AN IMPLEMENTATION OF MMI TRAINING 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 an implementation of MMI for large-vocabulary speech recognition, which has been applied to the Switchboard telephone speech corpus, and is, to the authors' knowledge, the largest-scale successful application of MMI training reported at this time. The techniques described were used in the Cambridge submission to the 2000 hub5 (Switchboard) evaluation, giving the lowest error rate of any of the submissions. The contribution of the MMI training was 2.5% absolute change in error rate giving 25% word error rate. Features of the implementation include the use of lattices to speed up the alignment, and alterations to the Extended Baum-Welch update equations.

1. INTRODUCTION

MMI training of HMMs is based on an objective function that looks like:

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

i.e, the probability of the speech data given the correct word sequence, divided by the probability for all possible word sequences. The union of all possible word sequences may be represented by a (very large) HMM, giving the alternative form:

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

where $\mathcal{M}_{\mathrm{rec}}$ is a recognition model. Such an approach might ordinarily involve doing a recognition pass on all the training data for each iteration of MMI training; but in [6] a recognition lattice (i.e, a lattice of words output by a recogniser) was used to constrain the alignment. Such an approach has been taken here.

The update equations used here for Gaussian updates are the Extended Baum-Welch update equations, first applied to continious HMMs by Normandin [3]. Theese equations are similar in appearance to the updates used for Maximum Likeliood estimation (ML), except that two sets of accumulators are used, and the equations involve a subtraction and contain a "smoothing" constant. The original presentation of the Extended Baum-Welch update equations [1] involved global smoothing constants; the lattice-based implementation in [6] had phone-level smoothing constants; the present implementation sets the smoothing constant at the level of the individual Gaussian but using an altered rule.

Novel update equations, not requiring smoothing constants, have been used for the transition probabilites and mixture weight updates; they will be described below.

Another change was that, while calculating the alignments, the likelihoods were scaled down by the normal language model scale factor, leaving the language model unscaled. This has the effect of increasing the diversity of states which are aligned to a particular frame, thus making a more robust objective function (less dependent on the choice of training data).

2. UPDATE EQUATIONS

2.1. Gaussian updates

The Extended Baum-Welch update equations for a single dimension of the mean and variance vectors are as follows:

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

The sums $\theta_{j,m}(\mathcal{O})$ and $\theta_{j,m}(\mathcal{O}^2)$ are sums of data and squared data respectively, weighted by occupancy, for mixture component m of state j. The occupancies (summed over time) are $\gamma_{j,m}$. The superscripts num and den refer to the model corresponding to the correct word sequence, and the model corresponding to the lattice of most plausible word sequences, respectively. $\hat{mu}$ and $\hat{sigma}$ are the new (updated) parameters. The equations given above are similar in form to the normal Maximum Likelihood update equations, and the occupancies and sums of data are defined in exactly the same way.

In previous work, the constant D has been set either as a global constant or at the phone level. It is generally set at twice the lowest positive value necessary to ensure all positive variance updates. It may be set at a global level, or on a per-phone basis as in [6]. In these experiments, D is set at a per-Gaussian level to the highest of the following two values: twice the value necessary to ensure positive variance updates, or: a further constant E times the denominator occupancy ($E\gamma^{\mathrm{den}}_{j,m}$). The value of the constant E then becomes important for smoothing. The values tried in these experiments were: $E = 1$, $E = 2$, and $E = \frac{1}{2}$ the maximum value of $\frac{D}{\gamma^{\mathrm{den}}_{j,m}}$ for any mixture, which results in $E = 2.0$ or higher for switchboard (rising as training progresses); it is more system-independent rule. These three approaches will be referred to as $E = 1$, $E = 2$, and $E = \mathrm{halfmax}$.

2.2. Mixture weight and transition probability updates.

The original Extended Baum-Welch update equation for mixture weights is not very effective, and instead an altered version by Merialdo [10] is generally used; however, the altered equations are only justified empirically and cannot be proved to work. The update rule suggested here can be justified, although a few assumptions have to be made. Moreover, the update is free of smoothing constants, and informal experiments with a different discriminative objective function showed it to have better performance than Merialdo's approximation.

The update consists of finding the mixture weights $\hat{c}_m$ which maximise the following auxiliary function:

$$F = \sum_{m=1}^{M} \gamma_m^{\text{num}} \log \hat{c}_m - \frac{\gamma_m^{\text{den}}}{\bar{c}_m} \hat{c}_m, \qquad (1)$$

subject to the sum-to-one constraint. $\bar{c}_m$ are the original weights and γ_m are the mixture occupancies; the state subscript j is omitted for brevity. The optimisation may be performed using a generic function-optimisation routine; but for faster updates the following procedure is recommended: set $\hat{c}_m^0 = \bar{c}_m$ for all m, and for $t = 0$ to, say, $10m$, update the weights as follows: Choose a weight m, and work out the updated value $\hat{c}_m^t$ which optimises the function F, assuming the other weights are scaled by $\frac{1-\hat{c}_m^t}{1-\hat{c}_m^{t-1}}$ to enforce the sum-to-one constraint. F may be written as a function of $\hat{c}_m^t$ in the form: $F = A \log \hat{c}_m^t - B\hat{c}_m^t + C \log(1 - \hat{c}_m^t) - D(1 - \hat{c}_m^t)$. Differentiating by $\hat{c}_m^t$, and renaming to x, we get: $\frac{\delta F}{\delta x} = \frac{A}{x} - B + D - \frac{C}{1-x}$, where all constants are ≥ 0. The special cases are: $A = 0$, $-B + D - C \leq 0 \rightarrow x = 0$; $C = 0, A - B + D \geq 0 \rightarrow x = 1$; failing those, multiplying by $(x)(1 - x)$ generates an expression which is quadratic (at most) in x, any zeros of which between 0 and 1 correspond to an optimal updated value.

