instead, SVMs can be written in terms of a kernel function (Boser *et al.* 1992; Vapnik 1995)

$$k(\boldsymbol{o}_i, \boldsymbol{o}_j) = \boldsymbol{\psi}(\boldsymbol{o}_i)^\mathsf{T}\boldsymbol{\psi}(\boldsymbol{o}_j) \tag{3.19}$$

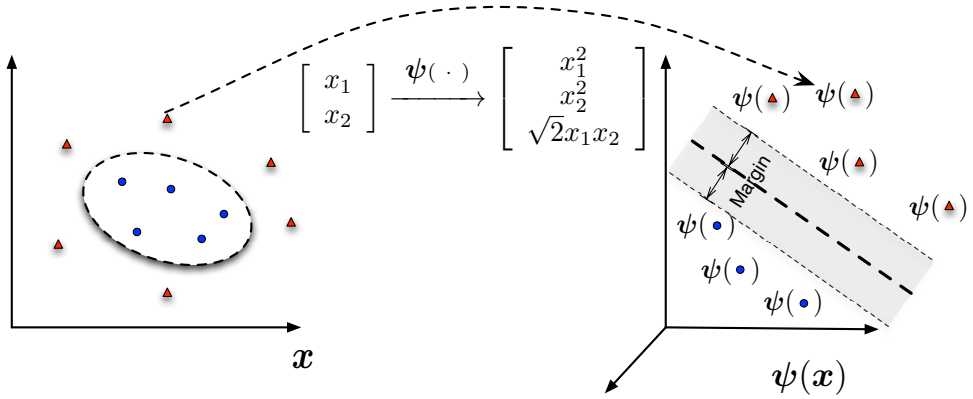


Figure 3.5 *An example of feature mapping $\boldsymbol{\psi}(\cdot)$ (associated with polynomial kernels) and dicesion boundaries.*

In general, this function may be any symmetric nonlinear function that satisfies Mercer's condition (Shawe-Taylor and Cristianini 2004; Vapnik 1995). Figure 3.5 shows an example of decision boundaries with (second order) polynomial kernels. Thus, the kernelized objective function of SVMs can be expressed as

$$\max_{\boldsymbol{\alpha}^{\texttt{dual}}} \sum_{i=1}^{R} \alpha_i^{\texttt{dual}} - \frac{1}{2} \sum_{i=1}^{R} \sum_{j=1}^{R} \alpha_i^{\texttt{dual}} \alpha_j^{\texttt{dual}} y_i y_j k(\boldsymbol{o}_i, \boldsymbol{o}_j) \tag{3.20}$$

The classification function of SVMs in equation (3.12) can also be represented based on kernels

$$y = \text{sign}\left(\sum_{i=1}^{R} \alpha_i^{\texttt{dual}} y_i k(\boldsymbol{o}, \boldsymbol{o}_i) + b \right). \tag{3.21}$$

Kernel Trick One advantage of expressing SVM training in terms of a kernel function is, for some feature-spaces, the feature-space mapping and inner-product operations can be combined into a single efficient calculation. It is not necessary to explicitly operate in the high-dimensional feature-space. This is especially important when the feature-space has a much higher dimensionality than the observation-space. Here significant computational savings can be achieved by using the kernel functions. For example, consider a two-dimensional input space $\boldsymbol{o} \in \mathbb{R}^2$ together with the feature

map

$$\boldsymbol{\psi} : \boldsymbol{o} = \begin{bmatrix} o_1 \\ o_2 \end{bmatrix} \in \mathbb{R}^2 \longmapsto \boldsymbol{\psi}(\boldsymbol{o}) = \begin{bmatrix} o_1^2 \\ o_2^2 \\ \sqrt{2}o_1 o_2 \end{bmatrix} \in \mathbb{R}^3 \tag{3.22}$$

The inner product in the feature space can be evaluated as follows

$$\begin{aligned} \boldsymbol{\psi}(\boldsymbol{o}_i)^\mathsf{T}\boldsymbol{\psi}(\boldsymbol{o}_j) &= \begin{bmatrix} o_{i1}^2 \\ o_{i2}^2 \\ \sqrt{2}o_{i1}o_{i2} \end{bmatrix}^\mathsf{T} \begin{bmatrix} o_{j1}^2 \\ o_{j2}^2 \\ \sqrt{2}o_{j1}o_{j2} \end{bmatrix} \\ &= (o_{i1}o_{j1} + o_{i2}o_{j2})^2 = \left(\boldsymbol{o}_i^\mathsf{T}\boldsymbol{o}_j\right)^2 = k(\boldsymbol{o}_i, \boldsymbol{o}_j) \end{aligned} \tag{3.23}$$

This equation illustrates that the inner product can also be computed implicitly through the kernel function $k(\boldsymbol{o}_i, \boldsymbol{o}_j) = \left(\boldsymbol{o}_i^\mathsf{T}\boldsymbol{o}_j\right)^2$ more efficiently. This technique is commonly referred as *kernel trick* (Aizerman *et al.* 1964).

Modularity According to equation (3.20), only the kernel values (instead of the features) are required to train SVM parameters. The kernel values of all the training data can be stored into a matrix $\mathbf{G}$, known as the *Gram matrix*,

$$\mathbf{G} = \begin{bmatrix} g_{11} & \cdots & g_{1R} \\ \vdots & \ddots & \vdots \\ g_{R1} & \cdots & g_{RR} \end{bmatrix}, \text{ where } g_{ij} = k(\boldsymbol{o}_i, \boldsymbol{o}_j). \tag{3.24}$$

The Gram matrix have dimensions $R \times R$, where R is the number of training samples. Thus the interaction between the training data and the learning algorithm is via the Gram matrix. This is illustrated in Figure 3.6. This modularity allows developing general learning algorithms and designing suitable kernels for specific problems independently. The same algorithm will work with any kernel and hence for data in any domain. The form of the kernel is data specific, but can be combined with different algorithms to solve a wide range of tasks (Hofmann *et al.* 2008; Shawe-Taylor and Cristianini 2004). All this leads to a very natural and elegant way to design learning systems, where modules are combined together to achieve complex learning systems

as shown in Figure 3.6. The data is processed using a kernel function to create a Gram matrix, followed by a learning algorithm to produce a classification function. This function is used to process unseen examples.

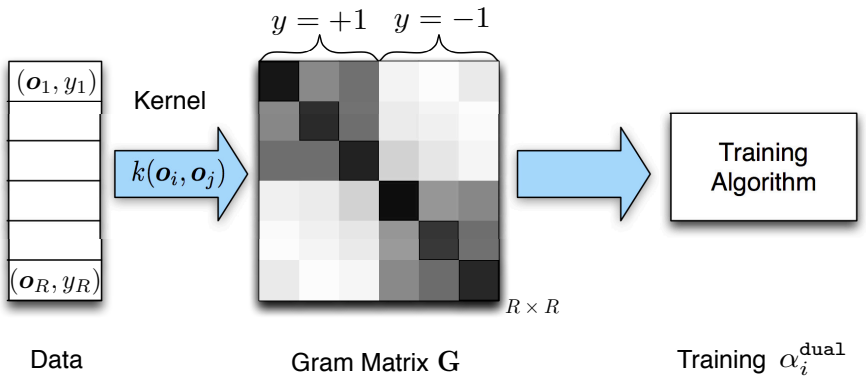


Figure 3.6 *Kernel methods offer a modular framework. In a first step, a dataset is processed into a Gram matrix. Data can be of various types. In a second step, a variety of learning algorithms can be used to analyze the data, using only the information contained in the Gram matrix.*

Form of Kernels Many different kernel functions have been proposed for mapping observations into a high-dimensional feature-space. A small selection of typically used kernels are given in Table 3.1. Some kernels—such as polynomial kernels—have a fixed dimensional feature-space, some—such as Laplacian and Radial Basis Function (RBF) kernels—have feature-spaces with an infinite number of dimensions (Shawe-Taylor and Cristianini 2004).

Kernel	Function Form	Kernel Parameters
Linear	$\boldsymbol{o}_i^\mathsf{T} \boldsymbol{o}_j$	-
Polynomial	$(\boldsymbol{o}_i^\mathsf{T} \boldsymbol{o}_j + c)^d$	order d, offset $c \geq 0$
Laplacian	$\exp\left(-\frac{\|\|\boldsymbol{o}_i - \boldsymbol{o}_j\|\|}{\sigma}\right)$	width $\sigma > 0$
RBF	$\exp\left(-\frac{\|\|\boldsymbol{o}_i - \boldsymbol{o}_j\|\|^2}{2\sigma^2}\right)$	width $\sigma > 0$

Table 3.1 *Commonly used static kernel functions*

3.2.2 Sequence Kernels

So far it has been assumed that the data has fixed-length feature vectors. As discussed earlier, in speech recognition the observation sequences typically have a variable length, $\mathbf{O} = \{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_T\}$. Several solutions have been proposed to enable SVMs to be applied to classification of variable-length data. These include *subsampling* (Ganapathiraju *et al.* 2000) and *sequence kernels* (Jaakkola and Haussler 1999; Smith and Gales 2002*b*). This section focuses on the sequence kernels. To distinguish these from the *static* kernels $k(\boldsymbol{o}_i, \boldsymbol{o}_j)$ introduced in previous section (in Table 3.1) for fixed-dimensional data, these kernels for variable-length data will be referred to as *dynamic* or *sequence* kernels and denoted as $K(\mathbf{O}_i, \mathbf{O}_j)$. These sequence kernels can map variable-length sequences into a fixed-dimensional feature space where an inner product can be calculated (Longworth 2010; Smith and Gales 2002*a*). Given a pair of observation sequences, $\mathbf{O}_i$ and $\mathbf{O}_j$, the sequence kernel may be expressed by

$$K(\mathbf{O}_i, \mathbf{O}_j) = \boldsymbol{\psi}(\mathbf{O}_i)^{\mathsf{T}} \boldsymbol{\Sigma}^{-1} \boldsymbol{\psi}(\mathbf{O}_j) \tag{3.25}$$

where $\boldsymbol{\psi}(\cdot)$ are feature functions, $\boldsymbol{\Sigma}^{-1}$ is a *metric* defines the distance in feature space (Jaakkola and Haussler 1999). Two simple examples of feature functions were given in equations (3.10) and (3.11) in Section 3.1.1. Additional examples of feature functions will be discussed in Chapter 5. Various forms of metric $\boldsymbol{\Sigma}$ have been investigated for many tasks (Campbell *et al.* 2006; Lodhi 2002; Zhang and Mak 2011). As SVMs are sensitive to data scaling (Shivaswamy and Jebara 2009), the following empirical covariance matrix is commonly used to normalize the data (Shawe-Taylor and Cristianini 2004)

$$\boldsymbol{\Sigma} = \frac{1}{R} \sum_{i=i}^{R} \left(\boldsymbol{\psi}(\mathbf{O}_i) - \boldsymbol{\mu}\right) \left(\boldsymbol{\psi}(\mathbf{O}_i) - \boldsymbol{\mu}\right)^{\mathsf{T}} \tag{3.26}$$

where $\boldsymbol{\mu} = \frac{1}{R} \sum_{i=1}^{R} \boldsymbol{\psi}(\mathbf{O}_i)$ and $\{\mathbf{O}_i\}_{i=1}^{R}$ are the training observation sequences. For high-dimensional score-spaces computing and storing $\boldsymbol{\Sigma}$ based on equation (3.26) can be computationally expensive (Layton 2006). To address this issue, further approximations may be applied, such as diagonal approximation $\mathrm{diag}(\boldsymbol{\Sigma})$ (Smith and

Gales 2002*a*). This provides a reasonable approximation to equation (3.26) while reducing the computational cost associated with inverting a full matrix to inverting a diagonal matrix.

Given a sequence kernel, SVMs can be applied to variable-length sequence data by solving the dual optimisation problem,

$$\max_{\boldsymbol{\alpha}^{\texttt{dual}}} \sum_{i=1}^{R} \alpha_i^{\texttt{dual}} - \frac{1}{2} \sum_{i=1}^{R} \sum_{j=1}^{R} \alpha_i^{\texttt{dual}} \alpha_j^{\texttt{dual}} y_i y_j K(\mathbf{O}_i, \mathbf{O}_j) \tag{3.27}$$

subject to the same constraints in equation 3.16. The classification function of these SVMs has the following form (Jaakkola and Haussler 1999)

$$y = \text{sign}\left(\sum_{i=1}^{R} \alpha_i^{\texttt{dual}} y_i K(\mathbf{O}, \mathbf{O}_i) + b \right). \tag{3.28}$$

3.2.3 *Relation to Logistic Regression*

In the previous sections, logistic regression and SVMs were introduced. Logistic regression can directly model the posterior probability of a class label. Although SVMs are introduced to model decision boundary between classes, it has been shown that the SVMs can also be related to probabilistic models (Grandvalet *et al.* 2006; Sollich 2002; Zhang *et al.* 2003). In the following we will illustrate that SVMs can be closely related to logistic regression models. Consider the logistic regression for binary classification described in equations (3.2) and (3.3). The posterior of a binary class label y given $\mathbf{O}$ can be written as,

$$P(y|\mathbf{O}) = \frac{1}{1 + e^{-y\, \boldsymbol{\alpha}^{\mathsf{T}} \boldsymbol{\psi}(\mathbf{O})}}, \quad y \in \{+1, -1\} \tag{3.29}$$

Given a training set $\{\mathbf{O}_i, y_i\}_{i=1}^{R}$, substituting equation (3.29) into criterion (3.9), the CML training of logistic regression with regularization $||\boldsymbol{\alpha}||$ can be achieved by *minimising*

$$\mathcal{F}_{\texttt{cml}}(\boldsymbol{\alpha}) = \underbrace{\frac{1}{2}||\boldsymbol{\alpha}||^2}_{\text{regularization}} + C \underbrace{\sum_{i=1}^{R} \log\left\{1 + e^{-y_i\, \boldsymbol{\alpha}^{\mathsf{T}} \boldsymbol{\psi}(\mathbf{O}_i)}\right\}}_{\text{logistic empirical risk}} \tag{3.30}$$

where $\frac{1}{2}||\boldsymbol{\alpha}||^2$ is the regularization term, $\log\left\{1+e^{-y_i\,\boldsymbol{\alpha}^\mathsf{T}\boldsymbol{\psi}(\mathbf{O}_i)}\right\}$ is a form of empirical risk (Zhang *et al.* 2003) and C is the balance parameter. For SVMs, substituting the constrains of equation (3.15) into the objective function, the SVM training can be rewritten as

$$\mathcal{F}_{\texttt{svm}}(\boldsymbol{\alpha}) = \frac{1}{2}||\boldsymbol{\alpha}||^2 + C\underbrace{\sum_{i=1}^{R}\max\left\{0, 1-y_i\boldsymbol{\alpha}^\mathsf{T}\boldsymbol{\psi}(\mathbf{O}_i)\right\}}_{L_1 \text{ empirical risk}} \tag{3.31}$$

where $\max\left\{0, 1-y_i\boldsymbol{\alpha}^\mathsf{T}\boldsymbol{\psi}(\mathbf{O}_i)\right\}$ is the ξ_i in equation (3.15). In (Zhang *et al.* 2003) the empirical risk $\log(1+e^{-z})$ in equation (3.30) has been considered as an approximation to $\max\{0, 1-z\}$ in (3.31), where $z = y_i\,\boldsymbol{\alpha}^\mathsf{T}\boldsymbol{\psi}(\mathbf{O}_i)$. This can be illustrated in Figure 3.7. Thus the CML trained logistic regression can be related to SVMs (Girosi 1998).

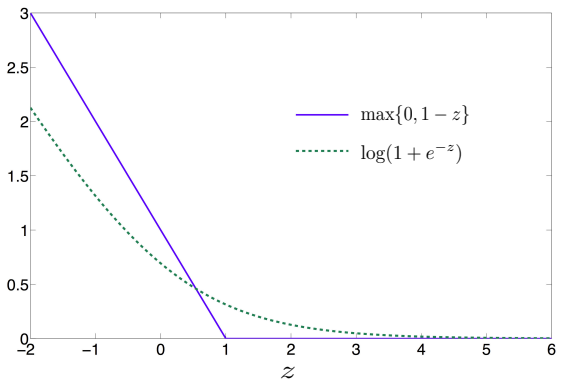


Figure 3.7 *A comparison between the empirical risks of L_1-SVMs (blue line) and logistic regressions (green dash).*

More interestingly, if logistic regression models are trained using large margin criteria it becomes equivalent to SVMs. To see this, the margin for logistic regression is defined as the distance between log-posteriors of correct classes y_i and incorrect class $\bar{y}_i$ (e.g., $y_i = 1$ and $\bar{y}_i = -1$). Thus the large margin training of logistic regression can be written as *minimising*

$$\mathcal{F}_{\texttt{lm}}(\boldsymbol{\alpha}) = \frac{1}{2}||\boldsymbol{\alpha}||^2 + C\sum_{i=1}^{R}\left[\mathcal{L}(y_i, \bar{y}_i) - \log\frac{P(y_i|\mathbf{O}_i)}{P(\bar{y}_i|\mathbf{O}_i)}\right]_+ \tag{3.32}$$

where $\log \frac{P(y_i|\mathbf{O}_i)}{P(\bar{y}_i|\mathbf{O}_i)}$ is the margin and the loss function $\mathcal{L}(y_i, \bar{y}_i)$ is introduced to enforce the margin to be greater than the loss. The hinge function $[\,\cdot\,]_+$ is introduced to ignore the data that beyond the margin and already classified correctly. Note that this equation is also similar to the large margin training of HMM in equation (2.36). Substituting equation (3.29) into (3.32):

$$\mathcal{F}_{\texttt{lm}}(\boldsymbol{\alpha}) = \frac{1}{2}||\boldsymbol{\alpha}||^2 + C\sum_{i=1}^{R}\left[\mathcal{L}(y_i, \bar{y}_i) - \log\frac{1 + e^{-\bar{y}_i\,\boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\psi}(\mathbf{O})}}{1 + e^{-y_i\,\boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\psi}(\mathbf{O})}})\right]_+ \tag{3.33}$$

For binary classification, the incorrect class is the negative of the correct class $\bar{y}_i = -y_i$, and the loss function $\mathcal{L}(y_i, \bar{y}_i)$ can always be $|y_i - \bar{y}_i|/2 = 1$. Thus equation (3.33) can be expressed as

$$\mathcal{F}_{\texttt{lm}}(\boldsymbol{\alpha}) = \frac{1}{2}||\boldsymbol{\alpha}||^2 + C\sum_{i=1}^{R}\Big[1 - y_i\boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\psi}(\mathbf{O}_i)\Big]_+ \tag{3.34}$$

This is actually the training criterion of SVMs in equation (3.31). Thus SVMs can be viewed as large margin trained logistic regression models.

3.3 Multi-Class SVMs

In the previous section SVM has been introduced as a binary classifier. Approaches that extend binary SVMs to multi-class classification can be divided into two groups. The first group—"indirect" approach is to break the problem down into a series of binary classification tasks. This process is known as a *reduction* (Bottou *et al.* 1994). A selection of reduction schemes are presented in Section 3.3.1. The second group—"direct" approaches, such as Multi-class SVM[3] (Crammer and Singer 2001), modified the SVM training and classification to support the multi-class problems are also presented in the Section 3.3.2.

[3]This section focuses on unstructured data, e.g., data in isolated word recognition. Another form of extension to SVMs, structured SVMs, for continuous speech recognition will be described in Chpater 6.

3.3.1 Combining Binary SVMs

In this section the first group of approaches is used to combine binary SVMs for multi-class classifications. These are reduction-style techniques which reduce the multi-class problem to a series of binary classification tasks. Two forms of classifier will be examined: *majority voting* and *tree-based classifiers*[4].

Majority Voting Various approaches exist for using simple voting schemes with the SVM. The simplest approach is to train a one-versus-the-rest classifier. However, in (Gautier 2008) it was found to yield poor performance in a speech recognition task. The alternative approach used in this work is to train a one-versus-one SVM classifier. Consider isolated word recognition tasks, given observation sequence $\mathbf{O}$ the following procedure can be performed (Gales *et al.* 2009):

1. For each word pair $\{w_l, w_j\}$ in the vocabulary
   (a) Classify: apply equation (3.28) to obtain the word label $w^{\texttt{svm}} \in \{w_l, w_j\}$
   (b) Voting: $\text{count}[w^{\texttt{svm}}] = \text{count}[w^{\texttt{svm}}] + 1$
2. Classification result, $w^{\texttt{vote}}$, for observation sequence $\mathbf{O}$ is given by:
   (a) If no ties in voting then $w^{\texttt{vote}} = \arg\max_{w^{\texttt{svm}}} \{\text{count}[w^{\texttt{svm}}]\}$
   (b) If only two words tie then $w^{\texttt{vote}}$ is determined using the result of classification in equation (3.28).
   (c) If more words tie then $w^{\texttt{vote}}$ can be determined using HMMs.

Tree-based Approaches Tree-based reduction methods are popular in pattern recognition literature (Beygelzimer *et al.* 2007; Kijsirikul and Abe 2002; Platt *et al.* 2000; Vural and Dy 2004). The basic process consists of converting the multi-class classification problem into sequence of binary tasks. The appropriate sequence of tasks is encoded into a binary tree. Figure 3.8 gives an example for a 4-class problem. There

[4] Note the *acoustic code-breaking* in (Venkataramani *et al.* 2003) may also be viewed as a reduction approach. Applying it to continuous speech recognition will be discussed in Section 3.4.

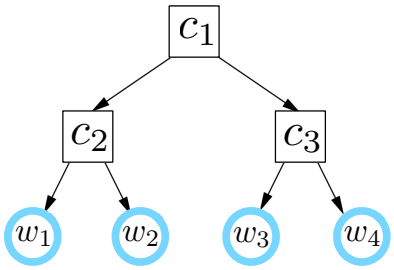


Figure 3.8 *Tree-based classifier for 4-class problem. Square boxes denote binary classifiers. Circles denote classes.*

are a number of options available for training and classification based on these binary tree classifiers. In this section three schemes will be considered: Directed Acyclic Graph (DAG) (Platt *et al.* 2000), Filter Tree (FT) (Beygelzimer *et al.* 2007) and Adaptive Directed Acyclic Graph (ADAG) (Kijsirikul and Abe 2002). All schemes have in common the way how the bottom layer classifiers (c_2 and c_3 in Figure 3.8) are trained. The difference lies in the way how the upper level classifier c_1 is trained. Additionally, classification is performed either top-down or bottom up. In the following each reduction scheme is described in more detail.

Directed acyclic graph (Platt *et al.* 2000): the upper-level classifier c_1 in Figure 3.8 is trained to classify $\{w_1, w_2\}$-vs-$\{w_3, w_4\}$. Examples belonging to classes w_1 and w_2 are taken as positive and examples from classes w_3 and w_4 as negative. Classification with the DAG is performed by starting at the top-level classifier c_1 and proceeding to the bottom following each binary decision.

Filter tree (Beygelzimer *et al.* 2007): improves the error characteristics of the DAG classifier. The key idea consists of training the upper level classifier c_1 using data that can be correctly classified by the lower level classifiers c_2 and c_3. Misclassified training examples are ignored. Therefore, training of the FT proceeds bottom-up. Classification is performed top-down in the same fashion as the DAG approach. It has been experimentally confirmed to yield lower error rates than the DAG approach (Aldamarki 2009; Gales *et al.* 2009).

Adaptive directed acyclic graph (Kijsirikul and Abe 2002): most of the errors with the DAG and FT usually results from the incorrect decisions made by the upper level classifier c_1 (Gautier 2008). The ADAG approach addresses this problem using the reversed classification and dynamic classifier selection. The classification proceeds from the bottom-up. All bottom layer classifiers are applied to the observation sequence $\mathbf{O}$. Let c_2 classify it as belonging to w_2 and c_3 as belonging to w_3. The upper level classifier c_1 in the ADAG approach is not fixed but changes dynamically according to decisions made by the lower level classifiers c_2 and c_3. In this example c_1 would use $\{w_2\}$-vs-$\{w_3\}$ classifier to make the final classification decision. The adaptive nature of the ADAG approach, however, requires that all possible pairwise classifiers are available during classification.

Table 3.2 provides details on the number of classifiers and binary classifications required for a general C-class reduction using the tree-based schemes. As a reference the first line gives the same characteristics for the majority voting described in previous section. The DAG and FT approaches allow to use the smallest number of classifiers ($C-1$) and perform the smallest number of classifications ($\log_2(C)$). These approaches however were found to be less accurate in practice as compared to the majority voting and ADAG (Aldamarki 2009; Gales *et al.* 2009).

Scheme	**number of SVMs**	**number of classifications**
Voting	$C(C-1)/2$	$C(C-1)/2$
DAG	$C-1$	$\log_2(C)$
FT	$C-1$	$\log_2(C)$
ADAG	$C(C-1)/2$	$C-1$

Table 3.2 *Number of SVMs and binary classifications required during classification with majority voting, directed acyclic graph (DAG), filter tree (FT) and adaptive directed acyclic graph (ADAG) reduction approaches.*

3.3.2 *Multi-Class SVMs*

Previously "indirect" ways to do multi-class classification with SVMs were described. There approaches break the classification problem down into a series of binary decision problems, thus they may require training $C(C-1)/2$ classifiers for C classes. An elegant alternative is to use "direct" approaches, for example Multi-class SVM, which attempt to find discriminant functions for all classes simultaneously by maximizing margins among classes (Crammer and Singer 2001; Weston and Watkins 1999).

Consider each class (e.g., word) w is parameterized with a weight vector $\boldsymbol{\alpha}_w$ that can be used to compute a discriminant score for observation sequence $\mathbf{O}$. The score is the inner product $\boldsymbol{\alpha}_w^{\mathsf{T}}\boldsymbol{\psi}(\mathbf{O})$, which can be also interpreted as a negated distance between the data point in feature space and the hyperplane whose normal vector is $\boldsymbol{\alpha}_w$. The decision rule is thus given by,

$$\hat{w} = \arg\max_w \boldsymbol{\alpha}_w^{\mathsf{T}}\boldsymbol{\psi}(\mathbf{O}) \tag{3.35}$$

The training goal is to make sure the score for the correct class is greater than the scores for the incorrect classes as much as possible. One such formulation is given in (Crammer and Singer 2001),

$$\begin{aligned} \min_{\boldsymbol{\alpha}_w} \quad & \frac{1}{2}\sum_w ||\boldsymbol{\alpha}_w||^2 + C\sum_{i=1}^{R}\xi_i \\ \text{s.t.} \quad & \text{For every training data } i = 1, \ldots, R, \\ & \quad \text{For every competing classes (words) } w \neq w_i : \\ & \quad\quad \boldsymbol{\alpha}_{w_i}^{\mathsf{T}}\boldsymbol{\psi}(\mathbf{O}_i) - \boldsymbol{\alpha}_w^{\mathsf{T}}\boldsymbol{\psi}(\mathbf{O}_i) \geq 1 - \xi_i \qquad \text{where } \xi_i \geq 0 \end{aligned} \tag{3.36}$$

where $\{(\mathbf{O}_i, w_i)\}_{i=1}^{R}$ is the training set and the value "1" on the right hand side of equation (3.36) is used to denote zero-one loss, namely, whether the data point is correctly classified or not.

Note that the objective function is essentially the same as the binary SVM. The only difference comes from the constraints, which essentially says that the score of the correct label $\boldsymbol{\alpha}_{w_i}^{\mathsf{T}}\boldsymbol{\psi}(\mathbf{O}_i)$ has to be greater than the score of any other classes $\boldsymbol{\alpha}_w^{\mathsf{T}}\boldsymbol{\psi}(\mathbf{O}_i)$,

so there are $C-1$ constraints in total where C is the total number of classes. There is one slack variable ξ_i for each example, shared among the $C-1$ constraints. In equation (3.36) the loss is a constant 1 for any misclassification. Other types of loss functions, for example, class-sensitive losses described in Section 2.3.4 can also be applied. Thus the extension of SVMs from binary to multi-class classification (e.g., isolated word recognition) is described. Extending multi-class SVMs to structured SVMs for continuous speech recognition will be discussed in Chapter 6.

3.4 Acoustic Code-Breaking

Previously three unstructured discriminative models—logistic regression, SVMs and multi-class SVMs were discussed in this chapter. In these models the output classes are considered as unstructured (atomic) labels. The parameters associated with different classes are assumed to be unshared. Although for some tasks such as isolated word recognition or speaker recognition these models can be applied (Birkenes *et al.* 2006; Keshet and Bengio 2008), for continuous speech recognition as the number of classes (sentences) increases such an approach quickly becomes impractical. There are several options to address this issue. One option is to use the structured discriminative models. This option will be discussed in Chapter 4. This chapter introduces an alternative which is to decompose the whole sentence recognition problem into a sequence of independent, typically, word classification sub-problems using *acoustic code-breaking* (ACB) schemes (Gales and Flego 2010; Venkataramani and Byrne 2005; Venkataramani *et al.* 2003). These sub-problems are then addressed by using unstructured models described in this chapter where parameters are associated with individual words.

Acoustic code-breaking is a rescoring approach to continuous speech recognition in which the whole-sentence recognition problem is transformed into multiple independent, word classification sub-problems (Venkataramani *et al.* 2003). This provides a general framework for incorporating models that may not be possible to apply to

continuous speech recognition tasks (Gales and Flego 2010; Venkataramani *et al.* 2003). For example, if the sub-problems are limited to word-pairs then the SVMs in Section 3.2 may be directly used. A range of acoustic code-breaking approaches have been proposed (Gales and Flego 2010; Layton 2006; Venkataramani *et al.* 2003). Common to these approaches is the use of HMM-based ASR systems to yield the 1-best hypothesis or word lattice. This section discusses two such approaches.

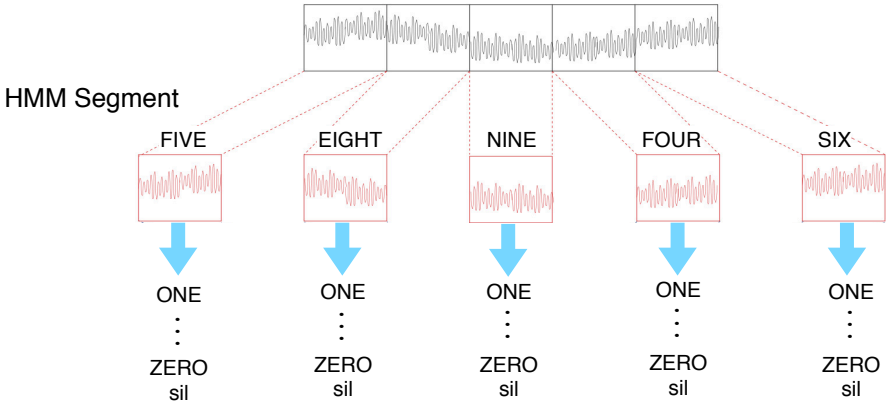
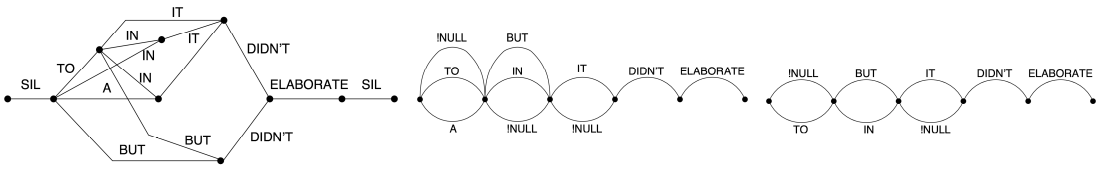


Figure 3.9 *1-best hypothesis based acoustic code-breaking: segment continuous speech for isolated word classification.*

Given a test utterance $\mathbf{O}$, the variant of acoustic code-breaking in (Gales and Flego 2010) makes use of existing HMM-based speech recognition system to produce a 1-best hypothesis with segmentation, as illustrated in Figure 3.9. Using the time stamp information provided by the HMM segmentation, the observation sequence is segmented into sub-sequences. Unstructured discriminative models can then be applied to classify each sub-sequence into one of the possible word label classes. This variant of acoustic code-breaking was applied to digit string recognition tasks (Gales and Flego 2010), where the vocabulary of words includes digits from one to nine, oh, zero and silence. Examples of unstructured discriminative models examined included SVMs (Gales and Flego 2010) where the multi-class decisions were made using the majority voting strategy described in Section 3.3.1, and multi-class SVMs (Zhang *et al.* 2010) described in Section 3.3.2.



(a) Word-level lattice (b) Confusion network (c) Pruned confusion network

Figure 3.10 *Confusion network based acoustic code-breaking.*

Alternatively, the variant of acoustic code-breaking in (Layton 2006), similar to the approaches in (Venkataramani *et al.* 2003), was proposed for binary, word-pair rescoring using SVMs and generative augmented models (Smith 2003). This approach is illustrated in Figure 3.10 and may be described in three steps. The first step makes use of an existing HMM-based speech recognition system to produce a word lattice, such as the one shown in Figure 3.10(a). Each arc in the lattice has a word label and time alignment data (not shown). As can be seen multiple hypotheses and segmentations are possible. A risk-based lattice cutting (Kumar and Byrne 2002) is performed in the second step to produce a *confusion network*, such as the one shown in Figure 3.10(b). As can be seen only one segmentation is retained in the confusion network. In addition to word label and time alignment data each arc has a posterior probability. The posterior probability is obtained in the lattice cutting procedure (Kumar and Byrne 2002). The confusion network is pruned in the third step to retain at most two and at least one arc for each segment as shown in Figure 3.10(c). The pruning is performed based on the posterior probabilities. The reduction is finished at this point. The next stage consists of rescoring the pruned confusion network. The SVMs are applied to each segment where the number of confusions is two. At the end there remains exactly one arc for each segment of the network. The final recognition hypothesis is obtained from the arc word labels. By default, in the acoustic code-breaking a common pool of SVM classifiers is maintained during rescoring. an alternative *ambitious* approach is proposed in (Venkataramani *et al.* 2003) where classifiers are trained on demand. In both cases SVMs are trained only for the most frequent confusions as determined from the training data. The number of classifiers is further bounded by the number

of available examples. For instance, in (Venkataramani *et al.* 2003) only 21 word-pair classifiers were used for rescoring. This limits possible gains from the acoustic code-breaking in large vocabulary tasks.

Chapter 4

Structured Discriminative Models

In the previous chapter, three unstructured discriminative models, logistic regression, SVMs and multi-class SVMs were introduced. In these models the class labels are assumed to be unstructured (atomic). Thus the parameters associated with each of these classes can be considered independent. Although in isolated word recognition/frame phone classification these models can be directly applied (Birkenes *et al.* 2006; Salomon *et al.* 2002), in continuous speech recognition the number of classes (sentences) increases makes such approaches impractical (because the number of parameters and classifications becomes very large). One solution to handle this problem is to use acoustic code-breaking based approaches described in Section 3.4, where continuous speech is segmented into words/sub-words observation sequences. For each segment, multi-class SVMs or logistic regression can then be applied in the some fashion as an isolated classification tasks (Birkenes *et al.* 2006; Zhang *et al.* 2010). However, this approach has two issues. First, the classification is based on one, fixed, segmentation. Second, each segment is treated independently.

An alternative option is to introduce structure into the discriminative model by

breaking down class labels into atomic units, such as words, phones or states, similar to the standard approach applied with HMMs described in Chapter 2. Thus the parameters for any class (sentence) can be constructed from a common set of basic units. This transforms unstructured discriminative models such as logistic regression into *structured discriminative models*, e.g., conditional random fields (CRFs) (Abdel-Haleem 2006), hidden CRFs (HCRFs) (Gunawardana *et al.* 2005) and segmental CRFs (SCRF) (Zweig and Nguyen 2009). The discriminative parameter estimation and adaptation to speaker and noise conditions are discussed in this chapter.

4.1 Log Linear Models

The log linear model is a special form of discriminative model, in which the logarithm of posteriors are linear w.r.t. the feature functions, e.g., the logistic regression in Section 3.1, maximum entropy models (Jaynes 2003) and conditional random fields (Lafferty *et al.* 2001). It directly model the posterior in the following form

$$P_{\texttt{LLM}}(\mathbf{w}|\mathbf{O};\boldsymbol{\alpha}) = \frac{1}{\mathcal{Z}(\boldsymbol{\alpha};\mathbf{O})}\exp\left(\boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O},\mathbf{w})\right), \tag{4.1}$$

where $\mathcal{Z}(\boldsymbol{\alpha};\mathbf{O})$ is a normalisation term, required to ensure that $P_{\texttt{LLM}}(\mathbf{w}|\mathbf{O})$ is a valid probability mass function, and $\mathbf{w}$ may be a label sequence, as in the case of conditional random fields (Abdel-Haleem 2006); or an unstructured label w, as in the case of logistic regression described in Section 3.1 and maximum entropy models in (Jaynes 2003). $\boldsymbol{\phi}(\mathbf{O},\mathbf{w})$ is the *joint* feature function[1] extracting the statistical characteristics of $(\mathbf{O},\mathbf{w})$ pair. Various forms of log linear models have been applied for discriminative language modeling (Rosenfeld 1994), natural language processing (NLP) (Berger *et al.* 1996), discriminative model combination (Beyerlein 1997), machine translation (Och and Ney 2002) and speech recognition (Heigold 2010; Heigold *et al.* 2009; Hifny *et al.* 2005; Zhang *et al.* 2010). The following section describes a specific forms of log linear models—conditional random fields.

[1]To distinguish features $\boldsymbol{\psi}(\mathbf{O})$ for observation sequence in unstructured discriminative models, the feature function $\boldsymbol{\phi}(\mathbf{O},\mathbf{w})$ for observation-label sequence pair in structured discriminative models is referred as *joint* feature in this work.

4.1.1 Conditional Random Fields

Conditional random fields (CRFs) (Abdel-Haleem 2006; Hifny and Renals 2009; Lafferty *et al.* 2001) are one of the most popular conditional models in the machine learning area. They are related to Maximum Entropy Markov Models[2] (McCallum and Freitag 2000). CRFs use a single exponential distribution to model the posterior probability of the state sequence $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_T\}$ given an observation sequence $\mathbf{O} = \{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_T\}$,

$$P_{\mathtt{crf}}(\boldsymbol{\theta}|\mathbf{O};\boldsymbol{\alpha}) = \frac{1}{\mathcal{Z}(\boldsymbol{\alpha};\mathbf{O})} \exp\left(\boldsymbol{\alpha}^{\mathsf{T}} \sum_{t=1}^{T} \boldsymbol{\phi}(\boldsymbol{o}_t, \theta_t, \theta_{t-1})\right) \tag{4.2}$$

where $\mathcal{Z}(\boldsymbol{\alpha};\mathbf{O})$ is defined as calculating over all possible label sequences[3],

$$\mathcal{Z}(\boldsymbol{\alpha};\mathbf{O}) = \sum_{\boldsymbol{\theta}\in\Theta_T} \exp\left(\boldsymbol{\alpha}^{\mathsf{T}} \sum_{t=1}^{T} \boldsymbol{\phi}(\boldsymbol{o}_t, \theta_t, \theta_{t-1})\right). \tag{4.3}$$

where Θ_T is the set of all state sequences that have length T. The dependencies of the label sequence are only limited to the current and previous labels (a first-order Markov assumption). The model in equation (4.2) is actually a linear-chain CRF (Lafferty *et al.* 2001) as shown in Figure 4.1. It is defined using a frame-level feature function, $\boldsymbol{\phi}(\boldsymbol{o}_t, \theta_t, \theta_{t-1})$, weighted by a model parameter $\boldsymbol{\alpha}$. An example of CRF features is the Gaussian sufficient statistics (Gunawardana *et al.* 2005). This feature will be described in Section 4.2.

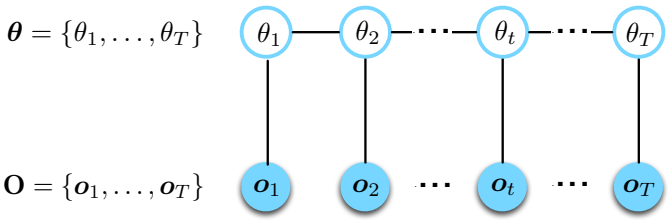


Figure 4.1 *An undirected graphical model for linear chain CRFs.*

[2] Maximum entropy Markov models are not introduced in this thesis since it has a major drawback called the label bias problem (Lafferty *et al.* 2001).

[3] The usage of global normalizer $\mathcal{Z}(\boldsymbol{\alpha};\mathbf{O})$ over all possible label sequences eliminates the label bias problem in maximum entropy Markov models (Lafferty *et al.* 2001).

The CRF in equation (4.2) can be expressed in the form of log linear model in (4.1) by constructing the joint feature space from frame-level features,

$$\boldsymbol{\phi}(\mathbf{O}, \boldsymbol{\theta}) = \sum_{t=1}^{T} \boldsymbol{\phi}(\boldsymbol{o}_t, \theta_t, \theta_{t-1}).$$

Note that although general CRF features can also depend on the complete observation sequence (Quattoni *et al.* 2004), $\boldsymbol{\phi}(\mathbf{O}, \boldsymbol{\theta})$, many applications of CRFs used the form in (4.2) for efficiency (Peng and McCallum 2006; Sha and Pereira 2003).

4.2 Hidden Conditional Random Fields

In the previous section CRFs were introduced as structured discriminative models for applications where the length of label sequence and observation sequence are the same. In CRFs an explicit one-to-one relationship between the observations $\boldsymbol{o}_t$ and state labels θ_t is assumed to exist. However, for applications such as speech recognition, this one-to-one relationship doesn't hold since the length of word sequence $\mathbf{w}$ is not same as observation sequence $\mathbf{O}$. Hidden conditional random fields (HCRFs) were introduced as one method to overcome this problem.

The hidden conditional random field (HCRF) (Gunawardana *et al.* 2005; Quattoni *et al.* 2007; Sung *et al.* 2007) is a latent-variable extension of standard CRFs that allows the one-to-one relationship between observations and labels to be relaxed. Similarly to CRFs, given an observation sequence $\mathbf{O} = \{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_T\}$, the posterior probability of a word/subword sequence $\mathbf{w} = \{w_1, \ldots, w_L\}$ for HCRF can be expressed as

$$P_{\texttt{hcrf}}(\mathbf{w}|\mathbf{O}; \boldsymbol{\alpha}) = \frac{1}{\mathcal{Z}(\boldsymbol{\alpha}, \mathbf{O})} \sum_{\boldsymbol{\theta} \in \Theta_{\mathbf{w}}^{T}} \exp\left(\boldsymbol{\alpha}^{\mathsf{T}} \boldsymbol{\phi}_{\mathtt{h}}(\mathbf{O}, \mathbf{w}, \boldsymbol{\theta})\right), \tag{4.4}$$

where $\boldsymbol{\alpha}$ is the vector of model parameters, $\boldsymbol{\phi}_{\mathtt{h}}(\mathbf{O}, \mathbf{w}, \boldsymbol{\theta})$ are HCRF features extracted from the observation sequence, word sequence and their latent alignment (state sequences $\boldsymbol{\theta}$), and $\Theta_{\mathbf{w}}^{T}$ is the set of all valid state sequences of $\mathbf{w}$ that have length T. The normalisation term $\mathcal{Z}(\boldsymbol{\alpha}, \mathbf{O})$ is obtained by marginalising the unnormalised models

over all possible word sequence $\mathbf{w}$ and state sequences $\boldsymbol{\theta}$.

$$\mathcal{Z}(\boldsymbol{\alpha}, \mathbf{O}) = \sum_{\mathbf{w} \in \mathcal{W}} \sum_{\boldsymbol{\theta} \in \Theta_{\mathbf{w}}^{T}} \exp\left(\boldsymbol{\alpha}^{\mathsf{T}} \boldsymbol{\phi}_{\mathtt{h}}(\mathbf{O}, \mathbf{w}, \boldsymbol{\theta})\right). \tag{4.5}$$

In a similar fashion to linear-chain CRFs, where a Markov dependency is assumed on the latent state sequence, $\mathcal{Z}(\boldsymbol{\alpha}, \mathbf{O})$ can be efficiently estimated using a forward-backward algorithm (Quattoni *et al.* 2007). Again the nature of joint feature $\boldsymbol{\phi}_{\mathtt{h}}(\mathbf{O}, \mathbf{w}, \boldsymbol{\theta})$ determines the form of dependencies and structures incorporated in the model. In the HCRF for speech recognition (Sung and Jurafsky 2009), the following frame-level features are used

$$\boldsymbol{\phi}_{\mathtt{h}}(\mathbf{O}, \mathbf{w}, \boldsymbol{\theta}) = \sum_{t=1}^{T} \boldsymbol{\phi}(\boldsymbol{o}_t, \theta_t, \theta_{t+1}) = \sum_{t=1}^{T} \begin{bmatrix} \boldsymbol{\phi}^{\mathtt{ac}}(\boldsymbol{o}_t, \theta_t) \\ \boldsymbol{\phi}^{\mathtt{lm}}(\theta_t, \theta_{t+1}) \end{bmatrix}, \tag{4.6}$$

where $\boldsymbol{\theta} \in \Theta_{\mathbf{w}}^{T}$ is a valid state sequence of $\mathbf{w}$ that has length T, and $\boldsymbol{\phi}^{\mathtt{ac}}(\boldsymbol{o}_t, \theta_t)$ and $\boldsymbol{\phi}^{\mathtt{lm}}(\theta_t, \theta_{t-1})$ are the frame-level acoustic feature and language feature, respectively. One way to define these frame-level features is to borrow the sufficient statistics associated with the latent variables in HMMs,

$$\boldsymbol{\phi}^{\mathtt{ac}}(\boldsymbol{o}_t, \theta_t) = \begin{bmatrix} \vdots \\ \delta(\theta_t = s_w)\boldsymbol{o}_t \\ \delta(\theta_t = s_w)\,\mathrm{diag}(\boldsymbol{o}_t \boldsymbol{o}_t^{\mathsf{T}}) \\ \vdots \end{bmatrix} \qquad \forall\, s_w \tag{4.7}$$

$$\boldsymbol{\phi}^{\mathtt{lm}}(\theta_t, \theta_{t+1}) = \begin{bmatrix} \vdots \\ \delta(\theta_t = s_w) \\ \delta(\theta_t = s_w, \theta_{t+1} = {s'}_{w'}) \\ \vdots \end{bmatrix} \qquad \forall\, s_w, {s'}_{w'} \tag{4.8}$$

where s_w and ${s'}_{w'}$ indicate states of each word in the vocabulary and $\delta(\theta_t = s_w)$ is equal to one when the state of frame t is s_w and zero otherwise. Thus the position of $\boldsymbol{o}_t$ and $\mathrm{diag}(\boldsymbol{o}_t \boldsymbol{o}_t^{\mathsf{T}})$ in the feature space $\boldsymbol{\phi}^{\mathtt{ac}}(\cdot)$ depends on its state label θ_t. As illustrated in equations (4.7) and (4.8), these acoustic and language features are composed of Gaussian sufficient statistics (which only depend on the observation and state at time

t), HMM-style state prior, state and word transition probabilities[4]. For this particular form of features, HCRFs can be shown be equivalent to discriminative training of HMMs (Heigold *et al.* 2007). This relationship can also be illustrated by comparing the undirected graphical model of HCRFs shown in Figure 4.2 with the directed graph of HMMs in Figure 2.2.

