Discriminative Training for Large Vocabulary Speech Recognition

Article · January 2003

1 author:

Daniel Povey
Johns Hopkins University
127 PUBLICATIONS **6,886** CITATIONS

SEE PROFILE

Discriminative Training for Large Vocabulary Speech Recognition

Daniel Povey

Peterhouse

July, 2004

Dissertation submitted to the University of Cambridge for the degree of Doctor of Philosophy July 2004

Declaration

This thesis is the result of my own work and includes nothing which is the outcome of work done in collaboration, except where stated (see Chapter 1). It has not been submitted in whole or in part for a degree at any other university.

The length of this thesis including footnotes and appendices is approximately 55,000 words.

Acknowledgements

Thanks to the Schiff foundation, which provided funding for my degree, and to Phil Woodland for his advice and patience. Some of the experiments reported here, particularly those involving lattice generation and language models used for discriminative training, are based on experiments originally performed by Phil Woodland; these were published for instance in [Woodland & Povey, 2002]. Thanks are also due to various members of the speech group for their input, involvement with work on discriminative training, help with the system architecture and interesting conversations, including, in no particular order, Mark Gales, Luis Uebel, Nathan Smith, Thomas Hain, Gunnar Evermann, Andrew Liu, Matt Stuttle, Do Yeong Kim, K.K. Chin, Ricky Chan, and Lan Wang. Also thanks to Patrick Gosling for top-notch system administration (not many sysadmins can debug the kernel). The use of HTK software was also essential to this research.

Summary

This thesis investigates the use of discriminative criteria for training HMM parameters for speech recognition, in particular the Maximum Mutual Information (MMI) criterion and a new criterion called Minimum Phone Error (MPE).

Investigations are conducted into the practical issues relating to the use of MMI for speech recognition, and an implementation is described which gives good improvements in Word Error Rate for state-of-the-art systems on large vocabulary tasks such as Switchboard, Broadcast News and American Business News. Features of this implementation include the use of lattices to represent alternative transcriptions of the training data; acoustic likelihood scaling to take account of less likely alternative sentences; unigram language models; and a particular way of setting the learning rate in the Extended Baum-Welch update formulae. Implemented in this way, MMI training gives improvements wherever there is a sufficiently large ratio of training data to Gaussians in the HMM set.

The concept of weak-sense auxiliary functions is introduced as a tool to help in function maximisation problems. A weak-sense auxiliary function is a function which has the same gradient as a function to be maximised, around some local point. This is useful for optimisation in cases where what is here termed a strong-sense auxiliary function (as used in Maximum Likelihood updates) cannot be found or does not give an efficient update rule. Weak-sense auxiliary functions can be considered a generalisation of gradient descent. The concept of weak-sense auxiliary functions is used to give a derivation for the Extended Baum-Welch update formulae which are use for MMI criterion optimisation.

A discriminative criterion is introduced called Minimum Phone Error, which is a smoothed measure of phone transcription error. Methods are described for the optimisation of the MPE objective function and experimental results are given on a number of different corpora. A technique called I-smoothing is introduced which improves generalisation with MMI and MPE training. I-smoothing can be viewed as a form of MAP estimation with a prior distribution centered around the ML parameter estimates. It is essential in order for MPE to give improvements over MMI.

Extensive experiments on a number of different large and medium-vocabulary corpora show that MPE reliably gives better results than MMI. Based on experiments reported here, discriminative training is predicted to give increasing improvement relative to MLE as more training data becomes available.

Contents

Notation

μ_{jm}	Mean vector for Gaussian m of state j of a HMM
c_{jm}	Gaussian mixture weight of Gaussian m of state j
Σ_{jm}	Covariance matrix of Gaussian m of state j
σ^2_{jm}	Variance of Gaussian m of state j (unidimensional case)
$b_j(\cdot)$	Output p.d.f of state j
$\mathbf{o}_r(t)$	Observation vector t from speech file r
$\mathcal{O}_r$	Speech file r (the sequence of observation vectors $\mathbf{o}_r(1) \dots \mathbf{o}_r(T_r)$
r	Index of speech file
R	Number of speech files
t	Time (measured in speech frames)
$\mathcal{N}(\mathbf{x}\|\mu, \Sigma)$	Gaussian probability density function
λ	All the parameters of an HMM or set of HMMs, i.e. the Gaussian means, variances and weight and transition values
$\mathcal{M}$	An HMM model topology, describing phone HMM topologies and how phone HMMs are combined into models of sentences.
$\gamma_{jm}(t)$	Occupation probability at time t for Gaussian m of state j
γ_{jm}	Summed occupation probability for Gaussian m of state j
$\theta_{jm}(\mathcal{O})$	Sum of data weighted by probability for Gaussian m of state j
$\theta_{jm}(\mathcal{O}^2)$	Sum of squared weighted data for Gaussian m of state j
$\mathcal{F}(\lambda)$	An objective function used in training HMMs (and other functions of high-dimensional parameters)
$\mathcal{G}(\lambda, \lambda')$	An auxiliary function used in training, where the starting HMM value is λ'
$Q(t, x, y\|\mu, \sigma)$	Likelihood of t points of data with sum x and sum-squared y, given Gaussian with mean and variance μ and σ
q	An arc in a time-marked phone lattice
$X = x(1) \dots x(T)$	A sequence of HMM states; each value $x(1) \dots x(T)$ identifies a state
$\mathcal{A}$	Curly letters denote a vector of unknown dimension
$\mathcal{F}(\mathcal{A})$	Functions with curly letters denote a function of a vector of unknown dimension

Chapter 1

Introduction

This thesis investigates the discriminative training of Hidden Markov Model (HMM) parameters. HMMs are introduced in detail in Chapter 2, but broadly speaking they are a statistical model of speech production. Their "parameters," which consist of means and variances of Gaussian distributions and various probabilities included in the model, are generally estimated as statistical distribution, so for instance the mean would be set to the mean of the appropriate observed data. Discriminative training of HMM parameters is the optimisation of the HMM as a classifier, so that the means and variances and other parameters are adjusted to as to improve the HMM's classification performance on the training data.

Most previous work on discriminative training for speech recognition has focused on two training criteria: the first of these, Maximum Mutual Information (MMI; also known as Maximum Conditional Likelihood) is the posterior probability assigned by the HMM to the correct transcription of the training data, which is to be maximised. The other training criterion, Minimum Classification Error (MCE), is a smoothed approximation to the sentence classification error, which is to be minimised.

The work presented in this thesis can be divided into two parts: firstly, work on the practical aspects of the use of MMI for large vocabulary speech recognition. This describes a specific implementation of MMI which uses lattices to speed up HMM training, and experimentally investigates various aspects of the training procedure. This work builds on previous work done in Cambridge [Valtchev et al., 1996], and was done in collaboration with Phil Woodland; much of the work on MMI has been published previously, e.g. in [Woodland & Povey, 2000].

The second part of the work relates to a new discriminative criterion, Minimum Phone Error (MPE). (Note however that this is equivalent to a criterion devised independently elsewhere, known as the Overall Risk Criterion [Na et al., 1995, Kaiser et al., 2000]). Work presented here shows that in combination with an appropriate technique to smooth parameters estimates for which there is insuf-

ficient training data, MPE can consistently improve on MMI and give useful improvements for large vocabulary recognition tasks.

Chapter 2 introduces the speech-to-text problem, introduces HMMs and gives a general overview of the history of automatic speech recognition.

Chapter 3 discusses the concept of a "discriminative objective function" and introduces the objective functions used in MMI, MCE and MPE.

In discriminative training, the maximisation of the discriminative objective function is a difficult problem. This is addressed in Chapter 4, which develops some theory relating to objective function maximisation.

Chapter 5 concerns the implementation of lattice-based MMI. It presents experimental results concerning various aspects of the implementation of MMI training: for example, the language models used in training, the size of the lattices, the speed of optimisation and the extent of scaling of log likelihoods.

Chapter 6 gives theory and experiments regarding various (mostly slight) alterations to the EB update equations.

Chapter 7 introduces the Minimum Phone Error (MPE) criterion, and describes the techniques used for its optimisation.

Chapter 8 experimentally investigates various practical issues relating to MPE training, and gives results comparing MPE with ML and MMI training under various conditions and for various large vocabulary corpora.

Chapter 9 contains a conclusion and summary.

Chapter 2

Introduction to speech recognition

2.1 The Speech Recognition Problem

The speech recognition problem, as it has traditionally been defined (and as defined in this thesis), is the task of taking an utterance containing a certain length of speech data (call this utterance $\mathcal{O}$) and transforming it into a text string $\mathcal{F}(\mathcal{O})$ which is as close as possible to the transcript that a careful human would generate.

The task is to find a function $\mathcal{F}(\cdot)$ which does the job as well as possible. The way in which the success of $\mathcal{F}(\cdot)$ is generally evaluated is to calculate $\mathcal{F}(\mathcal{O})$ for a number of speech files $\mathcal{O}$ comprising a *test set* of speech data, and calculate the *Error Rate* of the output (Figure 2.1).

The $\mathcal{F}(\cdot)$ we generally want is the one that has the lowest Error Rate.

2.1.1 Speech Recognition using Training Data

It was recognised early on that the function $\mathcal{F}(\cdot)$ cannot simply be crafted by hand. There are many reasons for this:

- It is too difficult to find a function $\mathcal{F}(\cdot)$ that would work well.
- One would have to try many functional forms for $\mathcal{F}(\cdot)$ and try them out on a limited amount of test data; this would lead to learning of the test data and possibly poor generalisation to other similar data.
- It would not be straightforward, having obtained a suitable $\mathcal{F}(\cdot)$ for a given language or task, to transfer it to another language or task.

Error Rate
The *Error Rate*, often expressed as a percentage figure, is defined as:

$$\text{Error Rate} = \frac{100(\#\text{substituted} + \#\text{deleted} + \#\text{inserted})}{\#\text{ref words}}$$

where #ref words is the number of words in the correct (reference) transcription. Substituted, deleted and inserted words are defined with respect to an alignment between the reference and hypothesis transcriptions. This alignment is chosen so as to minimise the error.
The Error Rate of current speech recognition systems ranges from around 50% for the most demanding tasks such as transcribing recordings of meetings, to under 1% for transcribing digit strings recorded under good conditions.

Figure 2.1: Error Rate

To avoid having to construct the final speech recogniser entirely by hand, modern speech recognition systems require *training data*. Modern speech recognition systems rely on statistical analysis of the training data to transcribe unseen speech. The training data consists of R speech files $\mathcal{O}_r$ for $r = 1 \ldots R$, each with a corresponding human-generated transcription s_r. The problem now becomes one of finding an appropriate function $\mathcal{F}(\mathcal{O}; \mathcal{O}_1, \mathcal{O}_2 \ldots \mathcal{O}_R)$ which will evaluate an unseen sentence $\mathcal{O}$ based on information gathered from the training data. The function $\mathcal{F}(\cdot)$ will then include a specification of how exactly the training files $\mathcal{O}_1, \mathcal{O}_2 \ldots \mathcal{O}_R$ are used to create a speech recognizer. Viewed in this light, it is clear that the key to the function $\mathcal{F}(\cdot)$ is how it generalises from the training examples to unseen data. Some generalisations will clearly be more appropriate than others, and the task of a speech recognition researcher is to find a function $\mathcal{F}(\cdot)$ (i.e. a speech recognition system) that makes the appropriate generalisations. To give an example: a speech recognition system that views the speech signal as a binary representation of a large number and tries to analyse it in terms of its prime factors will not get very far. The observation that different generalisations are needed for different circumstances is supported by the "No Free Lunch Theorem" [Wolpert, 1994] which says that no inductive problem can be solved by a single algorithm which will always be better than other algorithms regardless of the examples supplied.

2.2 Statistical Speech Recognition

For good speech recognition performance, it is essential to take into account that some sequences of words are more likely to be heard than others. Statistical models are used to estimate the prior probability that any given sentence s will be uttered. This part of the task is called *language modeling*. The aim of lan-

guage modeling is to find the probability of a sequence of words $w_1 w_2 \ldots w_N$. The most common language models are so-called n-gram models, in which the probability of the k'th word is made conditional on the previous n-1 words, so that $p(w_1 \ldots w_N) = \prod_{k=1}^{N} p(w_k | w_{k-1} \ldots w_{k-n+1})$. These probabilties can be estimated from observed counts in large collections of transcribed speech or other text. n-gram models tend to be used with values of n such as 3 or 4, so $p(w_k)$ depends on the preceding 2 or 3 words. A problem that arises is that of unseen examples. This is solved by *backoff*, in which the $n-$gram probabilities for a particular context are interpolated with the $(n-1)-$gram probabilities for a reduced context to give more robust probability estimates; and *discounting*, in which some of the probability mass assigned to observed words is assigned to unseen words to avoid giving them zero probability.

Most successful speech recognition systems use statistically-based approaches not only for language modeling but also for *acoustic modeling,* to evaluate how well a given speech file $\mathcal{O}$ matches a proposed sentence s. Acoustic and language information is combined through Bayes' Rule (Figure 2.2). This combines the information from two sources: a statistical model of word sequence probability (which sentences are more likely) and a model of speech production (how the properties of the speech signal relate to what is being said).

As applied to speech recognition, Bayes' Rule lets us write:

$$P(s|\mathcal{O}) = \frac{P(s)p(\mathcal{O}|s)}{p(\mathcal{O})}, \tag{2.1}$$

i.e, the posterior probability $P(s|\mathcal{O})$ of sentence s being the true transcription of the utterance $\mathcal{O}$, equals the likelihood $p(\mathcal{O}|s)$ of $\mathcal{O}$ being observed if s was spoken, multiplied by the prior probability $P(s)$ of s and divided by a normalising term. The posterior probability $P(s|\mathcal{O})$ gives information about which sentences are most likely to have been said. The s for which $P(s|\mathcal{O})$ is largest is the one most likely to have been said, so if the speech recogniser outputs this s it will have the greatest chance of being correct (assuming the model used is correct).

The *acoustic model* $P(\mathcal{O}|s)$ is generally based on a Hidden Markov Model (HMM); HMMs are explained further in Section 2.3.

Note that in practice the language model term $P(s)$ has to be scaled by a power to give $P(s)^{\alpha}$, for say $\alpha = 15$. This is to prevent the language model probabilities being overwhelmed by the acoustic likelihoods, which tend to be highly correlated between frames and overestimate the system's confidence in the acoustic information.

2.2.1 Pre-processing of speech

Until now, the symbol $\mathcal{O}$ has been used to represent an utterance without specifying its format. The raw data which a speech recogniser receives as input is the

Bayes' Rule
Bayes' Rule states that: $P(A|B) = \frac{P(A)P(B|A)}{P(B)}$, where $P(X|Y)$ is the probability of event X given that event Y is known to have happened; and $P(X)$ is the *prior probability* of event X if nothing else is known. Events X and Y might represent, for instance, breaking one's leg today and being run down by a car today. If one is known to have happened, it will affect our estimate of the likelihood that the other has happened.
Bayes' rule can be written $P(A|B) = \frac{P(A)p(B|A)}{p(B)}$ if B is a continuous variable (but A is discrete), with $p(X)$ being the value of the *probability density function* (p.d.f.) from which those events are drawn, at the point X.

Figure 2.2: Bayes' Rule

speech waveform itself, which consists of sampled voltages taken from a microphone at a rate of perhaps 16kHz. This data is *preprocessed* before being used by speech recognisers. This means taking short segments of, say, 20ms of speech, and using varioussignal processing techniques to turn it into a vector of typically 20-60 dimensions which describes the characteristics of the speech "frame". The vector typically represents the short-time spectral envelope of the speech signal. This is done for frames of audio data taken at intervals of 10ms or so and the resulting *feature vectors* are denoted $o(1) \ldots o(T)$. The feature vectors taken together are referred to with the symbol $\mathcal{O}$. See Figure 2.3 for a summary of some of the techniques used in feature extraction.

2.2.2 Speech units and Dictionaries

It is necessary in large vocabulary speech recognition to be able to relate written words to their pronunciation in *phones*. This makes it possible to synthesise words for which there no examples in the acoustic training data, which will always tend to be necessary even for very large amounts of acoustic training data. Phones are the basic sounds of a spoken language, and there generally thirty to forty phones for a typical language. A phone is written for instance as /k/ for the sound corresponding to the letter k, to distinguish it from the letter. Many phones are denoted using special symbols only found in the Phonetic Alphabet.

Dictionaries (or *lexicons*) are used in speech recognition to tell the system how words are pronounced by giving their sequence of phones.

2.3 HMMs in speech recognition

The central component of a modern speech recognition system is the Hidden Markov Model (HMM). A HMM is a statistical model for the production of

Speech Preprocessing

The speech preprocessor converts the approximately 20ms long *frames* of raw acoustic waveform data into *feature vectors* of perhaps 40 or more dimensions. The fundamental criterion for a good preprocessor is that it should make a speech recogniser work well, but in general that means that the features (components of the vector) should be:

- Independent– i.e. not very correlated with each other.
- Salient– i.e., should tell us something relevant to speech recognition. Generally this means it should contain information about the spectral envelope.

Well known preprocessing algorithms include those based on:

- *Mel Frequency Cepstral coefficients* (MFCC)– the *cepstrum* is what results from taking the Fourier transform of a log energy spectrum; the spectrum is first warped according to the Mel frequency scale [Davis & Mermelstein, 1980].
- Cepstral coefficients derived from a linear prediction-derived Mel frequency warped spectrum to take into account human frequency sensitivity (MF-PLP) [Hermansky, 1990].
- RASTA PLP, which is like MF-PLP but with techniques to normalise the effect of varying channel properties [Hermansky, Morgan, Bayya & Kohn, 1991].

Techniques used to supplement that initial preprocessing include:

- Delta and Delta-delta coefficients: these are the first and second differentials of the feature vector w.r.t time, often calculated as differences.

  Deltas (Δ) are the differential between successive frames; Delta-Deltas ($\Delta\Delta$) are the difference between successive Deltas. These are appended to the original vector of coefficients. (Deltas and Delta-deltas may actually be gradients estimated using a window of e.g. 5 frames rather than just a difference of successive frames).
- Linear Discriminant Analysis: a matrix transformation of the feature vector intended to maximise separation between different classes (e.g. phone classes) [Haeb-Umbach & Ney, 1992, Saon et al., 2000].
- Mean and Variance Normalisation: Normalising the data in a given feature dimension to have the same mean and variance for every file, to cancel some of the speaker and channel variation.
- Vocal Tract Length Normalisation: a technique to warp frequencies to cancel some speaker variations, especially the difference between male and female speakers [Lee & Rose, 1996, Welling et al., 1999].

Figure 2.3: Speech Preprocessing

sequences of symbols (or sequences of vectors of continuous numbers, in this case). A HMM has states $j = 1 \ldots N$, and transition probabilities a_{ij} between each pair of states. It also has a starting state and an ending state (although these are sometimes represented as starting and ending probabilities. Some states (*emitting* states) have output distributions $b_j(\mathbf{x})$ over the set of output symbols (which might for instance be a continuous vector $\mathbf{x}$, or a set of discrete symbols). It is most easily explained by means of a picture (Figure 2.4). This HMM has five states, and three *emitting* states. On the first time instant the HMM is in the state on the far left. At each step, the HMM pictured will output a real number drawn randomly from the distribution of the state it is in (if it is in an emitting state), and will then randomly change state according to the likelihoods indicated on the diagram. The sequence will finish when the exit state is reached, which means the sequence of output values has terminated. HMMs used for speech recognition would typically output continous vectors. This particular topology of HMM (i.e. three states with left-to-right transitions) is one commonly used to model phones; as an example, this HMM has been labeled /m/. Figure 2.5 shows two HMMs concatenated to form a word HMM for "me". Alternatively some small systems use a separate HMM for each word. Word HMMs would then be concatenated to form a HMM for a sequence of words. In *context-dependent* systems there may be a number of HMMs for a single phone, with different HMMs for different *phone contexts*, i.e. the one or two phones on either side.

Early HMMs used discrete output symbols, which were obtained by so-called "Vector Quantisation"; this refers to a process of clustering the feature vectors in a subset of the training data and then quantising each vector according to which cluster center it was closest to. At the moment essentially all work on speech recognition uses continuous vector-valued output symbols, which are generally modeled with a mixture-of-Gaussians distribution.

2.3.1 Definition of a HMM

For simplicity, let us assume that all states have an output distribution. In practice, phone or word HMMs have "non-emitting" states (with no output distribution) at the beginning and end, but HMMs with such states can be regarded as a shorthand for equivalent HMMs with all emitting states.

What is needed to define a HMM are a number of states $j = 1 \ldots J$, a *transition matrix* containing probabilities a_{ij} of transitions from state i to state j, and *output distributions* $b_j(\mathbf{x})$ for each state j, which will in general be Gaussian distributions or mixtures of Gaussians (i.e. sums of Gaussian p.d.f.'s). Constraints on the HMM include:

- Transition probabilities from a state sum to 1: $\sum_j a_{ij} = 1$.
- Output distributions $b_j(\mathbf{x})$ must integrate to 1.

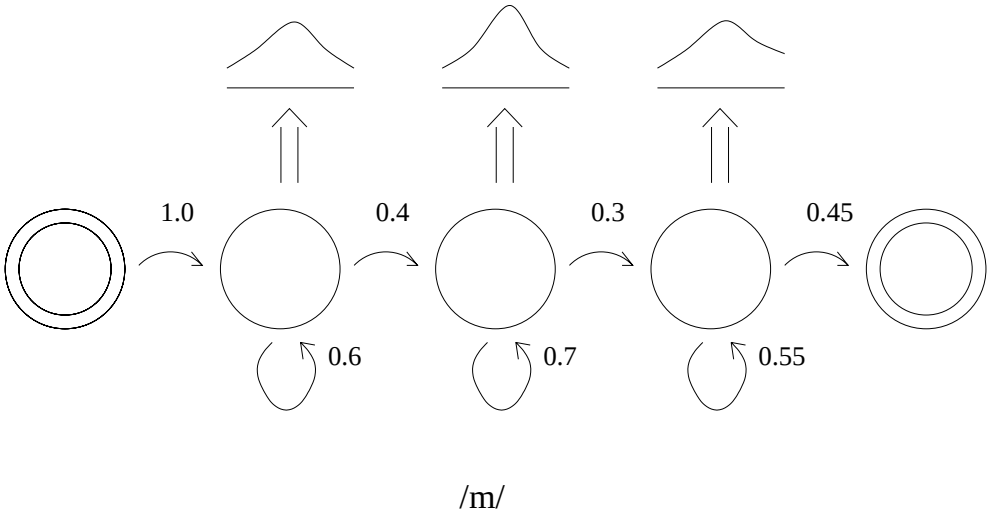


Figure 2.4: Example: a HMM for /m/

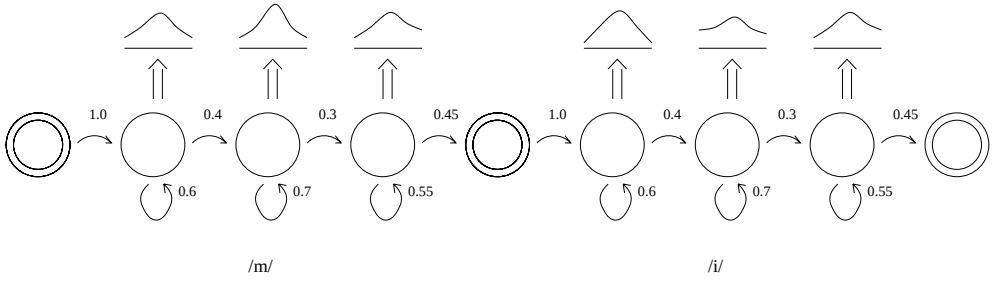


Figure 2.5: Example: a HMM for “me”

A particular HMM will be described by the symbols $\mathcal{M}$ and λ: $\mathcal{M}$ (the model) is the topology of the HMM (in a simple case this would be the number of states, number of Gaussians per state, a specification of which transitions between states are allowed) and λ is the model parameters– transition values a_{ij} (where allowed) and the parameters of the distributions $b_j(\cdot)$. In the case of continuous speech recognition, the symbol $\mathcal{M}$ is generally also used to indicate the way individual HMMs for phones are concatenated to form an HMM for whole sentences or sets of sentences.

The parameters λ refer to the transition and output-probability distributions for the phone HMMs.

It is sometimes necessary to refer to the concept of a *state sequence*, which is an ordered sequence of HMM states; let us use X to denote a state sequence of length T, comprising a series of states $x(1) \ldots x(T)$.

An HMM will have start and end states (possibly more than one of each) and associated probabilities, and these need to be specified. Since there are generally only one start and end state with start/end probabilities of 1, it is easiest to make this a constraint on state sequences X rather than complicate equations by introducing extra terms corresponding to the start and end probabilties.

Mixture-of-Gaussian HMMs

In most current speech recognition systems, the output probability density function $b_j(\mathbf{x})$ is a mixture of Gaussians $b_j(\mathbf{x}) = \sum_{m=1}^{M_j} c_{jm}\mathcal{N}(\mathbf{x}; \mu_{jm}, \sigma_{jm})$, where M is the number of Gaussians in state j, c_{jm} is the weight for Gaussian m of state j, and μ_{jm} and σ_{jm} are the means and variances of the Gaussians. The weights c_{jm} are constrained to sum to 1 for each state.

Mixtures of diagonal Gaussians are favored because they are robust to limited training data but can also take advantage of larger amount of training data (the number of Gaussians can be increased).

2.3.2 Likelihoods with HMMs

One of the things one can do with a HMM is to calculate the likelihood of a certain sequence of data given the HMM— i.e, the likelihood that the HMM would have generated that particular sequence.

The likelihood, which can be written as $p(\mathcal{O}|\mathcal{M}, \lambda)$, of a speech file $\mathcal{O}$ consisting of samples $\mathbf{o}(1) \ldots \mathbf{o}(T)$, being produced from a HMM $\mathcal{M}$ with parameters λ, is:

$$p(\mathcal{O}|\mathcal{M}, \lambda) = \sum_X \prod_{t=1}^{T} b_{x(t)}(\mathbf{o}(t)) a_{x(t)x(t+1)}, \tag{2.2}$$

i.e, the sum over all sequences X of the probability of the speech data $\mathcal{O}$ given that sequence. If $\sum_X$ is replaced by $\max_X$, the expression becomes the so-called *Viterbi* likelihood, which is the likelihood considering only the best path. This is usually quite close to the likelihood obtained by summing over all possible sequences, because the sum tends to be dominated by the largest term.

2.3.3 Context dependence and state tying

Modern continuous speech recognition systems model phones rather than words, which makes it possible to synthesise models for words with no examples in the training data. The most obvious way to model continuous speech on a phone basis is to use a single HMM per phone. A problem that arises is that a particular phone can be realised in very different ways when surrounded by different phones (this is known as *co-articulation*), or can even be omitted altogether (*elision*). One way to solve this problem is to use *context dependent* HMMs: for example, to have a different HMM for each phone with each combination of phones immediately to the left and right. This would be called a *triphone* system. The problem that then arises is that many of these phones-in-context will have few or no examples in the training data. The problem of contexts with few examples can be solved by a straightforward clustering algorithm, so that similar phones-in-context are pooled. But this still does not solve the problem of contexts that are not seen at all in the training data. This can be solved by *tree-based* clustering, in which the set of all contexts of a particular phone is successively divided [Young et al., 1994]. Tree-based clustering is applied to all systems used here.

2.3.4 Recognition and training with HMMs

There are two main situations in which it is necessary to calculate HMM likelihoods: during training of the HMM and recognition of speech data.

Training

When training HMMs for speech recognition we need a set of training files $\mathcal{O}_r$ ($r = 1 \ldots R$), each with a known sentence s_r corresponding to the utterance in that file. From these sentences, together with a lexicon (a dictionary, which will contain phonetic pronunciations of words) and a set of phone HMMs, we can construct a sentence model $\mathcal{M}_{s_r}$ from individual phone HMMs.

These models together with the speech data are then used to accumulate statistics which can be used to derive a better estimate of the HMM parameters by increasing the likelihood of the training data given the HMM. The process involves an implicit calculation of the likelihood of all possible paths X through the HMM.

This algorithm, known as the Baum-Welch algorithm, will be referred to again in Section 4.2.3. See for example [Rabiner, 1993] for a more complete description of this algorithm. Briefly, the training process, known as Estimation-Maximisation (E-M) operates in two phases. The first phase, the Estimation phase, involves accumulating certain statistics from the training data. This involves estimating the current distribution over the hidden state-sequence variable X (hence "estimation"). Although an explicit sum over the state-sequences would be computationallly impossible, it is sufficient to know more limited distributions such as the distribution over the currently active state for each frame of training data. These can be calculated using an efficient algorithm known as the forward-backward algorithm, which involves calculating probabilities of partial state sequences forwards and backwards through the training data.

The forward-backward algorithm calculates the posterior probability $\gamma_j(t)$ of being in state j at time t by calculating the likelihood of the data given partial state sequences. These partial likelihood expressions are $\alpha_j(t)$ for state sequences up to and including state j at time t and $\beta_j(t)$ for state sequences going from state j at time t to the end of the utterance. The forward likelihoods $\alpha_j(t)$ are calculated according to the recursion $\alpha_j(t) = \sum_{k=1}^{J} \alpha_k(t-1)t_{kj}b_j(t)$, with the base case at time 1 as $\alpha_j(1) = b_j(1)$ for the start state $j = 1$ and zero otherwise. The backward likelihoods are calculated as $\beta_j(t) = \sum_{k=1}^{J} \beta_k(t+1)t_{jk}b_k(t+1)$, with the base case at time T as $\beta_J(T) = 1$ for the end state J and zero otherwise. The probability of the data $p(\mathcal{O})$ is given as $\alpha_J(T)$, and the probability $\gamma_j(t)$ of occupying state j at time t is $\frac{\alpha_j(t)\beta_j(t)}{p(\mathcal{O})}$.

These occupation probabilities make it possible to gather sufficient statistics for a re-estimation formula that is guaranteed to converge to a local maximum. Defining γ_j as $\sum_{t=1}^{T} \gamma_j(t)$, $\theta_j(\mathcal{O})$ as $\sum_{t=1}^{T} \gamma_j(t)\mathcal{O}(t)$ and $\theta_j(\mathcal{O}^2)$ as $\sum_{t=1}^{T} \gamma_j(t)\mathcal{O}(t)^2$, then (giving the unidimensional case as a simple example) the Gaussian parameters can be re-estimated as $\mu_j = \frac{\theta_j(\mathcal{O})}{\gamma_j}$ and $\sigma_j^2 = \frac{\theta_j(\mathcal{O}^2)}{\gamma_j} - \mu_j^2$.

Recognition

Once HMMs for individual phones or phones-in-context have been trained, these can be combined with language model and pronunciation information (the lexicon) to create a single large HMM which is a general model of speech production. The language model dictates which words can follow which words (or sequences of words) and with what probability, and the lexicon provides a mapping from words to phone sequences. In practice many recognition systems do not actually construct this composite HMM all at once but construct parts of it as required.

Recognition proceeds by finding the most likely sequence of states through the composite HMM and finding the word sequence corresponding to this state-sequence. It is possible to find the best path through the HMM using the *viterbi*

algorithm. For each frame of speech data, this algorithm calculates the partial HMM likelihood for the best path up to and including each state, and stores for each state the preceding state in the best path. At the end of the utterance it is possible to trace back through these records of the best preceding state, to find the overall best state sequence. The time required for this algorithm is $O(tNs)$, where t is the length of the utterance, N is the number of states in the HMM and s is the average number of successor states of each state. It can be made faster by *pruning*, in which state sequences that do not look promising are discarded so that most of the states in the graph are not being considered at any given time.

N-best recognition

For some purposes it is necessary to calculate a number of the most likely word sequences corresponding to an utterance. This is possible using a so-called $N-$best version of the Viterbi algorithm. This algorithm will output the N most likely word sequences corresponding to an utterance. Instead of storing the best preceding state for each state on each time t, N so-called *tokens* are stored for each state; each token contains a record of a state, a likelihood and a preceding token. Any pair of tokens that represent the same word sequence are considered as competing with each other even if the state sequences differ; this enables the algorithm to find the N best word sequences.

Recognition with lattice output

$N-$best recognition is not be a very efficient way to find the best state sequences because alternative transcriptions of a few parts of an utterance can combine to produce a very large number of alternate sentences containing very few real alternatives. It is more efficient to encode the alternative outputs in a *lattice*, which is a graph encoding the various possible sentences in an efficient way. Lattice generation algorithms can be based on modifications of $N-$best viterbi recognition, or use other techniques.

2.4 History of speech recognition

The basic form of current speech recognition systems, which are based on HMMs with mixture-of-Gaussian output distributions, dates from the mid-1970s.

Some of the very early work on speech recognition was done in the former Soviet Union, which in the 1960s had an interest (premature, it turned out) in using computers for human language processing and translation. Only three years after the Fast Fourier Transform was made widely known, Vintsyuk published in 1968 his Dynamic Time Warping (DTW) algorithm [Vintsyuk, 1968]. A quite similar

algorithm was proposed independently in 1971 by Sakoe [Sakoe & Chiba, 1971]. These algorithms use a simple Dynamic Programming technique find a mapping between two sequences of observations that minimises a measure of distance between the two sequences– e.g, a test example and a word "template". An aspect of those algorithms which is still used today is the division of speech into short *frames* (e.g, 100 per second), and the use of signal processing techniques to extract the most salient properties of the local acoustic waveform into a "feature vector" describing the properties of the sound for that time frame.

Early speech recognition techniques attempted to match a segment of speech to a "template" word, possibly using a nonlinear approach to warp time when comparing the words, and used a distance measure between the two sets of feature vectors to measure how good a fit it was. The best matching word was proposed as the transcription of the waveform. A separate stage of processing was needed to find the word boundaries prior to template matching, because the DTW technique did not lend itself easily to modeling continuous sequences of words.

The use of the Hidden Markov Model (HMM) for speech recognition, was pioneered by researchers working with Fred Jelinek at IBM in the 1970s [Jelinek, 1997] and improved and popularised with the help of researchers at Bell Labs [Rabiner, 1993] and Carnegie Mellon University (CMU). The HMM made it possible to recognise large vocabulary speech by modeling speech sounds on a phone rather than whole-word basis and automating the process of finding the phone boundaries within words. A successful early speech recognition system which used HMMs was the Dragon system developed at CMU [Baker, 1975]; this was made more efficient in the later Harpy system [Lowerre, 1976] when "beam-search" was introduced. It took another ten years from the introduction of Dragon before research on Dynamic Programming algorithms was completely superseded by research on HMMs. Although HMMs initially used discrete output distributions (i.e, the feature vector was processed to produce one of a number of discrete "symbols", whose probabilities were estimated by the HMM), recent research has concentrated on HMMs with continuous output distributions using a mixture of Gaussians, as proposed in 1985 [Rabiner, Juang, Levinson & Sondhi, 1985].

Other improvements include the replacement of whole-word models with phone models and then context-dependent phone models [Schwartz et al., 1985]; Maximum A Posteriori (MAP) estimation as a means of adapting models trained on many speakers to a new speaker [Gauvain & Lee, 1994]; Maximum Likelihood Linear Regression (MLLR) as a means to do the same thing by transform matrices, needing less data [Leggetter & Woodland, 1995, Gales & Woodland, 1996]; techniques used to cluster phone models so as to generalise to unseen triphone contexts, e.g [Young et al., 1994]; vocal tract length normalisation for adaptation to male or female speakers, e.g [Lee & Rose, 1996, Welling et al., 1999]; Linear Discriminant analysis to obtain more useful or compact feature vectors, as proposed in [Haeb-Umbach & Ney, 1992], and improved versions of this using so-

called "heteroscedastic" methods [Kumar & Andreou, 1998, Saon et al., 2000].

Most systems currently used in large-vocabulary, multiple speaker speech recognition (e.g, those reported in the NIST 2001 workshop [NIST, 2001]), are standard mixture-of-Gaussian HMM systems relying mostly on the techniques mentioned above.

Apart from these mainstream techniques, other directions of research have included the investigation of neural nets (mainly as a front-end to HMM-based recognisers); other kinds of statistically-based modeling such as segmental models; and discriminative training, which is the subject of this thesis. These techniques have found use in small-vocabulary systems but have generally not proved very useful in reducing error rate on the larger-vocabulary, more difficult tasks. However, as will be shown in this thesis, discriminative training can be useful for large vocabulary tasks as well.

2.5 Discriminative training

A persistent strand in work in speech recognition over recent decades is discriminative training. As mentioned previously, discriminative training means the training of HMM parameters so as to optimise some measure of goodness-of-recognition of the training data. Two of the most widely known discriminative techniques are Maximum Mutual Information (MMI) [Bahl et al., 1986] and Minimum Classification Error (MCE) [Chou et al., 1993, Juang et al., 1997], although many more exist. Although many authors have demonstrated improvements on small-vocabulary tasks from discriminative training, consistent improvements in word error on large vocabulary tasks have been more elusive.

Probably the first published work to demonstrate the utility of MMI for a large vocabulary task was done in Cambridge [Valtchev et al., 1996]. The work described in this thesis extends that previous work and investigates in detail an altered implementation of MMI which gives more consistent gains on large vocabulary tasks. As in [Valtchev et al., 1996], lattices are used to encode competing hypotheses for recognition of the training data, but various changes and improvements are made to the training algorithm and many aspects of the training procedure are experimentally investigated. The other main contribution of this thesis is the presentation of a new discriminative training technique called Minimum Phone Error (MPE). This consistently gives better results than MMI (although by a small margin), and appears to be a promising technique for discriminative training.

Chapter 3

Discriminative objective functions

3.1 Introduction

HMMs are trained by optimising *objective functions*, otherwise known as *training criteria*. An objective function, which is to be either maximised or minimised depending on the particular objective function concerned, is a scalar function $\mathcal{F}(\lambda; \mathcal{O}_1 \ldots \mathcal{O}_R)$ of the parameters λ of the HMM set and the training data $\mathcal{O}_1 \ldots \mathcal{O}_R$.

Objective functions are useful because they express in a simple and compact form the essential aspects of a proposed HMM training technique, thus separating the function optimisation part of the system from the definition of the objective function itself. The function optimisation part of the system can then be judged by how much it increases the objective function. This division of the problem makes it easier to evaluate and improve the function optimisation, and easier to transfer the discriminative technique to new kinds of models and systems, than if the discriminative training were defined procedurally (as in the case in so-called "boosting" techniques, in which greater weight is given to misrecognised utterances).

Section 3.2 describes some of the best-known previously described discriminative objective functions. Section 3.3 introduces the new Minimum Phone Error (MPE) and Minimum Word Error (MWE) objective functions.

3.2 Previously proposed discriminative criteria

3.2.1 The MMI and MCE objective functions

The standard objective function used in Maximum Likelihood training can be written as follows:

$$\mathcal{F}_{\mathrm{MLE}}(\lambda) = \sum_{r=1}^{R} \log p_\lambda \left(\mathcal{O}_r | s_r\right) \tag{3.1}$$

where s_r is the correct transcription of the r'th speech file $\mathcal{O}_r$. This is the likelihood of the observations of training data given the correct-transcription HMM. Maximum Mutual Information (MMI) [Bahl et al., 1986] and Minimum Classification Error (MCE) [Chou et al., 1993, Juang et al., 1997] are probably the two most popular discriminative training criteria. The MMI objective function is as follows:

$$\mathcal{F}_{\mathrm{MMI}}(\lambda) = \sum_{r=1}^{R} \log \frac{p_\lambda \left(\mathcal{O}_r | s_r\right)^\kappa P(s_r)^\kappa}{\sum_s p_\lambda \left(\mathcal{O}_r | s\right)^\kappa P(s)^\kappa} \tag{3.2}$$

where $P(s)$ is the language model probability (including scales and word insertion penalties) for sentence s. The MMI criterion equals the posterior probability of the correct sentence s_r. Following [Schluter & Macherey, 1998], a probability scale κ is included since this is important if MMI training is to lead to good test-set performance.

The MCE objective function was originally defined for isolated word recognition, in which the utterance can be from one of a fixed number of classes $i = 1 \ldots M$. Class-conditional likelihoods $g_i(\mathcal{O}; \lambda)$ are defined as:

$$g_i(\mathcal{O}; \lambda) = \log p(\mathcal{O} | \mathcal{M}_i, \lambda), \tag{3.3}$$

where $\mathcal{M}_i$ is the HMM topology for the i'th class and λ represents the HMM parameters. A *misclassification measure* for each class is defined as follows:

$$d_i(\mathcal{O}) = -g_i(\mathcal{O}; \lambda) + \log \left[\frac{1}{M-1} \sum_{j, j \neq i} \exp g_j(\mathcal{O}; \lambda) \eta \right]^{1/\eta}, \tag{3.4}$$

which will tend to be positive if the system does not classify the utterance as being from class i, and negative if the utterance is classified as class i. The misclassification measures are then embedded in sigmoid functions:

$$l_i(\mathcal{O}) = \frac{1}{1 + \exp(-\gamma d_i(\mathcal{O}))}. \tag{3.5}$$

where $\eta > 0$ and $\gamma > 0$ are constants. The MCE objective function, which is to be minimised, is the sum of $l_i(\mathcal{O})$ over all the correct classes. The objective function is zero for each sentence that is correctly recognised, and one for each incorrect sentence; the transition between the two is "soft," and hence differentiable, controlled by the parameters γ and η. MCE is defined for the case where there are a fixed number of classes (as in isolated digit recognition, for instance); for continuous speech recognition, the classes can be defined as all possible word sequences, with the correct word sequence as the correct class. This makes it necessary to somehow enumerate a list of those word sequences that have a reasonable probability for each sentence of training data. This can be done using N-best lists [Chou et al., 1993], or more efficiently using lattices [Schluter & Macherey, 1998] as explained in the next section.

3.2.2 Implementations of MMI and MCE

Although both MMI and MCE training are popular, there are few direct comparisons between the two in the literature. It was found in [Reichl & Ruske, 1995] from experiments on a phone recognition task that MCE can outperform MMI; however, another author found for a continuous digit recognition task that MCE was only better for HMMs with relatively few Gaussians [Schluter, 2000].

Many successful implementations of MMI have been reported in the literature, but prior to the work described in this thesis there has been very little work on MMI for large vocabulary continuous speech recognition. The only prior example I am aware of is [Valtchev et al., 1996], although after the publication by myself and Phil Woodland [Povey & Woodland, 2001] of our success with MMI for large vocabulary speech recognition, other groups have reported success with MMI for large vocabulary tasks.

There appear to be no reports of large vocabulary implementations of MCE. It should be possible in principle to implement MCE training for LVCSR: in [Schluter & Macherey, 1998] an expression is given which unifies MMI and MCE into a single criterion suitable for use with lattices. This is:

$$\mathcal{F}(\lambda) = \sum_{r=1}^{R} f\left(\log \frac{p_\lambda^\kappa(\mathcal{O}_r|\mathcal{M}_{s_r})\,P(s_r)^\kappa}{p_\lambda^\kappa(\mathcal{O}_r|\mathcal{M}_{\mathrm{rec}_r})}\right), \tag{3.6}$$

where for MMI we set the function f to $f(x) = x$ and include all sentences in the composite model $\mathcal{M}_{\mathrm{rec}_r}$; and for MCE we set $f(x) = -\frac{1}{1+e^{\rho x}}$ and exclude the correct sentence from $\mathcal{M}_{\mathrm{rec}_r}$. κ is a scale on the language and acoustic probabilities, and might be set for instance to the inverse of the normal language model scale; s_r is the correct transcription of the r'th utterance. The equivalence with MCE is valid if $\kappa = \eta = \gamma$ and ignoring the factor $\frac{1}{M-1}$ in Equation (3.4). Note that language model terms such as $P(s_r)$ in Equation 3.6 are assumed to already be

scaled by the normal language model scale and to contain any insertion penalties used in recognition. $p_\lambda^\kappa(\mathcal{O}_r|\mathcal{M})$ indicates a likelihood calculated by scaling by κ all log sentence likelihoods before summing the likelihoods of sentences included in the model $\mathcal{M}$. MCE training can be implemented as a modification of the Extended Baum-Welch (EB) procedure for MMI training, by scaling the state occupation probabilities and sums of data accumulated from each training file by the value of $\partial f(x)/\partial x$, i.e. the differential of the sigmoid function, where x is the value of the MMI criterion for that file [Schluter & Macherey, 1998]. Experiments for continuous digit recognition showed that MCE only outperformed MMI for HMMs with a small number of Gaussians [Schluter, 2000].

Preliminary experiments performed by the author on the Resource Management corpus (not published) showed that excluding the correct sequence from the MMI denominator $\mathcal{M}_{\mathrm{rec}_r}$ (which is referred to in this thesis as $\mathcal{M}_{\mathrm{den}}$) degraded performance. The other change necessary to implement MCE (changing $f(x)$ from the identity function to a sigmoid) was not tried since the first change was not beneficial, and since the sigmoid introduces into the training algorithm an undesirable dependence on the way the data is segmented into training files. That is, if two recorded utterances are appended to form a single utterance this will affect the outcome of MCE training with that data. If the full set of training data were in a single file, MCE training implemented this way would be equivalent to MMI training. MCE training makes most sense when there is likely to be only a single error per file, e.g. in very small vocabulary or isolated-word recognition.

The problem with the MCE criterion for continuous speech recognition is that it is related to the sentence error rate whereas the sentences, which are generally defined to be equivalent to the files of training data, are just an arbitrary segmentation of the available data. What is of most interest is the Word Error Rate (WER), and it would seem to make sense to optimise this more directly.

3.2.3 Other discriminative criteria

Of the other discriminative criteria which have been proposed, one is Frame Discrimination [Kapadia, 1998, Povey & Woodland, 1999] which can be viewed as a modified form of MMI in which constraints on transitions between states are removed. Another is the technique of "Overall Risk Criterion Estimation" (ORCE) [Na et al., 1995] which applied to continuous speech recognition [Kaiser et al., 2000, Kaiser et al., 2002] is identical to the Minimum Word Error proposed here in Section 3.3.

The work on MWE described in this thesis was begun before [Kaiser et al., 2000] was published; MWE is identical to ORCE in principle but the work published here differs in some important aspects such as the use of lattices, MAP estimation of parameters (I-smoothing) and the way the smoothing constants in the EB update equations are set, as well as the emphasis on phone error rather than

word error. The Overall Risk Criterion, which minimises a measure of risk, may be particularly useful in specific tasks where the costs of different kinds of errors are known, for instance in speaker verification or speech recognition for control of particular devices. The methods described in this thesis are also relevant to discriminative training using the Overall Risk Criterion.

3.3 Minimum Phone Error (MPE)

A criterion was developed which seems to be more suitable for continuous speech recognition than MCE because it is more directly related to Word Error Rate which is the scoring criterion generally used in continuous speech recognition. The Minimum Phone Error (MPE) criterion is a smoothed approximation to the phone transcription accuracy measured on the output of a word recognition system given the training data. A related criterion, Minimum Word Error (MWE), is a similar approximation to the word transcription accuracy.

The objective function in MPE, which is to be maximised, is:

$$\mathcal{F}_{\mathrm{MPE}}(\lambda) = \sum_{r=1}^{R} \sum_{s} P_{\lambda}^{\kappa}(s|\mathcal{O}_r) A(s, s_r), \tag{3.7}$$

where λ represents the HMM parameters; $P_{\lambda}^{\kappa}(s|\mathcal{O}_r)$ is defined as the scaled posterior sentence probability of the sentence s being the correct one (given the model)

$$P_{\lambda}^{\kappa}(s|\mathcal{O}_r) = \frac{p_{\lambda}(\mathcal{O}_r|s)^{\kappa} P(s)^{\kappa}}{\sum_u p_{\lambda}(\mathcal{O}_r|u)^{\kappa} P(u)^{\kappa}} \tag{3.8}$$

where κ is a scaling factor typically less than one, $\mathcal{O}_r$ is the speech data for the r'th training sentence; and $A(s, s_r)$ is the raw phone transcription accuracy of the sentence s given the reference sentence s_r, which equals the number of reference phones minus the number of errors.

