Frame Discrimination Training of HMMs for Large Vocabulary Speech Recognition

D. Povey & P.C. Woodland

Cambridge University Engineering Department
Trumpington Street, Cambridge CB21PZ, UK
email: { dp10006, pcw } @eng.cam.ac.uk

Abstract

This report describes the implementation of a discriminative HMM parameter estimation technique known as Frame Discrimination (FD) for large vocabulary speech recognition, and reports improvements in accuracy over ML-trained and MMI-trained models. Features of the implementation include the use of an algorithm called the Roadmap algorithm which selects the most important Gaussians for a given input frame without calculating every Gaussian probability in the system, a new distance measure between Gaussian based on overlap (which is used in the Roadmap algorithm), and an investigation of improvements to the Extended Baum-Welch formulae. Frame Discrimination estimation is found to give error rates at least as good as MMI with considerably less computational effort.

1 Introduction

Discriminative HMM parameter re-estimation techniques, for example Maximum Mutual Information (MMI), have been widely reported in the literature to improve recognition results; but there have been relatively few reports of the application of these techniques to large vocabulary speech recognition. See, for example, [5, 8, 9, 10]. A good part of the reason for this is the extra computational effort involved in MMI training. In [8, 5], the use of recognition lattices as an approximation to MMI training was reported, which resulted in a considerable speedup relative to a more naive implementation, but it still took 15 times longer than conventional Maximum Likelihood (ML) training.

A discriminative criterion called Frame Discrimination (FD) was developed in [2]. Its efficient implementation for large vocabulary speeech recognition (LVCSR) is reported here. To implement FD efficiently the Roadmap algorithm was developed which finds the Gaussians in the HMM set which best match an input vector (i.e, highest probability), while only testing a fraction of the Gaussians in the HMM set (in the region of 1-10%). This is done by setting up links, or “roads” between Gaussians and navigating among them using a hill-climbing algorithm. The links are set up using a new distance measure, based on overlap of Gaussians. In re-estimating the HMMs the Extended Baum-Welch (EBW) formulae are used, and improvements to these formulae are proposed and tested here.

The rest of the report is structured as follows: Section 2 introduces the FD objective function; Section 3 details the optimisation approach used; Section 4 describes the Roadmap algorithm; and Section 5 describes an experimental evaluation of FD on the speech recognition tests.

2 The FD objective function

The FD objective function is related to the MMI objective function, which was first proposed in [12]. The MMI objective function is the posterior probability of the speech transcription, given the speech data:

$$\mathcal{F}_\lambda = \sum_{r=1}^{R} \log \frac{P_\lambda\left(\mathcal{O}_r|\mathcal{M}^{w_r}\right)P(w_r)}{\sum_{\hat{w}} P_\lambda\left(\mathcal{O}_r|\mathcal{M}^{\hat{w}}\right)P(\hat{w})}, \tag{1}$$

where w_r is the word sequence corresponding to training utterance r and $\mathcal{M}^w$ is the composite HMM corresponding to the word sequence w. $P(w)$ is the probability of the word sequence w, as given by the language model. The MMI objective function may be rewritten in terms of the transcription model M^{w_r} and the general model of speech production M^{gen}, (which may be the same as the model used in the speech recogniser), as follows:

$$\mathcal{F}_\lambda = \sum_{r=1}^{R} \log P_\lambda(\mathcal{O}_r|\mathcal{M}^{w_r}) - \log P_\lambda(\mathcal{O}_r|\mathcal{M}^{\mathrm{gen}}) \tag{2}$$

The model $\mathcal{M}^{w_r}$ is known as the numerator model, and $\mathcal{M}^{\mathrm{gen}}$ as the denominator model, because the subtraction of logs may be considered a division.

The FD objective function is an altered form of Equation 2 where the model $\mathcal{M}^{\mathrm{gen}}$ has been replaced by a model $\mathcal{N}$, which allows a superset of the state sequences allowed in $\mathcal{M}^{\mathrm{gen}}$. The hope is that, by allowing these extra state sequences, the alignment of a given speech frame to the states of the model $\mathcal{M}^{\mathrm{gen}}$ will be less dependent on the context of the speech frame, and more typical of the assignment of states to that frame in the language at large.

$$\mathcal{F}_\lambda = \sum_{r=1}^{R} \log P_\lambda(\mathcal{O}_r|\mathcal{M}^{w_r}) - \log P_\lambda(\mathcal{O}_r|\mathcal{N}) \tag{3}$$

In this report, and in [2], the particular form of frame discrimination used is zero memory frame discrimination. $\mathcal{N}$ is a zero memory Markov chain, whose output PDF consists of a weighted sum of all the PDFs in the HMM set so that

$$P_\lambda(\mathcal{O}_r|\mathcal{N}) = \prod_{t=1}^{T(r)} \sum_{i\in\mathcal{M}^{\mathrm{gen}}} b_i(x^r(t))P(q_i|\mathcal{N}),$$

where $x^r(t)$ are the vectors of speech data, $T(r)$ is the length of utterance r, and $b_i(x^r(t))$ is the output PDF of state i. The notation $\sum_{i\in\mathcal{M}^{\mathrm{gen}}}$ indicates summation over all the states in $\mathcal{M}^{\mathrm{gen}}$, i.e, all states in the HMM set. $P(q_i|\mathcal{N})$ is the prior probability of observing state q_i. The prior probability of each state is set proportionally to its occupation count as calculated by the forward-backward algorithm for a previous iteration of ML training.

3 Extended Baum-Welch (EBW) re-estimation

3.1 The EBW formulae

To optimise the parameters of HMMs when using discriminative criteria such as MMI or FD, the EBW re-estimation formulae can be used. The EBW algorithm for rational objective functions was introduced in [1] and extended in [4] for the continuous density HMMs considered here. The re-estimation formulae presented below have been found to work well in practice although they can be only proved to converge when a very large value of the constant D is used which in turn leads to very small changes in the model parameters on each iteration.