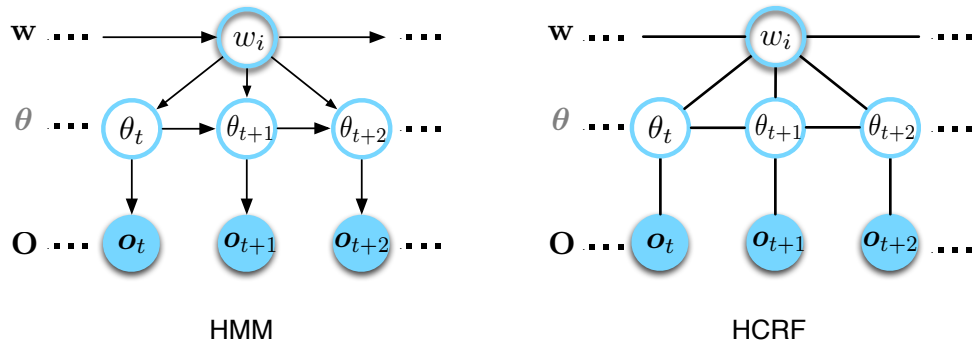


Figure 4.2 *An undirected graphical model for HCRFs with a specific alignment $\boldsymbol{\theta}$. This graphical model can be related to directed graph of HMMs in Figure 2.2.*

Note that this type of model allows the structure to be imposed in the feature function $\phi_{\mathrm{h}}(\mathbf{O}, \mathbf{w}, \boldsymbol{\theta})$. As there are latent states in HCRFs it is possible to compute local features for each state in particular words. Similar to a composite HMM illustrated in Figure 2.6, these local features associated with states and words can be combined together to yield a joint feature for a complete sentence. This is also the underlying form of augmented CRFs (Hifny and Renals 2009), where frame-level augmented observations are combined to predict a sentence.

4.3 Segmental Conditional Random Models

In the previous section CRFs and HCRFs with frame-level features are introduced. It is possible to relax the frame-level independece assumption in these model to segment-level to capture longer-span dependencies. This yields conditional augmented models

[4] Various forms of language features that can be applied to HCRFs will be discussed in Chapter 5.

(CAugs) (Layton 2006) and Segmental CRFs (SCRFs) (Zweig *et al.* 2011) expressed in the following form,

$$P_{\texttt{scrf}}(\mathbf{w}|\mathbf{O};\boldsymbol{\alpha}) = \frac{1}{\mathcal{Z}(\boldsymbol{\alpha},\mathbf{O})} \sum_{\boldsymbol{\theta}\in\Theta_{\mathbf{w}}^{T}} \exp\left(\boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\phi}_{\texttt{s}}(\mathbf{O},\mathbf{w},\boldsymbol{\theta})\right), \tag{4.9}$$

where $\boldsymbol{\theta}$ indicates the segmentation, and $\Theta_{\mathbf{w}}^{T}$ is the set of all possible segmentations for T frame observations and word sequence $\mathbf{w}$. The feature function, $\boldsymbol{\phi}_{\texttt{s}}(\mathbf{O},\mathbf{w},\boldsymbol{\theta})$, allows observations across a segment to be directly related (similar to generative segmental HMMs (Ostendorff *et al.* 1996)), other than through the state sequence in HCRFs,

$$\boldsymbol{\phi}_{\texttt{s}}(\mathbf{O},\mathbf{w},\boldsymbol{\theta}) = \sum_{i=1}^{|\mathbf{w}|} \boldsymbol{\phi}(\mathbf{O}_{i|\boldsymbol{\theta}}, w_i, w_{i-1}) = \sum_{i=1}^{|\mathbf{w}|} \begin{bmatrix} \boldsymbol{\phi}^{\texttt{ac}}(\mathbf{O}_{i|\boldsymbol{\theta}}, w_i) \\ \boldsymbol{\phi}^{\texttt{lm}}(w_i, w_{i-1}) \end{bmatrix} \tag{4.10}$$

where w_i is the word (or subword) label for the i-th segment. $\mathbf{O}_{i|\boldsymbol{\theta}}$ is the observation sequence for the i-th segment given a segmentation $\boldsymbol{\theta}$, $\boldsymbol{\phi}^{\texttt{ac}}(\mathbf{O}_{i|\boldsymbol{\theta}}, w_i)$ and $\boldsymbol{\phi}^{\texttt{lm}}(w_i, w_{i-1})$ are the segment-level acoustic feature and language feature, respectively. One simple form of $\boldsymbol{\phi}^{\texttt{ac}}(\mathbf{O}_{i|\boldsymbol{\theta}}, w_i)$ is described as follows,

$$\boldsymbol{\phi}^{\texttt{ac}}(\mathbf{O}_{i|\boldsymbol{\theta}}, w_i) = \begin{bmatrix} \vdots \\ \delta(w_i = w)\boldsymbol{\psi}(\mathbf{O}_{i|\boldsymbol{\theta}}) \\ \vdots \end{bmatrix} \qquad \forall\, w \tag{4.11}$$

where $\boldsymbol{\psi}(\mathbf{O}_{i|\boldsymbol{\theta}})$ is the log likelihood features described in Section 3.1.1 which can map variable-length observations into a fixed-dimension vector,[5]

$$\boldsymbol{\psi}(\mathbf{O}_{i|\boldsymbol{\theta}}) = \begin{bmatrix} \vdots \\ \log p_{\boldsymbol{\lambda}}(\mathbf{O}_{i|\boldsymbol{\theta}}|w = \texttt{dog}) \\ \log p_{\boldsymbol{\lambda}}(\mathbf{O}_{i|\boldsymbol{\theta}}|w = \texttt{cat}) \\ \vdots \end{bmatrix}.$$

[5] More examples of acoustic and language features that can be applied to SCRFs will be discussed in Chapter 5.

The feature $\boldsymbol{\phi}^{\mathtt{lm}}(w_i, w_{i-1})$ is related to unigram and bigram language models,

$$\boldsymbol{\phi}^{\mathtt{lm}}(w_i, w_{i-1}) = \begin{bmatrix} \vdots \\ \delta(w_i = w) \\ \delta(w_i = w, w_{i-1} = w') \\ \vdots \end{bmatrix} \quad \forall\, w, w' \tag{4.12}$$

where w and w' correspond to all possible words in the vocabulary. Other forms of segmental features will be discussed in Chapter 5. Thus the only difference between SCRFs in equation (4.9) and HCRFs in (4.4) is their feature functions. In HCRFs the frame-level features in (4.7) are used whereas in SCRFs segmental features can be incorporated. The graphical model for SCRFs is shown in Figure 4.3.

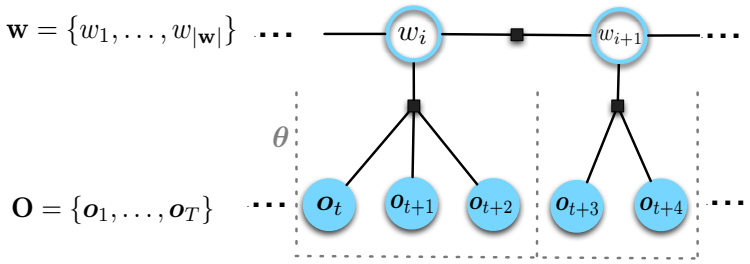


Figure 4.3 *A graphical model for SCRFs with a specific segmentation* $\boldsymbol{\theta}$.

4.4 Parameter Estimation for Discriminative Models

An appropriate training criterion is very important for speech recognition (Schlüter *et al.* 2001). In the same fashion as generative models, such as the HMM (Section 2.3) it is possible to use a range of discriminative criteria with the structured discriminative models (Gales *et al.* 2012). Examples include conditional maximum likelihood (CML), minimum word error (MWE)/minimum phone error (MPE) and maximum margin (MM) criteria. This section will discuss these criteria for structured discriminative models in Sections 4.1, 4.2 and 4.3.

4.4.1 Conditional Maximum Likelihood

The structured discriminative model parameter estimation based on the CML criterion, given a dataset $\mathcal{D}$ (see equation (2.27)), may be performed by maximising the following objective function

$$\mathcal{F}_{\texttt{cml}}(\boldsymbol{\alpha}|\mathcal{D}) = \sum_{r=1}^{R} \log P(\mathbf{w}_{\texttt{ref}}^{(r)}|\mathbf{O}^{(r)}; \boldsymbol{\alpha}), \tag{4.13}$$

where $\boldsymbol{\alpha}$ are model parameters for CRFs, HCRFs or SCRFs. It is worth noting that the CML objective function, above, is similar to the MMI objective function in equation (2.29). The difference between the two approaches lies in whether they update a generative model (MMI) or a discriminative model (CML).

For structured discriminative models, setting the gradient of equation (4.13) with respect to $\boldsymbol{\alpha}$ to zero does not yield analytic closed-form solutions. Instead, models are typically estimated using iterative algorithms, such as EM-style algorithms (Darroch and Ratcliff 1972; Lafferty *et al.* 2001) and gradient-based algorithms (Gunawardana *et al.* 2005; Layton 2006; Mahajan *et al.* 2006; Quattoni *et al.* 2004; Sha and Pereira 2003). In the case of CRFs (see equation (4.2)), the objective function is convex, causing it to have a single, global, maximum (Sutton and McCallum 2006). However, due to the additional latent variable, this is not the case for HCRFs and SCRFs. In these models the posterior of word sequence $\mathbf{w}$ in equations (4.4) and (4.9) involves summing over all segmentations, compared to CRFs in equation (4.2). Thus, for HCRFs and SCRFs, the CML objective function in equation (4.13) may be expressed as

$$\begin{aligned}\mathcal{F}_{\texttt{cml}}(\boldsymbol{\alpha}|\mathcal{D}) = \sum_{r=1}^{R} \Bigg[&\log \left(\sum_{\boldsymbol{\theta}} \exp \left(\boldsymbol{\alpha}^{\mathsf{T}} \boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}_{\texttt{ref}}^{(r)}, \boldsymbol{\theta}) \right) \right) - \\ &\log \left(\sum_{\mathbf{w}} \sum_{\boldsymbol{\theta}} \exp \left(\boldsymbol{\alpha}^{\mathsf{T}} \boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}, \boldsymbol{\theta}) \right) \right) \Bigg]\end{aligned} \tag{4.14}$$

where $\boldsymbol{\phi}(\mathbf{O}, \mathbf{w}, \boldsymbol{\theta})$ is the HCRF or SCRF feature. The first, *numerator*, term is the logarithm of the unnormalised posterior and second, *denominator*, term is the logarithm of the normalisation term $\mathcal{Z}(\boldsymbol{\alpha}, \mathbf{O})$.

Directly optimising the objective functions above for HCRFs/SCRFs is complicated due to the summation over all possible alignment and word sequences (Layton 2006; Nguyen and Zweig 2010). A related problem for HMMs was addressed by using the *lattice framework* (see Section 2.4.4 and (Valtchev *et al.* 1997)). The objective functions in equation (4.14) may be expressed in the lattice framework as

$$\mathcal{F}_{\texttt{cml}}(\boldsymbol{\alpha}|\mathcal{D}) = \sum_{r=1}^{R} \log\left([[\mathbb{L}_{\texttt{num}}^{(r)}]]\right) - \log\left([[\mathbb{L}_{\texttt{den}}^{(r)}]]\right), \tag{4.15}$$

where $\mathbb{L}_{\texttt{num}}^{(r)}$ and $\mathbb{L}_{\texttt{den}}^{(r)}$ are the numerator and denominator lattice respectively. These lattices are used to constrain the possible alignments and word sequences. $[[\cdot]]$ is the *lattice weight* (Layton 2006) which can be calculated using the lattice-based forward-backward algorithm described in (Nguyen and Zweig 2010). Alternatively, the lattice weight can also be approximated using more efficient lattice-based Viterbi algorithms described in (Layton 2006), which only calculates the "most likely" alignment instead of "all possible" alignments in equation (4.14). Thus, in practice, the discriminative parameters $\boldsymbol{\alpha}$ of these models can be optimized efficiently by maximising an objective function (4.15) based on standard stochastic gradient descent (Layton 2006; Nguyen and Zweig 2010).

4.4.2 Minmum Bayes' Risk

The structured discriminative model parameter estimation based on the MBR criterion may be performed by minimising the following objective function

$$\mathcal{F}_{\texttt{mbr}}(\boldsymbol{\alpha}|\mathcal{D}) = \sum_{r=1}^{R} \sum_{\mathbf{w}} P(\mathbf{w}|\mathbf{O}^{(r)}; \boldsymbol{\alpha}) \mathcal{L}(\mathbf{w}, \mathbf{w}_{\texttt{ref}}^{(r)}) \tag{4.16}$$

where $\mathbf{w}$ denotes all possible hypothesis, $P(\mathbf{w}|\mathbf{O}^{(r)}; \boldsymbol{\alpha})$ could be the posterior probability of CRFs, HCRFs or SCRFs, and $\mathcal{L}(\mathbf{w}, \mathbf{w}_{\texttt{ref}}^{(r)})$ is a loss functions. Similar to the MBR training of HMMs described in Section 2.3.4, the resulting objective function can be called *minimum word error* (MWE), *minimum phone error* (MPE), or *minimum phone frame error* (MPFE) for structured discriminative models depending on

whether the loss function is computed at the word, phone or frame level (Ragni 2013). Optimising HCRF/SCRF model parameters based on the MWE/MPE/MPFE objective function in equation (4.16) can be computationally expensive (Layton 2006; Zweig *et al.* 2011). In order to address computational issues, again, the lattice framework discussed in Sections 2.4.4 and 4.4.1 can be adopted (Layton 2006; Ragni 2013).

4.4.3 *Maximum Margin*

In order to robustly train a discriminative model which has good generalization in a high-dimensional space with limited data, large margin based approaches can be applied (Li *et al.* 2007; Taskar 2005). If the margin for the structured discriminative models is defined as the log-posterior ratio of the reference $\mathbf{w}_{\mathtt{ref}}^{(r)}$ and best competing hypothesis $\mathbf{w}$, as illustrated in Figure 4.4, large margin training for structured discriminative model can be expressed as *maximising*

$$\mathcal{F}_{\mathtt{mm}}(\boldsymbol{\alpha}) = \sum_{r=1}^{R} \left(\min_{\mathbf{w} \neq \mathbf{w}_{\mathtt{ref}}^{(r)}} \left\{ \log \left(\frac{P(\mathbf{w}_{\mathtt{ref}}^{(r)}|\mathbf{O}^{(r)};\boldsymbol{\alpha})}{P(\mathbf{w}|\mathbf{O}^{(r)};\boldsymbol{\alpha})} \right) \right\} \right) \tag{4.17}$$

Similar to the maximum margin training of HMMs in Section 2.3.5, the loss func-

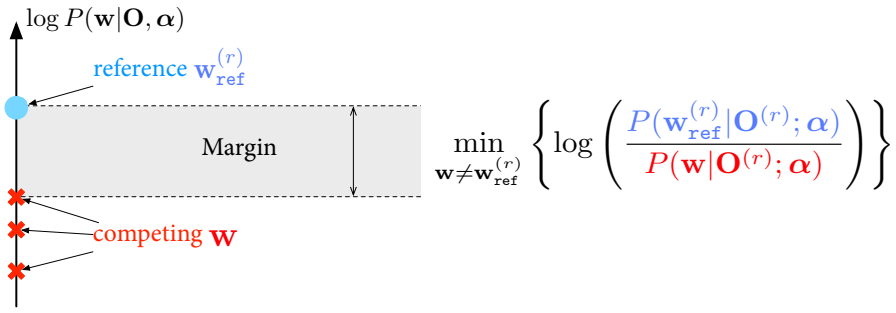


Figure 4.4 *The margin of structured discriminative models is defined in the log posterior domain between* $\mathbf{w}_{\mathtt{ref}}^{(r)}$ *and the "closest" competing hypothesis* $\mathbf{w}$.

tion is introduced to generalize the margin. To prevent the log-posterior ratio from growing arbitrary large, the hinge function $[\,\cdot\,]_+$ defined in equation (2.37) may also

be introduced. This leads to *minimising* the following expression

$$\mathcal{F}_{\texttt{mm}}(\boldsymbol{\alpha}) = \sum_{r=1}^{R} \left[\max_{\mathbf{w} \neq \mathbf{w}_{\texttt{ref}}^{(r)}} \left\{ \mathcal{L}(\mathbf{w}_{\texttt{ref}}^{(r)}, \mathbf{w}) - \log \left(\frac{P(\mathbf{w}_{\texttt{ref}}^{(r)} | \mathbf{O}^{(r)}; \boldsymbol{\alpha})}{P(\mathbf{w} | \mathbf{O}^{(r)}; \boldsymbol{\alpha})} \right) \right\} \right]_{+} \tag{4.18}$$

Note that this objective function is non-differentiable because of the $\max\{\cdot\}$ function. Thus gradient-based algorithms cannot be directly applied. Two approaches to deal with this problem are summarized as follows.

Approximate Margin One approach is to apply a soft-max approximation to this objective function, similar to the maximum margin (Sha and Saul 2007) and boosted MMI (Povey *et al.* 2008) training for HMMs. This yields a differentiable objective function. However this approximation minimises an upper bound of equations (4.18) rather than the criterion itself

$$\begin{aligned} \mathcal{F}_{\texttt{mm}}(\boldsymbol{\alpha}) \leq \frac{1}{R} \sum_{r=1}^{R} \Bigg[& - \log P(\mathbf{w}_{\texttt{ref}}^{(r)} | \mathbf{O}^{(r)}; \boldsymbol{\alpha}) \\ & + \log \Big\{ \sum_{\mathbf{w}} \exp \Big(\mathcal{L}(\mathbf{w}, \mathbf{w}_{\texttt{ref}}^{(r)}) \Big) P(\mathbf{w} | \mathbf{O}^{(r)}; \boldsymbol{\alpha}) \Big\} \Bigg]_{+} \end{aligned} \tag{4.19}$$

This equation (4.19) allows gradient-based optimisation to be applied for log linear models and HCRFs/SCRFs (Taskar 2005), but it is not the "exact" maximum margin in equation (4.18).

Exact Margin Alternatively, approaches of maximising the "exact" margin in (4.18) exist for specific models. For log linear models, substituting equation (4.1) into (4.18), the objective function becomes

$$\begin{aligned} \mathcal{F}_{\texttt{mm-llm}}(\boldsymbol{\alpha}) = \sum_{r=1}^{R} \Bigg[& \overbrace{- \boldsymbol{\alpha}^{\mathsf{T}} \boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}_{\texttt{ref}}^{(r)})}^{\text{linear}} + \\ & \underbrace{\max_{\mathbf{w} \neq \mathbf{w}_{\texttt{ref}}^{(r)}} \Big\{ \mathcal{L}(\mathbf{w}_{\texttt{ref}}^{(r)}, \mathbf{w}) + \boldsymbol{\alpha}^{\mathsf{T}} \boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}) \Big\}}_{\text{convex}} \Bigg]_{+} \end{aligned} \tag{4.20}$$

Note that the normalization term $\mathcal{Z}(\boldsymbol{\alpha}; \mathbf{O})$ in (4.1) cancels from denominator and numerator. This objective function is convex for $\boldsymbol{\alpha}$. Log linear models trained with this

objective function is closely related to the structured SVMs (see details in Chapter 6). Although the objective function (4.20) is non-differentiable, it can be directly solved using the sub-gradient algorithm in (Singer and Srebro 2007) or cutting plane algorithm (Joachims *et al.* 2009) described in Section 6.2.1.

For HCRFs and SCRFs, substituting equations (4.4) or (4.9) into (4.18), again the normalization terms in (4.4) or (4.9) *cancel out*. Thus the objective function of HCRFs/SCRFs becomes

$$\mathcal{F}_{\texttt{mm-scrf}}(\boldsymbol{\alpha}) = \sum_{r=1}^{R} \Bigg[\overbrace{-\log \sum_{\boldsymbol{\theta}^{(r)} \in \Theta_{(r)}^{T}} \exp\left(\boldsymbol{\alpha}^{\mathsf{T}} \boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}_{\texttt{ref}}^{(r)}; \boldsymbol{\theta}^{(r)})\right)}^{\text{concave}} + \tag{4.21}$$

$$\underbrace{\max_{\mathbf{w} \neq \mathbf{w}_{\texttt{ref}}^{(r)}} \left\{ \mathcal{L}(\mathbf{w}_{\texttt{ref}}^{(r)}, \mathbf{w}) + \log \sum_{\boldsymbol{\theta} \in \Theta_{\mathbf{w}}^{T}} \exp\left(\boldsymbol{\alpha}^{\mathsf{T}} \boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}; \boldsymbol{\theta})\right) \right\}}_{\text{convex}} \Bigg]_{+}$$

where $\Theta_{(r)}^{T}$ is the set of all valid state sequences of $\mathbf{w}_{\texttt{ref}}^{(r)}$ that have length T. There are two challenges to estimate $\boldsymbol{\alpha}$ by minimising function (4.21). First, the objective function is no longer convex. It consists of a concave and a convex functions. The optimisation can be solved using the concave-convex procedure (CCCP) (Yuille *et al.* 2002) and cutting plane algorithm (Joachims *et al.* 2009). These algorithms will be described in details in Chapter 6. Second, the training and decoding time can be slow because of summing over all possible segmentations. Thus the lattice-based framework (Layton 2006; Ragni 2013) may also applied to restrict the number of possible segmentations. An alternative method is using one Viterbi segmentation to approximate the summing over all segmentations (Ragni 2013). In this case, HCRFs/SCRFs will become structured SVMs with latent variables (see more details in Section 6.4.3).

4.5 Adaptation to Speaker and Noise Condition

For generative models, adaptation to a specific speaker or environment condition is an essential part of current speech recognition systems (Gales *et al.* 2012). A number of approaches have been proposed, including maximum a posteriori (MAP) adapta-

tion, linear transformation-based approaches, model-based noise compensation, and feature enhancement (for details and references, see (Gales and Young 2007)). Related approaches have been proposed for discriminative models as well. Note that in machine learning literature, the problem of handling a mismatch between training and test conditions is sometimes referred to as sample selection bias or covariate shift (Gales *et al.* 2012). These adaptation approaches can be devided into three broad categories: general adaptation, linear transformation approaches, and feature adaptation (Gales *et al.* 2012). Note that in contrast to the majority of adaptation approaches for generative models that are based on maximum likelihood, discriminative model adaptation is usually based on conditional maximum likelihood.

In (Chelba and Acero 2006), two schemes for adapting log-linear models—MAP adaptation and minimum divergence training—are discussed. These approaches yield a general adaptation framework that makes no assumption about the nature of the features in the model. MAP adaptation has also been applied to HCRFs (Sung *et al.* 2007). Though these general adaptation approaches can be used for discriminative models, they do not take advantage of any structure in the features. Alternatively, linear transformation-based approaches for log-linear models are described in (Sung *et al.* 2008) and (Loof *et al.* 2010). These schemes use techniques similar to the linear transformations for HMMs. Assumptions are made about the relationships between features. To date, they have only been applied to models where the features are very similar to those used in standard HMMs. Whether these approaches can be extended to more general features is an open question.

The final form of adaptation is related to the feature compensation schemes used for generative models. Rather than adapting the model parameters, the features are modified to make them independent of the speaker or environment. This is simplest to do when the feature extraction process is based on generative models (Gales and Flego 2010; Zhang *et al.* 2010). This approach is discussed in more detail in the Section 5.1.3.

4.6 Summary

In Chapters 2, 3 and 4 generative and discriminative models have been discussed. These models can also be categorized into *unstructured* and *structured* to deal with different types of data. For example, the unstructured generative model, naive Bayes, is introduced in Section 2.1; structured generative models, HMMs, are introduced in Section 2.2; unstructured discriminative models, logistic regression and SVMs, are described in Chapter 3; structured discriminative models, CRFs, HCRFs and SCRFs, are described in Chapter 4.

Generative and Discriminative Models Generative classifiers use models of the joint distribution, typically of the form $p(\mathbf{O}, \mathbf{w}) = p(\mathbf{O}|\mathbf{w})P(\mathbf{w})$. Discriminative models, on the other hand, focus on the conditional distribution $P(\mathbf{w}|\mathbf{O})$. As discussed earlier, one advantage of discriminative modeling is that it has more freedom to fit the data (Minka 2005; Sutton and McCallum 2006) and is more suitable to including rich, long-span features (Heigold 2010). The relationship between naive Bayes and logistic regression is a typical example of generative-discriminative pair (Ng and Jordan 2001). This mirrors the relationship between HMMs and HCRFs as illustrated in Figure 4.5.

Unstructured and Structured Models Unstructured models, for example the naive Bayes and logistic regression, assume that the class labels are atomic (denoted as w). The parameters for different classes are not shared. For isolated word recognition these models can be directly applied (Birkenes *et al.* 2006), however in continuous speech recognition as the number of classes (sentences) increases such approach becomes impractical. Structured models, for example HMMs, HCRFs and SCRFs, assume that there are structures in classes (denoted as $\mathbf{w}$) and parameters for different classes may be dependent. These classes (sentences) can be breaken down into some atomic units, such as words, phones or states. Thus the parameters for any classes (sentence) can be constructed from a common set of basic parameters associated to these atomic

units. The relationship between these unstructured and structured models are also illustrated in Figure 4.5.

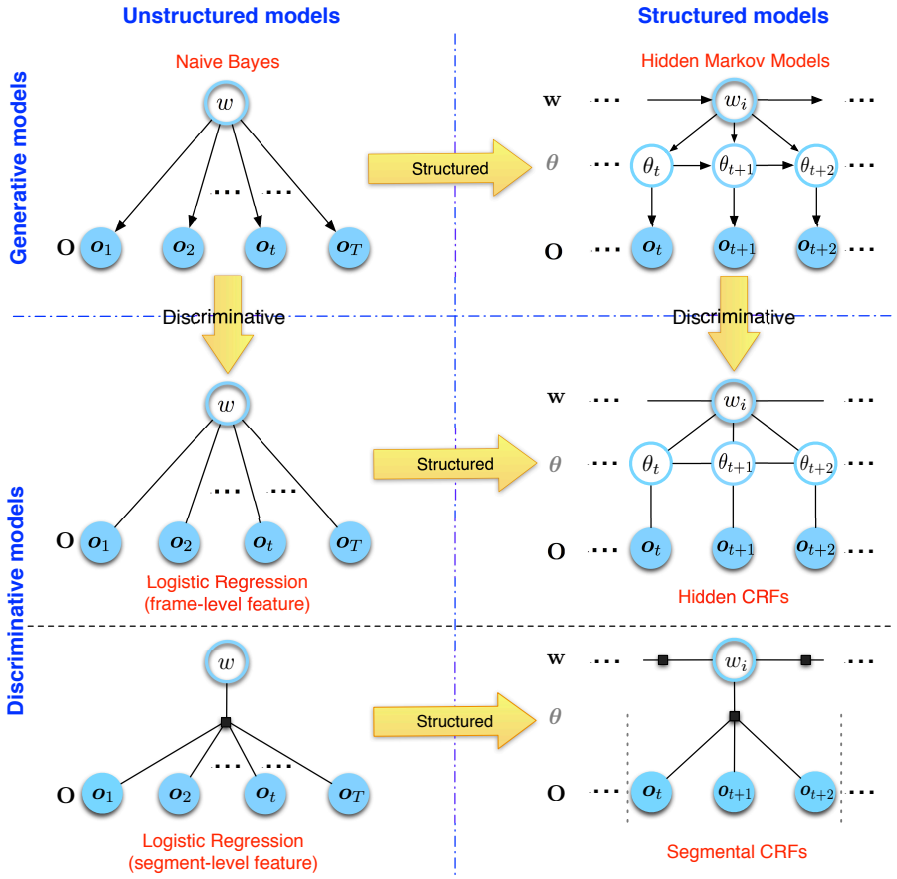


Figure 4.5 *Graphical model relationship. Naive Bayes and Logistic Regression form a generative-discriminative pair for classification. Their relationship mirrors that between HMMs and HCRFs for sequential structured data. SCRFs can be viewed as a segmental extension to HCRFs. Note that only one alignment/segmentation for HMMs, HCRFs and SCRFs is shown.*

Chapter 5

Feature Functions

In speech recognition the selection of features is important to the performance of the models. The previous chapters have assumed that appropriate (joint) feature functions for the models exist. For example, one general form of features in Chapter 4 can be expressed as,

$$\boldsymbol{\phi}(\mathbf{O}, \mathbf{w}; \boldsymbol{\theta}) = \begin{bmatrix} \boldsymbol{\phi}^{\mathtt{ac}}(\mathbf{O}, \mathbf{w}; \boldsymbol{\theta}) \\ \boldsymbol{\phi}^{\mathtt{lm}}(\mathbf{w}, \boldsymbol{\theta}) \end{bmatrix} \tag{5.1}$$

The term $\boldsymbol{\phi}^{\mathtt{ac}}(\mathbf{O}, \mathbf{w}; \boldsymbol{\theta})$ denotes the acoustic features which relate to the observation statistics; The term $\boldsymbol{\phi}^{\mathtt{lm}}(\mathbf{w}, \boldsymbol{\theta})$ denotes the language features which relate to pronunciation probabilities and word statistics. In this chapter, features for speech recognition are broadly categorized into acoustic features and language features. These can be split into frame level and segmental level features. These features which discussed in the following sections can be used with all of the discriminative models in this thesis.

5.1 Acoustic Features

Acoustic features can be split into the frame level and the segmental level. Each segment has an associated word (or subword) indicated by w_i, where $i = 1, \ldots, |\mathbf{w}|$. $|\mathbf{w}|$ represents the number of words (or subwords). For the notation in this thesis, the latent variables $\boldsymbol{\theta}$ is used to specify the hidden (subphone) states in frame-level features;

For segment-level features, the latent variables $\boldsymbol{\theta}$ indicate the boundary (the start and end frames) of each segment, w_i.

5.1.1 *Frame-level features*

The simplest form of acoustic feature is restricted to frame-level features in the same fashion as the HCRF features (Gunawardana *et al.* 2005),

$$\boldsymbol{\phi}^{\mathsf{ac}}(\mathbf{O}, \mathbf{w}; \boldsymbol{\theta}) = \sum_{t=1}^{T} \boldsymbol{\phi}^{\mathsf{ac}}(\mathbf{o}_t, \theta_t) = \sum_{t=1}^{T} \begin{bmatrix} \vdots \\ \delta(\theta_t = s_w)\boldsymbol{\psi}(\boldsymbol{o}_t) \\ \vdots \end{bmatrix} \forall s_w \tag{5.2}$$

where $\delta(\cdot)$ is the Kronecker delta function (indicator function), w is the word or phone label and s_w indicates the hidden (subphone) state in each w. $\boldsymbol{\psi}(\cdot)$ is the features extracted from frame-level observations (not depend on the state labels). Two examples of $\boldsymbol{\psi}(\boldsymbol{o}_t)$ are discussed in the following two subsections. The feature space in equation (5.2) can also be viewed as a tensor product of vectors $\boldsymbol{\delta}(\theta_t)$ and $\boldsymbol{\psi}(\boldsymbol{o}_t)$,

$$\boldsymbol{\phi}^{\mathsf{ac}}(\mathbf{O}, \mathbf{w}; \boldsymbol{\theta}) = \sum_{t=1}^{T} \boldsymbol{\delta}(\theta_t) \otimes \boldsymbol{\psi}(\boldsymbol{o}_t) \tag{5.3}$$

where the index vector $\boldsymbol{\delta}(\theta_t)$ can be expressed as

$$\boldsymbol{\delta}(\theta_t) = \begin{bmatrix} \vdots \\ \delta(\theta_t = s_w) \\ \vdots \end{bmatrix} \forall s_w \tag{5.4}$$

where s_w indicates the hidden state in each w. Thus the position of $\boldsymbol{\psi}(\boldsymbol{o}_t)$ in the joint feature space depends on its state label θ_t. The form of equation (5.3) is chosen to allow a simple comparison between frame-level features and the segmental features in equation (5.8).

Dot-product caculation One interesting property of this form of the joint feature space in equation (5.2) is the the dot product of the $\boldsymbol{\phi}^{\mathsf{ac}}(\mathbf{O}, \mathbf{w}; \boldsymbol{\theta})$ and discriminative

parameters $\boldsymbol{\alpha}^{\mathsf{ac}}$ can be evaluated by accumulating every frame score

$$\boldsymbol{\alpha}^{\mathsf{ac}\mathsf{T}}\boldsymbol{\phi}^{\mathsf{ac}}(\mathbf{O}, \mathbf{w}; \boldsymbol{\theta}) = \sum_{t=1}^{T} \boldsymbol{\alpha}^{(\theta_t)^{\mathsf{T}}}\boldsymbol{\psi}(\boldsymbol{o}_t), \text{ where } \theta_t \in \{s_w\} \tag{5.5}$$

where $\boldsymbol{\alpha}^{\mathsf{ac}\mathsf{T}} = [\ldots, \boldsymbol{\alpha}^{(s_w)^{\mathsf{T}}}, \ldots]$ and s_w indicates each hidden states of each w. This property allows frame-level Viterbi decoding. This will be discussed further in Section 6.3.1.

5.1.1.1 Gaussion statistic features

The simplest form of $\boldsymbol{\psi}(\boldsymbol{o}_t)$ uses the Gaussian sufficient statistics of observations:

$$\boldsymbol{\psi}(\boldsymbol{o}_t) = \begin{bmatrix} \boldsymbol{o}_t \\ \operatorname{diag}(\boldsymbol{o}_t \boldsymbol{o}_t^{\mathsf{T}}) \end{bmatrix} \tag{5.6}$$

Substituting equation (5.6) into (5.2) yields the HCRF features in (Gunawardana *et al.* 2005). The features in equation (5.6) also related to the HMM mean and covariance statistics (Heigold *et al.* 2007). Applying this feature to the HCRFs yields discriminatively trained HMMs (Heigold *et al.* 2007). This feature can also be extended to include features of higher-order statistics (Wiesler *et al.* 2009).

Another variation of this frame-level feature is to modify the form of observation $\boldsymbol{o}_t$. Rather than just considering a single frame $\boldsymbol{o}_t$, frames can be spliced together and optionally transformed (Hifny and Renals 2009), to form a larger observation vector. This is similar to the approach used with standard HMMs where delta and delta-delta features are appended.

5.1.1.2 MLP features

Much recent work in speech recognition uses a multi-layer perceptrons (MLP) as a feature extractor, trained to produce phoneme posterior probabilities – both probabilistic features (Zhu *et al.* 2004) and bottleneck configurations (Grézl *et al.* 2007) have been investigated and reported in literature – using either the traditional GMM-HMM (Zhu *et al.* 2004) or the "hybrid" MLP-HMM (Hinton *et al.* 2012) as the back-end classifier. The MLP features can also be incorporated together with PLP features

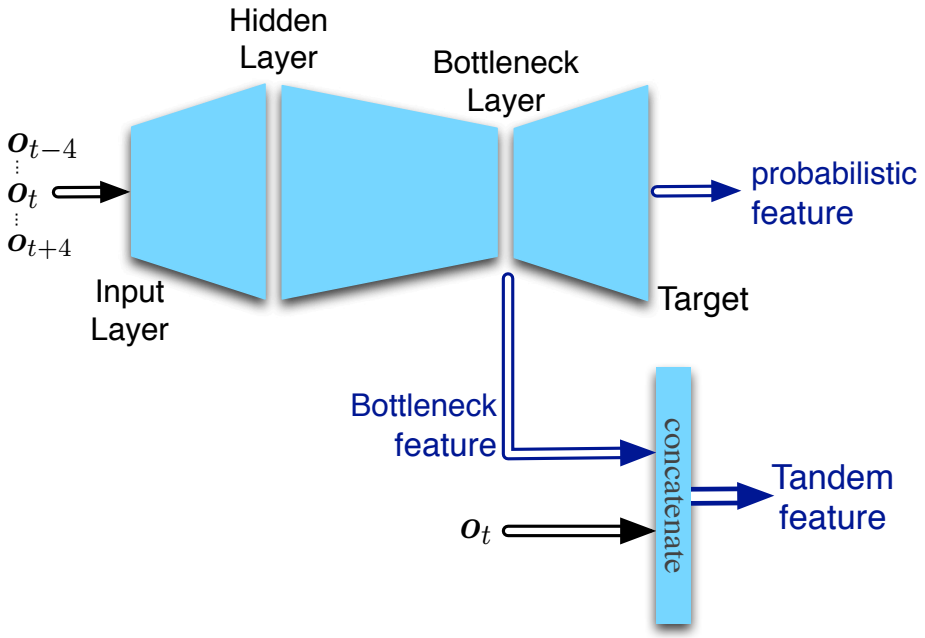


Figure 5.1 *The illustration of MLP based features: probabilistic features, bottleneck features and tandem features.*

as *Tandem features* (Zhu *et al.* 2005) as shown in Figure 5.1. The features can also be normalised through feature space projections such as heteroscedastic LDA (HLDA) (Kumar 1997) and CMLLR (Gales 1998). Previous research has demonstrated that using these features from deep neural network (DNN) for HMMs yields significant gains in LVCSR (Hinton *et al.* 2012). These MLP based features can be also applied to discriminative models (Morris and Fosler-Lussier 2008).

5.1.2 *Segment-level features*

The frame-level features described in Section 5.1.1 will generate T feature vectors for a T-length observation sequence. An alternative option is to derive features depend on all the observations associated with a segment to introduce some long-term dependencies. This serves as basis of segmental conditional random fields (SCRF) in Section 4.3. In contrast to the frame-level features (5.2), one general form of segment-level

features is

$$\boldsymbol{\phi}^{\mathtt{ac}}(\mathbf{O}, \mathbf{w}; \boldsymbol{\theta}) = \sum_{i=1}^{|\mathbf{w}|} \boldsymbol{\phi}^{\mathtt{ac}}(\mathbf{O}_{i|\boldsymbol{\theta}}, w_i) = \sum_{i=1}^{|\mathbf{w}|} \begin{bmatrix} \delta(w_i = v_1)\boldsymbol{\psi}(\mathbf{O}_{i|\boldsymbol{\theta}}) \\ \vdots \\ \delta(w_i = v_{\mathtt{M}})\boldsymbol{\psi}(\mathbf{O}_{i|\boldsymbol{\theta}}) \end{bmatrix} \tag{5.7}$$

where w_i is the label for the i-th segment, normally a word or subword. $\mathbf{O}_{i|\boldsymbol{\theta}}$ is the observation sequence for the i-th segment given the alignment $\boldsymbol{\theta}$. $\{v_k\}_{k=1}^{M}$ denotes, for example, all possible words or subwords in the dictionary, $\delta(w_i = v_k)$ is the Kronecker delta function, and $\boldsymbol{\psi}(\mathbf{O}_{i|\boldsymbol{\theta}})$ is the feature vector extracted for segment $\mathbf{O}_{i|\boldsymbol{\theta}}$ as shown in Figure 5.2. Note that the joint feature space in equation (5.7) can also be viewed as a tensor product of vectors $\boldsymbol{\delta}(w_i)$ and $\boldsymbol{\psi}(\mathbf{O}_{i|\boldsymbol{\theta}})$[1]

$$\boldsymbol{\phi}(\mathbf{O}, \mathbf{w}; \boldsymbol{\theta}) = \sum_{i=1}^{|\mathbf{w}|} \boldsymbol{\delta}(w_i) \otimes \boldsymbol{\psi}(\mathbf{O}_{i|\boldsymbol{\theta}}) \tag{5.8}$$

where the index vector $\boldsymbol{\delta}(w_i)$ can be expressed as

$$\boldsymbol{\delta}(w_i) = \begin{bmatrix} \delta(w_i = v_1) \\ \vdots \\ \delta(w_i = v_{\mathtt{M}}) \end{bmatrix}_{\mathtt{M}}$$

Again the position of $\boldsymbol{\psi}(\mathbf{O}_{i|\boldsymbol{\theta}})$ in the *joint* feature space depends on its label w_i. Figure 5.2 shows an example of using equation (5.7) to construct a joint feature space for data pair $(\mathbf{O}, \mathbf{w})$ given a segmentation $\boldsymbol{\theta}$.

The target of designing segment-level features $\boldsymbol{\psi}(\mathbf{O}_{i|\boldsymbol{\theta}})$ is to capture any long-term dependency in the data and to relax the frame-level independent assumption to segmental level. It is possible to hypothesize a range of segment-level features that could be used. However, one interesting approach is to consider this process in the context of sequence kernels and score spaces (Layton 2006). These sequence kernels map variable length sequences to a fixed-dimension vector in which the inner product can be computed. The advantage of discussing acoustic features in this framework is that

[1] Comparing equation (5.8) with (5.3) shows the difference between segmental features and frame-level features.

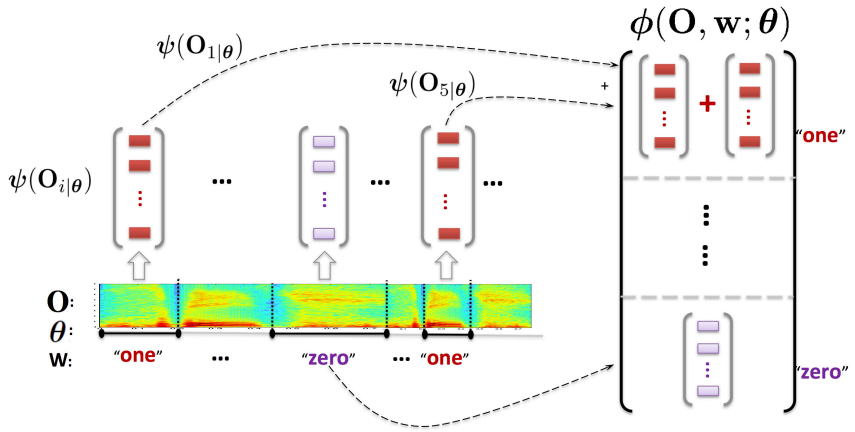


Figure 5.2 *Constructing the* joint *feature space from feature space.*

existing developments from machine learning can be used (Gales *et al.* 2012). To map each variable-length segment $\mathbf{O}_{i|\boldsymbol{\theta}}$ to a fixed dimensional vector $\boldsymbol{\psi}(\mathbf{O}_{i|\boldsymbol{\theta}})$, a range type of sequence kernel can be used. Of particular interest in this work are those sequence kernels based on generative models $\boldsymbol{\lambda}$. As well as yielding a mapping to a fixed vector size, these generative kernels allow standard speaker and noise adaptation approaches developed for CSR to be used to derive robust features (see more in section 6.5.5). Two examples of generative feature space $\boldsymbol{\psi}_{\boldsymbol{\lambda}}(\mathbf{O}_{i|\boldsymbol{\theta}})$ are discussed in the following subsections.[2]

Dot-product caculation One interesting property of the form of joint feature space in equation (5.7) is the the dot product of the $\boldsymbol{\phi}^{\mathsf{ac}}(\mathbf{O}, \mathbf{w}; \boldsymbol{\theta})$ and discriminative parameters $\boldsymbol{\alpha}^{\mathsf{ac}}$ can be evaluated by accumulating every segment score (Zhang *et al.* 2010)

$$\boldsymbol{\alpha}^{\mathsf{ac}\mathsf{T}}\boldsymbol{\phi}^{\mathsf{ac}}(\mathbf{O}, \mathbf{w}; \boldsymbol{\theta}) = \sum_{i=1}^{|\mathbf{w}|} \boldsymbol{\alpha}^{(w_i)^{\mathsf{T}}} \boldsymbol{\psi}(\mathbf{O}_{i|\boldsymbol{\theta}}), \tag{5.9}$$

where $\boldsymbol{\alpha}^{\mathsf{ac}\mathsf{T}} = [\boldsymbol{\alpha}^{(v_1)^{\mathsf{T}}}, \ldots \boldsymbol{\alpha}^{(v_k)^{\mathsf{T}}} \ldots, \boldsymbol{\alpha}^{(v_M)^{\mathsf{T}}}]$ and $w_i \in \{v_k\}_{k=1}^{M}$. This property means that a segmental-level Viterbi decoding for discriminative models becomes possible. This will be discussed further in Section 6.3.2.

[2] Other methods that utilize detections of longer-term acoustic events (Zweig and Nguyen 2009) are similar in spirit to the above feature-space paradigm. These features are not discussed in this thesis.

5.1.2.1 *Log-Likelihood features*

The simplest example of $\boldsymbol{\psi}_{\boldsymbol{\lambda}}(\mathbf{O}_{i|\boldsymbol{\theta}})$ is the log-likelihood feature space

$$\boldsymbol{\psi}_{\boldsymbol{\lambda}}(\mathbf{O}) = \begin{bmatrix} \log p_{\boldsymbol{\lambda}}(\mathbf{O}|v_1) \\ \vdots \\ \log p_{\boldsymbol{\lambda}}(\mathbf{O}|v_{\mathtt{M}}) \end{bmatrix} \tag{5.10}$$

where $\boldsymbol{\lambda}$ denotes the generative model parameters, and $p_{\boldsymbol{\lambda}}(\mathbf{O}|v_k)$ is the likelihood for generative model v_k. This feature space concatenates the log-likelihoods from all models, including the correct model and competing ones, to yield additional information from the observations. Note that each element in feature $\boldsymbol{\psi}_{\boldsymbol{\lambda}}(\mathbf{O})$ is a log likelihood which can be computed using the forward-backward algorithm described in Section 2.2.1.1. Thus given a segmentation $\boldsymbol{\theta}$ the acoustic score $\boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O}, \mathbf{w}; \boldsymbol{\theta})$ in equation (5.9) can also be calculated efficiently using the forward-backward algorithm.

One elegant property of this joint feature space is that it allows the standard HMM baseline to be obtained by simply setting the value of $\boldsymbol{\alpha}$ associated with the correct model to be one and zero for all competing models (see (5.10)), i.e., the sparse vectors $\boldsymbol{\alpha}^{(v_1)} = [1\,0\ldots0]^{\mathsf{T}}, \cdots, \boldsymbol{\alpha}^{(v_{\mathtt{M}})} = [0\,0\ldots1]^{\mathsf{T}}$,

$$\boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O}, \mathbf{w}; \boldsymbol{\theta}) = \sum_{i=1}^{|\mathbf{w}|} \log p_{\boldsymbol{\lambda}}(\mathbf{O}_{i|\boldsymbol{\theta}}|w_i)$$

If the parameters associated with the correct models are not one, they can viewed as acoustic de-weighting factors. But unlike the acoustic de-weighting in (Povey 2003) where a constant scalar is used, all the weights α here are class-dependent and can be learned under any criteria discussed in the Section 4.4. In practice this property offers the discriminative models such as multi-class SVMs or SCRFs an opportunity to initialise parameters such that the HMM classification performance (Zhang *et al.* 2010) can be achieved in the first training iteration of discriminative models. Note that this may not be possible with other forms of feature-functions.

5.1.2.2 *Derivative features*

A more general form for segmental features $\boldsymbol{\psi}_{\boldsymbol{\lambda}}(\mathbf{O}_{i|\boldsymbol{\theta}})$ is the derivative features (Jaakkola *et al.* 1999; Ragni and Gales 2011*a*)

$$\boldsymbol{\psi}_{\boldsymbol{\lambda}}(\mathbf{O}) = \begin{bmatrix} \log p_{\boldsymbol{\lambda}}(\mathbf{O}|v_1) \\ \boldsymbol{\nabla}_{\boldsymbol{\lambda}} \log p_{\boldsymbol{\lambda}}(\mathbf{O}|v_1) \\ \vdots \\ \log p_{\boldsymbol{\lambda}}(\mathbf{O}|v_1) \\ \boldsymbol{\nabla}_{\boldsymbol{\lambda}} \log p_{\boldsymbol{\lambda}}(\mathbf{O}|v_{\mathtt{M}}) \end{bmatrix}. \tag{5.11}$$

In addition to log-likelihoods the feature vector in equation (5.11) incorporates derivatives with respect to generative model parameters. Consider the the HMM parameters $\boldsymbol{\lambda}_{jm} = \{\boldsymbol{\mu}_{jm}, \boldsymbol{\Sigma}_{jm}\}$ for Gaussian component c_{jm}, the derivatives taken with respect to the mean $\boldsymbol{\mu}_{jm}$ can be derived as,

$$\boldsymbol{\nabla}_{\boldsymbol{\mu}_{jm}} \log p_{\boldsymbol{\lambda}}(\mathbf{O}|v_1) = \sum_t P(c_{jm,t}|\mathbf{O}) \boldsymbol{\Sigma}_{jm}^{-\frac{1}{2}} (\boldsymbol{o}_t - \boldsymbol{\Sigma}_{jm})$$

where j is the index of state and m is the index of component in that state. These derivatives are functions of component posterior probabilities, $P(c_{jm,t}|\mathbf{O})$, which depend on the whole observation sequence of that segment (as shown in equation (2.14)). This means that the use of derivative features can relax the frame-level independent assumption to the segmental level.[3] Alternatively, if GMMs are used, then the derivative feature space yields frame-level features; the derivative with respect to the component priors, for example, yields sparse GMM posterior features (Gales *et al.* 2012; Wiesler *et al.* 2011).