The criterion is an average over all possible sentences s (weighted by their likelihood given the recognition model) of the raw phone accuracy for that file. In terms of individual sentence-conditional likelihoods, expanding the terms $P_{\lambda}^{\kappa}(s|\mathcal{O}_r)$, the objective function can be expressed as:

$$\mathcal{F}_{\mathrm{MPE}}(\lambda) = \sum_{r=1}^{R} \frac{\sum_s p_{\lambda}(\mathcal{O}_r|s)^{\kappa} P(s)^{\kappa} A(s, s_r)}{\sum_u p_{\lambda}(\mathcal{O}_r|u)^{\kappa} P(u)^{\kappa}}. \tag{3.9}$$

It is important to emphasise that while the MPE criterion maximises a measure of phone transcription accuracy, this is done in the context of a word recognition system. So, given a number of competing word-level transcriptions of a sentence, the MPE criterion will try to make the more accurate transcriptions more likely; and it will measure accuracy based on how many phones are correct.

3.3.1 Scaling likelihoods

All sentence and language model log likelihoods in the MPE objective function (Equation (3.7)) are scaled by the constant κ. The function of this is to make the less likely sentences contribute to the objective function and make the MPE objective function more smoothly differentiable. The power κ typically equals the inverse of the language model scale used in recognition, or a value in the range $\frac{1}{10}$ to $\frac{1}{20}$ if no statistical language model is used.

Note that the language model probabilities $P(s)$ appearing in Equations (3.7) and (3.9) are assumed already to contain any likelihood scales and insertion penalties used in recognition. (This notation for scaling likelihoods is used following [Schluter & Macherey, 1998]). As the scale κ becomes large, the MPE criterion for each file approaches the value of $A(s, s_r)$ for the most likely transcription s of that file. As κ becomes smaller the criterion increasingly takes into account the accuracy of less likely sentences. This improves the ability of the trained system to generalise to unseen data, by taking into account more alternative hypotheses.

3.3.2 Minimum Word Error (MWE)

The MWE objective function is the same as MPE, except that the function $A(s, s_r)$ (which for MPE equals the number of reference phones minus phone errors) is calculated on a word rather than phone basis. MWE is a more effective criterion than MPE for maximising the training set word accuracy but consistently gives slightly poorer results on the test set (see Section 8.15).

It seems likely that as the amount of training data relative to HMM set size approaches infinity, MWE will give better results than MPE because it is a closer approximation to the word error. So far it has not been possible to verify this; for all ratios of training data to HMM set size tested, MPE seems to consistently give better test set results than MWE.

3.4 Comparison between objective functions: example

Imagine a simple task in which the only two sentences are "a" and "b", and "a" is the correct transcription of the current training file.

Defining $a = p_\lambda(\mathcal{O}|\text{"a"})P(\text{"a"})$ as the acoustic likelihood multiplied by the language model probability of the sentence "a", and b as the same for "b", the contribution of a particular training file to the four objective functions considered here is given as follows:

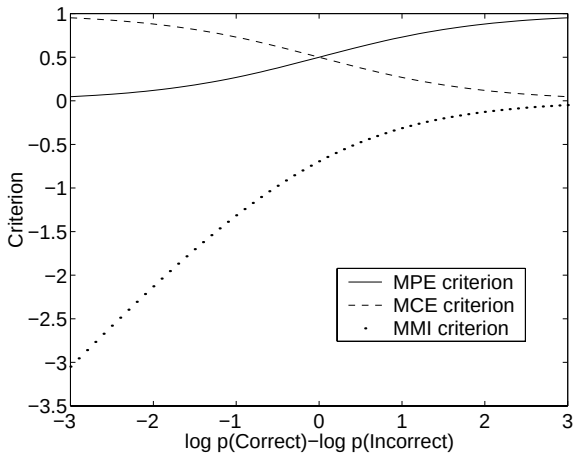


Figure 3.1: MPE vs. MMI criteria for one correct and one incorrect hypothesis.

$$\begin{aligned} \text{ML}: &\quad \log(a) \\ \text{MMI}: &\quad \log(\tfrac{a^\kappa}{a^\kappa+b^\kappa}) \\ \text{MCE}: &\quad \tfrac{b^\gamma}{a^\gamma+b^\gamma} \\ \text{MPE}: &\quad \tfrac{a^\kappa\times 1+b^\kappa\times 0}{a^\kappa+b^\kappa} \end{aligned}$$

Note that in this case the contribution to the MMI objective function is equal to the log of the contribution to the MPE objective function. This is so only because the raw phone accuracy of "a" happens to equal 1 and that of "b" happens to equal 0. The difference the log makes is that as the file becomes increasingly wrongly recognised (to the left of the graph in Figure 3.1) the MMI criterion continues to decrease while the MPE criterion approaches a minimum value. This means that words or files which have not been transcribed correctly during preparation of the training data, or are so poorly pronounced as to have no realistic chance of being correctly recognised, will not be given as much importance by MPE as by MMI.

In this particular example, if $\kappa = \gamma$ the MCE and MPE criteria are related by the equation: $\mathcal{F}_{\text{MPE}} = 1 - \mathcal{F}_{\text{MCE}}$, which makes them equivalent because MCE is minimised while MPE is maximised. Again, this equivalence relies on the choice of candidate phone sequences and is not general: for sequences of length greater than one phone, MPE and MCE become diffrent because MPE measures correctness on a phone rather than whole-sentence basis.

An important difference between MPE and the other criteria is that the weighting given by the MPE criterion to an incorrect hypothesised sentence depends on the

number of wrong phones is it, whereas the MMI and MCE criteria make a binary distinction based on whether the entire sentence is correct or not.

A clear advantage of MPE over MCE is that in the limit of very long sentences MCE would be of no use at all, since the posterior probability of the correct sentence would approach zero and the MCE criterion would have zero gradient w.r.t the log-likelihood of the correct HMM. This problem can be solved by using MCE at the level of words rather than sentences [Bauer, 2001], but only at the expense of deciding on an arbitrary word-level segmentation. In contrast, MPE is essentially invariant to the splitting or concatenation of the training data files.

Chapter 4

Function maximisation

This chapter gives a theoretical basis for the use of the Extended Baum-Welch (EB) equations in MMI optimisation, based on the concepts of strong-sense and weak-sense auxiliary functions which are introduced here.

Section 4.1 gives a brief introduction to the optimisation of discriminative objective functions. Section 4.2 introduces the concept of strong-sense and weak-sense auxiliary functions and shows how they relate to the Baum-Welch and Extended Baum-Welch equations respectively. Section 4.3 explains how prior information can be integrated into auxiliary functions for MAP updates, and derives the technique of I-smoothing which is a MAP update of the discriminatively trained parameters. Section 4.5 reviews the original justification of the EB update equations which were presented in [Gopalakrishnan et al., 1989].

4.1 Introduction

Maximum Likelihood parameter estimation for mixture-of-Gaussian HMMs is considered a solved problem; the standard approach is given in [Juang, 1985]. The estimation procedure is based on the Expectation-Maximisation (EM) technique [Dempster et al., 1977]. This is an iterative procedure in which each iteration is a two-step process, and which is guaranteed to converge to a local maximum of the data likelihood. The first step involves accumulating statistics which depend on the current estimated distribution of a hidden variable (the state-sequence variable in this case). The second step maximises a so-called "auxiliary function". This auxiliary function is a function which if its value is increased the likelihood of the data given the HMM is bound to increase also. The auxiliary function is an easier function to directly maximise than the true likelihood function; that is why it is useful.

For discriminative criteria, the optimisation problem is much more difficult because an update rule which is guaranteed to increase the objective function and

which also has fast convergence appears to be impossible to derive. Attempts to prove such update rules (e.g [Gopalakrishnan et al., 1989]) tend to only be successful in the limit as the change in parameters becomes extremely small for the discrete case, or infinitely small in the continuous case [Normandin & Morgera, 1991].

To solve the problem of optimising discriminative criteria, a number of gradient-based solutions have been proposed: for instance, Generalised Probabilistic Descent (GPD) [Juang & Katagiri, 1992] is a popular gradient-based technique. Gradient-based techniques are quite diverse; in the context of optimising the Frame Discrimination criterion, which is similar in principle to MMI, a number of gradient based techniques are compared in [Kapadia, 1998]. There is also the more "EM-like" technique of the Extended Baum-Welch (EB) formulae [Gopalakrishnan et al., 1989, Normandin & Morgera, 1991].

The work on discriminative training presented here has focused on the Extended Baum-Welch (EB) update equations. These have the advantage of being relatively simple to implement as they do not require statistics from more than one iteration of training. The EB update equations are also easy to combine with prior distributions over the HMM parameters.

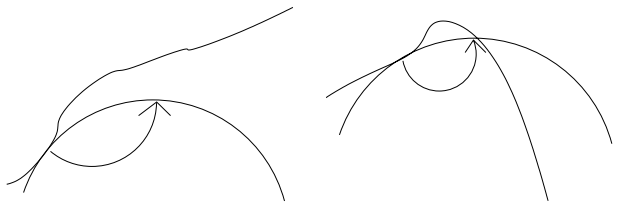
The EB and gradient-based approaches are almost equivalent given appropriate choice of smoothing constants and learning rates respectively [Schluter et al., 1997, Zheng et al., 2001], and the remaining differences between the two make little practical difference for Gaussian updates (e.g. see experiments reported in Chapter 6). One advantage of the EB approach is that the concept of auxiliary functions which is used to justify it can also lead to an effective and stable optimisation technique for the mixture weights and transition probabilities (Section 4.4), whereas with gradient descent it can be difficult to find the right tradeoff between speed and stability.

4.2 Strong-sense and weak-sense auxiliary functions

If a function $\mathcal{F}(\lambda)$ is to be maximised, then $\mathcal{G}(\lambda, \lambda')$ will be said to be a *strong-sense auxiliary function* for $\mathcal{F}(\lambda)$ *around* λ', iff

$$\mathcal{G}(\lambda, \lambda') - \mathcal{G}(\lambda', \lambda') \leq \mathcal{F}(\lambda) - \mathcal{F}(\lambda'), \tag{4.1}$$

where $\mathcal{G}(\lambda, \lambda')$ is a smooth function of λ. A strong-sense auxiliary function is the kind of auxiliary function used in Expectation-Maximisation (EM) [Dempster et al., 1977]. The idea is illustrated in Figure 4.1(a). A maximum w.r.t. λ of the function $\mathcal{G}(\lambda, \lambda')$ is found, indicated by the arrow. If this increases $\mathcal{G}$ (the lower line), it will also increase $\mathcal{F}$; if $\mathcal{G}$ is at a local maximum then $\mathcal{F}$ is also at a local maximum. These conditions follow from Eq. (4.1), and imply that repeated maximisation



(a) (b)

Figure 4.1: Use of (a) strong-sense and (b) weak-sense auxiliary functions for function optimisation

of the auxiliary function, forming a new auxiliary function around the current parameters on each iteration, is guaranteed to reach a local maximum of $\mathcal{F}(\lambda)$.

A *weak-sense auxiliary function* for $\mathcal{F}(\lambda)$ *around* λ' is a smooth function $\mathcal{G}(\lambda, \lambda')$ such that

$$\left.\frac{\partial}{\partial\lambda}\mathcal{G}(\lambda,\lambda')\right|_{\lambda=\lambda'} = \left.\frac{\partial}{\partial\lambda}\mathcal{F}(\lambda)\right|_{\lambda=\lambda'}. \tag{4.2}$$

The idea is shown in Figure 4.1(b). The gradients of the two functions are the same around the point $\lambda = \lambda'$. Maximising the function $\mathcal{G}(\lambda, \lambda')$ w.r.t. λ does not now guarantee an increase in $\mathcal{F}(\lambda)$. However if there is no change in λ after maximisation on a particular iteration, this implies that we have reached a local maximum of $\mathcal{F}(\lambda)$ (the gradient is zero at that point). If the update converges it will be to a local maximum of $\mathcal{F}(\lambda)$.

The weak-sense auxiliary function condition of Equation (4.2) can be considered a minimum condition for an auxiliary function used for optimisation. In addition the function should be chosen so as to ensure good convergence. Weak-sense auxiliary functions are not bound to be concave as in Figure 4.1(b), but an auxiliary function which is not concave is less likely to lead to good convergence.

Weak-sense auxiliary functions are useful when optimising functions containing some terms that can be optimised by strong-sense auxiliary functions but others that cannot. Weak-sense auxiliary functions make it possible to modify procedures based on strong-sense auxiliary functions (e.g. Expectation-Maximisation) rather than switching to entirely different techniques based on gradient descent.

4.2.1 Smoothing functions

A useful extra definition is that a *smoothing function* around λ' is a smooth function of λ, $\mathcal{G}(\lambda, \lambda')$, such that

$$\mathcal{G}(\lambda, \lambda') \leq \mathcal{G}(\lambda', \lambda') \tag{4.3}$$

for all λ. It has its maximum at the initial point λ' and thus if a smoothing function around λ' is added to an objective function, the resulting function is a strong-sense auxiliary function for that objective function around λ'.

A smoothing function can be added to a weak-sense auxiliary function to improve convergence without affecting the local gradient.

4.2.2 Examples of strong-sense and weak-sense auxiliary functions

Strong-sense auxiliary function

The classic example of what is here called a strong-sense auxiliary function, is the one used in HMM estimation. This is covered in Section 4.2.3. This section deals with a simple but interesting example. Suppose we have a function $f(x)$ which is a sum of cosine wave components with known fixed amplitudes K_n,

$$f(x) = \sum_{n=1}^{N} K_n \cos(2\pi n x), \tag{4.4}$$

and suppose that starting from some initial value of x we need to find a local maximum in the function. Strong-sense auxiliary functions provide an easy iterative solution to this problem that is guaranteed to converge. The aim of using strong-sense auxiliary functions is to turn the function into a simpler form whose maximum can be found analytically. There is an additive property in auxiliary functions which means each of the N terms in $f(x)$ can be tackled separately.

It is easy to find a strong-sense auxiliary function for each individual term $K_n \cos(2\pi n x)$. The approach will be to use strong-sense auxiliary function of the form $g(x) = A_n x + B_n x^2$. The idea is shown in Figure 4.2. Suppose the current value of x is x'. Let us define y as the difference $x - x'$ and use the more convenient shifted form $g(y) = A_n y + B_n y^2$ for the auxiliary function. The gradient of this around the point $x = x'$ ($y = 0$) equals A_n, and auxiliary functions always have the same gradient as the objective function around the local point, so

$$\begin{aligned} A_n &= \frac{\partial}{\partial x} K_n \cos(2\pi n x)\bigg|_{x=x'} \\ &= -2\pi n K_n \sin(2\pi n x'). \end{aligned}$$

The second differential of the auxiliary function w.r.t. x is a constant equal to $2B_n$. If this is set to less than or equal to the lowest ever value of the second differential w.r.t. x of the function $K_n \cos(2\pi nx)$, the two functions will never cross and the inequality of Equation (4.1) will hold. This can be visualised by looking at Figure 4.2. The second differential of the cosine function is $-(2\pi n)^2 K_n \cos(2\pi nx)$, and the minimum value of this is $-(2\pi n)^2|K_n|$, so an acceptable value for B_n is:

$$B_n = -0.5(2\pi n)^2|K_n|. \tag{4.5}$$

Figure 4.2 shows the auxiliary function $A_n y + B_n y^2$ (where $y = x - x'$) for an example sine wave. The auxiliary function has been shifted up for clarity and the current value $x' = 1.2$ is marked. Note that for some positions on the cosine wave it would be possible to set B to a value much closer to zero and $A_n y + B_n y^2$ would still be a strong-sense auxiliary function for the cosine wave around x'. This would lead to faster convergence towards a local maximum of $f(x)$ but would increase the complexity of the update equation so the option has not been pursued. It would also be technically possible to set B to a very large negative constant value, but this would lead to the optimisation having very slow convergence.

Summing the values of A_n and B_n over the values of n to get an auxiliary function for the full objective function of Equation (4.4), and using the fact that the maximum of the auxiliary function occurs at at $x = x' - \frac{A}{B}$ for an auxiliary function of this form, the updated value of x is:

$$x = x' - \frac{\sum_{n=1}^{N} -2\pi n K_n \sin(2\pi nx')}{\sum_{n=1}^{N} -0.5(2\pi n)^2|K_n|} \tag{4.6}$$

where the numerator of the fraction equals $\sum_{n=1}^{N} A_n$ and the denominator equals $\sum_{n=1}^{N} B_n$, assuming that the frequencies range from 1 to N. This should be applied iteratively, setting x' to the new value of x each time. For an example, suppose that $N = 5$, K_n are $3, -3, 3, -1$ and 2 respectively and the starting point is $x' = 0.6$. Figure 4.3 shows the objective function and the points visited by the optimisation (only three separate points are visible). The value of x has converged to within 5 decimal places after 5 iterations.

Weak-sense auxiliary function

Suppose there is a need to maximise a function of the form

$$f(x) = f_1(x) + (f_2(x))^2, \tag{4.7}$$

where $f_1(x)$ and $f_2(x)$ are sums of cosines as in Equation (4.4), with N cosine components and amplitudes J_n and K_n respectively for $n = 1 \ldots N$. In fact, the above approach using strong-sense auxiliary functions is applicable to this case

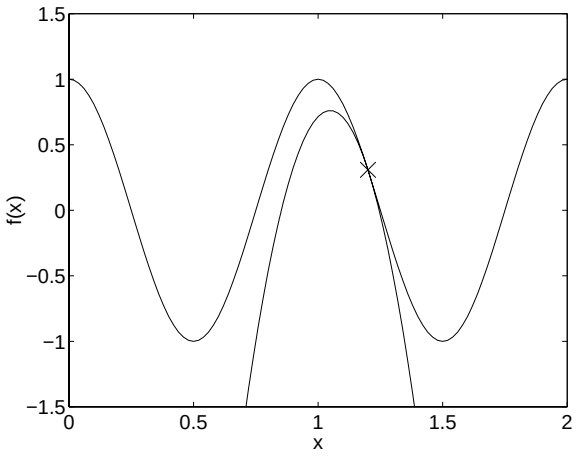


Figure 4.2: Strong-sense auxiliary function for a cosine wave

because the term $(f_2(x))^2$ can be expanded into a sum of cosines. But a different technique will be used in order to show the principle of weak-sense auxiliary functions.

Let the current value of x be x'. The following function of the variable x,

$$g(x) = f_1(x) + 2f_2(x)f_2(x'), \tag{4.8}$$

is a weak-sense auxiliary function for $f(x)$ around the point $x = x'$, because the differential w.r.t x where $x = x'$ equals $\frac{\partial f_1(x)}{\partial x}|_{x=x'} + 2\frac{\partial f_2(x)}{\partial x}|_{x=x'} f_2(x')$ for both the objective function and the auxiliary function. But this $g(x)$ is not an auxiliary function that can be maximised directly because it still contains cosines.

Strong-sense auxiliary functions as in Section 4.2.2 can be derived for the two terms in $g(x)$, the sum of which will be a strong-sense auxiliary function, say $h(x)$, for $g(x)$. Since $h(x)$ is a strong-sense auxiliary function for $g(x)$ around x' and $g(x)$ is a weak-sense auxiliary function for $f(x)$ around x', $h(x)$ is a weak-sense auxiliary function for $f(x)$ around x'.

The first term in $g(x)$ is a sum of cosines with coefficients J_n and the second a sum of cosines with coefficients $2f_2(x')K_n$. Applying the same techniques described in Section 4.2.2, strong-sense auxiliary functions can be derived for each of the two terms in $g(x)$. The sum of these two auxiliary functions will be called $h(x)$, and again using the shorthand $y = x - x'$, it is of the form $Ay + By^2$, where $A = \sum_{n=1}^{N} -2\pi n(J_n + f_2(x')K_n)\sin(2\pi n x')$ and $B = \sum_{n=1}^{N} -0.5(2\pi n)^2(|J_n| + |2f_2(x')K_n|)$. The maximum of $h(y)$ is $-\frac{A}{B}$; expressed in terms of x, the maximum is where $x = x' - \frac{A}{B}$, and this gives the update rule

$$x = x' - \frac{\sum_{n=1}^{N} -2\pi n(J_n + f_2(x')K_n)\sin(2\pi n x')}{\sum_{n=1}^{N} -0.5(2\pi n)^2(|J_n| + |2f_2(x')K_n|)}. \tag{4.9}$$

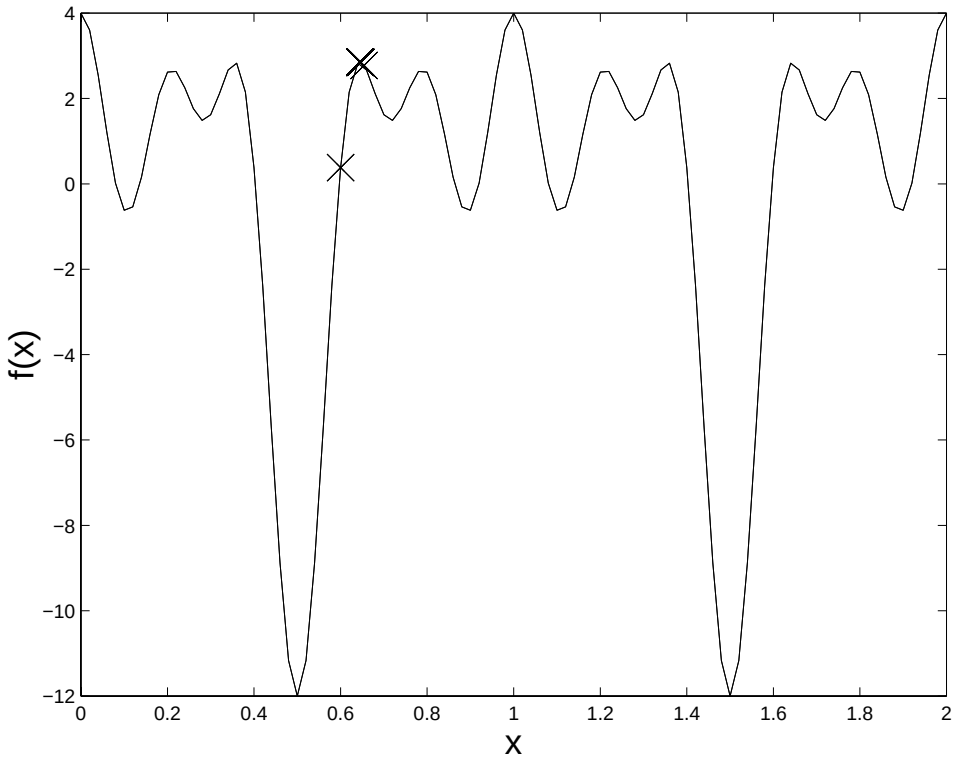


Figure 4.3: Strong-sense auxiliary function optimisation for a sum of cosines

Figure 4.4 shows the performance of the chosen weak-sense auxiliary function in maximising the objective function, using as an example $N = 5$, $J_n = 3, -3, 3, -1, 2$, $K_n = 1, -1, 2, 0.5, -0.5$. Optimisation is started separately from a range of starting points (circles) and run for ten iterations from each point (crosses). As can be seen, the optimisation leads to a local maximum in all cases.

It should be emphasised that weak-sense auxiliary functions have to be constructed with the help of some intuition about what is likely to be effective. Even a linear function of the parameters can be a valid weak-sense auxiliary function, but it would not give a useful update. An auxiliary function should be concave if it is to lead to finite updates. It should be clear from Figure 4.2 how essential it is that that auxiliary function have an appropriate second differential w.r.t. any parameter. A too-negative second differential will lead to very small updates; a second differential too close to zero will lead to very large changes in parameters which can cause the parameter values to diverge.

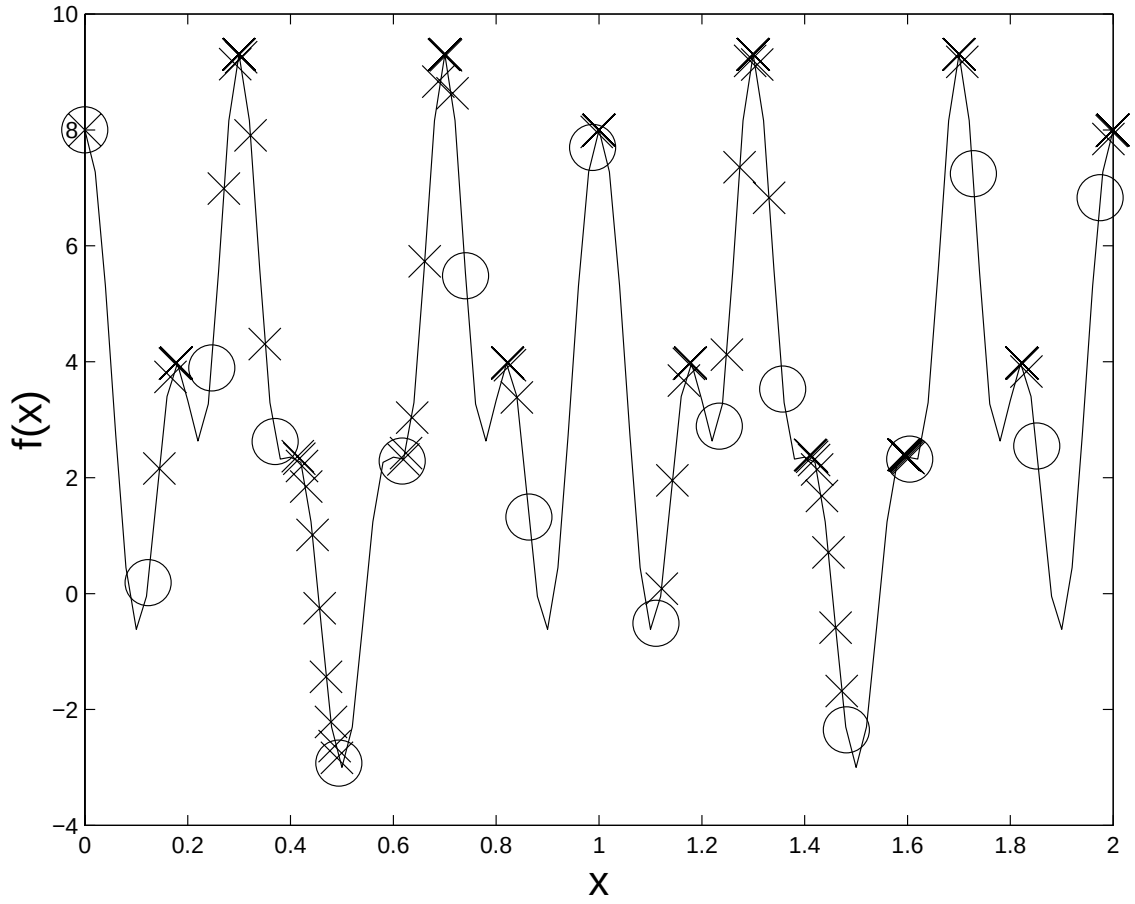


Figure 4.4: Weak-sense auxiliary function maximising a more complex function

4.2.3 Strong-sense auxiliary functions for ML estimation

An HMM likelihood $p(\mathcal{O}|\mathcal{M},\lambda)$ is a sum over state sequences $\sum_x p(\mathcal{O}|\mathcal{M},\lambda,x)$ where the x are different possible sequences of HMM states. If the objective function $\mathcal{F}(\lambda)$ to be maximised equals the log likelihood, $\log p(\mathcal{O}|\mathcal{M},\lambda)$, it can be written in general terms as follows:

$$\mathcal{F}(\lambda) = \log \sum_x f_x(\lambda), \tag{4.10}$$

where the x correspond to state sequences and $f_x(\lambda)$ are state-conditional likelihoods $p(\mathcal{O}|\mathcal{M},\lambda,x)$. If the optimisation is started at $\lambda = \lambda'$, a strong-sense auxiliary function for $\mathcal{F}(\lambda)$ is

$$\mathcal{G}(\lambda,\lambda') = \sum_x \frac{f_x(\lambda')}{\sum_y f_y(\lambda')} \log f_x(\lambda). \tag{4.11}$$

The inequality $\mathcal{G}(\lambda,\lambda') - \mathcal{G}(\lambda',\lambda') \leq \mathcal{F}(\lambda) - \mathcal{F}(\lambda')$ (Eq. (4.1)) can be shown to hold for the $\mathcal{F}(\lambda)$ and $\mathcal{G}(\lambda,\lambda')$ of Equations (4.10) and (4.11); it reduces to an equation involving the Kullback-Leibler distance[1].

The auxiliary function of Equation (4.11) is a sum of state-sequence log likelihoods $\log f_x(\lambda)$, weighted by the initial posterior probability $\frac{f_x(\lambda')}{\sum_y f_y(\lambda')}$ of the state sequence. It is more usefully expressed as a sum over the invididual Gaussians in the system, separating the p.d.f's for each Gaussian, rather than retaining the sum over the very large number of state sequences. This can be done using: $\log f_x(\lambda) = \sum_{t=1}^{T} \log a_{x_t x_{t+1}} + \log \mathcal{N}(o(t)|\mu_{x_t}, \sigma^2_{x_t})$, where a_{ij} are transition probabilities and the state output probabilities are assumed (for simplicity) to be unidimensional Gaussians; x_t is the state active at time τ of state-sequence x.

If the sum over state sequence posterior probabilities $\frac{f_x(\lambda')}{\sum_y f_y(\lambda')}$ for all sequences x that include state j at time τ, is written as $\gamma_j(t)$, the auxiliary function is as follows (ignoring transition values and assuming a single Gaussian per state):

$$\mathcal{G}(\lambda,\lambda') = \sum_{j=1}^{J}\sum_{t=1}^{T} \gamma_j(t) \log \mathcal{N}(o(t)|\mu_j,\sigma_j^2), \tag{4.12}$$

where μ_j and σ_j^2 are the updated mean and variance corresponding to the new parameters λ and $o(t)$ the value of the speech data at time t. The speech data $o(t)$ is considered to be a scalar for simplicity; the same approach is applicable in the usual multi-dimensional case.

The auxiliary function can equivalently be expressed by replacing the sum of log Gaussian likelihood functions $\log \mathcal{N}(\dots)$ with a single expression as follows:

$$\begin{aligned}\mathcal{G}(\lambda,\lambda') &= \sum_{j=1}^{J} -\frac{1}{2}\left(\gamma_j \log(2\pi\sigma^2) + \frac{\theta_j(\mathcal{O}^2) - 2\theta_j(\mathcal{O})\mu_j + \gamma_j\mu_j^2}{\sigma_j^2}\right) \\ &= \sum_{j=1}^{J} Q(\gamma_j, \theta_j(\mathcal{O}), \theta_j(\mathcal{O}^2)|\mu_j,\sigma_j^2)\end{aligned} \tag{4.13}$$

where $\theta_j(\mathcal{O}) = \sum_{t=1}^{T} \gamma_j(t) o(t)$ is the sum of data weighted by posterior probability, $\theta_j(\mathcal{O}^2)$ is the same sum over squared data $\theta_j(\mathcal{O}) = \sum_{t=1}^{T} \gamma_j(t) o(t)^2$ and $\gamma_j = \sum_{t=1}^{T} \gamma_j(t)$ is the occupancy of the state.

[1] This inequality can be quite simply derived as follows: The function $\mathcal{G}(\lambda,\lambda')$ can equivalently be expressed as: $\left(\log \sum_y f_y(\lambda)\right) + \sum_x \frac{f_x(\lambda')}{\sum_y f_y(\lambda')} \log \frac{f_x(\lambda)}{\sum_y f_y(\lambda)}$. The inequality of Equation (4.1) reduces to: $\sum_x \frac{f_x(\lambda')}{\sum_y f_y(\lambda')} \left(\log \frac{f_x(\lambda)}{\sum_y f_y(\lambda)} - \log \frac{f_x(\lambda')}{\sum_y f_y(\lambda')}\right) \leq 0$. This is of the form $\sum_x p(x) \log \frac{q(x)}{p(x)} \leq 0$; using $x - 1 \geq \log x$, this is implied by $\sum_x p(x) \left(\frac{q(x)}{p(x)} - 1\right) \leq 0$, or $\sum_x p(x) - q(x) \leq 0$, or $0 \leq 0$, since $p(x)$ and $q(x)$ sum to one.

The function $Q(\dots)$ is equivalent to a weighted product of Gaussian likelihoods,

$$Q(t, X, S|\mu, \sigma^2) = -\frac{1}{2}\left(t\log(2\pi\sigma^2) + \frac{S - 2X\mu + t\mu^2}{\sigma^2}\right). \tag{4.14}$$

The maximum of the function $Q(\gamma_j, \theta_j(\mathcal{O}), \theta_j(\mathcal{O}^2)|\mu_j, \sigma_j^2)$ occurs where $\mu_j = \frac{\theta_j(\mathcal{O})}{\gamma_j}$ and $\sigma_j^2 = \frac{\theta_j(\mathcal{O}^2) - \theta_j(\mathcal{O})^2}{\gamma_j}$. Unless the HMM parameters are at a local maximum of the likelihood function, this is guaranteed to lead to an increase in likelihood.

Mixture weights and transition probabilities

In the case of mixture-of-Gaussians HMMs with mixture weights c_{jm} for Gaussians $m = 1 \dots M$ for each state j the Gaussian occupation probabilities must be stored separately for each Gaussian, so for instance γ_j becomes γ_{jm}, and the same applies to the other statistics accumulated.

This affects the forward-backward algorithm used to gather statistics from each training file, but as this is a standard technique it will not be discussed further. The extra term that appears in the auxiliary function in the mixture-of-Gaussians case is $\sum_{j=1}^{J}\sum_{m=1}^{M}\gamma_{jm}\log c_{jm}$ which relates to the optimisation of the weights c_{jm}; the weights are subject to a sum-to-one constraint ($\sum_{m=1}^{M} c_{jm} = 1$ for each j) and the maximum of the auxiliary function is where $c_{jm} = \frac{\gamma_{jm}}{\sum_{m=1}^{M}\gamma_{jm}}$.

The part of the auxiliary function due to the transition values can be expressed as $\sum_{i=1}^{J}\sum_{j=1}^{J} t_{ij}a_{ij}$, where t_{ij} is defined as the occupation probability (summed over time τ) of state sequences x with a transition from state i at time $\tau - 1$ to state j at time τ. The solution for a row of transition matrix values (a_{ij} for some i) is analogous to the solution for weights c_{jm} in a state, i.e. $a_{ij} = \frac{t_{ij}}{\sum_{j=1}^{J} t_{ij}}$. Since transition values are analogous to weight values, they will not be treated separately in the context of discriminative updates.

4.2.4 Weak-sense auxiliary functions for MMI estimation

The MMI objective function is a difference of HMM log likelihoods, $\mathcal{F}(\lambda) = \log p(\mathcal{O}|\mathcal{M}^{\text{num}}, \lambda) - \log p(\mathcal{O}|\mathcal{M}^{\text{den}}, \lambda)$, where $\mathcal{M}^{\text{num}}$ and $\mathcal{M}^{\text{den}}$ are HMMs corresponding to the correct transcription and all possible transcriptions, respectively. Strong-sense auxiliary functions as for ML estimation, $\mathcal{G}^{\text{num}}(\lambda, \lambda')$ and $\mathcal{G}^{\text{den}}(\lambda, \lambda')$ can be derived separately for the two log-likelihoods $\log p(\mathcal{O}|\mathcal{M}^{\text{num}})$ and $\log p(\mathcal{O}|\mathcal{M}^{\text{den}})$: the auxiliary functions differ only in the model topology used to accumulate statistics from the training data. A difficulty arises because the second term is negated in the MMI objective function; strong-sense auxiliary functions cannot be used when the problem is negated since the inequality of Eq. (4.1) will no longer hold. However weak-sense auxiliary functions do not

suffer from this problem, and the difference $\mathcal{G}^{\text{num}}(\lambda, \lambda') - \mathcal{G}^{\text{den}}(\lambda, \lambda')$ is still a weak-sense auxiliary function for the MMI objective function.

However, $\mathcal{G}^{\text{num}}(\lambda, \lambda') - \mathcal{G}^{\text{den}}(\lambda, \lambda')$ is not a suitable auxiliary function to use in optimisation because it is convex for some Gaussian parameters (generally where the denominator occupation count is greater than the numerator count). It is necessary to ensure that the auxiliary function is concave. To do this we can add a smoothing function $\mathcal{G}^{\text{sm}}(\lambda, \lambda')$ which can in principle be any function with a zero differential w.r.t. λ around the current value $\lambda = \lambda'$. This will not affect the local differential and the result will be a still be a weak-sense auxiliary function for the MMI objective function. This leads to the following auxiliary function:

$$\mathcal{G}(\lambda, \lambda') = \mathcal{G}^{\text{num}}(\lambda, \lambda') - \mathcal{G}^{\text{den}}(\lambda, \lambda') + \mathcal{G}^{\text{sm}}(\lambda, \lambda'). \tag{4.15}$$

One possible form for $\mathcal{G}^{\text{sm}}(\lambda, \lambda')$ is:

$$\mathcal{G}^{\text{sm}}(\lambda, \lambda') = \sum_{j=1}^{J} Q(D_j, D_j \mu'_j, D_j(\mu'^2_j + \sigma'^2_j) | \mu_j, \sigma_j^2), \tag{4.16}$$

which has a zero differential w.r.t. the parameters σ_j^2 and μ_j evaluated at the old values σ'^2_j and μ'_j, so the auxiliary function is still a weak-sense auxiliary function for the objective function around λ'. D_j are positive smoothing constants for each state j (or each Gaussian j, m in the mixture-of-Gaussians case). A larger value of D_j will slow down the optimisation of the Gaussian mean and variance.

The total auxiliary function (considering only terms involving Gaussian parameters) now becomes:

$$\begin{aligned} \mathcal{G}(\lambda, \lambda') = \sum_{j=1}^{J} & Q(\gamma_j^{\text{num}}, \theta_j^{\text{num}}(\mathcal{O}), \theta_j^{\text{num}}(\mathcal{O}^2) | \mu_j, \sigma_j^2) \\ & - Q(\gamma_j^{\text{den}}, \theta_j^{\text{den}}(\mathcal{O}), \theta_j^{\text{den}}(\mathcal{O}^2) | \mu_j, \sigma_j^2) \\ & + Q(D_j, D_j \mu'_j, D_j(\mu'^2_j + \sigma'^2_j) | \mu_j, \sigma_j^2). \end{aligned} \tag{4.17}$$

The above analysis can trivially be extended for Gaussian mixtures with Gaussian components $m = 1 \ldots M_j$. Maximisation of the function in Eq. 4.17 leads to the Extended Baum-Welch (EB) update equations as follows:

$$\mu_{jm} = \frac{\left\{\theta_{jm}^{\text{num}}(\mathcal{O}) - \theta_{jm}^{\text{den}}(\mathcal{O})\right\} + D_{jm} \mu'_{jm}}{\left\{\gamma_{jm}^{\text{num}} - \gamma_{jm}^{\text{den}}\right\} + D_{jm}} \tag{4.18}$$

$$\sigma_{jm}^2 = \frac{\left\{\theta_{jm}^{\text{num}}(\mathcal{O}^2) - \theta_{jm}^{\text{den}}(\mathcal{O}^2)\right\} + D_{jm}(\sigma'^2_{jm} + \mu'^2_{jm})}{\left\{\gamma_{jm}^{\text{num}} - \gamma_{jm}^{\text{den}}\right\} + D_{jm}} - \mu_{jm}^2. \tag{4.19}$$

This solution is trivial to derive because the three $Q(\dots)$ functions can be combined by adding together the three different arguments. In the implementation used for work in this thesis, the Gaussian-specific smoothing constants D_{jm} are set on a per-Gaussian level to the larger of i) twice the smallest value needed to ensure positive variances, or ii) γ_{jm}^{den} times a further constant E, which is generally set to 1 or 2. This has been found experimentally to lead to good convergence (see Section 5.3.6). The weak-sense auxiliary function is valid for any values of D_{jm}.

Equations (4.18) and (4.19) are the Extended Baum-Welch update equations as given for instance in [Normandin & Morgera, 1991].

Relation of EB to GPD

Generalised Probabilistic Descent (GPD) [Juang & Katagiri, 1992] is an optimisation method based on gradient descent, based on the Probability Descent Theorem [Amari, 1967]. The parameters λ are changed by $\epsilon_t \mathbf{U} \nabla_t$ on each iteration, where $\mathbf{U}$ is a positive definite matrix, ∇_t is the differential of the objective function w.r.t. λ on iteration t, and ϵ_t is a time-dependent learning rate with the property that $\sum_{t=1}^{\infty} \epsilon_t = \infty$ and $\sum_{t=1}^{\infty} \epsilon_t^2 < \infty$: e.g., $\epsilon_t = \frac{1}{t}$ has this property. This guarantees convergence to a local maximum of the objective function.

GPD is equivalent to a particular case of a weak-sense auxiliary function. If the auxiliary function on iteration t is $\lambda^T \nabla_t + \frac{1}{2\epsilon_t}(\lambda - \lambda')^T \mathbf{U}^{-1} (\lambda - \lambda')$, the update will be the same as for GPD with learning rate $\epsilon_t = \frac{1}{t}$. This equivalence is interesting because it suggests that the smoothing constants D_{jm} could be set to values increasing on each iteration. The term $\frac{1}{2\epsilon_t}$ in the auxiliary function occurs in front of a smoothing function (i.e. a function with zero differential around λ'). It is known that GPD is guaranteed to converge when $\epsilon_t \propto \frac{1}{t}$. This suggests that setting the constants D_{jm} to values proportional to t might lead to good results. This possibility is explored experimentally in Section 8.13.

4.3 MAP updates

4.3.1 Incorporating priors into auxiliary functions

Any function is both a weak and strong-sense auxiliary function of itself around any point. Therefore if we add a log prior distribution $\log p(\lambda)$ to the MMI objective function making

$$\mathcal{F}(\lambda) = \log p(\mathcal{O}|\mathcal{M}^{\text{num}}) - \log p(\mathcal{O}|\mathcal{M}^{\text{den}}) + \log p(\lambda), \tag{4.20}$$

the extra term can simply be added to the auxiliary function leading to

$$\mathcal{G}(\lambda, \lambda') = \mathcal{G}^{\text{num}}(\lambda, \lambda') - \mathcal{G}^{\text{den}}(\lambda, \lambda') + \mathcal{G}^{\text{sm}}(\lambda, \lambda') + \log p(\lambda). \tag{4.21}$$

4.3.2 Priors over Gaussian parameters

A convenient prior distribution over a mean μ and variance σ^2 is:

$$\log p(\mu, \sigma^2) = k + Q(\tau, \tau\mu_{\text{prior}}, \tau(\sigma^2_{\text{prior}} + \mu^2_{\text{prior}})|\mu, \sigma^2), \tag{4.22}$$

where $Q(\dots)$, defined in Eq. 4.14, is equivalent to a log likelihood of τ points of data with mean μ_{prior} and variance σ^2_{prior}, and k is a normalisation term which can be ignored.

It is possible to obtain some intuition about this prior by considering the mean and variance separately. For the mean alone, considering the variance a constant, this prior is a Gaussian with variance $\frac{\sigma^2}{\tau}$, i.e. $\frac{1}{\tau}$ times the variance of the distribution itself, as in conventional MAP [Gauvain & Lee, 1994]. For the variance, defining $S = (\mu - \mu_{\text{prior}})^2 + \sigma^2_{\text{prior}}$, matching the first and second-order terms of the Taylor expansion around the value $\sigma^2 = S$ shows that the distribution over σ^2 is locally equivalent to a Gaussian distribution with mean S and variance $\frac{2S^2}{\tau}$. The prior over the variance differs from the standard approach to priors over variances [Gauvain & Lee, 1994], in that the mode of the variance prior varies as a function of the updated mean.

4.3.3 I-smoothing

The H-criterion (Section 4.3.4) uses a fixed interpolation between the MLE (H=0) and MMI (H=1) objective functions.

I-smoothing is a proposed alternative technique which is similar in effect to the H-criterion but uses a different interpolation constant for each Gaussian depending on the amount of data available. I-smoothing can be regarded as the use of a prior over the parameters of each Gaussian, with the prior being based on the ML statistics. The log prior likelihood is equal to

$$\log p(\mu_{jm}, \sigma^2_{jm}) = Q(\tau^I, \tau^I \frac{\theta^{\text{num}}_{jm}(\mathcal{O})}{\gamma^{\text{num}}_{jm}}, \tau^I \frac{\theta^{\text{num}}_{jm}(\mathcal{O}^2)}{\gamma^{\text{num}}_{jm}} | \mu_{jm}, \sigma^2_{jm}) + k \tag{4.23}$$

where k is a normalising term to make the distribution integrate to 1. This prior is proportional to the likelihood of τ^I points of data with mean and variance equal to the numerator (correct model) mean and variance, where τ^I is a constant affecting the narrowness of the prior. The prior likelihood can be integrated into the optimisation procedure by altering the numerator statistics as follows before

use in parameter update:

$$\gamma_{jm}^{\text{num}\prime} = \gamma_{jm}^{\text{num}} + \tau^I \tag{4.24}$$

$$\theta_{jm}^{\text{num}}(\mathcal{O})' = \theta_{jm}^{\text{num}}(\mathcal{O})\frac{\gamma_{jm}^{\text{num}} + \tau^I}{\gamma_{jm}^{\text{num}}} \tag{4.25}$$

$$\theta_{jm}^{\text{num}}(\mathcal{O}^2)' = \theta_{jm}^{\text{num}}(\mathcal{O}^2)\frac{\gamma_{jm}^{\text{num}} + \tau^I}{\gamma_{jm}^{\text{num}}}, \tag{4.26}$$

which arises from combining the statistics of the $Q(\dots)$ function of Equation (4.23) with the numerator statistics so that the numerator objective function now incorporates the prior. Typically τ^I is set to around 100 for MMI training. This is higher than the analogous constant τ typically used in MAP training for speaker adaptation, which might be set to around 10 or 20. This can be explained by observing that MMI-estimated parameters differ very little from the corresponding ML parameters, so a narrow prior is called for.

Section 8.1 gives experimental results showing the effect of I-smoothing on recognition results for MMI and MPE training on a variety of corpora.

4.3.4 The H-criterion

The H-critierion [Gopalakrishnan et al., 1988], which is related to I-smoothing, is an interpolation between the MMI and ML objective functions, so

$$\mathcal{F}(\lambda) = \sum_{r=1}^{R} \log p_\lambda\left(\mathcal{O}_r|s_r\right)^\kappa P(s_r)^\kappa - H \log \sum_s p_\lambda\left(\mathcal{O}_r|s\right)^\kappa P(s)^\kappa. \tag{4.27}$$

For H=1 this is equivalent to MMI (Equation (3.2)) and for H=0 it is equivalent to the ML criterion. In experiments on the Switchboard database reported in [Woodland & Povey, 2002], the H-criterion did not seem to reduce WER relative to MMI, although it may useful as a technique to make MMI training converge without over-training.

4.3.5 Dimension-specific I-smoothing

The constant τ for I-smoothing of discriminatively trained systems has generally been set to a constant value based on trial and error, but the probabilistic framework makes possible a more empirically motivated approach. This will be briefly presented as follows.