In the following, counts and other functions of the alignment will be given a superscript *num* or *den*, to indicate whether they pertain to the numerator models $\mathcal{M}^{w_r}$ or the denominator model $\mathcal{N}$.

The update equations for the mean vector $\mu_{j,m}$ of the m'th mixture component of state j, and the corresponding variance vector, $\sigma^2_{j,m}$, are as follows:

$$\hat{\mu}_{j,m} = \frac{\left\{\theta^{\mathrm{num}}_{j,m}(\mathcal{O}) - \theta^{\mathrm{den}}_{j,m}(\mathcal{O})\right\} + D\mu_{j,m}}{\left\{\gamma^{\mathrm{num}}_{j,m} - \gamma^{\mathrm{den}}_{j,m}\right\} + D}, \tag{4}$$

$$\hat{\sigma}^2_{j,m} = \frac{\left\{\theta^{\mathrm{num}}_{j,m}(\mathcal{O}^2) - \theta^{\mathrm{den}}_{j,m}(\mathcal{O}^2)\right\} + D(\sigma^2_{j,m} + \mu^2_{j,m})}{\left\{\gamma^{\mathrm{num}}_{j,m} - \gamma^{\mathrm{den}}_{j,m}\right\} + D} - \hat{\mu}^2_{j,m}, \tag{5}$$

where $\theta_{j,m}(\mathcal{O})$ represents the sum of the vectors of training data weighted by the probability of occupying that mixture component, i.e.:

$$\theta_{j,m}(\mathcal{O}) = \sum_{r=1}^{R}\sum_{t=1}^{T_r} \gamma^r_{j,m}(t)x^r(t),$$

and $\theta_{j,m}(\mathcal{O}^2)$ is a similar sum of squared input values. $\gamma^r_{j,m}(t)$ is the probability of occupying mixture m of state j at time t, and $\gamma_{j,m}$ is the count of the number of times mixture component m of state j is occupied, i.e.:

$$\gamma_{j,m} = \sum_{r=1}^{R} \sum_{t=1}^{T_r} \gamma^r_{j,m}(t).$$

3.2 Mixture Weight Updates

The formula used for continuous EBW updates is similar to the update for discrete output probabilities originally put forward in [1]. It is as follows:

$$\hat{c}_{j,m} = \frac{c_{j,m}\left\{\dfrac{\partial \mathcal{F}_\lambda}{\partial c_{j,m}} + C\right\}}{\sum_{\hat{m}} c_{j\hat{m}}\left\{\dfrac{\partial \mathcal{F}_\lambda}{\partial c_{j\hat{m}}} + C\right\}}, \tag{6}$$

where the derivatives $\frac{\partial \mathcal{F}_\lambda}{\partial c_{j,m}}$ are given by:

$$\frac{\partial \mathcal{F}_\lambda}{\partial c_{j,m}} = \frac{1}{c_{j,m}}(\lambda^{\text{num}}_{j,m} - \lambda^{\text{den}}_{j,m}). \tag{7}$$

However, these are not the values commonly used in Equation 6 when performing EBW re-estimation. Merialdo [13] found while performing gradient optimisation for disriminative training of discrete HMM systems that the gradients were excessively dominated by low valued parameters, due to the division by $c_{j,m}$ in Equation 7. He therefore improved convergence by using the alternative formula as follows:

$$\frac{\partial \mathcal{F}}{\partial c_{j,m}} \simeq \frac{\lambda^{\text{num}}_{j,m}}{\sum_{\hat{m}} \lambda^{\text{num}}_{j,\hat{m}}} - \frac{\lambda^{\text{den}}_{j,m}}{\sum_{\hat{m}} \lambda^{\text{den}}_{j,\hat{m}}} \tag{8}$$

This equation differs considerably from Equation 7. The most we can say is that the sign of the derivatives calculated both ways is likely to be the same, assuming the total numerator and denominator occupancies for the state are roughly equal. In experiments reported in [3], this approximation dramatically improved the rate of convergence for discrete HMMs. A look at Equations 6 and 8 shows why the equations are effective. The value of the approximation to the derivative as calculated in Equation 8 is normalised to lie between -1 and 1; this means that the same value of C will be appropriate for all mixtures.

A problem encountered in practice with these altered update equations is that, during training, the objective function starts to decrease again after increasing to near its maximum [3]. This is not surprising, since even the sign of the derivative in the approximation of Equation 8 may differ from the actual value: i.e, although Equation 8 may give good results, it is not a good approximation to the derivative.

3.3 New Mixture Weight Updates

A new set of mixture weight update equations were developed. These equations are based on heuristics about how the occupancies are expected to change as the mixture weights are changed. In the following explanation, mixture weights $c_{j,m}$ will be denoted c_m, the state j being assumed constant.

It is clear from the way HMMs work that increasing the value of a mixture weight will tend to increase its occupancies, and decreasing it will tend to decrease the occupancy. If it was known in advance what the effect of changing the mixture weights c_m would be on the occupancies γ^{num}_m and γ^{den}_m, then gradient descent could be performed efficiently based on this knowledge, without actually calculating the new occupancies in the normal way, e.g., by the Forward-Backward algorithm. Of course, this is not possible because it cannot be known exactly what the new occupancies will be. But it is possible to make non-infinitesimal updates by estimating limits on the change of the occupancies as mixture weights change. The limits that were estimated were:

The occupancy γ_m of a mixture with initial weight $\bar{c}_m$, initial occupancy $\bar{\gamma}_m$ and final weight c_m is bounded by $\bar{\gamma}_m$ and $\frac{c_m}{\bar{c}_m}\bar{\gamma}_m$. This is true for both numerator and denominator occupancies (γ_m^{num} and γ_m^{den}).