Note that by properly setting the $\boldsymbol{\alpha}^{(w)}$, it is also possible to view the dot product of $\boldsymbol{\alpha}^{(w)}$ and derivative features $\boldsymbol{\psi}_{\boldsymbol{\lambda}}(\mathbf{O})$ as a Taylor series expansion of the "true" log likelihood $\log p_{\boldsymbol{\lambda}}(\mathbf{O}|w)$ (Layton 2006). This can be seen from the following expressions. Given ML-estimated HMM parameters $\hat{\boldsymbol{\lambda}}$, the "true" log likelihood can be approxim-

[3] Higher-order derivatives offer more complex dependencies (Layton 2006).

ated by a first-order Taylor-series expansion,

$$\begin{aligned}\log p_{\boldsymbol{\lambda}}(\mathbf{O}|w) &= \log p_{\boldsymbol{\lambda}}(\mathbf{O}|w)|_{\boldsymbol{\lambda}=\hat{\boldsymbol{\lambda}}} + (\boldsymbol{\lambda}-\hat{\boldsymbol{\lambda}})^{\mathsf{T}}\boldsymbol{\nabla}_{\boldsymbol{\lambda}}\log p_{\boldsymbol{\lambda}}(\mathbf{O}|w)|_{\boldsymbol{\lambda}=\hat{\boldsymbol{\lambda}}} + \cdots \\ &\approx \begin{bmatrix} \\ \boldsymbol{\alpha}^{(w)} \\ \\ \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} \log p_{\boldsymbol{\lambda}}(\mathbf{O}|w)|_{\boldsymbol{\lambda}=\hat{\boldsymbol{\lambda}}} \\ \boldsymbol{\nabla}_{\boldsymbol{\lambda}}\log p_{\boldsymbol{\lambda}}(\mathbf{O}|w)|_{\boldsymbol{\lambda}=\hat{\boldsymbol{\lambda}}} \end{bmatrix} \end{aligned} \tag{5.12}$$

The first-order approximation here is only used for illustration purposes. In general Taylor-series of any length can be applied by adding higher-order derivatives to the feature space.

Another advantage of derivative features is that they contain more discriminative information. In order to illustrate this, consider the following example (Layton 2006). A discrete HMM with the topology shown in Figure 5.3 is used to model two classes w_1 and w_2. The observation for two classes are

$$\begin{aligned} w_1: &\qquad AAAA, BBBB \\ w_2: &\qquad AABB, BBAA \end{aligned}$$

The HMM parameters can be estimated by ML criteria given these observations. The resulting state transition and output probabilities are shown in Figure 5.3. Since all estimated distributions yield equal probabilities the HMM is not capable of distinguishing between the two classes. However the situation is different with derivative features. Table 5.1 shows values of selected derivatives. When the first and second order derivatives are computed with respect to output symbol A in state 2 (line 3 and 3) then all training examples may be correctly classified.

Table 5.1 *Feature values for second-order HMM score-space*

Feature	Class w_1		Class w_2	
	AAAA	BBBB	AABB	BBAA
Log-Like	-1.11	-1.11	-1.11	-1.11
∇_{2A}	0.50	-0.50	0.33	-0.33
$\nabla_{2A}\nabla_{2A}$	-3.83	0.17	-3.28	-0.61

Although the derivative feature has many theoretical advantages, in practice it normally requires significantly more memory to compute the high dimensional $\boldsymbol{\psi}_{\boldsymbol{\lambda}}(\mathbf{O})$.

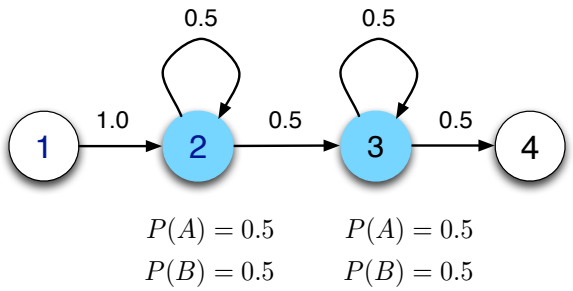


Figure 5.3 *Discrete HMM topology, transition and output probabilities.*

For example, for 39 dimension observations and 3-emitting-states HMM with 18 mixture component for each state, even just taking the derivatives with respect to means will lead to a $2106M$ dimensional features $\psi_{\boldsymbol{\lambda}}(\mathbf{O}_{i|\theta})$ for one segmentation, where M is the number of possible words or subwords in the dictionary. Although efficient methods for computing the derivative features exist (R. C. van Dalen, A. Ragni, and M. J. F. Gales 2013; Ragni and Gales 2011*b*), these derivative features are not evaluated in this thesis.

5.1.3 *Adaptation framework*

As discussed in Section 4.5, an alternative approach to adapting the discriminative classifiers to speaker and noise conditions is to modify the feature-functions. When the features are based on generative models this can be achieved by applying model-based adaptation/compensation schemes (Gales and Flego 2008). When the HMM is used as the generative model then the examples of model-based adaptation/compensation schemes include (constrained) MLLR discussed in Section 2.4.5.1 and vector Taylor series (VTS) discussed in Section 2.4.5.2.

The general feature-space adaptation/compensation framework in (Gales and Flego 2008) is illustrated by Figure 5.4. The shaded part in Figure 5.4 shows the model-based adaptation/compensation stage. Given observation sequence $\mathbf{O}$, the canonical model parameters $\boldsymbol{\lambda}'$ are modified to match the target speaker and noise conditions yielding the adapted model parameters. The score-space in the unshaded part of Figure 5.4

makes use of the adapted model to yield modified feature vectors for the discriminative classifier. The discriminative classifier examined in this framework include the SVM, multi-class SVM, SCRF and structured SVMs.

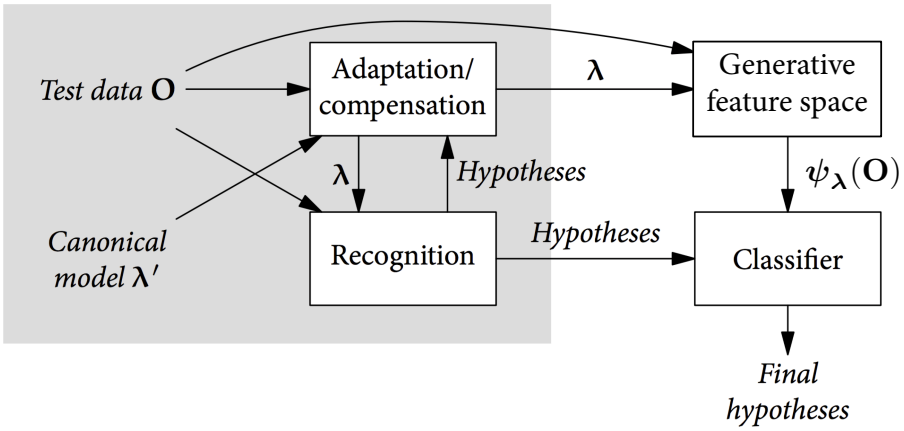


Figure 5.4 *Adaptation/compensation scheme for discriminative classifiers using feature-spaces based on generative models. The shaded region illusates the model-based adaptation/compensation stage.*

5.2 Language Features

The language features $\phi^{\mathtt{lm}}(\mathbf{w}, \boldsymbol{\theta})$ are mainly associated with state, phone or word sequences (Gales *et al.* 2012) and may provide various forms of information including lexical (Arisoy *et al.* 2010; Zweig and Nguyen 2009), syntactic (Arisoy *et al.* 2010; Collins *et al.* 2005) and semantic (Chen and Rosenfeld 1999; Khudanpur and Wu 2000). Similar to the acoustic features, the language features used in this thesis can also be categorised into frame-level (state-transition) features $\phi^{\mathtt{lm}}(\boldsymbol{\theta})$ and segment-level features $\phi^{\mathtt{lm}}(\mathbf{w})$.

The simplest form of frame-level language feature is the state transition features

used in HCRF (Gunawardana *et al.* 2005),

$$\boldsymbol{\phi}^{\mathtt{lm}}(\boldsymbol{\theta}) = \sum_{t=1}^{T} \boldsymbol{\phi}^{\mathtt{lm}}(\theta_t, \theta_{t+1}) = \sum_{t=1}^{T} \begin{bmatrix} \vdots \\ \delta(\theta_t = s_w) \\ \delta(\theta_t = s_w, \theta_{t+1} = s'_w) \\ \vdots \end{bmatrix} \forall\, s_w, s'_w \quad (5.13)$$

where s_w and s'_w are the states labels of word or subword w. This feature captures the state transition within each w.

One popular form of segmental (or supra-segmental) language features is based on the unigram and higher-order n-grams (Arisoy *et al.* 2010; Watanabe *et al.* 2010; Zweig and Nguyen 2009). For instance, features related to unigram and bigram language models can be expressed as

$$\boldsymbol{\phi}^{\mathtt{lm}}(\mathbf{w}) = \sum_{i=1}^{|\mathbf{w}|} \boldsymbol{\phi}^{\mathtt{lm}}(w_i, w_{i+1}) = \sum_{i=1}^{|\mathbf{w}|} \begin{bmatrix} \vdots \\ \delta(w_i = w) \\ \delta(w_i = w, w_{i+1} = w') \\ \vdots \end{bmatrix} \forall\, w, w' \quad (5.14)$$

where w and w' may correspond to any words or phones in the dictionary. It is also possible to apply the same concept to the segmentation and word features to represent pronunciation probability. Another example of supra-segmental language features is simply use the language model score,

$$\boldsymbol{\phi}^{\mathtt{lm}}(\mathbf{w}) = \begin{bmatrix} \log P(\mathbf{w}) \end{bmatrix} \quad (5.15)$$

where $P(\mathbf{w})$ is the language model (e.g., n-gram) probability for word sequence $\mathbf{w}$. If the parameter $\alpha^{\mathtt{lm}}$ associated to it is set to one, the original language model score can be achieved.

5.3 Joint Feature Space

To sum up, given observations $\mathbf{O} = \{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_T\}$, the corresponding labels $\mathbf{w} = \{w_1, \ldots, w_i, \ldots, w_{|\mathbf{w}|}\}$ and the segmentation $\boldsymbol{\theta}$, the joint feature $\boldsymbol{\phi}(\mathbf{O}, \mathbf{w}; \boldsymbol{\theta})$ can be

constructed to characterize the statistical dependencies of the $(\mathbf{O}, \mathbf{w})$ pair, by using the acoustic and language features described in this Chapter,

$$\boldsymbol{\phi}(\mathbf{O}, \mathbf{w}; \boldsymbol{\theta}) = \begin{bmatrix} \boldsymbol{\phi}^{\mathtt{ac}}(\mathbf{O}, \mathbf{w}; \boldsymbol{\theta}) \\ \boldsymbol{\phi}^{\mathtt{lm}}(\mathbf{w}) \end{bmatrix} \tag{5.16}$$

If the segmentation is specified at the state level, frame-level features can be applied; if the segmentation at word or subword level, then segmental features can be applied. When the frame-level or segmental features are based on generative models, applying the joint feature space offers an elegant way of combine generative and discriminative models. A summary of frame and segment-level features described here can be found in Table 5.2. One example of the joint feature space using segmental features is

$$\boldsymbol{\phi}(\mathbf{O}, \mathbf{w}; \boldsymbol{\theta}) = \begin{bmatrix} \sum_{i=1}^{|\mathbf{w}|} \delta(w_i = v_1) \boldsymbol{\psi}(\mathbf{O}_{i|\boldsymbol{\theta}}) \\ \vdots \\ \sum_{i=1}^{|\mathbf{w}|} \delta(w_i = v_{\mathtt{M}}) \boldsymbol{\psi}(\mathbf{O}_{i|\boldsymbol{\theta}}) \\ \log P(\mathbf{w}) \end{bmatrix}, \text{the corresponding } \boldsymbol{\alpha} = \begin{bmatrix} \boldsymbol{\alpha}^{(v_1)} \\ \vdots \\ \boldsymbol{\alpha}^{(v_{\mathtt{M}})} \\ \alpha^{\mathtt{lm}} \end{bmatrix} \tag{5.17}$$

where $\boldsymbol{\psi}(\mathbf{O}_{i|\boldsymbol{\theta}})$ could be the log-likelihood features in Section 5.1.2.1. This example of joint feature space will be used to evaluate the our models in the experiment (see Chapter 8).

Feature Type			Example Representation	Example papers
frame-level	$\boldsymbol{\phi}^{\mathtt{ac}}$	Gaussian statistics	$\boldsymbol{\psi}(\boldsymbol{o}_t) = \begin{bmatrix} \boldsymbol{o}_t \\ \mathrm{diag}(\boldsymbol{o}_t \boldsymbol{o}_t^{\mathsf{T}}) \end{bmatrix}$	(Gunawardana *et al.* 2005)
		MLP postierior	$\boldsymbol{\psi}(\boldsymbol{o}_t) = \begin{bmatrix} \vdots \\ P_{\mathtt{MLP}}(s\|\boldsymbol{o}_t) \\ \vdots \end{bmatrix}, \forall s$	(Morris and Fosler-Lussier 2008)
		Derivative	$\boldsymbol{\psi}(\boldsymbol{o}_t) = \begin{bmatrix} \vdots \\ \log p_{\boldsymbol{\lambda}}(\boldsymbol{o}_t\|s) \\ \boldsymbol{\nabla}_{\boldsymbol{\lambda}} \log p_{\boldsymbol{\lambda}}(\boldsymbol{o}_t\|s) \\ \vdots \end{bmatrix}, \forall s$	(Layton 2006)
	$\boldsymbol{\phi}^{\mathtt{lm}}$	state transition	$\begin{bmatrix} \vdots \\ \delta(\theta_t = s, \theta_{t+1} = s') \\ \vdots \end{bmatrix}, \forall s, s'$	(Gunawardana *et al.* 2005)
segment-level	$\boldsymbol{\phi}^{\mathtt{ac}}$	Log Likelihoods	$\boldsymbol{\psi}(\mathbf{O}) = \begin{bmatrix} \vdots \\ \log p_{\boldsymbol{\lambda}}(\mathbf{O}\|w) \\ \vdots \end{bmatrix}, \forall w$	(Zhang *et al.* 2010)
		Derivative	$\boldsymbol{\psi}(\mathbf{O}) = \begin{bmatrix} \vdots \\ \log p_{\boldsymbol{\lambda}}(\mathbf{O}\|w) \\ \boldsymbol{\nabla}_{\boldsymbol{\lambda}} \log p_{\boldsymbol{\lambda}}(\mathbf{O}\|w) \\ \vdots \end{bmatrix}, \forall w$	(Ragni and Gales 2011*b*)
		Event detector	$\boldsymbol{\psi}(\mathbf{O}) = \boldsymbol{\psi}(\mathbf{O}, w_i)$	(Zweig and Nguyen 2009)
	$\boldsymbol{\phi}^{\mathtt{lm}}$	word n-gram	$\begin{bmatrix} \vdots \\ \delta(w_i = w, w_{i+1} = w') \\ \vdots \end{bmatrix} \forall\, w, w'$	(Zweig and Nguyen 2009)

Table 5.2 *Summary of model features in common use.*

Chapter 6

Structured SVMs for Speech Recognition

6.1 Introduction to Structured SVMs

In the previous chapters two types of discriminative models: *unstructured* and *structured* discriminative models were introduced. Unstructured discriminative models, e.g. logistic regression model and support vector machines (SVMs), assume that the class labels are independent and have no structure. When applying these models to a complete utterance in continuous speech recognition, the space of possible classes becomes very large, e.g., a 6-digit length utterance yields 10^6 classes. One solution to deal with this, similar to acoustic code-breaking (Venkataramani *et al.* 2003), is to segment the continuous speech into words/sub-words observation sequences. For each segment, multi-class SVMs or logistic regression can be applied in the same fashion as an isolated classification tasks (Birkenes *et al.* 2006; Gales and Flego 2010; Zhang *et al.* 2010) as discussed in Section 3.2 and illustrated in Figure 3.9. However, this approach has two problems. First, the classification is based on one, fixed, segmentation generated by HMMs. Second, each segment is treated independently.

Training	Unstructured w	$\longrightarrow$	Structured $\mathbf{w}$
Max Likelihood	Naive Bayes $p(\mathbf{O}\|w)$	$\longrightarrow$	HMM $p(\mathbf{O}\|\mathbf{w})$
$\downarrow$	$\downarrow$		$\downarrow$
Max Conditional Likelihood	Logistic $P(w\|\mathbf{O})$	$\longrightarrow$	HCRF $P(\mathbf{w}\|\mathbf{O})$
$\downarrow$	$\downarrow$		$\downarrow$
Maximum Margin	SVM $\boldsymbol{\alpha}_w^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O})$	$\longrightarrow$	Struct SVM $\boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O},\mathbf{w})$

Table 6.1 *Summary of some unstructured and structured models for speech recognition. The gray row represents generative models and yellow rows represent discriminative models. Note that SVMs can be viewed as maximum margin trained logistic regression models (for details see Section 3.2.3). The structured SVMs can be related to maximum margin trained HCRFs/SCRFs (for details see Section 6.4).*

An alternative solution is to incorporate the structure into the model. This transforms the unstructured discriminative models to structured models. For logistic regressions, this structured extension leads to HCRFs/SCRFs as described in Chapter 4. For SVMs this yields structured SVMs which was originally proposed by (Joachims *et al.* 2009) in the machine learning field. Interestingly, it was shown that the SVMs can be related to a logistic regression model trained by the maximum margin criteria (for details see Section 3.2.3); This section will show that the structured SVMs can also be related to a maximum margin trained HCRFs/SCRFs described in Chapter 4. The relationships between commonly used unstructured and structured models are summarized here in Table 6.1. This chapter focuses on the structured SVMs (SSVM) framework and the practical issues for continuous speech recognition.

Structured Support Vector Machines Denote $\mathbf{O} = \{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_T\}$ as an observation sequence and $\mathbf{w} = \{w_1, \ldots, w_{|\mathbf{w}|}\}$ as the corresponding label sequence. In structured SVMs for continuous speech recognition, the goal is to learn a weight vector $\boldsymbol{\alpha}$. A linear discriminant function $\boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O},\mathbf{w})$ is then used to measure how well a label sequence $\mathbf{w}$ matches an observation sequence $\mathbf{O}$, such that

$$\mathbf{w}_{\boldsymbol{\alpha}} = \arg\max_{\mathbf{w}} \left\{ \boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O},\mathbf{w}) \right\} \tag{6.1}$$

is the recognized label sequence for a given $\mathbf{O}$, where $\boldsymbol{\alpha}$ is the discriminative parameter vector and $\phi(\mathbf{O}, \mathbf{w})$ is a *joint* feature vector characterizing the statistical dependencies of the $(\mathbf{O}, \mathbf{w})$ pair. Unlike multi-class SVMs where the weight vector $\boldsymbol{\alpha}_w$ is used for each class w to compute a score (Crammer and Singer 2001), SSVMs use a *joint* feature-space and a single weight vector, $\boldsymbol{\alpha}$, for the whole sentence $\mathbf{w}$.

To apply structured SVMs to large vocabulary continuous speech recognition three important decisions need to be made: the form of the joint features $\phi(\mathbf{O}, \mathbf{w})$ to use; the appropriate training criterion and efficient learning algorithm; and the efficient decoding algorithm based on the joint features. In Chapter 5 various type of features from frame level, segmental level to suprasegmental level were described. All these features can be used by structured SVMs for continuous speech recognition. The training and inference will be focuses in the following sections of this chapter.

6.2 Parameter Estimation

One important decision for structured support vector machines is the training criterion for parameter estimation. Note that there are two sets of parameters to be trained, the *generative parameters* $\boldsymbol{\lambda}$ (for the features)[1] and *discriminative parameters* $\boldsymbol{\alpha}$ (for the structured SVMs). A standard approach is to select values $\hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\alpha}}$ for the model parameters that maximise some training criterion $\mathcal{F}(\boldsymbol{\alpha}, \boldsymbol{\lambda})$,

$$\left\{\hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\alpha}}\right\} = \arg\max_{\boldsymbol{\lambda}, \boldsymbol{\alpha}} \left\{\mathcal{F}(\boldsymbol{\alpha}, \boldsymbol{\lambda})\right\}. \tag{6.2}$$

6.2.1 *Estimating Discriminative Parameters*

For simplicity, instead of training the generative and discriminative parameters together,[2] in this chapter we consider base model parameters $\boldsymbol{\lambda}$ is pre-trained using one of the criteria described in Section 2.3. Therefore the focus of this section is to train

[1] Some features such as likelihood features described in Section 5.1.2 are dependent on generative models.

[2] Training generative and discriminative parameters jointly will be discussed in Appendix A.

the discriminative parameters $\boldsymbol{\alpha}$ using a maximum margin training criterion $\mathcal{F}_{\texttt{mm}}$.

$$\hat{\boldsymbol{\alpha}} = \arg\max_{\boldsymbol{\alpha}} \left\{ \mathcal{F}_{\texttt{mm}}(\boldsymbol{\alpha}; \hat{\boldsymbol{\lambda}}) \right\}.$$

6.2.1.1 *Introducing latent variable—segmentation*

Given the training data $(\mathbf{O}^{(1)}, \mathbf{w}^{(1)}_{\texttt{ref}}), \ldots, (\mathbf{O}^{(R)}, \mathbf{w}^{(R)}_{\texttt{ref}})$ and the joint feature space, the discriminative parameters $\boldsymbol{\alpha}$ can be trained under some criterion. In Chapter 5 various forms of joint features were described. These features require a specific segmentation/alignment $\boldsymbol{\theta}$ (at word, phone, subphone or state level) as shown in Figure 6.1. Introducing the segmentation as a latent variable raises two questions: How to learn $\boldsymbol{\alpha}$ with $\boldsymbol{\theta}$ jointly in training? How to find $\mathbf{w}$ and $\boldsymbol{\theta}$ in decoding?

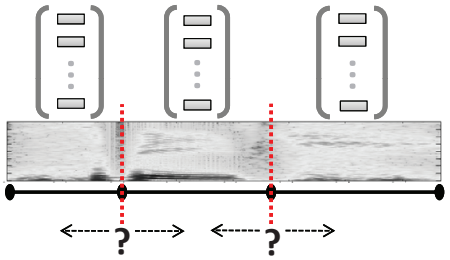


Figure 6.1 *The joint feature space depends on the segmenation/alignment.*

For decoding, in Section 6.3 we will show that given $\boldsymbol{\alpha}$ the optimal segmentation $\boldsymbol{\theta}$ can be inferred. For training, both $\boldsymbol{\alpha}$ and $\boldsymbol{\theta}$ are unknown and depend on one another. The segmentation may vary with $\boldsymbol{\alpha}$; and adjusting the segmentation will affect the optimal value of $\boldsymbol{\alpha}$. In following two sections, we will describe the maximum margin training of parameter $\boldsymbol{\alpha}$ first with the fixed segmentation $\boldsymbol{\theta}_{\boldsymbol{\lambda}}$ from HMMs. Then a joint training of $\boldsymbol{\alpha}$ and the optimal segmentation $\boldsymbol{\theta}_{\boldsymbol{\alpha}}$ is described as an extension.

6.2.1.2 *Training with fixed segmentation*

In this section we consider the "most likely" segmentation $\boldsymbol{\theta}_{\boldsymbol{\lambda}}$ generated using Viterbi search based on standard generative model HMMs $\boldsymbol{\lambda}$

$$\boldsymbol{\theta}_{\boldsymbol{\lambda}} = \arg\max_{\boldsymbol{\theta}} \; P(\boldsymbol{\theta}|\mathbf{w}) p_{\boldsymbol{\lambda}}(\mathbf{O}|\boldsymbol{\theta}, \mathbf{w}). \tag{6.3}$$

This is the typical approach also used in the SCRF (Zweig and Nguyen 2009), CAug (Layton 2006) and LLM (Zhang *et al.* 2010) based systems. Given segmentation $\boldsymbol{\theta}_{\boldsymbol{\lambda}}$ for each data pair, the joint feature space can be simplified as

$$\boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}^{(r)}) := \boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}^{(r)}; \boldsymbol{\theta}_{\boldsymbol{\lambda}}^{(r)})$$

Thus, the parameters of structured SVM can be trained by solving the following optimisation problem (Joachims *et al.* 2009):

$$\min_{\boldsymbol{\alpha}, \xi_r} \frac{1}{2}||\boldsymbol{\alpha}||^2 + \frac{C}{R}\sum_{r=1}^{R}\xi_r \tag{6.4}$$

s.t. For every utterance $r = 1, \ldots, R$,

For all competing hypothesis $\mathbf{w}_*^{(r)} \neq \mathbf{w}_{\mathtt{ref}}^{(r)}$:

$$\boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}_{\mathtt{ref}}^{(r)}) - \boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}_*^{(r)}) \geq \mathcal{L}(\mathbf{w}_{\mathtt{ref}}^{(r)}, \mathbf{w}_*^{(r)}) - \xi_r,$$

where $\xi_r \geq 0$ are the slack variables and $\mathcal{L}(\mathbf{w}_{\mathtt{ref}}^{(r)}, \mathbf{w}_*^{(r)})$ is the loss function between the reference $\mathbf{w}_{\mathtt{ref}}^{(r)}$ and its competing hypothesis $\mathbf{w}_*^{(r)}$. The constraints in equation (6.4) can be explained as follows. For every training pair $(\mathbf{O}^{(r)}, \mathbf{w}_{\mathtt{ref}}^{(r)})$, the best score of the reference pair should be greater than all competing pairs $(\mathbf{O}^{(r)}, \mathbf{w}_*^{(r)})$ by a margin determined by the loss. However the number of possible competing hypotheses $\mathbf{w}_*^{(r)}$ is huge. Therefore, the challenge is to solve an optimisation problem with a large number of constraints, although the number of active constraints that affect the solution is limited (this can be illustrated in the Figure 6.2).

Substituting the slack variable ξ_r from the constraints into the objective function, equation (6.4) can be reformulated as the *minimisation* of

$$\mathcal{F}_{\mathtt{mm}}(\boldsymbol{\alpha}; \hat{\boldsymbol{\lambda}}) = \frac{1}{2}||\boldsymbol{\alpha}||_2^2 + \frac{C}{R}\sum_{r=1}^{R}\Big[\overbrace{-\boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}_{\mathtt{ref}}^{(r)})}^{\text{linear}} + \underbrace{\max_{\mathbf{w}_*^{(r)} \neq \mathbf{w}_{\mathtt{ref}}^{(r)}} \Big\{\mathcal{L}(\mathbf{w}_{\mathtt{ref}}^{(r)}, \mathbf{w}_*^{(r)}) + \boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}_*^{(r)})\Big\}}_{\text{convex}}\Big]_+ \tag{6.5}$$

where $[\,]_+$ is the hinge-loss function. Because of the $\max(\cdot)$, the objective function is non-differentiable and non-smooth. However, the maximum of a set of linear func-

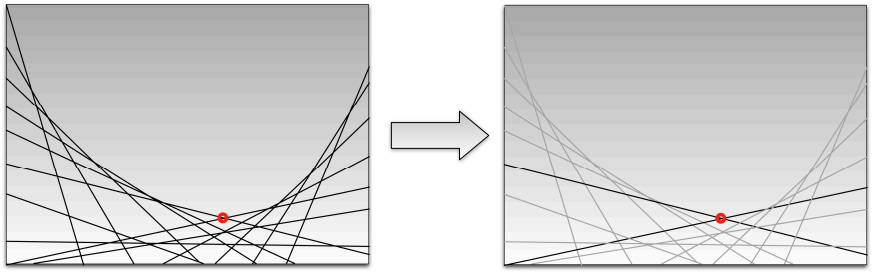


Figure 6.2 *The illustration of limited active constraints in a two-dimensional searching space. The background grey color represents the value of the objective function (darker means larger and lighter means smaller). Each line represents a linear constraint for the variable. The red circle is the optimal solution that minimises the objective function subject to the constraints. The number of active constraints that affect the sulution in this case is only two as illustrated in the right figure.*

tions is convex. Therefore, the objective function in equation (6.5) is convex as illustrated in the Figure 6.3.

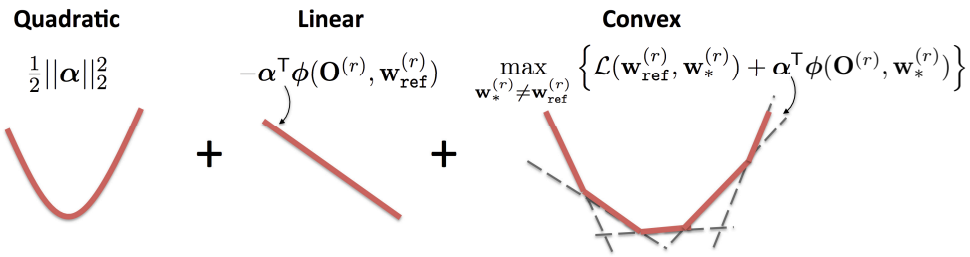


Figure 6.3 *Illustration of the convexity but non-differentiable objective function for $\boldsymbol{\alpha}$.*

The objective function in equation (6.4) is convex for $\boldsymbol{\alpha}$, however, solving this problem is not trivial because the criterion is non-differentiable. Existing algorithms for this problem fall into two groups. The first group of algorithms use generalized gradient-based approaches: subgradient methods (Singer and Srebro 2007) and exponentiated gradient methods (Globerson *et al.* 2007). The second group uses the cutting plane algorithm which does not take a single gradient step, but always takes

an optimal step in the current constraint set (Joachims *et al.* 2009; Tsochantaridis *et al.* 2005). In this work, the cutting plane algorithm summarized in Algorithm 1 is used to solve this convex optimisation problem.

Algorithm 1: n-slack Cutting plane algorithm for equation (6.4) or (6.5)

Input: $\{(\mathbf{O}^{(r)}, \mathbf{w}_{\mathtt{ref}}^{(r)})\}_{r=1}^{R}$, C and precision ε;
Initialize $\boldsymbol{\alpha}$ and empty constraint set: $\mathscr{W}_r \leftarrow \emptyset$;
repeat
  for $r = 1..R$ **do**
    */*generating best competing hypothesis*/*

$$\mathbf{w}_*^{(r)} \leftarrow \arg\max_{\mathbf{w}} \left\{ \mathcal{L}(\mathbf{w}, \mathbf{w}_{\mathtt{ref}}^{(r)}) + \boldsymbol{\alpha}^{\mathsf{T}} \boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}) \right\} \tag{6.6}$$

    if $\boldsymbol{\alpha}^{\mathsf{T}} \left[\boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}_{\mathtt{ref}}^{(r)}) - \boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}_*^{(r)}) \right] < \mathcal{L}(\mathbf{w}_{\mathtt{ref}}^{(r)}, \mathbf{w}_*^{(r)}) - \xi_r - \epsilon$ **then**
      / put it in constraint set */*
      $\mathscr{W}_r \leftarrow \mathscr{W}_r \cup \mathbf{w}_*^{(r)}$;
      / solving the n-slack QP using current constraint set */*

$$\begin{aligned} (\boldsymbol{\alpha}, \xi_r) \quad \leftarrow \quad & \min \frac{1}{2}||\boldsymbol{\alpha}||_2^2 + \frac{C}{R}\sum_{r=1}^{R}\xi_r \\ \text{s.t.} \quad & \forall\, \mathbf{w}_*^{(1)} \in \mathscr{W}_1 : \boldsymbol{\alpha}^{\mathsf{T}} \left[\boldsymbol{\phi}(\mathbf{O}^{(1)}, \mathbf{w}_{\mathtt{ref}}^{(1)}) - \boldsymbol{\phi}(\mathbf{O}^{(1)}, \mathbf{w}_*^{(1)}) \right] \geq \mathcal{L}(\mathbf{w}_{\mathtt{ref}}^{(1)}, \mathbf{w}_*^{(1)}) - \xi_1 \\ & \vdots \\ & \forall\, \mathbf{w}_*^{(R)} \in \mathscr{W}_R : \boldsymbol{\alpha}^{\mathsf{T}} \left[\boldsymbol{\phi}(\mathbf{O}^{(R)}, \mathbf{w}_{\mathtt{ref}}^{(R)}) - \boldsymbol{\phi}(\mathbf{O}^{(R)}, \mathbf{w}_*^{(R)}) \right] \geq \mathcal{L}(\mathbf{w}_{\mathtt{ref}}^{(R)}, \mathbf{w}_*^{(R)}) - \xi_R \end{aligned} \tag{6.7}$$

until *no* $\mathscr{W}_r$ *has changed during iteration*;
return $\boldsymbol{\alpha}$

Note that this algorithm includes multiple slack variables (each for a training example). Thus it was normally referred as n-slack cutting plane algorithm (Joachims *et al.* 2009). The basic concept of the algorithm is very simple. It iteratively constructs a constraint set $\mathscr{W} = \{\mathscr{W}_1, \ldots, \mathscr{W}_R\}$, starting with an empty set $\mathscr{W} = \emptyset$. The algorithm iterates through the training examples and finds the constraint that is violated most by the current solution $\{\boldsymbol{\alpha}, \xi_r\}$. In this work, this constraint is referred as the *most violated constraint*. The corresponding hypothesis $\mathbf{w}_*^{(r)}$ is referred as the *best competing hypothesis*. If this constraint is violated by more than the desired precision

ε, the constraint is added to constraint set $\mathscr{W}$ and the quadratic programming (6.7) is solved over the extended $\mathscr{W}$. The algorithm stops when no constraint is added in the previous iteration. This means that all constraints in problem (6.4) are satisfied up to a precision of ε. The algorithm is provably efficient (Joachims *et al.* 2009) as long as the best competing hypothesis $\mathbf{w}_*^{(r)}$ can be found efficiently. Inferring the best competing hypothesis will be discussed in Section 6.3.

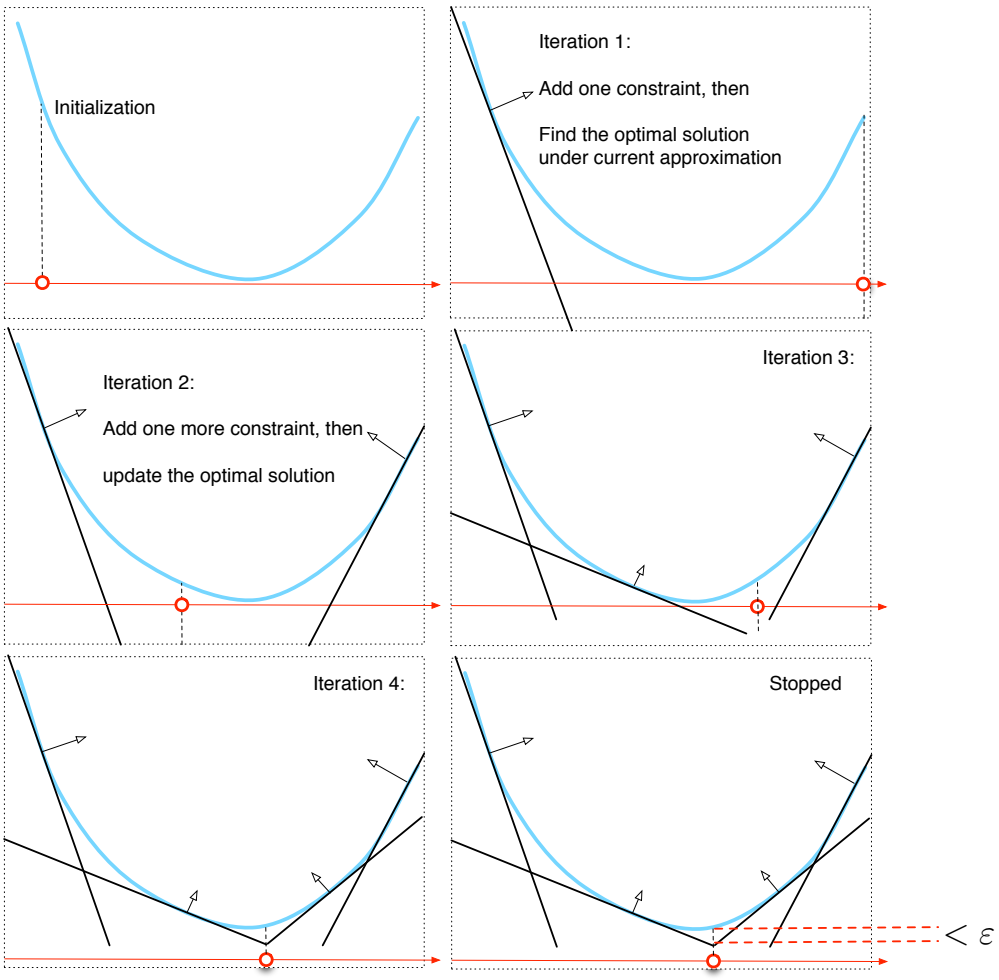


Figure 6.4 *Illustration of cutting plane algorithm in a one-dimensional optimisation problem: the light blue curve is the original objective function, while the black straight lines are the cutting planes and the red circles are the optimal solution under the current approximation .*

Figure 6.4 illustrates the successive steps in a cutting plane algorithm using a simple 1-dimensional optimisation problem. The left top figure shows the true objective function and the circle denotes our starting point. The starting point and the searching space of variable can be determined based on its prior distribution. This will be discussed in details in Section 6.5.2. In the first iteration the first cutting plane is generated, which is a linear approximation of the true objective function. Minimizing this linear approximation leads us to the current solution which is indicated by the red circle (in this simple illustration, the regularizer is ignored and the searching space is bounded). In the second iteration a second cutting plane is generated. The new approximated function now becomes a maximum over the two linear functions. By generating new cutting planes the approximation is improved, as illustrated in iteration 3 and 4. Note that in this process although the minimum value of approximated function (black lines) in each iteration is increasing , it gradually approaching the minimum value of objective function (blue curve). The cutting plane algorithm will stop when the difference between the true objective and the cutting plane approximation is less than a small value ε.

6.2.1.3 Training with optimal segmentation

In the previous section it has been shown that given the "most likely" segmentation $\boldsymbol{\theta}_{\boldsymbol{\lambda}}$, the discriminative parameter $\boldsymbol{\alpha}$ can be estimated. Although $\boldsymbol{\theta}_{\boldsymbol{\lambda}}$ is the optimal segmentation for the *generative* model, it may not be the best segmentation to characterize the dependencies on $(\mathbf{O}, \mathbf{w})$ pair for the *discriminative* model. The optimal segmentation may vary with $\boldsymbol{\alpha}$; and adjusting the segmentation will affect the optimal value of $\boldsymbol{\alpha}$. In this section, we consider both $\boldsymbol{\alpha}$ and $\boldsymbol{\theta}$ as unknown variables that depend on one another during training. The joint training algorithm of the structured SVM and the optimal segmentation is described with a maximum margin criterion below.

If the segmentation is optimized during training, it is necessary to modify the original structured SVM training algorithm in a similar fashion to the latent SVMs

(Felzenszwalb *et al.* 2010; Yu and Joachims 2009). The parameters and latent variables can be jointly trained by solving the following optimisation problem:

$$\begin{aligned} \min_{\boldsymbol{\alpha},\boldsymbol{\xi}} \quad & \frac{1}{2}||\boldsymbol{\alpha}||^2 + \frac{C}{R}\sum_{r=1}^{R}\xi_r \\ \text{s.t.} \quad & \text{For every utterance } r = 1,\ldots,R, \\ & \quad \text{For all competing hypothesis } \mathbf{w}_*^{(r)} \neq \mathbf{w}_{\mathtt{ref}}^{(r)} : \\ & \max_{\boldsymbol{\theta}^{(r)}} \left\{ \boldsymbol{\alpha}^{\mathsf{T}} \boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}_{\mathtt{ref}}^{(r)}; \boldsymbol{\theta}^{(r)}) \right\} - \max_{\boldsymbol{\theta}_*^{(r)}} \left\{ \boldsymbol{\alpha}^{\mathsf{T}} \boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}_*^{(r)}; \boldsymbol{\theta}_*^{(r)}) \right\} \geq \mathcal{L}(\mathbf{w}_{\mathtt{ref}}^{(r)}, \mathbf{w}_*^{(r)}) - \xi_r, \end{aligned} \tag{6.8}$$

Comparing with optimisation (6.4), the linear scoring function $\boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}_{\mathtt{ref}}^{(r)})$ in the constraints is replaced as $\max_{\boldsymbol{\theta}^{(r)}} \left\{ \boldsymbol{\alpha}^{\mathsf{T}} \boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}_{\mathtt{ref}}^{(r)}; \boldsymbol{\theta}^{(r)}) \right\}$, where

$$\boldsymbol{\theta}_{\boldsymbol{\alpha}}^{(r)} = \arg\max_{\boldsymbol{\theta}^{(r)}} \left\{ \boldsymbol{\alpha}^{\mathsf{T}} \boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}_{\mathtt{ref}}^{(r)}; \boldsymbol{\theta}^{(r)}) \right\}$$

is the optimal segmentation of utterance $(\mathbf{O}^{(r)}, \mathbf{w}_{\mathtt{ref}}^{(r)})$ given the current parameter $\boldsymbol{\alpha}$ and feature function $\boldsymbol{\phi}(\cdot)$. Substituting the slack variable ξ_r from the constraints into the objective function, equation (6.8) can be reformulated as the *minimisation* of

$$\mathcal{F}_{\mathtt{lm}}(\boldsymbol{\alpha}; \hat{\boldsymbol{\lambda}}) = \frac{1}{2}||\boldsymbol{\alpha}||_2^2 + \frac{C}{R}\sum_{r=1}^{R}\Big[\overbrace{-\max_{\boldsymbol{\theta}^{(r)}} \left(\boldsymbol{\alpha}^{\mathsf{T}} \boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}_{\mathtt{ref}}^{(r)}; \boldsymbol{\theta}^{(r)}) \right)}^{\text{concave}} + \underbrace{\max_{\mathbf{w}\neq\mathbf{w}_{\mathtt{ref}}^{(r)}, \boldsymbol{\theta}} \left\{ \mathcal{L}(\mathbf{w}_{\mathtt{ref}}^{(r)}, \mathbf{w}) + \boldsymbol{\alpha}^{\mathsf{T}} \boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}; \boldsymbol{\theta}) \right\}}_{\text{convex}} \Big]_+ \tag{6.9}$$

Note that the objective function in equation (6.9) comprises concave and convex parts as shown in Figure 6.5. The optimization problem has become non-convex after introducing the optimal segmentation $\boldsymbol{\theta}$, which is a common issue when working with latent variables.

To solve this non-convex optimisation problem with respective to $\boldsymbol{\alpha}$ in equation (6.9), an algorithm based on the concave-convex procedure (CCCP) (Felzenszwalb *et al.* 2010; Yuille *et al.* 2002) is proposed in Algorithm 2.[3] The algorithm is very intuitive. It works similar to the iterative process of EM. It alternates between optimizing the

[3]Note that in order to make the concave and convex terms in equation (6.9)independent to each other wichis required by CCCP, the loss function $\mathcal{L}(\mathbf{w}_{\mathtt{ref}}^{(r)}, \mathbf{w})$ should be independent to $\boldsymbol{\theta}^{(r)}$

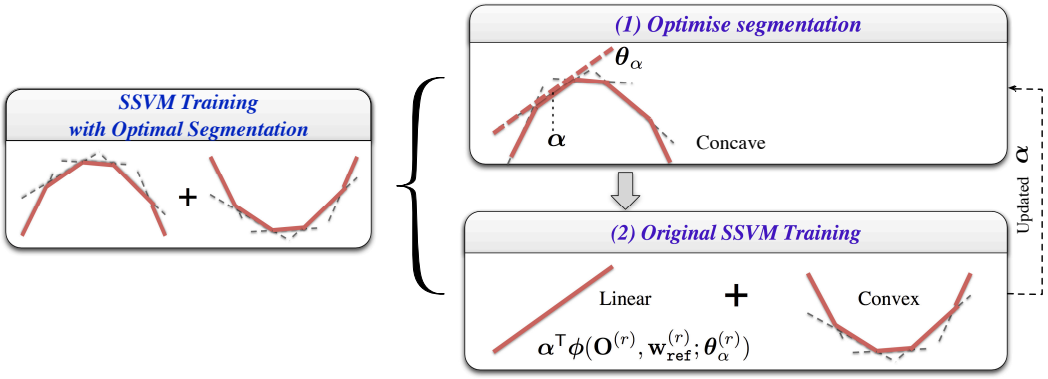


Figure 6.5 *The illustration of Step 1 and 2 in Algorithm 2 for joint learning the parameters $\boldsymbol{\alpha}$ of structured SVM and optimal segmentation $\boldsymbol{\theta}_{\boldsymbol{\alpha}}$.*

Algorithm 2: Structured SVM learning algorithm with optimal segmentation.

0. Initial: $\boldsymbol{\alpha} = [1, 0, 0 \ldots]$ and $\boldsymbol{\theta}^{(r)} = \boldsymbol{\theta}_{\boldsymbol{\lambda}}^{(r)}$;

1. **Fixing $\boldsymbol{\alpha}$, optimise** the reference segmentation $\boldsymbol{\theta}^{(r)}$ for each training pair $(\mathbf{O}^{(r)}, \mathbf{w}_{\mathtt{ref}}^{(r)})$ (using the Viterbi-style algorithm in Section 6.2.1.3):

$$\boldsymbol{\theta}_{\boldsymbol{\alpha}}^{(r)} = \arg\max_{\boldsymbol{\theta}^{(r)}} \left\{ \boldsymbol{\alpha}^\top \boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}_{\mathtt{ref}}^{(r)}; \boldsymbol{\theta}^{(r)}) \right\}, \ \forall\, r \tag{6.10}$$

2. **Fixing $\boldsymbol{\theta}_{\boldsymbol{\alpha}}^{(r)}$, optimise $\boldsymbol{\alpha}$** by *minimizing* the following convex upper bound (equation (6.9) $\leq$ (6.11)), using the cutting plane algorithm in Alg. 1 or Alg. 6:

$$\frac{1}{2}||\boldsymbol{\alpha}||_2^2 + \frac{C}{R}\sum_{r=1}^{R}\Big[\overbrace{-\boldsymbol{\alpha}^\top \boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}_{\mathtt{ref}}^{(r)}; \boldsymbol{\theta}_{\boldsymbol{\alpha}}^{(r)})}^{\text{linear}} + \underbrace{\max_{\mathbf{w}\neq\mathbf{w}_{\mathtt{ref}}^{(r)}, \boldsymbol{\theta}} \left\{ \mathcal{L}(\mathbf{w}_{\mathtt{ref}}^{(r)}, \mathbf{w}) + \boldsymbol{\alpha}^\top \boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}; \boldsymbol{\theta}) \right\}}_{\text{convex}} \Big]_+ \tag{6.11}$$

3. go back to Step 1 until converge;
return $\boldsymbol{\alpha}$;

latent variable $\boldsymbol{\theta}$ that explain the training pair $(\mathbf{O}^{(r)}, \mathbf{w}_{\mathtt{ref}}^{(r)})$ and solving the structural SVM optimization problem while treating the latent variables as completely observed. First, the optimal reference segmentation $\boldsymbol{\theta}_{\boldsymbol{\alpha}}^{(r)}$ for the current parameter $\boldsymbol{\alpha}$ is found in Step 1. This corresponds to finding the linear upper bound of the concave term of equation (6.38) as shown in Figure 6.5. Second, with the current reference seg-

mentation $\boldsymbol{\theta}_{\boldsymbol{\alpha}}^{(r)}$, the optimal value of $\boldsymbol{\alpha}$ based on (6.11) is found. These two steps can then be repeated. The procedures are illustrated in Figure 6.5. Note that the equation (6.11) in step 2 is a convex optimisation equivalent to the equation (6.5). There is no fundamental difference between $\max_{\mathbf{w},\boldsymbol{\theta}}\{\}$ and $\max_{\mathbf{w}}\{\}$ except the searching space. The searching algorithm for this will be discussed in Section 6.3. Thus equation (6.11) can be solved using Algorithm 1 described in the previous section.[4]

6.2.2 *Estimating Generative Parameters*

In the previous section, generative model parameters $\boldsymbol{\lambda}$ were fixed to create a static joint feature space. This allowed the discriminative parameters $\boldsymbol{\alpha}$ to be estimated using maximum margin training criteria. Any training criteria described in Section 2.3 can be applied to estimate the generative model parameters $\hat{\boldsymbol{\lambda}}$, e.g., the maximum likelihood criterion in equation (2.28), the MPE criterion in equation (2.33) or the maximum margin criterion in equation (2.36). The procedure of training generative models and structured SVMs is summarized in Figure 6.6. The joint estimation of the generative model and structured SVM parameters is described in the Appendix A as the future work.