As mentioned in Section 4.3.2, the meaning of the τ value in terms of the prior distributions over the parameters is for the mean $\tau = \frac{\sigma_{jm}^2}{\sigma_{\text{mean}}^2}$ and for the variance

$\tau = 2\frac{\mu_{\text{var}}^2}{\sigma_{\text{var}}^2}$. These formulae can be used to motivate a way of setting dimension-specific values of τ.

The modes of the prior distributions for means and variances in I-smoothing are set to the ML-estimated parameters. It is not clear how to calculate the variances of these distributions since the "perfect" discriminatively trained parameters (i.e. with infinite training data) are unavailable so we cannot know how much these tend to vary from the ML estimates. However, it is possible to calculate the τ values corresponding to the global distribution of parameters in the initial HMM set, and scale these appropriately to get the narrower priors needed for I-smoothing. Using $\mu_{\text{mean}}(d)$ and $\sigma_{\text{mean}}^2(d)$ to denote the mean and variance of the global distribution of means in the HMM set for dimension d, and $\mu_{\text{var}}(d)$ and $\sigma_{\text{var}}^2(d)$ for the variances, the τ values for each dimension can be calculated as $\frac{\mu_{\text{var}}(d)}{\sigma_{\text{mean}}^2(d)}$ and $2\frac{\mu_{\text{var}}(d)^2}{\sigma_{\text{var}}^2(d)}$ for the mean and variance respectively. These values of τ can vary from dimension to dimension between approximately 0.1 and 2 for the means and 1 and 5 for the variance. Since narrower priors (larger values of τ) are needed for I-smoothing, the τ values are calculated using the following heuristic:

$$\tau_{\text{mean}}(d) = \tau_{\text{min}} + \alpha\frac{\mu_{\text{var}}(d)}{\sigma_{\text{mean}}^2(d)} \tag{4.28}$$

$$\tau_{\text{var}}(d) = \tau_{\text{min}} + 2\alpha\frac{\mu_{\text{var}}(d)^2}{\sigma_{\text{mean}}^2(d)} \tag{4.29}$$

where $\alpha \leq 1$ is a scaling constant and τ_{min} is a minimum τ value which prevents too much variation in the values for different dimensions. Note that the formula for τ for the mean uses $\mu_{\text{var}}(d)$ as an approximate value for the individual variance σ_{jm}^2, to make the values of $\tau_{\text{mean}}(d)$ constant for each dimension. It has been found empirically that if a value τ was previously found to be appropriate, good results can be obtained if this is replaced with the formulae above using $\tau_{\text{min}} = 0.5\tau$ and $\alpha = 0.4$.

The values $\tau_{\text{mean}}(d)$ and $\tau_{\text{var}}(d)$ control the widths of the prior distributions for each dimension. The modes of the prior distributions can be supplied as usual by the ML estimates. With different τ values for the mean and variance it is difficult to find an elegant form of prior which does not change the form of the update equations. The approach taken was to update the mean and then the variance, with the variance estimate depending on the mean estimate. This can be expressed in terms of $Q(\dots)$ functions if the mean is first estimated by auxiliary functions of the form $Q(\dots|{\sigma'_{jm}}^2, \mu_{jm})$ with the old variance σ'_{jm} used as a fixed value for the variance; and the variance is then estimated using auxiliary functions of the form $Q(\dots|\sigma_{jm}^2, \mu_{jm})$ with the updated value of μ_{jm} fixed and only σ_{jm}^2 variable. The separation is necessary because of the different values of

τ. The update equations used are given as follows:

$$\mu_{jm} = \frac{\theta^{\text{num}}_{jm}(\mathcal{O}) - \theta^{\text{den}}_{jm}(\mathcal{O}) + D_{jm}\mu'_{jm} + \frac{\tau_{\text{mean}}(d)}{\gamma^{\text{num}}_{jm}}\theta^{\text{num}}_{jm}(\mathcal{O})}{\gamma^{\text{num}}_{jm} - \gamma^{\text{den}}_{jm} + D_{jm} + \tau_{\text{mean}}(d)} \tag{4.30}$$

$$\sigma^2_{jm} = \frac{S^{\text{num}}_{jm} - S^{\text{den}}_{jm} + S^{\text{sm}}_{jm} + \frac{\tau_{\text{var}}(d)}{\gamma^{\text{num}}_{jm}}S^{\text{num}}_{jm}}{\gamma^{\text{num}}_{jm} - \gamma^{\text{den}}_{jm} + D_{jm} + \tau_{\text{var}}(d)}. \tag{4.31}$$

where $S^{\text{num}}_{jm} = \theta^{\text{num}}_{jm}(\mathcal{O}^2) - 2\theta^{\text{num}}_{jm}(\mathcal{O})\mu_{jm} + \gamma^{\text{num}}_{jm}\mu^2_{jm}$ is the scaled variance of the numerator statistics around the updated mean and S^{den}_{jm} is similarly defined, and $S^{\text{sm}}_{jm} = D_{jm}({\sigma'_{jm}}^2 + {\mu'_{jm}}^2 - 2\mu'_{jm}\mu_{jm} + \mu^2_{jm})$. A more derivation of these formulae is not given since dimension-specific I-smoothing is not recommended as a standard technique.

Experiments on the effect of dimension-specific τ values in the context of MPE training are given in Section 8.12. This approach gives a small improvement over the baseline of a fixed τ in most cases but has not been adopted as a standard technique due to the increase in system complexity.

4.4 Mixture weights and transition probabilities

This section derives a new update rule for mixture weights and transition probabilities. The update rule is derived for the case of the weights; as observed in Section 4.2.3, the Gaussian weight optimisation is exactly analogous to the optimisation of a row of transition matrix values, so transition probabilities will not be considered separately. This section explains the basic approach to weight and transition probability optimisation used for most of the work presented in this thesis.

Updated mixture weights c_{jm} for a state j are calculated by maximising the following auxiliary function:

$$\mathcal{G}(\lambda, \lambda') = \sum_{m=1}^{M} \gamma^{\text{num}}_{jm} \log c_{jm} - \frac{\gamma^{\text{den}}_{jm}}{c'_{jm}} c_{jm}, \tag{4.32}$$

subject to the sum-to-one constraint; a similar method is used for each row of the transition matrices. c'_{jm} are the weights from the previous HMM set.

This auxiliary function can be shown to be a weak-sense auxiliary function for the MMI objective function as follows. Firstly, the function $\mathcal{G}^{\text{num}}(\lambda, \lambda') - \mathcal{G}^{\text{den}}(\lambda, \lambda')$ is a weak-sense auxiliary function for the MMI objective function, where $\mathcal{G}^{\text{num}}(\lambda, \lambda')$ and $\mathcal{G}^{\text{den}}(\lambda, \lambda')$ are auxiliary functions of the kind used in ML estimation (see

Section 4.2.3). The term in this function including the weights c_{jm} of state j is:

$$\sum_{m=1}^{M} \left(\gamma_{jm}^{\text{num}} - \gamma_{jm}^{\text{den}}\right) \log c_{jm}. \tag{4.33}$$

The auxiliary function of Equation (4.32) has the same differential as Equation (4.33) w.r.t. each c_{jm} at the initial position where each $c_{jm} = c'_{jm}$. Since Equation (4.33) is a weak-sense auxiliary function for the MMI objective function around λ', so is Equation (4.32).

The auxiliary function of Equation (4.32) is optimised as follows. For all j, m set $c_{jm}^{(0)} = c'_{jm}$ (i.e. to the original values from before the optimisation) and then for iterations $p = 0$ to, say, 100, set for all j, m:

$$c_{jm}^{(p+1)} = \frac{\gamma_{jm}^{\text{num}} + k_{jm} c_{jm}^{(p)}}{\sum_m \gamma_{jm}^{\text{num}} + k_{jm} c_{jm}^{(p)}}, \tag{4.34}$$

where

$$k_{jm} = \left(\max_m \frac{\gamma_{jm}^{\text{den}}}{c'_{jm}}\right) - \frac{\gamma_{jm}^{\text{den}}}{c'_{jm}}. \tag{4.35}$$

The values of $c_{jm}^{(p)}$ after 100 iterations are used for the updated values (convergence can be slow enough to make 100 iterations necessary for some values of parameters and statistics).

The proof that this formula will be effective is as follows. Suppose the current iteration of optimisation is p, and for a particular state j a strong-sense auxiliary function for the objective function[2] of Equation (4.32) is desired, which will help us find values $c_{jm}^{(p+1)}$ that increase the objective function. The objective function being optimised on iteration p is as follows:

$$\mathcal{F}(\lambda^{(p+1)}) = \sum_{m=1}^{M} \gamma_{jm}^{\text{num}} \log c_{jm}^{(p+1)} - \frac{\gamma_{jm}^{\text{den}}}{c'_{jm}} c_{jm}^{(p+1)}, \tag{4.36}$$

which is a function of the unknowns $c_{jm}^{(p+1)}$ for $m = 1 \ldots M$. The "starting point" for the optimisation is the values on the previous iteration, $c_{jm}^{(p)}$. Any function, for instance the one in Equation (4.36), is a strong-sense auxiliary function of itself. This property is unchanged by adding a smoothing function as defined in Equation 4.3. The smoothing function we add in order to make Equation (4.36) analytically tractable is $\sum_{m=1}^{M} k_{jm} \left(c_{jm}^{(p)} \log c_{jm}^{(p+1)} - c_{jm}^{(p+1)}\right)$, where k_{jm} are positive

[2]Note that the "objective function" referred to here is actually an auxiliary function as far as the MMI optimisation is concerned.

constants for each m. This function has its largest value at the "starting point" where $c_{jm}^{(p+1)} = c_{jm}^{(p)}$, hence it is a valid smoothing function. The auxiliary function now becomes:

$$\mathcal{G}(\lambda^{(p+1)}, \lambda^{(p)}) = \sum_{m=1}^{M} \gamma_{jm}^{\text{num}} \log c_{jm}^{(p+1)} - \frac{\gamma_{jm}^{\text{den}}}{c'_{jm}} c_{jm}^{(p+1)} + k_{jm} \left(c_{jm}^{(p)} \log c_{jm}^{(p+1)} - c_{jm}^{(p+1)} \right) \tag{4.37}$$

The k_{jm} are chosen so as to make the linear terms in $c_{jm}^{(p+1)}$ in Equation (4.36) all have the same coefficient, so that due to the sum-to-one constraint the linear terms reduce to a constant independent of the weights.

For optimisation which is as fast as possible, the coefficients k_{jm} are chosen to be the lowest values which will make the coefficients of the terms in $c_{jm}^{(p+1)}$ all be the same; the smallest of k_{jm} will always be zero. This leads to the expression in Equation (4.35) for k_{jm}.

The coefficients of terms in $c_{jm}^{(p+1)}$ become equal to $\frac{-\gamma_{jm}^{\text{den}}}{c'_{jm}} - k_{jm} = -\max_m \frac{\gamma_{jm}^{\text{den}}}{c'_{jm}}$, which is a constant independent of m. The coefficients of terms in $\log c_{jm}^{(p+1)}$ become equal to $\gamma_{jm}^{\text{num}} + c_{jm}^{(p)} k_{jm}$, and the maximum of the function can be found e.g. using Lagrangian multipliers; this leads to the update in Equation (4.34), which is analogous to the normal Baum-Welch update for weights.

Section 6.3 a more general version of this update rule is derived and experimental results for different update rules including the previous Extended Baum-Welch approach are compared in Section 6.4.3.

4.4.1 Priors for mixture weights and transition probabilities

In some cases there may not be enough training data for discriminative training of mixture weights and transition probabilities. In order to give good results in these cases, prior information can be incorporated into the auxiliary function. In the case of the weights, for instance, the prior distribution used is $\sum_m \tau^W \frac{\gamma_{jm}^{\text{num}}}{\sum_m \gamma_{jm}^{\text{num}}} \log c_{jm}$ which has its maximum where the updated mixture weights c_{jm} equal the Maximum Likelihood solution for the weights. τ^W is a constant which determines the width of the prior over the weights. This is implemented by altering the statistics as follows:

$$\gamma'^{\text{num}}_{jm} = \gamma_{jm}^{\text{num}} + \frac{\tau^W \gamma_{jm}^{\text{num}}}{\sum_m \gamma_{jm}^{\text{num}}} \tag{4.38}$$

prior to updating the weights.

A similar approach is used for each row of the transition matrix, controlled by a constant τ^T. Priors over weights and transition probabilities make essentially no difference to recognition results under the experimental setup used here, and so will not be discussed further. Experiments reported here do not use weight and transition smoothing unless otherwise indicated. A small value of $\tau^W = 1$ and $\tau^T = 1$ is recommended for robust discriminative training. Although it makes virtually no difference to test set results (Section 8.17), it may help in cases where some transition matrices have very little data available for training.

4.5 Previous work on Extended Baum-Welch updates

This section discusses the original proof for the Extended Baum-Welch (EB) equations, and explains how the proof given in this chapter differs from previous approaches. Section 4.5.1 briefly explains the Baum-Welch update for ML training. Section 4.5.2 explains the basis of the EB equations as applied to discrete HMMs. Section 4.5.3 discusses the continuous-density version of the EB equations. Section 4.5.3 discusses the smoothing constant D in the EB equations. Section 4.5.4 discusses more recent approaches to justifying the EB update equations. Section 4.5.5 explains the advantages of weak-sense auxiliary functions for proving the validity of the EB equations.

4.5.1 Baum-Welch update for ML training of discrete HMMs

The Baum-Eagon inequality [Baum et al., 1970] gives a way to iteratively maximise polynomials of variables where groups of the variables are subject to sum-to-one constraints, and all polynomial coefficients are positive. This is exactly the case encountered in the optimisation of discrete-probability HMM output likelihoods during Maximum Likelihood training.

The Baum-Eagon inequality is formulated for the case where there are variables x_{ij} in a matrix X containing rows with a sum-to-one constraint $\sum_j x_{ij} = 1$, and we are maximising a sum of polynomial terms in x_{ij} with nonnegative coefficients. The inequality is of the form given in Equation (4.1), making the auxiliary function what is here termed a strong-sense auxiliary function. The auxiliary function is the same as that given in Equation (4.11), a weighted sum of logarithms. Finding the maximum of the auxiliary function (e.g. using Lagrangian multipliers) leads to the following update, which is a "growth transformation" for the poly-

nomial:

$$x_{ij} = \frac{x'_{ij} \frac{\partial F}{\partial x_{ij}}\Big|_{X=X'}}{\sum_k x'_{ik} \frac{\partial F}{\partial x_{ik}}\Big|_{X=X'}}, \tag{4.39}$$

where x'_{ij} are the previous parameters and x_{ij} the updated parameters. A growth transformation is a transformation of the parameters X which will increase the objective function unless it is already at a local maximum.

The Baum-Welch update is an update procedure for HMMs which uses this growth transformation together with an algorithm known as the "forward-backward" algorithm for finding the relevant differentials efficiently.

4.5.2 Extended Baum-Welch for discriminative training of discrete HMMs

An update rule as convenient and provably correct as the Baum-Welch update is not available for discriminative training of HMMs, which is a harder optimisation problem. Early work on discriminative training used gradient descent. Alternative update methods were sought and this resulted in the Extended Baum-Welch equations (EB) [Gopalakrishnan et al., 1989].

The Extended Baum-Welch (EB) update equation as originally derived is applicable to rational functions of parameters which are subject to sum-to-one constraints. (A rational function is a ratio of two polynomials). The MMI objective function for discrete-probability HMMs is an example of such a function. The two essential points used to derive the EB update for MMI are:

1. Instead of maximising $f(X) = \frac{a(X)}{b(X)}$ for positive $a(X)$ and $b(X)$, we can instead maximise $g(X) = a(X) - kb(X)$ where $k = a(X')/b(X')$ and X' are the values from the previous iteration; increasing $g(X)$ will cause $f(X)$ to increase. This is because $g(X)$ (when scaled by the constant $\frac{1}{b(X')}$) is a strong-sense auxiliary function for $f(X)$ around X'.

2. If some terms in the resulting polynomial are negative, we can add to the expression a constant C times a further polynomial which is constrained to be a constant (e.g, $C \prod_i \sum_j x_{ij}$), so as to ensure that no product of terms in the final expression has a negative coefficient.

By applying these two ideas something of the form $\frac{a(X)}{b(X)}$, where $a(X)$ and $b(X)$ are polynomials in x_{ij}, can be massaged into a form suitable for the Baum-Welch update, which only works for polynomials with nonnegative coefficients.

The Baum-Welch update, given in its canonical form in Equation (4.39), may also be expressed as:

$$x_{ij} = \frac{\left.\frac{\partial F}{\partial \log(x_{ij})}\right|_{X=X'}}{\sum_k \left.\frac{\partial F}{\partial \log(x_{ik})}\right|_{X=X'}}. \tag{4.40}$$

Here, differentials are given with respect to the value of $\log x_{ik}$ (using $\frac{\partial F}{\partial \log(x_{ij})} = x_{ij}\frac{\partial F}{\partial x_{ij}}$); this makes the equation simpler and makes the differentials correspond directly to Gaussian occupancies γ_{jm}.

Equation (4.40) gives an update for polynomials with all positive coefficients (the ML case); for the more general class of functions considered in EB, the application of the two insights mentioned above leads to an update as follows:

$$x_{ij} = \frac{\left.\frac{\partial F}{\partial \log(x_{ij})}\right|_{X=X'} + Cx'_{ij}}{\sum_k \left.\frac{\partial F}{\partial \log(x_{ik})}\right|_{X=X'} + Cx'_{ik}}, \tag{4.41}$$

where C is a smoothing constant; C must be set according to a formula that for the case of HMM updates makes it far too large for practical use. In practice C is set to the smallest value necessary to make all updated values x_{ij} positive plus a small constant ϵ.

This approach is applicable to the optimisation of Gaussian mixture weights in MMI training. This update for mixture weights c_{jm} for state j, becomes:

$$c_{jm} = \frac{\frac{\partial}{\partial \log c_{jm}}\mathcal{F}_{\mathrm{MMI}}(\lambda)|_{\lambda=\lambda'} + Cc'_{jm}}{\sum_m \frac{\partial}{\partial \log c_{jm}}\mathcal{F}_{\mathrm{MMI}}(\lambda)|_{\lambda=\lambda'} + Cc'_{jm}}. \tag{4.42}$$

Expressed in terms of occupation counts, the update is as follows:

$$c_{jm} = \frac{\gamma_{jm}^{\mathrm{num}} - \gamma_{jm}^{\mathrm{den}} + Cc'_{jm}}{\sum_m \gamma_{jm}^{\mathrm{num}} - \gamma_{jm}^{\mathrm{den}} + Cc'_{jm}}, \tag{4.43}$$

so C would be set to $\epsilon + \max_{jm} \frac{\gamma_{jm}^{\mathrm{den}} - \gamma_{jm}^{\mathrm{num}}}{c'_{jm}}$. Unfortunately C is dominated by small-valued parameters and can become large, so training proceeds very slowly.

When Normandin [Normandin & Morgera, 1991] did his work on MMI training of continuous and discrete HMMs he found that for training discrete HMMs a slightly altered form of the EB formulae gave better performance. This was based on the idea in [Merialdo, 1988] of replacing the gradient $\partial\mathcal{F}/\partial x_{ij}$ (which can be very large for small-valued parameters) with a different formula, as follows. If the numerator and denominator occupancies are γ_m^{num} and γ_m^{den} and the mixture weights are c_m, he replaced:

$$\frac{\partial\mathcal{F}}{\partial c_{jm}} = \frac{\gamma_{jm}^{\mathrm{num}} - \gamma_{jm}^{\mathrm{den}}}{c'_{jm}} \tag{4.44}$$

with:

$$\frac{\partial \mathcal{F}}{\partial c_{jm}} = \frac{\gamma_{jm}^{\text{num}}}{\sum_m \gamma_{jm}^{\text{num}}} - \frac{\gamma_{jm}^{\text{den}}}{\sum_m \gamma_{jm}^{\text{den}}}, \tag{4.45}$$

where the differential $\frac{\partial \mathcal{F}}{\partial c_{jm}}$ is required in Equation (4.42). This substitution within the EB formula has since been used successfully (e.g, [Valtchev et al., 1996, Schluter, 2000]), although it cannot be guaranteed to work and does not in practice increase the criterion on every iteration, as Normandin noted. For this reason an alternative approach to mixture weight and transition probability training has been developed, as described in Section 4.4 (See Section 6.4.3 for an experimental comparison between the two).

4.5.3 EB for Continuous Density HMMs

Normandin also extended the EB equations to the case of continuous density HMMs [Normandin & Morgera, 1991]. This was done by viewing a Gaussian as a way of parametrising a discrete distribution. The resulting formulae for means and variances are the EB update formulae of Equations (4.18) and (4.19). The proof of the validity of the update rule [Normandin, 1991] was quite complex and was only valid in the limit of a "discretisation" step size becoming zero, so the proof is only valid for infinite D, as opposed to a finite but extremely large constant C for the discrete density case of the EB equations.

Setting the smoothing constant for continuous EB

Equations (4.18) and (4.19) contain a "smoothing constant" D_{jm} which controls the speed of optimisation for each Gaussian m of state j. The value of this constant is critical. In the original presentation of the EB equations, D_{jm} was set on a global level to twice the value necessary to ensure all positive variances. In earlier work carried out in Cambridge [Valtchev et al., 1996], D was set in the same way but separately for each phone HMM, which led to faster optimisation. For work carried out in this thesis D is set on a per-Gaussian level, to the larger of: (i) twice the value necessary for all positive variances, or (ii) a further constant E multiplied by the denominator occupancy γ_{jm}^{den}. E=1 or E=2 is generally used. Schluter [Schluter et al., 1997] suggested setting D for state j depending on a constant h to:

$$D_j = h \cdot \max\left\{ D_{\min}, \frac{1}{\beta} + \gamma_j^{\text{den}} - \gamma_j^{\text{num}} \right\}, \tag{4.46}$$

where $D_{\min}$ is the minimum value which guarantees all positive variances for state j. Values of h such as 1.1 or 2 were used; h is roughly similar to the constant

E used here. Section 5.3.6 gives an experimental comparison of Schluter's approach with that used here showing that Schluter's approach is more effective in optimising the criterion but the current approach is more effective in improving WER.

In a few experiments reported here, E = halfmax indicates the approach of setting E to half the maximum ratio of $\frac{D}{\gamma_{jm}^{\text{den}}}$ in the whole HMM set if D had been set according to the rule "twice the value necessary for positive variances in all updated Gaussians". In typical cases this resulted in a value of E which rose during optimisation; for example, from about 2 to 6 during successive iterations; this rule was felt desirable because in informal experiments on a variety of very different systems including a tied-mixture system this formulation was found to work well. However, a fixed value of E=1 or E=2 has generally been used since it is simpler.

4.5.4 Other approaches to justifying the EB equations

The discriminative versions of the EB update for Gaussians (Equations (4.18) and (4.19)) are poorly justified because the original proof of their validity [Normandin, 1991] only applied to the case of an infinite value of the smoothing constant D, whereas a finite value of D is used in practice.

Some authors in search of a simpler justification for EB have shown that similar equations can be derived in a gradient descent framework, as long as the step sizes are chosen appropriately [Schluter et al., 1997, Zheng et al., 2001]. Of course, this entails giving up any claim to be a so-called "growth transformation", i.e. an update rule which is guaranteed to increase the objective function if it is not already at a local maximum.

More recently, a novel proof of the EB equations has been proposed [Gunawardana, 2001]. The work is interesting in that it leads to a derivation of both the EB update equations and an update for discriminative MLLR. However, I do not believe it successfully demonstrates a way to set the smoothing constant D_{jm} to a finite value and still guarantee convergence. Indeed, I believe it is impossible to prove convergence for finite D without accumulating extra statistics (e.g., about the maximum length of training files) because for any finite change in the HMM parameters, there can potentially be an unbounded increase in the likelihood of a path in the denominator HMM.

Difficulty of proving convergence under finite D

The difficulty of proving convergence for any finite value of D arises from the problem that very small changes in Gaussian parameters can in principle cause very large increases in the likelihood of currently unlikely paths appearing in the

denominator HMM. Large increases in the likelihood of denominator paths are not reflected in the auxiliary function, which varies linearly with the log likelihood of such paths. There is no way to put a limit on the increase in likelihood of these unlikely paths unless the maximum range of the data values $o(t)$ is known and the maximum length of the training files is known. These can be used to get a crude limit on the maximum increase in likelihood of unlikely paths. For instance (considering one dimension), if the minimum and maximum values of $o(t)$ are a and b and the maximum training file length is T, consider a change in a Gaussian mean μ_{jm}. If this is increasing by a small amount ϵ, the maximum possible change in log likelihood of that Gaussian is $\epsilon\frac{b-\mu_{jm}}{\sigma_{jm}^2}$ (assuming μ is further from b than a). Let us suppose that a file of length T consists of data values just consisting of the maximum value b, and a wrong path allowed by the language model consists almost entirely of probabilities contributed by the Gaussian m of state j. This could cause a maximum increase in the log likelihood of a path equal to $\epsilon T\frac{b-\mu_{jm}}{\sigma_{jm}^2}$. The difficulty for optimisation arises from the fact that if this increase $\epsilon T\frac{b-\mu_{jm}}{\sigma_{jm}^2}$ is large (more, than, say, 1 or 2) and the posterior probability of the wrong path is initially much less than 1, the increase in log likelihood of the denominator model is not well reflected in the auxiliary function, which is a weighted sum of logs. For any given amount of smoothing function applied (controlled by E) there will be a point where the increase $\epsilon T\frac{b-\mu_{jm}}{\sigma_{jm}^2}$ in log likelihood of the path becomes large enough that the real objective function value decreases more than the auxiliary function (the exact amount will not be worked out here). This is a case of an $\exp(x)$ function (the real objective function) becoming larger than an x term (the auxiliary function). It is necessary to limit the change in log-likelihood of the path to the range where the difference between an exponential and a linear term is not too large: say, less than 1. In this case the change ϵ might have to be limited to $\frac{1}{T\frac{b-\mu_{jm}}{\sigma_{jm}^2}}$, which assuming $b-\mu_{jm}=5\sigma_{jm}$, might give $\epsilon \leq \frac{\sigma_{jm}}{5T}$, or $\frac{1}{5T}$ standard deviations, which obviously becomes very small as T becomes large.

4.5.5 Advantages of weak-sense auxiliary functions for deriving update equations

The main advantage of weak-sense auxiliary functions for deriving discriminative update rules such as the EB equation is that they are simple and flexible. A good example of this flexibility is that the derivation given in this chapter for Gaussian updates can easily be extended to the full-covariance case. The $Q(\dots)$ functions can easily be extended to full covariance matrices; the smoothing function used to ensure concavity can be based on the mean and covariance of the previous Gaussian parameters, and will obviously have zero differential w.r.t. the Gaussian parameters around the old parameter values. The whole objective function has

a similar functional form to the ML objective function for full variance updates, and can be solved in a similar way.

It would not have been possible to extend the original proof of the EB update equations for Gaussians [Normandin, 1991] in the same way, and if the EB equations were viewed as gradient descent [Schluter et al., 1997, Zheng et al., 2001] and the extension done on that basis, questions would have remained regarding the appropriate learning rates for the off-diagonal elements.

Chapter 5

Lattice-based MMI

One of the significant new pieces of work presented in this thesis is a framework for lattice-based MMI training which is effective in improving word error rates in Large Vocabulary Continuous Speech Recognition (LVCSR). This chapter explains the framework for lattice-based MMI training, and gives experimental results investigating some of the details of the training procedure.

Section 5.1 discusses the need for lattices in MMI training and the lattice format used. Section 5.2 discusses the two different ways in which probability scaling can be integrated into the lattice forward-backward algorithm. Section 5.3 gives experimental results investigating various aspects of the MMI training procedure. Section 5.4 summarises the chapter.

5.1 Use of lattices

5.1.1 Need for lattices

The MMI objective function can be expressed as a difference of HMM likelihoods. For R training files, this can be written

$$\mathcal{F}_{\mathrm{MMI}}(\lambda) = \sum_{r=1}^{R} \log p_\lambda^\kappa(\mathcal{O}_r|\mathcal{M}_r^{\mathrm{num}}) - \log p_\lambda^\kappa(\mathcal{O}|\mathcal{M}_r^{\mathrm{den}}), \tag{5.1}$$

where $\mathcal{M}_r^{\mathrm{num}}$ is the HMM corresponding to the correct transcription of utterance r and $\mathcal{M}_r^{\mathrm{den}}$ is the HMM corresponding to all possible transcriptions of the utterance. Probability calculations are carried out with likelihoods scaled by κ.

In order to speed up computation, the generic model $\mathcal{M}_r^{\mathrm{den}}$ can be represented by a model containing only those word sequences that have a reasonably high likelihood for the utterance. A set of possible word-sequences could in principle be represented by an N-best list (a list of the N most likely sentences), but a

much more efficient way to represent them is a lattice. This stores the alternate word sequences in the form of a graph in which the arcs correspond to words; word sequences are encoded by the paths through the graph. The information contained in the lattices used in HTK is detailed in Figure 5.1. One advantage of using lattices is that the same lattice can be used for each iteration of MMI training (and for multiple different MMI experiments, e.g. using different training setups). This separates the most time-consuming aspect of MMI training, which is finding the most likely word sequences, and makes it only necessary to do it once; this approach assumes that the initially generated lattice covers all the word sequences that will have a high probability even given the models generated during later iterations of training.

5.1.2 Previous use of lattices

Early work on speeding up computation for MMI estimation included the use of N-best lists [Chow, 1990] which are calculated once by a speech recognizer and then used to approximate the set of all sentences in the denominator of the MMI objective function. However, this becomes inefficient for very long sentences since the N-best list is a redundant way of storing the information. In [Normandin et al., 1994], lattices compactly encoding alternative hypotheses were generated in a format called a "looped lattice model" and used in MMI training on a 2000-word task.

In [Valtchev et al., 1996, Valtchev et al., 1997], MMI training was performed using lattices generated by the HTK large vocabulary recognition system (Figure 5.1) [Woodland et al., 1995]. Lattices were generated once and then used for multiple iterations of MMI training.

5.1.3 Issues relating to use of lattices

In this thesis, the lattice format of the HTK system is used (Figure 5.1). Lattices compactly encode the sequence and alignment of words and phones, for a set of alternative sentences. Lattices are generated to correspond to both numerator (correct-model) and denominator (recognition-model) HMMs. Some questions relating to MMI training in a lattice context include:

- What scale κ to apply to the likelihoods and at what stage to apply it (Gaussians vs. states vs. whole sentences or phones).
- What kind of language model (e.g, unigram, bigram or trigram) to use on the lattices (and what language model and pruning thresholds to use during lattice generation).
- What size of lattice is necessary for good results?

HTK lattice format
The HTK lattice format consists of nodes representing the starts and end times of words, with arcs representing words connecting the nodes. The lattice format supports:

- Word start and end times.
- Word name and identity of pronunciation variant.
- Acoustic scores of words.
- Language model scores.
- Times of phone boundaries within words, and phones (with context).

Lattices often contain repeated arcs for the same word, to encode slightly different start and end times or previous and following phone contexts.

Figure 5.1: HTK lattice format

- Is it helpful to regenerate lattices more than once during the training process?

These issues will be investigated experimentally in Section 5.3.

5.2 Scaling of likelihoods in lattices

A problem that arises in MMI training is that the likelihood of the data tends to be dominated by one of the sentences in the lattice, and this can lead to training that is not as smooth or robust as would be ideal. A solution to this problem is to introduce the scale κ on acoustic and language-model log likelihoods that appear in the MMI objective function (Equation (3.2)), to make less likely sentences more important. It has been found [Woodland & Povey, 2000] that for good results with MMI it is important to scale down acoustic and language model log likelihoods in this way. Note that since κ is generally the inverse of the original language model scale which has already been applied, the result is that the language model is unscaled and the acoustic model is scaled instead. This makes sense because the language model probabilities are "real" probabilities, whereas the acoustic model probabilities tend to be over-confident due to correlations between frames.

During the MMI training process, this scaling by κ is applied in the forward-backward algorithm which calculates occupation probabilties for HMM states during the E-M process. The lattices represent HMMs, so the forward-backward process is in principle identical to normal HMM training.

There are different ways to perform this scaling by κ during the forward-backward algorithm, and two particular ways have been tried here.

5.2.1 Exact-match forward-backward computation

In the *exact-match* technique, the word and phone boundaries obtained from the lattice are used during forward-backward computation and these are assumed not to change during training. A full forward-backward computation for each phone, within the given phone boundaries, is done with unscaled probabilities. Scaling of acoustic likelihoods is performed by scaling the whole-phone acoustic likelihoods obtained from forward-backward alignment and using these in a phone-level forward-backward algorithm to compute phone occupation probabilities, which are then combined with the within-phone occupation probabilities obtained previously to give state occupation probabilties.

Detail of exact-match algorithm

A more precise description of the exact-match computation is as follows. Let us denote a phone arc within the lattice with the letter q. Each phone arc has a known start and end time s_q and e_q. Forward-backward computation within a single arc gives us within-arc occupation probabilities for each Gaussian m of state j, which we can denote $\gamma_{qjm}^{\text{num}}(t)$ in the numerator case. These occupation probabilities would sum to 1 for each time t between the start and end times of the arc q. The forward-backward algorithm also generates an arc-likelihood $p(\mathcal{O}|q)$ from the beginning to the end of the arc, which is equal to the within-arc "forward" probability at the end state of the phone HMM at the end time e_q (i.e., the likelihood of the acoustic data aligned to the arc q).

These arc likelihoods $p(\mathcal{O}|q)$ are used in a forward-backward pass at the lattice-node level to estimate the arc posterior probability, γ_q (i.e. the probability of traversing that arc). The α (forward) and β (backward) likelihoods in the forward-backward algorithm, and the occupation probabilities γ_q, would be calculated as follows:

$$\alpha_1 = 1, \beta_Q = 1 \tag{5.2}$$

$$\alpha_q = p(\mathcal{O}|q)^\kappa \sum_{r \text{ preceding } q} \alpha_r t_{rq}^\kappa \tag{5.3}$$

$$\beta_q = \sum_{r \text{ following } q} p(\mathcal{O}|r)^\kappa \beta_r t_{qr}^\kappa \tag{5.4}$$

$$\gamma_q = \alpha_q \beta_q \tag{5.5}$$

where the notation $\sum_{r \text{ following } q}$ means arcs r that follow arc q in the lattice structure; the transition probability between arc q and arc r is t_{qr}.

If this is done for the numerator lattice structure giving occupation probabilities γ_q^{num} and for the denominator lattice structure giving occupation probabilities γ_q^{den}, the statistics needed for the EB formulae can then be gathered according to the formulae given as follows, which use the numerator case as an example:

$$\gamma_{jm}^{\text{num}} = \sum_{q=1}^{Q} \sum_{t=s_q}^{e_q} \gamma_{qjm}^{\text{num}}(t) \gamma_q^{\text{num}} \tag{5.6}$$

$$\theta_{jm}^{\text{num}}(\mathcal{O}) = \sum_{q=1}^{Q} \sum_{t=s_q}^{e_q} \gamma_{qjm}^{\text{num}}(t) \gamma_q^{\text{num}} \mathcal{O}(t) \tag{5.7}$$

$$\theta_{jm}^{\text{num}}(\mathcal{O}^2) = \sum_{q=1}^{Q} \sum_{t=s_q}^{e_q} \gamma_{qjm}^{\text{num}}(t) \gamma_q^{\text{num}} \mathcal{O}(t)^2, \tag{5.8}$$

where s_q and e_q are the start and end times of arc q.

The exact-match forward-backward procedure can be optimised for speed by exploiting the fact that a particular model with a particular start and end times often appears in parallel many times in the lattice. The within-arc probabilities γ_{qjm} and the likelihoods $p(\mathcal{O}|q)$ from the beginning to the end of the arc q will be the same for such repeated models and can be calculated just once, and a similar optimisation can be used to speed up the summation of Equations (5.6) to (5.8).

The exact-match forward-backward procedure roughly corresponds to an MMI criterion where the scale κ is applied to each sentence, as in Equation (3.2), repeated here:

$$\mathcal{F}_{\text{MMI}}(\lambda) = \sum_{r=1}^{R} \log \frac{p_\lambda(\mathcal{O}_r|s_r)^\kappa P(s_r)^\kappa}{\sum_s p_\lambda(\mathcal{O}_r|s)^\kappa P(s)^\kappa}, \tag{5.9}$$

which is the formulation of the MMI criterion given in [Schluter & Macherey, 1998]. The correspondence with the exact-match forward-backward procedure is not quite exact because the lattice may contain different pronunciations or alignments of a given sentence, whose likelihoods would be added together after being scaled rather than before as Equation 5.9 would imply; and if a sentence's true likelihood has considerable input from many different alignments which are not included in the lattice, the exact-match algorithm would fail to take these into account.

5.2.2 Full-search forward-backward computation

The previously described exact-match alignment procedure is the one used in all experiments unless otherwise stated. *Full-search* scaling is a different technique, in which a full forward-backward alignment is performed on the lattice, and the

lattice times (extended by a margin at the beginning and end of each model) are used to prune away some likelihood computations and speed up the procedure. Scaling is applied to the state output likelihoods, and the transition likelihoods are not scaled. Phone arcs are merged in all cases where the merging cannot affect the total likelihood of the lattice, i.e, where two instances of a single phone HMM attach at either their beginning or end to the same phone instance.

Note that as the scale κ becomes large the full-search and exact-match techniques become equivalent, as long as the most likely path is included in the recognition lattice.

A more complete description of full-search scaling is not given here since experiments (Section 5.3.5) show that compared to exact-match, full-search scaling yields poorer or similar recognition results, and requires more computation time. Full-search scaling gives particularly poor results when it is computed as exactly as possible, i.e. without using the phone start and end times to limit the alignment of states.

5.3 Experiments on lattice-based MMI

Experiments reported in this chapter investigate various aspects of the use of lattices for MMI training. For a comparison of MMI, I-smoothed MMI and MPE training under a wider variety of conditions, see Chapter 7.

5.3.1 Experimental setup

Experiments on lattice-based MMI are performed on the Switchboard, North American Business News (NAB, also known as Wall Street Journal), and Broadcast News (BN) corpora. Experiments on Switchboard use sets of training data of size 265 hours (h5train00), 68 hours (h5train00sub) and 18 hours (minitrain). Broadcast News training was with a 72 hour subset of training data and NAB training used the 66 hours of channel 1 (close-talking microphone) training data.

Experiments use gender independent, mixture-of-Gaussian HMMs with cross-word state-clustered triphones. The input data is Mel-Frequency Perceptual Linear Prediction (MF-PLP) coefficients, with delta and delta-delta coefficients, 39 dimensions in all.

There are about 6000 states per HMM set, with 12 Gaussians per state (mixture components) for NAB, BN and the 68-hour subset of Switchboard, h5train00sub, and 16 Gaussians per state for the 265-hour h5train00 training set on Switchboard. These are the typical sizes of HMM set as used in Cambridge for experiments with ML training with these corpora at the time these experiments were performed.

Recognition results are produced by rescoring lattices derived from ML-trained HMMs, using a trigram language model. This is faster than unconstrained recognition.

See Appendix A for a more complete description of the experimental setup.

5.3.2 Statistical significance

In this section, statistically significant intervals for WER are calculated for some typical experiments. Since these intervals fall within a fairly narrow range, these can be used as a guide to the significance of other experiments. The methods used for testing significance here are quite simple but do take into account the fact that the errors of different recognisers are correlated [Gillick, L., and Cox, S.J.]. A recent paper on significance testing is [Strik et al., 2000]; the published methods of comparing WERs tend to be very complicated and have not been used here since they tend to involve laborious pairwise comparisons of results. Instead, the ranges in which the statistical significance intervals fall have been calculated for each test set.

Suppose some variation in the training process leads to a transcription that differs by K words from the baseline transcript. These K words can be viewed as the outcome of a random process in which there are a number of locations in the transcription that are liable (with some probability) to change when the variation is made. If each of the locations will change with probability p, the variance associated with each location is $p - p^2$ (in the sense that this is the variance of the distribution of the number of changes). Let us assume for simplicity that the observed number of changes K exactly equals the expected number of changes, so there are K/p of these sites, so the variance in the number of changes is $K(p-p^2)/p = K(1-p)$ (this assumption makes no substantial difference for any reasonable values of K, e.g. $K > 10$). Assuming all changes in the transcription lead to a change in the number of word errors, the variance in the number of word errors will also be $K(1-p)$. Although not all changes in the transcript lead to a change in WER (since they might lead to replacements of one substitution for another, for example) it is simplest to assume that they do, and this will lead to a slightly more conservative estimate of significance. The expression $K(1-p)$ is largest when p is small, which is the easiest assumption to make as there is no way of directly measuring p, so the variance in the number of word errors is assumed to equal K. If there are N words in the reference transcript, the variance in the WER (if the WER is expressed as a proportion of 1 rather than a percentage) will therefore be K/N^2, and the standard deviation in WER is $\sqrt{K}/N$.

Since it will generally not be clear in advance whether a particular change in a program will increase or decrease the error rate, a two-tailed significance test is appropriate. A 95% significance test rejects the null hypothesis (no change in the variable) when the variable is outside the central 95% of the distribution, which

for a Gaussian means more than about 2 standard deviations either way. What this means in percentage WER terms for a result to be significant is indicated for each test set in Table 5.1 for two systems: a typical system with a "big change" (discriminative training vs. MLE) which means large K; and a typical system with a "small change", i.e. some relatively minor change in training setup. This will allow the reader to gauge the significance of results. Since the level of significance only changes with square root of the difference K, the change in WER that is significant does not vary much across different experiments, for a given test set.

Note that the 99% signifiance interval for the two-tailed test is 2.58 standard deviations as opposed to 1.96 for the 95% significance test, so the 99% significance interval is about 30% larger than the 95% significance interval.

5.3.3 Comparison with standard techniques

Of the standard techniques for comparing speech recognition output as used in the NIST scoring software [Pallett, 1990], the Matched Pairs Sentence Segment Word Error is the most powerful (most likely to find a difference), as it is based on individual sentence segments, rather than individual utterances (McNemar test) or speakers (sign test). This was compared with the current technique for the Broadcast News test set bneval96 for the MMI vs. non-MMI comparison as given in Table 5.1. The z statistic found by the test was 6.647 standard deviations for word error rates differing by 2.62% absolute. The two-standard-deviation 95% confidence interval, when expressed as a word error rate, would be $2.62\% \times \frac{2}{6.647}$, which is a 0.79% WER difference. Compare this to the 0.5% in table 5.1 for the same setup. This demonstrates that my approach is somewhat more powerful (or sensitive to differences) than the MAPSSWE test, which is not surprising since it considers individual words to be independent rather than individual sentence segments as in the case of MAPSSWE. The advantage of my approach is that it simplifies the process of comparison by working out approximate significance levels in advance for a given test set, and removing the need for pairwise comparisons of results with statistical scoring software. Also, the more powerful test probably corresponds more closely to the intuition of speech researchers about what is a reliable difference and what is not.

5.3.4 Default values and settings

In order to describe experimental setups as briefly as possible. some settings and parameters are only specified if they differ from a default. Default settings are as follows:

1. Full-search vs. Exact-match alignment: Exact-match is default.

Corpus	Test set	# ref words (N)	Very different systems (ML vs. MMI)	
			% difference start-end ($100K/N$)	95% sig interval ($100 \cdot 2\sqrt{K}/N$)
Swbd	eval98	41178	34%	0.57%
Swbd	eval97sub	11607	34%	1.08%
BN	bneval96	23152	14.6%	0.5%
WSJ	csrnab1_{dev,eval}	15231	3.7%	0.31%
Corpus	Test set	# ref words (N)	Similar systems (MMI, E=1 vs. E=2)	
			% difference start-end ($100K/N$)	95% sig interval ($100 \cdot 2\sqrt{K}/N$)
Swbd	eval98	41178	15%	0.38%
Swbd	eval97sub	11607	15%	0.71%
BN	bneval96	23152	8.6%	0.39%
WSJ	csrnab1_{dev,eval}	15231	2.5%	0.26%

Table 5.1: Significant intervals in WER for example systems.

2. Update equations: EB update with smoothing constant $E = 2.0$ unless stated otherwise.

3. Language model applied to lattices: unigram, with the same scale and insertion penalty as for recognition.

4. Probability scale factor κ: the inverse of the normal language model scale (so 1/12 for Switchboard, 1/15 for NAB and 1/14 for BN).

5. Lattices used: lattices are generated from recognition with a bigram language model, subsequent to which unigram language model probabilities are applied to the lattices. See Appendix A for a more complete description of how the lattices were created.

5.3.5 Full-search vs. Exact-match

Table 5.2 compares the Full-search and Exact-match alignment procedures. It can be seen that the Exact-match procedure gives better recognition performance. In addition, the full-search method is slower: it takes twice long as exact-match (for $\pm 0.05s$ pruning) or fifteen times as long (for $\pm 1.5s$ pruning). These pruning beams refer to the amount of extension of the HMM time boundaries from the phone-marked lattice. Experiments on the full 265h h5train00 training set for Switchboard (not shown) also supported the notion that exact-match alignment gives test-set results of the order of 0.5% absolute better than full-search.

	Iteration				
	0	1	2	3	4
	%WER on eval97sub				
Exact-match, E=1	46.0	43.8	44.1	43.9	44.3
Full-search, E=1, $\pm 0.05s$	46.0	44.2	44.4	44.7	44.6
Full-search, E=1, $\pm 1.5s$	46.0	44.0	44.3	44.4	44.9
Exact-match, E=2	46.0	44.5	43.7	43.9	43.8
Full-search, E=2, $\pm 0.05s$	46.0	44.8	44.3	44.0	44.1
Full-search, E=2, $\pm 1.5s$	46.0	44.0	44.3	44.4	44.9

Table 5.2: Full-search vs. Exact-match training on Switchboard, trained on h5train00sub (65h), ± latitude for full-search.

5.3.6 Alternative approaches to setting D_{jm}

As mentioned in Section 4.5.3, the Gaussian-specific smoothing constant D_{jm} is set to the larger of: (i) twice the value necessary for all positive variances, or (ii) a further constant E multiplied by the denominator occupancy γ_{jm}^{den}. The approach used in [Schluter et al., 1997] was to set $D_{jm} = h \cdot \max\left\{D_{\min}, \frac{1}{\beta} + \gamma_j^{\text{den}} - \gamma_j^{\text{num}}\right\}$, for h typically set to values such as 1.1 or 2.

# mix comps	train setup	Training Iteration 0	1	2	3	4
		Average %WER on csrnab1_{dev,eval}				
12	E=0.5	9.57	9.24	9.07	9.05	9.08
12	E=2	9.57	9.48	9.31	9.26	9.27
12	h=2, $\frac{1}{\beta}$=5	9.57	9.99	9.91	10.16	10.87
		MMI criterion / # frames				
12	E=0.5	-0.0128	-0.0083	-0.0061	-0.0048	
12	E=2	-0.0128	-0.0111	-0.0099	-0.0089	
12	h=2, $\frac{1}{\beta} = 5$	-0.0128	-0.0063	-0.0042	-0.0033	

Table 5.3: Different ways of setting D_{jm}

Table 5.3 compares Schluter's approach, controlled by h and β, with the approach described here using both E=0.5 and E=2 as baselines. E=0.5 is more similar in terms of optimising the criterion. The table shows that while Schluter's approach is more effective in optimising the MMI criterion, the current approach gives better test-set approach– in fact, with these settings Schluter's approach gives a degradation in results compared with ML. This shows the importance of taking error rate into account rather than criterion optimisation when designing

the optimisation technique. The problem may be that in about half of all cases the quantity $\frac{1}{\beta} + \gamma_j^{\text{den}} - \gamma_j^{\text{num}}$ will be zero or negative, so the smoothing constant D_{jm} is set to $hD_{\min}$ instead, i.e. twice the minimum quantity necessary to make all updated variances positive. Roughly speaking, this ensures that on each iteration at least one variance in about half of all Gaussians, is moved about half-way towards becoming zero; i.e. is halved. Such a large movement in parameter values on each iteration must prevent a smooth approach to equilibrium parameter values and introduces considerable randomness into the likelihoods used in recognition.