From these limits on the occupancies, an update rule is derived as follows. We will consider only the variation in the mixture weights of one state, so that the parameter set $\boldsymbol{\lambda}$ becomes a vector of mixture weights $(c_1 \dots c_M)$. Consider the function $\mathcal{G}(\hat{\boldsymbol{\lambda}}, \bar{\boldsymbol{\lambda}}) = \log \mathcal{F}(\hat{\boldsymbol{\lambda}}) - \log \mathcal{F}(\bar{\boldsymbol{\lambda}})$, where $\bar{\boldsymbol{\lambda}}$ is the initial set of parameters and $\hat{\boldsymbol{\lambda}}$ is the updated set. It is clear that if we ensure that $\mathcal{G}(\hat{\boldsymbol{\lambda}}, \bar{\boldsymbol{\lambda}}) > 0$ we guarantee an increase in the objective function: $F(\hat{\boldsymbol{\lambda}}) > F(\bar{\boldsymbol{\lambda}})$. The value of $\mathcal{G}(\hat{\boldsymbol{\lambda}}, \bar{\boldsymbol{\lambda}})$ may be expressed as the line integral:

$$\begin{aligned} \mathcal{G}(\hat{\boldsymbol{\lambda}}, \bar{\boldsymbol{\lambda}}) &= \oint_{\bar{\boldsymbol{\lambda}} \to \hat{\boldsymbol{\lambda}}} \nabla_{\boldsymbol{\lambda}} \mathcal{F}(\boldsymbol{\lambda}).d\boldsymbol{\lambda} \\ &= \oint_{\bar{\boldsymbol{\lambda}} \to \hat{\boldsymbol{\lambda}}} \textstyle\sum_{m=1}^{M} \frac{\partial F}{\partial (log c_m)} d(\log c_m). \end{aligned}$$

We can choose to integrate along any path between $\hat{\boldsymbol{\lambda}}$ and $\bar{\boldsymbol{\lambda}}$ along which $\mathcal{F}(\boldsymbol{\lambda})$ is defined: i.e, any path that preserves the sum-to-one constraint on the mixture weights. For convenience, we will choose the path corresponding to the straight line between $(\bar{c}_1 \dots \bar{c}_M)$ and $(\hat{c}_1 \dots \hat{c}_M)$, which can be mapped on to the space in which $\boldsymbol{\lambda}$ is defined by taking logs. The values $\frac{\partial \log \mathcal{F}}{\partial \log c_m}$ are given by $\gamma_m^{\text{num}} - \gamma_m^{\text{den}}$, so:

$$\mathcal{G}(\hat{\boldsymbol{\lambda}}, \bar{\boldsymbol{\lambda}}) = \oint_{\bar{\boldsymbol{\lambda}} \to \hat{\boldsymbol{\lambda}}} \sum_{m=1}^{M} (\gamma_m^{\text{num}} - \gamma_m^{\text{den}})\, d(\log c_m).$$

We only have bounds on these values as the weights c_m change; however, if we set the numerator occupancies at the bound $\gamma_m^{\text{num}} = \bar{\gamma}_m^{\text{num}}$ and the denominator occupancies at the bound $\gamma_m^{\text{den}} = \frac{c_m}{\bar{c}_m}\bar{\gamma}_m^{\text{den}}$, giving the function

$$\mathcal{H}(\hat{\boldsymbol{\lambda}}, \bar{\boldsymbol{\lambda}}) = \oint_{\bar{\boldsymbol{\lambda}} \to \hat{\boldsymbol{\lambda}}} \sum_{m=1}^{M} \left(\bar{\gamma}_m^{\text{num}} - \frac{c_m}{\bar{c}_m}\bar{\gamma}_m^{\text{den}} \right) d(\log c_m), \tag{9}$$

then $\mathcal{H}(\hat{\boldsymbol{\lambda}}, \bar{\boldsymbol{\lambda}}) \leq \mathcal{G}(\hat{\boldsymbol{\lambda}}, \bar{\boldsymbol{\lambda}})$. This is proved by a case split between c_m that are increasing and those that are decreasing. The path along which we are integrating corresponds to a straight line between $(\bar{c}_1 \dots \bar{c}_M)$ and $(\hat{c}_1 \dots \hat{c}_M)$, so each value $\log c_m$ is either increasing or decreasing, and does not alternate between the two. In order to prove that $\mathcal{H}(\hat{\boldsymbol{\lambda}}, \bar{\boldsymbol{\lambda}}) \leq \mathcal{G}(\hat{\boldsymbol{\lambda}}, \bar{\boldsymbol{\lambda}})$, it is sufficient to prove that for all m and for all valid sets of mixture weights $(c_1 \dots c_M)$, as $d(\log c_m)$ approaches zero,

$$\left(\bar{\gamma}_m^{\text{num}} - \frac{c_m}{\bar{c}_m}\bar{\gamma}_m^{\text{den}} \right) d(\log c_m) \leq \left(\gamma_m^{\text{num}} - \gamma_m^{\text{den}} \right) d(\log c_m). \tag{10}$$

For those c_m that are increasing, the inequality of Equation 10 can be proved from the facts that $d(\log c_m) > 0$ and $c_m \geq \bar{c}_m$, and our estimated bounds on the occupancies. Similar reasoning holds for those c_m that are decreasing.