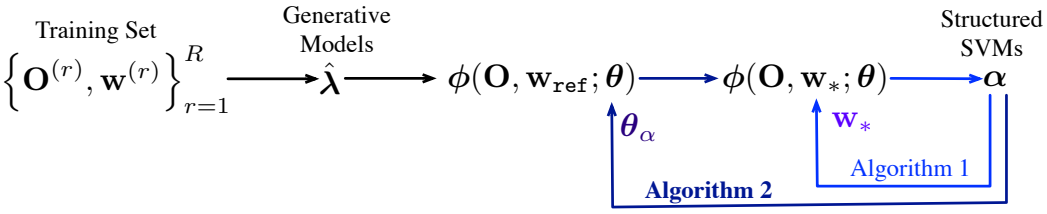


Figure 6.6 *The training process for generative models and structured SVMs, where $\mathbf{w}_*$ and $\boldsymbol{\theta}_{\boldsymbol{\alpha}}$ represent the best competing hypothesis and optimal segmentation, respectively, given the current $\boldsymbol{\alpha}$.*

[4] The equation (6.11) can also be solved using the more efficient 1-slack cutting plane algorithm (Alg. 6) described in Section 6.5.3.

6.3 Inference

Theoretically, the Algorithm 2 described in the previous section can be directly applied for model training. In practice, however, it is necessary to handle the exponentially large searching space of all the possible $\mathbf{w}$ and $\boldsymbol{\theta}$ in the following two issues during the training (for details see Algorithm 2):

- Inferring the optimal segmentation:

$$\max_{\boldsymbol{\theta}^{(r)}} \left\{ \boldsymbol{\alpha}^{\mathsf{T}} \boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}_{\mathtt{ref}}^{(r)}; \boldsymbol{\theta}^{(r)}) \right\}. \tag{6.12}$$

- Inferring the best competing hypothesis:

$$\max_{\mathbf{w} \neq \mathbf{w}_{\mathtt{ref}}, \boldsymbol{\theta}} \left\{ \mathcal{L}(\mathbf{w}, \mathbf{w}_{\mathtt{ref}}^{(r)}) + \boldsymbol{\alpha}^{\mathsf{T}} \boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}; \boldsymbol{\theta}) \right\}. \tag{6.13}$$

A similar problem arises in the decoding process of structured SVMs:

- Decoding with optimal segmentation:

$$\left\{ \mathbf{w}_{\boldsymbol{\alpha}}, \boldsymbol{\theta}_{\boldsymbol{\alpha}} \right\} = \arg\max_{\mathbf{w}, \boldsymbol{\theta}} \left\{ \boldsymbol{\alpha}^{\mathsf{T}} \boldsymbol{\phi}(\mathbf{O}, \mathbf{w}; \boldsymbol{\theta}) \right\} \tag{6.14}$$

In SCRF (Zweig and Nguyen 2009), CAug (Layton 2006) and log-linear model (Zhang *et al.* 2010) based systems, the segmentations, $\boldsymbol{\theta}_{\boldsymbol{\lambda}}$, are typically generated using standard generative model HMMs. These segmentations are fixed for both decoding and training. For inference this yields

$$\mathbf{w}_{\boldsymbol{\alpha}} = \arg\max_{\mathbf{w}} \left\{ \boldsymbol{\alpha}^{\mathsf{T}} \boldsymbol{\phi}(\mathbf{O}, \mathbf{w}; \boldsymbol{\theta}_{\boldsymbol{\lambda}}) \right\}, \tag{6.15}$$

$$\text{where } \boldsymbol{\theta}_{\boldsymbol{\lambda}} = \arg\max_{\boldsymbol{\theta}} \; P(\boldsymbol{\theta}|\mathbf{w}) p_{\boldsymbol{\lambda}}(\mathbf{O}|\boldsymbol{\theta}, \mathbf{w}). \tag{6.16}$$

Equation (6.16) can be solved using the Viterbi algorithm described in Section 2.2.2. Although $\boldsymbol{\theta}_{\boldsymbol{\lambda}}$ is the optimal segmentation for the *generative* model, it may not be the best segmentation to characterize the dependencies on $(\mathbf{O}, \mathbf{w})$ pair for the *discriminative* model. There is thus a potential mismatch between (6.15) and (6.16). The segmentation variable $\boldsymbol{\theta}$ should be optimised based on the discriminative models. Thus

the decoding formula (6.15) becomes (6.14). Essentially, equations (6.12), (6.13) and (6.14) are the same inference problem.[5] This section focuses on the decoding problem in equation (6.14). The remaining two problems can be solved by easily extending the algorithms we described below.[6]

6.3.1 *Inference with frame-level features*

Given a general joint feature space in the form of equation (5.16), the maximisation in equation (6.14) can be expressed as

$$\left\{\mathbf{w}_{\boldsymbol{\alpha}}, \boldsymbol{\theta}_{\boldsymbol{\alpha}}\right\} = \arg\max_{\mathbf{w}} \left\{ \max_{\boldsymbol{\theta} \in \boldsymbol{\Theta}_{\mathbf{w}}^{T}} \boldsymbol{\alpha}^{\mathtt{ac}\mathsf{T}} \boldsymbol{\phi}^{\mathtt{ac}}(\mathbf{O}, \mathbf{w}; \boldsymbol{\theta}) + \boldsymbol{\alpha}^{\mathtt{lm}\mathsf{T}} \boldsymbol{\phi}^{\mathtt{lm}}(\mathbf{w}) \right\} \tag{6.17}$$

where $\boldsymbol{\Theta}_{\mathbf{w}}^{T}$ is the set of all valid state sequences of $\mathbf{w}$ that have length T. If the feature function $\boldsymbol{\phi}^{\mathtt{ac}}(\mathbf{O}, \mathbf{w}; \boldsymbol{\theta})$ is at the frame level as described in Section 5.1.1, substituting equations (5.2), (5.13) and (5.14) into the (6.17) yields

$$\left\{\mathbf{w}_{\boldsymbol{\alpha}}, \boldsymbol{\theta}_{\boldsymbol{\alpha}}\right\} = \arg\max_{\mathbf{w}, \theta_1, \ldots, \theta_T} \left\{ \sum_{t=1}^{T} \boldsymbol{\alpha}^{\mathtt{ac}\mathsf{T}} \boldsymbol{\phi}^{\mathtt{ac}}(\mathbf{o}_t, \theta_t) + \boldsymbol{\alpha}^{\mathtt{lm}\mathsf{T}} \boldsymbol{\phi}^{\mathtt{lm}}(\mathbf{w}, \boldsymbol{\theta}) \right\} \tag{6.18}$$

where $\theta_t \in s^w$ is the state label of frame t, and s^w indicates a state of the word (or subword) w in $\mathbf{w}$. For the frame-level features in equation (5.2) expression in (6.18) is related to HMM inference (see Section 2.2.2). The search process (6.18) can be split into two part. First, for each frame t, similar to the calculation of emission probabilities for HMMs, the acoustic scores $\boldsymbol{\alpha}^{\mathtt{ac}\mathsf{T}} \boldsymbol{\phi}^{\mathtt{ac}}(\mathbf{o}_t, \theta_t)$ need to be computed. The complexity of computing the frame scores $\boldsymbol{\alpha}^{\mathtt{ac}\mathsf{T}} \boldsymbol{\phi}^{\mathtt{ac}}(\mathbf{o}_t, \theta_t)$ depends on the form of features. If the HCRF features $\boldsymbol{\phi}^{\mathtt{ac}}(\mathbf{o}_t, \theta_t)$ in equation (5.6) are used, the frame scores can be related to the HMM frame-level emission probabilities by properly setting the parameter $\boldsymbol{\alpha}^{\mathtt{ac}}$ (Heigold *et al.* 2007). The second part is to obtain the hidden state sequence, $\{\theta_1, \ldots, \theta_T\}$ and word labels, $\{w_1, \ldots, w_{|\mathbf{w}|}\}$. This can be efficiently implemented using the Viterbi algorithm (Viterbi 1982).

[5] The equation (6.12) is a special case of equation (6.14), and equation (6.14) is actually the equation (6.13) without the loss function.

[6] Incorporating the loss function to extend the algorithm will be discussed in Section 6.5.4.2.

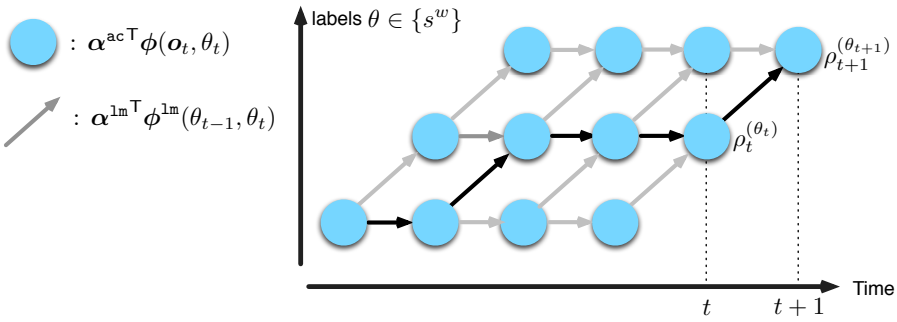


Figure 6.7 *Inference of structured SVMs with frame-level features. Each blue circle represents a frame-level acoustic score* $\boldsymbol{\alpha}^{\mathtt{ac}\mathsf{T}}\boldsymbol{\phi}^{\mathtt{ac}}(\mathbf{o}_t, \theta_t)$ *for state* θ_t*. Each arrow represents a language score* $\boldsymbol{\alpha}^{\mathtt{lm}\mathsf{T}}\boldsymbol{\phi}^{\mathtt{lm}}(\theta_{t-1}, \theta_t)$*.*

The search process is illustrated in the Figure 6.7. The best score for a state sequence $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_t\}$ ending with θ_t at time t is stored as $\rho_t^{(\theta_t)}$,

$$\rho_t^{(\theta_t)} = \max_{\theta_1,\ldots,\theta_{t-1}} \left\{ \sum_{\tau=1}^{t} \boldsymbol{\alpha}^{\mathtt{ac}\mathsf{T}}\boldsymbol{\phi}^{\mathtt{ac}}(\mathbf{o}_\tau, \theta_\tau) + \boldsymbol{\alpha}^{\mathtt{lm}\mathsf{T}}\boldsymbol{\phi}^{\mathtt{lm}}(\mathbf{w}, \boldsymbol{\theta}) \right\} \tag{6.19}$$

Given time $t+1$ and corresponding label θ_{t+1}, the acoustic score $\boldsymbol{\alpha}^{\mathtt{ac}\mathsf{T}}\boldsymbol{\phi}^{\mathtt{ac}}(\mathbf{o}_{t+1}, \theta_{t+1})$ and the language score $\boldsymbol{\alpha}^{\mathtt{lm}\mathsf{T}}\boldsymbol{\phi}^{\mathtt{lm}}(\theta_t, \theta_{t+1})$ for each w_t are computed. The best score for a label sequence ending with θ_{t+1} can then be expressed in the recursive form,

$$\rho_{t+1}^{(\theta_{t+1})} = \max_{\theta_t \in \{s^w\}} \left\{ \rho_t^{(\theta_t)} + \boldsymbol{\alpha}^{\mathtt{ac}\mathsf{T}}\boldsymbol{\phi}^{\mathtt{ac}}(\mathbf{o}_{t+1}, \theta_{t+1}) + \boldsymbol{\alpha}^{\mathtt{lm}\mathsf{T}}\boldsymbol{\phi}^{\mathtt{lm}}(\theta_t, \theta_{t+1}) \right\}. \tag{6.20}$$

Note that the language score $\boldsymbol{\alpha}^{\mathtt{lm}\mathsf{T}}\boldsymbol{\phi}^{\mathtt{lm}}(\theta_t, \theta_{t+1})$ could be related to the state transition probabilities and word bigram probabilities[7] as described in Section 5.2. By running the above Viterbi search from time 1 to T the optimal label sequence and segmentation can be obtained by tracing back the frame-level labels that maximising $\rho_T^{(\theta_T)}$. The complexity of the above searching process is $\mathcal{O}(MT)$ where M is the number of hidden states (or subphones) of all the words in the dictionary. Pruning options like beam pruning (Young *et al.* 2006) can be directly applied.

[7] Bigram probabilities can be captured by the state transition between the end of each word to the start of the next word.

6.3.2 *Inference with segmental features*

If the segmental features $\boldsymbol{\phi}^{\mathtt{ac}}(\mathbf{O}, \mathbf{w}; \boldsymbol{\theta})$ described in Section 5.1.2 are used, substituting equation (5.7) into (6.17) yields

$$\begin{aligned}\{\mathbf{w}_{\boldsymbol{\alpha}}, \boldsymbol{\theta}_{\boldsymbol{\alpha}}\} &= \arg\max_{\mathbf{w},\boldsymbol{\theta}} \left\{ \boldsymbol{\alpha}^{\mathsf{T}} \boldsymbol{\phi}(\mathbf{O}, \mathbf{w}; \boldsymbol{\theta}) \right\} \\ &= \arg\max_{w_1,\ldots,w_{|\mathbf{w}|}} \left\{ \max_{\boldsymbol{\theta}} \sum_{i=1}^{|\mathbf{w}|} \boldsymbol{\alpha}^{\mathtt{ac}\mathsf{T}} \boldsymbol{\phi}^{\mathtt{ac}}(\mathbf{O}_{i|\boldsymbol{\theta}}, w_i) + \boldsymbol{\alpha}^{\mathtt{lm}\mathsf{T}} \boldsymbol{\phi}^{\mathtt{lm}}(\mathbf{w}) \right\} \end{aligned} \tag{6.21}$$

The search process (6.21) can be split into two distinct terms. First given a segment (e.g., the i-th segment $i|\boldsymbol{\theta}$), the acoustic score $\boldsymbol{\alpha}^{\mathtt{ac}\mathsf{T}} \boldsymbol{\phi}^{\mathtt{ac}}(\mathbf{O}_{i|\boldsymbol{\theta}}, w_i)$ and the language score $\boldsymbol{\alpha}^{\mathtt{lm}\mathsf{T}} \boldsymbol{\phi}^{\mathtt{lm}}(\mathbf{w})$ need to be computed. Note that the language features $\boldsymbol{\phi}^{\mathtt{lm}}(\mathbf{w})$ only depends on the label sequences, e.g., the n-gram word features are independent of segmentation $\boldsymbol{\theta}$. The second is obtaining the optimal segmentation which requires a modified two-stage Viterbi search. This process is illustrated in Figure 6.8.

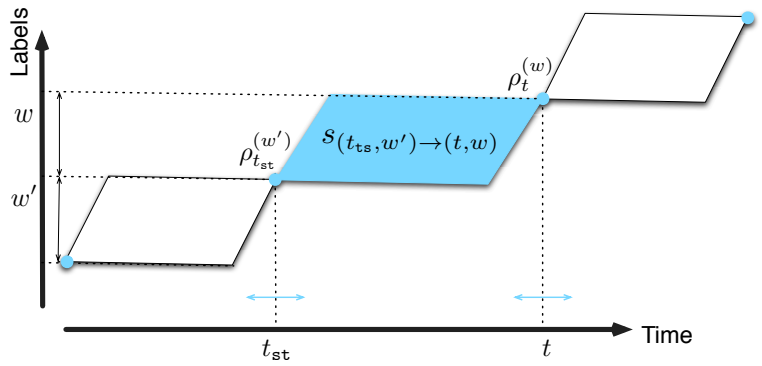


Figure 6.8 *Inference of structured SVMs with segment-level features. The points in blue represent the segment boundaries. The blue area* $s_{(t_{\mathtt{st}},w')\to(t,w)} = \boldsymbol{\alpha}^{\mathtt{ac}\mathsf{T}} \boldsymbol{\phi}^{\mathtt{ac}}(\mathbf{O}_{t_{\mathtt{ac}}:t}, w) + \boldsymbol{\alpha}^{\mathtt{lm}\mathsf{T}} \boldsymbol{\phi}^{\mathtt{lm}}(w', w)$ *is the score for segment* $(t_{\mathtt{ts}}, w') : (t, w)$.

This search process is similar to a semi-Markov search process (Fox 1968; Sarawagi and Cohen 2005). The best score (and alignment history) for a label sequence $\mathbf{w}$ ending with w' at time $t_{\mathtt{st}}$ is stored as $\rho_{t_{\mathtt{st}}}^{(w')}$,

$$\rho_{t_{\mathtt{st}}}^{(w')} = \max_{\mathbf{w},\boldsymbol{\theta}} \boldsymbol{\alpha}^{\mathsf{T}} \boldsymbol{\phi}(\mathbf{O}_{1:t_{\mathtt{st}}}, \mathbf{w}; \boldsymbol{\theta}). \tag{6.22}$$

Given time $t_{\mathtt{st}}$ (the start time of current decoding segment), the forward segmental score up-to time t for model w, is computed at the end state of that model,

$$s_{(t_{\mathtt{st}},w')\to(t,w)} = \boldsymbol{\alpha}^{\mathtt{ac}\mathsf{T}}\boldsymbol{\phi}^{\mathtt{ac}}(\mathbf{O}_{t_{\mathtt{ac}}:t}, w) + \boldsymbol{\alpha}^{\mathtt{lm}\mathsf{T}}\boldsymbol{\phi}^{\mathtt{lm}}(w', w) \tag{6.23}$$

The best score for a label sequence ending with w at time t can then be expressed as

$$\rho_t^{(w)} = \max_{t_{\mathtt{st}},w'} \left\{ \rho_{t_{\mathtt{st}}}^{(w')} + s_{(t_{\mathtt{ts}},w')\to(t,w)} \right\}, \tag{6.24}$$

where $s_{(t_{\mathtt{st}},w')\to(t,w)}$ is the score for segment $(t_{\mathtt{ts}}, w') : (t, w)$ which can be computed using the forward-backward algorithm[8] and $t_{\mathtt{st}} \in [0, t-1]$. By running the above Viterbi-style search from time 1 to T the optimal sentence and segmentation can be obtained by tracing back the model and time index maximising $\rho_T^{(w_{|\mathbf{w}|})}$. This decoding process is summarized in the Algorithm 3.

The complexity of the above process is $\mathcal{O}(MT^2)$, where M is the number of words or subwords in the dictionary. Pruning options are available, e.g., limiting w' in (6.24) to the top N models with highest scores and constraint the look-back time $t_{\mathtt{st}}$ to last $\mathcal{T}$ frames. Additionally, more efficient approximations, e.g., Gibbs sampling and variational methods (Ghahramani and Jordan 1997), could be used to reduce the computation load. However these are not investigated in this work.

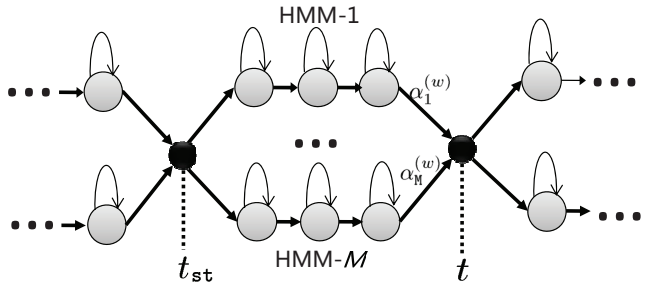


Figure 6.9 *Inference of structured SVMs with log-likelihood features in equation* (5.10). *The black circles indicate the synchronisation points where the M HMM log likelihoods and language model score are merged.*

[8] If the log likelihood features in (5.10) are used, this acoustic score can be computed using the standard forward-backward algorithm for HMMs as describeed in Section 5.1.2.1. If the derivative features in (5.11) are used, the score can be computed using the algorithm described in (R. C. van Dalen, A. Ragni, and M. J. F. Gales 2013).

Algorithm 3: Inference with segment-level features.

Input: observations $\mathbf{O}$
Output: word sequence $\hat{\mathbf{w}}$ with confidence score

/* Initialization */
for *each* w **do**
  $\rho_0^{(w)} \leftarrow 0$

/* Forward propagation */
for $t = 1, \ldots, T$ **do**
  for *each* w **do**
    for $t_{\mathtt{st}} = 1, \ldots, t-1$ **do**
      compute segmental score:
        $s_{(t_{\mathtt{st}},w')\rightarrow(t,w)} = \boldsymbol{\alpha}^{\mathtt{ac}\mathsf{T}}\boldsymbol{\phi}^{\mathtt{ac}}(\mathbf{O}_{t_{\mathtt{ac}}:t}, w) + \boldsymbol{\alpha}^{\mathtt{lm}\mathsf{T}}\boldsymbol{\phi}^{\mathtt{lm}}(w', w)$
    update best path score:
      $\rho_t^{(w)} \leftarrow \max_{t_{\mathtt{st}},w'} \left\{ \rho_{t_{\mathtt{st}}}^{(w')} + s_{(t_{\mathtt{ts}},w')\rightarrow(t,w)} \right\}$
    save best previous segment:
      $\mathrm{Prev}(t, w) \leftarrow \arg\max_{t_{\mathtt{st}},w'} \left\{ \rho_{t_{\mathtt{st}}}^{(w')} + s_{(t_{\mathtt{ts}},w')\rightarrow(t,w)} \right\}$

/* Backward trace */
$w \leftarrow \arg\max_{w'} \rho_T^{(w')}$
$(t, w) \leftarrow \mathrm{Prev}(T, w)$
while $t \neq 1$ **do**
  retrive $(t, w) \leftarrow \mathrm{Prev}(t, w)$, save word: $\hat{\mathbf{w}} \leftarrow [w, \hat{\mathbf{w}}]$
return $\hat{\mathbf{w}}$, $\max_{w'} \rho_T^{(w')}$

It may also interesting to point out that, if the log-likelihood feature in equation (5.10) is used, the expression in (6.21) can be related to the factorial HMM inference (Ghahramani and Jordan 1997),

$$\{\mathbf{w}_{\boldsymbol{\alpha}}, \boldsymbol{\theta}_{\boldsymbol{\alpha}}\} = \arg\max_{\mathbf{w}} \left\{ \max_{\boldsymbol{\theta}} \sum_{i=1}^{|\mathbf{w}|} \sum_{k=1}^{M} \alpha_k^{(w_i)} \log p_{\boldsymbol{\lambda}}(\mathbf{O}_{i|\boldsymbol{\theta}} | v_k) + \boldsymbol{\alpha}^{\mathtt{lm}\mathsf{T}} \boldsymbol{\phi}^{\mathtt{lm}}(\mathbf{w}) \right\}$$

where $\alpha_k^{(w_i)}$ is the k-th element in $\boldsymbol{\alpha}^{(w_i)}$, $\boldsymbol{\alpha}^{\mathtt{ac}} = [\boldsymbol{\alpha}^{(v_1)\mathsf{T}}, \ldots \boldsymbol{\alpha}^{(v_{\mathtt{k}})\mathsf{T}} \ldots, \boldsymbol{\alpha}^{(v_{\mathtt{M}})\mathsf{T}}]^{\mathsf{T}}$, $w_i \in \{v_k\}_{k=1}^M$, and $\{v_k\}_{k=1}^M$ is the dictionary. Thus the first step of search process, computing the segmental scores, can be achieved by the standard HMM forward-backward algorithm. The search process can be illustrated in Figure 6.9. The M phone

HMMs are shown in parallel with *synchronisation points* shown in black which are determined by the segment boundaries.

6.4 Relation to Prior Work

Previously, the training and inference algorithms of structured SVMs with optimal segmentation for speech recognition were introduced. The following sections will discuss the relationship between the proposed structured SVMs and several commonly used models in continuous speech recognition.

6.4.1 Relationship with Multi-class SVMs

In Sections 3.3.2 and 3.4 the multi-class SVMs for isolated word classification and for continuous speech recognition based on acoustic code-breaking were introduced. To see the relation to structured SVMs, the objective function of multi-class SVM in equation (3.36) need to be re-expressed, by stacking the class weight vectors to form a single weight vector $\boldsymbol{\alpha}$ and introducing the notation for a *joint* feature vector $\boldsymbol{\phi}(\mathbf{O}, w)$:

$$\boldsymbol{\alpha} = \begin{bmatrix} \boldsymbol{\alpha}_1 \\ \vdots \\ \boldsymbol{\alpha}_w \\ \vdots \\ \boldsymbol{\alpha}_C \end{bmatrix}, \quad \boldsymbol{\phi}(\mathbf{O}, w) = \begin{bmatrix} \mathbf{0} \\ \vdots \\ \boldsymbol{\psi}(\mathbf{O}) \\ \vdots \\ \mathbf{0} \end{bmatrix}. \tag{6.25}$$

where $\boldsymbol{\psi}(\mathbf{O})$ appears in the w-th block in $\boldsymbol{\phi}(\mathbf{O}, w)$. Thus the optimization problem (3.36) for multi-class SVM can be rewritten as

$$\min_{\boldsymbol{\alpha}, \xi_i} \quad \frac{1}{2}||\boldsymbol{\alpha}||^2 + C \sum_{i=1}^{R} \xi_i \tag{6.26}$$

s.t. For every training data $(\mathbf{O}_i, w_i)$, $i = 1, \ldots, R$,

For every competing classes (words) $w \neq w_i$:

$$\boldsymbol{\alpha}^{\mathsf{T}} \boldsymbol{\phi}(\mathbf{O}_i, w_i) - \boldsymbol{\alpha}^{\mathsf{T}} \boldsymbol{\phi}(\mathbf{O}_i, w) \geq 1 - \xi_i \qquad \text{where } \xi_i \geq 0$$

Comparing equations (6.26) and (6.4) shows that the multi-class SVM can be viewed as a simple instance of structured SVMs. If the data is unstructured (e.g., isolated words), the structured SVM in equation (6.4) will be the same as multi-class SVMs in equation (6.26). If the data is structured (e.g., continuous speech utterances), the multi-class SVM cannot be applied directly as discussed in Section 3.4. The continuous speech needs to be firstly segmented into words/sub-words observation sequences. The training/test data from the same utterance but belonging to different segments are treated independently. However, in structured SVMs, multiple segmentations and dependencies between different segments in the utterance are considered. Thus the structures in the whole utterance can be captured. The experimental results of these two models will be discussed in Section 8.1.2.1.

6.4.2 *Relationship with Log Linear Models*

Just as SVMs can be interpreted as maximum margin logistic regressions (see Section 3.2.3), the proposed structured SVM can be viewed as maximum margin log linear models with optimal segmentation.[9] To see this, the posterior of log linear model for hypothesized labels $\mathbf{w}$ given $\mathbf{O}$ with optimal segmentation $\boldsymbol{\theta}_{\boldsymbol{\alpha}}$ can be written as,[10]

$$P(\mathbf{w}|\mathbf{O};\boldsymbol{\theta}_{\boldsymbol{\alpha}},\boldsymbol{\alpha}) = \frac{\exp\left(\boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O},\mathbf{w};\boldsymbol{\theta}_{\boldsymbol{\alpha}})\right)}{\sum\limits_{\mathbf{w}'}\exp\left(\boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O},\mathbf{w}';\boldsymbol{\theta}'_{\boldsymbol{\alpha}})\right)}, \tag{6.27}$$

where $\boldsymbol{\theta}_{\boldsymbol{\alpha}}$ is the best segmentation that maximises posterior probability $P(\mathbf{w}|\mathbf{O};\boldsymbol{\theta},\boldsymbol{\alpha})$,

$$\boldsymbol{\theta}_{\boldsymbol{\alpha}} = \arg\max_{\boldsymbol{\theta}} P(\mathbf{w}|\mathbf{O};\boldsymbol{\theta},\boldsymbol{\alpha}) = \arg\max_{\boldsymbol{\theta}} \boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O},\mathbf{w};\boldsymbol{\theta})$$

Decoding with this log linear models can be simply expressed as

$$\mathbf{w}_{\boldsymbol{\alpha}} = \arg\max_{\mathbf{w}} P(\mathbf{w}|\mathbf{O};\boldsymbol{\theta}_{\boldsymbol{\alpha}},\boldsymbol{\alpha}) = \arg\max_{\mathbf{w}}\left\{\max_{\boldsymbol{\theta}}\ \boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O},\mathbf{w};\boldsymbol{\theta})\right\}$$

This yields both the optimal word sequence and alignment and is equivalent to structured SVM inference in equation (6.14).

[9] Comparing equations (4.20) and (6.5) suggests that structured SVMs can be viewed as maximum margin trained log linear models with a fixed segmentation. This section discusses the relationship in the case of optimal segmentation.

[10] To apply the log linear model for continuous speech recognition, a latent variable $\boldsymbol{\theta}$ need to be introduced to equation (4.1). This form of log linear model has been discussed in (Zhang *et al.* 2010).

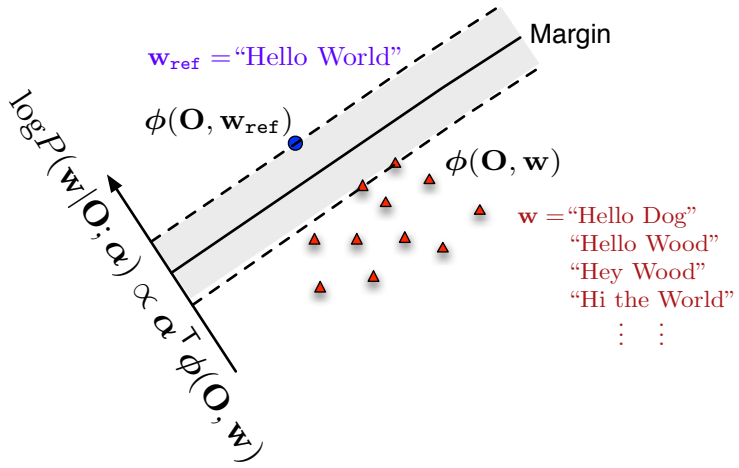


Figure 6.10 *The margin of log linear models is defined in log posterior domain between* $\mathbf{w}_{\mathtt{ref}}$ *and the best competing hypothesis* $\mathbf{w}$*. For simplicity the best segmentation* $\boldsymbol{\theta}_{\boldsymbol{\alpha}}$ *is not shown in this diagram.*

As discussed in Section 4.4.3, if the margin for the log linear models is defined as the log-posterior ratio of the reference $\mathbf{w}_{\mathtt{ref}}^{(r)}$ and best competing hypothesis $\mathbf{w}$, as illustrated in Figure 6.10, the maximum margin training for log linear model with optimal segmentation can be expressed as *minimising*

$$\mathcal{F}_{\mathtt{mm-llm}}(\boldsymbol{\alpha}, \boldsymbol{\lambda}) = \frac{1}{R} \cdot \sum_{r=1}^{R} \left[\max_{\mathbf{w} \neq \mathbf{w}_{\mathtt{ref}}^{(r)}} \left\{ \mathcal{L}(\mathbf{w}_{\mathtt{ref}}^{(r)}, \mathbf{w}) - \log \left(\frac{P(\mathbf{w}_{\mathtt{ref}}^{(r)}|\mathbf{O}^{(r)}; \boldsymbol{\theta}_{\boldsymbol{\alpha}}, \boldsymbol{\alpha})}{P(\mathbf{w}|\mathbf{O}^{(r)}; \boldsymbol{\theta}_{\boldsymbol{\alpha}}, \boldsymbol{\alpha})} \right) \right\} \right]_+ \tag{6.28}$$

Note that there are two sets of parameters, discriminative parameters $\boldsymbol{\alpha}$ and generative model parameters $\boldsymbol{\lambda}$ to extract features (see Section 5.1.2.1). One general extension of this criterion is to incorporate priors $P(\boldsymbol{\alpha}), P(\boldsymbol{\lambda})$ and then *minimise*

$$\mathcal{F}(\boldsymbol{\alpha}, \boldsymbol{\lambda}) = \mathcal{F}_{\mathtt{mm-llm}}(\boldsymbol{\alpha}, \boldsymbol{\lambda}) - \log\left(P(\boldsymbol{\alpha})\right) - \log\left(P(\boldsymbol{\lambda})\right). \tag{6.29}$$

In this work the generative model parameters, $\boldsymbol{\lambda}$, are assumed to have been trained and fixed. Equation (6.28) can then be expressed as

$$\begin{aligned} \mathcal{F}(\boldsymbol{\alpha}) = -\log(P(\boldsymbol{\alpha})) + \frac{1}{R} \sum_{r=1}^{R} \Bigg[& -\log P(\mathbf{w}_{\mathtt{ref}}^{(r)}|\mathbf{O}^{(r)}; \boldsymbol{\theta}_{\boldsymbol{\alpha}}, \boldsymbol{\alpha}) \\ & + \max_{\mathbf{w} \neq \mathbf{w}_{\mathtt{ref}}^{(r)}} \left\{ \mathcal{L}(\mathbf{w}, \mathbf{w}_{\mathtt{ref}}^{(r)}) + \log P(\mathbf{w}|\mathbf{O}^{(r)}; \boldsymbol{\theta}_{\boldsymbol{\alpha}}, \boldsymbol{\alpha}) \right\} \Bigg]_+ \end{aligned} \tag{6.30}$$

The prior $P(\boldsymbol{\alpha})$ is assumed to be Gaussian with a zero mean and scaled identity covariance matrix $C\mathbf{I}$, thus

$$\log P(\boldsymbol{\alpha}) = \log \mathcal{N}(\mathbf{0}, C\mathbf{I}) \propto -\frac{1}{2C}\boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\alpha}. \tag{6.31}$$

Note that other forms of prior distribution $P(\boldsymbol{\alpha})$, e.g., the Laplace distribution, can also be applied. This will lead to a L_1 norm (Kabán 2007), instead of L_2 norm in equation (6.31). Substituting equation (6.27) into (6.30) with this prior assumption and *canceling out* the normalization terms in (6.27), yields the objective function (6.9). Thus the structured SVM used in this work can also be viewed as a maximum margin trained log linear model with an optimal segmentation and a zero-mean Gaussian prior.

6.4.3 Relationship with HCRFs and SCRFs

Previously in Section 4.4.3, the maximum margin training for HCRFs and SCRFs has been discussed. As shown in equation (4.21) the objective function of HCRFs/SCRFs can be expressed as *minimising*

$$\mathcal{F}(\boldsymbol{\alpha}) = -\log(P(\boldsymbol{\alpha})) + \sum_{r=1}^{R}\Bigg[\overbrace{-\log\sum_{\boldsymbol{\theta}^{(r)}}\exp\left(\boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}_{\mathtt{ref}}^{(r)}; \boldsymbol{\theta}^{(r)})\right)}^{\text{concave}} + \underbrace{\max_{\mathbf{w}\neq\mathbf{w}_{\mathtt{ref}}^{(r)}}\left\{\mathcal{L}(\mathbf{w}_{\mathtt{ref}}^{(r)}, \mathbf{w}) + \log\sum_{\boldsymbol{\theta}}\exp\left(\boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}; \boldsymbol{\theta})\right)\right\}}_{\text{convex}}\Bigg]_{+} \tag{6.32}$$

where $P(\boldsymbol{\alpha})$ is the prior of discriminative parameters as discussed in previous section. It can be proved that this objective function consists of a concave and a convex functions. Thus it can be solved using the concave-convex procedure (CCCP). This is described in the following algorithm 4. To handle the issue of summing over all possible segmentations, the lattice-based framework (Layton 2006; Ragni 2013) can be applied. Note that solving the equation (6.34) requires searching for the best competing hypothesis. This can be achieved by using the Viterbi-style algorithm described in Section 6.3. For example, instead of searching one best path in Figure 6.7, the best

competing hypothesis can be found by using the same Viterbi search process but merging paths that have same word labels. This form of model is not investigated in this work.

Algorithm 4: The CCCP algorithm for HCRFs/SCRFs.

0. Initial: $\boldsymbol{\alpha}^{[0]} = [1, 0, 0 \ldots], \tau = 0$;

1. **Given** $\boldsymbol{\alpha}^{[\tau]}$, find the linear upper bound, $\ell^{[\tau]}(\boldsymbol{\alpha})$, for the concave part:

$$\ell^{[\tau]}(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^{\mathsf{T}} \left[\frac{\partial}{\partial \boldsymbol{\alpha}} \left(-\log \sum_{\boldsymbol{\theta}^{(r)}} \exp\left(\boldsymbol{\alpha}^{\mathsf{T}} \boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}^{(r)}_{\mathtt{ref}}; \boldsymbol{\theta}^{(r)}) \right) \right) \Bigg|_{\boldsymbol{\alpha}=\boldsymbol{\alpha}^{[\tau]}} \right] \tag{6.33}$$

2. **Given** $\ell^{[\tau]}(\boldsymbol{\alpha})$, find the optimal $\boldsymbol{\alpha}^{[\tau+1]}$ that minimising the following convex function using the cutting plane algorithm:

$$\boldsymbol{\alpha}^{[\tau+1]} \Leftarrow \arg\min_{\boldsymbol{\alpha}} \frac{1}{2}||\boldsymbol{\alpha}||_2^2 + \frac{C}{R} \sum_{r=1}^{R} \Big[\overbrace{\ell^{[\tau]}(\boldsymbol{\alpha}) +}^{\text{linear}} \tag{6.34}$$

$$\underbrace{\max_{\mathbf{w} \neq \mathbf{w}^{(r)}_{\mathtt{ref}}} \left\{ \mathcal{L}(\mathbf{w}^{(r)}_{\mathtt{ref}}, \mathbf{w}) + \log \sum_{\boldsymbol{\theta}} \exp\left(\boldsymbol{\alpha}^{\mathsf{T}} \boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}; \boldsymbol{\theta}) \right) \right\}}_{\text{convex}} \Big]_+$$

3. $\tau = \tau + 1$, go back to Step 1 until converge;
return $\boldsymbol{\alpha}^{[\tau]}$;

Alternatively, the maximum margin training of HCRFs/SCRFs can be approximated by using one Viterbi segmentation instead of summing over all segmentations,

$$\max_{\boldsymbol{\theta}} \boldsymbol{\alpha}^{\mathsf{T}} \boldsymbol{\phi}(\mathbf{O}, \mathbf{w}; \boldsymbol{\theta}) \approx \log \sum_{\boldsymbol{\theta}} \exp\left(\boldsymbol{\alpha}^{\mathsf{T}} \boldsymbol{\phi}(\mathbf{O}, \mathbf{w}; \boldsymbol{\theta}) \right) \tag{6.35}$$

Substituting equation (6.35) into equation (6.32) yields the objective function (6.9). Thus the proposed structured SVM with frame-level features (or segment-level features) can be related to a maximum margin trained HCRF (or SCRF) with a Viterbi segmentation and a zero-mean Gaussian prior.[11]

[11] As discussed in Sections 4.2 and 4.3 the only different between HCRFs and SCRFs is their feature functions. In HCRFs the frame-level features in (4.7) are used whereas in SCRFs segmental features can be incorporated.

6.5 Practical Issues

An efficient and robust implementation of the training and inference algorithm is important for continuous speech recognition systems. In this section several design options are described that have a substantial influence on computational efficiency. In order to address robustness issues when estimating parameters of many context-dependent phone classes from the limited amount of training data, the use of parameter tying is proposed in Section 6.5.1. A prior is introduced to reduce the number of training iterations in Section 6.5.2. To reduce the memory cost, 1-slack optimisation is used as an alternative to n-slack optimisation in Section 6.5.3. To reduce the training and decoding time, a lattice-based efficient search and paralleization strategy are proposed in Section 6.5.4.

6.5.1 *Parameter Tying*

For small vocabulary tasks, where whole-word generative models are used, the discriminative model parameters may be associated with the individual words (Zhang *et al.* 2010). When medium to large vocabulary CSR are considered there is an issue with directly using this feature space with context dependent phones. The set of all possible models $\{v_k\}_{k=1}^{M}$ yields a very large joint feature space. Although in theory this could be used, the number of discriminative model parameters becomes large. Two approaches were originally proposed in (Ragni and Gales 2011*b*) to address this problem, and are adopted in this work. The first approach is to reduce the dimension of the feature space $\psi(\cdot)$ by selecting a small set of "suitable" models. For example, instead of using the full feature space, the *matched-context* feature space of a segment $\mathbf{O}$ with the label $a{-}b{+}c$ can be used here as

$$\psi_{\boldsymbol{\lambda}}(\mathbf{O}) = \begin{bmatrix} \log p_{\boldsymbol{\lambda}}(\mathbf{O}|a{-}a{+}c) \\ \vdots \\ \log p_{\boldsymbol{\lambda}}(\mathbf{O}|a{-}y{+}c) \\ \log p_{\boldsymbol{\lambda}}(\mathbf{O}|a{-}z{+}c) \end{bmatrix}_{M_1} . \tag{6.36}$$

This reduces the dimensionality of the feature space $\psi_{\boldsymbol{\lambda}}(\cdot)$ from the number of context-dependent phones M to the number of monophones M_1. The second approach is to reduce the dimension by tying the discriminative model parameter $\boldsymbol{\alpha}$ using a phonetic decision tree (Ragni and Gales 2011*b*). For example, if v_i and v_j belong to the same leaf node in a decision tree, then $\boldsymbol{\alpha}^{(v_i)}$ and $\boldsymbol{\alpha}^{(v_j)}$ are tied. This means their corresponding features in the joint feature space can be merged. Thus the dimensionality of joint feature space reduced to $M_2 \times M_1 + 1$, where M_2 is the number of leaf nodes in decision tree. This is illustrated in Figure 6.11.

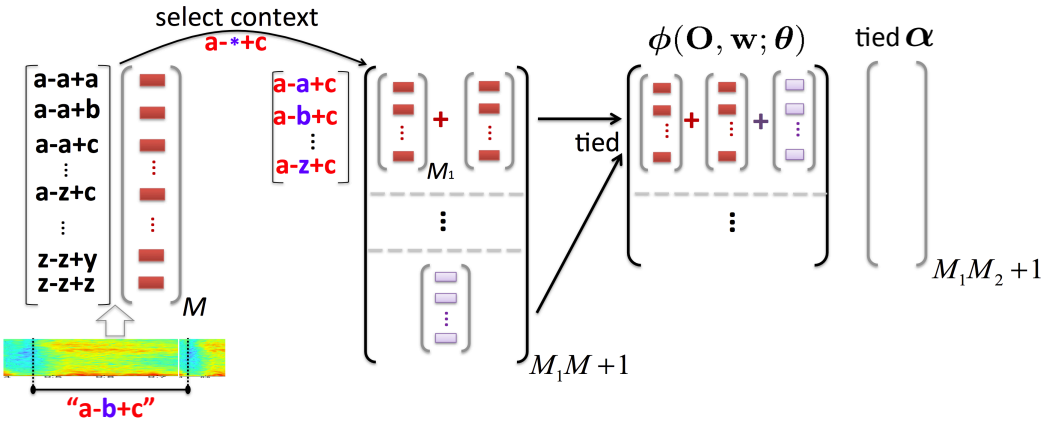


Figure 6.11 *Selecting matched context and discriminative parameter tying. The triphone label "a-b+c" denotes the context-dependent version of the phone "b" which is to be used when the left neighbour is the phone "a" and the right neighbour is the phone "c". The matched-context for "a-b+c" is "a- ∗ +c". The joint feature space is then constructed from the matched-context local features as illustrated before in Figure 5.2.*

6.5.2 *Form of Prior*

Section 6.4.2 has shown that when training standard structured SVMs, an implicit assumption is made that the prior distribution of $\boldsymbol{\alpha}$ is Gaussian, with zero mean and identity covariance matrix. However, for some feature spaces such as the log-likelihood feature space defined in equation (5.10), the prior mean, $\boldsymbol{\mu}$, should be non-zero. One appropriate form of prior mean is the one that yields the HMM baseline

performance[12] where

$$\arg\max_{\mathbf{w},\boldsymbol{\theta}} \boldsymbol{\mu}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O},\mathbf{w};\boldsymbol{\theta}) = \arg\max_{\mathbf{w}} \log\left(p(\mathbf{O}|\mathbf{w};\boldsymbol{\lambda})^{\frac{1}{\alpha_{\mathtt{lm}}}}P(\mathbf{w})\right)$$

The value of prior mean, $\boldsymbol{\mu}$, should thus be one for the correct class, zero otherwise, for example class v_1, $\boldsymbol{\mu}^{(v_1)} = [1, 0, \ldots, 0]^{\mathsf{T}}$. This motivates the need for a more general maximum margin training scheme that incorporate a general Gaussian prior $P(\boldsymbol{\alpha}) = \mathcal{N}(\boldsymbol{\alpha};\boldsymbol{\mu},\boldsymbol{\Sigma})$ into structured SVM training. Thus, the training in equation (6.9) can be generalized to *minimise*

$$\frac{1}{2}(\boldsymbol{\alpha}-\boldsymbol{\mu})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{\alpha}-\boldsymbol{\mu}) + \frac{C}{R}\sum_{r=1}^{R}\Big[-\max_{\boldsymbol{\theta}^{(r)}}\boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O}^{(r)},\mathbf{w}_{\mathtt{ref}}^{(r)};\boldsymbol{\theta}^{(r)}) + \max_{\mathbf{w}\neq\mathbf{w}_{\mathtt{ref}},\boldsymbol{\theta}}\left\{\mathcal{L}(\mathbf{w}_{\mathtt{ref}}^{(r)},\mathbf{w}) + \boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O}^{(r)},\mathbf{w};\boldsymbol{\theta})\right\}\Big]_{+} \tag{6.37}$$

This new expression is still concave-convex as long as the matrix $\boldsymbol{\Sigma}^{-1}$ is positive definite. Note the matrix $\boldsymbol{\Sigma}^{-1}$ can always be decomposed and merged into the feature space by using transformed features,

$$\bar{\boldsymbol{\phi}}(\mathbf{O},\mathbf{w};\boldsymbol{\theta}) = \boldsymbol{\Sigma}^{\frac{1}{2}}\boldsymbol{\phi}(\mathbf{O},\mathbf{w};\boldsymbol{\theta}).$$

In this work, the log-likelihood features are assumed to be consistently scaled, so that $\boldsymbol{\Sigma} = C\boldsymbol{I}$ is a reasonable approximation. In order to utilize the training framework based on equation (6.9), it is necessary to transform the parameters $\bar{\boldsymbol{\alpha}} = (\boldsymbol{\alpha}-\boldsymbol{\mu})$. Reformulating equation (6.37) in the form of (6.9)

$$\frac{1}{2}||\bar{\boldsymbol{\alpha}}||_2^2 + \frac{C}{R}\sum_{r=1}^{R}\Bigg[-\max_{\boldsymbol{\theta}^{(r)}}\left\{(\bar{\boldsymbol{\alpha}}+\boldsymbol{\mu})^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O}^{(r)},\mathbf{w}_{\mathtt{ref}}^{(r)};\boldsymbol{\theta}^{(r)})\right\} + \max_{\mathbf{w}\neq\mathbf{w}_{\mathtt{ref}}^{(r)},\boldsymbol{\theta}}\left\{\mathcal{L}(\mathbf{w}_{\mathtt{ref}}^{(r)},\mathbf{w}) + (\bar{\boldsymbol{\alpha}}+\boldsymbol{\mu})^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O}^{(r)},\mathbf{w};\boldsymbol{\theta})\right\}\Bigg]_{+} \tag{6.38}$$

Minimising equation (6.38) can be solved using Algorithm 5 and a modified version of Algorithm 2. Similar to Algorithm 2, once the optimal reference alignment $\boldsymbol{\theta}_{\boldsymbol{\alpha}}^{(r)}$ is

[12] Raising a fractional power $\frac{1}{\alpha_{\mathtt{lm}}}$ on HMM likelihoods known as acoustic deweighting (Woodland and Povey 2002).