5.3.7 Setting the constant E and number of iterations

	Training Iteration				
	0	1	2	3	4
	%WER on eval97sub				
E=1	46.0	43.8	44.1	43.9	44.3
E=2	46.0	44.5	43.7	43.9	43.8
E=halfmax (1.9 . . . 4.2)	46.0	44.4	43.7	44.0	43.6
	MMI criterion / # frames				
E=1	-0.054	-0.041	-0.034	-0.028	
E=2	-0.054	-0.046	-0.041	-0.036	
E=halfmax	-0.054	-0.046	-0.040	-0.037	

Table 5.4: Varying E for exact-match training on Switchboard (68h train), 12 mixcomp.

The EBW update is controlled by a smoothing constant E which controls the speed of optimisation; larger E means slower training.

Tables 5.4, 5.5 and 5.6 and 5.7 show the effect of setting E to various different values on the three corpora. Considering firstly the difference between E=1 and E=2, the faster update (E=1) works better than E=2 on NAB and BN but the slower update (E=2) works better on the 68h Switchboard task. None of these results are significant; see the lower part of Table 5.1 for the relevant intervals. No firm overall conclusion can be drawn regarding which of these two values gives better results; however, further experiments finding that the faster update works better under a variety of conditions for NAB can be seen later in Table 5.13. Although these results are not significant, they are consistent with the idea that more smoothing (higher E) is needed on the more confusable tasks (Switchboard and BN). This is not surprising when one considers the way the update equations work: any non-confusable data in the training set acts to slow down optimisation by increasing the Gaussian-specific value D_{jm}, which is

	Training Iteration								
	0	1	2	3	4	5	6	7	8
	%WER on eval97sub								
$E = 2.0$	44.4	42.8	41.9	41.7	41.3	41.3	41.2	41.3	41.4
E=halfmax (1.7 . . . 7.5)	44.4	42.6	41.9	41.6	41.4	41.2	41.2	41.3	41.2
	%WER on eval98								
$E = 2.0$	45.6	44.1	43.1	42.5	42.3	41.9	41.7	41.6	41.8
E=halfmax (1.7 . . . 7.5)	45.6	44.0	42.9	42.5	42.2	42.1	42.0	41.9	41.8
	Training Iteration								
	0		2		4		6		
	MMI criterion / # frames								
$E = 2.0$	-0.0602		-0.0496		-0.0461		-0.0408		
E=halfmax (1.7 . . . 7.5)	-0.0602		-0.0487		-0.0458		-0.0412		

Table 5.5: Varying E (exact-match training) on Switchboard, h5train00 (265h) training, 12 mixcomp.

proportional to the occupation count γ_{jm}^{den}.

As mentioned in Section 4.5.3, the value E=halfmax represents a way of setting E as follows: find the Gaussian-specific D values as twice the value for positive variance updates, and set E to half the maximum value of $D/\gamma_{jm}^{\text{den}}$, i.e, half the value it would have had to be to set D at that maximum value. E=halfmax is compared with E=2 in Tables 5.4 and 5.5, which show that although the MMI criterion is increased more slowly with E=halfmax, the final WER is generally slightly better. However this technique has not been used for further experiments because of concerns that it adds to the system complexity and may lack robustness because it sets E based on possibly atypical Gaussians. A similar effect may be obtained by setting E to a value linearly increasing with iteration, which is investigated in Section 8.13 in the context of MPE training.

The conclusion from these experiments is that a value of E in the range 1 to 2 is probably suitable; the appropriate value should be determined empirically based on recognition results on a development test set.

The optimal test set performance is generally reached between 2 and 8 iterations; typically training is continued for 4 iterations for MMI.

5.3.8 Lattice size

The lattices used for training MMI on the Switchboard 65h-trained (h5train00sub) system were further pruned to three different levels (Table 8.6) to investigate the effect of lattice size on MMI training. The pruning thresholds are given in natural log likelihood values relative to the most likely path. The original lattices

	Training Iteration				
	0	1	2	3	4
	Average %WER on csrnab1_{dev,eval}				
12-mix, E=0.5	9.57	9.24	9.07	9.05	9.08
12-mix, E=1	9.57	9.35	9.25	9.18	9.10
12-mix, E=2	9.57	9.48	9.31	9.26	9.27
12-mix, $E = 4$	9.57	9.57	9.48	9.40	9.30
1-mix, E=1	14.70	13.74	13.14	12.53	12.26
1-mix, E=2	14.70	14.30	13.69	13.23	13.16
	MMI criterion / # frames				
12-mix, $E = 0.5$	-0.0128	-0.0083	-0.0061	-0.0048	
12-mix, E=1	-0.0128	-0.0099	-0.0082	-0.0069	
12-mix, E=2	-0.0128	-0.0111	-0.0099	-0.0089	
12-mix, $E = 4$	-0.0128	-0.0119	-0.0111	-0.0104	
1-mix, E=1	-0.0126	-0.0192	-0.0174	-0.0161	
1-mix, E=2	-0.0126	-0.0203	-0.0191	-0.0182	

Table 5.6: Varying E for exact-match training on NAB

	avg %WER on bndev96 partitioned test set					ins/del ratio
	Training Iteration					
Training setup	0	1	2	3	4	4
E=1	29.6	28.4	28.0	27.9	27.9	0.95
E=2	29.6	28.9	28.3	28.1	28.0	0.98

Table 5.7: Varying smoothing constant E on Broadcast News: 12 mixcomp HMM set

(Baseline) were generated by recognising the data with a bigram language model, and then phone-marked with a unigram language model. They were then pruned with various thresholds, as shown in Figure 5.8. The lattice depths given are the average number of arcs crossing each time frame. The lattice generation tools do not generate the alternate time alignments for each word unless a particular time alignment is on the best Viterbi path for some sentence. This means that the lattice may be more complete than lattices of a similar depth generated using a tool that prints out all likely time alignments of each sentence.

Table 5.9 shows the effect of MMI training with these different sets of lattices. As can be seen, use of the pruned lattices degrades results but with the most mildly pruned (Pruned-1) lattices the difference is not apparent until the third iteration. It can be seen that the MMI criterion reaches a higher value with the

Lattice Ident	Lattice Depth	Prune Thresh
Baseline	124	-
Pruned-1	34	100
Pruned-2	8.4	50
Pruned-3	4.8	25

Table 5.8: Characteristics of different denominator lattices used for h5train00sub training.

Lattice Ident	Iteration 0	1	2	3	4
	%WER on eval97sub				
Baseline	46.0	43.8	44.1	43.9	44.3
Pruned-1	46.0	43.8	44.1	44.1	44.6
Pruned-2	46.0	44.1	43.9	44.6	45.5
Pruned-3	46.0	44.3	44.2	45.4	46.9
	%WER on eval98				
Baseline	46.6	44.9	44.2	44.3	44.4
Pruned-1	46.6	44.9	44.3	44.4	44.6
Pruned-2	46.6	44.9	44.3	44.9	45.8
Pruned-3	46.6	45.0	44.6	45.5	47.0
	MMI Criterion on pruned lattice				
Baseline	-0.0545	-0.0415	-0.0335	-0.0279	
Pruned-1	-0.0542	-0.0413	-0.0332	-0.0272	
Pruned-2	-0.0532	-0.0397	-0.0302	-0.0221	
Pruned-3	-0.0510	-0.0362	-0.0241	-0.0130	

Table 5.9: Lattice pruning for training on Switchboard, h5train00sub (65h) train, 12 mixcomp, Exact-match, E=1

Iteration	0	1	2	3	4	ins/del last iter
	%WER on eval97sub					
Unigram	46.0	44.5	43.7	43.9	43.8	0.22
Bigram	46.0	44.9	44.2	44.1	44.2	0.24
Zerogram	46.0	45.4				0.17
Zerogram, $ip = 2$	46.0	45.0	44.7	45.1	45.7	0.29
Unigram$^{0.5}$	46.0	44.9	44.6	45.0	45.4	0.12
Unigram$^{0.5}$, ip=0.625	46.0	44.8	44.3	44.4	44.6	0.20
Unigram$^{1.5}$	46.0	44.1	43.6	43.5	43.7	0.49
	%WER on eval98					
Unigram	46.6	45.4	44.7	44.4	44.3	0.28
Bigram	46.6	45.9	45.3	44.9	44.8	0.25
Zerogram	46.6	45.8				0.17
Zerogram, $ip = 2$	46.6	45.7	45.4	45.3	45.7	0.36
Unigram$^{0.5}$	46.6	45.6	45.0	45.1	45.2	0.17
Unigram$^{0.5}$, ip=0.625	46.6	45.6	45.0	44.6	44.6	0.24
Unigram$^{1.5}$	46.6	45.4	44.9	44.8	45.0	0.58

Table 5.10: Varying language model in training lattices, on Switchboard 65h (h5train00sub) training (Exact-match, E=2).

pruned lattices since there are fewer competing hypotheses (note that the MMI criterion reported is an approximation based on the hypotheses available in the lattice).

The conclusion from these experiments is that above a pruning threshold of 100, the difference in word error rate from lattice pruning is in the region of 0.2%, which is probably acceptable[1].

5.3.9 Lattice language model: zero-gram, unigram or bigram

This section investigates the effect of the language model used for discriminative training.

Some previous work has discussed the effect of the language models used in the "denominator HMM" used for MMI training. In [Normandin et al., 1994], it was found that MMI training without a language model degraded performance. It was suggested that the reason for this was that the language model acts like an

[1]Note: the version of the experiment of Table 5.9 reported in [Woodland & Povey, 2002] uses the wrong baseline recognition results (reporting results for E=2 not E=1), but the conclusions are unaffected.

	Training iteration					Ins/del
	0	2	4	6	8	ratio
	WER on eval98					
Bigram	45.6	43.8	42.8	42.1	41.8	0.24
Unigram	45.6	43.1	42.3	41.7	41.8	0.33
(MLE)	45.6					0.27

Table 5.11: Unigram vs. bigram training on Switchboard; h5train00 (265h) training, 16 mixcomp HMM set, E=2.

insertion penalty and the lack of it led to unrealistic sequences during training. In [Schluter et al., 1999], it was found that given a choice of zero-gram, unigram, bigram and trigram language models, a unigram language model worked best for MMI training irrespective of whether a unigram, bigram or trigram model was used for testing.

Notation and experimental conditions for language modeling experiments

In experiments reported below, bigram, unigram and scaled unigram language models are investigated for use in training. These language models were applied to a single set of lattices generated using a bigram language model (since it is time-consuming to regenerate the lattices). Testing was with a trigram LM in all cases. A scaled unigram, represented as e.g. unigram$^{0.5}$, indicates scaling both the language model log probability and the word insertion penalty by 0.5. This is in addition to any other scaling that might be performed, governed by the normal language model scale and by the general scale κ. Scaling by 0.0 leads to a zero-gram language model. Since this affects the insertion/deletion ratio on recognition, some scaled-down experiments use in addition a phone insertion penalty which is a log likelihood (not scaled by κ) inserted between phones to reduce the likelihood of longer transcriptions. This is used instead of a word insertion penalty to replace the language model's penalisation of long words (since they are generally less frequent). The phone insertion penalty is indicated for example as $ip = 2$ in tables, for an insertion penalty of 2. This corresponds to inserting a log probability of -2 between each pair of phones.

Effect of language model on Switchboard

Table 5.10 compares three kinds of language model on systems trained on the Switchboard h5train00sub (65h) training subset. The best of the language models for this system was a unigram LM.

	Amount of training data						
	1.125h	2.25h	4.5h	9h	18h	68h	265h
Change, ug vs. bg	+0.2	0.0	-0.5	+0.2	+0.0	-0.3	-0.6

Table 5.12: Unigram vs. bigram lattices for a small HMM set and varying training data on Switchboard, comparison of results from Table 8.8. Negative numbers mean unigram is better.

With a zerogram or scaled unigram LM, the test set results on the larger eval98 test set were worse than with unigram, even though the insertion/deletion ratio was adjusted by a phone insertion penalty to be about the same as baseline (unigram). This appears to be due to insertions of common words.

Table 5.11 compares unigram and bigram language models on the full h5train00 (265h) training set. There is no difference on the last iteration (8), although on previous iterations unigram training was significantly superior to bigram (the significance interval is about 0.4% for pairs of similar training setups such as these).

Table 5.12 summarises results which appear more fully in a different context, in Table 8.8 in Section 8.3. The results are for MMI training on Switchboard with a very widely varying amount of training data, using a small HMM set with 3088 states and 6 Gaussians per state (mixture components). The comparison is the absolute WER between unigram and bigram-trained HMM sets. In most cases unigram is better or equal; in two cases bigram gives slightly better results, although there does not appear to be a clear pattern to the difference. It was expected that bigram would become better with increasing amounts of training data, but this appears not to be the case.

Effect of language model on NAB

Table 5.13 shows that for training a typical system on the NAB corpus a unigram language model works better than bigram for a wide range of training conditions, but with an appropriate insertion penalty better results are gained with a zero-gram language model, and the best with a scaled-down unigram. However the best result (scaled-down unigram) is not significantly better than the most appropriate baseline, which is a unigram LM with E=1 and $\kappa = 1/15$.

Effect of language model on Broadcast News

Table 5.14 compares bigram, unigram and scaled-down unigram language models on BN, both with I-smoothing ($\tau = 50$) and without. These results do not show any clear advantage for any language model.

	avg %WER on csrnab1_{dev,eval} Iteration					relative	ins/del
	0	1	2	3	4	%impr	(it 4)
				Bigram			
bg, $\kappa = 1$, E=1	9.57	9.61	9.54	9.47	9.42	1.6%	0.96
bg, $\kappa = 1$, E=2	9.57	9.56	9.58	9.55	9.54	3.1%	0.95
bg, $\kappa = 1/15$, E=1	9.57	9.63	9.55	9.44	9.42	1.6%	0.96
bg, $\kappa = 1/15$, E=2	9.57	9.57	9.59	9.58	9.53	0.4%	0.95
				Unigram			
ug, $\kappa = 1$, E=1	9.57	9.33	9.36	9.35	9.30	2.8%	0.96
ug, $\kappa = 1$, E=2	9.57	9.55	9.41	9.33	9.34	2.4%	0.99
ug, $\kappa = 1/15$, E=1	9.57	9.35	9.25	9.19	9.10	4.9%	0.89
ug, $\kappa = 1/15$, E=2	9.57	9.48	9.31	9.26	9.28	3.0%	0.94
				Scaled/zerogram			
zg, $\kappa = 1/15$, E=1, ip=1.25	9.57	9.40	9.23	9.13	9.07	5.2%	0.89
zg, $\kappa = 1/15$, E=1, ip=1.75	9.57	9.34	9.31	9.17	9.18	4.1%	1.10
zg, $\kappa = 1/15$, E=1, ip=2	9.57	9.32	9.31	9.21	9.12	4.7%	1.17
ug$^{0.5}$,$\kappa = 1/15$,E=1, ip=0.625	9.57	9.37	9.31	9.06	9.02	5.7%	0.88

Table 5.13: Varying language model in training lattices, on NAB: zero/uni/bigram, $E = 1, 2$, scaled/not scaled, 12 mixcomp

	avg %WER on bndev96 partitioned test set						ins/del ratio
	Iteration						
Training setup	0	1	2	3	4	5	5
bg,	29.6	29.0	28.4	28.1	28.1	27.9	0.81
ug,	29.6	28.4	28.0	27.9	27.9	28.0	0.98
ug$^{0.5}$	29.6	28.7	28.1	27.9	27.8	27.6	0.85
ug, τ=50	29.6	28.4	27.9	27.8	27.7	27.5	1.04
ug$^{0.5}$, τ=50, ip=0.625	29.6	28.7	28.1	27.9	27.8	27.6	0.94

Table 5.14: Bigram vs Unigram vs Unigram$^{0.5}$, for MMI on Broadcast News, 12 mixcomp, with and without I-smoothing, E=1

Insertions of common words

Comparison between the test-set recognition output from a model trained with a zero-gram language model with the most appropriate insertion penalty, and one trained with a unigram language model shows an interesting difference in frequency of the word "that". On the Switchboard eval97sub test set, the zero-gram trained output has about twice as many instances of the word "that" as the unigram-trained output. A smaller increase in "that" was seen on the NAB recognition output. MMI training with a zero-gram language model increases the likelihood of the models of "that" because a zero-gram LM with per-phone insertion penalty gives it a much lower likelihood than a regular language model ("that" being a very common word), and the acoustic model tries to compensate.

Summary of effects of LM during training

In summary, the results are in agreement with the conclusions of previous authors in that a unigram language model is better than bigram or no language model. With the use of a per-phone insertion penalty, better results can sometimes be got with a zero-gram (no language model) or scaled-down unigram, but this is not consistent across test sets and is associated with an increase in insertions of common words. Note that these experiments applied various language models to lattices generated with a bigram language model, but experiments with MPE training seem to indicate that using a unigram language model right from the start is the best approach (see Table 8.26).

5.3.10 Probability scale

The probability scale κ, like the language model, has an effect on the test set insertion/deletion ratio. When κ is decreased from its default value, an insertion penalty has to be used to prevent an increase in word deletions when testing.

Table 5.15 shows the results of varying the probability scale κ on Switchboard. The default value of κ is the inverse of the normal language model scale, 1/12 in the case of Switchboard. This appears to be close to the optimal value; no improvement was gained by varying κ from the default, even when the training insertion penalty is varied so as to keep the testing insertion/deletion ratio constant.

Table 5.16 shows the effect of varying scale κ on on NAB. Again, if ip is set to 0 the optimal scale seems to be close to the default value of 1/15. But when the insertion penalty ip is varied to keep the insertion/deletion ratio roughly constant, a smaller scale can give better results. For example, at a scale of κ =1/60, an insertion penalty of 0.5 and I-smoothing at τ=50, the error rate is 8.75; for comparison, with τ=50 and κ =1/15 the error rate is 9.19.

	ins pen	Test %WER on eval98 (and ins/del ratio) Probability scale κ 1/24	1/12	1/6
MLE (no scale)			46.6 (0.25)	
MMI, E=1, (4 its)	0.0	44.9 (0.18)	44.4 (0.29)	45.0 (0.35)
	0.25	44.6 (0.31)		
MPE, E=2, τ = 50 (8 its)	0.0	43.3 (0.26)	43.1 (0.34)	44.5 (0.39)
	0.2	43.2 (0.36)		
	0.4	43.7 (0.48)		

(a) h5train00sub, 68h

	ins pen	Test %WER on eval98 (and ins/del ratio) Probability scale κ 1/12	1/6
MLE (no scale)		45.6 (0.27)	
MPE, E=2, τ = 100 (4 its)	0.0	41.6 (0.28)	42.7 (0.33)

(b) h5train00, 265h

Table 5.15: Varying κ (probability scale) and ip (insertion penalty), on Switchboard: unigram lattices.

Training setup	# its	ip	Test %WER on on csrnab1_{dev,eval} (ins/del ratio) Probability scale κ (default=1/15) 1/120	1/60	1/30	1/15	1/7.5
MLE (no scale)						9.57 (0.95)	
MMI:							
E=1	4	0.0		9.73 (0.43)	9.03(0.67)	9.10 (0.89)	9.19(0.93)
		0.3			8.87(0.86)		
		0.4			8.91(0.99)		
		0.5		9.10 (0.74)			
		0.75		9.08 (0.99)			
E=2	4	0.75		8.92 (1.04)			
E=1, τ=50	4	0.0				9.19 (0.91)	
		0.3			8.83 (0.92)		
		0.5		8.75 (0.84)			
	3	0.7	8.73 (0.80)				
	4	0.7	9.03 (0.78)				
E=2, τ=50	4	0.5		8.99 (0.88)			

Table 5.16: Varying κ (probability scale) and ip (insertion penalty), on NAB; unigram lattices.

	avg %WER on bndev96 partitioned test set					ins/del ratio
	Iteration					
Training setup	0	1	2	3	4	4
$\kappa = 1/14$, E=1, τ=0	29.6	28.4	28.0	27.9	27.9	0.96
$\kappa = 1/14$, E=2, τ=0	29.6	28.9	28.3	28.1	28.0	0.98
$\kappa = 1/14$, E=1, τ=50	29.6	28.4	27.9	27.8	27.7	0.98
$\kappa = 1/56$, E=2, τ=50, ip=0.5					27.3	0.86

Table 5.17: Varying κ (probability scale) and ip (insertion penalty), on Broadcast News Channel 1: MMI, 12 mixcomp, unigram lattices.

Table 5.17 is an experiment on Broadcast News which compares some baseline setups for MMI training with the optimal setup from NAB (an extra factor of 1/4 for the scale κ and a phone insertion penalty of 0.5). This gives an improvement of 0.4% absolute compared with the best baseline; the 95% significance interval is in the region of 0.4% (Table 5.1).

An experiment with the same setup on Resource Management failed to find an improvement from this more severe probability scaling: the improvement was the same with either baseline or more strongly scaled training with WER reduced in both cases from 4.1% (MLE baseline) to 3.9% on all four test sets together.

Summary of results from probability scaling

In summary, although the baseline approach of setting κ to the inverse of the recognition language-model scale is a simple way to get good results, two out of the four corpora (NAB and BN) showed an improvement from a smaller probability scale (1/4 the normal scale) coupled with a phone insertion penalty used during training to prevent changes in the insertion/deletion ratio. But there was no improvement on the Switchboard and Resource Management tasks.

Why does κ affect the insertion/deletion ratio?

As can be seen for example from Table 5.16, varying the language model scale has an effect on the insertion/deletion ratio. This appears to be related to changes in specific HMMs causing changes in frequencies of common short words, for reasons set out as follows.

The change in insertion/deletion ratio appears not to be related to changes in the transition likelihoods: when the transition values of the trained models on the $\kappa = 1/7.5$ and $\kappa = 1/30$ experiments from Table 5.16 were set back to their untrained values, the insertion/deletion ratio changed by less than 0.01 in the

	avg %WER on csrnab1_{dev,eval}						rel	
	Iteration						% impr	ins/del
MMI (E=1)	0	1	2	3	4	5	(it 4)	(it 4)
κ=1/15	9.57	9.35	9.25	9.19	9.10		4.9%	0.89
κ=1/15, new lats	9.57	9.26	9.15	9.05	9.06	9.06	5.3%	0.72
κ=1/60, ip=0.5, i=50	9.57	9.18	9.01	8.92	8.75	8.68	8.6%	0.84
κ=1/60, ip=0.5, i=50, new lats	9.57	9.26	9.00	8.93	8.81	8.75	7.9%	0.86
MPE (E=2)	0	2	4	6	8	10	(it 8)	(it 8)
κ=1/15	9.57	9.06	8.96	8.98	9.00	9.00	6.0%	1.01
κ=1/15, new lats	9.57	9.13	8.96	8.98	9.02	9.00	5.8%	1.01

Table 5.18: Lattice regeneration vs. original lattices, on NAB, 12 mix-comps, unigram lattices.

direction of the baseline ($\kappa = 1/15$) value. With a low acoustic scale ($\kappa = 1/30$) the trained variances were slightly smaller than with an acoustic scale closer to 1 ($\kappa = 1/7.5$). 90% of $\kappa = 1/7.5$ variances were within $-0.180 \ldots 0.177$ of their MLE-trained value, and for $\kappa = 1/30$ this was $-0.232 \ldots 0.179$. However this would be expected to produce more, rather than fewer, insertions in the $\kappa = 1/30$ system since it would accentuate the range of acoustic likelihoods and be equivalent to a scaling-down of the language model. It seems likely by elimination that the effect is due to changes in HMMs used in specific common words, which appear to change in frequency when the scale κ is changed.

5.3.11 Lattice regeneration

Experiments on the NAB and Switchboard corpora gave mixed results on the effect of regenerating lattices after some initial iterations of MMI training. Generation of lattices by recognition of the training data with MMI-trained models in a NAB experiment, and doing MMI training from scratch on a composite of the original and new lattices, made only a slight improvement (0.04% absolute) to baseline MMI results and degraded two other discriminative criteria (I-smoothed MMI and MPE); none of these results are significant.

A 0.3% absolute improvement was found when lattice time boundaries were re-aligned after some iterations of MMI training for a Switchboard task with 265h of data. The details of these experiments are given as follows.

Lattice regeneration and merging on NAB

Table 5.18 shows the effect of regenerating the lattices used for MMI training after 4 iterations of baseline MMI training (as in the top row of the table) and then

	Iteration			
MMI (E=1)	0	1	2	3
κ=1/15	-0.0128	-0.0099	-0.0082	-0.0069
κ=1/15, new lats	-0.0124	-0.0108	-0.0096	-0.0088
κ=1/60, $ip = 0.5$, $i = 50$	-0.0134	-0.0124	-0.0116	-0.0108
κ=1/60, $ip = 0.5$, $i = 50$, new lats	-0.0139	-0.0130	-0.0123	-0.0117
MPE (E=2)	0	2	4	6
κ=1/15	0.917	0.946	0.953	0.957
κ=1/15, new lats	0.915	0.942	0.949	0.953

Table 5.19: Lattice regeneration vs. original lattices, on NAB: Criterion optimisation (critera calculated using training lattices).

combining these with the original lattices to produce composite lattices containing all pairs of paths. Experiments marked "new lats" used these composite lattices to discriminatively train HMMs starting from an ML-trained system. A slight improvement of 0.04% absolute is obtained with the baseline MMI setup (top two rows) but this is not matched with the other two configurations tested, which are respectively I-smoothed MMI (see Section 4.3) with a different probability scale and Minimum Phone Error (MPE). It might be that the standard MMI showed more improvement because the lattices were specifically generated from it. In any case, the improvement even in the case of standard MMI is relatively small.

Table 5.19 shows the effect of the combined lattices on optimisation of the MMI and MPE criteria. In all cases the criterion is optimised more slowly with the combined lattices. This is what is expected since a greater number of confusable sentences are more difficult to "learn".

Lattice re-alignment on Switchboard

In an experiment where MMI training (E=halfmax) was performed on the Switchboard h5train00 (265h) training set, there was an improvement of 0.3% on eval98 (from 41.8 to 41.5, relative to a baseline of 45.6) when instead of training for 8 iterations using the original lattices, the lattice phone boundaries were re-aligned halfway through training[2].

In summary, lattice regeneration or re-alignment can give small improvements of up to 0.3% absolute with MMI training but this has not generally been done since it is very computationally expensive. In general it is sufficient to use a single set of lattices, but if computation time is not a problem the lattices can be regenerated halfway through training and this may improve results.

[2]The results reported in [Woodland & Povey, 2002] overestimated this improvement through using a baseline from only 4 iterations of training

5.4 Conclusions

This chapter has explained the framework for lattice-based MMI training and reported experiments investigating various aspects of the training procedure.

The conclusions from the experimental results are:

- MMI training can be effective for competitive LVCSR systems.
- The exact-match approach to probability scaling should be used.
- Setting the constant E to 2 (or possibly 1), and running training for 4 to 8 iterations, is suggested; experiments should be done on each corpus to find the best regime.
- Lattices generated with a pruning threshold larger than about 100 are sufficient; regeneration of the lattices after some iterations of training may give a small improvement but it not necessary.
- A unigram language model should be applied to the lattices, or possibly a scaled-down unigram if a phone insertion penalty is added.
- A probability scale equal to the inverse of the normal language model scale should be used. On some corpora there may be an advantage in using a smaller value, e.g. 1/4 this value, and introducing an (unscaled) phone insertion penalty of about 0.5 during training to prevent this changing the insertion/deletion ratio.

Chapter 6

Variants of the EB Update Formulae

6.1 Introduction

This chapter investigates variants of the Extended Baum-Welch (EB) update equations. Alternative update equations, based on similar principles but using slightly different assumptions, are derived and investigated experimentally.

Section 6.2 derives a number of variants of the EB update for the Gaussian parameters; Section 6.3 derives a new update for weights and transitions; Section 6.4 reviews previous work on the topic and gives experimental results, and Section 6.5 gives conclusions.

6.2 Variants of the EB update for Gaussians

This section derives various alternatives to the EB update for Gaussians.

Section 6.2.1 presents the idea of using a linear function of the HMM parameters for the denominator auxiliary function. Section 6.2.2 derives update equations from this linear-denominator approach. Section 6.2.3 gives a variant to the linear-denominator approach, based on Newton's method of function optimisation.

6.2.1 Linear denominator auxiliary functions

The MMI objective function can be expressed as

$$\mathcal{F}_{\mathrm{MMI}}(\lambda) = \log p_\lambda^\kappa(\mathcal{O}|\mathcal{M}_{\mathrm{num}}) - \log p_\lambda^\kappa(\mathcal{O}|\mathcal{M}_{\mathrm{den}}, \lambda), \tag{6.1}$$

where $\mathcal{M}_{\mathrm{num}}$ and $\mathcal{M}_{\mathrm{den}}$ are the models corresponding to the correct word sequence and all possible word sequences respectively, κ is a probability scale used in

calculating model likelihoods, and λ is the HMM parameters; for simplicity, just one training file is considered in this notation. This can equivalently be written as

$$\mathcal{F}_{\mathrm{MMI}}(\lambda) = \mathcal{F}^{\mathrm{num}}_{\mathrm{MMI}}(\lambda) - \mathcal{F}^{\mathrm{den}}_{\mathrm{MMI}}(\lambda), \tag{6.2}$$

to make the application of auxiliary functions clearer. An auxiliary function $\mathcal{G}_{\mathrm{MMI}}(\lambda, \lambda')$ is sought which will ideally be the same as $\mathcal{F}_{\mathrm{MMI}}(\lambda)$ at the value $\lambda = \lambda'$ but less everywhere else, i.e. a strong-sense auxiliary function around λ' (see Equation 4.1 for the definition of a strong-sense auxiliary function). The resulting auxiliary function will be written as follows:

$$\mathcal{G}_{\mathrm{MMI}}(\lambda, \lambda') = \mathcal{G}^{\mathrm{num}}_{\mathrm{MMI}}(\lambda, \lambda') - \mathcal{G}^{\mathrm{den}}_{\mathrm{MMI}}(\lambda, \lambda'). \tag{6.3}$$

In obtaining an auxiliary function for the expression, the two terms can be handled separately. For $\mathcal{G}^{\mathrm{num}}_{\mathrm{MMI}}(\lambda, \lambda')$ the auxiliary function used in normal Baum-Welch updates for ML training can be used; this is a strong-sense auxiliary function for the first term $\mathcal{F}^{\mathrm{num}}_{\mathrm{MMI}}(\lambda)$ (see Section 4.2.3). The case of $\mathcal{G}^{\mathrm{den}}_{\mathrm{MMI}}(\lambda, \lambda')$ is more difficult because its likelihood function is negated. Ideally a function is needed which is the same $\log p(\mathcal{O}|\mathcal{M}_{\mathrm{den}}, \lambda)$ at $\lambda = \lambda'$ but more, not less, everywhere else, i.e. is a strong-sense auxiliary function for $\mathcal{F}^{\mathrm{den}}_{\mathrm{MMI}}(\lambda)$. As discussed in Section 4.5.4, this seems to be impossible without accumulating other kinds of statistics; but the aim is to construct a weak-sense auxiliary function which will work in practice. to a strong-sense auxiliary function.

An obvious choice for $\mathcal{G}^{\mathrm{den}}_{\mathrm{MMI}}(\lambda, \lambda')$ is a simple linear function of all the means and variances, with a gradient corresponding to the current differential of the denominator likelihood function, i.e a gradient equal to $\frac{\partial}{\partial \lambda} \log p(\mathcal{O}|\mathcal{M}_{\mathrm{den}}, \lambda)|_{\lambda=\lambda'}$. There is good reason to believe that this should usually be more than the real likelihood function $\log p^{\kappa}_{\lambda}(\mathcal{O}|\mathcal{M}_{\mathrm{den}}, \lambda)$ where $\lambda \neq \lambda'$. It is clear that the function $\log p^{\kappa}_{\lambda}(\mathcal{O}|\mathcal{M}_{\mathrm{den}})$ is a roughly concave function[1] of λ, because a log Gaussian likelihood is concave. There is no way one can be sure that $p(\mathcal{O}|\mathcal{M}_{\mathrm{den}}, \lambda)$ might not have separate peaks with valleys (regions of concavity) in between, and in fact it seems certain for multiple Gaussian systems at least that this would be the case (consider moving the parameters of two Gaussians in a mixture slowly and simultaneously towards the parameters of the other Gaussian). However, $p(\mathcal{O}|\mathcal{M}_{\mathrm{den}}, \lambda)$ must certainly have more concavity than convexity [2] and it can expected that if it is assumed to be concave and a flat function is used for $\mathcal{G}^{\mathrm{den}}_{\mathrm{MMI}}(\lambda, \lambda')$ the update formulae will generally work. Note that although this concavity assumption is not strictly true and the resulting auxiliary function will not be a strong-sense

[1] Note that, unintuitively, concave functions are functions like $-x^2$ that have a negative second differential everywhere.

[2] This could probably be defined more precisely but it would not lead to a proof of the equations derived here so the issue will not be pursued.

auxiliary function for $\mathcal{F}_{\text{MMI}}(\lambda)$ around λ', it will still be a weak-sense auxiliary function so convergence will be to the correct point.

A linear auxiliary function depends on the representation of parameters (e.g, whether the variance is represented as $1/\sigma^2$ or $\log\sigma^2$). A linear function with one representation will not be linear with the other. Concavity and convexity are also not invariant to these different representations. Various approaches are examined here.

Simplicity of linear-denominator approach

An advantage of the approach of using a linear function for the denominator likelihood is that it is a way of deriving a (weak-sense) auxiliary function for the MMI objective function that can be expressed as a simple rule applicable to other situations, saying: for any terms in the objective function for which strong-sense auxiliary functions exist, use a strong-sense auxiliary function; for any other terms, use a linear function. As will become clear in Section 6.2.2 this leads to something very similar to the EB update where $E = 1$, without having to arbitrarily devise a formula for the Gaussian-specific constant D_{jm}.

6.2.2 Optimisation of linear denominator auxiliary function

If a linear auxiliary function for the denominator model is written as $G^{\text{den}}_{\text{linear}}(\lambda, \lambda')$, and the normal ML auxiliary function as applied to the denominator as $G^{\text{den}}_{\text{MLE}}(\lambda, \lambda')$, a combination of the two kinds of auxiliary function can be used as a more general choice for $\mathcal{G}^{\text{den}}_{\text{MMI}}(\lambda, \lambda')$, using a constant $E \geq 1$:

$$\mathcal{G}^{\text{den}}_{\text{MMI}}(\lambda, \lambda') = (1 - E)G^{\text{den}}_{\text{MLE}}(\lambda, \lambda') + EG^{\text{den}}_{\text{linear}}(\lambda, \lambda'). \tag{6.4}$$

When $E = 1$ this is a linear denominator auxiliary function, but higher values of E are permitted for extra smoothing and slower updates. Note that the whole term $\mathcal{G}^{\text{den}}_{\text{MMI}}(\lambda, \lambda')$ is negated in the MMI objective function so the effect is to give $G^{\text{den}}_{\text{MLE}}(\lambda, \lambda')$, a concave function, a zero or positive coefficient in the final objective function. Note that the term $G^{\text{den}}_{\text{linear}}(\lambda, \lambda')$ is not uniquely defined as it depends on which form of the variances is defined as the parameter: i.e, $\log\sigma^2$, $\frac{1}{\sigma^2}$, etc. $G^{\text{den}}_{\text{linear}}(\lambda, \lambda')$ is the linear combination of the parameters λ such that its differential with respect to the parameter set λ is the same of the normal ML auxiliary function $G^{\text{den}}_{\text{MLE}}(\lambda, \lambda')$ at the point $\lambda = \lambda'$ used to obtain the statistics. As will be seen, expressing the variance as $\frac{1}{\sigma}$ and ensuring that $E \geq 1$ leads to an update formula that will always give positive variances, which seems a desirable feature for an update formula to have.

Defining $\bar{y}^{\text{num}}_{jm}$ as the average data value $\theta^{\text{num}}_{jm}(\mathcal{O})/\gamma_{jm}$ for the numerator, and $\bar{y^2}^{\text{num}}_{jm}$ as the average squared data value $\theta^{\text{num}}_{jm}(\mathcal{O}^2)/\gamma_{jm}$, and likewise for the denominator

model, the numerator part of objective function for a single dimension and a single Gaussian j, m is:

$$\mathcal{G}^{\text{num}}_{\text{MMI}}(\lambda, \lambda') = -0.5\gamma^{\text{num}}_{jm} \left[\log(\sigma^2) + \frac{\bar{y^2}^{\text{num}}_{jm} - 2\bar{y}^{\text{num}}_{jm}\mu_{jm} + \mu^2_{jm}}{\sigma^2_{jm}} \right] \tag{6.5}$$

Denominator auxiliary function, linear in $\log(\sigma^2)$

If we choose to make the linear part of $\mathcal{G}^{\text{den}}_{\text{MMI}}(\lambda, \lambda')$ linear in $\log(\sigma^2)$, then defining $S^{\text{num}}(\mu)$ as the variance of the numerator statistics around μ, i.e $\bar{y^2}^{\text{num}} - 2\bar{y}^{\text{num}}\mu + \mu^2$, and likewise for the denominator, the denominator part of the auxiliary function becomes:

$$\begin{aligned} \mathcal{G}^{\text{den}}_{\text{MMI}}(\lambda, \lambda') = & -0.5(1-E)\gamma^{\text{den}}_{jm} \left[\log(\sigma^2_{jm}) + \frac{S^{\text{den}}(\mu)}{\sigma^2_{jm}} \right] \\ & - 0.5E\gamma^{\text{den}}_{jm} \left[\log(\sigma^2_{jm}) \left(1 - \frac{S^{\text{den}}(\mu_{\text{orig}})}{\sigma^2_{j,m,\text{orig}}} \right) + \mu_{jm} \frac{2(\mu_{j,m,\text{orig}} - \bar{y}^{\text{den}}_{jm})}{\sigma^2_{j,m,\text{orig}}} \right]. \end{aligned} \tag{6.6}$$

where $\sigma^2_{j,m,\text{orig}}$ and σ^2_{jm} are the original and updated variances; and likewise for the means. The first term corresponds to the "EM-like" part of the denominator auxiliary function $(1-E)G^{\text{den}}_{\text{MLE}}(\lambda, \lambda')$, and the second term is the linear part of the auxiliary function $EG^{\text{den}}_{\text{linear}}(\lambda, \lambda')$; it is linear in the updated parameters.

Denominator auxiliary function, linear in $1/\sigma^2$

If we choose to make the linear part of $\mathcal{G}^{\text{den}}_{\text{MMI}}(\lambda, \lambda')$ linear in $\frac{1}{\sigma^2}$ (which seems the other obvious choice given the form of the Gaussian likelihood function) then the denominator part becomes:

$$\begin{aligned} \mathcal{G}^{\text{den}}_{\text{MMI}}(\lambda, \lambda') = & -0.5(1-E)\gamma^{\text{den}}_{jm} \left[\log(\sigma^2_{jm}) + \frac{S^{\text{den}}(\mu)}{\sigma^2_{jm}} \right] \\ & - 0.5E\gamma^{\text{den}}_{jm} \left[\frac{1}{\sigma^2_{jm}} (S^{\text{den}}(\mu_{\text{orig}}) - \sigma^2_{j,m,\text{orig}}) + \mu_{jm} \frac{2(\mu_{j,m,\text{orig}} - \bar{y}^{\text{den}}_{jm})}{\sigma^2_{j,m,\text{orig}}} \right]. \end{aligned} \tag{6.7}$$

Denominator auxiliary function, linear in $1/\sigma$

Neither of the above two approaches is ideal, the main problem being that the auxiliary function used for MLE (and for the numerator part of MMI) becomes linear at one end when expressed either in terms of $\frac{1}{\sigma^2}$ or $\log \sigma^2$, and this can

lead to negative updates. If expressed in terms of $\frac{1}{\sigma}$ the ML part of the auxiliary function is concave everywhere, which is more desirable. The denominator part then becomes:

$$\mathcal{G}^{\text{den}}_{\text{MMI}}(\lambda,\lambda') = -0.5(1-E)\gamma^{\text{den}}_{jm}\left[\log(\sigma^2_{jm}) + \frac{\bar{y^2}^{\text{den}}_{jm} - 2\bar{y}^{\text{den}}_{jm}\mu_{jm} + \mu^2_{jm}}{\sigma^2_{jm}}\right] - 0.5E\gamma^{\text{den}}_{jm}\left[2/\sigma_{jm}\frac{S^{\text{den}}(\mu_{\text{orig}}) - \sigma^2_{j,m,\text{orig}}}{\sigma_{j,m,\text{orig}}} + \mu_{jm}\frac{2(\mu_{j,m,\text{orig}} - \bar{y}^{\text{den}}_{jm})}{\sigma^2_{j,m,\text{orig}}}\right]. \tag{6.8}$$

Update for mean

The update equation for the mean is as follows, irrespective of the representation of the variance:

$$\hat{\mu} = \mu_{\text{orig}} + \frac{\gamma^{\text{num}}\frac{\bar{y}^{\text{num}}-\mu_{\text{orig}}}{\hat{\sigma}^2} + (E-1)\gamma^{\text{den}}\frac{\bar{y}^{\text{den}}-\mu_{\text{orig}}}{\hat{\sigma}^2} - E\gamma^{\text{den}}\frac{\bar{y}^{\text{den}}-\mu_{\text{orig}}}{\sigma^2_{\text{orig}}}}{\frac{\gamma^{\text{num}}+(E-1)\gamma^{\text{den}}}{\hat{\sigma}^2}} \tag{6.9}$$

where $\hat{\sigma}^2_{\text{orig}}$ and σ^2_{orig} are the original and updated variances.

Updates for variance

There are alternative updates for the variances. If the auxiliary function in Equation 6.6 (linear in $\log\sigma^2$) is used the variance update becomes as follows:

$$\hat{\sigma^2} = \frac{\gamma^{\text{num}}S^{\text{num}}(\hat{\mu}) + (E-1)\gamma^{\text{den}}S^{\text{den}}(\hat{\mu})}{\gamma^{\text{num}} + (E-1)\gamma^{\text{den}} - E\gamma^{\text{den}}(1 - \frac{S^{\text{den}}(\mu_{\text{orig}})}{\sigma^2_{j,m,\text{orig}}})} \tag{6.10}$$

If the auxiliary function in Equation 6.7 (linear in $\frac{1}{\sigma^2}$) is used,

$$\hat{\sigma^2} = \frac{\gamma^{\text{num}}S^{\text{num}}(\hat{\mu}) + (E-1)\gamma^{\text{den}}S^{\text{den}}(\hat{\mu}) - E(S_{\text{den}}(\mu_{\text{orig}}) - \sigma^2_{j,m,\text{orig}})}{\gamma^{\text{num}} + (E-1)\gamma^{\text{den}}} \tag{6.11}$$

If the auxiliary function in Equation 6.8 (linear in $\frac{1}{\sigma}$) is used,

$$\begin{aligned} a &= (\gamma^{\text{num}} + (E-1)\gamma^{\text{den}}) \\ b &= E\gamma^{\text{den}}\frac{S^{\text{den}}(\mu_{\text{orig}}) - \sigma^2_{j,m,\text{orig}}}{\sigma_{j,m,\text{orig}}} \\ c &= -(\gamma^{\text{num}}S^{\text{num}}(\hat{\mu}) + (E-1)\gamma^{\text{den}}S^{\text{den}}(\hat{\mu})) \end{aligned}$$

and the updated σ is given by

$$\hat{\sigma} = \frac{-b + \sqrt{(}b^2 + 4ac)}{2a}. \tag{6.12}$$

In all of these equations $\hat{\mu}$ and $\hat{\sigma}$ are interdependent and must be repeatedly updated until they converge. Out of the three equations the last one (Equation 6.12) is the only one that can be guaranteed to result in a positive variance (for $E \geq 1$). This stems from the fact that out of all the simple representations of the variance $(\sigma^2, \sigma, 1/\sigma^2, \log\sigma, 1/\sigma)$ the only one in which the Gaussian likelihood function is concave everywhere in the allowed region (and does not approach linearity at either extreme) is $1/\sigma$. This means that if $\frac{1}{\sigma}$ is chosen to represent the parameter, the gradient of the numerator part of the Gaussian likelihood function will dominate the linear denominator part for extreme values of $1/\sigma$, so the auxiliary function is bound to have a maximum within the allowed range. Experiments are only performed with this last version of the update equations.

6.2.3 Newton's method for optimisation of Gaussians

As an alternative way of deriving updates for the linear-denominator auxiliary function of Equation 6.4, a method based on Newton's method was devised. Rather than find the maximum of the auxiliary function, Newton's method is used to find an update based on the first and second differentials of the auxiliary function w.r.t. each parameter. Off-diagonal elements in the Hessian are ignored. This differs from a gradient descent approach using estimates of the second differential such as Quick-prop in that the second differential is obtained from the auxiliary function, which is not an estimate of the second differential of the true objective function but will in general be more negative.

Differentiation of $\mathcal{G}_{\mathrm{MMI}}(\lambda, \lambda')$ as in Equation 6.4 gives:

$$\left.\frac{\partial \mathcal{G}}{\partial \mu}\right|_{\lambda=\lambda'} = \gamma^{\mathrm{num}} \frac{\bar{y}^{\mathrm{num}} - \mu_{\mathrm{orig}}}{\sigma^2_{\mathrm{orig}}} - \gamma^{\mathrm{den}} \frac{\bar{y}^{\mathrm{den}} - \mu_{\mathrm{orig}}}{\sigma^2_{\mathrm{orig}}} \tag{6.13}$$

$$\left.\frac{\partial^2 \mathcal{G}}{\partial \mu^2}\right|_{\lambda=\lambda'} = -(\gamma^{\mathrm{num}} + (E-1)\gamma^{\mathrm{den}}) \frac{1}{\sigma^2_{\mathrm{orig}}}. \tag{6.14}$$

Denoting $\log\sigma^2$ as L,

$$\left.\frac{\partial \mathcal{G}}{\partial L}\right|_{\lambda=\lambda'} = 0.5\gamma^{\mathrm{num}} \left(\frac{S^{\mathrm{num}}(\mu_{\mathrm{orig}})}{\sigma^2_{\mathrm{orig}}} - 1 \right) - 0.5\gamma^{\mathrm{den}} \left(\frac{S^{\mathrm{den}}(\mu_{\mathrm{orig}})}{\sigma^2_{\mathrm{orig}}} - 1 \right) \tag{6.15}$$

$$\left.\frac{\partial^2 \mathcal{G}}{\partial L^2}\right|_{\lambda=\lambda'} = -0.5 \left(\gamma^{\mathrm{num}} \frac{S^{\mathrm{num}}(\mu_{\mathrm{orig}})}{\sigma^2_{\mathrm{orig}}} + (E-1)\gamma^{\mathrm{den}} \frac{S^{\mathrm{den}}(\mu_{\mathrm{orig}})}{\sigma^2_{\mathrm{orig}}} \right). \tag{6.16}$$

The update for the means is $\hat{\mu} = \mu - \frac{\partial \mathcal{G}/\partial \mu}{\partial^2 \mathcal{G}/\partial \mu^2}$, with a similar update for $L = \log\sigma^2$ except that the step size is limited to 0.5 (larger sizes of update may be inaccurate as the quadratic approximation to the auxiliary function is only valid locally).