We can obtain a closed form for $\mathcal{H}(\hat{\boldsymbol{\lambda}}, \bar{\boldsymbol{\lambda}})$ by integrating. Since each element of the summation in Equation 9 only depends on one mixture weight c_m, $\mathcal{H}(\hat{\boldsymbol{\lambda}}, \bar{\boldsymbol{\lambda}})$ can be written as:

$$\sum_{m=1}^{M} \int_{\log c_m = \log \bar{c}_m}^{\log \hat{c}_m} \left(\bar{\gamma}_m^{\text{num}} - \frac{c_m}{\bar{c}_m}\bar{\gamma}_d^{\text{den}} \right) d(\log c_m).$$

Integrating over each of the $(\log c_m)$, and noting that $c_m = \exp(\log c_m)$, which is unchanged by integration w.r.t. $\log c_m$, we have:

$$\mathcal{H}(\hat{\boldsymbol{\lambda}}, \bar{\boldsymbol{\lambda}}) = K + \sum_{m=1}^{M} \bar{\gamma}_m^{\text{num}} \log \hat{c}_m - \bar{\gamma}_m^{\text{den}} \frac{\hat{c}_m}{\bar{c}_m}. \tag{11}$$

Since $\mathcal{H}(\bar{\boldsymbol{\lambda}}, \bar{\boldsymbol{\lambda}}) = 0$, choosing $(\hat{c}_1 \dots \hat{c}_M)$ in order to maximise $\mathcal{H}(\hat{\boldsymbol{\lambda}}, \bar{\boldsymbol{\lambda}})$ can only result in $\mathcal{H}(\hat{\boldsymbol{\lambda}}) \geq 0$, which in turn guarantees an increase (or no change) in the objective function $\mathcal{F}(\boldsymbol{\lambda})$. This maximisation was done numerically using general purpose optimisation routines since there seems to be no analytical solution for an arbitrary number of mixtures. $(\hat{c}_1 \ldots \hat{c}_M)$ are the new mixture weights. Note that in the special case where $\hat{\gamma}_m^{\mathrm{den}} = 0$ for all m, i.e, when we have no denominator occupancies, there is an analytical solution and it coincides with the BW updates for Maximum Likelihood training.

These equations show a slight departure in approach from previous attempts to derive update rules. Derivations of update rules have tended to start from assumptions which are valid, leading to update equations that make too little change and which therefore have to be altered to provide reasonable performance (e.g, by changing values of smoothing constants, or tweaking the equations). The approach taken here is to start from assumptions which are not guaranteed to be valid, but which seem likely to result in reasonable updates, and to derive the update equations directly from these assumptions. The techniques used in proving the validity of the update formulae are derived from those used by previous authors [1, 4].

Experimental results failed to show a significant difference between these update equations and the standard ones. However, these equations were used in experiments reported here because there seemed to be a slight improvement. Mixture weights make little difference anyway in mixture-of-Gaussian speech recognition systems, so it is not surprising that little effect was seen. Work is under way to extend this approach to Gaussian updates. It would also be interesting to try these update equations on a discrete or semi-tied HMM system, where the choice of mixture weight update is more critical.

3.4 Setting the constant D

D is the smoothing constant in Equations 4 and 5 for updating the Gaussian parameters. In [4], where the EBW updates for continuous HMMs were introduced, and in subsequent work with continuous HMMs, D was set to twice the minimum positive value needed to ensure that all variances were positive. Alterations are made to this approach for the current work: these are described below. These alterations are reported in detail because they were essential in getting the system to register an improvement from FD training.

The value of the constant D is important: too low a figure results in slow convergence, and too high a figure will result in instability. Inspection of Equations 4 and 5 shows that D must have about the same magnitude as the occupancies (or counts) $\gamma_{j,m}$. Thus, a value of D which is high enough to smooth a frequently used phone model may be too large for a less frequently used model. Accordingly, for work with large HMM systems D has been set at a phone level (e.g., [5, 8]).

A suitable value of D is normally found by calculating the minimum positive value which ensures that all variance updates are positive, and doubling it. Doubling D is supposed to provide a margin which ensures that the value chosen is considerably larger than any value which gives negative updates. However, it was observed that if the minimum value which ensures positive updates is close to zero (i.e, considerably smaller than the occupancy counts $\gamma_{j,m}^{\mathrm{num}}$ and $\gamma_{j,m}^{\mathrm{den}}$), then doubling it will have little effect. Extreme values of mean and variance could still result. An attempt was made to correct this by introducing a floor on D.

3.5 Flooring D

In experiments reported here, D was set on a phone-by-phone basis as in [5], subject to a floor at the maximum of $\gamma_{j,m}^{\mathrm{num}}$ or $\gamma_{j,m}^{\mathrm{den}}$ for any mixture component in the phone. The use of a floor was found to improve both convergence of the FD criterion and recognition performance. Figure 1 shows the effect on FD criterion optimisation and recognition performance of three different floors on D: zero, the maximum value of any of $\gamma_{j,m}^{\mathrm{num}}$ or $\gamma_{j,m}^{\mathrm{den}}$ for any mixture component in the phone, and the maximum value of $\gamma_{j,m}^{\mathrm{den}}$ in the phone. This was for FD training of a single Gaussian system on the RM corpus; the details of the experiment are the same as for similar experiments described later.

Figure 2 shows the effect of setting D per mixture component as compared to per phone, both subject to a floor. This is for the 6-mixture RM system, as described below. In the mixture-level case, D was floored at the greater of the numerator or denominator counts $\gamma_{j,m}^{\mathrm{num}}$ and $\gamma_{j,m}^{\mathrm{den}}$; in the phone-level case, it was floored at the largest of either of these counts for any mixture in the phone. Although the results in Figure 2 seem to recommend the use of a mixture-level D, for other experiments reported here D is set on a per-phone basis since at the time they were carried out the possibility of setting it on a mixture level had not been investigated.