Algorithm 5: Structured SVM training with Gaussian prior

0. Initial: $\bar{\boldsymbol{\alpha}} = [0, 0, 0 \ldots]$, $\boldsymbol{\mu} = [1, 0, 0 \ldots]$;

1. **Fixing** $\bar{\boldsymbol{\alpha}}$, **optimise** the reference alignment $\boldsymbol{\theta}_{\boldsymbol{\alpha}}^{(r)}$, $\forall\, r$,

$$\boldsymbol{\theta}_{\boldsymbol{\alpha}}^{(r)} = \arg\max_{\boldsymbol{\theta}^{(r)}} \Big\{ \left(\bar{\boldsymbol{\alpha}} + \boldsymbol{\mu}\right)^{\mathsf{T}} \boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}_{\mathtt{ref}}^{(r)}; \boldsymbol{\theta}^{(r)}) \Big\}, \tag{6.41}$$

2. **Fixing** $\boldsymbol{\theta}_{\boldsymbol{\alpha}}^{(r)}$, **optimise** $\bar{\boldsymbol{\alpha}}$ by *minimizing*:

$$\frac{1}{2}||\bar{\boldsymbol{\alpha}}||_2^2 + \frac{C}{R}\sum_{r=1}^{R}\Big[-\bar{\boldsymbol{\alpha}}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}_{\mathtt{ref}}^{(r)}; \boldsymbol{\theta}_{\boldsymbol{\alpha}}^{(r)}) + \max_{\mathbf{w}\neq\mathbf{w}_{\mathtt{ref}}^{(r)}, \boldsymbol{\theta}} \Big\{ \bar{\mathcal{L}}(\mathbf{w}_{\mathtt{ref}}^{(r)}, \mathbf{w}) + \bar{\boldsymbol{\alpha}}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}; \boldsymbol{\theta}) \Big\} \Big]_{+} \tag{6.42}$$

where $\bar{\mathcal{L}}(\mathbf{w}_{\mathtt{ref}}^{(r)}, \mathbf{w}) = \boldsymbol{\mu}^{\mathsf{T}}\Delta\boldsymbol{\phi}^{(r)} + \mathcal{L}(\mathbf{w}_{\mathtt{ref}}^{(r)}, \mathbf{w})$.
$\bar{\boldsymbol{\alpha}}$ in problem (6.42) can be learned using Algorithm 1.

3. go back to Step 1, until converge **return** $\boldsymbol{\alpha} = \bar{\boldsymbol{\alpha}} + \boldsymbol{\mu}$;

given, then equation (6.38) can be expressed as (6.42) which is exactly the same form as (6.11) with a new score-augmented loss function,

$$\bar{\mathcal{L}}(\mathbf{w}_{\mathtt{ref}}^{(r)}, \mathbf{w}) = \underbrace{\boldsymbol{\mu}^{\mathsf{T}}\Delta\boldsymbol{\phi}^{(r)}}_{\text{score loss}} + \underbrace{\mathcal{L}(\mathbf{w}_{\mathtt{ref}}^{(r)}, \mathbf{w})}_{\text{transcription loss}} \tag{6.39}$$

$\boldsymbol{\mu}^{\mathsf{T}}\Delta\boldsymbol{\phi}^{(r)}$ can be viewed as an acoustic and language score loss, where

$$\Delta\boldsymbol{\phi}^{(r)} = \boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}; \boldsymbol{\theta}) - \boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}_{\mathtt{ref}}^{(r)}; \boldsymbol{\theta}_{\boldsymbol{\alpha}}^{(r)}).$$

Inference with structured SVMs based on $\bar{\boldsymbol{\alpha}}$ can be written as

$$\{\mathbf{w}_{\boldsymbol{\alpha}}, \boldsymbol{\theta}_{\boldsymbol{\alpha}}\} = \arg\max_{\mathbf{w},\boldsymbol{\theta}} \Big(\left(\bar{\boldsymbol{\alpha}} + \boldsymbol{\mu}\right)^{\mathsf{T}} \boldsymbol{\phi}(\mathbf{O}, \mathbf{w}; \boldsymbol{\theta}) \Big). \tag{6.40}$$

One interesting property of (6.38) is that even if $\bar{\boldsymbol{\alpha}}$ is not well trained, e.g., in the early training iteration, with a proper $\boldsymbol{\mu}$ the algorithm will still generate sensible competing hypothesis and segmentation using equation (6.40). This is particularly helpful in reducing the convergence time in medium to large vocabulary CSR (see Section 8 for more details).

6.5.3 1-slack optimisation

There are two forms of cutting plane algorithms (Joachims *et al.* 2009), n-slack and 1-slack algorithms. The algorithm described previously in Algorithm 1 is the n-slack version. One issue of the n-slack algorithm is that, in theory, the n-slack algorithm can add R constraints at every iteration (as shown in equation (6.7) of Algorithm 1:

$$\boldsymbol{\alpha}^{\mathsf{T}}\left[\boldsymbol{\phi}(\mathbf{O}^{(r)},\mathbf{w}_{\mathtt{ref}}^{(r)})-\boldsymbol{\phi}(\mathbf{O}^{(r)},\mathbf{w}_{*}^{(r)})\right]\geq\mathcal{L}(\mathbf{w}_{\mathtt{ref}}^{(r)},\mathbf{w}_{*}^{(r)})-\xi_{r},\quad r=1,\ldots,R$$

where R is the size of training set. This means that the training of structured SVMs for large vocabulary continuous speech recognition is still a challenging problem (because each constraint contains a large dimensional joint feature vector). To reduce the number of constraints in every iteration, the 1-slack algorithm (Joachims *et al.* 2009) can be applied. The first step of 1-slack algorithm is to reformulate the optimization problems (6.4) as

$$\begin{aligned}&\min_{\boldsymbol{\alpha},\xi}\frac{1}{2}||\boldsymbol{\alpha}||^{2}+\frac{C}{R}\cdot\xi \qquad (6.43)\\ &\text{s.t.}\ \forall\left(\mathbf{w}_{*}^{(1)},\ldots,\mathbf{w}_{*}^{(R)}\right)\in\mathscr{W}^{R},\\ &\qquad\boldsymbol{\alpha}^{\mathsf{T}}\sum_{r=1}^{R}\left[\boldsymbol{\phi}(\mathbf{O}^{(r)},\mathbf{w}_{\mathtt{ref}}^{(r)})-\boldsymbol{\phi}(\mathbf{O}^{(r)},\mathbf{w}_{*}^{(r)})\right]\geq\sum_{r=1}^{R}\mathcal{L}(\mathbf{w}_{\mathtt{ref}}^{(r)},\mathbf{w}_{*}^{(r)})-\xi\end{aligned}$$

where $\xi\geq 0$ is the only one slack variable shared across all constraints. Each constraint in equation (6.43) depends on a combination of $\left(\mathbf{w}_{*}^{(1)},\ldots,\mathbf{w}_{*}^{(R)}\right)\in\mathscr{W}^{R}$. The equivalence between (6.4) and (6.43) can be observed by substituting slack variables into their objective functions:

$$\begin{aligned}&\frac{1}{2}||\boldsymbol{\alpha}||^{2}+\frac{C}{R}\sum_{r=1}^{R}\max_{\mathbf{w}_{*}^{(r)}\in\mathscr{W}}\left[\mathcal{L}(\mathbf{w}_{\mathtt{ref}}^{(r)},\mathbf{w}_{*}^{(r)})-\boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O}^{(r)},\mathbf{w}_{\mathtt{ref}}^{(r)})+\boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O}^{(r)},\mathbf{w}_{*}^{(r)})\right]_{+}=\\ &\frac{1}{2}||\boldsymbol{\alpha}||^{2}+\frac{C}{R}\max_{\left(\mathbf{w}_{*}^{(1)},\ldots,\mathbf{w}_{*}^{(R)}\right)\in\mathscr{W}^{R}}\sum_{r=1}^{R}\left[\mathcal{L}(\mathbf{w}_{\mathtt{ref}}^{(r)},\mathbf{w}_{*}^{(r)})-\boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O}^{(r)},\mathbf{w}_{\mathtt{ref}}^{(r)})+\boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O}^{(r)},\mathbf{w}_{*}^{(r)})\right]_{+}\end{aligned}$$

Note that the max over the combination space $\mathscr{W}^{R}$ in the second line distributes over the independent summands $\left[\mathcal{L}(\mathbf{w}_{\mathtt{ref}}^{(r)},\mathbf{w}_{*}^{(r)})-\boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O}^{(r)},\mathbf{w}_{\mathtt{ref}}^{(r)})+\boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O}^{(r)},\mathbf{w}_{*}^{(r)})\right]$. Thus it is equivalent to the first line. Based on equation (6.43), the 1-slack algorithm for

structured SVMs is described in Algorithm 6. The training process is similar to the n-slack version in Algorithm 1. It iteratively constructs a working set $\mathscr{W}^R$ of constraints. In each iteration, it finds the best competing hypothesis for each training utterance (equation (6.45)), adds them to the working set, and computes the solution over the current set $\mathscr{W}^R$ (equation (6.44)). This 1-slack algorithm stops when no constraint can be found that is violated by more than the desired precision ε.

Algorithm 6: 1-slack Cutting plane algorithm for equation (6.4)

Input: $\{(\mathbf{O}^{(r)}, \mathbf{w}^{(r)}_{\mathtt{ref}})\}_{r=1}^{R}$, C and precision ε;
Initial empty constraint set: $\mathscr{W}^R \leftarrow \emptyset$;
repeat
/ solving the 1-slack QP using current constraint set */*

$$
\begin{aligned}
(\boldsymbol{\alpha}, \xi) \leftarrow \quad & \min_{\boldsymbol{\alpha}, \xi \geq 0} \quad \frac{1}{2}||\boldsymbol{\alpha}||_2^2 + \frac{C}{R}\xi \\
& \text{s.t.} \quad \forall \left(\mathbf{w}_*^{(1)}, \ldots, \mathbf{w}_*^{(R)}\right) \in \mathscr{W}^R : \\
& \qquad \boldsymbol{\alpha}^{\mathsf{T}} \sum_{r=1}^{R} \left[\boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}^{(r)}_{\mathtt{ref}}) - \boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}_*^{(r)})\right] \geq \sum_{r=1}^{R} \mathcal{L}(\mathbf{w}^{(r)}_{\mathtt{ref}}, \mathbf{w}_*^{(r)}) - \xi
\end{aligned}
\tag{6.44}
$$

for $r = 1..R$ **do** */*generating best competing hypothesis*/*

$$
\mathbf{w}_*^{(r)} \leftarrow \arg\max_{\mathbf{w}} \left\{\mathcal{L}(\mathbf{w}, \mathbf{w}^{(r)}_{\mathtt{ref}}) + \boldsymbol{\alpha}^{\mathsf{T}} \boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w})\right\} \tag{6.45}
$$

$\mathscr{W}^R \leftarrow \mathscr{W}^R \cup \left(\mathbf{w}_*^{(1)}, \ldots, \mathbf{w}_*^{(R)}\right)$; */* put it in constraint set */*

until */* no constraint can be found that is violated by more than ε */*

$$
\boldsymbol{\alpha}^{\mathsf{T}} \sum_{r=1}^{R} \left[\boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}^{(r)}_{\mathtt{ref}}) - \boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}_*^{(r)})\right] \geq \sum_{r=1}^{R} \mathcal{L}(\mathbf{w}^{(r)}_{\mathtt{ref}}, \mathbf{w}_*^{(r)}) - \xi - \varepsilon;
$$

return $\boldsymbol{\alpha}$

Note that the 1-slack algorithm adds at most a single constraint per iteration as shown in Algorithm 6. Conversely, the n-slack algorithm can add R constraints at every iteration. For example, in the AURORA 4 experiments in Chapter 8, 1-slack algorithms produced less than 300 active constraints, whereas n-slack algorithms produced more than $50,000$ constraints after 20 iterations (still far from convergence). This makes n-slack algorithms impractical for large vocabulary CSR, since each con-

straint includes a 2210 dimensional joint feature vector. 20 iterations of n-slack optimisation required more around 18G of memory for AURORA 4. This rapidly becomes impractical using the current computer infrastructure. Thus for AURORA 4 experiments, only the result of 1-slack algorithm (with proper prior) is shown in Section 8. This trend can also be demonstrated using the AURORA 2 small vocabulary task, in which either algorithms can be applied. For AURORA 2 the n-slack algorithm produces 642 support vectors and costs 922M memory, whereas 1-slack algorithm only produces 29 support vectors and costs 83M memory. This also means that in the 1-slack algorithm the QP problem (6.44) on current working sets that need to be solved in each iteration is much smaller and faster.

6.5.3.1 Caching

Another interesting property about 1-slack algorithm (Algorithm 6) is that the constraints depend on $\sum_{r=1}^{R}\left[\phi(\mathbf{O}^{(r)}, \mathbf{w}_{\mathtt{ref}}^{(r)}) - \phi(\mathbf{O}^{(r)}, \mathbf{w}_{*}^{(r)})\right]$ rather than the individual $\phi(\mathbf{O}^{(r)}, \mathbf{w}_{*}^{(r)})$. Thus, a competing hypotheses, $\mathbf{w}_{*}^{(r)}$, can be involved in the set of active constraints many times. To avoid the computational cost of repeatedly searching for (the same) $\mathbf{w}_{\boldsymbol{\alpha}}^{(r)}$ in the large space (or lattice), the 10 most recently used features, $\phi(\mathbf{O}^{(r)}, \mathbf{w}_{*}^{(r)})$, for each training sample are cached. Therefore the search process for the best competing hypothesis (6.45) becomes

for $r = 1, \ldots, R$ **do**

$\mathbf{w}_{*}^{(r)} \leftarrow$ search equation (6.45) in the *caches*

end for

if $\sum_{r=1}^{R}\left[\phi(\mathbf{O}^{(r)}, \mathbf{w}_{\mathtt{ref}}^{(r)}) - \phi(\mathbf{O}^{(r)}, \mathbf{w}_{*}^{(r)})\right]$ remains **then**

for $r = 1, \ldots, R$ **do**

$\mathbf{w}_{*}^{(r)} \leftarrow$ search equation (6.45) in the full space (or lattices)

end for

end if

The aim of the caching strategy is to reduce the number of calls to search in the decoding space (or lattice).

6.5.3.2 *Pruning*

For both the n-slack and 1-slack algorithms, constraints added to the working set in early iterations often become inactive later. These "inactive constraints" mean that the constraint is not a support vector (Vapnik 1995), i.e. the corresponding dual parameter $\alpha^{\texttt{dual}}$ is zero. This will be discussed in more detail in Chapter 7. Those constraints that remain inactive can be removed without affecting the final solution. This is practically useful since it leads to a smaller QP problem (equation 6.44) to be solved in later iterations. In this work constraints that have not been active for more than 50 iterations are pruned to reduce the memory cost and the size of QP problem.

6.5.3.3 *Convergence*

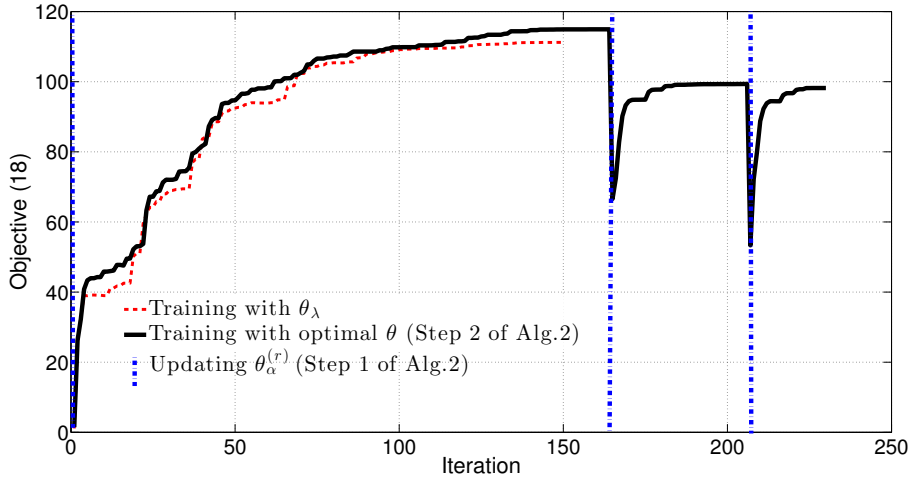


Figure 6.12 *Learning curves of structured SVMs. Dashed curve: training with HMM segmentation $\boldsymbol{\theta}_{\boldsymbol{\lambda}}$. Vertical dash-dotted lines: optimising reference segmentation $\boldsymbol{\theta}_{\boldsymbol{\alpha}}^{(r)}$* (6.10). *Solid curve: training with optimal competing segmentation $\boldsymbol{\theta}$* (6.11).

According to Theorem 2 in (Yuille *et al.* 2002), iterating step 1 and step 2 of Algorithm 2 is guaranteed to monotonically decrease the objective function (6.9) and will converge to a minimum or saddle point. The proof of the convergence is described in Appendix B. An example of the criterion value for this algorithm against iteration is shown in Figure 6.12 using the AURORA 2 data (see details about AURORA 2 in Chapter 8). Every point in Figure 6.12 is a minimum solution of the QP problem (6.44) in Algorithm 6 under the current set of constraints. The criterion increases because the cutting plane algorithm keeps adding constraints to the QP (Joachims *et al.* 2009) to get closer to the "real" minimum. However when $\boldsymbol{\theta}_{\boldsymbol{\alpha}}^{(r)}$ is updated the objective function drops because the linear part of (6.11) decreases[13]. The gap between the solid curve and dashed curve indicates the differences from optimising the segmentation, $\boldsymbol{\theta}$ in (6.11), compared to the one obtained from the generative model, $\boldsymbol{\theta}_{\boldsymbol{\lambda}}$ in (6.3).

6.5.4 *Efficient search*

Theoretically, the maximum margin training criterion discussed in Section 6.2 can be directly applied to train the model parameters. In practice, to make these algorithms applicable to larger vocabulary systems additional speed improvements are required. There are three search sub-problems that must be solved efficiently (see Fig. 6.15): the best reference segmentation in equation (6.10) of Algorithm 2; the best competing hypothesis in equation (6.6) of Algorithm 1; and the decoding with optimal segmentation in equation (6.14). These three search problems can be solved using the Viterbi-style inference algorithm described in the Section 6.3.

6.5.4.1 *Lattices constrained search*

In small vocabulary speech recognition task, it is feasible to search all possible segmentations and competing hypothesis in all three search problems (see Section 6.3). However, it is not practical for larger tasks because of the large search space of all

[13] The object drops also because the set of previous constraints discarded. Although in theory the previous constraints could be kept, for implementation simplicity this was not performed.

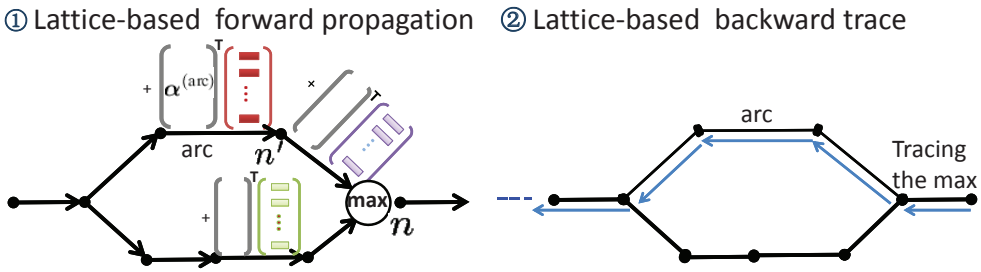


Figure 6.13 *Inference based on the lattices using arc-level forward-backward algorithm.*

possible $\mathbf{w}$ and $\boldsymbol{\theta}$. Similar to discriminative training in (Povey 2003), numerator and denominator lattices $\mathbb{L}^{\texttt{num}}$ and $\mathbb{L}^{\texttt{den}}$ are generated to restrict the search space. Thus "all possible" segmentations/hypothesis are given by arcs/paths in the lattice. Then a lattice-based search algorithm is used to find the best competing path among the lattices. Figure 6.13 shows a lattice search where n is a node in the $\mathbb{L}^{\texttt{den}}$, n' is one of its previous nodes, and $\boldsymbol{\rho}_n$ is the best path score at node n. Thus, the best competing path (hypothesis) in equation (6.6) can then be found using the following arc-level recursion

$$\boldsymbol{\rho}_n = \max_{n' \in \mathbb{L}^{\texttt{den}}} \{\boldsymbol{\rho}_{n'} + s_{n' \to n}\} \tag{6.46}$$

where $s_{n' \to n}$ is the segmental score for the arc between n' and n (see equation (6.23)) with an arc-level loss. Note that the MPE approximate loss (Povey 2003) can be computed at the arc level (see further details in the next section). This lattice based arc-level Viterbi search is a degenerate version of (6.24). Similarly, equation (6.10) can also be efficiently searched in the numerator lattice $\mathbb{L}^{\texttt{num}}$.

6.5.4.2 Loss Function

Searching for the best competing hypothesis in equation (6.13) during training requires the loss function $\mathcal{L}(\mathbf{w}^{(r)}_{\texttt{ref}}, \mathbf{w})$ to be computed. In theory any loss functions, e.g., Levantine distance, can be used. In this work, the MPE loss proposed by (Povey 2003) is applied. This enables the loss $\mathcal{L}(\mathbf{w}^{(r)}_{\texttt{ref}}, \mathbf{w})$ to be approximated at an arc-by-arc

level,

$$\mathcal{L}(\mathbf{w}, \mathbf{w}_{\mathtt{ref}}^{(r)}) \approx \sum_{\mathtt{arc}} \mathcal{L}(\mathtt{arc}, \mathbf{w}_{\mathtt{ref}}^{(r)}; \boldsymbol{\theta}^{(r)})$$

where $\mathcal{L}(\mathtt{arc}, \mathbf{w}_{\mathtt{ref}}^{(r)}; \boldsymbol{\theta}^{(r)})$ is a segmental loss based on alignment $\boldsymbol{\theta}$,

$$\mathcal{L}(\mathtt{arc}, \mathbf{w}_{\mathtt{ref}}^{(r)}; \boldsymbol{\theta}^{(r)}) = \begin{cases} \mathtt{arc} \neq \mathtt{sil} \rightarrow \begin{cases} \#\mathtt{sub} = \max(1 - c(\mathtt{arc}), 0) \\ +\#\mathtt{ins} = \max(t(\mathtt{arc}) - 1, 0) \\ +\#\mathtt{del} = \max(1 - t(\mathtt{arc}), 0) \end{cases} \\ \mathtt{arc} = \mathtt{sil} \rightarrow \#\mathtt{ins} = t(\mathtt{arc}) \end{cases} \tag{6.47}$$

where $t(\mathtt{arc})$ is the approximate total number of non-silence words (or phones) in the reference that align with $\mathtt{arc}$, and $c(\mathtt{arc})$ is the approximate number of correct words that align with word $\mathtt{arc}$. The total $t(\mathtt{arc}) \geq 0$ is found by summing, for each reference word z in the utterance that overlaps with $\mathtt{arc}$, the proportion of each word z that overlaps with $\mathtt{arc}$ as a fraction of the length of the arc z. The number of correct words $0 \leq c(\mathtt{arc}) \leq 1$ is the largest amount of overlap between a word z in the utterance that is the same word as $\mathtt{arc}$, again as a fraction of the length of z. Figure 6.14 gives an example of calculating the approximate loss for a single reference and hypothesis sentence. In this case, the exact word-level Levenshtein distance and the approximate loss are equal (both equal to 2). Thus, each of these segmental loss can be computed and incorporated into (6.46) for the lattice search.

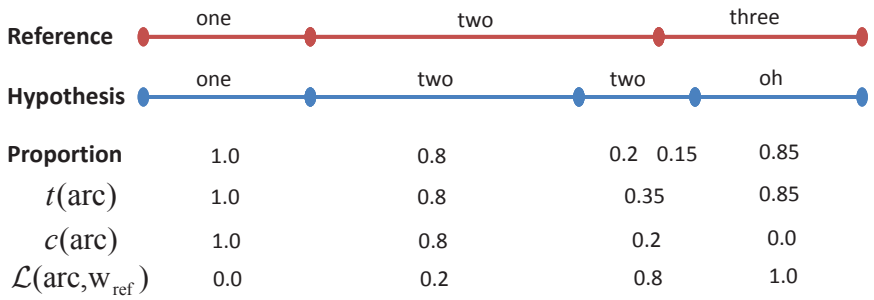


Figure 6.14 *The illustration of calculating the segmental loss.*

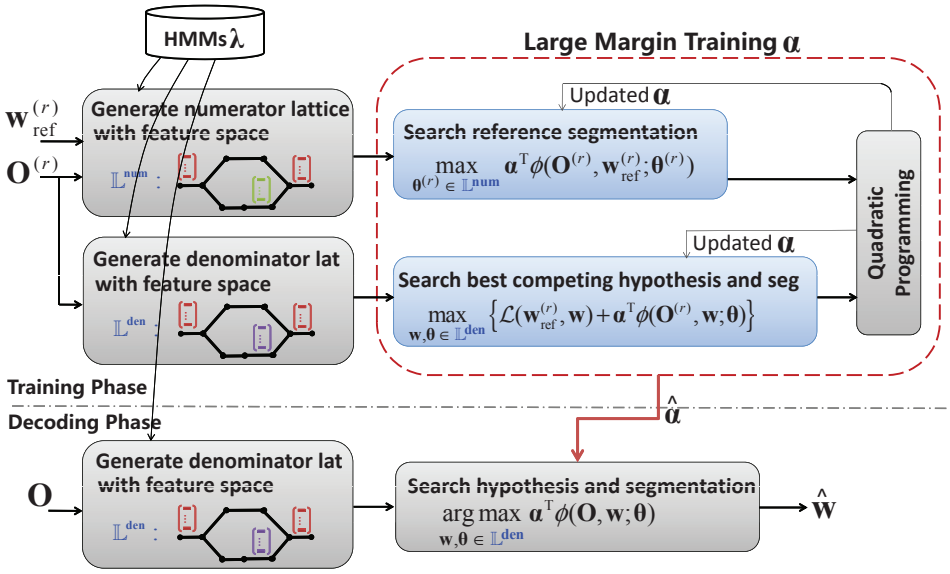


Figure 6.15 *The diagram of training and decoding for structured SVMs. The blue blocks indicate steps that can be parallelized.*

6.5.4.3 *Parallelization*

For large scale applications, the computational load during training is dominated by finding the best competing hypothesis/segmentation. For the n-slack algorithm (Algorithm 1), in order to run in parallel on many machines, the sequential update mode of the standard cutting plane algorithm needs to be modified to a batch-mode update. This can be implemented by holding the update of QP problem (equation (6.7)) until constraints from every training utterance have been produced. Note that for the n-slack algorithm, this parallelization may decrease the performance slightly (Zhang and Gales 2011*b*)[14]. However the 1-slack algorithm (Algorithm 6) used in this work can be easily parallelized without any degradation in performance. Paralleling the loop for equation (6.45) will lead to a substantial speed-up in the training process. In theory, it is also possible to parallelize the QP optimization (6.44) by several small problems using low-rank approximation technics (Zhu *et al.* 2007). However, this is not investigated in this work.

[14] Because in the sequential mode n-slack algorithm, $\boldsymbol{\alpha}$ can be updated after every training sample. This allows the algorithm to potentially find better competing $\mathbf{w}$ for the subsequence samples, but it can not be parallelized.

6.5.5 *Adaptation to Speaker and Noise Condition*

In speech recognition, the acoustic conditions during training and testing are seldom matched (due to inter-speaker variability, intra-speaker variability, background noise and channel distortions). For HMMs, as discussed in Section 2.4.5, a range of model adaptation researches have been devoted to handling this problem including: maximum a posteriori (MAP) adaptation; linear transformation-based approaches; model-based noise compensation; and feature enhancement. For details and references see (Gales 2011; Gales and Young 2007). When applying these concepts to structured SVMs there are two options. First, the discriminative model parameters, $\boldsymbol{\alpha}^{\mathsf{T}} = [\boldsymbol{\alpha}^{(v_1)^{\mathsf{T}}}, \ldots, \boldsymbol{\alpha}^{(v_{\mathtt{M}})^{\mathsf{T}}}]$, can be adapted. However with very limited data in the target domain, in these experiments a single utterance, this is very difficult.

Alternatively, the HMM parameters $\boldsymbol{\lambda}$ associated with the joint feature space can be adapted. This is discussed in the feature adaptation framework in Section 5.1.3. The HMM parameters $\boldsymbol{\lambda}$ can be adapted using any model-based compensation scheme. In this work VTS compensation described in Section 2.4.5.2 is used to handle background noise. The noise model parameters are estimated using maximum likelihood estimation (Liao and Gales 2006). Thus in the target condition the parameters of proposed structured SVMs $\boldsymbol{\alpha}$ can be assumed to be speaker and noise-independent, whereas the HMM parameters $\boldsymbol{\lambda}$ and joint feature spaces $\boldsymbol{\phi}(\mathbf{O}, \mathbf{w})$ are speaker and noise-dependent.

6.6 Summary

This chapter proposes a structured SVMs (SSVM) framework suitable for medium to large vocabulary CSR. The features described in Chapter 5 can be directly applied. These features usually depend on the segmentation of the observations (Zhang *et al.* 2010; Zweig and Nguyen 2009). This segmentation is itself a function of the model. A Viterbi-like algorithm is described to obtain the *optimal segmentation* using the current discriminative model parameters. This chapter also describes an efficient max-

imum margin training scheme based on lattices. Standard SSVMs are shown to be related to maximum margin log linear model with a zero mean Gaussian prior of the discriminative parameter. However, depending on the property of the feature space, a non-zero mean may be more appropriate. An approach to incorporate a more general Gaussian prior into SSVM training is detailed. An important feature is that the prior is used in a form that allows the cutting plane algorithm to be directly applied. Using an appropriate prior can reduce the convergence time in large scale application. Furthermore, in order to reduce the number of constraints during parameter optimisation on larger tasks, 1-slack cutting plane algorithm is used rather than the standard n-slack algorithm. To speed up the training process, caching and parallelization strategies are also proposed.

Chapter 7

Kernelized Structured SVMs for Speech Recognition

In the previous chapters various discriminative models for speech recognition have been discussed, e.g., hidden Conditional Random Fields (HCRFs) (Gunawardana *et al.* 2005), segmental Conditional Random Fields (SCRFs) (Zweig and Nguyen 2009) , Conditional Augmented models (C-Aug) (Layton 2006) and Structured Support Vector Machines (SSVMs) (Zhang and Gales 2011*b*). However, all these models require the joint feature space $\phi(\mathbf{O}, \mathbf{w})$ to be *explicitly* defined and computed. Thus, the computational cost and memory requirement are at least propositional to the number of features.

To avoid computing the possibly extremely high (or infinite) dimensional features, several methods based on the kernel trick have been developed (Shawe-Taylor and Cristianini 2004). These methods handle the inner product in high-dimensional feature $\phi(\mathbf{O})$ of observation sequence $\mathbf{O}$ using a non-linear kernel function, $K(\mathbf{O}_i, \mathbf{O}_j) =$

$\phi(\mathbf{O}_i)^{\mathsf{T}}\phi(\mathbf{O}_j)$. Thus the similarity between two observation sequences can be measured based on the kernel function, instead of computing the high-dimensional $\phi(\cdot)$ and the dot product. Although kernel methods has been partially evaluated for frame-level phoneme classification tasks (Kubo *et al.* 2010; 2011), not much work has been reported on maximum margin kernel methods for continuous speech recognition.

To kernelize these models in Chapters 4 and 6 for continuous speech recognition, this chapter proposes a *joint* kernel $K\left((\mathbf{O}_i, \mathbf{w}_i), (\mathbf{O}_j, \mathbf{w}_j)\right) = \phi(\mathbf{O}_i, \mathbf{w}_i)^{\mathsf{T}}\phi(\mathbf{O}_j, \mathbf{w}_j)$, which defines a similarity between observation-word sequence pairs, $(\mathbf{O}, \mathbf{w})$. The proposed joint kernel can be decomposed at the frame or segment level, which allows efficient maximum margin training and decoding. One elegant property of this framework is the interface between the speech data and the learning algorithm is made uniquely through the kernel function (as illustrated in Figure 7.1). This modularity allows developing general learning algorithms and designing suitable kernels for specific problems independently. The same algorithm will work with any kernel and hence for data in any domain. Another advantage of kernelization is it allows nonlinear decision boundary in the joint space. Although this chapter focuses on kernelizing the SSVMs, the algorithms described here can be directly applied to kernelize any structured discriminative models, e.g., hidden CRFs and segmental CRFs (Zhang and Gales 2013*a*).

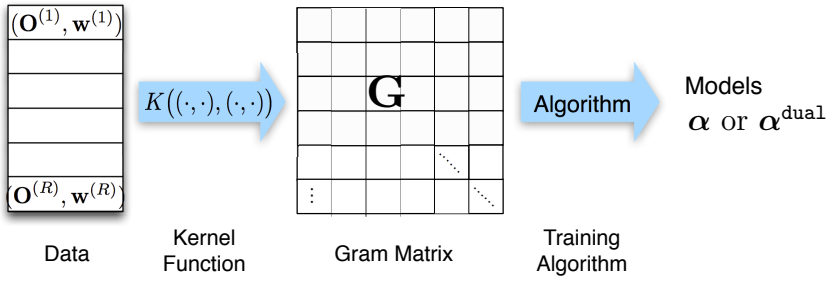


Figure 7.1 *Kernel methods offer a modular framework. In the first step, a dataset is processed into a Gram matrix. In the second step, a variety of learning algorithms can be used to analyze the data, using only the information contained in the Gram matrix. Note in binary SVMs, the dimension of Gram matrix is equal to the size of training set R. In multi-class SVMs, the dimension becomes RM, where M is the number of classes. This Chapter will show that the dimension of Gram matrix in structured SVMs becomes infinite.*

7.1 Maximum Margin Training with Kernels

7.1.1 Dual representation

Given an observation sequence, $\mathbf{O} = \{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_T\}$ and the corresponding label sequence $\mathbf{w} = \{w_1, \ldots, w_{|\mathbf{w}|}\}$, both training and decoding processes for many discriminative models, e.g., segmental CRFs and structured SVMs, can be expressed as an inner product $\boldsymbol{\alpha}^\mathsf{T}\boldsymbol{\phi}(\mathbf{O}, \mathbf{w})$ between model parameters and features.[1] In the previous discussion this is computed by assuming that there is an *explicit* representation of each of these. It is also possible to consider a more general form to compute the inner product — using kernel functions (in the same way as non-linear SVMs (Vapnik 1995)). This allows the "kernel trick" to be used to avoid explicitly computing and saving the large feature space.

Similar to SVMs (Vapnik 1995), to kernelize the structured SVMs, it is necessary to rewrite the 1-slack training Algorithm 6 described in Section 6.5.3 in the dual form (Boyd and Vandenberghe 2009; Joachims *et al.* 2009). Note that the model parameters $\boldsymbol{\alpha}$ are not trained directly in this case. Instead the dual variables $\boldsymbol{\alpha}^{\mathtt{dual}} = \left[\alpha_1^{\mathtt{dual}}, \ldots, \alpha_\tau^{\mathtt{dual}}, \ldots, \alpha_n^{\mathtt{dual}}\right]$ are learned by solving the following dual optimization of equation (6.44)

$$\begin{aligned} \max_{\alpha_\tau^{\mathtt{dual}} \geq 0} \quad & -\frac{1}{2}\boldsymbol{\alpha}^{\mathtt{dual}\mathsf{T}}\mathbf{G}\boldsymbol{\alpha}^{\mathtt{dual}} + \sum_{\tau=1}^{n}\alpha_\tau^{\mathtt{dual}}\mathcal{L}_\tau \\ \text{s.t.} \quad & \sum_{\tau=1}^{n}\alpha_\tau^{\mathtt{dual}} = C \end{aligned} \tag{7.1}$$

where n is the number of training iterations in Algorithm 6, $\mathcal{L}_\tau$ is the average loss at iteration τ

$$\mathcal{L}_\tau = \frac{1}{R}\sum_{r=1}^{R}\mathcal{L}(\mathbf{w}_{\mathtt{ref}}^{(r)}, \mathbf{w}_\tau^{(r)})$$

where $\mathbf{w}_\tau^{(r)}$ is the competing word sequence for the $r^{\mathtt{th}}$ utterance in the $\tau^{\mathtt{th}}$ iteration.

[1] For simplicity the segmentation $\boldsymbol{\theta}$ is ignored in this chapter. It can be easily involved without affecting the kernelization described below.

$\mathbf{G} = [g_{t,\tau}]_{n\times n}$ is the Gram matrix with elements

$$g_{t,\tau} = \frac{1}{R^2}\left[\sum_{i=1}^{R}\left(\boldsymbol{\phi}(\mathbf{O}^{(i)}, \mathbf{w}_{\mathtt{ref}}^{(i)}) - \boldsymbol{\phi}(\mathbf{O}^{(i)}, \mathbf{w}_{\tau}^{(i)})\right)\right]^{\mathsf{T}} \left[\sum_{j=1}^{R}\left(\boldsymbol{\phi}(\mathbf{O}^{(j)}, \mathbf{w}_{\mathtt{ref}}^{(j)}) - \boldsymbol{\phi}(\mathbf{O}^{(j)}, \mathbf{w}_{t}^{(j)})\right)\right] \tag{7.2}$$

Note that the dual optimization (7.1) only depends on the Gram matrix $\mathbf{G} = [g_{t,\tau}]_{n\times n}$[2], where $g_{t,\tau}$ depends on the inner product of the joint feature vectors $\boldsymbol{\phi}(\cdot)$, and thus can be replaced by the *joint* kernel function $K(\cdot,\cdot)$,

$$g_{t,\tau} = \frac{1}{R^2}\sum_{i=1}^{R}\sum_{j=1}^{R}\Big[K\left((\mathbf{O}^{(i)}, \mathbf{w}_{\mathtt{ref}}^{(i)}), (\mathbf{O}^{(j)}, \mathbf{w}_{\mathtt{ref}}^{(j)})\right) - K\left((\mathbf{O}^{(i)}, \mathbf{w}_{\mathtt{ref}}^{(i)}), (\mathbf{O}^{(j)}, \mathbf{w}_{\tau}^{(j)})\right) - K\left((\mathbf{O}^{(j)}, \mathbf{w}_{\mathtt{ref}}^{(j)}), (\mathbf{O}^{(i)}, \mathbf{w}_{t}^{(i)})\right) + K\left((\mathbf{O}^{(i)}, \mathbf{w}_{t}^{(i)}), (\mathbf{O}^{(j)}, \mathbf{w}_{\tau}^{(j)})\right)\Big] \tag{7.3}$$

where $K(\cdot,\cdot)$ is a *joint* kernel function

$$K\left((\mathbf{O}^{(i)}, \mathbf{w}^{(i)}), (\mathbf{O}^{(j)}, \mathbf{w}^{(j)})\right) = \boldsymbol{\phi}(\mathbf{O}^{(i)}, \mathbf{w}^{(i)})^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O}^{(j)}, \mathbf{w}^{(j)}) \tag{7.4}$$

The inner product in equation (7.4) is computed either explicitly or via the Kernel function. Note that if only the linear kernel is used, it is typically more efficient to compute $g_{t,\tau}$ in the form of equation (7.2). If nonlinear kernels are applied (e.g., kernels that imply high-dimension features), the equation (7.3) may be more efficient.

The joint kernels are easier to describe analytically, since they express the correlation between two $(\mathbf{O}, \mathbf{w})$ pairs (Weston *et al.* 2005). More details will be discussed in Section 7.2. Thus the interface between the speech data and the learning algorithm is made uniquely through this joint kernel function. The form of output parameter $\boldsymbol{\alpha}^{\mathtt{dual}}$ is described in the following section.

[2] It is more efficient to store the Gram Matrix $\mathbf{G}$ instead all the joint features during the training.

7.1.2 *Kernel algorithm*

The kernelized training algorithm can be simply described in three steps. First, solve the dual quadratic program (7.1) based on the current Gram matrix $\mathbf{G}$. At iteration n this will return an n-dimensional $\boldsymbol{\alpha}^{\texttt{dual}}$. Second, use this $\boldsymbol{\alpha}^{\texttt{dual}}$ to find the "best" competing hypothesis $\mathbf{w}_{n+1}^{(r)}$ for each utterance r in parallel. These $\left\{\mathbf{w}_{n+1}^{(1)}, \ldots, \mathbf{w}_{n+1}^{(R)}\right\}$ will be used to compute the losses and evaluate the kernel functions. Third, accumulate the kernel values in equation (7.3) to compute $[g_{1,n+1}, \ldots, g_{n+1,n+1}]^{\mathsf{T}}$ and update the Gram matrix by one more column and row as illustrated in Figure 7.2. The process is summarized in Alg. 7. The algorithm is guaranteed to converge as long as the Gram matrix $\mathbf{G}$ is positive definite.

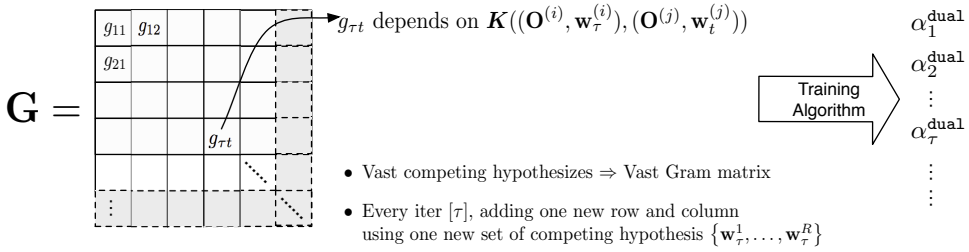


Figure 7.2 *Gram matrix and kernel algorithm. The training process is detailed in Algorithm 7 and* $\mathbf{G} = [g_{t,\tau}]_{n\times n}$ *is computed in equation (7.3). The matrix is infinity large in theory. In every iteration the matrix is grow by one dimension.*

Interestingly, in kernelized binary SVMs the size of Gram matrix $\mathbf{G}_{R\times R}$ is fixed (as there are only two classes) (Vapnik 1995); however for kernelized structured SVMs the size of $\mathbf{G}_{n\times n}$ is infinite in theory (as there are infinite number of possible classes). Note that the ideal Gram matrix is not full rank. Although there are infinite possible competing hypotheses, the number of the competing hypotheses that will affect the results is limited as discussed in Section 6.2.1. Therefore, the training process can be viewed as a selection of "important" rows and columns from the ideal infinite Gram matrix, by iteratively searching the "best" competing hypotheses. Thus, during training the size of the Gram matrix $\mathbf{G}_{n\times n}$ is dynamic and depends on the number of "best" competing hypotheses has been found. In practice, the vector $\boldsymbol{\alpha}^{\texttt{dual}}$ is usually sparse.

Algorithm 7: Kernel algorithm for structured SVMs (dual version of Alg. 6)

Input: $\{(\mathbf{O}^{(r)}, \mathbf{w}^{(r)}_{\mathtt{ref}})\}_{r=1}^{R}$ and joint kernel function K;
repeat
/ Step-1: Solve current dual program */*

$$\boldsymbol{\alpha}^{\mathtt{dual}} \leftarrow \max_{\alpha_\tau^{\mathtt{dual}} \geq 0} \; -\frac{1}{2}\boldsymbol{\alpha}^{\mathtt{dual}\,\mathsf{T}}\mathbf{G}\boldsymbol{\alpha}^{\mathtt{dual}} + \sum_{\tau=1}^{n}\alpha_\tau^{\mathtt{dual}}\mathcal{L}_\tau \qquad (7.5)$$

$$\text{s.t.} \quad \sum_{\tau=1}^{n}\alpha_\tau^{\mathtt{dual}} = C$$

/ Step-2: Find "best" competing hypothesis */*
for $r = 1..R$ **do**

$$\mathbf{w}^{(r)}_{n+1} \leftarrow \arg\max_{\mathbf{w}} \left\{\mathcal{L}(\mathbf{w}^{(r)}_{\mathtt{ref}}, \mathbf{w}) + \boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w})\right\} \qquad (7.6)$$

where $\boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w})$ is implicitly computed in (7.8).

/ Step-3: Update Gram matrix* $\mathbf{G}$ **/*
Use $\mathbf{w}^{(r)}_{n+1}$ and (7.3) to compute a new column: $[g_{1,n+1}, \ldots, g_{n+1,n+1}]^{\mathsf{T}}$,
Update $\mathbf{G}_{n\times n} \rightarrow \mathbf{G}_{(n+1)\times(n+1)}$;
$n = n + 1$;

until */* no new "best" competing hypothesis can be found */* ;
return $\boldsymbol{\alpha}^{\mathtt{dual}}$

In analogy to binary SVMs, we can refer to those $\mathbf{w}^{(1)}_\tau, \ldots, \mathbf{w}^{(R)}_\tau$ with non-zero $\alpha_\tau^{\mathtt{dual}}$ as Support Vectors. However, note that Support Vectors in the 1-slack formulation are linear combinations of multiple examples.

To reduce the memory cost, in this work $\mathbf{w}^{(1)}_\tau, \ldots, \mathbf{w}^{(R)}_\tau$ and the τ^{th} row of Gram matrix can be pruned when the corresponding $\alpha_\tau^{\mathtt{dual}}$ remains 0 for more than 100 iterations. In this case the corresponding competing hypotheses are treated as "inactive". It is also possible to use a uniformly random sampling to select a subset of training data in every iteration of step 2 in Algorithm 7. Thus the kernel computation in $g_{t,\tau}$ can be significantly reduced from $\mathcal{O}(R^2)$ to $\mathcal{O}(R'^2)$ (Yu and Joachims 2008), where R' is the number of samples. There has also been extensive work on speeding up kernel methods based on Gram matrix approximation for binary classfication. The Nystrom method has been proposed in (Williams and Seeger 2001) to approximate the kernel

matrix used for Gaussian Process classification. Low-rank approximation has been exploited to speed up the training of kernel SVMs (Fine and Scheinberg 2002). But these are not examined this work.