%WER on eval98					
MLE	MMI iter 4	Weights as MLE	Transitions as MLE	Means as MLE	Variances as MLE
46.6	44.3	44.4	44.4	45.2	45.7

Table 6.1: Results for MMI-trained system on 65h h5train00sub with various kinds of parameters reset to original MLE values. Training is exact-match, $E = 2$.

6.3 Mixture weights and transition updates

This section discusses mixture weight and transition updates, including a derivation of an update rule which is a more general version of the one described in Section 4.4.

Section 6.3.1 discusses the relative importance of weight, transition and Gaussian parameter updates. Section 6.3.2 describes the standard EB update equations for weights and transitions. Section 6.3.3 derives a new weight/transition auxiliary function, which is a more general form of the one described in Section 4.4. Section 6.3.4 derives an update formula for the new auxiliary function.

6.3.1 Relative importance of weights and transitions.

Gaussian weights c_{jm} and transition likelihoods a_{ij} make relatively little difference to recognition results. Table 6.1 shows the effect of resetting various parameters of a MMI-trained system to the original MLE parameters; the weights and transitions are trained using the standard approach used in this thesis, which is the one described in Section 4.4. It is unclear whether resetting the means or variances to the original values will give a similar result to not training them at all, but these results do confirm that for the style of system used here, weights and transitions make very little difference to recognition results: they each seem to make only about 0.1% absolute difference to test-set WER.

6.3.2 Standard weight and transition updates

The baseline weight/transition update is discussed at more length in Section 4.5.2. This is the Extended Baum-Welch update coupled with an altered formula for the differential of the likelihood function with respect to the weights, as suggested in [Normandin & Morgera, 1991]. The update is:

$$\hat{c}_{jm} = \frac{c_{jm}(\partial/\partial c_{jm}\mathcal{F}_{\mathrm{MMI}}(\lambda) + C)}{\sum_m c_{jm}(\partial/\partial c_{jm}\mathcal{F}_{\mathrm{MMI}}(\lambda) + C)} \tag{6.17}$$

where $\partial/\partial c_{jm}\mathcal{F}_{\mathrm{MMI}}(\lambda)$, which is properly given by $\frac{1}{c_{jm}}(\gamma_{jm}^{\mathrm{num}} - \gamma_{jm}^{\mathrm{den}})$, is calculated instead according to the altered formula $\partial F/\partial c_{jm} = \frac{\gamma_{jm}^{\mathrm{num}}}{\sum_m \gamma_{jm}^{\mathrm{num}}} - \frac{\gamma_{jm}^{\mathrm{den}}}{\sum_m \gamma_{jm}^{\mathrm{den}}}$. C is set globally to the smallest value which gives all positive updates, plus a constant ϵ. Rows of transition matrices are updated according to an analogous formula.

6.3.3 New weight and transition updates

Since the baseline mixture weight update described in the previous section is based on an ad-hoc formula for the differentials and cannot even be proved to converge to the correct point (assuming it converges), an alternative formulation was sought.

As discussed in Section 4.2, the important thing is that the gradient of the auxiliary function w.r.t the parameter should be the same as the gradient of the real objective function, at the point where the parameters equal the previous parameters.

That condition (the differentials being equivalent to the differentials of the real objective function w.r.t. the parameters, at the starting point) would satisfied by the following auxiliary function for state j, which is extracted from two Baum-Welch type auxiliary functions subtracted from each other $(\mathcal{G}^{\mathrm{num}}(\lambda, \lambda') - \mathcal{G}^{\mathrm{den}}(\lambda, \lambda'))$, and is a weak-sense auxiliary function for the MMI objective function:

$$\mathcal{G}(\lambda, \lambda') = \sum_m (\gamma_{jm}^{\mathrm{num}} - \gamma_{jm}^{\mathrm{den}}) \log c_{jm}. \tag{6.18}$$

However, optimising the above while enforcing positive weights would make some weights zero. The following auxiliary function has the same differentials w.r.t. the weights where the weights equal the original values c_{jm}^{orig}, so it is a weak-sense auxilary function for the MMI objective function around λ':

$$\mathcal{G}(\lambda, \lambda') = \sum_m \gamma_{jm}^{\mathrm{num}} \log c_{jm} - \frac{\gamma_{jm}^{\mathrm{den}}}{C} \left(\frac{c_{jm}}{c_{jm}^{\mathrm{orig}}} \right)^C. \tag{6.19}$$

The smoothing constant $C > 0$ controls speed of convergence: large C leads to slow updates, small C leads to faster updates. $C = 1$ is considered the default setting: this leads to the formulation given in Section 4.4.

6.3.4 Mixture weight auxiliary function optimisation

Optimisation for $C \leq 1$

It is possible to find a strong-sense auxiliary function for the "objective function" of Equation 6.19 which is guaranteed to converge for $(0 < C \leq 1)$. Note that

there are two levels of auxiliary function here: Equation 6.19 is a weak-sense auxiliary function for the MMI objective function, and in this section a strong-sense auxiliary function for Equation 6.19 is derived in order to maximise it.

The optimisation procedure is as follows. For all j, m set $c_{jm}^{(0)} = c_{jm}^{\text{orig}}$ (i.e. to the original values from before the optimisation, which will henceforth be denoted with the superscript "orig") and then for iterations $p = 0$ to, say, 100, set for all j, m:

$$c_{jm}^{(p+1)} = \frac{\gamma_{jm}^{\text{num}} + c_{jm}^{(p)} k_{jm}^{(p)}}{\sum_m \gamma_{jm}^{\text{num}} + c_{jm}^{(p)} k_{jm}^{(p)}}, \tag{6.20}$$

where

$$k_{jm}^{(p)} = \left(\max_m \frac{\gamma_{jm}^{\text{den}}}{c_{jm}^{\text{orig}}} \left(\frac{c_{jm}^{(p)}}{c_{jm}^{\text{orig}}} \right)^{C-1} \right) - \frac{\gamma_{jm}^{\text{den}}}{c_{jm}^{\text{orig}}} \left(\frac{c_{jm}^{(p)}}{c_{jm}^{\text{orig}}} \right)^{C-1}. \tag{6.21}$$

The values of c_{jm} after 100 iterations are used for the updated values (convergence can be slow enough to make 100 iterations necessary for some values of parameters and statistics).

The proof that this formula is guaranteed to monotonically increase the function in Equation 6.19 is as follows. Suppose the current iteration is p and we wish to find an auxiliary function for the function of Equation 6.19, giving more optmial values $c_{jm}^{(p+1)}$. The objective function being optimised on iteration p is as follows:

$$\mathcal{F}(\lambda) = \sum_{m=1}^{M} \gamma_{jm}^{\text{num}} \log c_{jm}^{(p+1)} - \frac{\gamma_{jm}^{\text{den}}}{C} \left(\frac{c_{jm}^{(p+1)}}{c_{jm}^{\text{orig}}} \right)^{C} \tag{6.22}$$

which is a function of the unknowns $c_{jm}^{(p+1)}$ for $m = 1 \ldots M$. The starting point for the optimisation is the values on the previous iteration, $c_{jm}^{(p+1)} = c_{jm}^{(p)}$. As explained in Section 4.2, any function, including the one in Equation 6.22, is a strong-sense auxiliary function of itself, and this property is unchanged by adding a smoothing fuction, which is defined as a function which has its greatest value at the starting point of the optimisation. Firstly, we can replace the term $-\frac{\gamma_{jm}^{\text{den}}}{C} \left(\frac{c_{jm}^{(p+1)}}{c_{jm}^{\text{orig}}} \right)^{C}$, which is a power of $c_{jm}^{(p+1)}$, with a linear function of $c_{jm}^{(p+1)}$: $-c_{jm}^{(p+1)} \frac{\gamma_{jm}^{\text{den}}}{c_{jm}^{\text{orig}}} \left(\frac{c_{jm}^{(p)}}{c_{jm}^{\text{orig}}} \right)^{(C-1)}$, for each m. This has the same differential w.r.t. $c_{jm}^{(p+1)}$ at $c_{jm}^{(p+1)} = c_{jm}^{(p)}$ and is flat whereas the original function was either concave or flat (remembering that $0 < C \leq 1$ and the minus sign), so this change is equivalent to adding a function which has its maximum at the starting point $c_{jm}^{(p+1)} = c_{jm}^{(p)}$; it is therefore a valid

smoothing function. This gives an auxiliary function $\mathcal{G}(\lambda^{(p+1)}, \lambda^{(p)})$ as follows:

$$\mathcal{G}(\lambda^{(p+1)}, \lambda^{(p)}) = \sum_m \gamma_{jm}^{\text{num}} \log c_{jm}^{(p+1)} - c_{jm}^{(p+1)} \frac{\gamma_{jm}^{\text{den}}}{c_{jm}^{\text{orig}}} \left(\frac{c_{jm}^{(p)}}{c_{jm}^{\text{orig}}} \right)^{(C-1)} \tag{6.23}$$

Equation 6.23, in which $c_{jm}^{(p+1)}$ are the variables, is still not analytically tractable in terms of finding the maximum. Another smoothing function must be added, this time with the objective of canceling out the linear terms in $c_{jm}^{(p+1)}$ (the terms in $\log c_{jm}^{(p+1)}$ are more tractable). The new smoothing function is $\sum_{m=1}^{M} k_{jm}^{(p)} \left(c_{jm}^{(p)} \log c_{jm}^{(p+1)} - c_{jm}^{(p+1)} \right)$, where $k_{jm}^{(p)}$ are positive constants for each m. This function has its largest value at the "starting point" $c_{jm}^{(p)}$; this can easily be verified by inspection. The auxiliary function now becomes:

$$\mathcal{H}(\lambda^{(p+1)}, \lambda^{(p)}) = \sum_{m=1}^{M} \gamma_{jm}^{\text{num}} \log c_{jm}^{(p+1)} - c_{jm}^{(p+1)} \frac{\gamma_{jm}^{\text{den}}}{c_{jm}^{\text{orig}}} \left(\frac{c_{jm}^{(p)}}{c_{jm}^{\text{orig}}} \right)^{(C-1)} + k_{jm}^{(p)} \left(c_{jm}^{(p)} \log c_{jm}^{(p+1)} - c_{jm}^{(p+1)} \right). \tag{6.24}$$

The values $k_{jm}^{(p)}$ are chosen so as to make the coefficients of the linear terms in $c_{jm}^{(p+1)}$ in Equation 6.24 all become the same, so that due to the sum-to-one constraint the linear terms reduce to a constant independent of the new weights $c_{jm}^{(p+1)}$. For optimisation which is as fast as possible, the coefficients $k_{jm}^{(p)}$ are chosen to be the lowest values which will make the coefficients of the terms in $c_{jm}^{(p+1)}$ all be the same; the smallest $k_{jm}^{(p)}$ will always be zero. This leads to the expression in Equation 6.21 for $k_{jm}^{(p)}$.

The coefficients of terms in $c_{jm}^{(p+1)}$ become equal to $c_{jm}^{(p+1)} \frac{\gamma_{jm}^{\text{den}}}{c_{jm}^{\text{orig}}} \left(\frac{c_{jm}^{(p)}}{c_{jm}^{\text{orig}}} \right)^{(C-1)} - k_{jm}^{(p)}$ (extracted from Equation 6.24) which equals $-\max_m \frac{\gamma_{jm}^{\text{den}}}{c_{jm}^{\text{orig}}} \left(\frac{c_{jm}^{(p)}}{c_{jm}^{\text{orig}}} \right)^{C-1}$; this is a constant independent of m.

The coefficients of terms in $\log c_{jm}^{(p+1)}$ become equal to $\gamma_{jm}^{\text{num}} + c_{jm}^{(p)} k_{jm}^{(p)}$; this leads to the update in Equation 6.20, which can be derived using Lagrangian multipliers in the same way as the normal Baum-Welch update for weights.

In the special case of $C = 1$, the expression in Equation 6.21 for the constants $k_{jm}^{(p)}$ can be expressed more simply as:

$$k_{jm}^{(p)} = \left(\left(\max_m \frac{\gamma_{jm}^{\text{den}}}{c_{jm}^{\text{orig}}} \right) - \frac{\gamma_{jm}^{\text{den}}}{c_{jm}^{\text{orig}}} \right). \tag{6.25}$$

The optimisation is usually complete to machine accuracy within 10 or 20 iterations; however some mixtures of Gaussians are more intransigent. Applying the

transformation 100 times per Gaussian mixture seems sufficient to give the exact solution for almost all Gaussian mixures, as judged from correspondence with an alternative technique.

Optimisation for $C > 1$

The approach given above is only guaranteed to converge for $C \leq 1$, since $\mathcal{G}(\lambda^{(p+1)}, \lambda^{(p)})$ (Equation 6.23) is only a strong-sense auxiliary function for $\mathcal{F}(\lambda^{(p+1)})$ (Equation 6.22) around $\lambda^{(p)}$, if $C \leq 1$. But it is a weak-sense auxiliary function for any value of C, so if the optimisation method described above does converge it will be to the correct point. In practice the optimisation does not converge for $C = 2$. But the optimisation can be made to converge if more of a smoothing function is added at a later stage to slow down optimisation by increasing $k_{jm}^{(p)}$. Replacing the term $\max_m \frac{\gamma_{jm}^{\text{den}}}{c_{jm}^{\text{orig}}} \left(\frac{c_{jm}^{(p)}}{c_{jm}^{\text{orig}}} \right)^{C-1}$ of Equation 6.21 with $C \max_m \frac{\gamma_{jm}^{\text{den}}}{c_{jm}^{\text{orig}}} \left(\frac{c_{jm}^{(p)}}{c_{jm}^{\text{orig}}} \right)^{C-1}$ where $C > 1$ seems to lead to satistfactory convergence for $C = 2$ (the only case which was tried for $C > 1$).

6.4 Experimental results

6.4.1 Previous comparisons of of parameter updates for MMI.

In [Kapadia, 1998], a comparison between Extended Baum-Welch (EB) and a number of gradient-based algorithms was performed. As a result of his experiments Kapadia favoured the "On-line Manhattan Quick-Prop" algorithm, although the EB update was also effective. The comparison was on the basis of function optimisation rather than recognition results. "Quick-Prop" is a method which uses the gradients on two subsequent iterations to update parameters using Newton's method[3], with the limitation that the step size of a parameter may not increase by more than a certain factor (say, 2). Manhattan refers to using a fixed step size for all parameters on the first iteration; "On-line" means the data is broken up into subsets which are used in rotation (but the gradient from the previous iteration is taken to mean the last iteration with the same subset). However, that method seemed to have the potential for instability (as demonstrated by some of Kapadia's experiments using that technique for ML training), and furthermore is more complex to implement than EB since it requires statistics from previous iterations.

An experimental comparison between EB and gradient descent (GD) appears in [Schluter, 2000]. In that work, EB was compared with a gradient descent

[3]Ignoring the off-diagonal elements of the Hessian.

method with learning rates chosen to make the two almost equivalent (although differing slightly for the variances). Both forms of update used the Normandin-style EB update of Section 6.3.2 for the weights. No consistent difference in test-set performance was seen between the two approaches (although EB updates seemed to give better training set results). Note that Schluter uses a slightly different formula for setting E from the one used in work presented here (see Section 4.5.3).

In [Zheng et al., 2001], something very similar was done. EB was put in a GD framework, resulting (like the above) in equations that differed only in the variance update. The authors chose the same formula for setting as is used here (see Section 4.5.3), using $E = 1$ for relatively fast training. Again, no significant difference between EB and GD was found.

The "Newton's method" based technique compared experimentally in this chapter with other techniques is essentially the the same as the approach investigated in [Schluter, 2000] and [Zheng et al., 2001]. Again, no significant differences with EB are found.

6.4.2 Comparison of Gaussian updates

Table 6.2 compares three different update schemes for Gaussians, applied to optimising the MMI criterion on the 68h training subset of Switchboard (the setup is described in Appendix A).

The criteria compared are:

1. EB: Extended Baum-Welch update
2. "Flat-denominator" update (Section 6.2.2)
3. "Newton's method": Newton's-method based optimisation for "Flat-denominator" auxiliary function (Section 6.2.3)

All experiments reported in this section use the standard weight and transition updates described in Section 4.4, which is equivalent to the more general one described in Section 6.3 with the smoothing constant C set to 1.

Table 6.2 shows the difference in test set results between the three update formulae. There is no consistent difference in recognition performance. In optimising the criterion the standard EB equations seem to be best, although not on every iteration of training.

The difference between the parameters of the updated model sets is summarised as follows:

- The Newton's-method update formula only differs from EB for the variances. After one iteration of the Newton's-method update, almost all updated variances are slightly larger with the Newton's-method update than

	Training iteration				
	0	1	2	3	4
	%WER on eval97sub				
EB	46.0	44.5	43.7	43.9	43.8
Flat-denominator	46.0	44.4	43.7	43.9	43.9
Newton's-method	46.0	44.4	43.7	44.0	43.8
	%WER on eval98				
EB	46.6	45.4	44.7	44.4	44.3
Flat-denominator	46.6	45.3	44.8	44.4	44.2
Newton's-method	46.6	45.4	44.8	44.4	44.2
	MMI criterion / n frames				
EB, $E = 2$	-0.05446	-0.04634	-0.04057	-0.03616	
Flat-denominator	-0.05446	-0.04630	-0.04063	-0.03632	
Newton's-method	-0.05446	-0.04636	-0.04068	-0.03635	

Table 6.2: Alternative Gaussian update formulae tested on Switchboard: h5train00sub (65h) training, 12 mixture components, E =2 in all cases.

with the EB update, the average difference being around 10% of the total change.

- The flat-denominator formula differs from the EB formulae for both means and variances. Means differ in both directions by about 10% of the total change due to the update and variances by about 15% of the total change, with the variances generally being larger than EB as for the Newton's-method update.

6.4.3 Comparison of weight and transition updates

Table 6.3 compares the "new" update of Section 6.3 using $C = 1$ for both weights and transitions, with the use of the EB formulae to weights or transitions. The "new" update is considered the baseline, and the EB update is tried for the weights and transitions separately. Where the EB formulae (Equation 6.3.2) are used, C is set to the lowest value which gives all-positive updates for that type of parameter (weights or transitions) multiplied by 1.1; this value of C is shown in Table 6.3. For transitions the value of C is much smaller.

As can be seen from the MMI criterion reported in Table 6.3, the EB optimisation is slower than the new update for the weights and faster than the new update for the transitions. Using the (faster) EB optimisation for the transitions seems to degrade WER; for the weights, the use of the (slower) EB optimisation gives no consistent difference. Thus, there is no consistent difference in either criterion optimisation or recognition results between the EB and new update equations.

Update type		Training iteration				
Weight	Trans	0	1	2	3	4
		%WER on eval97sub				
New	New	46.0	44.5	43.7	43.9	43.8
EB	New	46.0	44.8	44.1	44.0	43.9
New	EB	46.0	44.6	44.3	44.5	44.3
		%WER on eval98				
New	New	46.6	45.4	44.7	44.4	44.3
EB	New	46.6				44.2
New	EB	46.6				44.5
		MMI criterion / n frames				
New	New	-0.05446	-0.04634	-0.04057	-0.03616	
EB	New	-0.05446	-0.04718	-0.04162	-0.03712	
New	EB	-0.05446	-0.04585	-0.03981	-0.03534	
		EB Smoothing constant C for weights or transitions				
EB	New	0.23	0.16	0.12	0.07	[weight C]
New	EB	0.07	0.04	0.02	0.02	[trans C]

Table 6.3: Comparison between new and EB weight/transition updates: h5train00sub (65h) training, 12 mixture components, Gaussian update: EB, E=2

However, the new update equations are favoured since the update, if it converges, will at least converge to the correct point; this does not hold true of the EB equations with altered differentials.

Tables 6.4 and 6.5 show the effect of varying the smoothing constant C in the new update for weights and transitions respectively (Note– this C is different from the C which appears in the EB equations and which is calculated based on a fixed formula). $C = \infty$ means the parameters are not changed. A smaller value of C leads to a faster increase in the criterion, and this is reflected in the critereion values. The standard update of $C = 1$ for both weights and transitions seems to be close to the optimal value with respect to WER. Recognition results on the larger eval98 test set show an insignificant improvement of 0.1% if either the weights are updated more slowly ($C = 2$) or the transitions faster ($C = 0.5$).

Note that on the first iteration or two of update a faster speed of weight update seems to lead to better WER. However these gains are lost on later iterations. It may be the case that fast weight updates help recognition directly, but cause problems on later iterations of training because data gets distributed less evenly among the Gaussians.

Value of C for...		Iteration				
Weights	Transitions	0	1	2	3	4
		%WER on eval97sub				
0.5	1	46.0	44.1	43.9	43.7	43.9
1	1	46.0	44.5	43.7	43.9	43.8
2	1	46.0	44.7	43.9	43.9	43.7
∞	1	46.0	45.1	44.5	44.0	43.8
		%WER on eval98				
0.5	1	46.6				44.3
1	1	46.6				44.3
2	1	46.6				44.2
∞	1	46.6				44.3
		MMI criterion / n frames				
0.5	1	-0.0545	-0.0451	-0.0392	-0.0349	
1	1	-0.0545	-0.0463	-0.0406	-0.0362	
2	1	-0.0545	-0.0469	-0.0414	-0.0370	
∞	1	-0.0545	-0.0476	-0.0423	-0.0381	

Table 6.4: Effect of varying smoothing constant C in new update for weights: h5train00sub (65h) training, 12 mixture components, Gaussian update: EB, E=2

6.4.4 Combining the best settings

Combining a "flat-denominator" (E =2) update with a slow weight update (C = 2) and a fast transition update ($C = 0.5$) gave an WER of 44.1% on the eval98 test set on the fourth iteration, and 43.8% on eval97sub. This is 0.2% better on the larger eval98 test set and the same on eval97sub, compared with the standard EB update with $C = 1$ for weights and transitions. This is not significant, so these changes have not been used for further work– the standard updates have been retained, with original EB formulation for the Gaussians, and the update of Section 4.4 for the weights and transitions which is equivalent to the update of Section 6.3 with $C = 1$.

6.5 Conclusion

This chapter investigated various modifications of and alternatives to the standard optimisation approach for MMI that was previously described in Chapter 4. Although some of these modifications led to a very slight improvement in WER on testing, the total improvement was not significant, and the standard update formulae of Chapter 4 have been used for further work. Experiments by other authors, describing other modifications to the EB formulae, were reviewed; they

Value of C for...		Iteration				
Weights	Transitions	0	1	2	3	4
		%WER on eval97sub				
1	0.5	46.0	44.5	43.6	43.9	43.8
1	1	46.0	44.5	43.7	43.9	43.8
1	2	46.0	44.4	43.8	44.0	43.7
1	∞	46.0	44.4	43.8	43.9	43.8
		%WER on eval98				
1	0.5	46.6				44.2
1	1	46.6				44.3
1	2	46.6				44.3
1	∞	46.6				44.4
		MMI criterion / n frames				
1	0.5	-0.0545	-0.0462	-0.0401	-0.0359	
1	1	-0.0545	-0.0463	-0.0406	-0.0362	
1	2	-0.0545	-0.0464	-0.0407	-0.0363	
1	∞	-0.0545	-0.0464	-0.0408	-0.0364	

Table 6.5: Effect of varying smoothing constant C in new update for transitions: h5train00sub (65h) training, 12 mixture components, Gaussian update: EB, E=2

also showed that no significant WER gains can be obtained by using alternative update equations of the kind investigated here.

The exercise of devising an auxiliary function with a linear denominator auxiliary function was useful as it led to an update very similar to the EB update with $E = 1$, and with the state-specific smoothing constants set in essentially the same way, but avoiding the arbitrariness of the formula for setting the Gaussian-specific smoothing constant D_{jm}.

Chapter 7

Minimum Phone Error Training

7.1 Introduction

The Minimum Phone Error (MPE) criterion, previously introduced in Chapter 3, is a smoothed phone transcription accuracy. A related criterion, Minimum Word Error (MWE), is a smoothed word accuracy. Both criteria consist of an average of the transcription accuracies of sentences s, weighted by the probability of s given the model:

$$\mathcal{F}_{\mathrm{MPE}}(\lambda) = \sum_{r=1}^{R} \sum_{s} P_{\lambda}^{\kappa}(s|\mathcal{O}_r) A(s, s_r) \tag{7.1}$$

where $P_{\lambda}^{\kappa}(s|\mathcal{O}_r)$ is defined as the scaled posterior sentence probability $\frac{p_{\lambda}(\mathcal{O}_r|s)^{\kappa} P(s)^{\kappa}}{\sum_u p_{\lambda}(\mathcal{O}_r|u)^{\kappa} P(u)^{\kappa}}$ of the hypothesised sentence s.

The function $A(s, s_r)$ equals the number of phones in the reference transcription s_r for file r, minus the number of phone errors; in MWE this is calculated as a word-level rather than phone-level accuracy.

The MPE objective function, like the MMI objective function, aims to make sure that the the training data is correctly recognised; but in MPE each phone error in the data can contribute at most one unit to the objective function, rather than an unlimited quantity in the case of MMI (as the likelihood difference between the correct and incorrect versions increases). This makes the MPE objective function less sensitive to portions of the training data that are poorly transcribed or so unclear that they cannot be expected to be correctly transcribed.

The lattice-based implementation of MPE/MWE which will be described in this chapter is very similar to the implementation of MMI described in Chapter 5. Changes to the training algorithm are required at the stage at which statistics are accumulated from the training data. A similar amount of computing resources

are required as for MMI[1].

This chapter is organised as follows. Section 7.2 explains an approximate method used to optimise the MPE objective function; Section 7.3 discusses a more exact implementation, and Section 7.4 summarises the MPE training process and gives various further details. Experimental results for MPE are given in Chapter 8.

7.2 Optimisation of the MPE objective function

The EB update formulae were developed for the optimisation of the MMI objective function, and were originally proved for that case. The same approach is not directly applicable to MPE. In the MPE objective function, which is given as follows in an expanded form,

$$\mathcal{F}_{\mathrm{MPE}}(\lambda) = \sum_{r=1}^{R} \frac{\sum_s p_\lambda(\mathcal{O}_r|s)^\kappa P(s)^\kappa A(s,s_r)}{\sum_u p_\lambda(\mathcal{O}_r|u)^\kappa P(u)^\kappa}, \tag{7.2}$$

the factors $A(s,s_r)$ which are multiplied by each sentence likelihood in the numerator are not necessarily positive, and the individual fractions are added together rather than multiplied as in MMI. This makes it difficult to derive the EB equations in the original way, as in [Gopalakrishnan et al., 1989, Normandin & Morgera, 1991].

The solution used here is to use an intermediate weak-sense auxiliary function, based on a sum over phone arcs. The lattice for each training file r is composed of phone arcs $q = 1 \ldots Q_r$, each with given start and end times. For each phone arc q, the likelihood of the speech data from the beginning to the end of the arc can be calculated; let this be called $p(\mathcal{O}|q)$.

The weak-sense auxiliary function used to make the MPE objective function more tractable is:

$$\mathcal{H}_{\mathrm{MPE}}(\lambda,\lambda') = \sum_{r=1}^{R}\sum_{q=1}^{Q_r} \left.\frac{\partial \mathcal{F}_{\mathrm{MPE}}}{\partial \log p(\mathcal{O}|q)}\right|^{(\lambda=\lambda')} \log p(\mathcal{O}|q) \tag{7.3}$$

This function $\mathcal{H}_{\mathrm{MPE}}(\lambda,\lambda')$ is a weak-sense auxiliary function for $\mathcal{F}_{\mathrm{MPE}}(\lambda)$ around $\lambda = \lambda'$, for the following reason: the only change in $\mathcal{F}_{\mathrm{MPE}}(\lambda)$ as λ is changed comes via the sentence likelihoods $p_\lambda(\mathcal{O}_r|s)$, and the only variables in these that vary with λ are the arc likelihoods $p(\mathcal{O}|q)$. An approximation to $\mathcal{F}_{\mathrm{MPE}}(\lambda)$ which is linear in the values of $\log p(\mathcal{O}|q)$ will have the same differential w.r.t λ where $\lambda = \lambda'$, and will therefore be a weak-sense auxiliary function for $\mathcal{F}_{\mathrm{MPE}}(\lambda)$ around $\lambda = \lambda'$.

[1] in both cases, about 0.5× real-time per iteration of training for a typical training setup on the Switchboard corpus when run on Pentium III processors at 800 MHz.

The value $\frac{\partial \mathcal{F}_{\text{MPE}}}{\partial \log p(\mathcal{O}|q)}|^{(\lambda=\lambda')}$ is a scalar value calculated for each arc q, and can be either positive or negative. The two cases can be separated, making the analogy with MMI clearer:

$$\begin{aligned}\mathcal{H}_{\text{MPE}}(\lambda,\lambda') = & \sum_{r=1}^{R}\sum_{q=1}^{Q_r}\max(0,\tfrac{\partial \mathcal{F}_{\text{MPE}}}{\partial \log p(\mathcal{O}|q)}|^{(\lambda=\lambda')})\log p(\mathcal{O}|q) \\ - & \sum_{r=1}^{R}\sum_{q=1}^{Q_r}\max(0,-\tfrac{\partial \mathcal{F}_{\text{MPE}}}{\partial \log p(\mathcal{O}|q)}|^{(\lambda=\lambda')})\log p(\mathcal{O}|q),\end{aligned} \tag{7.4}$$

where the first term corresponds to the numerator model in MMI and the second to the denominator. As for MMI, two sets of accumulated statistics are stored: one for the numerator, and one for the denominator.

(For the Gaussian updates the two sets of statistics may be compressed by saving only their difference). For I-smoothing with MPE, a third set of statistics, the "mle" statistics, are stored. The "mle" statistics are standard statistics as used for Maximum Likelihood estimation, and are obtained from a lattice forward-backward algorithm applied to the lattice of correct sentence (the numerator lattice in MMI).

The final auxiliary function (without the smoothing term added yet) is:

$$\mathcal{F}_{\text{MPE}}(\lambda,\lambda') = \sum_{r=1}^{R}\sum_{q=1}^{Q_r}\tfrac{\partial \mathcal{F}_{\text{MPE}}}{\partial \log p(\mathcal{O}|q)}|^{(\lambda=\lambda')}\mathcal{F}_{\text{MLE}}(\lambda,\lambda',r,q) \tag{7.5}$$

if $\mathcal{F}_{\text{MLE}}(\lambda,\lambda',r,q)$ is understood to be the normal auxiliary function for the arc likelihood $p(\mathcal{O}|q)$ for arc q from lattice r, as would be used for ML estimation. This expression is a weak-sense auxiliary function for Equation 7.3, and is therefore a weak-sense auxiliary function for the MPE objective function.

In storing statistics for updating the MPE objective function, an important definition is:

$$\gamma_q^{\text{MPE}} = \frac{1}{\kappa}\frac{\partial \mathcal{F}_{\text{MPE}}}{\partial \log p(\mathcal{O}|q)}, \tag{7.6}$$

which is the differential of the objective function w.r.t the arc log likelihood $\log p(\mathcal{O}|q)$, for the phone arc q, scaled by $\frac{1}{\kappa}$ which is an arbitrary scale introduced for consistency with MMI and to simplify the calculation of γ_q^{MPE}.

Once this value γ_q^{MPE} is calculated, the statistics needed to optimise the objective function can easily be calculated. In a modification of Equations (5.6) to (5.8) for MMI, the numerator and denominator statistics are accumulated according to the following equations for the numerator:

$$\gamma_{jm}^{\text{num}} = \sum_{q=1}^{Q}\sum_{t=s_q}^{e_q}\gamma_{qjm}(t)\max(0,\gamma_q^{\text{MPE}})) \tag{7.7}$$

$$\theta_{jm}^{\text{num}}(\mathcal{O}) = \sum_{q=1}^{Q}\sum_{t=s_q}^{e_q}\gamma_{qjm}(t)\max(0,\gamma_q^{\text{MPE}})\mathcal{O}(t) \tag{7.8}$$

$$\theta_{jm}^{\text{num}}(\mathcal{O}^2) = \sum_{q=1}^{Q}\sum_{t=s_q}^{e_q}\gamma_{qjm}(t)\max(0,\gamma_q^{\text{MPE}})\mathcal{O}(t)^2, \tag{7.9}$$

and as follows for the denominator:

$$\gamma_{jm}^{\text{den}} = \sum_{q=1}^{Q} \sum_{t=s_q}^{e_q} \gamma_{qjm}(t) \max(0, -\gamma_q^{\text{MPE}})) \tag{7.10}$$

$$\theta_{jm}^{\text{den}}(\mathcal{O}) = \sum_{q=1}^{Q} \sum_{t=s_q}^{e_q} \gamma_{qjm}(t) \max(0, -\gamma_q^{\text{MPE}}) \mathcal{O}(t) \tag{7.11}$$

$$\theta_{jm}^{\text{den}}(\mathcal{O}^2) = \sum_{q=1}^{Q} \sum_{t=s_q}^{e_q} \gamma_{qjm}(t) \max(0, -\gamma_q^{\text{MPE}}) \mathcal{O}(t)^2, \tag{7.12}$$

where s_q and e_q are the start and end times of arc q, and $\gamma_{qjm}(t)$ are the occupation probabilities for Gaussians conditional on the arc being q. This follows from Equation (7.4), in which arcs with positive γ_q^{MPE} are considered as numerator arcs, and arcs with negative γ_q^{MPE} are considered as denominator arcs. The same proof used to show that the MMI objective function can be optimised with an auxiliary function $\mathcal{G}(\lambda, \lambda')$ of the form given in Equation (4.17), applies to the function $\mathcal{H}_{\text{MPE}}(\lambda, \lambda')$ of Equation (7.4) which is has the same form as the MMI objective function. The final auxiliary function $\mathcal{G}(\lambda, \lambda')$ is the same for MPE as MMI and the EB update equations are applied in the same way. The only difference with MPE is that the statistics are accumulated according to Equations 7.7 to 7.12 rather than Equations 5.6 to 5.8.

7.2.1 Calculating $A(s, s_r)$ for approximate MPE

To calculate the statistics for MPE it is necessary to calculate the scaled differential γ_q^{MPE} of the MPE criterion w.r.t. the log likelihood of each arc. At some point this will involve implicitly calculating the function $A(s)$ for each sentence in the lattice.

The function $A(s)$ for a sentence s ideally equals the number of correct phones minus the number of insertions (deletions and substitutions show up as reductions in the number of correct phones), but an approximation may be used to avoid the need for a full alignment. The exact form of the function (i.e., the number of correct phones minus insertions) could equivalently be expressed as a sum of a phone-level accuracy $A(q)$ over all phones q in s, where $A(q)$ is defined as follows:

$$A(q) = \left\{ \begin{array}{l} 1 \text{ if correct phone} \\ 0 \text{ if substitution} \\ -1 \text{ if insertion} \end{array} \right\}. \tag{7.13}$$

Since the computation of the above expression requires alignment of the reference an hypothesis sequences, and this is computationally expensive, an approximation is used as follows. Given a hypothesis phone q, a phone z is found in the reference

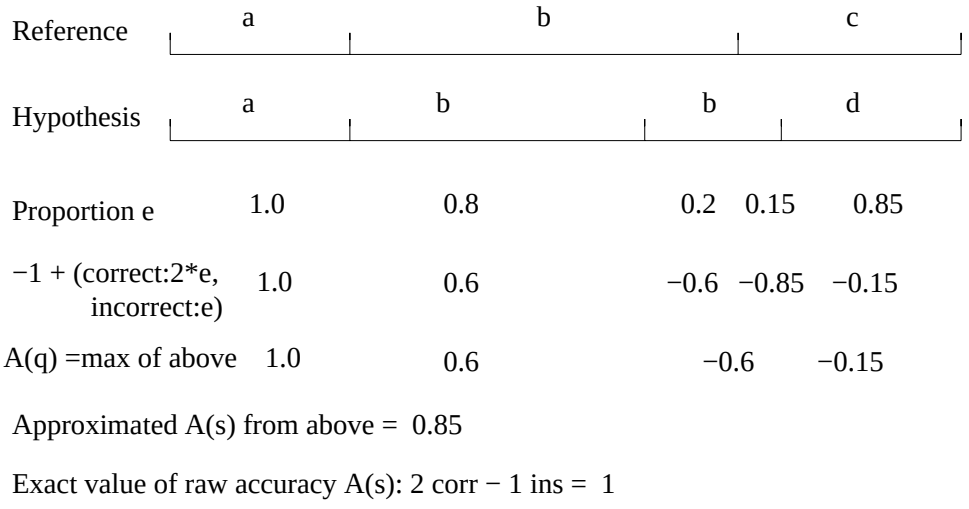


Figure 7.1: Calculating approximated accuracy

transcript which overlaps in time with q; and if the proportion of the length of z which is overlapped is $e(q,z)$,

$$A(q) = \max_z \left\{ \begin{array}{l} -1 + 2e(q,z) \text{ if z and q are same phone} \\ -1 + e(q,z) \text{ if different phones} \end{array} \right\}. \tag{7.14}$$

This is efficient to compute because it is a purely local function of the hypothesis and reference phones. The phone z is chosen so as to make $A(q)$ as large as possible. The expressions in Equation (7.14) represent tradeoffs between an insertion and a correct phone or substitution respectively, and are a solution to the problem that a single reference phone might be used more than once by a hypothesis sentence. In this implementation the reference phone z is chosen from a lattice encoding alternate alignments of the correct sentence. The expression in Equation (7.14) can be shown never to exceed the ideal value of Equation (7.13) provided the reference transcript has a single time alignment, i.e ignoring the fact of there being alternate paths in the reference lattice. The reference lattice may have multiple paths due to alternate pronunciations of words.

This approximation is easy to implement in a lattice context and seems to give good results. Note that unless indicated otherwise all experiments use context-independent phones for purposes of calculating phone accuracy, as opposed to matching the contexts as well, since this has been found experimentally to be the best approach (Section 8.17.2).

Figure 7.1 gives an example of calculating approximated accuracy for a single hypothesis and reference transcript. The calculated value (0.85) is compared to the exact value (1) and is slightly less. With a single reference transcript, the approximation will always be less than or equal to the true value. An approximate

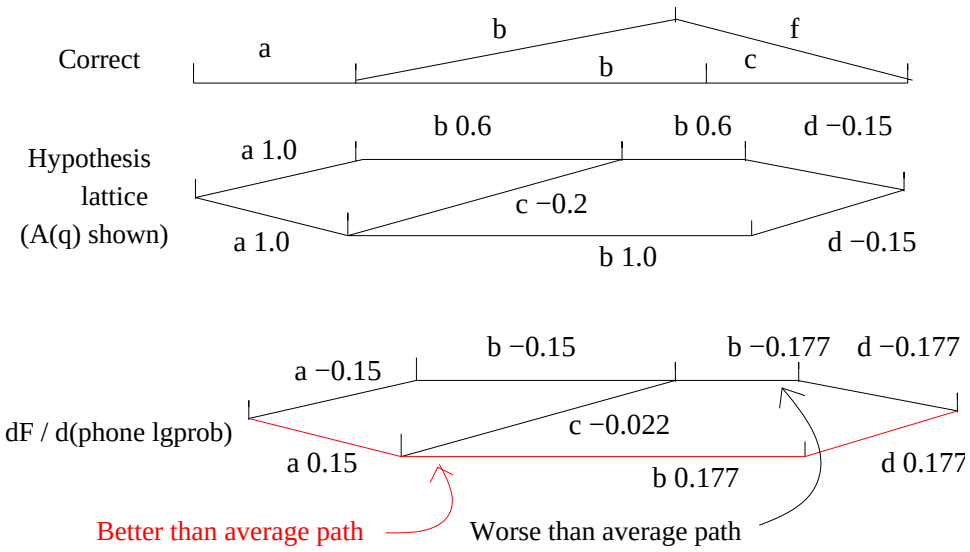


Figure 7.2: Calculating approximated accuracy in a lattice context

alignment can be turned into an exact alignment by redistributing the reference phones among hypothesis phones. A reference phone that is shared can always be given to one or other of the sharing hypothesis phones without decreasing the accuracy: this can be shown by considering the various ways that a reference phone can be shared and showing in each case which hypothesis phone it should go to. Unallocated reference phones will be implicitly counted as deletions. Any exact alignment of reference to hypothesis will give an accuracy less than or equal to the true accuracy; it follows that the approximated accuracy is always less than or equal to the true accuracy.

Figure 7.2 shows the same process in a lattice context. The correct transcription may contain alternate paths if alternate pronunciations appear in the dictionary; the reference phone z may be chosen from any path to maximise the phone accuracy. The hypothesis/recognition lattice (middle) is shown with the function $A(q)$ indicated for each phone arc.

The bottom lattice in Figure 7.2 shows the differential of the MPE objective function w.r.t. the log likelihoods of the different phone arcs (assuming the three paths in the lattice have equal likelihoods). It does not show how this differential is calculated, which is described in Section 7.2.2. But it makes clear a few properties of this differential, such as the fact that it is positive for paths that are more correct than average and negative for less correct paths; and that it will sum to zero for all arcs crossing a particular time instant.

To summarise, the approximated phone accuracy $A(q)$ is found for each hypothesis arc q as follows: for each arc z in the reference transcript which overlaps in time with q, let $e(q, z)$ be the number of frames q and z overlap, divided by

the length in frames of z. These values $e(q,z)$ are used in the expression in Equation (7.14) to calculate the approximated value of $A(q)$.

Silences

Silence and short pause models, are handled as follows: silences are ignored when they appear in the hypothesis transcript by being given a A of zero, but appear in the reference transcript where they may be used as the reference phone z when calculating $A(q)$: hypothesis phones which align to a silence phone would be counted as substitutions since the hypothesis phone q will never be silence itself. This was found to work better in terms of test-set performance than ignoring the silences and short pauses in both reference and hypothesis lattices (see experiments in Section 8.17.3).

Approximate vs. exact MPE

This approximate method of calculating $A(s)$ described above been compared with a more exact technique, and has been found to give slightly better test-set results than the exact technique on the Switchboard corpus (although worse for Wall Street Journal). The exact technique is described in Section 7.3. Experiments comparing the two are given in Section 8.14, and show no clear difference in performance.

7.2.2 Differentiating the MPE objective function for approximate MPE

The key quantity required in MPE training is:

$$\gamma_q^{\mathrm{MPE}} = \frac{1}{\kappa} \frac{\partial \mathcal{F}_{\mathrm{MPE}}}{\partial \log p(\mathcal{O}|q)}$$

for each arc q, which is the scaled differential of the MPE objective function w.r.t. each arc log likelihood. This is analogous to to an occupation probability that would arise in ML or MMI training; if positive, it is treated for purposes of accumulating statistics as an MMI numerator occupation probability and, if negative, an MMI denominator occupation probability.

This quantity can be found using the formula:

$$\gamma_q^{\mathrm{MPE}} = \gamma_q (c(q) - c^r_{\mathrm{avg}}), \tag{7.15}$$

where γ_q is the likelihood of the arc q as derived from a forward-backward likelihood computation over the arcs, $c(q)$ is the average accuracy $A(s)$ of sentences passing through the arc q, and c^r_{avg} is the average $A(s)$ of all the sentences in the

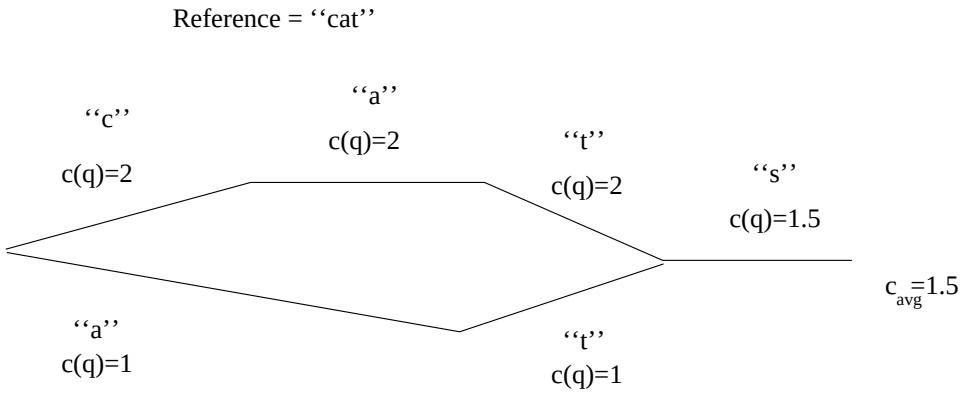


Figure 7.3: Example showing values of $c(q)$

recognition lattice for the r'th training file. (All these averages are weighted by the sentence likelihood).

An example giving values of $c(q)$ is shown in Figure 7.3. As mentioned, $c(q)$ is the average accuracy of sentences passing through the arc q, weighted by probability. This example assumes that the two alternate paths are equally likely, i.e. γ_q equals 0.5 for the top and bottom paths. Arcs on the top path have a $c(q)$ of 2, which equals the number of correct phones (3) minus 1 for one insertion error. Arcs q on the bottom path have $c(q) = 1$ because there are two errors. In this case, the expression $\gamma_q^{\mathrm{MPE}} = \gamma_q(c(q) - c^r_{\mathrm{avg}})$ equals 0.25 for the top arcs and -0.25 for the bottom arcs, and zero for the ending arc. These values ($\pm$ 0.25) are the largest values of γ_q^{MPE} possible where the alternative sentences do not differ in correctness by more than 1. The values of γ_q^{MPE} would become smaller if the sentence were less evenly matched in likelihood.

The expression for γ_q^{MPE} given in Equation (7.15) can be demonstrated to be correct as follows. The MPE objective function,

$$\mathcal{F}_{\mathrm{MPE}}(\lambda) = \sum_{r=1}^{R} \frac{\sum_s p_\lambda(\mathcal{O}_r|s)^\kappa P(s)^\kappa A(s, s_r)}{\sum_u p_\lambda(\mathcal{O}_r|u)^\kappa P(u)^\kappa},$$

can be split up into those sentences s which include a particular arc q ($q \in s$) and those which do not.

$$\mathcal{F}_{\mathrm{MPE}}(\lambda) = \sum_{r=1}^{R} \frac{\sum_{s:q\in s} p_\lambda(\mathcal{O}_r|s)^\kappa P(s)^\kappa A(s, s_r) + \sum_{s:q\notin s} p_\lambda(\mathcal{O}_r|s)^\kappa P(s)^\kappa A(s, s_r)}{\sum_{u:q\in u} p_\lambda(\mathcal{O}_r|u)^\kappa P(u)^\kappa + \sum_{u:q\notin u} p_\lambda(\mathcal{O}_r|u)^\kappa P(u)^\kappa}, \tag{7.16}$$

Differentiating this w.r.t. the arc log likelihood $\log p(\mathcal{O}|q)$ is possible by considering that for a sentence s which includes arc q ($q \in s$) the differential of its likelihood $p_\lambda(\mathcal{O}_r|s)^\kappa$ w.r.t. $\log p(\mathcal{O}|q)$ equals $\kappa p_\lambda(\mathcal{O}_r|s)^\kappa$ and for other sentences

($q \notin s$) the differential of $p_\lambda(\mathcal{O}_r|s)^\kappa$ w.r.t $\log p(\mathcal{O}|q)$ is zero, so by the product rule for fractions ($\frac{\partial}{\partial x}\frac{a}{b} = \frac{\partial a/\partial x}{b} - \frac{a\partial b/\partial x}{b^2}$),

$$\frac{\partial \mathcal{F}_{\text{MPE}}(\lambda)}{\partial \log p(\mathcal{O}|q)} = \kappa \frac{\sum_{s:q\in s} p_\lambda(\mathcal{O}_r|s)^\kappa P(s)^\kappa A(s,s_r)}{\sum_u p_\lambda(\mathcal{O}_r|u)^\kappa P(u)^\kappa} - \kappa \frac{\sum_s p_\lambda(\mathcal{O}_r|s)^\kappa P(s)^\kappa A(s,s_r)}{\sum_u p_\lambda(\mathcal{O}_r|u)^\kappa P(u)^\kappa} \frac{\sum_{s:q\in s} p_\lambda(\mathcal{O}_r|s)^\kappa P(s)^\kappa}{\sum_u p_\lambda(\mathcal{O}_r|u)^\kappa P(u)^\kappa}. \quad (7.17)$$

The expression is equivalent to equation (7.15), considering that the factor κ cancels with the $\frac{1}{\kappa}$ in the definition $\gamma_q^{\text{MPE}} = \frac{1}{\kappa}\frac{\partial \mathcal{F}_{\text{MPE}}}{\partial \log p(\mathcal{O}|q)}$, that the occupation probability γ_q equals $\frac{\sum_{s:q\in s} p_\lambda(\mathcal{O}_r|s)^\kappa P(s)^\kappa}{\sum_u p_\lambda(\mathcal{O}_r|u)^\kappa P(u)^\kappa}$, that the average correctness c^r_{avg} equals $\kappa\frac{\sum_s p_\lambda(\mathcal{O}_r|s)^\kappa P(s)^\kappa A(s,s_r)}{\sum_u p_\lambda(\mathcal{O}_r|u)^\kappa P(u)^\kappa}$ and that the average correctness of arc q equals $\frac{\sum_{s:q\in s} p_\lambda(\mathcal{O}_r|s)^\kappa P(s)^\kappa}{\sum_{u:q\in u} p_\lambda(\mathcal{O}_r|u)^\kappa P(u)^\kappa}$. The value of $c(q)$, which is the (weighted) average value of $A(s)$ for sentences s including the phone arc q, is calculated as follows, in an algorithm similar to the forward backward algorithm.