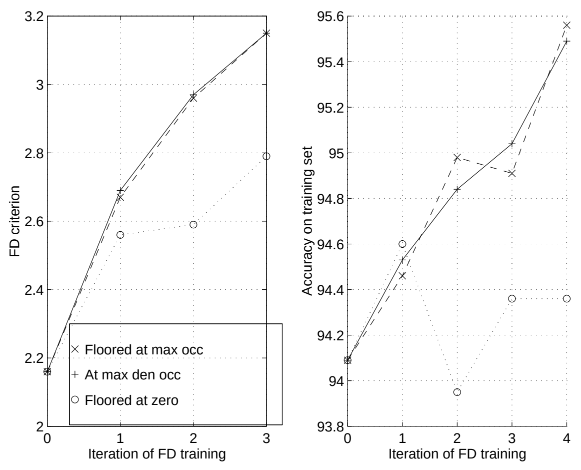

Figure 1: Comparison of various floorings on D, when set on a phone level

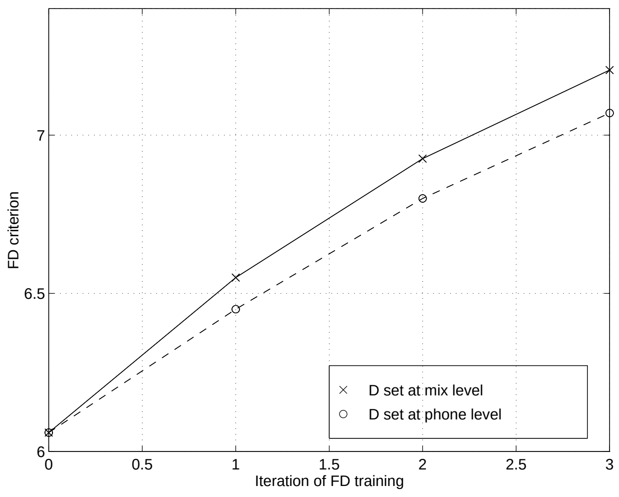

Figure 2: Setting D on a mixture vs. phone level.

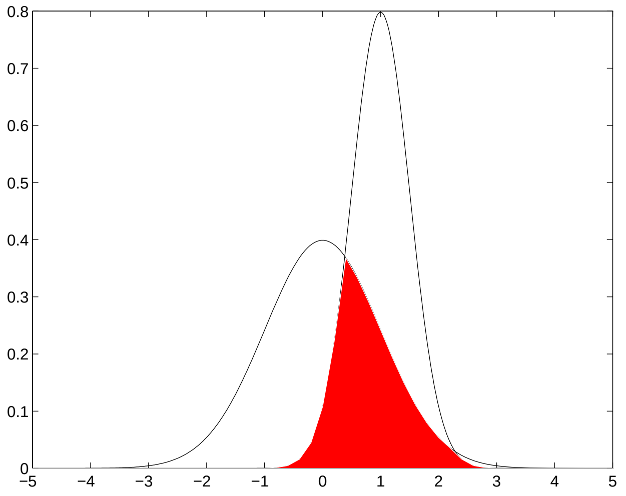

Figure 3: Overlap of univariate Gaussians

3.6 Implementation Considerations

In re-estimating the parameters it is necessary to calculate the denominator occupancies $\gamma_{j,m}^{r,\mathrm{den}}(t)$ for each time frame and each mixture component in the HMM set:

$$\gamma_{j,m}^{r,\mathrm{den}}(t) = \frac{c_{j,m} b_{j,m}(x^r(t)) P(q_i|\mathcal{N})}{\sum_{j \in \mathcal{M}_{\mathrm{gen}}} \sum_{m=1}^{M_j} c_{j,m} b_{j,m}(x^r(t)) P(q_j|\mathcal{N})} \tag{12}$$

where $b_{j,m}(\cdot)$ is the Gaussian associated with mixture m of state j, $c_{j,m}$ is the mixture weight for the Gaussian, M_j is the number of Gaussians in the mixture for state j and $P(q_j|\mathcal{N})$ is the prior probability for state j. It follows from Equation 12 that $b_{j,m}(x^r(t))$ must be calculated for each Gaussian in the system and for every time frame, and thus the overall computation is dominated by calculation of the denominator occupancies. In the case of the numerator occupancies, beam pruning applied to the forward-backward algorithm may be used to optimise their computation, and in any case the numerator model (transcription model) for a given utterance is unlikely to contain all states in the HMM set. To make FD practical for large HMM systems (12) should be computed for just the most likely Gaussians in the system (which together contribute nearly all the log likelihood per frame) and the denominator of (12) computed over just those Gaussians. Therefore, the Roadmap algorithm was developed with the aim of finding the most likely Gaussians in the system for each speech frame.

4 The Roadmap algorithm

The purpose of this algorithm is to reliably find those Gaussians in the system which best match the input for each time frame, while minimising computation. It operates by setting up for each Gaussian a list of the most similar Gaussians in the system, forming a “roadmap”– hence the name. Search is local, centering around those Gaussians that have already been found to score best.

4.1 Distance Measure

A widely used measure of the distance between two Gaussians is the divergence. However for current purposes it was found that the divergence gives too high a value for the difference between two Gaussians when they have very different variances. Therefore an alternative distance measure was sought and one based on Gaussian “overlap” developed.

The overlap between two univariate Gaussians is shown in Figure 3, being defined as:

$$O(g^{(1)}(\cdot), g^{(2)}(\cdot)) = \int_{x=-\infty}^{+\infty} min(g^{(1)}(x), g^{(2)}(x))dx$$

where $g^{(1)}(\cdot)$ and $g^{(2)}(\cdot)$ represent the Gaussian functions. A suitable distance measure between univariate Gaussians is the negative log of the overlap. To deal with multivariate Gaussians with diagonal covariance matrices, the distance between corresponding univariate Gaussians is summed over all dimensions to finally give a distance measure:

$$\delta(\mathbf{g}^{(1)}(\cdot), \mathbf{g}^{(2)}(\cdot)) = \sum_{i=1}^{d} -\log O(g_i^{(1)}(\cdot), g_i^{(2)}(\cdot)),$$

where $\mathbf{g}$ is a multivariate Gaussian $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \mathbf{\Sigma})$ and $g_i(\cdot)$ is the univariate Gaussian $\mathcal{N}(x|\mu_i, \Sigma_{ii})$.