7.1.3 *Relationship between dual and primal parameters*

In a similar fashion to SVMs, the model parameter $\boldsymbol{\alpha}$ can be retrieved from $\boldsymbol{\alpha}^{\mathtt{dual}}$ by linearly combining the joint features of the reference and competing hypotheses,

$$\boldsymbol{\alpha} = \frac{1}{R} \sum_{\forall \tau, r} \alpha_{\tau}^{\mathtt{dual}} \left[\boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}_{\mathtt{ref}}^{(r)}) - \boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}_{\tau}^{(r)}) \right] \tag{7.7}$$

where $r = 1, \ldots, R$ and τ corresponds to all the support vectors ("active" competing hypotheses). Using the dual parameters $\boldsymbol{\alpha}^{\mathtt{dual}}$, one can compute the inner products of primal $\boldsymbol{\alpha}$ and joint features via the kernel functions,

$$\begin{aligned} \boldsymbol{\alpha}^{\mathsf{T}} \boldsymbol{\phi}(\mathbf{O}, \mathbf{w}) &= \left[\frac{1}{R} \sum_{\forall \tau, r} \alpha_{\tau}^{\mathtt{dual}} \left[\boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}_{\mathtt{ref}}^{(r)}) - \boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}_{\tau}^{(r)}) \right] \right]^{\mathsf{T}} \boldsymbol{\phi}(\mathbf{O}, \mathbf{w}) \quad (7.8) \\ &= \frac{1}{R} \sum_{\forall \tau, r} \alpha_{\tau}^{\mathtt{dual}} \left[K\left((\mathbf{O}, \mathbf{w}), (\mathbf{O}^{(r)}, \mathbf{w}_{\mathtt{ref}}^{(r)}) \right) - K\left((\mathbf{O}, \mathbf{w}), (\mathbf{O}^{(r)}, \mathbf{w}_{\tau}^{(r)}) \right) \right] \end{aligned}$$

Further details about equation (7.8) are discussed in Section 7.3.

7.2 Form of joint Kernels

To avoid working in the high-dimensional features space, in the previous section the joint kernel is introduced to replace the inner product of joint features. The purpose of the joint kernel, $K\left((\mathbf{O}^{(i)}, \mathbf{w}^{(i)}), (\mathbf{O}^{(j)}, \mathbf{w}^{(j)})\right)$, is to describe a non-linear similarity between two observation-label pairs by mapping the pairs into a joint feature space. Unlike traditional kernels which only encode the information about observations and labels independently of each other, joint kernels can also encode known dependencies between observations and labels. Joint kernels have already been studied in (Joachims *et al.* 2009; Weston *et al.* 2005). In theory any function in the form of (7.4) can be

treated as a joint kernel. However, this general utterance-level form may make the training and decoding slow. To enable efficient decoding, frame-level and segment-level features were introduced in Chapter 5. This section derives their corresponding joint kernel functions.

7.2.1 *Frame-level kernels*

If the frame-level features $\boldsymbol{\phi}(\mathbf{O}, \mathbf{w}; \boldsymbol{\theta})$ in equation (5.2) are used (implicitly), the corresponding joint kernel function between utterances $\left(\mathbf{O}^{(i)}, \mathbf{w}^{(i)}\right)$ and $\left(\mathbf{O}^{(j)}, \mathbf{w}^{(j)}\right)$ can be derived as

$$
\begin{aligned}
K\left((\mathbf{O}^{(i)}, \mathbf{w}^{(i)}), (\mathbf{O}^{(j)}, \mathbf{w}^{(j)})\right) &= \left(\sum_{t=1}^{T_i} \boldsymbol{\phi}(\mathbf{o}_t^{(i)}, \theta_t^{(i)})\right)^{\mathsf{T}} \left(\sum_{t=1}^{T_j} \boldsymbol{\phi}(\mathbf{o}_t^{(j)}, \theta_t^{(j)})\right) \\
&= \left(\sum_{t=1}^{T_i} \begin{bmatrix} \vdots \\ \delta(\theta_t^{(i)} = s^w)\boldsymbol{\psi}(\boldsymbol{o}_t^{(i)}) \\ \vdots \end{bmatrix}\right)^{\mathsf{T}} \left(\sum_{t=1}^{T_j} \begin{bmatrix} \vdots \\ \delta(\theta_t^{(j)} = s^w)\boldsymbol{\psi}(\boldsymbol{o}_t^{(j)}) \\ \vdots \end{bmatrix}\right)
\end{aligned} \tag{7.9}
$$

where θ_t is the state label of frame t and s^w indicates the state of word w. $\boldsymbol{\psi}(\cdot)$ is the features extract from frame-level observations (not depend on the labels). Two examples of $\boldsymbol{\psi}(\cdot)$ were shown in Section 5.1.1.1 and 5.1.1.2. The frame-level joint kernel can then be expressed as[3]

$$
K\left((\mathbf{O}^{(i)}, \mathbf{w}^{(i)}), (\mathbf{O}^{(j)}, \mathbf{w}^{(j)})\right) = \sum_{t=1}^{T_i} \sum_{t'=1}^{T_j} \delta(\theta_t^{(i)} = \theta_{t'}^{(j)})\, k\left(\boldsymbol{\psi}(\boldsymbol{o}_t^{(i)}), \boldsymbol{\psi}(\boldsymbol{o}_{t'}^{(j)})\right) \tag{7.10}
$$

where $k(\cdot, \cdot)$ could be any static kernels used for binary SVMs (Shawe-Taylor and Cristianini 2004) (as also discussed in Section 3.2.1), e.g., linear kernels, polynomial kernels and Radial Basis Function (RBF) kernels. Using the joint kernel in equation (7.10) has three advantages. First, explicitly computing the joint features $\boldsymbol{\phi}(\mathbf{O}, \mathbf{w}; \boldsymbol{\theta})$ in equation(5.2) can be avoided. Second, equation (7.10) implies a general way to expand

[3] For simplicity, the language features are ignored in the kernels. The language scores can be easily computed and merged into the acoustic scores using the transitional way. This will not affect any training and inference algorithms in this chapter.

the feature space $\boldsymbol{\psi}(\cdot)$ by embedding it into a static kernel $k(\cdot,\cdot)$. If a linear kernel is applied, equation (7.10) will become (7.9). Alternatively, if a RBF kernel is applied, an infinite dimensional feature space is implied. If an arc-cosine kernel is applied, the corresponding feature mapping can be viewed as an infinite neural network (Cho and Saul 2010). Third, this joint kernel can be decomposed into a set of frame-level kernels k. If the state labels $\theta_t^{(i)} \neq \theta_{t'}^{(j)}$, the term δ will be zero and there is no need to compute the kernel $k(\cdot,\cdot)$. This makes efficient kernel-based decoding become possible (more details are described in Section 7.3).

7.2.2 *Segment-level kernels*

If the segment-level features $\boldsymbol{\phi}(\mathbf{O},\mathbf{w};\boldsymbol{\theta})$ in equation (5.1.2) are used (implicitly), let $\mathbf{O} = \{\mathbf{O}_1,\ldots,\mathbf{O}_m,\ldots\}$ is the observation sequence and $\mathbf{w} = \{w_1,\ldots,w_m,\ldots\}$ is the word sequence, where $(\mathbf{O}_m, w_m)$ is the $m^{\mathtt{th}}$ segment, the corresponding joint kernel function between utterances $\left(\mathbf{O}^{(i)},\mathbf{w}^{(i)}\right)$ and $\left(\mathbf{O}^{(j)},\mathbf{w}^{(j)}\right)$ can be derived as[4]

$$
\begin{aligned}
K\left((\mathbf{O}^{(i)},\mathbf{w}^{(i)}),(\mathbf{O}^{(j)},\mathbf{w}^{(j)})\right) &= \left(\sum_{m=1}^{|\mathbf{w}^{(i)}|}\boldsymbol{\phi}(\mathbf{O}_m^{(i)},w_m^{(i)})\right)^{\mathsf{T}}\left(\sum_{m'=1}^{|\mathbf{w}^{(j)}|}\boldsymbol{\phi}(\mathbf{O}_{m'}^{(j)},w_{m'}^{(j)})\right)\\
&= \left(\sum_{m=1}^{|\mathbf{w}^{(i)}|}\begin{bmatrix}\delta(w_m=v_1)\boldsymbol{\psi}(\mathbf{O}_m^{(i)})\\ \vdots \\ \delta(w_m=v_{\mathtt{M}})\boldsymbol{\psi}(\mathbf{O}_m^{(i)})\end{bmatrix}\right)^{\mathsf{T}}\left(\sum_{m'=1}^{|\mathbf{w}^{(j)}|}\begin{bmatrix}\delta(w_{m'}=v_1)\boldsymbol{\psi}(\mathbf{O}_{m'}^{(j)})\\ \vdots \\ \delta(w_{m'}=v_{\mathtt{M}})\boldsymbol{\psi}(\mathbf{O}_{m'}^{(j)})\end{bmatrix}\right)
\end{aligned}
\tag{7.11}
$$

where $\boldsymbol{\psi}(\cdot)$ is the features extract from segmental observations. Two examples of $\boldsymbol{\psi}(\cdot)$ were shown in Section 5.1.2.1 and 5.1.2.2. Thus the segment-level joint kernel can be expressed as

$$
K\left((\mathbf{O}^{(i)},\mathbf{w}^{(i)}),(\mathbf{O}^{(j)},\mathbf{w}^{(j)})\right) = \sum_{m=1}^{|\mathbf{w}^{(i)}|}\sum_{m'=1}^{|\mathbf{w}^{(j)}|}\delta(w_m^{(i)}=w_{m'}^{(j)})\,k\left(\boldsymbol{\psi}(\mathbf{O}_m^{(i)}),\boldsymbol{\psi}(\mathbf{O}_{m'}^{(j)})\right)
\tag{7.12}
$$

[4] For simplicity, the segmentation $\boldsymbol{\theta}$ is ignored. It can be included in the kernels without affecting any properties described in this chapter.

where $k(\cdot,\cdot)$ could be any static kernels discussed in Section 3.2.1, e.g., linear kernels, polynomial kernels and RBF kernels. Similar to the frame-level case, using the joint kernel in equation (7.12) has same advantages. In this case, the joint kernel can be decomposed into a set of segment-level kernels k. If the word (or subword) labels $w_m^{(i)} \neq w_{m'}^{(j)}$, the term δ will be zero and there is no need to compute the kernel $k(\cdot,\cdot)$. This makes efficient kernel-based decoding become possible (more details are described in Section 7.3). The relationship between the kernel $k(\cdot,\cdot)$ and joint kernel $K\left((\cdot,\cdot),(\cdot,\cdot)\right)$ in (7.12) can be illustrated in Fig 7.3.

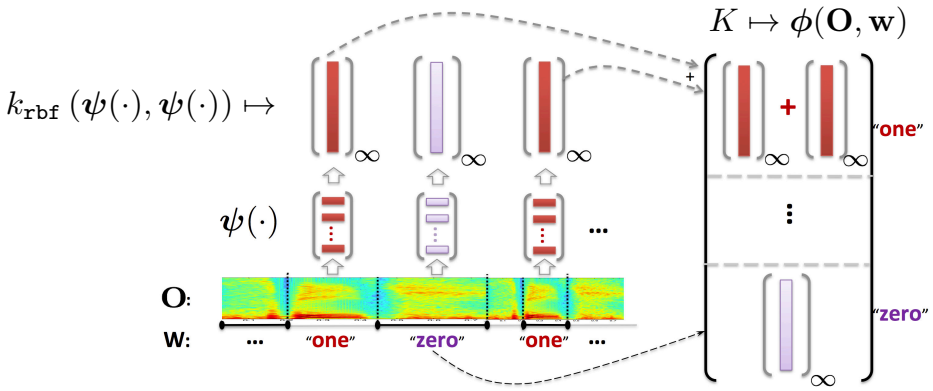


Figure 7.3 *An illustration of joint kernel and its joint feature space, where* $K\left((\cdot,\cdot),(\cdot,\cdot)\right) = \phi(\cdot,\cdot)^{\mathsf{T}}\phi(\cdot,\cdot)$. *On each segment mapping* $\psi(\cdot)$ *is used to extract the segmental features. Two examples of* $\psi(\cdot)$ *are the log likelihood features in Section 5.1.2.1 and derivative features in Section 5.1.2.2. The RBF kernel,* $k_{\texttt{rbf}}(\cdot,\cdot)$, *is applied on top of the segmental features. The resulting infinite features are concatenated (implicitly) to yields the joint feature space* $\phi\left(\mathbf{O}, \mathbf{w}\right)$.

7.3 Inference with Kernels

Similar to the inference discussion in Section 6.3, the Algorithm 7 described in previous section requires the following inference subproblem[5]:

- Inferring the most competing hypothesis using kernels for each training utter-

[5]Subsituting the equation (7.8) into the inference problems (6.13) and (6.14), equations (7.13) and (7.14) can be derived.

ance $\left(\mathbf{O}^{(r)}, \mathbf{w}^{(r)}_{\mathtt{ref}}\right)$:

$$\max_{\mathbf{w}\neq\mathbf{w}_{\mathtt{ref}}} \left\{\mathcal{L}(\mathbf{w}^{(r)}_{\mathtt{ref}}, \mathbf{w}) + \boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w})\right\} =$$
$$\max_{\mathbf{w}\neq\mathbf{w}_{\mathtt{ref}}} \left\{\mathcal{L}(\mathbf{w}^{(r)}_{\mathtt{ref}}, \mathbf{w}) + \sum_{\substack{\tau=1,\ldots n \\ r'=1,\ldots,R}} \alpha^{\mathtt{dual}}_{\tau} \left[K\left((\mathbf{O}^{(r)}, \mathbf{w}), (\mathbf{O}^{(r')}, \mathbf{w}^{(r')}_{\mathtt{ref}})\right) - K\left((\mathbf{O}^{(r)}, \mathbf{w}), (\mathbf{O}^{(r')}, \mathbf{w}^{(r')}_{\tau})\right)\right]\right\} \tag{7.13}$$

where $\mathbf{w}^{(r')}_{\tau}$ is the competing hypothesis of utterance r' found in iteration τ. The similar problem arises in the decoding process of kernelized structured SVMs or kernelized log linear models:

- Decoding a test utterance $\mathbf{O}$ based on kernels:

$$\max_{\mathbf{w}} \left\{\boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O}, \mathbf{w})\right\} =$$
$$\max_{\mathbf{w}} \sum_{\substack{\tau=1,\ldots n \\ r=1,\ldots,R}} \alpha^{\mathtt{dual}}_{\tau} \left[K\Big((\mathbf{O}, \mathbf{w}), \underbrace{(\mathbf{O}^{(r)}, \mathbf{w}^{(r)}_{\mathtt{ref}})}_{\text{reference}}\Big) - K\Big((\mathbf{O}, \mathbf{w}), \underbrace{(\mathbf{O}^{(r)}, \mathbf{w}^{(r)}_{\tau})}_{\text{competing}}\Big)\right] \tag{7.14}$$

where $\mathbf{w}^{(r)}_{\tau}|^{r=1,\ldots,R}_{\tau=1,\ldots,n}$ are the competing word sequences found in the training phase. Relating this formulation to support vectors in the SVM classification(Vapnik 1995), here $\mathbf{w}^{(r)}_{\tau}$ and $\mathbf{w}^{(r)}_{\mathtt{ref}}$ can also be viewed as support vectors. Essentially, equations (7.13) and (7.14) are the same inference problem. This section focuses on the decoding problem in equation (7.14). There are two joint kernels in this equation. The front can be viewed as a similarity between test utterance and a training reference. The other can be viewed as a similarity between test utterance and the set of competing hypothesis. Solving (7.13) can be implemented by extending the algorithms in the following two subsections to indicate the loss $\mathcal{L}(\mathbf{w}^{(r)}_{\mathtt{ref}}, \mathbf{w})$. Incorporating this loss function has already discussed in Section 6.5.4.2.

7.3.1 *Inference with frame-level kernels*

When the frame-level kernels in (7.10) are used, the inference problem in equation (7.14) can be expressed as

$$\begin{aligned}\hat{\mathbf{w}} &= \arg\max_{\mathbf{w}} \sum_{\substack{\tau=1,\dots n \\ r=1,\dots,R}} \alpha_\tau^{\texttt{dual}} \left[K\left((\mathbf{O}, \mathbf{w}), \overbrace{(\mathbf{O}^{(r)}, \mathbf{w}_{\texttt{ref}}^{(r)})}^{\text{reference}} \right) - K\left((\mathbf{O}, \mathbf{w}), \overbrace{(\mathbf{O}^{(r)}, \mathbf{w}_{\tau}^{(r)})}^{\text{competing}} \right) \right] \\ &= \arg\max_{\theta_t \in \{s^w\}} \sum_{\substack{\tau=1,\dots n \\ r=1,\dots,R}} \alpha_\tau^{\texttt{dual}} \left[\sum_{t=1}^{T} \sum_{t'=1}^{T_r} \delta(\theta_t = \theta_{t',\texttt{ref}}^{(r)})\, k\left(\boldsymbol{\psi}(\boldsymbol{o}_t), \boldsymbol{\psi}(\boldsymbol{o}_{t'}^{(r)})\right) - \right. \\ &\qquad \left. \sum_{t=1}^{T} \sum_{t'=1}^{T_r} \delta(\theta_t = \theta_{t',\tau}^{(r)})\, k\left(\boldsymbol{\psi}(\boldsymbol{o}_t), \boldsymbol{\psi}(\boldsymbol{o}_{t'}^{(r)})\right) \right] \end{aligned} \tag{7.15}$$

where $\theta_t \in \{s^w\}$ is the hidden state (or subphone) label of frame t and s^w indicates the state of word w. $\theta_{t',\texttt{ref}}^{(r)}$ is the hidden state of frame t' in the utterance $(\mathbf{O}^{(r)}, \mathbf{w}_{\texttt{ref}}^{(r)})$. $\theta_{t',\tau}^{(r)}$ is the hidden state of frame t' in the utterance $(\mathbf{O}^{(r)}, \mathbf{w}_{\tau}^{(r)})$. Note that the form of frame-level kernel in (7.10) allows the computation of $k(\cdot,\cdot)$ between two observations in equation (7.15) can be skipped, if their state label are different (term δ will be zero). This makes efficient kernel-based decoding become possible.

Let $\mathcal{T}_{\texttt{ref}}(\theta_t)$ and $\mathcal{T}_\tau(\theta_t)$ denote all the frames in $\left\{\mathbf{O}^{(r)}, \mathbf{w}_{\texttt{ref}}^{(r)}\right\}_{r=1}^{R}$ and $\left\{\mathbf{O}^{(r)}, \mathbf{w}_{\tau}^{(r)}\right\}_{r=1}^{R}$ that have the same state label with θ_t respectively. Thus equation (7.15) can be simplified as

$$\arg\max_{\theta_t \in \{s^w\}} \sum_{t=1}^{T} \left[\sum_{\tau=1}^{n} \alpha_\tau^{\texttt{dual}} \left(\sum_{t' \in \mathcal{T}_{\texttt{ref}}(\theta_t)} k\left(\boldsymbol{\psi}(\boldsymbol{o}_t), \boldsymbol{\psi}(\boldsymbol{o}_{t'})\right) - \sum_{t' \in \mathcal{T}_\tau(\theta_t)} k\left(\boldsymbol{\psi}(\boldsymbol{o}_t), \boldsymbol{\psi}(\boldsymbol{o}_{t'})\right) \right) \right] \tag{7.16}$$

The search process (7.16) can be split into two part. First, for each frame t and each state θ_t, the frame-level kernel scores need to be computed,

$$s(\theta_t) = \sum_{\tau=1}^{n} \alpha_\tau^{\texttt{dual}} \left(\sum_{t' \in \mathcal{T}_{\texttt{ref}}(\theta_t)} k\left(\boldsymbol{\psi}(\boldsymbol{o}_t), \boldsymbol{\psi}(\boldsymbol{o}_{t'})\right) - \sum_{t' \in \mathcal{T}_\tau(\theta_t)} k\left(\boldsymbol{\psi}(\boldsymbol{o}_t), \boldsymbol{\psi}(\boldsymbol{o}_{t'})\right) \right) \tag{7.17}$$

The second is obtaining the hidden state sequence, $\{\theta_1, \ldots, \theta_T\}$ and word labels, $\{w_1, \ldots, w_{|\mathbf{w}|}\}$. This can be efficiently implemented using the Viterbi algorithm (Viterbi 1982).

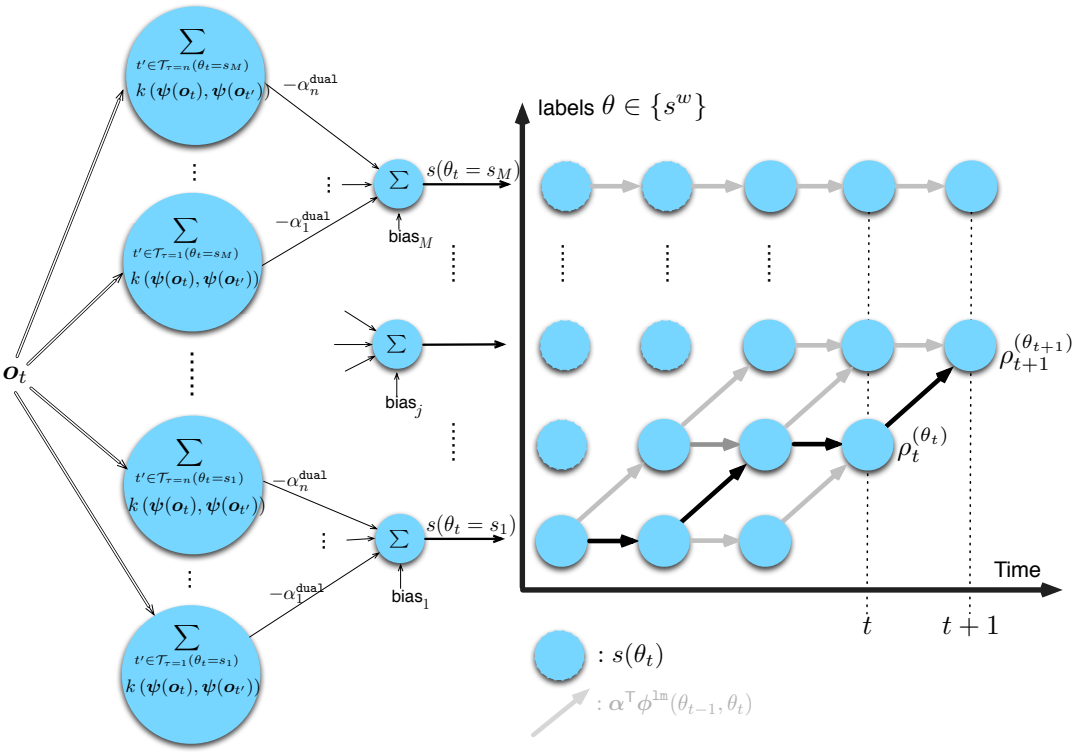


Figure 7.4 *Inference with farme-level kernels. Each big node (computing kernels) can also be viewed as a nonlinear operation $f_{\tau,\theta_t}(\boldsymbol{o}_t)$ to the input vector $\boldsymbol{o}_t$, where $\tau = 1, \ldots, n$ is the index of support vectors ("active" competing hypotheses), θ_t is the state label and $\texttt{bias}_1 = \sum_{\tau=1}^{n} \alpha_\tau^{\texttt{dual}} \sum_{t' \in \mathcal{T}_{\texttt{ref}}(\theta_t = s_1)} (\boldsymbol{\psi}(\boldsymbol{o}_t), \boldsymbol{\psi}(\boldsymbol{o}_{t'}))$. Note that this diagram can be related to the decoding process of hybrid systems with one layer MLP.*

Let $\rho_t^{(\theta_t)}$ denote the best score for a state sequence $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_t\}$ ending with θ_t at time t. Given time $t+1$ and corresponding label θ_{t+1}, the score $s(\theta_{t+1})$ can be computed in equation (7.17). The best score for a label sequence ending with θ_{t+1} can then be expressed in the recursive form,

$$\rho_{t+1}^{(\theta_{t+1})} = \max_{\theta_t \in \{s^w\}} \left\{ \rho_t^{(\theta_t)} + s(\theta_{t+1}) \right\} \tag{7.18}$$

The inference process is illustrated in the Figure 7.4. Note that the language score $\boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\phi}^{\mathtt{lm}}(\theta_{t-1}, \theta_t)$ described in Section 6.3.1 (or its kernelized form) can also be incorporated into equation (7.18). However this is not discussed in this chapter. By running the above Viterbi search from time 1 to T the optimal label sequence and segmentation can be obtained by tracing back the frame-level labels that maximising $\rho_T^{(\theta_T)}$. The complexity of the above searching process is $\mathcal{O}(MT)$ where M is the number of hidden states (or subphones) of all the words in the dictionary. Pruning options such as beam pruning (Young *et al.* 2006) can be directly applied.

7.3.2 *Inference with segment-level kernels*

When the segment-level kernels in (7.12) are used, the observation and word sequences can be split into segment level. Let $\mathbf{O}^{(r)} = \{\mathbf{O}^{(r)}_{\mathtt{ref},1}, \ldots, \mathbf{O}^{(r)}_{\mathtt{ref},m}, \ldots\}$ denote the observation sequence for utterance r and $\mathbf{w}^{(r)}_{\mathtt{ref}} = \{w^{(r)}_{\mathtt{ref},1}, \ldots, w^{(r)}_{\mathtt{ref},m}, \ldots\}$ denote the corresponding reference, where $(\mathbf{O}^{(r)}_{\mathtt{ref},m}, w^{(r)}_{\mathtt{ref},m})$ is their $m^{\mathtt{th}}$ segment. Additionally, let $\mathbf{O}^{(r)} = \{\mathbf{O}^{(r)}_{\tau,1}, \ldots, \mathbf{O}^{(r)}_{\tau,m}, \ldots\}$ denote the observation sequence for utterance r and $\mathbf{w}^{(r)}_{\tau} = \{w^{(r)}_{\tau,1}, \ldots, w^{(r)}_{\tau,m}, \ldots\}$ denote the corresponding competing hypothesis found in training iteration τ, where $(\mathbf{O}^{(r)}_{\tau,m}, w^{(r)}_{\tau,m})$ is their $m^{\mathtt{th}}$ segment. The inference problem in equation (7.14) can be expressed as

$$
\begin{aligned}
\hat{\mathbf{w}} &= \arg\max_{\mathbf{w}} \sum_{\substack{\tau=1,\ldots n \\ r=1,\ldots,R}} \alpha_\tau^{\mathtt{dual}} \left[K\left((\mathbf{O}, \mathbf{w}), \overbrace{(\mathbf{O}^{(r)}, \mathbf{w}^{(r)}_{\mathtt{ref}})}^{\text{reference}} \right) - K\left((\mathbf{O}, \mathbf{w}), \overbrace{(\mathbf{O}^{(r)}, \mathbf{w}^{(r)}_{\tau})}^{\text{competing}} \right) \right] \\
&= \arg\max_{w_1,\ldots,w_m,\ldots} \sum_{\substack{\tau=1,\ldots n \\ r=1,\ldots,R}} \alpha_\tau^{\mathtt{dual}} \left[\sum_{m=1}^{|\mathbf{w}|} \sum_{m'=1}^{|\mathbf{w}^{(r)}_{\mathtt{ref}}|} \delta(w_m = w^{(r)}_{\mathtt{ref},m'})\, k\left(\boldsymbol{\psi}(\mathbf{O}_m), \boldsymbol{\psi}(\mathbf{O}^{(r)}_{\mathtt{ref},m'}) \right) - \right. \\
&\qquad \left. \sum_{m=1}^{|\mathbf{w}|} \sum_{m'=1}^{|\mathbf{w}^{(r)}_{\tau}|} \delta(w_m = w^{(r)}_{\tau,m'})\, k\left(\boldsymbol{\psi}(\mathbf{O}_m), \boldsymbol{\psi}(\mathbf{O}^{(r)}_{\tau,m'}) \right) \right]
\end{aligned}
\tag{7.19}
$$

Let $\mathcal{A}_{\texttt{ref}}$ and $\mathcal{A}_\tau$ denote all the segments in the $\left\{\mathbf{O}^{(r)}, \mathbf{w}^{(r)}_{\texttt{ref}}\right\}_{r=1}^{R}$ and $\left\{\mathbf{O}^{(r)}, \mathbf{w}^{(r)}_{\tau}\right\}_{r=1}^{R}$, respectively. Thus the equation (7.19) can be simplified as

$$\underset{w_1,\ldots,w_m,\ldots}{\arg\max} \sum_{m=1}^{|\mathbf{w}|} \left\{ \sum_{\tau=1}^{n} \alpha_\tau^{\texttt{dual}} \left[\sum_{\substack{w_m = w^{(r)}_{\texttt{ref},m'} \\ m' \in \mathcal{A}}} k\left(\boldsymbol{\psi}(\mathbf{O}_m), \boldsymbol{\psi}(\mathbf{O}^{(r)}_{\texttt{ref},m'})\right) - \sum_{\substack{w_m = w^{(r)}_{\tau,m'} \\ m' \in \mathcal{A}_\tau}} k\left(\boldsymbol{\psi}(\mathbf{O}_m), \boldsymbol{\psi}(\mathbf{O}^{(r)}_{\tau,m'})\right) \right] \right\} \tag{7.20}$$

Equation (7.20) shows that the kernel-based decoding can be decomposed at the segmental level. Thus the word sequence $\mathbf{w}$ that maximises (7.20) can be efficiently found via a segment-level Viterbi search.

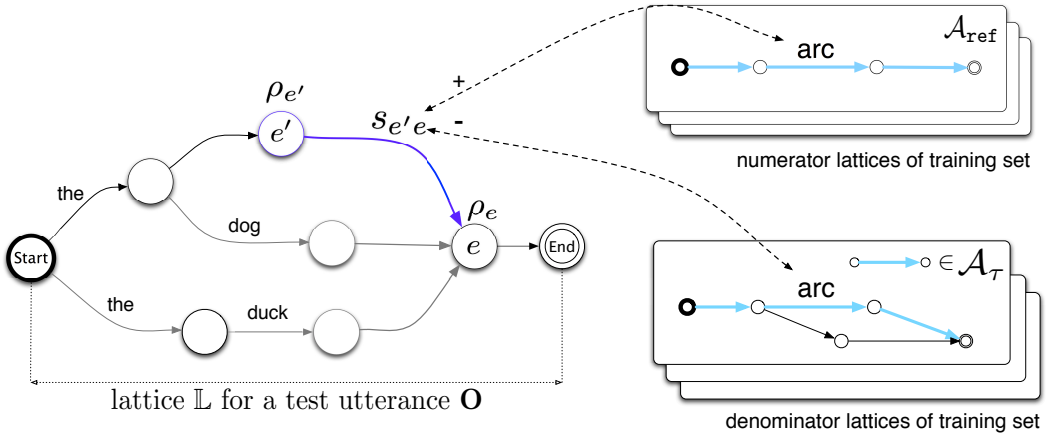


Figure 7.5 *Inference with segment-level kernels based on lattices. $\mathcal{A}_{\texttt{ref}}$ denotes all the arcs in the numerator lattices. $\mathcal{A}_\tau$ denotes those the arcs in the competing hypotheses (paths in the denominator lattices) that were generated in training iteration τ using equation (7.6).*

In practice, similar to the discriminative training in (Valtchev *et al.* 1997), lattices are usually generated during training and decoding to restrict the search space of $\mathbf{w}$. The search process (7.20) can thus be split into two distinct terms. First given a segment (an arc in the decoding lattice), e.g., the arc between nodes e' and e as shown in Figure 7.5, the segmental kernel score need to be computed,

$$s_{e'e} = \sum_{\tau=1}^{n} \alpha_\tau^{\texttt{dual}} \sum_{\substack{\texttt{arc}=e'e \\ \texttt{arc}\in\mathcal{A}}} k(\mathbf{O}_{e'e}, \mathbf{O}_{\texttt{arc}}) - \sum_{\tau=1}^{n} \alpha_\tau^{\texttt{dual}} \sum_{\substack{\texttt{arc}=e'e \\ \texttt{arc}\in\mathcal{A}_\tau}} k(\mathbf{O}_{e'e}, \mathbf{O}_{\texttt{arc}}) \tag{7.21}$$

The second is to obtain the best path with maximum score in the lattice. This requires an arc-level Viterbi search,

$$\rho_e = \max_{e' \in \mathbb{L}} \{\rho_{e'} + s_{e'e}\} \tag{7.22}$$

where e is a node in the test lattice $\mathbb{L}$, e' is one of its previous nodes, and ρ_e is the best path score at node e as shown in Figure 7.5. $s_{e'e}$ is the decomposed kernel scores for the arc $e'e$ shown in equation (7.21). The search process is illustrated in Figure 7.5 and summarized in the following algorithm.

Algorithm 8: Inference with segment-level kernels.

Input: observations $\mathbf{O}$, lattice $\mathbb{L}$
Output: word sequence $\hat{\mathbf{w}}$ (best path in $\mathbb{L}$) with confidence score

/* Initialization */
sort all the nodes in $\mathbb{L}$ in the order of time.
set score $\rho_{\texttt{start}} = 0$

/* Forward propagation */
for *each node e in $\mathbb{L}$ (in Forward direction)* **do**
  for *each previous node e' that connected to e* **do**
    compute $s_{e'e}$ using equation (7.21)
  update node score: $\rho_e \leftarrow \max_{e' \in \mathbb{L}} \{\rho_{e'} + s_{e'e}\}$;
  save previous node: $\text{Prev}(e) \leftarrow \arg\max_{e' \in \mathbb{L}} \{\rho_{e'} + s_{e'e}\}$.

/* Backward trace */
$e \leftarrow$ the end node
while $e \neq$ *start node* **do**
  retrive $e' \leftarrow \text{Prev}(e)$;
  save word: $\hat{\mathbf{w}} \leftarrow [e'e, \hat{\mathbf{w}}]$
  $e \leftarrow e'$
return $\hat{\mathbf{w}}$, $\rho_{\texttt{end}}$

7.4 Relation To Prior Work

The work presented in this chapter is a kernelized version of structured SVMs described in previous chapter. However, many other models for speech recognition can

also be kernelized. In the following we discuss the relationship of this work to some of these alternative models.

7.4.1 Relation to kernelized log linear models

According to equation (6.30) and the previous discussion in Section 6.4.2, structured SVMs can be viewed as maximum margin log linear models. Therefore the training and inference algorithms proposed in this chapter can also be applied to kernelize log linear models (Gales *et al.* 2012; Zhang and Gales 2013*a*). A kernelized log linear model was also proposed in (Kubo *et al.* 2011). Note that the kernel in (Kubo *et al.* 2011) was defined on the frame-level whereas the joint kernel in this work is defined on the sentence-level. The kernel algorithm in (Kubo *et al.* 2011) is based on MMI criteria, whereas the algorithm here is based on maximum margin training. Furthermore, the work in (Kubo *et al.* 2011) is actually a low-rank approximation of kernel methods whereas in this paper the exact Gram matrix was used. To the best of our knowledge, this work is the first attempt at a sentence-level large-margin kernel method for CSR.

7.4.2 Relation to classical kernel methods

The joint kernels and the Gram matrix are already discussed in Section 7.1.1. It may also interesting to emphasis the differences between the joint kernel and classical kernel methods.

Kernel function In classical kernel methods, the kernel function $K(\mathbf{O}^{(i)}, \mathbf{O}^{(j)}) = \boldsymbol{\phi}(\mathbf{O}^{(i)})^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O}^{(j)})$ is used to measure the similarity between observation sequences regardless of labels. This is partly because the initial research on kernel methods focused on binary classification tasks, of which the SVM is the typical example. However, continuous speech recognition requires kernels to measure a similarity between observation-label pairs, i.e., $K\left((\mathbf{O}^{(i)}, \mathbf{w}^{(i)}), (\mathbf{O}^{(j)}, \mathbf{w}^{(j)})\right) = \boldsymbol{\phi}(\mathbf{O}^{(i)}, \mathbf{w}^{(i)})^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O}^{(j)}, \mathbf{w}^{(j)})$. This *joint* kernel can encode more than just information about observations or label sequences independent of each other;

it can also encode known dependences (or correlations) between observations and label sequences. Rather than just being able to say whether two observations are similar (or class), the joint kernels offer a way of telling in which parts are they alike. They are able to specify which parts of the observations are more likely to trigger which parts of the labels. These can be seen in equation (7.12).

Gram matrix In classical kernel methods for binary SVMs, the size of Gram matrix $\mathbf{G}_{R\times R}$ is fixed. The dimension of Gram matrix is equal to the size of training set R. This is because there are only two classes in this case. Each element in the Gram matrix computes a similarity between two feature vectors. In multi-class SVMs, the dimension of Gram matrix becomes RM, where M is the number of classes (Crammer and Singer 2001). However, in joint kernel methods for structured SVMs, the dimension of $\mathbf{G}_{n\times n}$ is infinite in theory. This is because there are infinite number of possible classes (each class is a hypothesis for an utterance). Each element in Gram matrix depends on similarities between four sets of R joint feature vectors (see equation (7.3)). The Gram matrix of joint kernels is dynamic during training (it grows one dimension every iteration). The differences between two the Gram matrixes are shown in Figure 7.6.

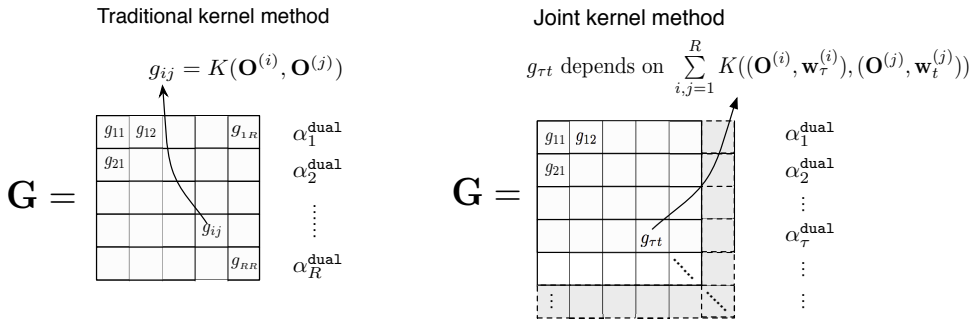


Figure 7.6 *Comparison between joint kernels and trainditional kernels.*

Support vectors In classical kernel methods, support vectors are the individual training examples $\phi(\mathbf{O}^{(i)})$, where the corresponding $\alpha_i^{\mathtt{dual}} > 0$. In joint kernel methods, support vectors are the linear combinations of multiple examples,

$\phi(\mathbf{O}^{(i)}, \mathbf{w}_\tau^{(i)})|_{i=1,\ldots,R}$, where the corresponding $\alpha_\tau^{\texttt{dual}} > 0$ and τ is the training iteration.

7.5 Summary

This chapter proposes a kernelized method in structured SVMs for continuous speech recognition. Kernelizing the structured SVMs has two advantages. First, it avoids explicitly computing and storing the high dimensional features. Second, it introduces nonlinearity to the decision boundary of the structured SVMs. This chapter has two main contributions. First, traditional kernels for speech recognition focused on measuring the similarity between two observation sequences. The proposed *joint* kernels define a sentence-level similarity between two observation-label sequence pairs. Second, this chapter addresses how to efficiently employ kernels in maximum margin training and decoding based on lattices. Future work will examine the RBF and derivative kernels in the proposed framework.

Chapter 8

Experiments

This chapter describes the evaluation of the structured SVMs described in Chapter 6 for speech recognition. The models are compared with several commonly used generative and discriminative models. To illustrate that the proposed structured SVMs can be adapted to mismatched acoustic condition, the noise-corrupted corpus AURORA 2 and 4 were used. The AURORA 2 corpus is used to contrast the performance of 1-slack and n-slack algorithms, and to demonstrate the gains from optimising the segmentation and modeling the prior. The AURORA 4 experiments are used to illustrate the performance of the proposed SSVM framework for medium vocabulary speech recognition. The 5K Wall Street Journal (WSJ0) data, the clean part of AURORA 4, were used to evaluate the performance of structured SVMs excluding the noise affects. Several commonly used generative and discriminative models, e.g., HMMs, binary SVMs, Multi-class SVMs and SCRFs are also examined for comparison.

8.1 AURORA 2 Task

This section describes the experiments on the AURORA 2 tasks with word-based models. AURORA 2 is a standard small vocabulary noise corrupted continuous digit recognition task (Pearce and Hirsch 2000). The vocabulary size, M, is only 12 (one to nine, plus zero, oh and silence). The utterances are one to seven digits long based

on the TIDIGITS database with noise artificially added. The training data is available in two conditions: clean and multi-style. The clean training data consists of 8440 digit strings up to 7 digits long spoken by 55 male and 55 female US-English speakers. The multi-style training data was obtained by artificially corrupting the clean training data using 4 types of noise N1 to N4: subway, babble, car and exhibition hall. The SNR of ranged in 5 dB increments: $0, 5, 10, 15$ and 20 dB. The test set was obtained by artificially corrupting digit strings spoken by 52 male and 52 female US-English speakers in clean conditions using 8 types of noise where SNR ranged in 5 dB increments: $0, 5, 10, 15$ and 20 dB. The test set is split into 3 sets: A, B and C. Set A contains clean data corrupted using the same 4 types of noises as the multi-style training data. Set B contains clean data corrupted using 4 different types of noise, namely restaurant, street, airport and train station. In this case there exists a mismatch between training and test data also for the multi-condition training. This will show the influence on recognition when considering different noises from the ones used for training. Set C contains half of the clean data corrupted by one type of noise from each set and a channel distortion. Subway and street are used as noise signals. The number of utterances in each set is $20002, 20002$ and 10001 respectively.

8.1.1 Experimental Setup

8440 clean mixed-gender training utterances, about 4 hours total, from 110 speakers, were used to train the acoustic generative models (HMMs). A 39 dimensional feature vector, $\boldsymbol{o}_t$, is extracted consisting of 12 MFCCs appended with the zeroth cepstrum, delta and delta-delta coefficients. The generative models, HMMs, used in this experiment were 16 emitting states whole word digit models. The HMM state output distribution is a Gaussian mixture model (GMM) with 3 components and diagonal covariance matrices. No language model was used, any length digit sequences were allowed. The HMM parameters were ML estimated described in Section 2.3.1 on the clean training data. The average WER performance of this, clean trained, HMM system was 43.31%. The first row in Table 8.1 shows the word error rate (WER) per-

Model	Set A	Set B	Set C	Avg.
HMM	43.9	46.6	35.7	43.3
HMM-VTS	9.8	9.1	9.5	9.5

Table 8.1 *WER performance of Clean-trained and VTS-compensated HMMs on AURORA 2 task. The baseline generative system is HMM-VTS.*

formance for clean trained HMMs in each test sets. In order to address the mismatch in noise conditions between the clean training data and the noise-corrupted test sets, the VTS model-based compensation (for details see Section 2.4.5.2) was applied following the procedure described in (Gales and Flego 2010). An initial estimate of the background additive noise for each utterance was obtained using the first and last 20 frames of the utterance. This was then used as the noise model for VTS compensation and each utterance recognised. This hypothesis was used to estimate a per-utterance noise model in an ML-fashion. The final recognition output used this ML-estimated noise model for VTS compensation. The second row in Table 8.1 shows the WER performance for VTS-based HMMs in each test sets. Compared with clean trained HMMs, a significant improvement is observed in each test set for HMM-VTS system. Note the word error rate for the clean, uncompensated, HMMs on test Set A was 43.9%. There is thus an 78% relative reduction in error rate by using VTS model-based compensation on Set A. For reference, the detailed results of HMM-VTS baseline on different SNR and noise types of Set A are also shown in Table 8.2. As expected, as the SNR decreases the WER increases. Due to the poor performance, the detailed results of clean trained HMMs on different noise types are no shown here.

To evaluate the benefit of the proposed structured SVM framework, a range of configurations were compared. The baseline generative system was an HMM based on VTS compensation. These compensated HMMs were also used to derive: the noise robust joint feature space; the word-level segmentation for the binary and multi-class SVMs (in the acoustic code-breaking framework described in Section 3.4); and the lattices for the structured SVM training and inference. All three test sets, A, B and C, were used for evaluating the schemes. For sets A and B, there were a total of 8 noise

SNR (dB)	Noise Type				Avg
	N1	N2	N3	N4	
20	1.78	1.87	1.55	1.54	1.69
15	2.67	2.63	2.00	2.13	2.36
10	5.13	4.20	3.37	4.91	4.39
05	12.25	12.03	8.20	12.40	11.20
00	32.18	37.70	21.92	26.47	29.55
Avg	10.80	11.69	7.41	9.49	9.84

Table 8.2 *Performance (WER %) of VTS-based HMMs on AURORA 2 test set A.*

conditions (4 in each) at 5 different SNRs, 0dB to 20dB. For test set C there were two additional noise conditions at the same range of SNRs. In addition to background additive noise convolutional distortion was added to test set C. Set A was used as the development set for tuning parameters for all systems, such as the penalty factor C in multi-class SVMs and structured SVMs.

The parameters of structured SVMs were trained using the same subset of the multi-condition training data as (Gales and Flego 2010): three of the four subsets (N2-N4) and three of five SNRs (10dB, 15dB, 20dB). This allows the generalisation of the SVMs, SCRFs and structured SVMs to unseen noise conditions to be evaluated on test set A as well as the test sets B and C, as no data from noise condition N1 and SNRs 5dB and 0dB were used. Note this makes the experiments of SVMs, SCRFs and structured SVMs hard to compare with other approaches where none of the multi-style training data was used. However the baseline VTS experiments are comparable. For SVMs and Multi-class SVMs the following segmental log-likelihood feature space (see details in Section 5.1.2.1) is used

$$\boldsymbol{\psi}_{\boldsymbol{\lambda}}(\mathbf{O}) = \begin{bmatrix} \log p_{\boldsymbol{\lambda}}(\mathbf{O}|\text{“one”}) \\ \vdots \\ \log p_{\boldsymbol{\lambda}}(\mathbf{O}|\text{“zero”}) \\ \log p_{\boldsymbol{\lambda}}(\mathbf{O}|\text{“oh”}) \\ \log p_{\boldsymbol{\lambda}}(\mathbf{O}|\text{“sil”}) \end{bmatrix}_{12} \tag{8.1}$$

where $p_{\boldsymbol{\lambda}}(\mathbf{O}|\text{“}\texttt{sil}\text{”})$ is the likelihood of segment $\mathbf{O}$ for “silence” model in HMM set. For SCRFs and structured SVMs, the following joint feature space (see details in Section 5.3) is used

$$\boldsymbol{\phi}(\mathbf{O}, \mathbf{w}; \boldsymbol{\theta}) = \begin{bmatrix} \sum_{i=1}^{|\mathbf{w}|} \delta(w_i = \text{“}\texttt{one}\text{”}) \boldsymbol{\psi}(\mathbf{O}_{i|\boldsymbol{\theta}}) \\ \vdots \\ \sum_{i=1}^{|\mathbf{w}|} \delta(w_i = \text{“}\texttt{sil}\text{”}) \boldsymbol{\psi}(\mathbf{O}_{i|\boldsymbol{\theta}}) \end{bmatrix}_{144} \tag{8.2}$$

where $\boldsymbol{\psi}(\mathbf{O}_{i|\boldsymbol{\theta}})$ is defined in equation (8.1).

8.1.2 Results and Discussion

This section compares and discusses the results of a range of models and setups. First, to illustrate the benefit of modeling the structure in the whole utterance, unstructured models–SVMs and Multi-class SVMs, and structured models–SSVMs, are compared in Section 8.1.2.1. To demonstrate the influence of different training criteria, SCRFs trained using the Conditional Maximum Likelihood (CML) and Minimum Word Error (MWE) criteria and structured SVM trained using maximum margin criterion are evaluated in Section 8.1.2.2. To evaluate refining algorithms of structured SVMs described in Section 6.2.1.3, the n-slack and 1-slack algorithms with fixed/optimal segmentation are examined in Section 8.1.2.3. Finally, the results of kernelized structured SVMs are discussed in Section 8.1.2.4. This illustrates the benefit of using nonlinear feature expansion (implicitly).