Computation for approximate MPE

Let the symbols α_q and β_q denote the forward and backward likelihoods used in the forward-backward algorithm to calculate occupancies $\gamma_q = \frac{\alpha_q\beta_q}{p(\mathcal{O})}$. The symbols α'_q and β'_q are used to define analogous quantities used in calculating average accuracies: α'_q represents the average accuracy of partial phone sequences leading up to q (including q itself), and β'_q represents the average accuracy of partial phone sequences following q, so that the average accuracy $c(q)$ of phone sequences including q equals $\alpha'_q\beta'_q$. These quantities are calculated as follows:

```
for q = 1 ... Q
    if q is a starting arc (no transitions to q)
        α_q = p(O|q)^κ
        α'_q = A(q)
    else
        α_q = Σ_{r preceding q} α_r t^κ_{rq} p(O|q)^κ
        α'_q = (Σ_{r preceding q} α'_r α_r t^κ_{rq}) / (Σ_{r preceding q} α_r t^κ_{rq}) + A(q)
    end
end
for q = Q ... 1
    if q is an ending arc (no transitions from q)
        β_Q = 1
        β'_Q = 0
    else
        β_q = Σ_{r following q} t^κ_{qr} p(r)^κ β_r
```

$$\beta'_q = \frac{\sum_{r \text{ following } q} t_{qr}^{\kappa} p(r)^{\kappa} \beta_r (\beta'_r + A(r))}{\sum_{r \text{ following } q} t_{qr}^{\kappa} p(r)^{\kappa} \beta_r}$$

end
end

$$c^r_{\text{avg}} = \frac{\sum_{\text{arcs } q \text{ at end of lattice}} \alpha'_q \alpha_q}{\sum_{\text{arcs } q \text{ at end of lattice}} \alpha_q}$$

$x = \sum_{\text{arcs } q \text{ at end of lattice}} \alpha_q$
for $q = 1 \ldots Q$
$\gamma_q = \frac{\alpha_q \beta_q}{x}$
$c(q) = \alpha'_q + \beta'_q$
$\gamma_q^{\text{MPE}} = \gamma_q (c(q) - c^r_{\text{avg}})$
end

where $A(q)$ is the (approximated) contribution of phone q to the sentence correctness, $p(\mathcal{O}|q)$ is the likelihood of the data aligned to phone arc q, derived from an unscaled forward-backward probability calculation within q, t_{qr} are lattice transition probabilities derived from the language model, and the notation $\sum_{r \text{ preceding } q}$ indicates summation over phone arcs r that directly precede q in the lattice. The scaled differential w.r.t. the arc log likelihood can then be calculated according to the formula $\gamma_q^{\text{MPE}} = \gamma_q (c(q) - c^r_{\text{avg}})$. This formulation assumes that the arcs are sorted in order of time.

7.3 Exact implementation of MPE

Problems with approximated MPE

The above implementation uses an approximate value for $A(q)$. Not only does the approximation result in an underestimate of the true phone accuracy of a hypothesis sequence, but the process gives a higher accuracy to paths matching longer pronunciations of words (if there is a choice). The highest possible value of $A(s)$ for a hypothesis s equals the length in phones of the correct sentence, and a choice between different correct sentences arises because there are alternative pronunciations of words. Thus, the approximate technique gives an unsatisfactory preference to longer reference pronunciations.

A more exact implementation of MPE has been devised. This is referred to as exact MPE.

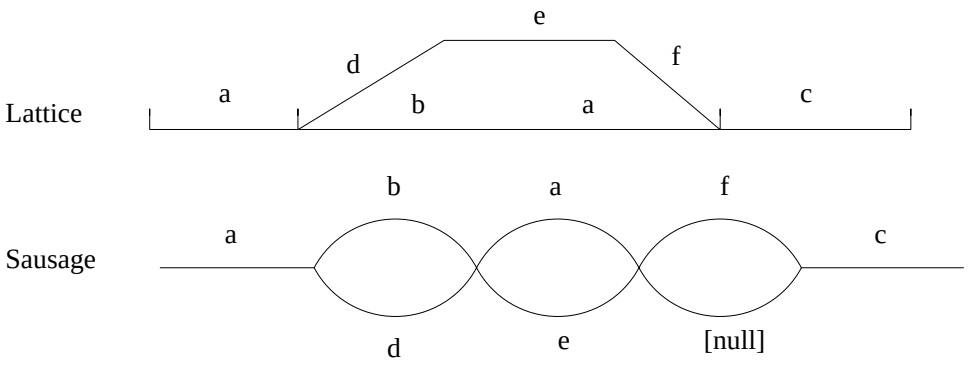


Figure 7.4: Sausage representation of numerator

7.3.1 Sausage strings

Firstly, the numerator lattice encoding alternative pronunciations is represented as a string of phone "sausages", not including silences or short pauses. (A similar idea in the context of word hypotheses was introduced in [Mangu et al., 2002]). The idea is shown in Figure 7.4. The length P of the the phone sausage string equals the summed length of the longest pronunciations of words; each position $p = 1 \ldots P$ in the sausage string contains a number of alternate reference phones r_{pa}, for alternatives $a = 1 \ldots n_p$. In addition each position may possibly contain an "empty" phone, as seen in the next-to-last position of Figure 7.4. The presence of an empty phone in position p is indicated by a Boolean value e_p being true. In this implementation, if a word has alternate pronunciations these will start at the same position in the sausage string and if some pronunciations are longer than others empty phones will appear towards the end of the word (this case is shown in Figure 7.4).

7.3.2 Raw Error

Once the reference sausage string is defined, the alignment of the hypothesis sentence will be implicitly calculated in a process which calculates the "MPE occupancy" γ_q^{MPE}. The raw accuracy of a hypothesis sentence s can be calculated as $P - E(s)$, where P is the length of the sausage string and $E(s)$ is the number of phone errors in s; this gives a better consistency between hypotheses of different length since the criterion is now related to the error in the sentence. The negated error $-E(s)$ is now used in lattice calculations in the same way as $A(s)$ was used previously, except that for purposes of reporting the criterion the sausage string length P is added to it.

The raw error consists of the total number of substitutions, deletions and insertions. The negated raw error which is actually used consists of -1 for a substitu-

tion or deletion, and I for an insertion, where I might equal -0.85 or -0.9. This introduces a flexibility which can avoid putting too much emphasis on correcting insertions of phones, which otherwise can lead to a reduction in the test set insertion/deletion ratio.

Calculating sentence Raw Error for a single sentence

The example of calculating negated Raw Error for a single sentence is considered here in order to clarify the lattice algorithm which will be needed.

The traceback algorithm needed to find the best path is viewed as a forward backward type algorithm over the sentence, in which each of the phones in positions $t = 1 \ldots T$ in the hypothesis sentence is considered as a separate phone depending on the starting position $p = 1 \ldots P + 1$ of the reference phone it is aligned to. A transition from one phone to the next, with given positions p for the two phones, only has nonzero probability if the position of the first phone is the best choice, i.e. gives the best error value up to and including the next phone. The process will be described exactly below.

The negated error up to and including the t'th phone, where the t'th phone is aligned to reference position p, is called $\alpha'(t, p)$. This is demonstrated in Figure 7.5, where $\alpha'(t, p)$ is indicated next to each phone. A similar error $\beta'(t, p)$ is the error from but not including t, p to the end of the sentence. Two new quantities are introduced which will prove useful later. $\alpha(t, p)$ is the probability that the current position appears in the backtrace of the file including information gathered up to and not including the present phone, and is always 1 in this case (it becomes useful when alternate transcriptions with different probabilites are considered). $\beta(t, p)$ is a probability from the present phone to the end, which equals 1 if there is a path from t, p to the end of the sentence $T + 1, P + 1$, and zero otherwise.

The negated sentence error $\alpha'(t, p)$, the same quantity shown in Figure 7.5, is calculated as follows:

```
for p = 1...P+1,
    α'(1,1) = −err(1,1,p)
end
for t = 2...T
    for p = 1...P+1,
        α'(t,p) = max_{p'=1}^{P+1} α'(t−1,p') − err(t,p',p)
    end
end
```

The function $\mathrm{err}(t, p_1, p_2)$ is the error (insertions, deletions and substitutions) due

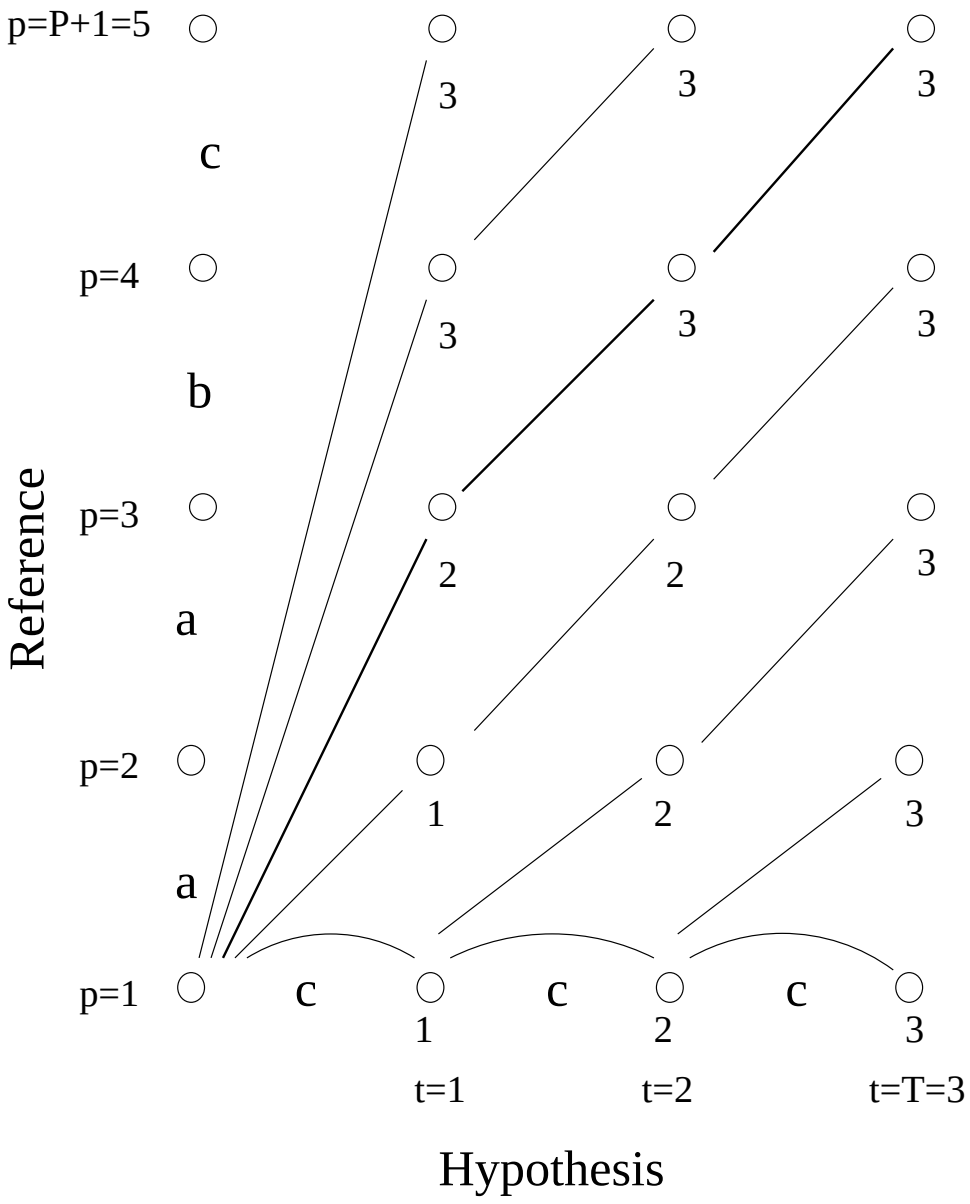


Figure 7.5: Recursive calculation of phone error, showing accumulated error $\alpha'(t, p)$

to the hypothesis phone at time t, aligned between reference positions p_1 and p_2 (these are positions p as indicated on the y-axis of Figure 7.5). If $p_1 = p_2$, the hypothesis phone is counted as an insertion, and if $p_1 < p_2$ the error is very large to indicate an invalid transition. An efficient implementation of this function is explained in Section 7.3.4.

The backward part of the algorithm calculates the backward error $\beta'(t,p)$ which is the negated error from position t,p to the end of the sentence not including phone t, i.e. the error after t assuming phone t ends at position p. In addition, the forward probability $\alpha(t)$, is defined, which is always one, and which is a shorthand for the notation $\alpha(t,p)$ as it is bound to be the same for all p; and a backward probability $\beta(t,p)$ which in this case is either zero or one. The reason for defining these quantities will become clear when the alignment procedure is combined with the lattice forward-backward algorithm.

$\alpha(t) = 1$ for all t
$\beta(t,p) \leftarrow 0$ for all t,p

$\beta(T, P+1) \leftarrow 1$

`for` $t = T-1 \ldots 1$ `% Calculate backward probabilities...`
    `for` $p = 1 \ldots P+1$
        $p' = \text{startof}(t+1, p)$
        $\beta(t,p') \leftarrow \beta(t,p') + \beta(t+1,p)$
    `end`
`end`

`for` $p = 1 \ldots P+1$ `% Calculate backward correctness...`
    $\beta'(T,p) = -\text{err}(T,p,P)$
`end`

`for` $t = T-1 \ldots 1$
    `for` $p = 1 \ldots P+1$
        $\beta'(t,p) = \max_{p'=1}^{P+1} \beta'(t+1,p') - \text{err}(t+1,p,p')$
    `end`
`end`

where

startof(t,p) is the best starting position of phone t given that it ends at p, which corresponds to the position at time $t-1$ of the traceback line in Figure 7.5, starting at position t,p.
$\text{startof}(t,p) = \text{argmax}_{p'=1}^{P+1} \alpha(t-1,p') - \text{err}(t,p',p)$

$\mathrm{err}(t, p_1, p_2)$ is the error contributed by hypothesis phone t, given that it aligns between positions p_1 and p_2 as in Figure 7.5.
see Section 7.3.4 for implementation details.

and variables are as follows:

$\beta'(t, p)$ is a negated error from t, p to the end of the sentence.
$\alpha'(t, p)$ is a forward negated error up to position t, p.
$\beta(t, p)$ is 1 if traceback from the end as in Figure 7.5 passes through the current position and 0 otherwise.
$\alpha(t)$ = a forward probability, always 1 in this case

With these quantities defined, average sentence error can be calculated at an arbitrary point t in the hypothesis sentence, as $\sum_{p=1}^{P+1} \alpha(t)\beta(t,p)(\alpha'(t,p)+\beta'(t,p))$. The ability to calculate this at any point in the sentence is important, as it will be used in the lattice version of the algorithm to calculate the average correctness of sentences passing through a local phone q.

This algorithm can be viewed as a forward-backward algorithm if the propagation of errors is considered in terms of transition probabilities. The transition between position p of phone q and position p' of following phone q' is 1 if the previous position p is the best choice, i.e. leads to the best phone error, and zero otherwise.

7.3.3 Calculating Raw Error in a lattice context.

The algorithm described above can be generalised to a lattice representation of sentences. The resulting algorithm is similar but not identical to the effect of individually aligning each sentence in the lattice. The alignment obtained is a constrained alignment in that the start of each phone in the lattice can in general align only to a single position $p = 1 \ldots P + 1$; there is not complete freedom to have different positions for different contexts.

The forward part of the algorithm calculates a forward probability $\alpha(q)$ for each arc, including the acoustic likelihood $p(\mathcal{O}|q)$ of the arc q itself; and the forward error $\alpha'(q, p)$ which includes the error up to but not including q itself.

```
for q = 1...Q,
    if q is a starting arc (no transitions to q)
        for p = 1...P+1
            α(q) = p(O|q)^κ
            α'(q,p) = err(q,1,p)
```

```
        end
    else
        for p = 1 ... P + 1
            α(q) = Σ_{q' preceding q} α(q') t^κ_{q'q} p(O|q)^κ
            α'(q,p) = (Σ_{q' preceding q} (max_{p'=1}^{P+1} α'(q',p') − err(q,p',p)) α(q') t^κ_{q'q} p(O|q)^κ) / (Σ_{q' preceding q} α(q') t^κ_{q'q} p(O|q)^κ)
        end
    end
end
```

Note that the maximum in the expression $\sum_{q' \text{ preceding } q} \max_{p'=1}^{P+1} \alpha'(q', p') - \text{err}(q, p', p)$ is taken separately for each preceding arc q'. This allows slightly more freedom in alignment than if the maximum had been taken outside the sum.

The backward part of the algorithm is more complex because the backward probabilities may not be the same: as seen in Figure 7.5, the forward probability to each position $1 \ldots P+1$ is always the same as all positions connect back to the start; but not all positions connect to the end so the backward probability may be zero. The algorithm is as follows. Note that $\beta(q, p)$ stores the likelihood from the end back to but not including the arc q, and $\beta'(q, p)$ stores the correctness back to and not including the contribution of q itself. This algorithm visits each arc q and propagates the backward likelihood and error *back* from q, rather than visiting each arc q and calculating the backward likelihood and error of q. This is for reasons of efficiency.

Q = #arcs in lattice
P = #sausage string positions

$\beta(q, p) \leftarrow 0$ for all q, p
$\beta'(q, p) \leftarrow 0$ for all q, p

```
for q = Q ... 1
    if q is an ending arc (no transitions from q) then
        β'(q, P+1) ← 0
        β(q, P+1) ← 1
    else
        for p = 1 ... P + 1
            if β(q,p) > 0
                for q' in arcs preceding q
                    p' = startof(q', q, p)
                    β_tmp ← t^κ_{q'q} p(O|q)^κ β(q,p)
                    β'_tmp ← β'(q,p) − err(q, p', p)
```

$$\beta'(q',p') \leftarrow \frac{\beta'(q',p')\beta(q',p') \; + \; \beta'_{\text{tmp}}\beta_{\text{tmp}}}{\beta(q',p')+\beta_{\text{tmp}}}$$
$$\beta(q',p') \leftarrow \beta(q',p') + \beta_{\text{tmp}}$$

```
                end
            end
        end
    end
end
```

where

startof(q',q,p) is the best starting position of phone q given that it ends at p, considering only the sentences passing through the previous arc q':
startof$(q',q,p) = \text{argmax}_{p'=1}^{P+1}\alpha'(q',p') - \text{err}(q,p',p)$

err(q,p_1,p_2) is the error contributed by hypothesis phone arc q, given that it aligns between positions p_1 and p_2 as in Figure 7.5; see Section 7.3.4 for implementation details.

A potentially confusing part of this algorithm is the part involving β_{tmp} and β'_{tmp}. The quantity $\beta'(q',p')$ is an average backward error, weighted by the probability of sentences passing through q'. Let us suppose some arc following arc q' is q. It is necessary to weight by the backward probability $t^{\kappa}_{q'q}p(\mathcal{O}|q)^{\kappa}\beta(q,p)$ in order to calculate this weighted average over sentences. The expression $t^{\kappa}_{q'q}p(\mathcal{O}|q)^{\kappa}\beta(q,p)$ includes the backward probability $\beta(q,p)$ of the following arc q, plus the data likelihood of q and the transition probability. The value β_{tmp} stores this backward likelihood contributed by the arc q being considered, and β'_{tmp} stores the correctness contributed by this particular following arc. If the backward likelihood $\beta(q,p)$ is zero for a particular q and p, which will usually be the case as only some positions connect to the end, $\beta'(q,p)$ will never be calculated and will remain zero.

The correctness $c(q)$ of an arc q is given by:

$$c(q) = \frac{\sum_{p=1}^{P+1}\alpha(q)\beta(q,p)(\alpha'(q,p)+\beta'(q,p))}{\sum_{p=1}^{P+1}\alpha(q)\beta(q,p)}. \tag{7.18}$$

The total error of the sentence c_{avg} is given by the sum over ending arcs q:

$$c^{r}_{avg} = \frac{\sum_{q \text{ in ending arcs}}\alpha(q)\alpha'(q,P+1)}{\sum_{q \text{ in ending arcs}}\alpha(q)}, \tag{7.19}$$

or equivalently the sum over starting arcs (useful as a check):

$$c^r_{avg} = \frac{\sum_{q \text{ in starting arcs}} \sum_{p=1}^{P+1} \beta(q,p)\alpha(q)(\beta'(q,p) + \alpha'(q,p))}{\sum_{q \text{ in starting arcs}} \sum_{p=1}^{P+1} \beta(q,p)\alpha(q)}. \tag{7.20}$$

Defining x as the total (scaled) probability $p(\mathcal{O}_r)$ of the speech file given the lattice, i.e.

$$x = \sum_{q \text{ in ending arcs}} \alpha(q) \tag{7.21}$$

the occupation probability for an arc q, which would normally given by $\frac{\alpha(q)\beta(q)}{x}$, is now given by $\frac{\sum_{p=1}^{P+1} \alpha(q)\beta(q,p)}{x}$. The scaled differential of the MPE objective function w.r.t. arc q is given by:

$$\begin{aligned} \gamma_q^{\text{MPE}} &= \gamma_q(c(q) - c^r_{\text{avg}}) \\ &= \sum_{p=1}^{P} \frac{\beta(q,p)\alpha(q)}{x}(\beta'(q,p) + \alpha'(q,p) - c^r_{\text{avg}}) \end{aligned} \tag{7.22}$$

7.3.4 Error from individual hypothesis phones

The function $\text{err}(q, p_1, p_2)$, required in the algorithm described above, gives the contribution to the sentence error from phone arc q beginning and ending in positions p_1 and p_2 of the sausage string, with these positions being between 1 and $P + 1$ as shown in Figure 7.5. Note that there are P sets of phones in the sausage string; the positions that vary between 1 and $P+1$ refer to the points at the beginning and ends of the sets of phones. Let the italicized "*position*" refer to the actual sets of phones $p = 1 \ldots P$ in the sausage string and un-italicized "position" refer to the locations $p = 1 \ldots P+1$ at the beginning and end of these sets of phones.

As mentioned in Section 7.3.1, each *position* indexed by p ($p = 1 \ldots P$) has n_p alternate phones r_{pa} for $a = 1 \ldots n_p$, and if the Boolean value e_p is true then sausage string *position* p also contains an "empty phone", i.e. a transition with no phone associated with it, due to shorter pronunciations of words.

The error $\text{err}(q, p_1, p_2)$ is calculated as follows. Define d as the number of potential deletions, i.e. the number of *positions* in the range $p = p_1 \ldots p_2 - 1$ which do not contain an empty phone ($\neg e_p$). Define the condition $\text{corr}_{\text{nonempty}}$ as true iff the phone of arc q matches one of the sausage string *positions* $p = s_q \ldots s_r - 1$ not containing an empty phone ($\neg e_p$), and $\text{corr}_{\text{empty}}$ if it matches one of the *positions* containing an empty phone.

```
if q is a silence phone, or another non-scored phone
    if p_2 < p_1
        err(q, p_1, p_2) = ∞ (path not allowed)
    else
        err(q, p_1, p_2) = d (return the number of potential deletions)
    end
else

    if p_2 < p_1
        err(q, p_1, p_2) = ∞ (path not allowed)
    else if d = 0 and corr_empty
        err(q, p_1, p_2) = 0 (correct phone, no deletions)
    else if d = 0 and ¬corr_empty
        err(q, p_1, p_2) = I (e.g., 0.9 or 1: i.e. one insertion)
    else if d > 0 and ¬corr_nonempty
        err(q, p_1, p_2) = d − 1 (d − 1 deletions, 1 correct)
    else (d > 0 and ¬corr_nonempty)
        err(q, p_1, p_2) = d ( d − 1 deletions and 1 substitution; or d deletions)
    end
end
```

Note that this algorithm assumes that $I \leq 1$, as the variable I was introduced only in order to reduce the significance of insertions.

Efficiency for exact MPE

The algorithm as described in the last few pages is not very efficient, because for each phone q there are two nested iterations over the position $p = 1 \ldots P$, inside which is a call to the function $\mathrm{err}(q, p, p')$ which itself takes linear time in the difference between p and p'. Therefore, the algorithm without pruning can take time $O(QP^3)$. There are three ways that this is sped up.

- The inner loop over p is combined with the calculation of the function $\mathrm{err}(q, p, p')$ to make each call to $\mathrm{err}(q, p, p')$ $O(1)$ rather than $O(P)$.
- The time information in the numerator (correct sentence) and denominator lattices is used to obtain lower and upper limits on the positions p which each arc q might occupy, and these limits are extended by (in this case) a margin of 4 on each side so that the iterations over sausage string position sp never include more than about 10 different values of p.

- All lattice arcs which are less likely than a specified limit (around 0.0002) are pruned away.

These optimisations taken together make the additional calculations used in exact MPE an insignificant fraction of the total calculation of discriminative training, much of which is taken up in doing within-arc forward-backward calculations to calculate the within-arc data likelihoods $p(\mathcal{O}|q)$ for each q.

7.4 MPE optimisation: further details

7.4.1 I-smoothing

As explained above, during the alignment (or Estimation) phase of MWE/MPE optimisation two sets of statistics are gathered: γ_{jm}^{num}, $\theta_{jm}^{\text{num}}(\mathcal{O})$ and $\theta_{jm}^{\text{num}}(\mathcal{O}^2)$ which correspond to the numerator statistics of MMI, and a similar set of statistics with the superscript "den" which correspond to the denominator statistics of MMI. These are both derived from a single lattice of recognised training data in a process which also requires the correct transcription in a phone-marked lattice form for purposes of calculating correctness of phones. The application of the technique of I-smoothing to the update process requires a third set of statistics, which are written with the superscript "mle". These are derived from the alignment of the correct utterance in the same way as for ML estimation (or for the numerator of MMI training). The following changes to the statistics are made prior to the Extended Baum-Welch update, replacing Equations 4.24 to 4.26:

$$\gamma'^{\text{num}}_{jm} = \gamma^{\text{num}}_{jm} + \tau \tag{7.23}$$

$$\theta'^{\text{num}}_{jm}(\mathcal{O}) = \theta^{\text{num}}_{jm}(\mathcal{O}) + \frac{\tau}{\gamma^{\text{mle}}_{jm}}\theta^{\text{mle}}_{jm}(\mathcal{O}) \tag{7.24}$$

$$\theta'^{\text{num}}_{jm}(\mathcal{O}^2) = \theta^{\text{num}}_{jm}(\mathcal{O}^2) + \frac{\tau}{\gamma^{\text{mle}}_{jm}}\theta^{\text{mle}}_{jm}(\mathcal{O}^2) \tag{7.25}$$

A similar change is required to the equations for smoothing of weights and transitions, which is discussed for the MMI case in Section 4.4.1. However, experiments reported in Section 8.17.1 show that this makes no difference in practice.

I-smoothing is equivalent to combining the objective function with the log prior distribution equal to $Q(\tau^I, \tau^I\mu_{\text{prior}}, \tau^I(\sigma^2_{\text{prior}} + \mu^2_{\text{prior}})|\mu_{jm}, \sigma^2_{jm})$ for each Gaussian m of state j (considering only one-dimensional Gaussians for simplicity), where μ_{prior} and σ^2_{prior} will equal the Maximum Likelihood estimates of the parameters of that Gaussian, τ^I is a value set by hand (typically 50) and $Q(\tau^I, \tau^I\mu_{\text{prior}}, \tau^I(\sigma^2_{\text{prior}} + \mu^2_{\text{prior}})|\mu, \sigma^2)$ is the log likelihood of τ^I points of data drawn from a distribution with mean μ_{prior} and variance σ^2_{prior}.

I-smoothing is discussed more thoroughly in Section 4.3.2. As reported in the following chapter, the use of I-smoothing (i.e. priors over Gaussian parameter values) is essential if MPE is to give improvements over MMI training.

7.4.2 Miscellaneous details of MPE training

In the algorithm described in Section 7.2.2, the value of $\gamma_q^{\mathrm{MPE}} = \gamma_q(c(q) - c^r_{\mathrm{avg}})$ is accumulated for each phone arc q in the lattice; but for reasons of efficiency, in the current implementation the sum of the value of γ_q^{MPE} over all arcs of a particular phone HMM with a particular start and end time, are added together and the arc is then treated as a single arc for purposes of accumulating data (a similar optimisation is used for calculating the within-arc likelihood $p(\mathcal{O}|q)$ for such duplicated arcs). This will sometimes affect the statistics accumulated if the values of γ_q^{MPE} for that identical group of arcs differ in sign, but will not affect the fixed point of the update formula.

As with MMI, transition statistics are required to update the transition matrices; rows of the transition matrix are updated as for Gaussian weights, as described in Section 4.4. The method of accumulating the transition statistics has not been described here but it is straightforward by analogy with MLE training, and involves a sum over arcs very similar to the one used for Gaussian occupation counts in Equations 7.7 to 7.12.

With MPE more iterations are generally required before the lowest WER is reached, than for MMI training. Around 8 iterations are generally required with the smoothing constant E set to the normal value of 2, as opposed to the MMI case where optimal WER may be reached after around 4 iterations. (A value of $E = 1$ or $E = 2$ is generally used for MMI).

For diagnostic purposes, the value of the MPE criterion is reported relative to the number of phones in the reference transcript. The value of c_{avg} (i.e., the MPE criterion for a given file) is summed over all files, and this value is divided by the total number of phones in all the reference transcripts (this might involve making an arbitrary decision about which reference pronunciations to use). The result will be less than 1, and is comparable to the accuracy on the training data. For exact MPE, the criterion reported is the sum of sausage string lengths P plus the summed average negated errors c^r_{avg}, all divided by the sum of sausage string lengths P. This will give a value between 0 and 1.

Chapter 8

Experiments with MPE Training

This chapter presents experiments relating to MPE training, including comparisons with MMI. The experimental conditions are as given in Section 5.3.1.

Section 8.1 compares MPE and MMI training. Section 8.2 investigates the effect of the size of training lattices. Section 8.3 investigates how the improvement from MPE training changes with size of training set. Section 8.4 investigates how the improvement changes with changing HMM set complexity. Section 8.5 investigates how improvement changes as the ratio of data to HMM set size changes. Section 8.7 investigates the optimal value of the probability scale κ. Section 8.8 investigates the optimal speed of training, as controlled by the smoothing constant E. Section 8.9 investigates the best language model to use for the training lattices. Section 8.11 investigates the optimal value of the I-smoothing constant τ^I. Section 8.12 investigates the effect of doing I-smoothing on a dimension-specific basis. Section 8.13 examines the possibility of having the smoothing constant E varying with iteration, which is suggested by analogy with Generalised Probabilistic Descent (GPD). Section 8.14 compares exact and approximate MPE. Section 8.15 compares MWE and MPE training. Section 8.17 reports work on miscellaneous other details of MPE training. Section 8.18 discusses results on the combination of discriminative training with other techniques.

8.1 MPE vs. MMI

This section compares three techniques: MMI, I-smoothed MMI and MPE. The techniques are tested on four different corpora: Switchboard, Broadcast News, North American Business News (NAB), and Resource Management.

Training Type (training iteration)	WER Training Subset			WER Test	
	Full bg	Lat bg	Lat ug	eval98	eval97sub
MLE baseline	26.3	26.0	41.8	46.6	46.0
MMI E=2,τ^I=0 (4)	18.6	19.4	30.1	44.3	43.9
MMI E=1,τ^I=200 (6)	19.7	20.3	32.2	43.8	43.1
MPE E=2,τ^I=50 (8)	20.6	20.7	27.9	43.1	42.1

(a) h5train00sub, 68h subset

Training Type (training iteration)	WER Training Subset			WER Test	
	Full bg	Lat bg	Lat ug	eval98	eval97sub
MLE baseline	30.1	29.8	47.2	45.6	44.4
MMI E=2,τ^I=0 (8)	23.2	23.7	37.7	41.8	41.2
MMI E=2,τ^I=100 (8)				41.6	41.0
MMI E=1,τ^I=200 (8)	22.2	23.0	35.8	41.4	40.5
MPE E=2,τ^I=100 (8)	23.9	23.9	34.4	40.8	39.8

(b) h5train00, full 265h train

Table 8.1: Training & test WERs for MMI, I-smoothed MMI and MPE for (a) 68 hour and (b) 265 hour training sets.

(a) Minitrain, 18h subset (12 mixture component system)

Training Type (training iteration)	WER Training Subset		WER Test
	Lat bg	Lat ug	eval97sub
MLE baseline	25.6	38.29	50.6
MMI, E=2 (4)	17.80	24.69	50.2
MMI, E=2, τ^I=100 (6)	19.45	28.49	49.6
MPE, E=2.0,τ^I=100 (8)	20.31	25.50	48.1

Table 8.2: MMI vs. I-smoothed MMI and MPE using unigram training lattices on Switchboard: Minitrain.

8.1.1 Experiments on Switchboard

Table 8.1 shows both the training and test WERs for training on a) the 68 hour and b) the full 265 hour training set for standard MMI, MMI with I-smoothing and MPE with I-smoothing. In both cases, the test-set WER varies according to the relation I-smoothed MPE $<$ I-smoothed MMI $<$ MMI. For larger amounts of data (not Minitrain), it is MPE that gives the greatest reduction in training set WER on the unigram lattices on which the system is trained. However, it does not give as large a reduction in training set WER as MMI when tested with a bigram language model; this shows that MPE may be more specific to the language model used, at least as far as its effect on the training set error is concerned.

Table 8.2 shows experiments on the Minitrain subset of Switchboard which con-

firm the sequence MPE < I-smoothed MMI < MMI as regards the test-set WER.

8.1.2 Experiments on Broadcast News

Train setup (# iters)	F0	F1	F2	F4	F5	FX	Avg	%rel
MLE	11.6	26.2	38.7	24.6	24.8	55.4	29.6	impr
MMI E=1, τ^I=0 (4)	12.0	24.4	34.5	22.8	23.4	51.8	27.9	5.7%
MMI E=1, τ^I =100 (4)	11.1	24.4	35.6	22.8	22.6	52.7	27.8	6.1%
MPE E=2, τ^I =50 (8)	10.6	22.9	33.7	21.4	22.8	48.9	26.2	11.5%

Table 8.3: Comparison of discriminative training criteria on Broadcast News.

Table 8.3 compares the three objective functions on the Broadcast News corpus, tested on the 1996 partitioned evaluation data with a trigram language model with 15 million n-grams. See Table A.4 for the meaning of the various focus conditions. In this case I-smoothed MMI is only slightly better than MMI, but MPE gives twice as much improvement, a significant difference, and is better than MMI (and ML) on all of the focus conditions.

8.1.3 Experiments on Resource Management

	Test set					Relative
Criterion used (iter)	feb'89	feb'91	sep'92	oct'89	all	Improvement
MLE	2.9	3.0	6.8	3.8	4.1	–
MMI E=2, τ^I=0 (4)	2.8	2.6	6.4	3.5	3.8	7.5%
MMI E=2, τ^I=100 (4)	2.7	2.9	6.5	3.5	3.9	5.0%
MPE, E=2, τ^I=50 (6)	2.8	2.8	6.4	3.8	4.0	2.5%

Table 8.4: Comparison of discriminative training criteria on Resource Management.

Table 8.4 gives test-set results for MMI, I-smoothed MMI and MPE training on Resource Management. In this case MPE gives less improvement than MMI. As discussed in Section 8.5, this result is not surprising since the amount of improvement from MPE is related to the amount of training data available per Gaussian. There is very little training data available per Gaussian for this HMM set size on Resource Management, and at this ratio of training data to Gaussians MPE normally gives little or no improvement (see Section 8.5).

Criterion used (iter)	%WER			avg rel % impr
	csrnab1_dev	csrnab1_eval	avg	
MLE baseline	9.34	9.80	9.57	
MMI ug, E=1 (4)	8.80	9.40	9.10	4.9%
MMI ug, E=1, τ^I=100 (4)	8.89	9.51	9.20	4.0%
MPE 12-mix, ug, E=2, τ^I=50 (8)	8.70	9.29	8.99	6.3%

Table 8.5: Comparison of discriminative training criteria on NAB: 12 Gaussians/state.

Lattice Ident	Lattice Depth	Prune Thresh
Baseline	124	-
Pruned-1	34	100
Pruned-2	8.4	50
Pruned-3	4.8	25

Table 8.6: Characteristics of different denominator lattices used for h5train00sub training.

8.1.4 Experiments on NAB/Wall Street Journal

Table 8.5 shows test results for models with 12 Gaussians per state trained and tested on the clean speech channel (Channel 1) of the NAB database. As before, MPE outperforms MMI (but not significantly; the 95% significance interval is about 0.3% (see Table 5.1). In this case, I-smoothing with MMI does not improve results relative to MMI.

8.1.5 Summary of comparisons of MPE and MMI

In summary, on the three larger corpora where there is a reasonably large amount of training data for each Gaussian to be trained, MPE gives an improvement over MMI. The difference between MPE and MMI is explored further in Section 8.5.

8.2 Effect of lattice size

In order to test the effect of lattice sizes for MPE training, the lattices from the Switchboard 65h-trained (h5train00sub) system were further pruned to three different levels, using pruning beams of 100, 50 and 25 (in $\log_e$ units) respectively. The depths of these lattices have been given in Table 5.8.

	Iteration					Degradation
	0	1	2	3	4	(it 4)
MMI unigram, $\kappa = 1/12$, E=1, Baseline	46.6	44.9	44.2	44.3	44.4	0.0
MMI unigram, $\kappa = 1/12$, E=1, Pruned-1	46.6	44.9	44.3	44.4	44.6	+0.2
MMI unigram, $\kappa = 1/12$, E=1, Pruned-2	46.6	44.9	44.3	44.9	45.8	+1.4
MMI unigram, $\kappa = 1/12$, E=1, Pruned-3	46.6	45.0	44.6	45.5	47.0	+2.6
	0	2	4	6	8	(it 8)
MPE unigram, $\kappa = 1/12$, E=2, Baseline	46.6	44.3	43.6	43.3	43.1	0.0
MPE unigram, $\kappa = 1/12$, E=2, Pruned-1	46.6	44.3	43.6	43.2	43.1	0.0
MPE unigram, $\kappa = 1/12$, E=2, Pruned-2	46.6	44.6	44.0	43.6	43.5	+0.4
MPE unigram, $\kappa = 1/12$, E=2, Pruned-3	46.6	44.7	44.4	44.3	44.3	+1.2

Table 8.7: Effect of varying lattice size on switchboard: %WERs on eval98

Table 8.7 shows the effect of varying the pruning of lattices used for MMI and MPE training. In both MMI and MPE training, a reduction in lattice size degrades performance, but MPE is less sensitive to a reduction in lattice size.

8.3 Effect of training set size

An experiment was performed in which a small HMM set (6 Gaussians/state, 3088 states) which was initially trained on the 18h Minitrain subset of Switchboard data, was trained on widely varying amounts of data from 1.125h to 265h, to see how the improvement from MPE training varies with training set size. The subsets of training data are the full 265h set h5train00, the 68h set h5train00sub, the 18h Minitrain subset and randomly chosen subsets of Minitrain of decreasing size. The absolute test-set %WERs from these experiments are given in Table 8.8.

Comparisons between the different training criteria for varying amounts of training data are given in Table 8.9, which compares MMI, I-smoothed MMI and MPE with MLE. The result is that both MPE and MMI training outperform MLE only when there is enough training data available; but MPE outperforms MMI for all amounts of training data. When the training data falls below some point between 4.5h and 2.25h (about 60 frames/Gaussian), I-smoothed MMI begins to outperform I-smoothed MPE. This indicates that at very small amounts of training data MMI may generalise better, but the absolute improvement from I-smoothed MMI in the range where it outperforms MPE is so small (0.6%) that for all practical purposes, as far as these experiments are concerned MPE can be considered the criterion of choice.

The results in Table 8.8 also have implications for the difference between using bigram and unigram training lattices; these are discussed in Section 8.9.

	# its	Amount of training data 1.125h	2.25h	4.5h	9h	18h	68h	265h
Avg Frames/Gaussian		22	44	87	174	349	1320	5150
Gauss/Hour		16k	8k	4120	2060	1030	272	70
Original HMM set (trained on 18h)						57.6		
MLE	4	77.8	67.3	62.0	59.3	57.6	55.9	55.7
MMI(bg, E=2) from 18h HMM set	4	58.9	58.0	58.1	57.1	56.7	54.3	54.0
MMI(bg, E=2) after ML	4	80.6	69.2	63.2	59.4	56.9	53.7	53.2
MMI(ug, E=2) after ML	4	80.8	69.2	62.7	59.6	56.9	53.4	52.6
MMI(ug, E=2,τ^I=100) after ML	6	78.2	66.7	61.6	58.0	56.1	54.0	52.8
MPE(bg, E=1.5,τ^I=50) after ML	6	80.4	67.9	60.8	57.7	55.4	51.9	50.0
MPE(ug, E=1.5,τ^I=50) after ML	6	79.5	66.7	60.7	57.2	54.6	52.2	50.6

Table 8.8: Varying amounts of training data on Switchboard. %WERs tested on eval97sub using fast single-pass decoding with a bigram LM; small (3088 state, 6-Gaussian/state) HMM set.

	Amount of training data
Absolute changes in %WER:	1.125h	2.25h	4.5h	9h	18h	68h	265h
MMI (ug) vs. MLE	+3.0	+1.9	+0.7	+0.3	-0.7	-2.5	-3.1
MMI (ug, τ^I=100) vs. MLE	+0.4	-0.6	-0.4	-1.3	-1.5	-1.9	-2.9
MPE (ug, τ^I=50) vs. MLE	+1.7	+0.6	-1.3	-2.1	-3.0	-3.7	-5.1

Table 8.9: Varying the amount of training data for ML,MMI and MPE: Switchboard. Comparisons between different criteria: absolute %WER change.

	# Gaussians / state						
	1	2	4	12	16	24	32
Avg Frames/Gaussian	3710	1860	928	309	232	155	116
Avg Gauss/hour	96	194	388	1160	1552	2330	3100
	Avg %WER on csrnab1_{dev,eval}						
MLE	14.70	12.53	10.86	9.57	9.38	9.23	9.19
MMI ug, E=1 (iter 4)	12.26	10.72	10.01	9.10		9.03	8.97
MMI ug, E=1, τ^I=100 (iter 4)	12.18	10.81	9.82	9.20		8.86	9.05
MPE ug, E=2, τ^I=50 (iter 8)	12.00	10.67	9.61	9.00	8.86	8.57	8.81

Table 8.10: Varying the number of Gaussians per mixture: NAB SI-284 (66h) discriminative training.

Paradoxically, when MMI training is started directly from the baseline HMM set trained on the 18h Minitrain subset, without prior ML training on the chosen subset of data, (this experiment is labeled "from 18h ML" in Table 8.8) MMI gives much less degradation when training on small amounts of data than ML training on the same amount of data. This seems to be because with small amounts of training data the MMI training algorithm quickly "learns" the data and the parameters do not change further, retaining a memory of the original ML system trained on 18 hours of data. This is due to the nature of the update equations used and the way the smoothing constant D is set: there is very little change to the parameters once the training data becomes correctly recognised.

To summarise the results of this experiment: in this case, MPE training seems to outperform MLE when more than about 60 frames of data per Gaussian is available; MMI starts to outperform MLE after about 200 frames per Gaussian. MPE always outperforms standard MMI, and in the region where discriminative training is worthwhile it always outperforms I-smoothed MMI.

8.4 Effect of varying Gaussians/state

Table 8.10 gives discriminative training results for versions of an HMM set with widely varying numbers of Gaussians per state, tested on the American Business News corpus. All HMM sets have 6399 tree-clustered states.

Table 8.11 gives comparisons between various different criteria across the different systems. The third line compares MPE and MMI; MPE seems to outperform standard and I-smoothed MMI for all ratios of training data to size of HMM set. I-smoothed MMI does not give a consistent improvement over MMI in this case.

These results are in agreement with the conclusion from previous experiments on Switchboard with varying amounts of training data (Table 8.8) that MPE

	# Gaussians / state						
	1	2	4	12	16	24	32
	%WER change on csrnab1_{dev,eval}						
MMI vs. MLE	-2.44	-1.81	-0.85	-0.47		-0.20	-0.22
MMI, τ^I=100 vs. τ^I=0	-0.08	+0.09	-0.19	+0.10		-0.17	+0.08
MPE (E=2) vs. MMI	-0.26	-0.05	-0.40	-0.10		-0.29	-0.16
MPE (E=2) vs. MLE	-2.70	-1.86	-1.25	-0.57	-0.52	-0.66	-0.38
MPE (E=1) vs. MMI	+0.72	+0.58	-0.12	-0.10		-0.25	-0.20

Table 8.11: Varying Gaussians per mixture: comparisons between different criteria.

	ID	#states	# mix /state	train data (hr)	#frames /gauss	Test set
Switchboard	SW1	6165	16	265	967	eval98
Switchboard	SW2	6165	12	68	330	eval98
Switchboard	SW3	3088	6	265	5150	eval97sub
Switchboard	SW4	3088	6	68	1320	eval97sub
Switchboard	SW5	3088	6	18	350	eval97sub
Switchboard	SW6	3088	6	4.5	88	eval97sub
Switchboard	SW7	3088	6	1.125	22	eval97sub
WSJ/NAB	WSJ1	6399	12	66	309	csrnab1_{dev,eval}
WSJ/NAB	WSJ2	6399	4	66	928	csrnab1_{dev,eval}
WSJ/NAB	WSJ3	6399	1	66	3713	csrnab1_{dev,eval}
Resource Management	RM1	1582	6	3.8	144	oct89,feb91,feb92,sep92
Broadcast News	BN1	6684	12	72	323	bndev96

Table 8.12: Training set sizes and HMM set sizes for various setups

gives an improvement above about 60 frames of data per Gaussian; the limit is not reached here, but MPE continues to give small improvements down to the experimental limit of 116 frames of data per Gaussian. Small improvements are also obtained with MMI training down to 116 frames per Gaussian, unlike in the Switchboard case (Table 8.8) where MMI stopped being effective around 200 frames per Gaussian.