The use of the overlap-based distance measure in the Roadmap algorithm decreases the average reduction in total log probability per frame by a factor of 7 relative to the case where divergence is used and the measure may have utility in other applications where a distance measure between two Gaussians is required.

4.2 Setting Up The Similarity Relation

For the roadmap algorithm to operate, for each Gaussian a list of other similar Gaussians is required. Here follows a description of the algorithm used in obtaining it.

The first stage is to obtain, for each Gaussian a, a list of the closest n Gaussians in the system, according to the distance measure defined above. In experiments reported here, $n = 20$. The algorithm used to do this is described in Section 4.3.

The second stage adds to the similarity list of Gaussians close to a, those b such that a is in the list of b. This avoids the problem case where a Gaussian is not very close to any other Gaussians, and may never itself appear in any of these lists.

The third stage of building the similarity lists removes redundant entries: entries are not required if there already exists another indirect route via an intermediate Gaussian. Redundancy is defined more precisely in terms of the distance of the indirect route from a to b via c. The condition for the path between a and b being redundant is:

$$\exists c : \delta(a,c) < 0.9\delta(a,b) \,\wedge\, \delta(c,b) < 0.9\delta(a,b) \,\wedge\, \delta(a,c) + \delta(c,b) < 1.7\delta(a,b). \tag{13}$$

The removal of all these redundant links causes a modest increase in the performance of the Roadmap algorithm.

Finally the similarity lists for each Gaussian are sorted in order of distance with the closest Gaussians first in the list.

4.3 Setting up the initial similarity lists

As mentioned above, the first stage of the initialisation of the list of similar Gaussians consists of obtaining, for each Gaussian a, a list of the closest n Gaussians in the system. This process dominates the initialisation. A naive implementation would involve finding the distance between each pair of Gaussians, and would take time proportional to the square of the number of Gaussians in the system. This is clearly not suitable for very large HMM sets. An approximate scheme was therefore used which nevertheless found the closest n Gaussians almost without fail. The algorithm is iterative, requiring perhaps ten iterations, indexed by i, to converge. In the following description, $S_i(a)$ refers to the approximation at the i'th iteration to the list of the n closest Gaussians to a; it is a list of n or less Gaussians. Each iteration consists of two stages, as follows:

Initialisation Initialise $S_i(a) \leftarrow S_{i-1}(a)$ for all a, or to the empty set if $i = 0$.

Stage 1 For each pair of Gaussians a and c which are linked by roads via some other Gaussian b, we evaluate the overlap-based distance measure $\delta(a, c)$, and add a to $S_i(c)$ and vice versa. If as a result $|S_i(a)| > n$, we remove from it the furthest element from a; likewise for $S_i(c)$.

Stage 2 For each Gaussian a we test a number (20 in this case) of randomly chosen Gaussians, as in Stage 1. This is to "seed" the algorithm, and is done every time rather than just at the start because this was found to improve performance.

The algorithm iterates until the percentage change in the summed distances $\sum_a \sum_{b \in S_i(a)} \delta(a,b)$ is small (<0.05%). This usually happens in about ten iterations. Spot checks are carried out to test the accuracy of the algorithm, by finding the closest Gaussians to a small number of Gaussians by brute force and comparing them with the results of this algorithm. The n closest Gaussians are found almost without fail.

The essence of the algorithm is as described above. But there are some important details which are required for sufficiently fast operation. Firstly, it is important not to test the pair (a,c) any more than necessary, as calculation of overlap is time-consuming. Therefore, two optimisations are made. This first is that a hash table is used to store those Gaussians c for which the pair (a,c) has already been tested, for the current value of a. This avoids calculating the distance between a given pair more than once per iteration. The second optimisation aims to avoid calculating the distance between a given pair many times on different iterations. We store for each entry c in $S_i(a)$ the iteration at which it was added to that set. These numbers can be used to work out, for each a, those c which we know to have been tested on a previous iteration, and these distances do not have to be calculated again. This set of previously tested Gaussians is also stored as a hash table.

For further speedup, an approximation to the overlap formula for a single dimension of a Gaussian was used in finding the lists of closest Gaussians. The real formula is too inefficient, as it involves working out the places where the Gaussian likelihoods are equal and using a table of the Gaussian integral to calculate three separate areas. The alternative used was as follows. It only applies where $\sigma_2 \leq \sigma_1$; the Gaussians were swapped in the other case.

$$\sigma = \frac{\sigma_2}{\sigma_1} \tag{14}$$

$$\mu = \frac{\mu_2 - \mu_1}{\sigma_1} \tag{15}$$

$$\log(\text{overlap}) \simeq \frac{-0.0557}{\sigma} + 0.3137\sigma - 0.3292\mu^2 - 0.0037\frac{\mu^2}{\sigma} + 0.0429\mu^2\sigma - (0.3137 - 0.0557) \tag{16}$$

This approximation had very little effect on the lists of closest Gaussians. The real formula for overlap, rather than the approximation, was used when testing for redundant links using the criterion mentioned in Equation 13.