8.1.2.1 Unstructured and Structured Models

As discussed in Chapters 3 and 6, structured models can capture the dependencies in the whole utterance which may potentially lead to a better result than unstructured models. To examine this, several unstructured and structured discriminative models are evaluated and compared in Table 8.3. The binary and multi-class SVM are unstructured models where the observation sequence is first segmented into words based on HMMs and individual “segmented” words classified independently. All discriminative

models using the same 12-dimensional log-likelihood feature described in equation (8.1). The difference in performance between the structured SVM and binary/multi-class SVM systems shows the impact of fixing the segmentation rather than including structures in the model. Note that in binary/multi-class SVMs each segment is recognized independently. In structured SVMs, the information from different segments of whole utterance are used together to make a decision for decoding. As shown in Table 8.3, modeling the structures in the whole sentence yields about 6% relative improvement over the multi-class SVMs, since both systems effectively use the same feature space.

Model	# of Parameters	Set A	Set B	Set C	Avg.
HMM-VTS	$46,732$ ($\boldsymbol{\lambda}$)	9.8	9.1	9.5	9.5
SVM	+792 ($\boldsymbol{\alpha}_{w_i,w_j}$)	9.1	8.6	9.3	8.9
MSVM	+144 ($\boldsymbol{\alpha}_{w}$)	8.3	8.1	8.6	8.3
SSVM	+144 ($\boldsymbol{\alpha}$)	7.8	7.3	8.0	7.6

Table 8.3 *AURORA 2 results (WER %) of VTS based HMMs, binary SVMs (in Section 3.3.1) Multi-class SVMs (MSVM) (in Section 3.3.2), and structured SVMs using n-slack algorithms (in Section 6.2.1). The binary and multi-class SVM systems are based on the acoustic code-breaking framework discussed in Section 3.4. Note that there are $\frac{M(M-1)}{2}$ binary SVMs, where $M = 12$ is the number of words. For all discriminative models, M-dimensional log-likelihood features described in equation* (5.1.2.1) *are used.*

It is also interesting to compare the multi-class SVM in equation (3.36) and binary SVM voting schemes described in Section 3.3.1. Note that there are $\frac{M(M-1)}{2}$ binary SVMs as shown in Table 3.2, where M is the number of words. Each binary SVM is for a competing word pairs. As shown in Table 8.3, the multi-class SVM trained using equation (3.36) yields a 9% reduction in word error rate when compared to the majority voting of binary SVMs. This is mainly because in multi-class SVMs training all the class labels are considered (either as the "true" class or "competing" classes) during the optimisaiton as shown in equation (3.36). However, in binary SVMs, only two classes are considered each time during training as shown in equation (3.15).

The overall gain from using structured SVMs over the VTS-compensated HMMs

(baseline system) is over 20%. To illustrate the performance of structured SVMs excluding the affect of model compensation, the system based on the "uncompensated" HMMs was also evaluated. The performance was 37.8% on set A. Compared with the "uncompensated" HMM baseline of 43.9% in Table 8.1, a consistent improvement is achieved. The results on different noise condition can be seen in Table 8.4. As shown in this table, the major improvement is coming from low SNR data. For reference, the detailed results of structured SVMs based on VTS-HMM systems on Set A are also shown in Table 8.5. Comparing Tables 8.5 and 8.2 shows the improvement of structured SVMs in different SNR and noise types.

SNR (dB)	**Test Set A**			
	With VTS		Without VTS	
	HMM	SSVM	HMM	SSVM
20	1.69	1.25	5.30	3.36
15	2.36	1.76	16.32	10.78
10	4.39	3.33	40.45	30.75
05	11.20	8.66	69.87	61.02
00	29.55	23.90	87.36	83.08
Avg	9.84	7.78	43.86	37.88

Table 8.4 *Performance (WER %) of VTS-based or uncompensated HMMs and structured SVMs.*

SNR (dB)	**Noise Type**				**Avg**
	N1	N2	N3	N4	
20	1.38	1.36	1.16	1.08	1.25
15	2.15	1.78	1.55	1.54	1.76
10	3.90	3.17	2.51	3.76	3.33
05	9.00	10.19	6.68	8.79	8.66
00	22.87	30.44	20.19	22.09	23.90
Avg	7.86	9.39	6.42	7.45	7.78

Table 8.5 *Performance (WER %) of Structured SVM (SSVM) on AURORA 2 test set A.*

Note that as shown in Table 8.3 the number of parameters in HMM and proposed

structured SVM system are in the same range—more than $45,000$ for HMMs and only 144 additional for structured SVMs. Thus, the improvement obtained were not just the result of increasing parameters.

8.1.2.2 Structured Discriminative Models

In this section, two forms of structured discriminative models, structured SVMs as proposed in Chapter 6 and SCRFs described in Section 4.3, were evaluated. The performance of VTS-compensated HMM, SCRFs and structured SVMs with different training criteria is shown in Table 8.6. For structured SVMs and SCRFs, the same joint feature space, described in equation (8.2), were used (see details in Section 5.3). The SCRFs were trained using Conditional Maximum Likelihood (CML) and Minimum Word Error (MWE) criteria (see Section 4.4) with L_2 regularization. The VTS-compensated HMM was used to produce a pair of numerator word lattices, which encodes the reference label sequence, and denominator word lattice, which encodes a large number of possible label sequences, for each training sequence. The numerator lattices contained only the most likely Viterbi segmentation for the reference transcriptions. Thus in SCRF the summation over all possible segmentation was replaced by maximisation (Ragni 2013). The denominator lattices contained one or more alignments for each word sequence.

The discriminative model parameters associated with all feature spaces were initialised in the way that the WER performance of the VTS-compensated HMM can be achieved in the first iteration. For the log-likelihood feature in equation (8.1), the discriminative model parameter associated with "correct" likelihood were initialised to one and rest to zero (see details in Section 5.1.2.1). As shown in Table 8.6, for SCRFs the use of MWE criterion offers small but consistent improvement over the CML criterion. Examining the results in Table 8.6 also shows that the structured SVM achieved the best results among all the systems. Comparing the CML and MWE trained SCRFs, the structured SVM yields 7% and 3% relative improvement, respectively. Further improvement can be achieved by structured SVMs using some of the algorithms de-

scribed in Section 6.2.1.3. The results will be discussed in the next section. For reference, the detailed results on different SNR of Set A are also shown in Table 8.7.

Model	Param.	Criteria	Set A	Set B	Set C	Avg.
HMM-VTS	46,732	ML	9.8	9.1	9.5	9.5
SCRF	+144	CML	8.1	7.7	8.3	8.1
		MWE	8.1	7.4	8.2	7.8
SSVM	+144	MM (n-slack)	7.8	7.3	8.0	7.6

Table 8.6 *AURORA 2 results (WER %) of VTS based HMMs, SCRFs (see Section 4.3), and structured SVMs with n-slack algorithms (see Section 6.2.1). The same segmental features are used for SCRFs and structured SVMs. ML=maximum likelihood, CML=conditional maximum likelihood, MWE=minimum word error and MM=maximum margin.*

SNR	Set A				Avg. of Set A, B and C			
(dB)	HMM	MSVM	SCRF	SSVM	HMM	MSVM	SCRF	SSVM
20	1.7	1.5	1.4	1.3	1.6	1.4	1.4	1.2
15	2.4	2.0	1.9	1.8	2.4	2.0	2.0	1.8
10	4.4	3.6	3.5	3.3	4.3	3.6	3.6	3.4
05	11.2	9.2	8.9	8.7	10.7	9.1	8.8	8.5
00	29.6	25.1	24.9	23.9	28.5	25.4	24.5	23.5
Avg	9.8	8.3	8.1	7.8	9.5	8.3	8.1	7.6

Table 8.7 *AURORA 2 results (WER %) of VTS based HMMs, MSVMs, SCRFs (with CML training) and structured SVMs (with n-slack Algorithm) in different SNRs conditions.*

8.1.2.3 Training Algorithms for Structured SVMs

In the previous sections, the structured SVMs were trained with n-slack algorithm and fixed segmentation. In this section, structured SVMs trained with n-slack and 1-slack algorithms with fixed/optimal segmentation are evaluated. Examining the results in Table 8.8, the first two lines show that optimising the segmentation yields small, but consistent, gains, in performance over using the HMM-based alignment $\boldsymbol{\theta}_{\boldsymbol{\lambda}}$, about 3% relative reduction on average. Currently the performance gains from optimising the

alignments are small. However this is felt to be due to the use of whole-word models for the AURORA 2 task. Thus the alignment is only defined at the word-level. The results in the second and third lines show the benefit of using the 1-slack algorithm. The WER is almost the same[1] as the n-slack algorithm but with far fewer support vectors (29 compared with 629) and less memory cost (83M compared with 922M). This also means that in the 1-slack algorithm the QP problem (6.44) that need to be solved in each iteration is much smaller and faster. This makes 1-slack algorithms more practical for large vocabulary CSR.

Small gains are also observed when training SSVMs with a general Gaussian prior using 1-slack algorithm (last two lines in the table). The mean of Gaussian prior was set as the $\boldsymbol{\alpha}$ learned using 1-slack algorithm (the second last line in the table). Note that according to Section 6.5.2, the covariance matrix of Gaussian prior for parameter $\boldsymbol{\alpha}$ in this work is assumed to be a scaled identical matrix, $\boldsymbol{\Sigma} = C\boldsymbol{I}$. Thus the hyper-parameter C can be viewed as the variance of parameter $\boldsymbol{\alpha}$. With a proper $\boldsymbol{\mu}$, the variance C can be very small. This means the number of training iterations can be significantly reduced. In fact, this is main purpose of introducing algorithm 1-slack-$\boldsymbol{\mu}$ (Alg. 5). From another perspective, the hyper-parameter C is the penalty factor used to control the balance between generalisation and training errors. Using a larger C helps to reduce training errors but also will reduce the generalisation ability and increase the training iterations. The impact of penalty factor C in equation (6.4) for structured SVMs are shown in Figure 8.1.

Training Algorithm	$\boldsymbol{\theta}$	# SV	**Set A**	**Set B**	**Set C**	**Avg.**
n-slack (Alg. 1)	$\boldsymbol{\theta}_{\boldsymbol{\lambda}}$	629	7.8	7.3	8.0	7.6
n-slack (Alg. 2+1)	$\boldsymbol{\theta}_{\boldsymbol{\alpha}}$	642	7.6	7.1	7.8	7.4
1-slack (Alg. 2+6)	$\boldsymbol{\theta}_{\boldsymbol{\alpha}}$	29	7.6	7.3	7.9	7.5
1-slack-$\boldsymbol{\mu}$ (Alg. 5+6)	$\boldsymbol{\theta}_{\boldsymbol{\alpha}}$	30	7.5	7.1	7.9	7.4

Table 8.8 *AURORA 2 results (WER %) of SSVMs trained using n-slack algorithm without/with optimising $\boldsymbol{\theta}$ and 1-slack algorithms without/with Gaussian prior (Alg. 5+6). SV is short for support vectors.*

[1]The difference between the results of n-slack and 1-slack algorithms is coming from the rounding error during optimisation.

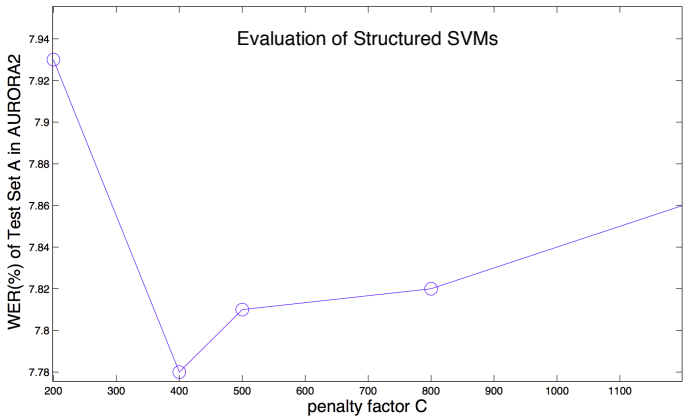


Figure 8.1 *Effect of the penalty factor C in equation* (6.4) *of structured SVMs. The models are trained using n-slack algorithm with fixed segmentation.*

8.1.2.4 Kernelized Structured SVMs

In this section, the kernelized structured SVM proposed in Chapter 7, was evaluated. The main purpose of this experiment is to illustrate that the training and decoding of structured SVMs can be achieved without computing the high-dimension joint feature space explicitly. The kernelized training process is described in details in Alg. 7. It can be summarized in three steps. First, construct the Gram matrix using the training data with references and competing hypothesis currently found. Second, solve a quadratic program based on the current Gram matrix $\mathbf{G}$. At iteration this will return an dual parameter $\boldsymbol{\alpha}^{\mathtt{dual}}$. Third, use this $\boldsymbol{\alpha}^{\mathtt{dual}}$ to find the "best" competing hypothesis for each utterance in parallel. Note that, in kernelized binary SVMs the size of Gram matrix $\mathbf{G}_{R\times R}$ is fixed (as there are only two classes); however for kernelized structured SVMs the size of $\mathbf{G}_{n\times n}$ is dynamic during training. It depends on the number of "best" competing hypotheses has been found. To reduce the memory cost, in this work the τ^{th} row of Gram matrix can be pruned when the corresponding $\alpha_{\tau}^{\mathtt{dual}}$ remains 0 for more than 100 iterations. In this case the corresponding competing hypotheses are treated as "inactive".

Model	Criterion	Kernel	Set A	Set B	Set C	Avg.
HMM-VTS	ML	–	9.8	9.1	9.5	9.5
MSVM	MM	linear	8.3	8.1	8.6	8.3
SCRF	CML	linear	8.1	7.7	8.3	8.1
SSVM (Alg. 7)	MM	linear	7.9	7.3	8.0	7.7
SSVM (Alg. 7)	MM	2^{nd}-poly	7.6	7.1	7.9	7.5

Table 8.9 *Results (WER %) of VTS based HMMs, Multi-class SVMs, SCRFs and structured SVMs with linear and 2nd order polynomial kernels (see the form of polynomial kernel in Table 3.1).*

Several forms of joint kernels have been discussed in Section 7.2 and Table 3.1. In this experiment, the following form of joint kernels is used,

$$K\left((\mathbf{O}^{(i)},\mathbf{w}^{(i)}),(\mathbf{O}^{(j)},\mathbf{w}^{(j)})\right)=\sum_{m=1}^{|\mathbf{w}^{(i)}|}\sum_{m'=1}^{|\mathbf{w}^{(j)}|}\delta(w_m^{(i)}=w_{m'}^{(j)})\ \left(\boldsymbol{\psi}(\mathbf{O}_m^{(i)})^{\mathsf{T}}\boldsymbol{\psi}(\mathbf{O}_{m'}^{(j)})+1\right)^2 \tag{8.3}$$

where the bias c of the polynomial kernel in Table 3.1 is set to 1 and order d is set to 2. The segmental feature $\boldsymbol{\psi}(\cdot)$ is defined in equation (8.1). Examining the results in Table 8.9 shows that the structured SVMs with 2nd order polynomial kernel achieved the best results among all the systems. This is mainly because the implicit polynomial expansion on feature space can make some potentially linear non-separable data separable. This can be seen from the Figure 3.5. The overall gain from using kernelized structured SVMs over the VTS-compensated HMM system is over 22%. The gain from using polynomial kernels over linear kernels is 3%[2].

Note that without kernelization, it is impractical to apply structured SVMs with a polynomial kernel, since it requires computing and keeping all the high dimensional joint features explicitly. However, in Alg. 7 only the Gram matrix is required. The results of the structured SVM with linear kernel is slightly different from the one shown in Table 8.8. This is due to two reasons. First, kernelized structured SVMs are based on dual optimization, whereas the results in previous section are based on primal form. There are rounding errors between two forms of optimisation. Second, only 1-slack

[2] This gain is statistical significant based on a matched-parir significance test at a 95% confidence level (Gillick and Cox 1989).

variable and fixed segmentations are used in this experiment. These segmentations are given by the lattices generated from HMM-VTS system. Optimising segmentations and incorporating a general prior for kernelized structured SVMs will be investigated in future work.

8.2 AURORA 4 Task

AURORA 4 is a medium vocabulary task based on the Wall Street Journal (WSJ) data. The training set is available in two conditions: clean and multi-style. The clean training data is the WSJ0 subset of WSJ SI-284 data (Paul and Baker 1992) consisting of 7138 utterances spoken by 83 speakers and totalling 14 hours of speech. The multi-style training data was obtained by artificially corrupting the clean training data using 6 types of noise and two microphone conditions where SNR ranged 10-20 dB. The test set was obtained by artificially corrupting a subset of the development set of 1992 November NIST evaluation (Paul and Baker 1992) using 6 types of noise under two microphone conditions where SNR ranged 5-15 dB. The test set is split into 4 sets: A, B, C and D. Set A contains clean data, set B contains data corrupted by 6 types of noise, set C contains data corrupted by channel distortion (desk-mounted secondary microphones recorded) and set D contains data corrupted by the noise and channel distortion. The average SNR in noise-corrupted data is 10 dB. The number of utterances in each set is 330, 1980, 330 and 1980 respectively.

8.2.1 Experimental Setup

In the previous experiments on Aurora 2 task, the acoustic model parameters were associated with individual words. For the models used in Aurora 4 task, the acoustic model parameters are associated with individual context-dependent phones. Four configurations of canonical HMMs were considered. The first repeats the setup where the HMMs were trained using clean data (SI-84 WSJ0 part, 14 hours) and then compensated with VTS compensation. The HMMs are cross-word context-dependent tri-

phone models with 3 emitting states. The state output distribution of the HMMs is a GMM with 16 components and diagonal covariance matrices. The HMM states were tied into 3143 physical states using state-level phonetic decision tree clustering. In the second, more advanced systems, VTS-adaptive training (VAT) was used to obtain the canonical HMM (Flego and Gales 2009; Kalinli *et al.* 2009). In the third MPE and VAT training was used to obtain the canonical HMM (Flego and Gales 2011). In the final experiment clean trained HMMs without VTS were applied to demonstrate the performance of structured SVMs excluding noise.

In each configuration, the canonical/compensated HMM was used to produce a word lattice. The word lattice was phone-marked to segment each word arc into a sequence phone arcs consistent with the underlying pronunciation. For each phone arc, the acoustic model score, the context-dependent phone HMM log likelihood, was replaced by the dot-product between the discriminative model parameters and the corresponding feature vector. The phone arc transitions were set to incorporate the bigram language model and pronunciation probabilities.

To compare the SCRFs and structured SVMs, the same joint feature space is applied for both models,

$$\boldsymbol{\phi}(\mathbf{O}, \mathbf{w}; \boldsymbol{\theta}) = \begin{bmatrix} \sum_{i=1}^{|\mathbf{w}|} \delta(w_i = v_1) \boldsymbol{\psi}(\mathbf{O}_{i|\boldsymbol{\theta}}) \\ \vdots \\ \sum_{i=1}^{|\mathbf{w}|} \delta(w_i = v_M) \boldsymbol{\psi}(\mathbf{O}_{i|\boldsymbol{\theta}}) \\ \log P(\mathbf{w}) \end{bmatrix}, \tag{8.4}$$

where $\{v_1, \ldots, v_M\}$ indicate all context-dependent phones in the dictionary, and the *match-context* segmental feature $\boldsymbol{\psi}(\mathbf{O}_{i|\boldsymbol{\theta}})$ is used for each segment $\mathbf{O}_{i|\boldsymbol{\theta}}$ (see details in Section 6.5.1). For example, if the label of segment $\mathbf{O}_{i|\boldsymbol{\theta}}$ is "k- ae +t", the following *match-context* segmental feature $\boldsymbol{\psi}(\mathbf{O}_{i|\boldsymbol{\theta}})$ will be applied,

$$\boldsymbol{\psi}(\mathbf{O}_{i|\boldsymbol{\theta}}) = \begin{bmatrix} \log p_{\boldsymbol{\lambda}}(\mathbf{O}_{i|\boldsymbol{\theta}} | \text{“k}-\text{aa}+\text{t”}) \\ \log p_{\boldsymbol{\lambda}}(\mathbf{O}_{i|\boldsymbol{\theta}} | \text{“k}-\text{ae}+\text{t”}) \\ \vdots \end{bmatrix}_P, \tag{8.5}$$

where P is the number of monophones ($P = 47$ in this system). A large number of context-dependent phones had limited or no examples in the multi-style training data. In order to address robustness issues when training the SCRF and structured SVM, the discriminative parameters can be tied between context-dependent phones using model-level phonetic decision tree clustering (Ragni 2013). In this experiment, monophone-level tying was applied to all context-dependent phones. This mapped $\{v_1, \ldots, v_M\}$ to 47 physical context-dependent phones. Thus, effectively only $47 \times 47 + 1$ dimensional features were used.

In the first three configurations, both the SCRFs and structured SVMs are trained on the multi-style data. The SCRF and structured SVM were trained within the feature-space adaptation and compensation framework to yield noise and speaker independent discriminative model parameters. The MPE criterion was used to yield estimates. In order to train the SCRF and structured SVM, the multi-style training data was used. The VTS-compensated HMM was used to produce a pair of numerator lattice, which encodes the reference transcription with one or more pronunciations, and denominator lattice, which encodes a large number of possible transcriptions with one or more pronunciations, for each training sequence. The numerator and denominator lattices were phone-marked. Evaluation was performed using the standard 5000- word WSJ0 bigram model on four noise-corrupted test sets based on NIST Nov'92 WSJ0 test set. Set B is used as the development set for tuning parameters of all systems.

8.2.2 *Results and Discusssion*

8.2.2.1 *VTS-based systems*

The first configuration used clean trained HMMs with VTS compensation. Table 8.10 shows the AURORA4 results of SSVMs trained with a general Gaussian prior (Algorithm 5). The mean of the prior was set as the parameters of CML trained SCRFs. The SCRFs (Ragni and Gales 2011*b*) and structured SVMs based on the same 2210 ($47 \times 47 + 1$) dimensional joint features described in equations (5.17) and (5.10). Com-

pared to the CML trained SCRFs, structured SVMs yielded a 3.4% relative reduction in WER. For this task, the n-slack algorithm cannot be applied due to memory issues (more than 18G required) described in Section 6.5.3. The 1-slack algorithm without a proper prior is also impractical as the number of iterations required for converge becomes very large for this size of feature space (over 1000 iterations were run without convergence). The only algorithm that can be applied is the proposed 1-slack-$\boldsymbol{\mu}$ algorithm (it converged in 258 iterations), as the prior yields sensible model parameters when there are few constraints as described in Section 6.5.2. As discussed before, the covariance matrix of Gaussian prior for parameter $\boldsymbol{\alpha}$ in this work is assumed to be a scaled identical matrix, $\boldsymbol{\Sigma} = C\boldsymbol{I}$ (see Section 6.5.2). Thus the hyper-parameter C can be viewed as the variance of parameter $\boldsymbol{\alpha}$. The 1-slack-$\boldsymbol{\mu}$ algorithm allows the variance C to be very small, if a proper $\boldsymbol{\mu}$ is given. This means the number of training iterations can be significantly reduced. For small vocabulary tasks, searching for the best competing hypothesis and segmentation over all possible paths is feasible. However, it is not practical to do this for AURORA4. Therefore the search space of all possible $\mathbf{w}, \boldsymbol{\theta}$ here are restricted by the lattices. Given the lattices, the decoding complexities of structured SVMs and HMMs are in the same order of magnitude.

Model	Param.	Criterion	Test Set WER (%)				Avg
			A	B	C	D	
HMM-VTS	3.98M	ML	7.1	15.3	12.2	23.1	17.8
SCRF	+2210	CML	7.2	14.7	11.1	22.8	17.4
		MPE	7.3	14.7	11.2	22.7	17.4
SSVM	+2210	MM (1-slack-$\boldsymbol{\mu}$)	7.4	14.2	11.3	21.9	16.8

Table 8.10 *AURORA 4 Results based on VTS-compensated HMMs. For structured SVMs, 1-slack-$\boldsymbol{\mu}$ means Algoritms 5+6. All possible hypothesis $\mathbf{w}$ and segmentations $\boldsymbol{\theta}$ are restricted by lattices generated by HMM-VTS.*

8.2.2.2 VAT-based systems

The second configuration used a VTS adaptively trained (VAT) HMM system. Note in this configuration both the generative and discriminative models were trained on

multi-style data. Table 8.11 shows the performance of the baseline VAT system, the SCRFs (Ragni and Gales 2011*b*) and SSVMs based on the same 2210 dimensional joint features. These features were extracted using the likelihoods of VTS adaptively trained HMMs. Comparing the VAT in Table 8.11 (line 1) and the VTS in Table 8.10 (line 1) shows gains of about 2% absolute. Compared to HMM-VAT system and SCRFs, the structured SVMs gain on average about 4% and 2% relative improvements.

Model	Criterion	Test Set WER (%)				Avg
		A	B	C	D	
HMM-VAT	ML	8.6	13.8	12.0	20.1	16.0
SCRF	CML	7.8	13.6	11.3	20.2	15.8
	MPE	7.7	13.5	11.2	20.0	15.7
SSVM	MM (1-slack-$\boldsymbol{\mu}$)	7.5	13.3	11.1	19.6	15.4

Table 8.11 *AURORA 4 Results based on VAT trained HMMs. For structured SVMs, 1-slack-$\boldsymbol{\mu}$ means Algoritms 5+6. All possible hypothesis* $\mathbf{w}$ *and segmentations* $\boldsymbol{\theta}$ *are restricted by lattices generated by HMM-VAT.*

8.2.2.3 Discrimiantive trained VAT-based systems

The third configuration used an MPE and VAT trained HMM system. In this configuration the generative and discriminative models were both discriminatively trained. Table 8.12 shows the performance of baseline MPE-VAT HMMs and structured SVMs based on the same dimensional joint features described in previous configurations. Here the features were extracted using likelihoods of MPE-VAT HMMs. In comparison with the MPE-VAT HMMs, the proposed SSVMs on average yield 2% relative improvement.

8.2.2.4 Clean trained systems

In the fourth configuration both generative and discriminative models were trained and evaluated on the clean part of AURORA4 (the standard 5K WSJ0 setup). This configuration is used to illustrate the performance of proposed SSVMs excluding the noise affects. Structured SVMs were based on the same dimensional joint features de-

Model	Criterion	Test Set WER (%)				Avg
		A	B	C	D	
HMM-MPE-VAT	MPE	7.2	12.8	11.5	19.7	15.3
SSVM	MM (1-slack-$\boldsymbol{\mu}$)	6.9	12.7	11.2	19.4	15.0

Table 8.12 *AURORA 4 Results based on MPE-VAT trained HMMs. For structured SVMs, 1-slack-$\boldsymbol{\mu}$ means Algoritms 5+6. All possible hypothesis* $\mathbf{w}$ *and segmentations* $\boldsymbol{\theta}$ *are restricted by lattices generated by MPE-VAT HMMs.*

scribed in previous configurations. Here the features were extracted using likelihoods of clean HMMs. The WER (%) of the clean HMMs and proposed structured SVMs are shown in Table 8.13. Note the HMM performance 7.3% in Table 8.13 is slightly worse than 7.1%, the VTS set A result in Table 8.10, because VTS on clean data is actually performing utterance-dependent normalisation. The relative improvement is 7%.

Model	Criterion	Test Set WER (%)
HMM	ML	7.3
SSVM	MM (1-slack-$\boldsymbol{\mu}$)	6.8

Table 8.13 *WSJ0 Results based on clean trained HMMs. All possible hypothesis* $\mathbf{w}$ *and segmentations* $\boldsymbol{\theta}$ *are restricted by lattices generated by clean trained HMMs.*

Chapter 9

Conclusion

This thesis investigated discriminative approaches for speech recognition. Previous work in this area is extended in two important directions. First, instead of using CML training which is commonly used for discriminative models, this thesis describes efficient maximum margin training framework for structured discriminative models. Second, unstructured models, SVMs, are extended to sentence-level for continuous speech recognition. We shown that the resulting models in both cases are the same — known as structured SVMs. The major contribution of this work is presented in Chapters 6, 7 and 8. Chapter 6 describes a structured SVM framework suitable for medium to large vocabulary continuous speech recognition. Chapter 7 describes kernelized algorithms for structured SVMs.

An important aspect of structured SVMs is the form of features. Several previously proposed features in the field are summarized in Chapter 5 for this framework . Since some of these features can be extracted based on generative models, this provides an elegant way to combine generative and discriminative models. To apply the structured SVMs to continuous speech recognition, a number of issues need to be addressed. First, features require a segmentation to be specified. To incorporate the optimal segmentation into the training process, the training algorithm is modified making use of the concave-convex optimisation procedure. A Viterbi-style algorithm is described for

inferring the optimal segmentation based on the structured SVM parameters. Second, structured SVMs can be shown as maximum margin log linear models using a zero mean Gaussian prior of the discriminative parameter. However this form of prior is not appropriate for all features. An extended training algorithm is proposed that allows general Gaussian priors to be incorporated into the large margin training. Third, to speed up the training process, strategies of parameter tying, 1-slack optimisation, caching competing hypotheses, lattice constrained search and parallelization, are also described. Finally, to avoid explicitly computing in the high dimensional feature space and to achieve the nonlinear decision boundaries, kernel based training and decoding algorithms are also proposed. The performance of structured SVMs is evaluated on small and medium to large speech recognition tasks: AURORA 2 and 4. The results show that the proposed structured SVMs achieved the best results among all the examined generative and discriminative models, such as HMMs, SVMs and SCRFs.

9.1 Future work

There are many points discussed in this thesis that may benefit from further investigation. A number of suggestions for these future directions are given below.

- In this work the generative model parameters $\boldsymbol{\lambda}$, are assumed to have been trained. Joint learning $\{\boldsymbol{\lambda}, \boldsymbol{\alpha}\}$ in the maximum margin framework will be investigated in the future. The theory of joint maximum margin training is briefly discussed in Appendix A.

- As discussed in Chapters 5 and 7, derivative features can capture long-term dependent, discriminative information. Future work will also include evaluating the derivative kernels for structured SVMs.

- Many recent works in speech recognition are based on DNN-HMM hybrid system. One nature extension to this is to combine DNN with discriminative models. As discussed in Section 5.1.1, one general form of joint feature spaces for

discriminative models is

$$\boldsymbol{\phi}^{\mathtt{ac}}(\mathbf{O}, \mathbf{w}; \boldsymbol{\theta}) = \sum_{t=1}^{T} \boldsymbol{\phi}^{\mathtt{ac}}(\mathbf{o}_t, \theta_t) = \sum_{t=1}^{T} \begin{bmatrix} \vdots \\ \delta(\theta_t = s)\boldsymbol{\psi}(\boldsymbol{o}_t) \\ \vdots \end{bmatrix} \forall s$$

where the frame-level feature $\boldsymbol{\psi}(\boldsymbol{o}_t)$ can be extracted using DNNs,

$$\boldsymbol{\psi}(\boldsymbol{o}_t) = \begin{bmatrix} \vdots \\ P_{\mathtt{DNN}}(s|\boldsymbol{o}_t) \\ \vdots \end{bmatrix}, \forall s$$

The parameters of the model can be learned using the structured SVM algorithms proposed in Chapter 6 (e.g., Algorithm 6). Thus the resulting system can be called the DNN-SSVM hybrid system.

- According to the (Andras 2002), the SVM can be related to neural networks with regularization. The Figure 7.4 also suggests that the structured SVM can be related to hybrid systems with a "shallow" (one layer) neural network. One interesting extension of this work is to investigate the structured SVMs with deep structures. The concept of this model is illustrated in Figure 9.1. In the first iteration, as shown in the top diagram of Figure 9.1, the structured SVM in Chapter 6 with frame-level acoustic and language features can be learned. This step is like a max-margin sequential training of one-layer hybrid systems. In the second iteration, fix the mapping parameters in the one layer, the second-layer and state transition parameters can also be trained using structured SVM algorithms. In the third iteration, as shown in the bottom diagram of Figure 9.1, fix the mapping parameters in the first two layers, the third layer and state transition parameters can be trained in the same way. This process can be continued until the number of "deep" layers is satisfied. The resulting model can be called deep structured SVMs.

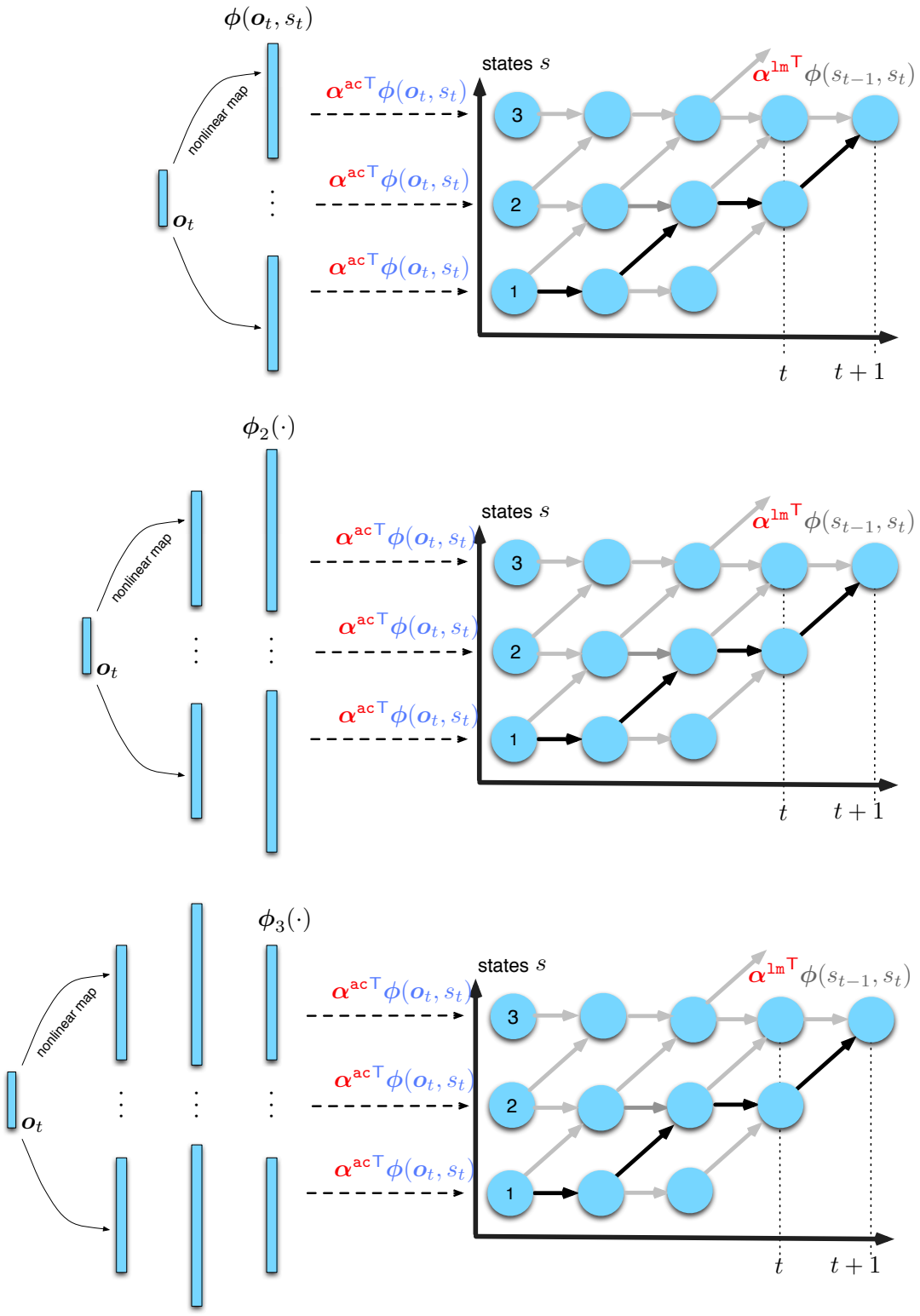


Figure 9.1 *The illustration of deep structured SVMs.*

Appendices

Appendix A

Parameters Joint Estimation of Structured SVMs

Previously, in Section 6.2 the discriminative parameters $\boldsymbol{\alpha}$ and generative parameters $\boldsymbol{\lambda}$ are estimated separately using different objective functions. In this section[1], the generative parameters and discriminative parameters are trained jointly,

$$\left\{\hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\alpha}}\right\} = \arg\min_{\boldsymbol{\lambda},\boldsymbol{\alpha}} \mathcal{F}(\boldsymbol{\alpha}, \boldsymbol{\lambda}). \tag{A.1}$$

where the objective function $\mathcal{F}(\boldsymbol{\alpha}, \boldsymbol{\lambda})$ is extended from equation (6.9)

$$\begin{aligned}\mathcal{F}(\boldsymbol{\alpha}, \boldsymbol{\lambda}) = C||\boldsymbol{\alpha}||^2 + C'||\boldsymbol{\lambda}||^2 + \frac{1}{R}\sum_{r=1}^{R}\Big[- \max_{\boldsymbol{\theta}} \left(\boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}_{\mathtt{ref}}^{(r)}; \boldsymbol{\theta}, \boldsymbol{\lambda})\right) \\ + \max_{\mathbf{w}\neq\mathbf{w}_{\mathtt{ref}},\boldsymbol{\theta}} \left\{\mathcal{L}(\mathbf{w}, \mathbf{w}_{\mathtt{ref}}^{(r)}) + \boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}; \boldsymbol{\theta}, \boldsymbol{\lambda})\right\}\Big]_+\end{aligned} \tag{A.2}$$

where $\boldsymbol{\phi}(\mathbf{O}, \mathbf{w}; \boldsymbol{\theta}, \boldsymbol{\lambda})$ could be any features described in Chapter 5. If the features are not based on the generative models, e.g., log likelihood features and derivative

[1] This section will be our future work and only the theory is discussed.

features, it becomes $\phi(\mathbf{O}, \mathbf{w}; \boldsymbol{\theta})$. Using the coordinate descent method, minimising equation (A.2) with respective to $\boldsymbol{\lambda}$ and $\boldsymbol{\alpha}$ can be convert to solve the following two optimisation problems, alternatively,

$$\hat{\boldsymbol{\alpha}}^{[\tau]} = \arg\min_{\boldsymbol{\alpha}} \left\{ \mathcal{F}(\boldsymbol{\alpha}, \boldsymbol{\lambda}^{[\tau]}) \right\}, \tag{A.3}$$

$$\hat{\boldsymbol{\lambda}}^{[\tau+1]} = \arg\min_{\boldsymbol{\lambda}} \left\{ \mathcal{F}(\boldsymbol{\alpha}^{[\tau]}, \boldsymbol{\lambda}) \right\}. \tag{A.4}$$

where $[\tau]$ denotes the τ-th iteration and the estimation from the maximum likelihood criterion (2.28) could be used as the initialization $\boldsymbol{\lambda}^{[0]}$. The iteration will stop when $\Delta\mathcal{F}(\boldsymbol{\alpha}^{[\tau]}, \boldsymbol{\lambda}^{[\tau]}) < \epsilon$.

Note that given the $\boldsymbol{\lambda}^{[\tau]}$, equation (A.3) is actually the large margin training objective function of $\boldsymbol{\alpha}$ in equation (6.9). The term $||\boldsymbol{\lambda}||$ in equation (A.2) is the regularization. According to (Keshet and Bengio 2008; Sha and Saul 2007), the parameters can also be regularized through $\sum_{jm} \text{trace}(\boldsymbol{\Lambda}_{jm})$, where $\boldsymbol{\Lambda}_{jm}$ is reparameterization matrix derived from $\{c_{jm}, \boldsymbol{\mu}_{jm}, \boldsymbol{\Sigma}_{jm}\}$ of component m of state j in HMM,

$$\boldsymbol{\Lambda}_{jm} = \begin{bmatrix} \boldsymbol{\Sigma}_{jm}^{-1} & -\boldsymbol{\Sigma}_{jm}^{-1}\boldsymbol{\mu}_{jm} \\ -\boldsymbol{\mu}_{jm}^{\mathsf{T}}\boldsymbol{\Sigma}_{jm}^{-1} & \boldsymbol{\mu}_{jm}^{\mathsf{T}}\boldsymbol{\Sigma}_{jm}^{-1}\boldsymbol{\mu}_{jm} + \vartheta_{jm} \end{bmatrix} \tag{A.5}$$

$$\vartheta_{jm} = \log c_{jm} - \frac{1}{2}\log(|\boldsymbol{\Sigma}_{jm}|) - \frac{D}{2}\log(2\pi). \tag{A.6}$$

Therefore given $\boldsymbol{\alpha}^{[\tau]}$, the joint large margin training criterion for generative parameters $\boldsymbol{\lambda}$ in equation (A.4) can be expressed as

$$\mathcal{F}(\boldsymbol{\alpha}^{[\tau]}, \boldsymbol{\lambda}) = C' \sum_{j,m} \text{tr}(\boldsymbol{\Lambda}_{jm}) + \frac{1}{R}\sum_{r=1}^{R} \Big[-\max_{\boldsymbol{\theta}} \left({\boldsymbol{\alpha}^{[\tau]}}^{\mathsf{T}} \boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}_{\mathtt{ref}}^{(r)}; \boldsymbol{\theta}, \boldsymbol{\lambda}) \right) \tag{A.7}$$
$$+ \max_{\mathbf{w} \neq \mathbf{w}_{\mathtt{ref}}, \boldsymbol{\theta}} \left\{ \mathcal{L}(\mathbf{w}, \mathbf{w}_{\mathtt{ref}}^{(r)}) + {\boldsymbol{\alpha}^{[\tau]}}^{\mathsf{T}} \boldsymbol{\phi}(\mathbf{O}^{(r)}, \mathbf{w}; \boldsymbol{\theta}, \boldsymbol{\lambda}) \right\} \Big]_{+}$$

Note that the large margin training in equation A.7 is different from the large margin training of HMM described in Section 2.3.5 (Sha and Saul 2007):

$$\mathcal{F}_{\mathtt{LM-hmm}}(\boldsymbol{\lambda}) = C' \sum_{j,m} \text{tr}(\boldsymbol{\Lambda}_{jm}) + \frac{1}{R}\sum_{r=1}^{R} \Big[-\log p(\mathbf{O}^{(r)} | \mathbf{w}_{\mathtt{ref}}^{(r)}; \boldsymbol{\lambda}) \tag{A.8}$$
$$+ \min_{\mathbf{w} \neq \mathbf{w}_{\mathtt{ref}}^{(r)}} \left\{ \mathcal{L}(\mathbf{w}, \mathbf{w}_{\mathtt{ref}}^{(r)}) + \log p(\mathbf{O}^{(r)} | \mathbf{w}; \boldsymbol{\lambda}) \right\} \Big]_{+}.$$

Comparing with the large margin HMM in equation (A.8), the joint large margin training of $\boldsymbol{\lambda}$ in equation (A.7) is more general in two aspects. First, if the log likelihood features in Section 5.1.2 are used, each log likelihood in equation A.8 is just a element of the features in equation (A.7). Second, to obtain a convex optimization for large margin training, in (Sha and Saul 2007) the log likelihood is calculated through the Viterbi path:

$$\log p(\mathbf{O}|\mathbf{w};\boldsymbol{\lambda}) \approx \sum_{t=1}^{T} \log a(s_{t-1}, s_t) - \sum_{t=1}^{T} \boldsymbol{z}^{\mathsf{T}} \boldsymbol{\Lambda}_{s_t} \boldsymbol{z} \tag{A.9}$$

where $\boldsymbol{s} = [s_1, s_2, \ldots, s_T]$ is the state sequence for $(\mathbf{O}, \mathbf{w})$ generated by Viterbi algorithm under the maximum likelihood criteria, a_{ij} is the transition probability, $\boldsymbol{z} = [\boldsymbol{o}_t \; 1]^{\mathsf{T}}$ and $\boldsymbol{\Lambda}$ is defined in equation A.5. However, in equation (A.7) the segmentation $\boldsymbol{\theta}$ is optimized under the large margin criteria as described in Section 6.2.1.3. Note that $\boldsymbol{z}^{\mathsf{T}} \boldsymbol{\Lambda}_{s_t} \boldsymbol{z}$ in equation A.9 is linear in parameters $\boldsymbol{\Lambda}_{jm}$. If the log likelihood features in equation (5.1.2) are used, substituting equation (A.9) to every element of features, the optimization in equation (A.7) will also become convex for generative parameters $\boldsymbol{\lambda}$. Comparing to Figure 6.6, the procedure of the joint training process is demonstrated in the following diagram.