8.5 Ratio of data to HMM set size

The ratio of the length of training data to the HMM set size (in Gaussians) is a good predictor of the amount of improvement to be gained from discriminative training. This section compares results from a number of different experiments,

ID	E	#iters	%WER MLE-MMI	%relative improvement
SW1	2	8	45.6-41.8	8.3
SW2	2	4	46.6-44.3	4.9
SW3	2	4	55.7-52.6	5.6
SW4	2	4	55.9-53.4	4.5
SW5	2	4	57.6-56.9	1.2
SW6	2	4	62.0-62.7	-1.1
SW7	2	4	77.8-80.8	-3.9
WSJ1	1	4	9.57-9.10	4.9
WSJ2	1	4	10.86-10.01	7.8
WSJ3	1	4	14.7-12.26	16.6
RM1	2	4	4.13-3.82	7.5
BN1	1	4	29.6-27.9	5.7

Table 8.13: MMI training on various corpora showing relative improvements.

ID	E	τ^I	#iters	%WER MLE-MMI	%relative improvement
SW1	1	200	8	45.6-41.4	9.2
SW2	1	200	6	46.6-43.8	6.0
SW3	2	100	4	55.7-52.8	5.2
SW4	2	100	4	55.9-54.0	3.4
SW5	2	100	4	57.6-56.1	2.6
SW6	2	100	4	62.0-61.6	0.6
SW7	2	100	4	77.8-78.2	-0.5
WSJ1	1	100	4	9.57-9.20	3.9
WSJ2	1	100	4	10.86-9.82	9.6
WSJ3	1	100	4	14.7-12.18	17.1
RM1	2	100	4	4.13-3.89	5.8
BN1	1	100	4	29.6-27.8	6.1

Table 8.14: I-smoothed MMI on various corpora showing relative improvements.

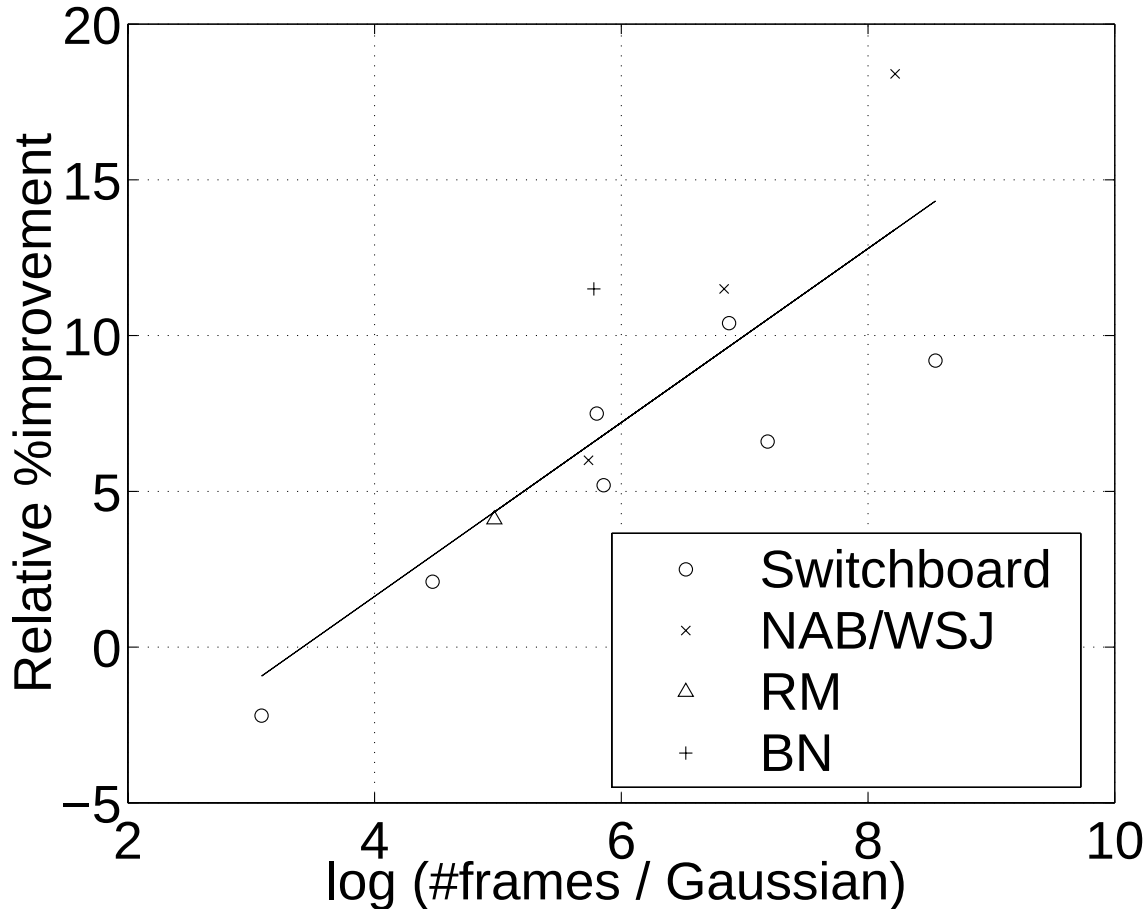


Figure 8.1: MPE: relative improvements in %WER over various corpora, predicted by $\log_e$(#frames/Gaussian)

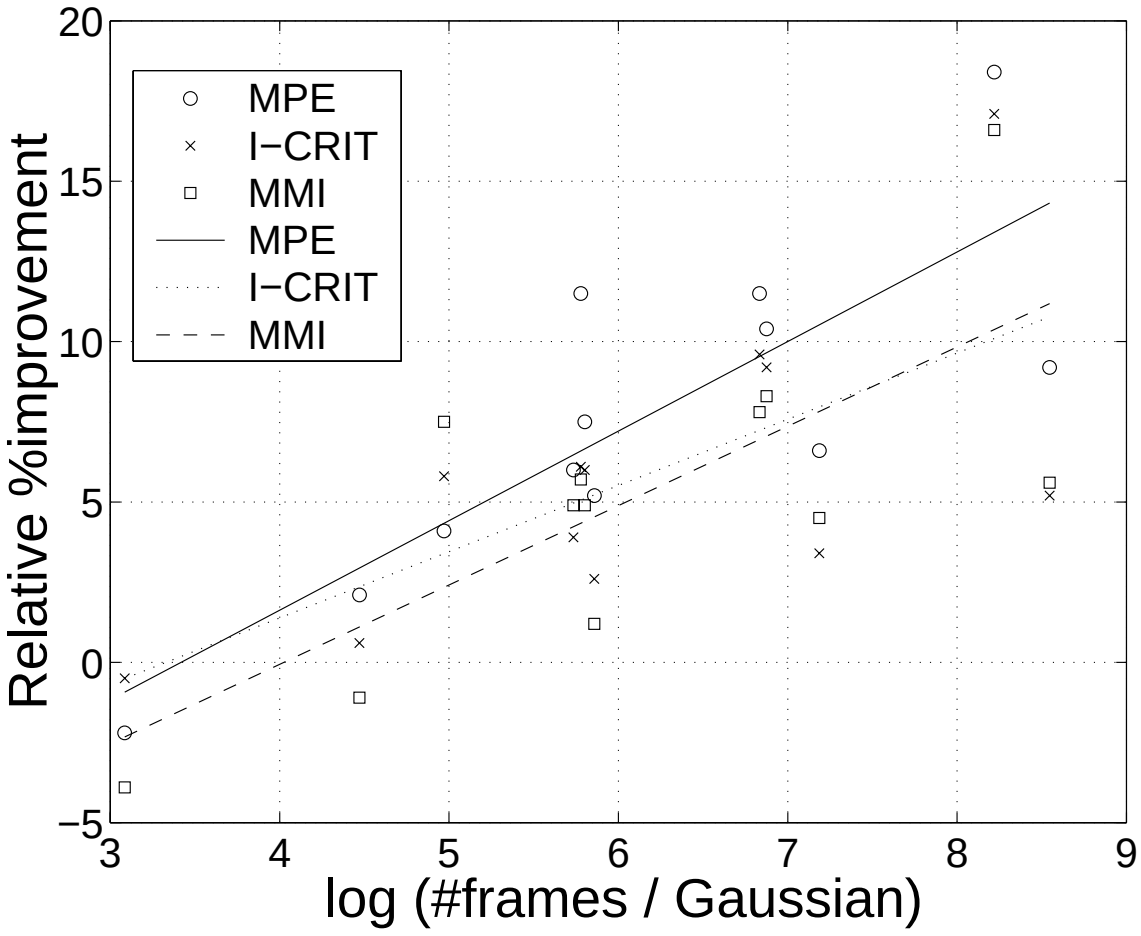


Figure 8.2: Relative improvement from MPE, I-Crit (I-smoothed MMI) and MMI on various corpora and setups, plotted against the $\log_e$(#frames/Gaussian) of training data.

ID	E	τ^I	#iters	%WER MLE-MPE	%relative improvement
SW1	2	100	8	45.6-40.8	10.5
SW2	2	50	8	46.6-43.1	7.5
SW3	1.5	50	6	55.7-50.6	9.2
SW4	1.5	50	6	55.9-52.2	6.6
SW5	1.5	50	6	57.6-54.6	5.2
SW6	1.5	50	6	62.0-60.7	2.1
SW7	1.5	50	6	77.8-79.5	-2.2
WSJ1	2	50	8	9.57-9.00	6.0
WSJ2	2	50	8	10.86-9.61	11.5
WSJ3	2	50	8	14.7-12.0	18.4
RM1	2	50	6	4.13-3.96	4.1
BN1	2	50	8	29.6-26.2	11.5

Table 8.15: MPE training on various corpora showing relative improvements

in which MMI, I-smoothed MMI and MPE training are tested on a number of different corpora. The training setups, giving different amounts of training data and sizes of training set, are shown in Table 8.12. Tables 8.13, 8.14, and 8.15 give the specific training setups (number of iterations, amount of I-smoothing, smoothing constant E) and improvements for the three different criteria.

Regression analysis was used to predict the relative improvement based on the amount of training data and size of the HMM set. Figure 8.1 shows for MPE the relative improvement predicted by the $\log_e$ (#frames of training data per Gaussian). The average squared error with this method is 9.00; adding the log (number of Gaussians) to the regression analysis as a second predictor variable only reduces this to 8.50. The #frames/Gaussian seems to predict most of the variation in relative improvement.

Figure 8.2 compares MPE, I-smoothed MMI and MMI for the various different corpora and HMM sets. The regression analysis shows that MPE tends to outperform MMI at all relative amounts of training data, but at very small amounts of training data I-smoothed MMI can be better than MPE. However, the point where the two meet is where the improvement is zero and there would be no point in discriminative training in any case. I-smoothed MMI is approximately the same as MMI for large amounts of training data but better than MMI at small ratios.

The predicted relative improvement as a function of $L = \log_e$ (#frames/Gaussian), is:

- MPE: $-9.54 + 2.79L$.

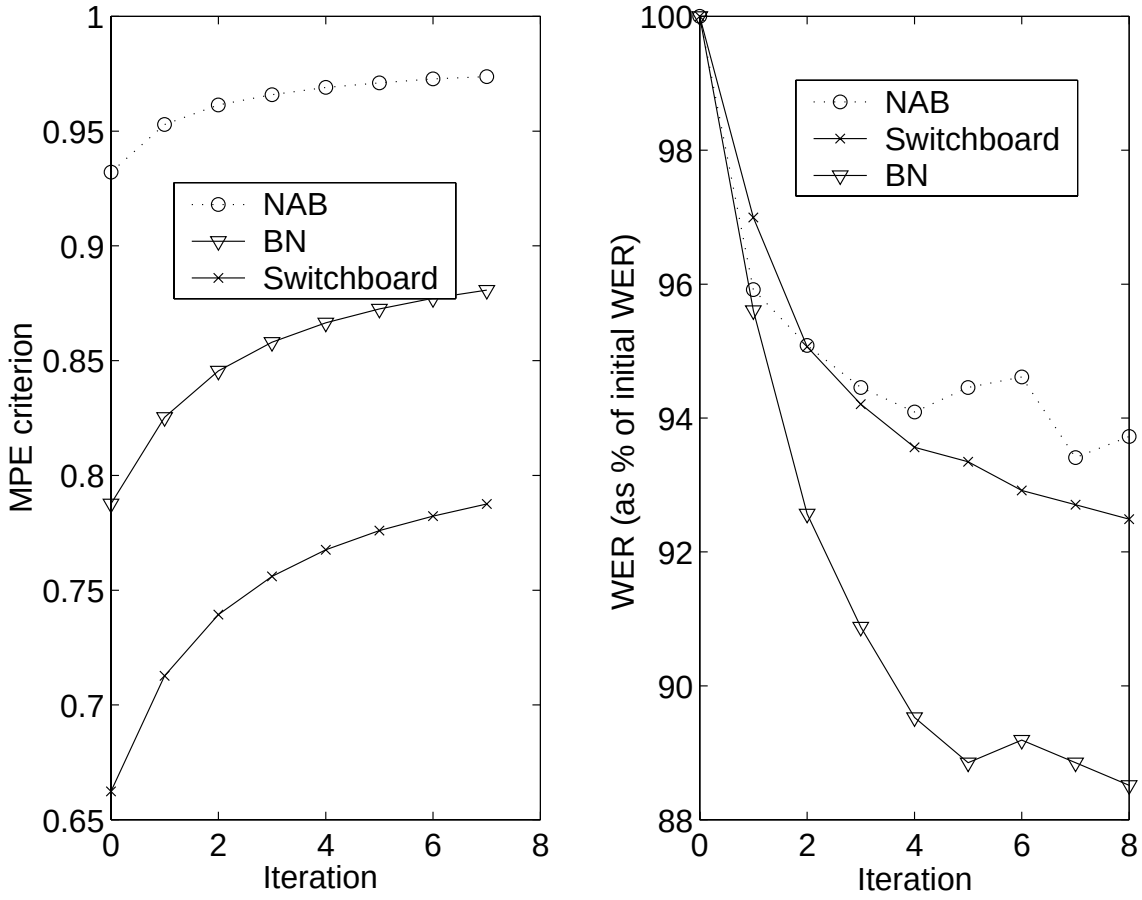


Figure 8.3: Convergence of MPE criterion and WER for three corpora

- MMI: $-9.95 + 2.47L$.
- MMI+I-smoothing: $-6.87 + 2.06L$.

This does not predict the relative improvement very exactly (the standard deviation is about 3% relative change in %WER) but may be useful in deciding whether to use discriminative training.

8.6 Convergence of training

Figure 8.3 displays the convergence of the MPE criterion (normalised by the number of words in the reference transcript), and the changes in test-set WER as training proceeds. This is for the standard HMM set sizes and training sets for the three major corpora, with the smoothing constant E set to 2 in all cases. As can be seen, there is a smooth optimisation of the criterion and a fairly smooth

	Test %WER on eval98 (and ins/del ratio)		
	Probability scale κ		
	1/24	1/12	1/6
MLE baseline (no scale)		46.6 (0.25)	
MPE ug, E=2, τ^I = 50 (8 its) ip=0.0	43.3 (0.26)	43.1 (0.34)	44.5 (0.39)
ip=0.2	43.2 (0.36)		
ip=0.4	43.7 (0.48)		

(a) h5train00sub, 68h

	Test %WER on eval98 (and ins/del ratio)	
	Probability scale κ	
	1/12	1/6
MLE baseline (no scale)	45.6 (0.27)	
MPE ug, E=2, τ^I = 100 (4 its) ip=0.0	41.6 (0.28)	42.7 (0.33)

(b) h5train00, 265h

Table 8.16: %WERs on Switchboard, tested on eval98, for different probability scales κ and insertion penalty ip for MPE.

decrease in WER. For standard MPE experiments, training is continued for 8 iterations; as can be seen from the graphs, the test-set WER is beginning to converge after 8 iterations.

8.7 Effect of probability scale κ

The effect of the probability scale on MMI training is covered in Section 5.3.10.

Table 8.16 shows the results of varying the probability scale κ used in MPE training. The default value is the inverse of the normal language model scale, 1/12 in the case of Switchboard training. This appears to be close to the optimal value for both MMI and MPE training under these test conditions (note that this will vary as a function of the frame rate of the feature vectors, and other factors). For the full training set, only a larger probability scale is tried since it is expected that the optimal value should be larger (closer to ∞, which is more similar to recognition) as the training data is increased.

Note that as κ becomes smaller, more and more sentences in the denominator model begin to contribute significantly to the probability of the data. This reduces the variance of the updated parameters because the denominator statistics used in the update equations become for each Gaussian a sum over a larger number of parts of training data.

# its	ins pen	Test %WER on on csrnab1_{dev,eval} (ins/del ratio) Probability scale κ (default=1/15) 1/120	1/60	1/30	1/15	1/7.5
MLE (no scale))					9.57 (0.95)	
MPE (2)	0.0			8.94 (0.81)	9.06 (0.96)	9.13 (1.05)
(8)	0.0			9.04 (0.84)	9.00 (1.05)	9.24 (1.25)
(2)	0.85		8.80 (0.96)			
(8)	0.85		9.83 (1.31)			
(2)	1.25	8.93 (0.94)				
(5)	1.25	9.72 (0.90)				

Table 8.17: Varying probability scaling for MLE and MPE (12 mixcomp, ug, E=2, τ^I=50), on NAB.

It appears that there is an interaction between the probability scale and the ratio of insertions to deletions found in subsequent recognition. The insertion/deletion ratios are shown in Table 8.16 and they increase with the probability scale.

By adding an "insertion penalty" (*ip* in Table 8.16) at each phone transition during training, it is possible to correct for this and vary the scale κ without affecting the insertion/deletion ratio. However, even when the insertion penalty is optimised there is no improvement from varying the scale κ away from 1/12. (Note that the log insertion penalty is not scaled by κ, and that a positive value of *ip* makes an extra phone more likely).

Table 8.17 shows the effect of varying the scale κ for MPE training on NAB/Wall Street Journal. Better results can be achieved with a value of κ smaller than the default value of 1/15, but the optimum is reached earlier in training and the system seems more vulnerable to overtraining.

In summary, setting κ to the inverse of the normal language model scale, which tends to be around 12-15, is a safe approach. A smaller value of κ sometimes helps, but if a smaller value is used the system can overtrain more easily (so later iterations of training give worse results).

8.8 Effect of smoothing constant E.

The smoothing constant E controls the speed of training, and is generally set to 1 or 2. A smaller value leads to faster training. See Section 4.5.3 for an explanation of how E relates to the update equations.

Table 8.18 gives results for a range of values of E for MPE training on NAB. On this corpus the best test set WER after 4 iterations seem to be achieved with E=1 or 2, with 0.5 and 4 giving worse results. In this case the best test set WERs

	avg %WER on csrnab1_{dev,eval}					Value of MPE objf
	Iteration					
	2	4	6	8	10	7
MLE	9.57					0.917
MPE ug, $\kappa = 1/15$, E=0.5 τ^I=50	9.06	9.10	9.07	9.09		0.957
MPE ug, $\kappa = 1/15$, E=1, τ^I=50	9.00	9.05	8.92	9.00		0.958
MPE ug, $\kappa = 1/15$, E=2, τ^I=50	9.06	8.96	8.98	9.00	8.96	0.958
MPE ug, $\kappa = 1/15$, E=4, τ^I=50	9.14	9.06	9.06	8.87	8.99	0.956

Table 8.18: Varying E on NAB for MMI and MPE: 12 Gaussians/state.

	avg %WER on csrnab1_{dev,eval}					
	1-comp	2-comp	4-comp	12-comp	24-comp	32-comp
MLE	14.70	12.53	10.86	9.57	9.23	9.19
MPE ug, E=1, τ^I=50 (iter 8)	12.98	11.30	9.89	9.00	8.61	8.77
MPE ug, E=2, τ^I=50 (iter 8)	12.00	10.67	9.61	9.00	8.57	8.81
	MPE objf value					
MLE	0.883	0.893	0.902	0.917	0.932	0.939
MPE ug, E=1, τ^I=50 (iter 8)	0.931	0.938	0.946	0.9580	0.9662	0.9696
MPE ug, E=2, τ^I=50 (iter 8)	0.933	0.941	0.948	0.9581	0.9660	0.9693
[Difference E=2 vs. E=1]	0.002	0.003	0.002	0.0001	-0.0002	-0.0003

Table 8.19: E=1 vs. E=2 on North American Business News (NAB) for MPE with varying Gaussians/state (mixture components).

are obtained with the values of E that optimise the criterion best.

Table 8.19 compares E=1 and E=2 for MPE training on NAB using HMM sets with a varying number of Gaussians per state. There seems to be an interaction between the number of Gaussians per state, and the the value of the smoothing constant E. A larger value of E (for slower training) seems to work better for smaller HMM sets; with large HMM sets there is no consistent difference in WER between E=1.0 and E=2.0. It is still the case that the best error rate coincides with the highest value of the criterion, although there tends to be a higher error rate with E=1 than would be expected from the difference in criterion alone. If optimisation rates are equal, slower training (larger E) is best.

Table 8.20 compares training with E=1 and E=2 for MPE trained on the 65h subset of Switchboard training data. E=2 seems to work considerably better than E=1 in terms of both error rates and criterion. This is probably due to the speed of update being too fast, leading to overshoot of the parameters.

	Iteration %WER on eval98					MPE objf
	0	2	4	6	8	7
MLE	46.6					0.662
MPE ug, $\kappa = 1/12$, E=1, τ^I=50	46.6	43.9	43.4	44.1	44.6	0.781
MPE ug, $\kappa = 1/12$, E=2, τ^I=50	46.6	44.3	43.6	43.3	43.1	0.787

Table 8.20: E=1 vs. E=2 for Switchboard MPE training. Training set is h5train00sub (68h); 12 Gaussians/state.

	avg %WER on csrnab1_{dev,eval}				
	1-comp	2-comp	4-comp	12-comp	32-comp
MLE	14.70	12.53	10.86	9.57	9.19
MPE ug, E=2, τ^I=50 (iter 8)	12.00	10.67	9.61	9.00	8.81
MPE bg, E=2, τ^I=50 (iter 8)	12.27	10.89	10.37	9.24	9.53
Difference, ug vs. bg	-0.27	-0.22	-0.76	-0.24	-0.72

Table 8.21: Unigram vs. Bigram language models for training with MPE on North American Business News (NAB): varying Gaussians/state.

In summary, from the experiments reported here the best test set results tend to be obtained with the value of E that optimises the criterion fastest, or sometimes a slightly larger value. The best value of E depends on the corpus and the number of Gaussians per state in the HMM set (less Gaussians $\rightarrow$ larger E). In general E=2 is a good value, but for corpora with lower error rates (e.g, NAB) where larger HMM sets are used, E=1 may be better.

8.9 Language model for MPE: Unigram vs Bigram.

MPE training requires a language model to be used for the sentence likelihood $P(s)$ that appears in the MPE objective function. Only unigram and bigram language models are considered here: in Section 5.3.9, scaled-down unigram and zero-gram language models are also considered for MMI training but were not found to be generally useful; unpublished experiments have found scaled-down unigram and zero-gram language models to be an unpromising approach for MPE training. It was shown in [Schluter et al., 1999] that a unigram language model for MMI training gave better test-set results than a bigram, trigram or zero-gram language model, regardless of the testing LM.

	WER Training Subset			WER Test (ins/del)	
	Full bg	Lat bg	Lat ug	eval97sub	eval97sub ug-bg
MLE baseline	24.75	25.6	38.29	50.6 (0.17)	
MPE $E = 2$, $\tau^I = 50$ (iter 8) bg	16.95	17.01	30.56	48.6 (0.24)	
MPE $E = 2$, $\tau^I = 50$ (iter 8) ug	20.10	20.31	25.50	48.1 (0.28)	-0.5

Table 8.22: Unigram vs. Bigram for training on Switchboard; Minitrain (18h) training, 12-comp HMM set (WERs given with ins/del ratios in brackets).

	Amount of training data						
	1.125h	2.25h	4.5h	9h	18h	68h	265h
Change, ug vs. bg for MMI	+0.2	0.0	-0.5	+0.2	+0.0	-0.3	-0.6
Change, ug vs. bg for MPE	-0.9	-1.2	-0.1	-0.5	-0.8	+0.3	+0.6

Table 8.23: Unigram vs. bigram lattices for training on Switchboard, comparison of results from Table 8.8. Negative numbers mean unigram is better.

It should be noted that the scaling factors are applied to the language models as follows: the normal language model scale and word insertion penalties used in testing are applied to the n-gram language model when creating the lattice, and the resulting log likelihoods are later scaled down by the same scale κ which is applied to the acoustic probabilities in discriminative training. As a result the final language model is generally unscaled because κ is generally set to the inverse of the testing LM scale.

Table 8.21 compares training with bigram and unigram LMs on the NAB corpus, for varying numbers of Gaussians per state. A unigram language model appears to generalise better than bigram to the test set, at all numbers of mixture components. This may be because, with a unigram language model, a wider range of incorrect words have significant probability in the denominator lattice, which decreases the variance of the estimates of the MMI-updated parameters because the denominator statistics for each Gaussian include a wider range of examples.

Table 8.22 compares lattice training on the Switchboard corpus (minitrain subset) using a unigram or bigram language model. Test set results are 0.5% better with unigram training. Training set results are also given, showing that as expected the training set results improve more with the same language model that was used to train.

Table 8.23 shows the differences between bigram and unigram language models in both MMI and MPE training, on Switchboard for a wide range of training set sizes (1.125h to 265h) using a small HMM set. The experiments are described in Section 8.3 (see Table 8.8). The difference between unigram and bigram for MMI

does not show any consistent pattern, but for MPE training unigram lattices perform better for limited training data, while bigram lattices perform better where a large amount of training data is available. This is expected since a bigram language model simulates recognition more precisely than unigram, but a unigram may generalize better when there is limited training data. However the results should be treated with some caution since the 95% confidence interval for testing on eval97sub is in the region of 0.7 - 1.0% WER for typical experiments (Table 5.1).

Tables 8.24 and 8.25 compare bigram and unigram training on the 65h subset and the full training set on Switchboard. In both cases unigram lattices seem to give better performance, although the difference is small (0.3% and 0.2% absolute).

To summarise, a unigram language model almost always performs better than bigram, but in some experiments with a very small HMM set but large training set (i.e. plenty of training data per Gaussian) a bigram language model was better. Note that there may be some benefit in combining different LMs and different sets of lattices, as described in Section 8.17.5.

	%WER on eval98					Ins/del ratio (last iter)	Diff ug-bg (last iter)
MLE	46.6					0.27	
	Training iteration						
	0	2	4	6	8		
MPE κ =1/12, E=2, τ^I =50 bg	46.6	44.5	43.6	43.6	43.4	0.29	
ug	46.6	44.3	43.6	43.3	43.1	0.34	-0.3

Table 8.24: Unigram vs. bigram training on Switchboard: h5train00sub (68h) training, 12-comp HMM set.

	%WER on eval98					Ins/del ratio (last iter)	Diff ug-bg (last iter)
MLE	45.6					0.27	
	Training iteration						
	0	2	4	6	8		
MPE κ =1/12, E=2, τ^I =50 bg	45.6	43.2	41.3	40.8	40.6	0.26	
ug	45.6	42.5	41.3	40.6	40.4	0.32	-0.2

Table 8.25: Unigram vs. bigram training on Switchboard: h5train00 (265h) training, 16-comp HMM set.

	avg %WER on csrnab1_{dev,eval}					Value of objf
	Iteration					
	0	2	4	6	8	7
MPE ug, $\kappa = 1/15$, E=2.0 τ^I=50						
bg-ug lattices (default)	9.57	9.09	9.00	9.05	8.96	0.973
ug-ug lattices	9.57	8.85	8.72	8.78	8.74	0.964
MMI ug, $\kappa = 1/15$, E=2.9						
bg-ug lattices (default)	9.57	9.33	9.27	9.07	8.99	-0.0068
ug-ug lattices	9.57	9.32	9.11	8.96	8.93	-0.0079

Table 8.26: Training with unigram lattices created from rescoring bigram lattices (bg-ug) or unigram lattices (ug-ug); tested on NAB.

8.10 Generation of lattices

The unigram lattices used for the experiments presented above were generated in a two-stage process where initial lattices were created using a bigram language model and a unigram language model was used in a second pass to create phone-marked lattices with unigram LM probabilities. The exact pruning beams used in this process are given in Section A.3 of Appendix A. The effect of this method is that the unigram lattices will lack paths which had a very low bigram probability.

Experiments on NAB and Switchboard were performed, in order to examine the effect of creating unigram lattices using unigram language models right from the start. These are referred to as ug-ug lattices, as distinct from the bg-ug lattices in which a bigram LM was used on the first pass of recognition.

Table 8.26 compares these two types of lattices for MPE and MMI training. The lattices created from unigram recognition were about four times larger than the ones created with a bigram, and give better results for both MPE and MMI training; but not significantly so. The difference appears to be even smaller for more difficult tasks. Experiments performed in preparation for the 2003 NIST evaluation of a Switchboard system compared these two kinds of lattices for MPE training. The unigram-generated lattices were only about 30% larger than the bigram-generated ones, and the difference in recognition results was only about 0.1% absolute improvement (which could be due to the larger lattice size). The difference between NAB and Switchboard is possibly related to the fact that on Switchboard the difference in perplexity between unigram and bigram language models is much less.

8.11 Effect of I-smoothing constant τ^I.

(a) h5train00sub (68h)

Training Type	Train Subset WER		MPE	Test WER
	Full bg	Lat ug	objf	eval98
MLE baseline	26.3	41.8	0.66	46.6
MPE ug,E=2,τ^I=0 (iter 8)	25.5	28.5	0.80	50.7
MPE ug,E=2,τ^I=25 (iter 8)	20.0	26.2	0.81	43.1
MPE ug,E=2,τ^I=50 (iter 8)	20.6	27.9	0.79	43.1
MPE ug, E=2,τ^I=100 (iter 8)	21.6	29.9	0.77	43.3

(b) h5train00 (265h)

Training Type	Train Subset WER		Test WER
	Full bg	Lat ug	eval98
MLE baseline	30.1	47.2	45.6
MPE ug,E=2,τ^I=100 (8)	23.9	34.4	40.8
MPE ug,E=2,τ^I=50 (8)	24.0	32.7	40.4

Table 8.27: Different values of τ^I for I-smoothing of MPE on Switchboard.

Figure 8.27 (a) shows the effect of varying the I-smoothing constant τ^I for MPE training on the 68h subset of h5train00. The value of the MPE criterion (column 3) shows that higher values of τ^I (more smoothing) acts against optimisation of the MPE criterion, as would be expected. Training set results with the same language model used for training (unigram, column 2) show that MPE gives good training set improvements with all values of τ^I used. However, generalisation to the training set with a different language model or to the test set is very poor without any smoothing ($\tau^I = 0$), with a significant degradation in test-set performance relative to MLE. The best test-set performance is around the range $\tau^I = 25$ to 50.

Figure 8.27 (b) gives the same experiment on the full 265h training set. Again $\tau^I = 50$ performs better than $\tau^I = 100$.

	avg %WER on csrnab1_{dev,eval}
MLE	9.57
MPE $\tau^I = 25$, $E = 1$ (iter 8)	9.36
MPE $\tau^I = 50$, $E = 1$ (iter 8)	8.99

Table 8.28: Different values of τ^I for I-smoothing of MPE on NAB, 12 mix-comp.

Figure 8.28 gives results for MPE training with $\tau^I = 25$ and $\tau^I = 50$ on a 12

NAB system with 12 mixture components. As with previous experiments, the results are consistent with $\tau^I = 50$ being the best value for MPE training.

To summarise, τ^I=50 generally seems to be a suitable value.

8.12 Dimension-specific I-smoothing

In Section 4.3.5, a method is described for setting the τ^I values for I-smoothing in a dimension-specific way. The mean μ_{mean} and variance σ^2_{mean} of the means in dimension d, and the same statistics μ_{var} and σ^2_{var} of the variances, are calculated. Dimension-specific τ^I values are then calculated using the following formula:

$$\tau^I_{\text{mean}}(d) = \tau_{\text{min}} + \alpha \frac{\mu_{\text{var}}(d)}{\sigma^2_{\text{mean}}(d)} \quad (8.1)$$

$$\tau^I_{\text{var}}(d) = \tau_{\text{min}} + 2\alpha \frac{\mu_{\text{var}}(d)^2}{\sigma^2_{\text{mean}}(d)}, \quad (8.2)$$

where τ_{min} and α are free parameters to be set. Means and variances are then updated using Equations (4.30) and (4.31). On experiments with ML adaptive training (not published), it was found that setting τ_{min} to $0.5\tau^I$ and α to $0.4\tau^I$, where τ^I was the previous smoothing constant, works well. $\tau^I = 50$ is the normal smoothing constant for MPE, so most experiments on dimension-specific I-smoothing use $\tau^I_{\text{min}} = 25$ and $\alpha = 20$. Figure 8.4 shows what effect this rule has in terms of the individual τ^I values. The smoothing for the variances is much greater than the previous value of 50, and the smoothing for the mean varies below and above 50. Note that for MPE the prior distributions in Equations (4.30) and (4.31) are based around the ML statistics ("mle"), not the numerator statistics ("num") which are now different from the ML statistics.

Tables 8.29, 8.30 and 8.31 compare default and dimension-specific I-smoothing on NAB, Switchboard and Broadcast News. The NAB results are with a number of different combinations of constants α and τ_{min}, and use an HMM set with 4 Gaussians per state (trained on 1/3 of the training data) and one with 1 Gaussian per state, in order to have results from a reasonably wide range of systems. There is an improvement of 0.25% absolute relative to baseline MPE on the 12 Gaussian/state NAB system, but the results on other systems are not so encouraging, with 0.05% absolute improvement on the 4 Gaussian/state NAB system with reduced (1/3) training data, 0.06% absolute improvement on the 1 Gaussian/state NAB system, 0.1% absolute on the Switchboard system and no improvement on Broadcast News. The improvements from dimension-specific smoothing seem to be consistent, although small, and fit in well with the notion that I-smoothing can be viewed as a MAP technique.

In the context of MAP for ML adaptation, improvements of up to about 0.4% absolute were gained by using dimension-specific values of τ^I, relative to nor-

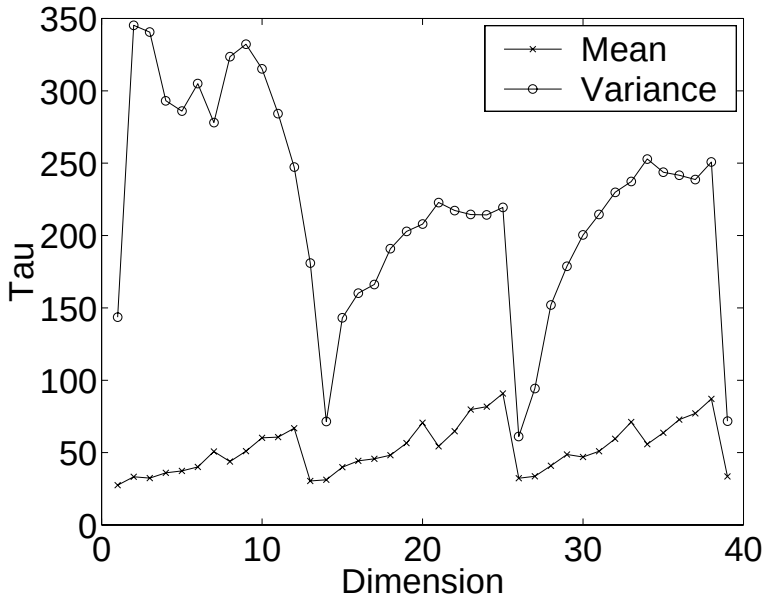


Figure 8.4: Dimension-specific τ^I.

#mix comp	Setup	avg %WER on csrnab1_{dev,eval} Iteration 0	2	4	6	8	MPE objf 7
12	τ^I=50	9.57	9.06	8.96	8.98	9.00	0.958
	τ^I=25	9.57	8.97	8.99	9.05	9.09	0.964
	$\tau^I_{\min}$=25, α=20	9.57	8.99	8.87	8.69	8.75	0.955
	$\tau^I_{\min}$=25, α=10	9.57	8.97	8.96	8.83	8.73	0.958
	$\tau^I_{\min}$=50, α=10	9.57	9.08	8.94	8.88	8.86	0.953
	$\tau^I_{\min}$=50, α=20	9.57	9.09	8.94	8.87	8.85	0.951
4	1/3 data, τ^I=50	10.97	10.33	10.24	10.31	10.43	0.953
	1/3 data, $\tau^I_{\min}$=25, α=20	10.97	10.39	10.34	10.24	10.38	0.949
1	τ^I=50	14.70	13.05	12.47	12.15	11.99	0.934
	$\tau^I_{\min}$=25, α=20	14.70	13.19	12.52	12.11	11.93	0.921

Table 8.29: MPE with constant vs. dimension-specific τ^I, on NAB: E=2, unigram lattices.

	avg %WER on eval98				Value of MPE objf
	Iteration				
	0	4	6	8	7
MPE, τ^I=50	46.6	43.6	43.3	43.1	0.788
MPE, τ^I=25	46.6	43.3	43.1	43.1	0.788
MPE, τ^I_{min}=25, α=20	46.6	43.5	43.2	43.0	0.772

Table 8.30: Constant vs. dimension-specific τ^I, on Switchboard (h5train00sub): E=2, unigram lattices, 12 mix-comp.

	avg %WER on bndev96					Value of MPE objf
	Iteration					
	0	2	4	6	8	7
MPE, τ^I=50	29.6	27.4	26.5	26.4	26.2	0.881
MPE, τ^I_{min}=25, α=20	29.6	27.5	26.6	26.2	26.2	0.870

Table 8.31: Constant vs. dimension-specific τ^I, on Broadcast News: E=2, unigram lattices, 12 mix-comp.

mal MAP adaptation as in [Gauvain & Lee, 1994]. In those experiments (not published), a Switchboard-trained HMM set was adapted to the Voicemail task. Those experiments substituted a previously used value of τ^I (10) with $\tau^I_{\text{min}} = 0.5\tau^I$ and $\alpha = 0.4\tau^I$, so $\tau^I_{\text{min}} = 5$ and $\alpha = 4$. Most improvement was seen for intermediate amounts of adaptation data, which is to be expected since for zero or infinite adaptation data the value of τ^I is irrelevant.

To summarise the effect of varying τ^I per dimension: for MPE training, it gives a small but consistent improvement over a number of different corpora; it can also give a larger improvement for MAP adaptation in an MLE context. However, the technique has not been used for further work since it increases the system's complexity.

8.13 Smoothing constant E varying with iteration

The smoothing constant E which sets the speed of training for both MMI and MPE is generally set to a constant value of E=1 or 2. As described in Section 4.2.4 there is an analogy between the EB update equations and the tech-

	increment per iteration	avg %WER on csrnab1_{dev,eval} Iteration 0	2	4	6	8	Value of MPE objf 7
MPE, τ^I=50, E=2		9.57	9.06	8.96	8.98	9.00	1.05
MPE, τ^I=50 E=2	0.25	9.57	9.09	9.03	9.00	8.93	0.9577
MPE, τ^I=50 E=1	0.25	9.57	8.97	9.03	8.92	8.88	0.9587

Table 8.32: Constant vs. increasing E on NAB: unigram lattices, 12 mix-comp.

	increment per iteration	avg %WER on bndev96 Iteration 0	2	4	6	8	Value of MPE objf 7
MPE, τ^I=50, E=2		29.6	27.4	26.5	26.4	26.2	0.881
MPE, τ^I=50, E=2	0.25	29.6	27.5	26.6	26.3	26.2	0.877
MPE, τ^I=50, E=2	0.5	29.6	27.5	26.7	26.4	26.2	0.874
MPE, τ^I=50, E=1	0.25	29.6	27.0	26.3	26.2	26.1	0.883

Table 8.33: Constant vs. increasing E on Broadcast News: unigram lattices.

nique of Generalised Probabilistic Descent (GPD), which shows that convergence is guaranteed if the smoothing constant E is linearly increased from iteration to iteration. The effect of increasing E in this way was investigated. In all experiments, E was set to a value of 1 or 2 on the first iteration, and increased by a constant amount on each iteration. As shown in Table 8.32 this approach gives a small improvement in recognition results on the NAB corpus, starting from both E=1 and E=2. Starting from E=1 and increasing by 0.25 per iteration (to nearly 3 after 8 iterations), the criterion is optimised faster than the E=2 baseline and the final recognition results were improved. Results from Broadcast news (Table 8.33) testing this same configuration (E=1, increasing by 0.25 per iteration) find an improvement of 0.1% absolute, and the same improvement is found on the eval98 test set for Switchboard (Table 8.13), with an improvement of 0.3% on the smaller eval97sub test set.

In summary, increasing the smoothing constant E with iteration seems to give a small but robust improvement in recognition results. The regime which consists of starting with E=1 and increasing by 0.25 on each iteration (i.e. ending around E=3 after 8 iterations) seems to give good results.

	increment per iteration	%WER on eval98 Iteration 0	4	6	8	final ins/del ratio
MLE		46.6				0.27
MPE ug, τ^I=50, E=2		46.6	43.6	43.3	43.2	0.34
MPE ug, τ^I=50, E=1	0.25	46.6	43.4	43.2	43.1	0.37
		%WER on eval97sub				
MLE		46.0				0.27
MPE ug, τ^I=50, E=2		46.0	42.6	42.4	42.2	0.28
MPE ug, τ^I=50, E=1	0.25	46.0	42.5	42.3	41.9	0.30

Table 8.34: Constant vs. increasing E on Switchboard: h5train00sub (68h) training

# mix comp		avg %WER on csrnab1_{dev,eval} Iteration 0	2	4	6	8	ins/del ratio 8
12	MPE, E=2, τ^I=50 (inexact)	9.57	9.09	9.00	8.99	8.94	1.07
	MPE ug, E=2, τ^I=50, I=1.0	9.57	9.16	9.00	8.94	8.86	1.06
	MPE ug, E=2, τ^I=50, I=0.9	9.57	9.16	8.95	8.95	8.88	1.10
1	MPE ug, E=2, τ^I=50 (inexact)	14.70	13.08	12.48	12.08	11.99	0.81
	MPE ug, E=2, τ^I=50, I=1.0	14.70	12.86	12.26	11.96	11.77	0.76
	MPE ug, E=2, τ^I=50, I=0.9	14.70	12.82	12.27	11.91	11.83	0.76

Table 8.35: Exact vs. inexact implementation of MPE training: NAB

8.14 Exact vs. approximate MPE

Previous experiments have been based on an approximate method of assigning correctness to phones, as described in Section 7.2.1. A more exact implementations, using alignment to "phone sausages," is described in Section 7.3. This section compares the approximate and exact implementations of MPE. As mentioned in Section 7.3, the exact implementation of MPE requires a constant I, which is the error given to an insertion, and which may be set to a value different from 1 in order to prevent too much penalisation of insertions. A larger value of I leads to fewer test-set insertions, so a smaller insertion/deletion ratio.

Table 8.35 compares inexact and exact MPE for systems trained on NAB with 1 and 12 Gaussians per state. The value of I, the error of an insertion, is set both to 1 (the default value) and 0.9, as experiments with a previous implementation of exact MPE found this necessary to correct a changed insertion/deletion ratio.

	Iteration				ins/del
	0	4	6	8	ratio
	%WER on eval98				final
MLE	46.6				0.27
MPE ug, E=2, τ^I=50 (inexact baseline)	46.6	43.6	43.3	43.2	0.34
MPE ug, E=2, τ^I=50, I=1	46.6	43.7	43.4	43.2	0.33
MPE ug, E=2, τ^I=50, I=0.9	46.6	43.8	43.3	43.1	0.35
	%WER on eval97sub				
MLE	46.0				0.24
MPE ug, E=2, τ^I=50 (inexact baseline)	46.0	42.6	42.4	42.2	0.28
MPE ug, E=2, τ^I=50, I=1	46.0	43.2	43.0	42.9	0.27
MPE ug, E=2, τ^I=50, I=0.9	46.0	42.9	42.8	42.8	0.28

Table 8.36: Exact vs. inexact implementation of MPE training: Switchboard, h5train00sub (68h) training data

	avg %WER on bndev96					ins/del ratio
	Iteration					
	0	2	4	6	8	8
MPE ug, E=2, τ^I=50 (inexact)	29.6	27.4	26.5	26.3	26.3	1.09
MPE ug, E=2, τ^I=50, I=1.0	29.6	27.4	26.5	26.4	26.3	1.07
MPE ug, E=2, τ^I=50, I=0.9	29.6	27.5	26.6	26.2	26.3	1.12

Table 8.37: Exact vs. inexact implementation of MPE training: Broadcast News, 12 mix-comp

	avg %WER on csrnab1_{dev,eval}					ins/del ratio
	Iteration					
	0	2	4	6	8	8
1-comp, MWE ug, E=2, τ^I=25 (inexact baseline)	14.70	13.39	12.86	12.77	12.68	0.56
1-comp, MWE ug, E=2, τ^I=25 I=1.0	14.70	13.02	12.46	12.34	12.20	0.67

Table 8.38: Exact vs. inexact implementation of MWE (Minimum Word Error) training: NAB.

There appears to be no clear difference between the different values of I, but exact MPE with I=1 gives a non-significant improvement of 0.08% absolute on the 12 mixture component system and 0.22% on the 1 mixture component system.

Table 8.14 shows similar experiments on the Switchboard corpus. Although exact MPE with I set to 0.9 gives a 0.2% improvement on the larger eval98 test set, there is a 0.6% increase in WER on the eval97sub test set, which is about 1/3 as large as eval98, so overall there is no change. Note that in the transcriptions of the training data used in these experiments a number of common words had been deleted by mistake and this may affect these results.

Table 8.37 compares approximated and exact MPE on Broadcast News. There is no difference between exact and approximate MPE in this case.

In summary, there appears to be no consistent difference between approximate and exact MPE but if exact MPE is used the value I=0.9 may be better than I=1.0 (I is the error due to an insertion).

8.14.1 Exact vs. inexact implementation for Minimum Word Error (MWE) training.

The Minimum Word Error (MWE) criterion is analogous to Minimum Phone Error, except that the phone accuracy $\mathrm{A}(s, s_r)$ which appears in the objective function in Equation (3.7) is caclulated as a word accuracy. The same techniques used to implement both "exact" and "inexact" MPE can trivially be transferred to MWE.

Table 8.38 compares the exact and inexact implementations of MWE on a 1-Gaussian per state NAB system. The exact implementation of MWE gives a 0.48% absolute improvement compared with the inexact implementation; however, it is still worse than exact or inexact MPE (comparable results for MPE are in Table 8.35). This suggests that the exact implementation may be superior for MWE, but the issue has not been pursued since MWE is not the technique of choice, as shown in the next section.

	avg %WER on csrnab1_{dev,eval}					ins/del ratio
	Iteration					
	0	2	4	6	8	8
1-comp, MPE ug, E=2, τ^I=50	14.70	13.05	12.47	12.15	12.00	0.82
1-comp, MWE ug, E=2, τ^I=12.5	14.70	13.07	12.77	12.60	12.81	0.72
1-comp, MWE ug, E=2, τ^I=25	14.70	13.12	12.83	12.68	12.62	0.62
1-comp, MWE ug, E=2, τ^I=50	14.70	13.31	12.84	12.71	12.58	0.58

Table 8.39: Comparison between MPE training, and MWE with varying I-smoothing.