4.4 Finding the Best Gaussians

This section concerns the run-time operation of the Roadmap algorithm. It is a hill-climbing algorithm which for each speech frame starts from an initial set of Gaussians and aims to terminate having calculated a set of Gaussians including the most likely ones for the input speech vector. The initial set of Gaussians could either be a single random Gaussian or a number of the best Gaussians from the last speech frame. Firstly the log likelihood of each of the initial set of Gaussians is evaluated. For the Gaussians which are most likely the Gaussians closest to them (as determined by the similarity lists) are examined. The idea is that the algorithm will eventually go towards the region of Gaussians which are most likely given the input speech vector.

In this algorithm, we do not know when the most likely Gaussian in the entire system has been evaluated, so we use heuristics to tell us when to stop. At the end, all Gaussians $b_{j,m}$ which have been evaluated are returned, along with the calculated values $b_{j,m}(x^r(t))$. These can then be used to calculate the occupancies $\gamma^r_{j,m}(t)$ used in the EBW update equations.

In the following description of the Roadmap algorithm, Gaussian functions will be denoted a. The rule by which a Gaussian is chosen to be computed is as follows: from among those Gaussians which have already been evaluated, take the Gaussian a which gives the highest likelihood for the input. Then evaluate the first Gaussian in a's list, i.e, that closest to a, if it has not already been evaluated. Otherwise compute the next in a's list. If all Gaussians in a's list have been evaluated, the same procedure is followed for the Gaussian which gives the next best likelihood for the input. If all Gaussians in the lists of all those which have been computed have themselves also been evaluated, then evaluate a random Gaussian. This situation can occur if there are no links (“roads”) from an isolated region of Gaussians. The algorithm terminates when all the Gaussians close to a fixed number (perhaps 20) of the best Gaussians have been tested.

The set of Gaussians which is initially examined may consist of either a single arbitrary Gaussian or the best M Gaussians from the last input frame. In the experiments reported here, the best 20 from the last input frame were used. It is found that in practice the Roadmap algorithm can reliably find the most likely Gaussians in the system for each frame while only evaluating a small percentage of them (typically between 1 and 10%, decreasing with increasing system size).

4.5 Performance

The performance of the Roadmap algorithm is judged by the average number of Gaussians calculated per time frame and the average decrease in total likelihood of the input per time frame. This decrease in likelihood represents the sum of the Gaussian likelihoods that are not calculated by the algorithm. In tests on a HMM system with 9,500 Gaussian mixtures the Roadmap algorithm gave only a 0.004 decrease in log likelihood per frame while on average calculating 3.7% of the Gaussians in the system.

For comparison a number of different schemes of Gaussian selection based on vector quantisation (VQ) techniques, which have been widely reported in the literature to reduce the number of Gaussians computed in an HMM-based speech recognition, were also examined. One such VQ scheme with 256 codebook entries and using a two level VQ to speed up codebook entry calculation gave an average decrease in log likelihood per frame of 0.3 while computing 4% of the Gaussians in the system.

It is important to know what effect the calculation of only a fairly small subset of the Gaussians has on the performance of the trained models, i.e., what loss in total log likelihood is acceptable. Experiments showed that there was essentially no loss in recognition performance with a reduction in log likelihood per frame of up to 0.01 and the experiments reported below aimed to keep the approximation from using the Roadmap algorithm within this bound.

5 Experimental Evaluation

Speech recognition experiments to evaluate FD have been conducted on both the 1,000 word Resource Management (RM) task and on the North American Business (NAB) News task using a 65k word recognition system. In all cases initial MLE trained models were used and then subsequent FD training was performed.

5.1 Resource Management Experiments

For the RM experiments, a set of decision-tree state-clustered cross-word triphones were trained using MLE on the SI-109 training set (3990 utterances) using HTK in the manner described in [7]. The input speech for this system was parameterised as Mel-frequency cepstral coefficients (MFCCs) and the normalised log energy; and the first and second differentials of these values.

The final RM model set had 1577 clustered speech states and versions with a single Gaussian per state and 6 Gaussians per state were created. The models were tested using the standard word-pair grammar on the 4 RM speaker independent test sets (feb89, oct89, feb91 and sep92) which each contain 300 utterances.

After the MLE models had been created a number of iterations of FD training were performed on both the single Gaussian and 6 mixture component systems. Figure 4 shows that the FD objective function increases as training proceeds and gives the changes in error rate. Note that the 6-component system shows evidence of over-training.

	feb89	oct89	feb91	sep92	overall
MLE	6.99	7.68	7.49	11.61	8.44
FD iter 4	5.51	6.07	6.52	8.73	6.73

Table 1: % word error for single Gaussian RM system with MLE and FD training.

	feb89	oct89	feb91	sep92	overall
MLE	2.77	4.02	3.30	6.29	4.10
FD iter 4	2.81	3.39	2.90	5.94	3.76

Table 2: % word error for 6 Gaussian per state RM system with MLE and FD training

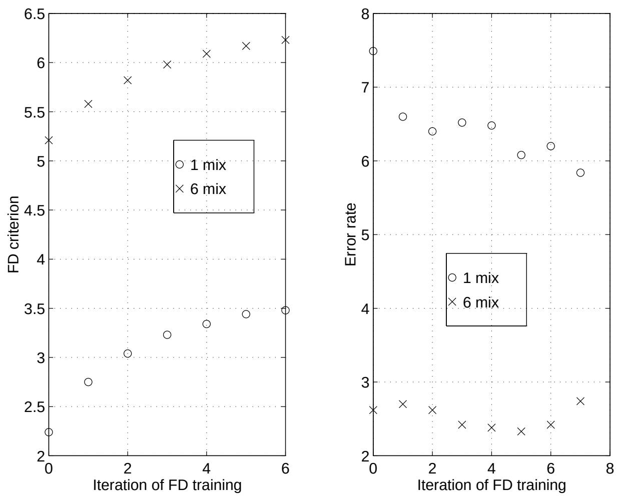

Figure 4: FD criterion and RM feb91 accuracy varying with time

Table 1 and Table 2 show the results of FD on the single and 6 Gaussian per state systems. The single Gaussian system shows an overall decrease in WER of 20.3% after 4 iterations of FD and the 6 mixture system an 8.3% reduction.