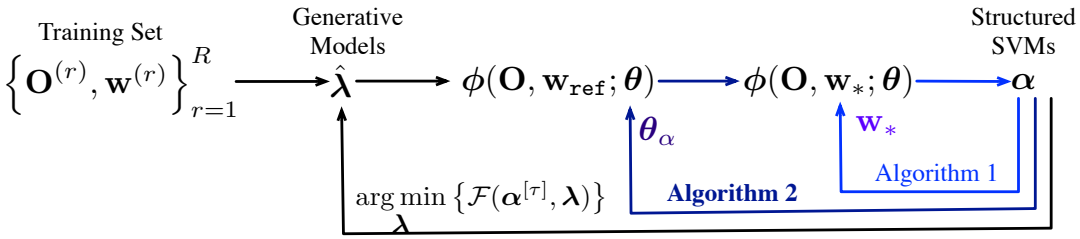


Figure A.1 *The joint training process for generative parameter $\boldsymbol{\lambda}$ and structured SVM parameters $\boldsymbol{\alpha}$.*

Appendix B

Convergence of Training Structured SVM with Optimal Segmentation

For simplicity, the proof considers one utterance r only. Since the regularization term $\frac{1}{2}||\boldsymbol{\alpha}||_2^2$ and the summation of all the training utterances will not affect the convexity, incorporating them in the following derivation is simple. Thus, here our target is *minimising* the following objective,

$$\Bigg[\underbrace{-\max_{\boldsymbol{\theta}^{(r)}}\Big(\boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O}^{(r)},\mathbf{w}_{\mathtt{ref}}^{(r)},\boldsymbol{\theta}^{(r)};\boldsymbol{\lambda})\Big)}_{\text{concave: } N(\boldsymbol{\alpha})}+\underbrace{\max_{\mathbf{w}\neq\mathbf{w}_{\mathtt{ref}},\boldsymbol{\theta}}\Big\{\mathcal{L}(\mathbf{w},\mathbf{w}_{\mathtt{ref}}^{(r)})+\boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O}^{(r)},\mathbf{w},\boldsymbol{\theta};\boldsymbol{\lambda})\Big\}}_{\text{convex: } D(\boldsymbol{\alpha})}\Bigg] \tag{B.1}$$

Instead of solving the above non-convex problem, in Algorithm 1, given the current $\boldsymbol{\alpha}^{[\tau]}$, we *first* optimise reference alignment $\boldsymbol{\theta}^{(r)}$ for training pair $(\mathbf{O}^{(r)},\mathbf{w}_{\mathtt{ref}}^{(r)})$ as

$$\hat{\boldsymbol{\theta}}^{(r)[\tau]}=\arg\max_{\boldsymbol{\theta}^{(r)}}{\boldsymbol{\alpha}^{[\tau]}}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O}^{(r)},\mathbf{w}_{\mathtt{ref}}^{(r)},\boldsymbol{\theta}^{(r)};\boldsymbol{\lambda}) \tag{B.2}$$

Secondly, fixing $\hat{\boldsymbol{\theta}}^{(r)[\tau]}$, Algorithm 1 optimise $\boldsymbol{\alpha}^{[\tau+1]}$ by *minimizing* the following convex upper bound,

$$\Bigg[\underbrace{-\left(\boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O}^{(r)},\mathbf{w}_{\mathtt{ref}}^{(r)},\hat{\boldsymbol{\theta}}^{(r)[\tau]};\boldsymbol{\lambda})\right)}_{\text{Linear: } L(\boldsymbol{\alpha})=\nabla N(\boldsymbol{\alpha}^{[\tau]})^{\mathsf{T}}\boldsymbol{\alpha}}+\underbrace{\max_{\mathbf{w}\neq\mathbf{w}_{\mathtt{ref}},\boldsymbol{\theta}}\left\{\mathcal{L}(\mathbf{w},\mathbf{w}_{\mathtt{ref}}^{(r)})+\boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O}^{(r)},\mathbf{w},\boldsymbol{\theta};\boldsymbol{\lambda})\right\}}_{\text{convex: } D(\boldsymbol{\alpha})}\Bigg] \tag{B.3}$$

According to the Eq.(B.2), the linear part $L(\boldsymbol{\alpha})$ and numerator part $N(\boldsymbol{\alpha})$ has the following relationship,

$$\boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O},\mathbf{w}_{\mathtt{ref}},\hat{\boldsymbol{\theta}}^{[\tau]})=L(\boldsymbol{\alpha})=\left[\nabla N(\boldsymbol{\alpha}^{[\tau]})\right]^{\mathsf{T}}\boldsymbol{\alpha}=\left[\nabla\max_{\boldsymbol{\theta}}\left(\boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{O},\mathbf{w}_{\mathtt{ref}},\boldsymbol{\theta})\right)|_{\boldsymbol{\alpha}^{[\tau]}}\right]^{\mathsf{T}}\boldsymbol{\alpha} \tag{B.4}$$

Denote $\boldsymbol{\alpha}^{[\tau+1]}$ is the solution of above convex optimisation, we have

$$\nabla D(\boldsymbol{\alpha}^{[\tau+1]})+\nabla L(\boldsymbol{\alpha}^{[\tau+1]})=0 \tag{B.5}$$

Substitute Eq.(B.4) into Eq.(B.5), we have

$$\nabla D(\boldsymbol{\alpha}^{[\tau+1]})+\nabla\left\{\nabla N(\boldsymbol{\alpha}^{[\tau]})^{\mathsf{T}}\boldsymbol{\alpha}\right\}=\nabla D(\boldsymbol{\alpha}^{[\tau+1]})+\nabla N(\boldsymbol{\alpha}^{[\tau]})=0 \tag{B.6}$$

Because numerator term $N(\boldsymbol{\alpha})$ is concave and denominator term $D(\boldsymbol{\alpha})$ is convex, for any $\boldsymbol{\alpha}^{[\tau]}$ and $\boldsymbol{\alpha}^{[\tau+1]}$ we have,

$$\frac{N(\boldsymbol{\alpha}^{[\tau+1]})-N(\boldsymbol{\alpha}^{[\tau]})}{\boldsymbol{\alpha}^{[\tau+1]}-\boldsymbol{\alpha}^{[\tau]}}\leq\nabla N(\boldsymbol{\alpha}^{[\tau]}) \tag{B.7}$$

$$\frac{D(\boldsymbol{\alpha}^{[\tau]})-D(\boldsymbol{\alpha}^{[\tau+1]})}{\boldsymbol{\alpha}^{[\tau]}-\boldsymbol{\alpha}^{[\tau+1]}}\geq\nabla D(\boldsymbol{\alpha}^{[\tau+1]}) \tag{B.8}$$

Therefore,

$$N(\boldsymbol{\alpha}^{[\tau]})+D(\boldsymbol{\alpha}^{[\tau]})\geq N(\boldsymbol{\alpha}^{[\tau+1]})+D(\boldsymbol{\alpha}^{[\tau+1]})+\left(\boldsymbol{\alpha}^{[\tau]}-\boldsymbol{\alpha}^{[\tau+1]}\right)\left(\nabla D(\boldsymbol{\alpha}^{[\tau+1]})+\nabla N(\boldsymbol{\alpha}^{[\tau]})\right) \tag{B.9}$$

If $\boldsymbol{\alpha}^{[\tau]}$ and $\boldsymbol{\alpha}^{[\tau+1]}$ are optimised using the Algorithm 1, they satisfy Eq.(B.6). Substitute Eq.(B.6) into the above equation, we have

$$N(\boldsymbol{\alpha}^{[\tau]})+D(\boldsymbol{\alpha}^{[\tau]})\geq N(\boldsymbol{\alpha}^{[\tau+1]})+D(\boldsymbol{\alpha}^{[\tau+1]}) \tag{B.10}$$

which means every iteration the new $\boldsymbol{\alpha}^{[\tau+1]}$ will decrease the objective function.

Bibliography

Y. H. Abdel-Haleem (2006). *Conditional Random Fields For Continuous Speech Recognition*. Ph.D. thesis, University of Sheffield.

A. Acero (1993). *Acoustical and Environmental Robustness in Automatic Speech Recognition*. Kluwer Academic Publishers.

A. Acero, L. Deng, T. Kristjansson, and J. Zhang (2000). "HMM Adaptation using Vector Taylor Series for Noisy Speech Recognition." In *Proceedings of ICSLP*. vol. 3, pp. 869–872.

M. Aizerman, E. Braverman, and L. Rozonoer (1964). "Theoretical Foundations of rhe Potential Function Method in Pattern Recognition Learning." *Automation and remote control* 25, pp. 821–837.

H. Aldamarki (2009). *Filter Trees for Noise Robust Small Vocabulary Speech Recognition*. Ph.D. thesis, Cambridge University.

P. Andras (2002). "The Equivalence of Support Vector Machine and Regularization Neural Networks." *Neural Processing Letters* 15 (2), pp. 97–104.

E. Arisoy, M. Saraclar, B. Roark, and I. Shafran (2010). "Syntactic and Sub-Lexical Features for Turkish Discriminative Language Models." In *Proceedings of ICASSP*. Dallas, Texas, USA, pp. 5538–5541.

L. R. Bahl, P. F. Brown, P. V. de Souza, and R. L. Mercer (1986). "Maximum Mutual Information Estimation of Hidden Markov Model Parameters for Speech Recognition." In *Proceedings of ICASSP*. Tokyo, Japan, pp. 49–52.

C. Bahlmann, B. Haasdonk, and H. Burkhardt (2002). "On-line Handwriting Recognition with Support Vector Machines - A Kernel Approach." In *In Proceedings of IWFHR*. pp. 49–54.

J. K. Baker (1975). "The Dragon System - An Overview." *IEEE Transactions on Acoustics, Speech and Signal Processing* ASSP-23 (1), pp. 24–29.

G. H. Bakir, T. Hofmann, B. Schölkopf, A. J. Smola, B. Taskar, and S. V. N. Vishwanathan (2007). *Predicting Structured Data (Neural Information Processing)*. The MIT Press.

L. E. Baum, T. Petrie, G. Soules, and N. Weiss (1970). "A Maximisation Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains." *Annals of Mathematical Statistics* 41, pp. 164–171.

Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin (2003). "A Neural Probabilistic Language Model." *Journal of Machine Learning Research* 3, pp. 1137–1155.

A. L. Berger, S. A. Della Pietra, and V. J. Della Pietra (1996). "A Maximum Entropy Approach to Natural Language Processing." *Computational Linguistics* 22, pp. 39–71.

P. Beyerlein (1997). "Discriminative Model Combination." In *Proceedings of ASRU*. IEEE, pp. 238–245.

A. Beygelzimer, J. Langford, and P. Ravikumar (2007). "Multiclass Classification with Filter Trees." In *Proceedings of workshop on Information Theory and Applications*.

J. A. Bilmes (2003). "Buried Markov Models: A Graphical-Modelling Approach to Automatic Speech Recognition." *Computer Speech and Language* 17 (2-3), pp. 213–231.

O. Birkenes (2007). *A Framework for Speech Recognition Using Logistic Regression*. Ph.D. thesis, Norwegian University of Science and Technology.

O. Birkenes, T. Matsui, and K. Tanabe (2006). "Isolated-Word Recognition with Penalized Logistic Regression Machines." In *ICASSP*. vol. 1, pp. 405–408.

C. M. Bishop (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

B. E. Boser, I. M. Guyon, and V. Vapnik (1992). "A Training Algorithm For Optimal Margin Classifiers." In *Proceedings of Computational Learning Theory*. pp. 144–152.

L. Bottou, C. Cortes, J. Denker, H. Drucker, I. Guyon, L. Jackel, Y. LeCun, U. Muller, E. Sackinger, P. Simard, and V. Vapnik (1994). "Comparison of Classifier Methods: A Case Study in Handwritting Digit Recognition." In *Proceedings of the International Conference On Pattern Recognition*. Jerusalem, Israel, vol. 2, pp. 77–82.

H. Bourlard and N. Morgan (1994). *Connectionist Speech Recognition: A Hybrid Approach*, vol. 247. Springer.

H. Bourlard and N. Morgan (1998). "Hybrid HMM/ANN Systems for Speech Recognition: Overview and New Research Directions." In *Adaptive Processing of Sequences and Data Structures*, Springer, pp. 389–417.

S. Boyd and L. Vandenberghe (2009). *Convex Optimization*. Cambridge University Press.

P. F. Brown, V. J. Della Pietra, P. V. de Souza, J. C. Lai, and R. L. Mercer (1992). "Class-Based N-Gram Models of Natural Language." *Computational Linguistics* 18 (4), pp. 467–479.

C. J. C. Burges (1998). "A Tutorial on Support Vector Machines for Pattern Recognition." *Data Mining and Knowledge Discovery* 2, pp. 121–167.

W. Byrne (2006). "Minimum Bayes Risk Estimation and Decoding in Large Vocabulary Continuous Speech Recognition." *IEICE Transactions* 89-D (3), pp. 900–907.

W. M. Campbell, D. E. Sturim, and D. A. Reynolds (2006). "Support Vector Machines Using GMM Supervectors for Speaker Verification." *IEEE Signal processing letters* 13, pp. 308–311.

O. Chapelle, P. Haffner, and V. Vapnik (1999). "SVMs for Histogram-Based Image Classification."

C. Chelba and A. Acero (2006). "Adaptation of Maximum Entropy Capitalizer: Little Data Can Help a Lot." *Computer Speech and Language* 20, pp. 382–399.

S. F. Chen and J. T. Goodman (1998). "An Empirical Study of Smoothing Techniques for Language Modelling." Tech. Rep. TR-10-98, Harvard University.

S. F Chen and R. Rosenfeld (1999). "Efficient Sampling and Feature Selection in Whole Sentence Maximum Entropy Language Models." In *Proceedings of ICASSP*. IEEE, vol. 1, pp. 549–552.

S. S. Chen and R. A. Gopinath (1997). "Model Selection in Acoustic Modelling." In *Proceedings of European Conference on Speech Communication and Technology*. Rhodes, Greece, pp. 1087–1090.

Y. Cho and L. K. Saul (2010). "Large-margin Classification in Infinite Neural Networks." *Neural Computation* 22 (10), pp. 2678–2697.

W. Chou, C. H. Lee, and B. H. Juang (1993). "Minimum Error Rate Training based on N-Best String Models." In *Proceedings of ICASSP*. Minneapolis, USA, pp. 652–655.

M. Collins, B. Roark, and M. Saraclar (2005). "Discriminative Syntactic Language Modeling for Speech Recognition." In *Proceedings of ACL*. Ann Arbor, USA, pp. 507–514.

C. Cortes and V. Vapnik (1995). "Support Vector Networks." *Machine Learning* pp. 273–297.

K. Crammer and Y. Singer (2001). "On the Algorithmic Implementation of Multiclass Kernel-Based Vector Machines." In *Journal of Machine Learning Research*, MIT Press, vol. 2, pp. 265–292.

G. E. Dahl, D. Yu, L. Deng, and A. Acero (2012). "Context-dependent Pre-trained Deep Neural Networks for Large Vocabulary Speech Recognition." *IEEE Transactions on Audio, Speech, and Language Processing* 20 (1), pp. 30–42.

J. N. Darroch and D. Ratcliff (1972). "Generalised Iterative Scaling for Log-Linear Models." *The Annals of Mathematical Statistics* pp. 1470–1480.

S. B. Davis and P. Mermelstein (1980). "Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences." *IEEE Transactions on Speech and Audio Processing* 28 (4), pp. 357–366.

J. R. Deller, J. G. Proakis, and J. H. L. Hansen (2000). *Discrete-Time Processing of Speech Signals*. IEEE New York, NY, USA.

A. P. Dempster, N. M. Laird, and D. B. Rubin (1977). "Maximum Likelihood from Incomplete Data via the EM Algorithm." *Journal of the Royal Statistical Society* 39, pp. 1–39.

R. O. Duda, P. E. Hart, and D. G. Stork (2001). *Pattern Classification*. Wiley-Interscience, 2nd edn.

P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan (2010). "Object Detection with Discriminatively Trained Part-Based Models." *IEEE Transaction on Patern Analysis and Machine Intelligence* 32, pp. 1627–1645.

S. Fine and K. Scheinberg (2002). "Efficient SVM Training using Low-Rank Kernel Representations." *The Journal of Machine Learning Research* 2 (1), pp. 243–264.

F. Flego and M. J. F. Gales (2009). "Discriminative Adaptive Training with VTS and JUD." In *Proceedings of ASRU*. IEEE, Merano, Italy, pp. 170–175.

F. Flego and M. J. F. Gales (2011). "Factor Analysis based VTS and JUD Noise Estimation and Compensation." Tech. Rep. CUED/F-INFENG/TR653, Cambridge University. Available from: http://mi.eng.cam.ac.uk/∼mjfg.

Jr. G. D. Forney (1973). "The Viterbi Algorithm." *Proceedings of the IEEE* 61 (3), pp. 268–278.

B. L Fox (1968). *Semi-Markov Processes*. RAND Corporation.

N. Friedman, D. Geiger, and M. Goldszmidt (1997). "Bayesian Network Classifiers." *Journal of Machine learning* 29 (2-3), pp. 131–163.

S. Furui (1981). "Cepstral Analysis Technique for Automatic Speaker Verification." *IEEE Transactions on Acoustics, Speech, and Signal Processing* 29, pp. 254–272.

M. J. F. Gales (1995). *Model-based Techniques for Noise Robust Speech Recognition*. Ph.D. thesis, Cambridge University.

M. J. F. Gales (1998). "Maximum Likelihood Linear Transformations for HMM-based Speech Recognition." *Computer Speech and Language* 12 (2), pp. 75–98.

M. J. F. Gales (2007). "Discriminative Models for Speech Recognition." In *Proceedings of ITA Workshop*. University of California, San Diego, USA, pp. 170–176.

M. J. F. Gales (2011). "Model-based Approaches to Handling Uncertainty." In *Robust Speech Recognition of Uncertain or Missing Data*, Springer, pp. 101–125.

M. J. F. Gales and F. Flego (2008). "Discriminative Classifiers With Generative Kernels For Noise Robust Speech Recognition." Tech. Rep. CUED/F-INFENG/TR605, Cambridge University.

M. J. F. Gales and F. Flego (2010). "Discriminative Classifiers With Adaptive Kernels For Noise Robust Speech Recognition." *Computer Speech and Language* 24 (4), pp. 648–662.

M. J. F. Gales, D. Y. Kim, P. C. Woodland, H. Y. Chan, D. Mrva, R. Sinha, and S. E. Tranter (2006). "Progress in the CU-HTK Broadcast News transcription system." *IEEE Transactions Audio, Speech and Language Processing* 14 (5), pp. 1513–1525.

M. J. F. Gales, A. Ragni, H. A. l. Damarki, and C. Gautier (2009). "Support Vector Machines for Noise Robust ASR." In *Proceedings of ASRU*. IEEE, pp. 205–210.

M. J. F. Gales, S. Watanabe, and E. Fosler-Lussier (2012). "Structured Discriminative Models For Speech Recognition." *Signal Processing Magazine, IEEE* 29 (6), pp. 70–81.

M. J. F. Gales and S. J. Young (1996). "Robust Continuous Speech Recognition using Parallel Model Combination." *IEEE Transactions on Speech and Audio Processing* 4 (5), pp. 352–359.

M. J. F. Gales and Steve Young (2007). "The Application of Hidden Markov Models in Speech Recognition." In *Foundations and Trends in Signal Processing*.

A. Ganapathiraju, J. Hamaker, and J. Picone (2000). "Hybrid SVM/HMM Architectures for Speech Recognition." In *Proceedings of International Conference on Spoken Language Processing (ICSLP)*. Beijing, China.

C. Gautier (2008). *Filter Trees for Combining Binary CLassifiers*. Ph.D. thesis, Cambridge University.

L.-J. Gauvain and C.-H. Lee (1994). "Maximum a Posteriori Estimation for Multivariate Gaussian Mixture Observations of Markov Chains." *IEEE Transactions on Speech and Audio Processing* 2 (2), pp. 291–298.

Z. Ghahramani and M. I. Jordan (1997). "Factorial Hidden Markov Models." *Journal of Machine Learning* 29, pp. 245–273.

L. Gillick and S. J. Cox (1989). "Some Statistical Issues in The Comparison of Speech Recognition Algorithms." In *Proceedings of ICASSP*. IEEE, pp. 532–535.

F. Girosi (1998). "An Equivalence Between Sparse Approximation and Support Vector Machines." *Neural computation* 10 (6), pp. 1455–1480.

A. Globerson, T. Y. Koo, X. Carreras, and M. Collins (2007). "Exponentiated Gradient Algorithms for Log-Linear Structured Prediction." In *Proceedings of ICML*. pp. 305–312.

N. K. Goel and A.G. Andreou (1998). "Heteroscedastic Discriminant Analysis and Reduced-Rank HMMs for Improved Speech Recognition." *Speech Communications* 26, pp. 283–297.

I. J. Good (1953). "The Population Frequency of Species and the Estimation of Population Parameters." *Biometrika* 40, pp. 237–264.

Y. Grandvalet, J. Mariéthoz, and S. Bengio (2006). "A Probabilistic Interpretation of SVMs with an Application to Unbalanced Classification." *Advances in Neural Information Processing Systems* .

F. Grézl, M. Karafiát, S. Kontár, and J. Cernocky (2007). "Probabilistic and Bottle-neck Features for LVCSR of Meetings." In *Proceedings of ICASSP*. vol. 4, pp. IV–757.

A. Gunawardana, M. Mahajan, A. Acero, and John C. Platt (2005). "Hidden Conditional Random Fields for Phone Classification." In *Proceedings of Interspeech*. Lisbon, Portugal, pp. 1117–1120.

G. Heigold (2010). *A Log-Linear Discriminative Modeling Framework for Speech Recognition*. Ph.D. thesis, RWTH Aachen University, Aachen, Germany.

G. Heigold, R. Schlüter, and H. Ney (2007). "On the Equivalence of Gaussian HMM and Gaussian HMM-like Hidden Conditional Random Fields." In *Proceedings of Interspeech*. pp. 1721–1724.

G. Heigold, G. Zweig, X. Li, and P. Nguyen (2009). "A Flat Direct Model For Speech Recognition." In *ICASSP*. pp. 3861–3864.

H. Hermansky (1990). "Perceptual Linear Predictive (PLP) Analysis of Speech." *Journal of the Acoustic Society of America* 87 (4), pp. 1738–1752.

H. Hermansky, D. P. W. Ellis, and S. Sharma (2000). "Tandem Connectionist Feature Extraction For Conventional HMM Systems." In *Proceedings of ICASSP*. IEEE, vol. 3, pp. 1635–1638.

Y. Hifny and S. Renals (2009). "Speech Recognition Using Augmented Conditional Random Fields." *IEEE Transactions on Audio, Speech, and Language Processing* 17 (2), pp. 354–365.

Y. Hifny, S. Renals, and N. D. Lawrence (2005). "A Hybrid Maxent/HMM based ASR System." In *Proceedings of Interspeech*. pp. 3017–3020.

G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, *et al.* (2012). "Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups." *Signal Processing Magazine* 29 (6), pp. 82–97.

G. Hinton, S. Osindero, and Y.-W. Teh (2006). "A Fast Learning Algorithm for Deep Belief Nets." *Neural computation* 18 (7), pp. 1527–1554.

T. Hofmann, B. Schölkopf, and A. J. Smola (2008). "Kernel Methods in Machine Learning." *The annals of statistics* 36, pp. 1171–1220.

T. Jaakkola, M. Diekhans, and D. Haussler (1999). "Using the Fisher Kernel Method to Detect Remote Protein Homologies." In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*. AAAI Press, pp. 149–158.

T. Jaakkola and D. Haussler (1999). "Exploiting Generative Models in Disciminative Classifiers." In D. A. Kohn (ed.), *Advances in Neural Information Processing Systems*, MIT Press, pp. 487–493.

E. T. Jaynes (2003). *Probability Theory: The Logic of Science*. Cambridge University Press.

T. Jebara (2001). *Discriminative, Generative and Imitative Learning*. Ph.D. thesis, MIT, Media Lab, USA.

F. Jelinek (1969). "A Fast Sequential Decoding Algorithm Using A Stack." *IBM Journal on Research and Development* 13 (6), pp. 675–685.

F. Jelinek (1998). *Statistical Methods for Speech Recognition*. MIT Press.

T. Joachims (1999). "Making Large-Scale SVM Learning Practical." In B. Schölkopf, C. Burges, and A. Smola (eds.), *Advances in Kernel Methods - Support Vector Learning*, MIT Press, Cambridge, MA, chap. 11, pp. 169–184.

T. Joachims, T. Finley, and C.-N. J. Yu (2009). "Cutting-Plane Training of Structural SVMs." *Journal of Machine Learning* 77 (1), pp. 27–59.

M. I. Jordan *et al.* (1995). "Why The Logistic Function? A Tutorial Discussion on Probabilities and Neural Networks."

B.-H. Juang, W. Chou, and C.-H. Lee (1995). "Minimum Classification Error Rate Methods for Speech Recognition." *IEEE Transactions on speech and audio processing* 5 (3), pp. 257–265.

B.-H. Juang and S. Katagiri (1992). "Discriminative Learning for Minimum Error Classification." *IEEE Transactions on Signal Processing* 40 (12), p. 3043.

A. Kabán (2007). "On Bayesian Classification with Laplace Priors." *Pattern Recognition Letters* 28 (10), pp. 1271–1282.

J. Kaiser, B. Horvat, and Z. Kačič (2000). "A Novel Loss Function for The Overall Risk Criterion Based Discriminative Training of HMM Models." In *Proceedings of ICSLP*. Beijing, China, vol. 2, pp. 887–890.

O. Kalinli, M. L. Seltzer, and A. Acero (2009). "Noise Adaptive Training Using a Vector Taylor Series Approach For Noise Robust Automatic Speech Recognition." In *ICASSP*. pp. 3825–3828.

S. M. Katz (1987). "Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer." In *IEEE Transactions on Acoustics, Speech and Signal Processing*. vol. 35, pp. 400–401.

J. Keshet and S. Bengio (2008). *Automatic Speech and Speaker Recognition: Large Margin and Kernel Methods*. John Wiley & Sons.

J. Keshet, C.-C. Cheng, M. Stoehr, and D. A. McAllester (2011). "Direct Error Rate Minimization of Hidden Markov Models." In *Proceedings of Interspeech*. pp. 449–452.

S. Khudanpur and J. Wu (2000). "Maximum Entropy Techniques for Exploiting Syntactic, Semantic and Collocational Dependencies in Language Modeling." *Computer Speech and Language* 14 (4), pp. 355–372.

B. Kijsirikul and S. Abe (2002). "Multiclass Support Vector Machines using Adaptive Directed Acyclic Graphs." In *International Joint Conference on Neural Network*.

K. Knill, M. J. F. Gales, S. Rath, P. C. Woodland, C. Zhang, , and S.-X. Zhang (2013). "Investigation of Multilingual Deep Neural Networks for Spoken Term Detection." In *Proceedings of ASRU*. Olomouc, Czech Republic.

A. Krogh, M. Brown, I. S. Mian, K. Sjlander, and D. Haussler (1994). "Hidden Markov Models in Computational Biology: Applications to Protein Modeling." *Journal of Molecular Biology* 235, pp. 1501–1531.

Y. Kubo, S. Watanabe, A. Nakamura, E. McDermott, and T. Kobayashi (2010). "A Sequential Pattern Classifier Based on Hidden Markov Kernel Machine and Its Application to Phoneme Classification." *Journal of Selected Topics in Signal Processing* 4 (6), pp. 974–984.

Y. Kubo, S. Wiesler, R. Schluter, H. Ney, S. Watanabe, A. Nakamura, and T. Kobayashi (2011). "Subspace Pursuit Method for Kernel-Log-Linear Models." In *ICASSP*. pp. 4500–4503.

N. Kumar (1997). *Investigation of Silicon-Auditory Models and Generalization of Linear Discriminant Analysis for Improved Speech Recognition*. Ph.D. thesis, Johns Hopkins University.

S. Kumar and W. J. Byrne (2002). "Risk-based Lattice Cutting for Segmental Minimum Bayes-Risk Decoding." In *Proceedings of International Conference on Spoken Language Processing (ICSLP)*. Denver, Colorado, USA.

S. Lacoste-Julien (2010). *Discriminative Machine Learning with Structure*. Ph.D. thesis, EECS Department, University of California, Berkeley.

J. D. Lafferty, A. McCallum, and F. C. N. Pereira (2001). "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data." In *Proceedings of ICML*. San Francisco, USA, pp. 282–289.

M. Layton (2006). *Augmentd Statistical Models for Classifying Sequence Data*. Ph.D. thesis, Cambridge University.

L. Lee and R. C. Rose (1996). "Speaker Normalization Using Efficient Frequency Warping Procedures." In *Proceedings of ICASSP*. vol. 1, pp. 353–356.

C. J. Leggetter and P. C. Woodland (1995). "Maximum Likelihood Linear Regression for Speaker Adaptation of Continuous Density Hidden Markov Models." *Computer Speech and Language* 9, pp. 171–185.

J. Li, M. Yuan, and C.-H. Lee (2007). "Approximate Test Risk Bound Minimization Through Soft Margin Estimation." *IEEE Transactions on Audio, Speech and Language Processing* 15 (8), pp. 2393–2404.

H. Liao and M. J. F. Gales (2006). "Joint Uncertainty Decoding for Robust Large Vocabulary Speech Recognition." Tech. Rep. CUED/F-INFENG/TR552, Cambridge University.

X. Liu and M. J. F. Gales (2007). "Automatic Model Complexity Control using Marginalized Discriminative Growth Functions." *IEEE Transactions on Speech and Audio Processing* 15 (4), pp. 1414–1424.

H. Lodhi (2002). "Text Classification using String Kernels." *Journal of machine learning research* 2, pp. 419–444.

C. Longworth (2010). *Kernel Methods for Text-Independent Speaker Verification*. Ph.D. thesis, Cambridge University.

J. Loof, R. Schlüter, and H. Ney (2010). "Discriminative Adaptation for Log-Linear Acoustic Models." In *Proceedings of Interspeech*. pp. 1648–1651.

M. Layton and M.J.F. Gales (2004). "Maximum Margin Training of Generative Kernels." Tech. Rep. CUED/F-INFENG/TR.484, Department of Engineering, University of Cambridge.

W. Macherey, L. Haferkamp, R. Schlüter, and H. Ney (2005). "Investigations on Error Minimizing Training Criteria for Discriminative Training in Automatic Speech Recognition." In *Proceedings of Interspeech*. vol. 2005, pp. 2133–2136.

M. Mahajan, A. Gunawardana, and A. Acero (2006). "Training Algorithms for Hidden Conditional Random Fields." In *Proceedings of ICASSP*. pp. 273–276.

L. Mangu, E. Brill, and A. Stolcke (1999). "Finding Consensus among Words: Lattice-based Word Error Minimization." In *Proceedings of Eurospeech*.

A. McCallum and D. Freitag (2000). "Maximum Entropy Markov Models for Information Extraction and Segmentation." In *Proceedings of ICML*. Morgan Kaufmann, pp. 591–598.

A. McCallum, K. Nigam, *et al.* (1998). "A Comparison of Event Models for Naive Bayes Text Classification." In *AAAI Workshop on Learning for Text Categorization*. vol. 752, pp. 41–48.

E. McDermott, T. J. Hazen, J. Le Roux, A. Nakamura, and S. Katagiri (2007). "Discrimiantive Training for Large-Vocabulary Speech Recognition Using Minimum Classification Error." *IEEE Transactions on Audio, Speech, and Language Processing* 15 (1), pp. 203–223.

T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur (2010). "Recurrent Neural Network based Language Model." In *Proceedings of Interspeech*. pp. 1045–1048.

T. Minka (2005). "Discriminative Models, Not Discriminative Training." Tech. Rep. MSR-TR-2005-144, Microsoft Research.

M. Mohri, F. Pereira, and M. Riley (2002). "Weighted Finite-State Transducers in Speech Recognition." *Computer Speech and Language* 16, pp. 69–88.

G. L. Moore (2001). *Adaptive Statistical Class-based Language Modelling*. Ph.D. thesis, Cambridge University.

P. J. Moreno (1996). *Speech Recognition in Noisy Environments*. Ph.D. thesis, Carnegie Mellon University.

J. Morris and E. Fosler-Lussier (2008). "Conditional Random Fields for Integrating Local Discriminative Classifiers." *IEEE Transactions on Audio, Speech, and Language Processing* 16 (3), pp. 617–628.

N. Cristianini and J. Shawe-Taylor (2000). *Support Vector Machines and other Kernel-based Learning Methods*. Cambridge University Press.

K. Na, B. Jeon, D.-I. Chang, S.-I. Chae, and S. Ann (1995). "Discriminative Training of Hidden Markov Models using Overall Risk Criterion and Reduced Gradient Method." In *Proceedings of the Eurospeech*. pp. 97–100.

A. Y. Ng and M. I. Jordan (2001). "On Discriminative VS. Generative Classifiers: A Comparison of Logistic Regression and Naive Bayes." In T. G. Dietterich, S. Becker, and Z. Ghahramani (eds.), *Advances in Neural Information Processing Systems*. MIT Press, Cambridge, MA, USA.

P. Nguyen and G. Zweig (2010). "Extensions to the SCARF Framework." Tech. Rep. MSR-TR-2010-129, Microsoft Research.

Y. Normandin (1991). *Hidden Markov Models, Maximum Mutual Information Estimation, and the Speech Recognition Problem*. Ph.D. thesis, McGill University.

F. J. Och and H. Ney (2002). "Discriminative Training and Maximum Entropy Models for Statistical Machine Translation." In *Proceeding of the fortieth annual meeting of the Association for Computational Linguistics*. Philadelphia, USA, pp. 295–302.

J. J. Odell (1995). *The Use of Context in Large Vocabulary Speech Recognition*. Ph.D. thesis, Cambridge University.

J. J. Odell, V. Valtchev, P. C. Woodland, and S. J. Young (1994). "A One Pass Decoder Design for Large Vocabulary Recognition." In *Proceedings of the workshop on Human Language Technology*. Association for Computational Linguistics, pp. 405–410.

S. Ortmanns, H. Ney, and X. Aubert (1997). "A Word Graph Algorithm for Large Vocabulary Continuous Speech Recognition." *Computer Speech and Language* 11 (1), pp. 43–72.

M. Ostendorff, V. V. Digalakis, and O. A. Kimball (1996). "From HMMs to Segment Models: A Unified View of Stochastic Modelling for Speech Recognition." *IEEE Transactions on Speech and Audio Processing* 4, pp. 360–378.

D. B. Paul and J. M. Baker (1992). "The Design for the Wall Street Journal-based CSR Corpus." In *Proceedings of the workshop on Speech and Natural Language*. pp. 357–362.

D. Pearce and H.-G. Hirsch (2000). "The Aurora Experimental Framework for The Performance Evaluation of Speech Recognition Systems Under Noisy Conditions." In *Proceedings of ICSLP*. Beijing, China, pp. 29–32.

F. Peng and A. McCallum (2006). "Information Extraction from Research Papers Using Conditional Random Fields." *Information Processing and Management* 42 (4), pp. 963–979.

J. Platt (1999). "Fast Training of Support Vector Machines using Sequential Minimal Optimization." In B. Schölkopf, C. J. C. Burges, and A. J. Smola (eds.), *Advances in Kernel Methods — Support Vector Learning*. MIT Press, Cambridge, MA, pp. 185–208.

J. Platt, N. Christianini, and J. Shawe-Taylor (2000). "Large Margin DAGs for Multiclass Classification." In S. A. Solla, T. K. Leen, and K.-R. Müller (eds.), *Advances in Neural Information Processing Systems*. MIT Press, pp. 547–553.

D. Povey (2003). *Discriminative Training for Large Vocabulary Speech Recognition*. Ph.D. thesis, Cambridge University.

D. Povey, D. Kanevsky, B. Kingsbury, B. Ramabhadran, G. Saon, and K. Visweswariah (2008). "Boosted MMI for Model and Feature-Space Discriminative Training." In *Proceedings of ICASSP*. pp. 4057–4060.

A. Quattoni, M. Collins, and T. Darrell (2004). "Conditional Random Fields for Object Recognition." In *Advances in Neural Information Processing Systems*. MIT Press, pp. 1097–1104.

A. Quattoni, S. Wang, L.-P. Morency, M. Collins, T. Darrell, and M. Csail (2007). "Hidden-State Conditional Random Fields." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29 (10), pp. 1848–1852.

R. C. van Dalen, A. Ragni, and M. J. F. Gales (2013). "Efficient Decoding with Generative Score-Spaces Using the Expectation Semiring." In *ICASSP*. pp. 7619–7623.

L. R. Rabiner (1989). "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition." In *Proceedings of the IEEE*. vol. 77, pp. 257–286.

A. Ragni (2013). *Discriminative models for speech recognition*. Ph.D. thesis, Cambridge University.

A. Ragni and M. J. F. Gales (2011*a*). "Derivative Kernels for Noise Robust ASR." In *Proceedings of ASRU*. Hawaii, USA, pp. 119–124.

A. Ragni and M. J. F. Gales (2011*b*). "Structured Discriminative Models for Noise Robust Continuous Speech Recognition." In *Proceedings of ICASSP*. Prague, Czech Republic, pp. 4788–4791.

B. Roark (2009). "A Survey of Discriminative Language Modeling Approaches for Large Vocabulary Continuous Speech Recognition." In J. Keshet and S. Bengio (eds.), *Automatic Speech and Speaker Recognition: Large Margin and Kernel Methods*, John Wiley & Sons, Ltd, Chichester, UK, chap. 8, pp. 117–138.

T. Roos, H. Wettig, P. Grünwald, P. Myllymäki, and H. Tirri (2005). "On Discriminative Bayesian Network Classifiers and Logistic Regression." *Machine Learning* 59 (3), pp. 267–296.

R. Rosenfeld (1994). *Adaptive Statistical Language Modeling: A Maximum Entropy Approach*. Ph.D. thesis, Carnegie Mellon University.

J. Salomon, S. King, and M. Osborne (2002). "Framewise Phone Classification Using Support Vector Machines." In *Proceedings of Interspeech*. pp. 2645–2648.

G. Saon and D. Povey (2008). "Penalty Function Maximization for Large Margin HMM Training." In *Proceedings of the Interspeech*. pp. 920–923.

S. Sarawagi and W. Cohen (2005). "Semi-Markov Conditional Random Fields For Information Extraction." In L. K. Saul, Y. Weiss, and L. Bottou (eds.), *Advances in Neural Information Processing Systems*. MIT Press, Cambridge, MA, pp. 1185–1192.

R. Schlüter, W. Macherey, B. Müller, and H. Ney (2001). "Comparison of Discriminative Training Criteria and Optimization Methods for Speech Recognition." *Speech Communication* 34 (3), pp. 287–310.

F. Seide, G. Li, and D. Yu (2011). "Conversational Speech Transcription Using Context-Dependent Deep Neural Networks." In *Proceedings of Interspeech*. pp. 437–440.

F. Sha and F. Pereira (2003). "Shallow Parsing with Conditional Random Fields." In *Human Language Technology*. pp. 213–220.

F. Sha and L. K. Saul (2007). "Large Margin Hidden Markov Models for Automatic Speech Recognition." In B. Schölkopf, J. Platt, and T. Hofmann (eds.), *Advances in Neural Information Processing Systems*. MIT Press, pp. 1249–1256.

C. E. Shannon (1948). "A Mathematical Theory of Communication." *Bell System Technical Journal* 27, pp. 379–423.

J. Shawe-Taylor and N. Cristianini (2004). *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA.

P. Shivaswamy and T. Jebara (2009). "Relative Margin Machines." In *Advances in Neural Information Processing Systems 21*, MIT Press, pp. 1481–1488.

Y. Singer and N. Srebro (2007). "Pegasos: Primal Estimated Sub-Gradient Solver for SVM." In *Proceedings of ICML*. pp. 807–814.

D. A. Smith and J. Eisner (2006). "Minimum Risk Annealing for Training Log-Linear Models." In *Proceeding of the joint conference of the International Committee on Computational Linguistics and the Association for Computational Linguistics*. Sydney, Australia, pp. 787–794.

N. D. Smith (2003). *Using Augmented Statistical Models and Score Spaces for Classification*. Ph.D. thesis, University of Cambridge.

N. D. Smith and M. J. F. Gales (2002*a*). "Speech Recognition using SVMs." In *Advances in Neural Information Processing Systems*. MIT Press, pp. 1197–1204.

N. D. Smith and M. J. F. Gales (2002*b*). "Using SVMs to Classify Variable Length Speech Patterns." Tech. Rep. CUED/F-INFENG/TR.412, Department of Engineering, University of Cambridge.

A. J. Smola (2000). *Advances in Large Margin Classifiers*. the MIT Press.

P. Sollich (2002). "Bayesian Methods for Support Vector Machines: Evidence and Predictive Class Probabilities." *Journal of Machine Learning* 46 (1), pp. 21–52.

H. Somers (1992). *An Introduction to Machine Translation*. Academic Press, London.

Y.-H. Sung, C. Boulis, and D. Jurafsky (2008). "Maximum Conditional Likelihood Linear Regression and Maximum A Posteriori for Hidden Conditional Raddom Fields Speaker Adaptation." In *Proceedings of ICASSP*. Las Vegas, USA, pp. 4293–4296.

Y.-H. Sung, C. Boulis, C. Manning, and D. Jurafsky (2007). "Regularization, Adaptation, and Non-Independent Features Improve Hidden Conditional Random Fields for Phone Classification." In *Proceedings of ASRU*. pp. 347–352.

Y.-H. Sung and D. Jurafsky (2009). "Hidden Conditional Random Fields for Phone Recognition." In *Proceedings of ASRU*. IEEE, pp. 107–112.

C. Sutton and A. McCallum (2006). "An Introduction to Conditional Random Fields for Relational Learning." In L. Getoor and B. Taskar (eds.), *Introduction to Statistical Relational Learning*, MIT Press.

K. Tanabe (2001). "Penalized Logistic Regression Machines: New Methods for Statistical Prediction." *Proceedings of IBIS, Tokyo* pp. 71–76.

B. Taskar (2005). *Learning Structured Prediction Models: A Large Margin Approach*. Ph.D. thesis, Stanford University, CA, USA.

B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin (2005). "Learning Structured Prediction Models: A Large Margin Approach." In *Proceedings of ICML*. ACM, pp. 896–903.

H. Thompson (1990). "Best-First Enumeration of Paths Through a Lattice — An Active Chart Parsing Solution." *Computer Speech and Language* 4 (3), pp. 263–274.

L. Tóth, A. Kocsor, and J. Csirik (2005). "On Naive Bayes in Speech Recognition." *Journal of Applied Mathematics and Computer Science* 15 (2), p. 287.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun (2005). "Large Margin Methods for Structured and Interdependent Output Variables." *Journal of Machine Learning Research* 6, pp. 1453–1484.

L. F. Uebel and P. C. Woodland (1999). "An Investigation into Vocal Tract Length Normalisation." In *Eurospeech*.

V. Valtchev, J. J. Odell, P. C. Woodland, and S. J. Young (1997). "MMIE Training of Large Vocabulary Recognition Systems." *Speech Communication* 22, pp. 303–314.

R. C. van Dalen, J. Yang, M. J. F. Gales, A. Ragni, and S.-X. Zhang (2012). "Generative Kernels and Score-Spaces for Classification of Speech: Progress Report." Tech. Rep. CUED/F-INFENG/TR676, Cambridge University. Available from: http://mi.eng.cam.ac.uk/∼mjfg/Kernel.

V. N. Vapnik (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag, New York.

V. Venkataramani and W. J. Byrne (2005). "Lattice Segmentation and Support Vector Machines for Large Vocabulary Continuous Speech Recognition." In *Proceedings of ICASSP*. pp. 817–820.

V. Venkataramani, S. Chakrabartty, and W. J. Byrne (2003). "Support Vector Machines for Segmental Minimum Bayes Risk Decoding of Continuous Speech." In *Proceedings of ASRU*. pp. 13–18.

O. Viikki and K. Laurila (1998). "Cepstral Domain Segmental Feature Vector Normalization for Noise Robust Speech Recognition." *Speech Communication* 25, pp. 133–147.

A. J. Viterbi (1982). "Error Bounds for Convolutional Codes and Asymptotically Optimum Decoding Algorithm." *IEEE Transactions on Information Theory* 13, pp. 260–269.

V. Vural and J. G. Dy (2004). "A Hierarchical Method for Multi-Class Support Vector Machines." In *Proceedings of ICML*.

S. Watanabe, T. Hori, and A. Nakamura (2010). "Large Vocabulary Continuous Speech Recognition using WFST-based Linear Classifier for Structured Data." In *Proceedings of Interspeech*. pp. 346–349.

J. Weston, B. Schölkopf, and O. Bousquet (2005). "Joint Kernel Maps." In *IWANN*. pp. 176–191.

K. Weston and C. Watkins (1999). "Support Vector Machines for Multi-Class Pattern Recognition." In *Proceedings of European Symposium on Artificial Neural Networks*. vol. 4, pp. 219–224.

S. Wiesler, M. Nußbaum-Thom, G. Heigold, R. Schlüter, and H. Ney (2009). "Investigations on Features for Log-Linear Acoustic Models in Continuous Speech Recognition." In *Proceedings of ASRU*. IEEE, Merano, Italy, pp. 52–57.

S. Wiesler, A. Richard, Y. Kubo, R. Schlüter, and H. Ney (2011). "Feature Selection for Log-Linear Acoustic Models." In *Proceedings of ICASSP*. pp. 5324–5327.

C. Williams and M. Seeger (2001). "Using the Nyström Method to Speed Up Kernel Machines." In *Advances in Neural Information Processing Systems*. MIT Press, pp. 682–688.

P. C. Woodland and D. Povey (2002). "Large Scale Discriminative Training of Hidden Markov Models in Speech Recognition." *Computer Speech and Language* 16 (1), pp. 25–48.

J. Ye, R. J. Povinelli, and M. T. Johnson (2002). "Phoneme Classification Using Naive Bayes Classifier in Reconstructed Phase Space." In *Digital Signal Processing Workshop*. IEEE, pp. 37–40.

S. J. Young (1995). "Large Vocabulary Continuous Speech Recognition: A Review." In *Proceedings of ASUR*. IEEE, Snowbird, Utah, pp. 3–28.

S. J. Young, G. Evermann, M. J. F. Gales, T. Hain, D. Kershaw, X. Liu, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. C. Woodland (2006). *The HTK Book (for HTK Version 3.4)*. University of Cambridge, `http://htk.eng.cam.ac.uk`.

S. J. Young, G. Evermann, M. J. F. Gales, T. Hain, D. Kershaw, X. Liu, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. C. Woodland (2009). *The HTK Book (for HTK Version 3.4.1)*. University of Cambridge, `http://htk.eng.cam.ac.uk`.

S. J. Young, J. J. Odell, and P. C. Woodland (1994). "Tree-based state tying for high accuracy acoustic modelling." In *Proceedings ARPA Workshop on Human Language Technology*. pp. 307–312.

C.-N. Yu and T. Joachims (2008). "Training Structural Svms with Kernels Using Sampled Cuts." In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge Discovery and Data Mining*. ACM, pp. 794–802.

C.-N. Yu and T. Joachims (2009). "Learning Structural SVMs with Latent Variables." In *Proceedings of ICML*. ACM, pp. 1169–1176.

A. Yuille, A. Rangarajan, and A. L. Yuille (2002). "The Concave-Convex Procedure (CCCP)." In *Advances in Neural Information Processing Systems*. MIT Press, pp. 915–936.

J. Zhang, R. Jin, Y. Yang, and A. G. Hauptmann (2003). "Modified Logistic Regression: An Approximation to SVM and Its Applications in Large-Scale Text Categorization." In *ICML*. pp. 888–895.

S.-X. Zhang and M. J. F. Gales (2010). "Structured Log Linear Models for Noise Robust Speech Recognition." Tech. Rep. CUED/F-INFENG/TR658, Cambridge University. Available from: http://mi.eng.cam.ac.uk/∼sxz20.

S.-X. Zhang and M. J. F. Gales (2011*a*). "Extending Noise Robust Structured Support Vector Machines to Larger Vocabulary Tasks." In *Proceedings of ASRU*. Hawaii, USA.

S.-X. Zhang and M. J. F. Gales (2011*b*). "Structured Support Vector Machines for Noise Robust Continuous Speech Recognition." In *Proceedings of Interspeech*. Florence, Italy, pp. 989–992.

S.-X. Zhang and M. J. F. Gales (2013*a*). "Kernelized Log Linear Models for Continuous Speech Recognition." In *Proceedings of ICASSP*. Vancouver, Canada, pp. 6950–6954.

S.-X. Zhang and M. J. F. Gales (2013*b*). "Structured SVMs for Automatic Speech Recognition." *IEEE Transactions Audio, Speech and Language Processing* 21 (3), pp. 544–555.

S.-X. Zhang and M. W. Mak (2011). "Optimized Discriminative Kernel for SVM Scoring and its Application to Speaker Verification." *IEEE Transactions on Neural Networks* 22 (2), pp. 173–185.

S.-X. Zhang, A. Ragni, and M. J. F. Gales (2010). "Structured Log Linear Models for Noise Robust Speech Recognition." *Signal Processing Letters, IEEE* 17, pp. 945–948.

J. Zheng and A. Stolcke (2005). "Improved Discriminative Training using Phone Lattices." In *Proceedings of Interspeech*. Lisbon, Portugal, pp. 2125–2128.

K. Zhu, H. Wang, H. Bai, J. Li, Z. Qiu, H. Cui, and E. Y. Chang (2007). "Parallelizing Support Vector Machines on Distributed Computers." In *Advances in Neural Information Processing Systems*. pp. 257–264.

Q. Zhu, B. Y. Chen, N. Morgan, and A. Stolcke (2004). "On using MLP Features in LVCSR." In *Proceedings of Interspeech*. pp. 921–924.

Q. Zhu, B. Y. Chen, N. Morgan, and A. Stolcke (2005). "Tandem Connectionist Feature Extraction for Conversational Speech Recognition." In *Machine Learning for Multimodal Interaction*. Springer, pp. 223–231.

G. Zweig and P. Nguyen (2009). "A Segmental CRF Approach to Large Vocabulary Continuous Speech Recognition." In *Proceedings of ASRU*. pp. 152–157.

G. Zweig, P. Nguyen, D. V. Compernolle, K. Demuynck, L. Atlas, P. Clark, G. Sell, M. Wang, F. Sha, H. Hermansky, D. Karakos, A. Jansen, S. Thomas, G. S. V. S. Sivaram, S. Bowman, and J. Kao (2011). "Speech Recognition with Segmental Conditional Random Fields: A Summary of the JHU CLSP 2010 Summer Workshop." In *Proceedings of ICASSP*. Prague, Czech Republic, pp. 5044–5047.