	%WER on eval98					Final ins/del ratio
	Iteration					
	0	2	4	6	8	
MLE	45.6					0.30
MPE ug, E=2, τ^I=50	45.6	42.5	41.3	40.6	40.4	0.21
MWE bg, E=2, τ^I=25 (exact, I=1)	45.6	42.9	41.7	41.3	40.9	0.23

Table 8.40: MPE (inexact) vs. MWE (exact) on Switchboard: h5train00 (265h) training, bigram lattices.

8.15 MWE vs MPE training.

Table 8.39 compares MPE training with MWE training, with the latter at a range of values of τ^I for I-smoothing. This shows that MPE gives better test set results than MWE, and shows that τ^I=25 is a suitable value for training of the MWE criterion. The result of this experiment is that MPE shows a clear advantage over MWE.

	Training Subset WER		Test WER
	Full bg	Lat ug	eval98
MLE baseline	26.3	41.8	46.6
MPE ug, E = 2,τ^I=50 (8 iters)	20.6	27.9	43.1
MWE ug, E=2,τ^I=25 (8 iters)	20.2	25.9	43.3

Table 8.41: MPE vs. MWE on Switchboard: h5train00sub (68h) training, unigram lattices.

Since MWE might be expected to perform well with an excess of training data, an experiment was performed on the full 265h h5train00 training database on Switchboard (Table 8.40). Inexact MPE is compared with exact MWE since pre-

vious experiments made it seem likely that those were the best configurations for MPE and MWE respectively on Switchboard. MWE still gives less improvement than MPE.

Table 8.41 compares MPE and MWE (both with inexact implementations) trained on the smaller h5train00sub training set with unigram lattices, and gives bigram and unigram training set results. MWE performs slightly worse on the test set but better on the training set.

In summary, it seems to be generally true that MPE gives slightly better test-set performance than MWE.

8.16 Full covariance MPE

The Extended Baum-Welch update equations can trivially be extended to the full covariance case, by replacing the scalar individual means and variances with vectors and matrices respectively. The derivation of the update equations using weak-sense auxiliary functions is trivial to extend to the full-covariance case. The only difficulty is in calculating the minimum value of the Gaussian-specific smoothing constant D_{jm} which will give positive variance updates. This value is needed in setting D_{jm} which is set to the maximum of twice this value or E times γ_{jm}^{den}. The normal approach, which reduces to solving a quadratic equation, does not work with full covariances. In the full covariance case it is done by starting at $D_{jm} = \frac{1}{2}E\gamma_{jm}^{\text{den}}$ and successively doubling until the updated variance matrix is positive definite, and then doubling again to get the final value of D_{jm}.

8.16.1 Variance smoothing

For full covariance MPE, it is necessary to use different values of smoothing constant τ^I for the mean and variance as the latter needs more smoothing: two values τ^I_{mean} and τ^I_{var} are used. This requires the use of the form of the EB update equations given in Equations (4.30) to (4.31) which allows separate values of τ^I for means and variances. In addition, following preliminary experiments in MLE training, the Maximum Likelihood estimate of the covariance, which is used as the prior in I-smoothing, has its off-diagonal elements smoothed according to the equation $\Sigma_{jm}{}'_{rc} = \Sigma_{jm_{rc}} \frac{\gamma_{jm}^{\text{mle}}}{\gamma_{jm}^{\text{mle}} + \tau_{\text{offdiag}}}$, with τ_{offdiag} set to, for instance, 200. This is is also performed on the initial iteration of experiments reported below, in which a single iteration of ML training is used to get an initial full-covariance model from a diagonal-Gaussian model.

Table 8.42 gives results from diagonal and full-covariance training on the NAB corpus. Full-covariance systems are much better in absolute terms than their MLE counterparts with the same number of mixture components. However,

	τ^I			avg %WER on csrnab1_{dev,eval}					ins/del ratio
				Iteration					
Diag-cov baselines				0	2	4	6	8	8
1-comp, MPE ug, E=2	50			14.70	13.08	12.48	12.08	11.99	0.81
4-comp, MPE ug, E=2	50			10.86	9.99	9.66	9.58	9.61	0.97
12-comp, MPE ug, E=2	50			9.57	9.09	9.03	9.03	8.96	1.05
	τ^I_{mean}	τ^I_{var}	τ_{offdiag}	Iteration					
Full-covariance				0 (MLE)	1	2	3	4	4
1-comp, MPE ug, E=2	50	200	100	10.10	9.45	9.48	9.44	9.60	1.07
1-comp, MPE ug, E=2	50	500	200	10.11	9.33	9.34	9.30	9.21	1.01
4-comp, MPE ug, E=2	50	500	200	8.86	8.50	8.39	8.27	8.22	1.16
12-comp, MPE ug, E=2	50	500	200	9.07	8.77	8.65	8.38	8.50	1.21

Table 8.42: Full-covariance vs diagonal MPE on NAB, for varying #mix comps

MPE training gives less relative improvement: 18.4%, 11.5% and 6.4% relative for the 1,4 and 12 mixture component diagonal systems respectively, but 8.9%, 7.2% and 6.3% relative for the full-covariance systems. This is expected since overtraining is more of a problem with more complex systems and this can reduce the improvement from discriminative training. The MPE criterion has a much higher value for both ML and MPE-trained HMM sets, for the full-covariance system: for the 12 Gaussian/state HMM set, it rises during training from 0.932 to 0.978 in the diagonal case and from 0.970 to 0.988 in the full-covariance case.

It is interesting to note that the MPE-trained full covariance 4 mixture component system, at 8.22% WER, has the best WER of any system reported here. Similar results cannot be obtained by using more diagonal mixture components. In Table 8.10, the best number of Gaussians per state was found to be 24, with a WER of 8.57% after MPE training.

Table 8.43 gives similar experiments on the Switchboard corpus. Again full covariances improve performance, by about 2.6% absolute for ML-trained systems and 1.7% absolute for MPE-trained systems. The diagonal system given as the baseline here is the largest one used for experiments here; although it has fewer parameters than the full-covariance systems the improvement from increasing the number of Gaussians to the optimal number is expected to be less than 0.5%, i.e. less than the improvement the shift to full variances gives. (This was tested for the Wall Street Journal system in Table 8.10).

Although fairly large improvements have been obtained with full-variance Gaussians, they may not be very useful in practice because they add an order of magnitude to the time taken to compute likelihoods during recognition and the amount of memory needed; and because it is not known whether these improve-

				%WER on eval97sub					
				Iteration					
(Diagonal covariance)	τ^I			0	2	4	6	8	
MLE 16-comp				44.4					
MPE 16-comp, E=2	50			44.4	41.6	40.3	40.0	39.8	
				Iteration					
(Full covariance)	τ^I_{mean}	τ_{var}	τ_{offdiag}	0	1	2	3	4	5
MPE 12-comp, E=2	50	500	200	42.3	40.0	38.5	38.3	38.3	38.1
MPE 4-comp, E=2	50	500	200	43.8	41.7	40.7	39.9	39.2	39.1
				%WER on eval98					
				Iteration					
(Diagonal covariance)	τ^I			0	2	4	6	8	
MLE 16-comp				45.6					
MPE 16-comp, E=2	50			45.6	42.7	41.6	41.0	40.8	
				Iteration					
(Full covariance)	τ^I_{mean}	τ_{var}	τ_{offdiag}	0	1	2	3	4	5
MPE 12-comp, E=2	50	500	200	43.0			39.4		39.1
MPE 4-comp, E=2	50	500	200	44.6			40.3		39.3

Table 8.43: Full-covariance vs diagonal for MPE on Switchboard: h5train00 (265h) training, unigram lattices, varying #mix-comps

	avg %WER on csrnab1_{dev,eval}					ins/del ratio
	Iteration					
	0	2	4	6	8	8
MPE ug, E=2, τ^I=50 (baseline)	9.57	9.09	9.00	9.05	8.96	1.05
MPE ug, E=2, τ^I=50, $\tau^W = 5$,$\tau^T = 5$	9.57	9.12	9.04	9.03	8.96	1.02

Table 8.44: Effect of prior distributions over weight and transition values: NAB, 12 mix-comp.

ments would remain in a system involving MLLR adaptation and feature data projected using a HLDA transformation.

8.17 Miscellaneous details of MPE implementation.

8.17.1 Transition and weight priors

As explained in Section 4.4.1, it is possible to use the Maximum Likelihood estimates of the weight and transition values to form a prior distribution for combination with the discriminative objective function. These priors are expected to be more important for MPE training than MMI training since for MPE training it sometimes happens that certain Gaussians have a zero numerator occupation probability, and this can lead to some weights being set to zero. Experiments are only performed for MPE training. This is not expected to be a significant issue since even not updating the weights and transitions at all makes little difference (Section 6.4.3).

Table 8.44 gives experimental results comparing the use of weight and transition smoothing with a baseline system without smoothing. There is no consistent difference when weight and transition smoothing is added. For further work a small setting $\tau^W = 1$, $\tau^T = 1$ is recommended since this seems likely to give more sensible transition and weight values where no confusable examples of a particular state or transition matrix are available.

8.17.2 Context dependent MPE

Comparison of phones can be done either using or ignoring the phone context. Unless stated otherwise, experiments here ignore the context for purposes of comparing phones in the algorithms (exact and inexact) that differentiate the MPE objective function. (Note however that the context will affect the local

	Training Subset WER		Test WER	
	Full bg	Lat ug	eval98	eval97sub
MLE baseline	26.3	41.8	46.6	46.0
MPE E = 2,τ^I=100 (8 iters)	21.6	29.9	43.3	42.3
MPE-CD,E=2,τ^I=100 (8 iters)	20.7	28.5	43.4	42.5

Table 8.45: MPE: context-dependent (CD) vs. context-independent on Switchboard, h5train00sub (68h) training set.

	Iteration					final	final
	0	2	4	6	8	ins/del	train-lattice
	%WER on eval98					ratio	%WER (ug)
MLE	46.6					0.27	41.8
MPE ug, E=2, τ^I=50	46.6	44.3	43.6	43.3	43.1	0.34	27.9
MPE ug, E=2, τ^I=50 (no silence)	46.6	44.4	43.6	43.1	43.0	0.30	27.7
	%WER on eval97sub						
MPE ug, E=2, τ^I=50	46.0	43.3	42.7	42.4	42.1	0.28	
MPE ug, E=2, τ^I=50 (no silence)	46.0	43.6	42.9	42.6	42.4	0.25	

Table 8.46: MPE training with silence and short pause included in reference (baseline), and excluded: Switchboard, h5train00sub (68h) training sets

differential γ_q^{MPE} for each phone arc q since the errors are propagated forwards and backwards in time). Table 8.45 compares context-independent (baseline) and context-dependent (CD) MPE. Context-dependent MPE seems to give more improvement on the training set but less on the test set, although by a small margin. As with the difference between MPE and MWE, reducing the specificity of the comparison seems to result in better generalization.

8.17.3 Silence in the reference transcription

As mentioned in Section 7.2.1, experiments reported here have, unless otherwise indicated, used the technique named approximate MPE to differentiate the MPE objective function; this is based on an approximate alignment using the time information in the lattices. The silence and short pause models were by default left in the reference (correct transcription) lattices but were given a phone accuracy (PhoneAcc) of zero in the hypothesis (recognition) lattice. This allows some phones to be counted as substitutions that might otherwise be counted as insertions.

Since the exclusion of silence from the reference phones affects the insertion/deletion ratio, some of the experiments described in this section were performed using an error for an insertion that differs from 1 and an altered approximation for phone

	avg %WER on csrnab1_{dev,eval}					ins/del ratio
	Iteration					
	0	2	4	6	8	8
1-comp, MPE ug, E=2, τ^I=50 (baseline)	14.70	13.05	12.47	12.15	12.00	0.82
(no silence, I =0.75)	14.70	12.98	12.40	12.13	12.04	0.87
(no silence, I =0.85)	14.70	12.97	12.33	12.09	12.10	0.82
12-comp, MPE ug, E=2, τ^I=50 (baseline)	9.57	9.06	8.96	8.98	9.00	1.05
(no silence, I =0.75)	9.57	9.27	9.39	9.40	9.33	1.07

Table 8.47: Comparison between inclusion (baseline) and exclusion of silence in reference phones: NAB

correctness. This is done by replacing the two expressions in the approximation of Equation (7.14) with the altered expressions $-I+(1+I)e(q,z)$ and $I(e(q,z)-1)$. This improved recognition results relative to leaving I at 1, and kept the insertion/deletion ratio (last column of Table 8.47) around the normal value.

Table 8.46 shows the effect of excluding silence and short pause models from the correct-transcription lattice when calculating phone accuracy, on a Switchboard system. There is a 0.1% improvement on the eval98 test set and a 0.3% degradation on the smaller eval97sub test set, so overall there is no change. On the other hand, there is a degradation on the NAB corpus (Table 8.47) on systems with either 1 or 12 Gaussians per state.

Note that a more appropriate way to remove silence from the reference transcription might have been to redistribute the time of it between adjacent phones, since this still leaves the approximation used in approximate MPE valid (i.e. the approximated correctness will still be less than or equal to the true value). But this was not tried since it would increase the complexity of the system.

The conclusion is that it is best to leave silence and short pause models in the reference transcription, for approximate MPE. The fact that it is necessary to leave the silence in the reference transcription for approximate MPE is a rather unsatisfactory state of affairs; it would seem more elegant to use exact MPE, which involves no such arbitrary features (except for altering the correctness of an insertion) and does not consistently give either better or worse results than approximate MPE. However, exact MPE is considerably more complex to implement than approximate MPE. Work is ongoing into alternative approximations to the phone error.

8.17.4 MPE with variance flooring

In training of continuous HMMs, variances are typically cut off at some minimum value. In systems built with HTK, a variance floor is typically set to some small fraction of the variance of the data for each dimension (e.g. 1/100). This particular detail is rarely mentioned in speech research publication, but in [Lee, Giachin, Rabiner, Pieraccini & Rosenberg, 1992], it is mentioned that variances are floored to the 20th percentile of the distribution of variances in the HMM set. That method was tried for MPE training, and found to give an improvement relative to the default approach used in HTK. The improvement was around 0.1% absolute for MPE training with the default NAB setup, and 0.2-0.3% absolute for the default MPE training setup with the 68h h5train00sub training set for Switchboard, evaluated on the eval98 test set. The variance floor was applied after each iteration of updating the HMM with EB, using a floor calculated from the HMM values after applying the EB update equations.

8.17.5 MPE and combination of language models

In experiments performed by H.Y. Chan in preparations for the Cambridge submission to the NIST RT03 evaluation for Conversational Telephone Speech (see [Woodland et al., 2003]), it was found that by combining different language models a quite large improvement of 0.5% absolute could be obtained relative to an initial WER of around 30%. These WERs were evaluated without MLLR speaker adaptation or gender-dependent testing, and the baseline is the WER with one of the two original language models (other experiments suggested the two independently were expected to give similar results). The way they were combined is by calculating the statistics $\gamma_{j,m}^{\text{num}}$, $\theta_{j,m}^{\text{num}}(\mathcal{O})$, $\theta_{j,m}^{\text{num}}(\mathcal{O}^2)$ and the den and mle versions of the same quantities, separately using the two language models, and adding them together before each iteration of EB update. This corresponds to maximising an objective function that looks like:

$$\mathcal{F}_{\text{MPE}}(\lambda) = \sum_{r=1}^{R} \frac{\sum_s p_\lambda(\mathcal{O}_r|s)^\kappa P_1(s)^\kappa \text{A}(s,s_r)}{\sum_u p_\lambda(\mathcal{O}_r|u)^\kappa P_1(u)^\kappa} + \frac{\sum_s p_\lambda(\mathcal{O}_r|s)^\kappa P_2(s)^\kappa \text{A}(s,s_r)}{\sum_u p_\lambda(\mathcal{O}_r|u)^\kappa P_2(u)^\kappa}. \tag{8.3}$$

where there are two sets of language model probabilities $P_1(s)$ and $P_2(s)$ as opposed to the normal objective function as given in Equation (7.2) which has only one language model probability function $P(s)$. The two language models used here were a unigram and pruned bigram LM (i.e. a bigram where the bigram entries which least affect the perplexity are pruned away). Note that the two different sets of lattices were created with different HMM sets, one MLE-

trained and one partially MPE trained, so this may be partly responsible for the improvement gained from combining the lattices.

This result needs to be treated with a little caution since if the two sets of lattices were identical but were combined in this way, the effect would not be equivalent to using one set of lattices but would be identical to the effect of reducing the constraint τ^I used in I-smoothing from 50 to 25. However, reducing τ^I to 25 never normally seems to help (Section 8.11) so this is probably not the explanation for the improved results.

8.18 Combination of discriminative training with other techniques

Here follows a brief summary of experiments performed in Cambridge and elsewhere (mostly by people other than myself, acknowledged where necessary) that bear on the interaction of discriminative training with various other techniques.

8.18.1 Discriminative training with MLLR

This section summarises results relating to the combination of discriminative training and MLLR speaker adaptation [Gales & Woodland, 1996], which were obtained during experiments performed for the Cambridge University HTK submission for the NIST evaluation of Hub-5 (Switchboard) systems.

Other experiments mentioned here relate to Speaker Adaptive Training (SAT), which is the training of a HMM set so as to maximise the likelihood of the data when the HMM set is transformed using MLLR for each training speaker.

MPE with MLLR and SAT

In the 2002 NIST evaluation for hub5 (Switchboard) [Woodland et al., 2002], in a system which already included HLDA, MPE training improved WER on the dev01sub test set from 33.3% and 30.1% (9.6% relative); with MLLR the change was from 30.7% to 28.5% (7.2% relative). The difference between ML-SAT (Speaker Adaptive Training) and MPE-SAT was from 29.7% WER to 28.0% (5.7% relative), so both MLLR and SAT reduce the relative improvement from MPE but both improve the absolute performance of the MPE-trained system. That is, MPE give 9.6% relative improvement on its own but 7.2% when baseline and MPE systems were adapted with MLLR, and 5.7% with MLLR and SAT.

MMI with MLLR

In [Woodland & Povey, 2002], MMI training was compared with MLE in systems with and without MLLR. Training was on the h5train00 (265h) training set and testing was on the eval98 test set. Without MLLR, MMI (full-search implementation with E=1) improved results from 44.6 to 42.5 (4.7% relative); with MLLR, the improvement was from 42.1 to 39.9 (5.2% relative). In this case, the improvement was not decreased by MLLR.

Results on this topic were reported in [McDonough et al., 2002] tested with a small HMM set on the English Spontaneous Scheduling Task (ESST). The finding was that MMI works well in combination with MLLR and SAT, but may not work well with MLLR alone.

The overall conclusion from these experiments is that the improvements from discriminative training and MLLR and MLLR+SAT may not always be entirely additive, but discriminative training will generally still give some improvement.

8.18.2 MPE with HLDA

On the evaluation system reported in [Woodland et al., 2002], MPE without HLDA (and without MLLR adaptation) gave an improvement from 35.1% 31.4% (10.5% relative). With HLDA (and without MLLR) the improvement was from 33.3% to 30.1% (9.6% relative). Based on these results there appears to be little interaction between MPE and HLDA, although again the relative improvement from combining the two techniques is less than additive.

8.19 Summary

This chapter has presented experiments on MPE and MWE on a variety of corpora. General conclusions from these experiments include:

- MPE consistently gives more improvement than MMI.
- The improvement from discriminative training rises as the amount of training data per Gaussian increases.

The main conclusions regarding the implementation of MPE are:

- The probability scale κ is best set to around the inverse of the normal language model scale factor (which tends to be in the range 10-15) or perhaps to a smaller value under some conditions.
- The lattices for training of MPE should be made with a sufficiently wide pruning beam; MPE is less sensitive to the lattice size than MMI.

- The smoothing constant E which controls training speed can generally be set to 2, or set to a value starting at 1 and increasing by 0.25 on each iteration.
- The language model in the training lattices should be a unigram or some other very weak language model.
- The value τ^I for I-smoothing is best set to around 50; it can also be set on a dimension-specific basis which gives better results.
- Exact MPE gives about the same improvement as approximate MPE; it can help slightly for exact MPE to give slightly less emphasis to insertion errors (I=0.9).
- The use of variance flooring can improve the results of discriminative training.
- The improvement from other techniques such as MLLR and HLDA is generally less than additive with the improvement from discriminative training.

Chapter 9

Conclusions and further work

This work has investigated discriminative training for HMM-based large vocabulary speedh recognition systems. Its main contributions are summarised in Sections 9.1 to 9.3 below and further work is discussed in Section 9.4

9.1 MMI implementation

Work presented in this thesis investigated a number of issues relating to the implementation of MMI training for large vocabulary speech recognition. The main results of this research are as follows:

- Recognition lattices can be used to speed up MMI training.
- Scaling down probabilities during lattice alignment improves test set results.
- Best results were acheived by using the exact model boundaries from the lattice.
- Other factors important to the implementation are the language model used in the lattice, and the update equations.
- MMI training gives consistent improvements of the order of 5-10% relative on large vocabulary corpora.

The value of this research is, firstly, that this implementation of MMI is the first published example of an MMI training setup that reliably gives improvements across different large vocabulary corpora; and, secondly, that various aspects of the training process which are of practical importance have been investigated.

9.2 Theory for discriminative training

The concept of strong-sense and weak-sense auxiliary functions was introduced, which provides a relatively easy way to derive the Extended Baum-Welch update for means and variances and also leads to a novel approach to the optimisation of Gaussian weights and transitions. Strong-sense auxiliary functions are the kinds of auxiliary function used in E-M training, and provide strict guarantees on convergence. A weak-sense auxiliary function is a function with the same gradient as the objective function, around the local parameter values. These auxiliary functions make it possible to conveniently derive effective update rules when other approaches such as Estimation-Maximisation or gradient descent are not easily applicable.

9.3 MPE

A new discriminative objective function called Minimum Phone Error (MPE) has been introduced. MPE is a smoothed approximation to the error rate of a word recogniser applied to the training set, where this error rate is evaluated on a phone basis for better test-set generalisation. Experiments on various corpora show that MPE reliably gives improvements over MMI, as long as prior distributions over the Gaussian parameters are used to give robust parameter estimates, in a technique named I-smoothing.

Many aspects of the MPE training procedure have been investigated experimentally on a number of corpora, primarily Switchboard, Broadcast News and North American Business news. These experiments compared MPE training with MMI training under a variety of conditions and investigated various aspects of the MPE training procedure such as the method of calculating the differential of the objective function, and other aspects of MPE training such as the language model and the probability scale. More general issues were also investigated, such as how discriminative training is affected by the size of HMM set and the amount of training data. Results on the combination of MPE with other techniques such as MLLR adaptation were also reviewed.

9.4 Further work

One line of further work that is currently being pursued is an investigation into the combination of MAP adaptation with discriminative training, specifically for use in adapting models trained with one task to another task [Povey et al., 2003].

MPE also seems a suitable criterion to use to train some more general parameters of the speech system, such as input transformations. Some initial work on this

topic seems promising, but so far a robust technique that works across various training and test conditions has not been found.

A problem in discriminative training that has not been completely solved is the problem of ensuring robust estimates of discriminatively trained parameters, i.e. ensuring that the limited amount of training data does not introduce too much randomness into the discriminatively estimated parameters. The importance of this is shown by the enormous difference between the training and testing error rates of discriminatively trained systems. Many of the modifications discussed here, such as the use of a unigram language model, scaled down probabilities, and I-smoothing, are intended to solve this problem. However, there may be other ways to do this more effectively.

Appendix A

Experimental setup

Testing was performed using the following corpora and training sets:

- Switchboard using 265 hours of training data, and 65 hour and 18 hour subsets
- North American Business News (NAB; also known as "Wall Street Journal") with 66 hours of training data
- Broadcast News (BN) training on a 72 hour subset of data
- The speaker independent training setup on Resource Management (RM) with 3.8 hours of training data.

This appendix describes the training data and experimental conditions used for experiments described in this thesis. Section A.1 describes the common features of these systems; Section A.2 describes the training and testing data used for each corpus, and Section A.3 explains the details of the creation of lattices for discriminative training.

These systems are the standard systems used in Cambridge at the time this work was done, for the test sets concerned, although some features (e.g. MLLR) were not included in the work done here. See for example [Hain et. al, 2000] for a Switchboard system description, [Woodland et. al, 1999] for Broadcast News and [Woodland at al., 1996] for the NAB corpus.

A.1 Baseline system: common features

The input data for the most of the systems consists of PLP coefficients [Hermansky, 1990] derived from a mel-scale filter bank (MF-PLP) with 13 coefficients including the log energy c_0, and their first and second-order differentials. The Resource Management system used the same configuration except using Mel Frequency cepstral

Training Set	Number of States	Hours of Training	# Gauss/ state	# Gauss/hour training
Switchboard: Minitrain	3088	18	12	2060
Switchboard: Minitrain	3088	18	6	1030
Switchboard: h5train00sub	6168	68	12	1090
Switchboard: h5train00	6168	265	16	370
Broadcast News	6684	72	12	1110
WSJ/NAB (Acoustic Channel 1)	6399	66	12	1160
Resource Management	1582	3.8	6	2500

Table A.1: Sizes of model sets and training databases

coefficients (MFCC). These differences are summarised in Table A.2 along with the different forms of mean and variance normalisation. Note that mean and variance normalisation is performed per conversation side in the case of Switchboard, and on the other systems cepstral normalisation is done on a per-segment basis.

The HMMs used were gender independent cross-word triphones built using decision-tree state clustering [Young et al., 1994]. Each state had the same number of Gaussians, except states in the silence and "short pause" HMMs, which had twice the number of Gaussians. Short pause models are like silence but have a "skip transition" whereby the model can be bypassed; words can be followed by either silence or short pause, with silence but not short pause being counted for purposes of choosing context-dependent models. Conventional MLE training was used to initialise the HMMs prior to discriminative training.

Unless otherwise specified, recognition experiments for Switchboard and NAB used lattice rescoring of word lattices derived using ML-estimated HMMs with a trigram language model. The use of lattices in this way increases the speed of recognition. Recognition for Broadcast News was done using single-pass decoding with a trigram language model. Recognition for Resource Management uses single-pass decoding with word-pair grammar with no language model likelihoods except a word insertion penalty.

The pronunciation dictionaries used in training and testing for all tasks except RM were originally based on the 1993 LIMSI WSJ lexicon, but have been considerably extended and modified; the RM dictionary is made by SRI.

Table A.1 summarises the sizes of the model sets and training databases used in the various systems.

Training Set	Signal Parametrisation	Mean/Variance Normalisation
Switchboard	PLP-MFCC, 13+Δ+ΔΔ	Means & variances
Broadcast News	PLP-MFCC, 13+Δ+ΔΔ	Means
WSJ/NAB (Acoustic Channel 1)	PLP-MFCC, 13+Δ+ΔΔ	Means
Resource Management	MFCC, 13+Δ+ΔΔ	None

Table A.2: Input parametrisation

Training Set	Total Time (hrs)	Number of Conversation Sides	
		SWB1	CHE
Minitrain	18	398	–
h5train00sub	68	862	92
h5train00	265	4482	235

Table A.3: Training sets used on Switchboard

A.2 Data sets

A.2.1 Switchboard system

Three different training sets were used in Switchboard evaluation. These are the Minitrain, h5train00sub and h5train00 sets, consisting of 18, 68 and 265 hours of data respectively (see Table A.3). The Minitrain set, defined by BBN, used BBN-provided transcriptions, while the h5train00 sets used transcriptions based on those provided by Mississippi State University (MSU) in their January 2000 release. All the training sets contain data from the Switchboard I (SWB1) corpus and the h5train00 sets also contain Call Home English (CHE) data. The h5train00sub set is a subset of h5train00 and covers all of the training speakers in the SWB1 portion of h5train00, and a subset of the CHE data.

The sizes of the HMM sets used for the three systems are given in Table A.1. For experiments using the Minitrain subset two different sizes of HMM were used for different experiments, as shown in the table.

Two test sets were used: eval98 and the smaller eval97sub set. These consist respectively of the 1998 Hub5 evaluation data set, containing 40 sides of Switchboard II (SWB2) and 40 CHE sides (in total about 3 hours of data); and a subset of the 1997 evaluation set, containing 10 conversation sides of SWB2 data and 10 of CHE.

A.2.2 NAB (Wall Street Journal) system

The NAB system used HMMs trained on the channel 1 (close-talking microphone) of the SI-284 Wall Street Journal database of read speech (66 hours of data).

The NAB-trained HMMs were tested on the 1994 DARPA Hub1 development and evaluation test sets, denoted csrnab1_dt and csrnab1_et respectively, with a combined length of about 50 minutes, and are tested by lattice rescoring of 65k word vocabulary trigram lattices. This setup is the same as reported for Frame Discrimination (FD) training in [Povey & Woodland, 1999] and for MMI and FD training in [Povey & Woodland, 2001].

A.2.3 Broadcast News system.

The Broadcast News database consists of recorded television shows. The BN system was trained using the 72 hour 1997 training data set (BNtrain97). The test data is the 1996 "partitioned evaluation" development test data (BNdev96pe), which is partitioned into 6 different "focus conditions" (Table A.4). The overall length of the test data is about 2.1 hours.

Focus	Description	% of all data
F0	baseline broadcast speech (clean, planned)	29.7%
F1	spontaneous broadcast speech (clean)	32.7%
F2	low fidelity speech (wideband/narrowband)	8.7%
F3	speech in the presence of background music	7.0%
F4	speech under degraded acoustical conditions	9.1%
F5	non-native speakers (clean, planned)	1.5%
F6	all other speech (e.g. spontaneous non-native)	11.4%

Table A.4: Focus conditions for Broadcast News 1996 Partitioned Evaluation test set.

Testing was by single-pass decoding with a 65k word trigram language model.

A.2.4 Resource Management system.

The Resource Management database consists of clean read speech from a limited vocabulary task consisting of commands to an (imaginary) computer system relating to a shipping database. The RM system was trained on the speaker independent portion of the Resource Management training dataset, 3.84 hours of data. Testing was on the four speaker independent test sets, dated February '89, October '89, February '92 and September '92, 300 sentences in all (about 2500 words).

	General beam	Word-end beam	Lattice-output beam	Lattice depth
	First pass (bigram)			
Switchboard	200	105	150	
NAB	225	115	200	
BN	210	110	175	
	Second pass (unigram)			
Switchboard	225	125	175	125
NAB	225	125	200	16
BN	225	125	200	172

Table A.5: Lattice pruning beams: natural log. Note that language model likelihoods and not acoustic likelihoods are scaled, and the frame rate is 100 frames per second.

A.3 Lattice creation

For all sets of experiments except Resource Management, word lattices for MMI training were created using a bigram language model. Unless stated otherwise, unigram probabilities were actually applied to these lattices for MMI training. Experiments reported in (Section 8.10) suggest that it is possibly better to use unigram rather than bigram probabilities in all stages of lattice generation. Language model probabilities were applied with the same scales and insertion penalties normally used for testing. Word lattices for Resource Management were created using an unconstrained language model (all word-pairs allowed) with no language model likelihoods except a word insertion penalty.

Table A.5 shows the pruning beams used to generate the lattices on the three large corpora, and the average lattice depths (average number of word instances crossing each time) for each corpus. The smaller size for NAB is due to the data being less confusable in that corpus. These figures are thresholds on natural-log likelihood of paths in the lattice, relative to the log likelihood of the best path.

Bibliography

[Amari, 1967] Amari, S. (1967). "A Theory of Adaptive Pattern Classifiers", *IEEE Transactions on Electronic Computers*, vol. EC-16, no. 3, pp. 299-307.

[Bahl et al., 1986] Bahl, L. R., Brown, P. F., de Souza, P. V., & Mercer, R. L. (1986). "Maximum Mutual Information Estimation of Hidden Markov Model Parameters for Speech Recognition", *Proc. ICASSP'86*, pp. 49-52.

[Bahl et al., 1996] Bahl L. R., Padmanabhan M., Nahamoo D., Gopalakrishnan P. S. (1996). "Discriminative Training of Gaussian Mixture Models for Large Vocabulary Speech Recognition Systems," *Proc. ICASSP'96*, vol. 2, pp. 613-616.

[Baker, 1975] Baker, J. K. (1975). "The Dragon System - An Overview", *IEEE Transsations on Acoustics, Speech and Signal Processing*, vol. 23, pp. 24-29.

[Baum et al., 1970] Baum L., Petrie T., Soules G., & Weiss N. (1970). "A Maximization Technique Occuring in the Statistical Analysis of Probablistic Functions of Markov Chains." *The Annals of Mathematical Statistics*, vol. 41, no. 1, pp. 164-171.

[Bauer, 2001] Bauer, J. G. (2001). "On the Choice of Classes in MCE Based Discriminative HMM-Training for Speech Recognizers used in the Telephone Environment", *Proc. Eurospeech'01.*

[Bellegarda & Nahamoo, 1989] Bellegarda J. & Nahamoo D. (1989). "Tied Mixture Continuous Parameter Models for Large Vocabulary Isolated Speech Recognition," *Proc. ICASSP'89* pp. 13-16.

[Bellegarda & Nahamoo, 1990] Bellegarda, J. and Nahamoo, D. (1990). "Tied Mixture Continuous Parameter Modeling for Speech Recognition.", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, pp. 2033-2045.

[Chou et al., 1993] Chou W., Lee C.H., Juang B.H. (1993). "Minimum Error Rate Training Based On N-Best String Models", *Proc. ICASSP'93*, pp. 652-655.

[Chow, 1990] Chow Y.L. (1990). "Maximum Mutual Information Estimation of HMM Parameters for Continuous Speech Recognition Using the N-Best Algorithm", *Proc. ICASSP'90.*

[Davis & Mermelstein, 1980] Davis, S.B.; Mermelstein, P. (1980). "Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences", *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. ASSP-28, pp. 357-366.

[Dempster et al., 1977] Dempster A.P., Laird N.M. & Rubi D.B. (1977), "Maximum Likelihood from Incomplete Data via the EM Algorithm", *Journal of the Royal Statistical Society*, vol. 39, pp. 1-88.

[Gales & Woodland, 1996] Gales M.J.F. & Woodland P.C. (1996). "Mean and Variance Adaptation Within the MLLR Framework", *Computer Speech & Language*, vol. 10, pp. 249-264.

[Gauvain & Lee, 1994] Gauvain, J & Lee, C. (1994). "Maximum A Posteriori Estimation for Multivariate Gaussian Mixture Observations of Markov Chains", *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 2, no. 2, pp. 291-299.

[Gillick, L., and Cox, S.J.] Gillick, L. & Cox, S.J. (1989). "Some Statistical Issues in the Comparison of Speech Recognition Algorithms," *Proc. ICASSP'89*: 532-535.

[Gopalakrishnan et al., 1988] Gopalakrishnan P.S., Kanevsky D., Nadas A., Nahamoo D., Picheny M.A. (1988) "Decoder Selection Based on Cross-Entropies", *Proc. ICASSP'88*, pp. 20-23.

[Gopalakrishnan et al., 1989] P.S. Gopalakrishnan, D. Kanevsky, A. Nadas, and D. Nahamoo (1989). "Generalization of the Baum Algorithm to Rational Objective Functions", *Proc. ICASSP'89*, pp. 631-634.

[Gunawardana, 2001] A. Gunawardana (2001). "Maximum Mutual Information Estimation of Acoustic HMM Emission Densities," *CLSP Research Note* no. 40.

[Haeb-Umbach & Ney, 1992] Haeb-Umbach R. & Ney H. (1992). "Linear Discriminant Analysis for Improved Large Vocabulary Continuous Speech Recognition", *Proc. ICASSP'92, San Francisco*, pp. 13-16.

[Hain et. al, 2000] Hain T., Woodland P.C., Evermann G. & Povey D. (2000). "The CU-HTK March 2000 Hub5e Transcription System", *Proc. Speech Transcription Workshop, College Park, Maryland.*

[Hermansky, 1990] Hermansky, H. (1990). "Perceptual Linear Predictive (PLP) Analysis of Speech", *Journal of the Acoustical Society of America*, vol. 87, pp. 1738-1752.

[Hermansky, Morgan, Bayya & Kohn, 1991] Hermansky, H., Morgan, N., Bayya, A, and Kohn, P (1991). *Rasta-PLP Speech Analysis*. ICSI Technical Report TR-91-069, Berkeley, California.

[Huang & Jack, 1989] Huang X. D., Jack M. A. (1989). "Semi-Continuous Hidden Markov Models for Speech Signals", *Computer Speech and Language*, vol. 3, no. 3, pp. 239-252.

[Jelinek, 1997] Jelinek F. (1997). "Statistical Methods for Speech Recognition." MIT Press.

[Juang, 1985] Juang B.-H. (1985). "Maximum-Likelihood Estimation for Mixture Multivariate Stochastic Observations of Markov Chains," *AT&T Technical Journal*, July-August 1985, vol. 64, no. 6.

[Juang & Katagiri, 1992] Juang, B.-H. & Katagiri, S. (1992). "Discriminative Learning for Minimum Error Classification", *IEEE Transactions on Signal Processing*, vol. 40, no. 12, pp. 3043-3053.

[Juang et al., 1997] Juang B.H., Chou W., and Lee C.H. (1997). "Minimum Classification Error Rate Methods for Speech Recognition", *IEEE Transactions on Speech and Audio Processing*, vol. 5, pp. 266-277.

[Kaiser et al., 2000] Kaiser Z., Horvat B. & Kacic Z., "A Novel Loss Function For the Overall Risk-Criterion Based Discriminative Training of HMM Models," *Proc. ICSLP'2000.*

[Kaiser et al., 2002] Kaiser J, Horvat B. & Kacic Z., "Overall Risk Criterion Estimation of Hidden Markov Model Parameters", *Speech Communication* vol. 38, no. 3-4, pp. 383-398.

[Kapadia, 1998] Kapadia S. (1998). *Discriminative Training of Hidden Markov Models*, Ph.D. Thesis, Cambridge University Engineering Dept.

[Kumar & Andreou, 1998] Kumar N. & Andreou A. G. (1998) "Heteroscedastic Discriminant Analysis and Reduced Rank HMMs for Improved Speech Recognition", *Speech Communcation*, vol. 26, pp. 283-297.

[Lee, Giachin, Rabiner, Pieraccini & Rosenberg, 1992] Lee C-H., Giachin E., Rabiner L.R., Pieraccini R. & Rosenberg A.E. (1992). "Improved Acoustic Modeling for Large Vocabulary Continuous Speech Recognition," *Computer Speech and Language* no. 6, pp. 103-127.

[Lee & Rose, 1996] Lee, L., & Rose, R. (1996). "Speaker Normalization Using Efficient Frequency Warping Procedures", *Proc. ICASSP '96*, Atlanta, GA, pp. 353-356.

[Leggetter & Woodland, 1995] Leggetter, C. J. & Woodland, P. C. (1995). "Maximum Likelihood Linear Regression for Speaker Adaptation of Continuous Density HMMs", *Computer Speech and Language*, vol. 9, pp. 171-186.

[Lowerre, 1976] Lowerre, B. T. (1976). "The Harpy Speech Recognition System", *Ph.D. Thesis, Carnegie Melon University*

[Mangu et al., 2002] Mangu L., Brill E. & Stolcke A. (2000). "Finding Consensus in Speech Recognition: Word Error Minimization and Other Applications of Confusion Networks." *Computer, Speech and Language*, vol. 14, no. 4, pp. 373-400.

[McDonough et al., 2002] J. McDonough, T. Schaaf & A. Waibel (2002). "On Maximum Mutual Information Speaker-Adapted Training", *Proc. ICASSP'02.*

[Merialdo, 1988] Merialdo B. (1988) "Phonetic Recognition using Hidden Markov Models and Maximum Mutual Information Training", *Proc. ICASSP'88*, pp. 111-114.

[Na et al., 1995] K. Na, B. Jeon, D. Chang, S. Chae, S. Ann, "Discriminative Training of Hidden Markov Models Using Overall Risk Criterion and Reduced Gradient Method", *Proc. Eurospeech'95*, pp. 97-100.

[NIST, 2001] *NIST Large Vocabulary Speech Recognition Workshop, May 3-4 2001: National Institute of Standards & Technology, Gaithersburg, Md.* http://www.nist.gov/speech/tests/ctr/h5_2001/postwshp.htm

[Normandin, 1991] Normandin Y. (1991). *Hidden Markov Models, Maximum Mutual Information Estimation and the Speech Recognition Problem.* Ph.D. thesis, Dept. of Elect. Eng., McGill University, Montreal.

[Normandin & Morgera, 1991] Normandin Y., Morgera S.D. (1991). "An Improved MMIE Training Algorithm for Speaker-Independent, Small Vocabulary, Continuous Speech Recognition", *Proc. ICASSP'91*, pp. 537-540.

[Normandin et al., 1994] Normandin Y., Lacouture R., Cardin R. (1994). "MMIE Training for Large Vocabulary Continuous Speech Recognition", *Proc. ICSLP'94*, pp. 1367-1370.

[Pallett, 1990] Pallett D. et al. (1990). "Tools for the Analysis of Benchmark Speech Recognition Tests," *Proc. ICASSP '90*, pp. 97-100.

[Paul, 1990] Paul D. B. (1990). "The Lincoln Tied Mixture HMM Continuous Speech Recogniser", *Proc DARPA Speech and Natural Language Workshop, Hidden Valley, Pennsylvania*, pp. 332-336.

[Povey & Woodland, 1999] Povey D. & Woodland P.C. (1999). "Frame Discrimination Training of HMMs for Large Vocabulary Speech Recognition" *Proc. ICASSP'99*, pp. 333-336.

[Povey & Woodland, 2001] Povey D. & Woodland P.C. (2001). "Improved Discriminative Training Techniques for Large Vocabulary Continuous Speech Recognition", *Proc. ICASSP'01*.

[Povey et al., 2003] Povey D., Gales M.J.F. & Woodland P.C. (2003). "Discriminative MAP for Acoustic Model Adaptation," *Proc. ICASSP'03*.

[Rabiner, Juang, Levinson & Sondhi, 1985] Rabiner, L. R., Juang, B. H., Levinson, S. E., & Sondhi, N. M. (1985). "Recognition of Isolated Digits Using Hidden Markov Models With Continuous Mixture Densities", *AT&T Tech. J.*, vol. 64, pp. 1211-1234.

[Rabiner, 1989] L. R. Rabiner (1989). "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", *Proceedings of the IEEE* (February, 1989), vol. 77 , no. 2, pp. 257-285.

[Rabiner, 1993] Rabiner L. & Juang, B-H (1993). *Fundamentals of Speech Recognition*, Englewood Cliffs NJ: PTR Prentice Hall (Signal Processing Series).

[Reichl & Ruske, 1995] Reichl W., Ruske G. (1995) "Discriminative Training for Continuous Speech Recognition", *Proc. Eurospeech'95*, pp. 537-540.

[Sakoe & Chiba, 1971] Sakoe, H. & Chiba, S. (1971). "A Dynamic Programming Approach to Continuous Speech Recognition", *Proc. 7th International Conference on Acoustics*

[Saon et al., 2000] Saon, G., Padmanabhan, M., Gopinath, R., & Chen., S. (2000) "Maximum Likelihood Discriminant Feature Spaces", *Proc. ICASSP'2000*.

[Schluter et al., 1997] Schluter R., Macherey W., Kanthak S., Ney H., Welling L. (1997). "Comparison of Optimization Methods for Discriminative Training Criteria", *Proc. EUROSPEECH'97*, pp. 15-18.

[Schluter & Macherey, 1998] Schluter R. & Macherey W. (1998). "Comparison of Discriminative Training Criteria", *Proc. ICASSP'98*, pp. 493-496.

[Schluter et al., 1999] Schluter R., Muller B., Wessel F. & Ney H. (1999). "Interdependence of Language Models and Discriminative Training", *Proc. IEEE ASRU Workshop*, Keystone, Colorado, pp. 119-122.

[Schluter, 2000] Schlueter R. (2000). *Investigations on Discriminative Training*, PhD. thesis, Aachen university.

[Schwartz et al., 1985] R. Schwartz, Y. Chow, O. Kimball, S. Roucos, M. Krasner, and J. Makhoul. (1985). "Context-dependent Modeling for Acoustic-Phonetic Recognition of Continuous Speech",, *Proc. ICASSP'85*, pp. 1205-1208.

[Strik et al., 2000] Strik, H., Cucchiarini, C. and Kessens, J. (2000). "Comparing the Recognition Performance of CSRs: In Search of an Adequate Metric and Statistical Significance Test", Proceedings of ICSLP' 00, , Beijing, 740-744, 2000.

[Valtchev et al., 1996] Valtchev, V., Woodland, P.C., Young, S.J. (1996). "Lattice-based Discriminative Training for Large Vocabulary Speech Recognition", *Proc. ICASSP'96*, pp. 605-608.

[Valtchev et al., 1997] Valtchev V., Odell J.J., Woodland P.C. & Young S.J. (1997). "MMIE Training of Large Vocabulary Speech Recognition Systems", *Speech Communication*, vol. 22, pp. 303-314.

[Vintsyuk, 1968] Vintsyuk, T.K. (1968). "Speech Recognition by Dynamic Programming", *Kibernetika (Cybernetics)*, vol. 4, pp. 81-88.

[Welling et al., 1999] Welling L., Kanthak S., & Ney H. (1999). "Improved methods for vocal tract normalization", *Proc. ICASSP'99*, 761-764.

[Wolpert, 1994] Wolpert, D. H. (1994), "The Relationship Between PAC, the Statistical Physics Framework, the Bayesian Framework, and the VC Framework", in D. H. Wolpert, ed., *The Mathematics of Generalization*, Addison Wesley.

[Woodland et al., 1995] Woodland P.C., Leggetter C.J., Odell J.J., Valtchev V. & Young S.J. (1995). "The 1994 HTK Large Vocabulary Speech Recognition System", *Proc. ICASSP'95*, pp. 73-76.

[Woodland at al., 1996] Woodland P.C., Gales M.J.F., Pye D. Valtchev V. (1996). "The HTK Large Vocabulary Recognition System for the 1995 ARPA h3 Task," *Proc. DARPA Speech Recognition Workshop '96*, pp. 99-104.

[Woodland et. al, 1999] Woodland P.C, Hain T, Moore G.L., Niesler T.R., Povey D., Tuerk A. & Whittaker E.W.D. (1999). "The 1998 HTK Broadcast News

Transcription System: Development and Results", *Proc 1999 DARPA Broadcast News Transcription and Understanding Workshop, Herndon, VA.*

[Woodland & Povey, 2000] P.C. Woodland & D. Povey (2000). "Large Scale Discriminative Training for Speech Recognition", *Proc. ISCA ITRW ASR2000*, Paris, pp. 7-16.

[Woodland & Povey, 2002] P.C. Woodland & D. Povey (2002). "Large Scale Discriminative Training of Hidden Markov Models for Speech Recognition", *Computer Speech & Language*, Jan 2002, vol. 16, no 1, pp. 25-48,

[Woodland et al., 2002] P.C. Woodland, G. Evermann, M. Gales, T. Hain, A. Liu, G. Moore, D. Povey & L. Wang (2002). "CU-HTK April 2002 Switchboard System," *Proc. NIST Rich Transcription Workshop*, 2002.

[Woodland et al., 2003] P.C. Woodland et. al (2003) "CU-HTK STT Systems for RT03," *Proc. NIST Rich Transcription Workshop*, 2003.

[Young et al., 1994] Young, S.J., Odell, J.J. & Woodland, P.C. (1994) Tree-based State Tying for High Accuracy Acoustic Modelling. *Proc. ARPA Human Language Technology Workshop,* Plainsboro, NJ., pp. 307-312.

[Zheng et al., 2001] Zheng J., Butzberger J., Franco H., & Stolcke A. (2001). "Improved Maximum Mutual Information Estimation Training of Continuous Density HMMs", *Proc. Eurospeech'01*, pp. 679-81.