5.2 NAB Experiments

The HMMs used in these experiments were based on the HMM-1 set described in [6]. This decision-tree state-clustered cross-word triphone set of HMMs had 6399 speech states and was trained using MLE on the Wall Street Journal SI-284 training set (about 66 hours of data). Here a version of those models trained on cepstra derived from Mel frequency perceptual linear prediction (MF-PLP) analysis was used. Versions of these models with 1,2,4 and 12 mixture components per state were created using MLE, and then for each of these 4 iterations of FD training applied.

The models were tested on the 1994 DARPA Hub-1 development and evaluation test sets, which are denoted csrnab1_dt and csrnab1_et, using a trigram language model estimated from the 1994 NAB 227 million word text corpus. The same underlying HMM set (but trained using MFCCs) was used in [5] to evaluate the performance of lattice-based MMIE so this serves as a useful point of comparison.

Num mix	csrnab1_dt		csrnab1_et		% WER
Comps	MLE	FD	MLE	FD	reduction
1	13.64	11.95	15.64	14.32	10.4
2	11.84	10.58	13.19	12.04	9.7
4	10.67	9.77	11.25	10.84	6.0
12	9.30	8.99	9.96	9.85	2.2

Table 3: % word error rates on NAB test sets

Table 3 gives the performance of the FD on NAB and shows that the reduction in WER decreases as model complexity increases. The single and two Gaussian per state systems have a 10% relative word error reduction while the 12 mixture component system has a reduction in error of just 2%. However it should be noted that the FD models gave improvements over MLE in all cases.

Table 4 compares the NAB reductions in word error for the comparable tests tests reported in [5]. The results are encouraging, with FD giving more improvement than MMIE in most cases.

Num Mix	csrnab1_dt		csrnab1_et	
Comps	FD	MMIE	FD	MMIE
2	10.6	8.4	8.7	8.8
12	3.3	0.6	1.1	-1.2

Table 4: Comparison of FD and MMIE systems giving % word error reductions relative to MLE

5.3 Computational Cost of FD

For the experiments above the computational cost of FD is very important. As previously discussed, the most computationally intensive part of FD training is calculating the occupation probabilities and finding the most likely Gaussians in the system. Using the Roadmap algorithm, calculation of the these denominator occupancies for FD took about five times as long as for the numerator, meaning that this implementation of FD is about six times slower than conventional MLE training. The efficient lattice-based MMIE training procedure discussed in [5] is 15-20 times slower than MLE (ignoring the time to create the initial word lattices). Therefore it appears that FD is about three times faster than the lattice based MMIE procedure.

6 Conclusions

The report has described an implementation of FD training. FD is a promising objective function which seems to give good results for the tasks reported here. It has described the Roadmap algorithm which aims to find the most likely Gaussians from a large set of Gaussians, without calclulating all the conditional probabilities. A distance measure based on overlap (used in the Roadmap algorithm) was introduced. An investigation was made into the best way to set the smoothing constant in the EBW equations, with substantial improvements in convergence and recognition performance as a result of the changes made, and a new set of mixture update equations, with an interesting theoretical basis, was introduced.

Results reported here show that FD gives considerable reductions in word error for simple models and also gives useful increases in accuracy for more more complex speech models with more mixture components. The improvements from FD are comparable or greater than those given by MMIE on the tasks reported here, and FD as implemented here is more computationally efficient.

References

[1] Gopalakrishnan P.S., Kanevsky D., Nadas A. & Nahamoo D. (1991) An Inequality for Rational Functions with Applications to Some Statistical Estimation Problems. *IEEE Trans. on Information Theory* **37**, No. 1, pp 107-113.

[2] Kapadia S. (1998) Discriminative Training of Hidden Markov Models, *Ph.D. thesis, Cambridge University Engineering Dept.*

[3] 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.*

[4] Normandin Y. (1991) An Improved MMIE Training Algorithm for Speaker-Independent, Small Vocabulary, Continuous Speech Recognition, *ICASSP'91* pp. 537-540

[5] Valtchev V., Odell J.J., Woodland P.C. & Young S.J. (1997) MMIE training of large vocabulary speech recognition systems. *Speech Communication*,**22**, pp. 303-314.

[6] 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*, Vol. 1, pp. 73-76, Detroit.

[7] Young S.J., Odell J.J. & Woodland P.C. (1994) Tree-based State Tying for High Accuracy Acoustic Modelling. *Proc. Human Language Technology Workshop*. pp. 307-312, Plainsboro, NJ.

[8] Valtchev, V., Woodland, P.C., Young, S.J., 1996. Lattice-based discriminative training for large vocabulary speech recognition, *ICASSP'96*, Vol. 2, pp. 605-608

[9] 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. 613-61, Atlanta.

[10] Normandin Y, Lacouture R. & Cardin R. MMIE Training for Large Vocabulary Continuous Speech Recognition. *ICSLP 94*, pp. 1367-1370.

[11] Chow Y.L, Maximum Mutual Information Estimation of HMM Parameters for Continuous Speech Recognition using the N-Best Algorithm, *Proc ICASSP'90*, Vol. 2, pp. 701-704, Albuquerque.

[12] 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. Internat. Conf. Acoust. Speech Signal Process., Tokyo*, pp. 49-52.

[13] Merialdo B., 1988. Phonetic recognition using hidden Markov models and maximum mutual information training. *Proc. Internat. Conv. Acoust. Speech Signal Processing, New York*, Vol. 1, pp. 111-114.