FRAME DISCRIMINATION TRAINING OF HMMS FOR LARGE VOCABULARY SPEECH RECOGNITION

D. Povey & P.C. Woodland

Cambridge University Engineering Dept, Trumpington St., Cambridge, CB2 1PZ U.K.
Email: {dp10006,pcw}@eng.cam.ac.uk

ABSTRACT

This paper describes the application of a discriminative HMM parameter estimation technique called Frame Discrimination (FD), to medium and large vocabulary continuous speech recognition. Previous work has shown that FD training can give better results than maximum mutual information (MMI) training for small tasks. The use of FD for much larger tasks required the development of a technique to be able to rapidly find the most likely set of Gaussians for each frame in the system. Experiments on the Resource Management and North American Business tasks show that FD training can give comparable improvements to MMI, but is less computationally intensive.

1. INTRODUCTION

Previous research has shown that the accuracy of a speech recognition system trained using Maximum Likelihood Estimation (MLE) can often be improved further using discriminative training. All such techniques normally give much greater improvements in recognition accuracy on the training data than on the test set except where the number of parameters to be estimated is very low. Furthermore the computation required to optimise discriminative objective functions is much higher than for standard Baum-Welch (i.e. MLE) training.

This paper investigates a recently proposed [2] discriminative objective function called Frame Discrimination (FD). In [2] it was shown that on small isolated word recognition tasks FD gave improved generalisation compared to maximum mutual information estimation (MMIE) and yielded superior test-set accuracy. Here we investigate the extension of FD to large vocabulary continuous speech recognition.

FD consists of a class of objective functions, of the form:

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

where $\mathcal{O}_r$ represents the speech data for the r'th utterance, and $\mathcal{M}_{w_r}$ the model corresponding to its transcription w_r. The HMM $\mathcal{N}$, also termed the *denominator* model (while $\mathcal{M}_{w_r}$ is the *numerator* model), is derived from the model $\mathcal{M}^{\mathrm{gen}}$ which is used to recognise speech. If $\mathcal{N}$ were equal to $\mathcal{M}^{\mathrm{gen}}$, we would have the MMIE objective function. However in order to improve generalisation, FD uses a model $\mathcal{N}$ which is less constrained than the recognition network. In particular we focus on zero memory frame discrimination. In this case, $\mathcal{N}$ 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 $x^r(t)$ is the vector representing frame t of utterance r which is of length $T(r)$, 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 the recognition model $\mathcal{M}^{\mathrm{den}}$, i.e, all states in all HMMs in the system. The term $P(q_i|N)$ represents the prior probability of observing state q_i. This prior probability can conveniently be found from the state occupation counts from the forward-backward algorithm used in MLE training.

2. EXTENDED BAUM-WELCH RE-ESTIMATION

To optimise the parameters of HMMs when using rational objective functions such as FD, the extended Baum-Welch (EBW) re-estimation formulae can be used. The EBW algorithm for rational objective functions was introduced in [1] and developed in [3] for the continuous density HMMs considered here. The EBW algorithm has been successfully applied to MMIE optimisation for both small and large vocabulary tasks [5].

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.

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

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

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

where the superscript den indicates the denominator HMM $\mathcal{N}$. $\theta_{j,m}(\mathcal{O})$ represents the sum of the vectors x of the training data weighted by the probability of occupying mixture m of state j at that time frame when x occurs, i.e:

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

where $\gamma^r_{j,m}(t)$ is the probability of occupying mixture m of state j at time t. $\gamma_{j,m}$ represents the estimated count of the number of times mixture component m of state j is occupied:

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

The constant D was set on a phone-by-phone basis as in [5], subject to a floor at the maximum of $\gamma_{j,m}$ and $\gamma^{\text{den}}_{j,m}$ in the phone. The use of a floor was found to improve both convergence of the FD criterion and recognition performance.

The standard update equations for the mixture weights [3, 5] can be used for estimating the mixture weights or as an alternative the formulation given in [4] can be used.

2.1. Implementation Considerations

In re-estimating the parameters it is necessary to calculate the posterior probability of each Gaussian in the system for each input vector i.e.

$$\gamma^{r,\text{den}}_{j,m}(t) = \frac{c_{j,m} b_{j,m}(x^r(t)) P(q_i|\mathcal{N})}{\sum_{j \in \mathcal{M}_{\text{rec}}} \sum_{m=1}^{M_j} c_{j,m} b_{j,m}(x^r(t)) P(q_j|\mathcal{N})} \quad (1)$$

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 and M_j the number of Gaussians in the mixture for state j. Therefore $b_{j,m}(x^r(t))$ must be calculated for each Gaussian in the system and for every time frame and this calculation dominates the overall computation required. Furthermore, for large vocabulary speech recognition the HMM sets used often contain a very large number of Gaussian components and therefore complete computation of the denominator of (1) would make the algorithm impractical.

To make FD practical for large HMM systems (1) 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 (1) 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.

3. THE ROADMAP ALGORITHM

The purpose of this algorithm is to find those Gaussians which best match the input for each time frame, while minimising computation. Associated with each Gaussian in the system is a list of similar Gaussians which is used to navigate towards the best Gaussian in the system. A detailed description of the algorithm is given in [4].

3.1. 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 overstates the difference between the Gaussians when they have very different variances. Therefore an alternative distance measure was sought and one based on Gaussian "overlap" developed.

Here the overlap between two univariate Gaussians is defined as:

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

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:

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

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

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.

3.2. Setting Up The Similarity Relation

For the roadmap algorithm to operate, for each Gaussian a list of other similar Gaussians is required. This set of lists is called the "roadmap" and following the analogy the lists represent the nearby set of Gaussians to which there are "roads".

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, $n = 20$. A naive implementation would involve finding the distance between each pair of Gaussians, and would have taken 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 iterative scheme was therefore used which avoids this exhaustive search, but finds the n closest Gaussians almost without fail. On each iteration, the algorithm only examines Gaussians for potential inclusion in a's list that are already in the similarity lists for Gaussians currently directly "connected" to a. At the end of each iteration the n most similar Gaussians are placed in a new list for a. Therefore on each iteration the distance between at most n^2 Gaussian are computed for each a. However the number is considerably less than this since redundant computation is avoided.

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 appears 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:

$$\delta(a,c) < 0.9\delta(a,b) \wedge \delta(c,b) < 0.9\delta(a,b) \wedge \delta(a,c) + \delta(c,b) < 1.7\delta(a,b).$$

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 which the closest Gaussians first in the list.

3.3. Finding the Best Gaussians

The Roadmap algorithm is a hill-climbing algorithm which for each speech frame starts from an initial set of Gaussians and aims to terminate with the most likely Gaussians for the input speech vector. Firstly the log likelihood of each of the initial set of Gaussians is evaluated. For the Gaussians which are most likely the

Gaussians close to those (from 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.

For such an algorithm, there is no way to know when (or if) the most likely Gaussian in the entire system has been evaluated. The best that can be done is to evaluate a fixed number of Gaussians N, and hope that the best Gaussians will be among them. 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 extended BW 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 set of Gaussians which is initially examined consists 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).

3.4. Performance

The performance of the Roadmap algorithm is judged by two measures: 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 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.

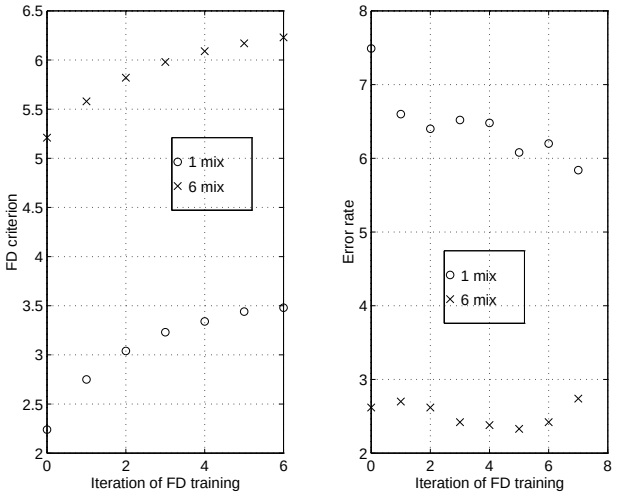

Figure 1: FD criterion and RM feb91 accuracy varying with time

4. 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.

4.1. 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 1 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.

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

	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

overall decrease in WER of 20.3% after 4 iterations of FD and the 6 mixture system an 8.3% reduction.

4.2. 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 different numbers of 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.

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 (from [5]) systems giving % word error reductions relative to MLE

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.

4.3. 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 five 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.

5. CONCLUSIONS

The paper has reported an implementation of FD training, and introduced the Roadmap algorithm which finds the set of most likely Gaussians in the system and is key to the efficient implementation of FD with large HMM sets. A distance measure for Gaussians based on the notion of overlap was introduced and shown to be very effective.

Experimental results 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 are similar to those previously reported for MMIE but the FD implementation is considerably more computationally efficient.

6. 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] Povey D. & Woodland P.C. (1998) *An Investigation of of Frame Discrimination for Continuous Speech Recognition.* Technical Report CUED/F-INFENG/TR.332, Cambridge University Engineering Dept.

[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.