A proof that maximising the auxiliary function of Equation 1 will increase the objective function is given in [4]. It is based on the assumption that as each mixture weight is varied, the occupancies that would be obtained from forward-backward alignment will vary by a factor that is between 1 and the ratio of the new to the old mixture weights. For purposes of the proof the mixture weight occupancies must also be assumed to be independent of other parameters in the HMM.

Experiments on a number of different systems have shown that this update rule optimises the criterion faster than the Baum-Welch/Merialdo update in most cases.

The update for a row of a transition matrix is the same as the update for a set of mixture weights, in that there is a sum-to-one constraint and we have transition counts (similar to occupancies) for both the numerator and denominator HMM. The same technique is used. Note that for both the mixture weights and transition probabilities, if the denominator occupancies are zero the update is equivalent to the ML update.

3. LATTICE-BASED IMPLEMENTATION

In [6], the use of recognition lattices to implement large-scale MMI was reported. The lattice format which HTK supports is composed of nodes which represent instants in time, and vertices which represent words. The words can optionally be labeled with their phones-in-context, with start and end times for each phone. That implementation of the Forward-Backward algorithm for the word lattices was based on calculating the likelihood corresponding to each lattice vertex (i.e, word), given the start and end time for the word, and then combining these likelihoods to get the alpha and beta values for each time frame within each word, as in the Forward-Backward algorithm.

There are two different implementations used in the experiments reported here, which will be labeled *exact-match* and *full-search*. The *exact-match* experiments are similar to the ones mentioned above, in that the log likelihood of each segment was calculated based on its start and end times in the lattice, and the occupancies of the segments were then calculated by combining this information. However, the segments in this case were phone segments, not word segments. The *full-search* implementation sought to do a full forward-backward search constrained by the lattice, optimised by using the lattice times (extended by a short period) for pruning. In both cases, the search was also optimised as far as possible by combining redundantly repeated models. Different optimisations were possible in the two cases. In the *exact-match* case, only one forward-backward pass is necessary for a given model with given start and end times, no matter how many times it appears in the lattice. In the *full-search* case, if two instances of a model in the lattice may be combined to make an equivalent but smaller lattice, this can be done regardless of the start and end times. On the whole, the *exact-match* procedure seems to be simpler to code, quicker to run, and at least as effective.

An additional essential detail is that, as in [6], if the numerator (correct word sequence) happens not to be present in the recognition lattice, it is added to it before processing.

3.1. Details of *full-search* implementation

In both implementations, the starting point is bigram recognition lattices marked with times at the phone level, and with a lattice depth (average number of arcs per time frame) of several hundred. For *full-search*, the lattice is compacted as follows: if two phones share the same phone context, are in the same position in the same word, and overlap in time, they are combined; the extent in time of the new phone is the union of those of the two original ones. The set of transitions of the combined phone is set to the union of the original sets of transitions, with duplicates removed; in a bigram or unigram grammar, the inter-word transitions' probabilities should never disagree. This process of lattice reduction decreases the lattices' depth by a factor of about 10. A full forward-backward search on the resulting lattice is then performed, with the time information of each phone, extended by a small margin, used for pruning. Probability scaling is done by scaling down the log probabilities of the state output distributions. This spreads out the time alignments as well as increasing the diversity of words with significant probabilities.

3.2. Details of *exact-match* implementation

The approach of the *exact-match* is to calculate the likelihood of each phone segment in the lattice, based on its start and end times, and then calculate the occupancy of each phone segment using a Forward-Backward-like algorithm. These occupancies can then be used in a second ("backward") pass across each individual phone segment, in which accumulators are stored. There are two advantages to this approach: firstly, there may be many instances of a phone with a particular start and end time, which can exploited for a speed increase. Secondly, the segment-level log likelihoods can be scaled while calculating segment occupancies; this is a way

of scaling down sentence log likelihoods as in Equation 2 without spreading out the time alignments. A different solution to the same problem suggested in [9] is to calculate word-posterior probabilities and combine these with within-word Viterbi alignments. In fact, as far as recognition is concerned it does not seem to be essential to solve this problem, as the *exact-match* and *full-search* techniques give similar results.

4. SCALED-DOWN PROBABILITY

A problem with MMI training of HMMs is that overfitting to the training data can take place, resulting in good training set but poor test set performance. One attempt to overcome this problem is the Frame Discrimination criterion [5], [4], which discriminates against a one-state model which is a weighted sum of all states in the recognition HMM. What has been done here is to scale down the likelihoods while performing Forward-Backward alignment, in order to increase the range of alignments. The scale chosen is the inverse of the normal language model scale factor, so in effect the output probabilities are scaled down while the language model probabilities are left unscaled. The intended result is that at most points in time, more than one word has a significant probability. In the *exact-match* case, the calculated likelihoods of the segments are scaled down while calculating segment occupancies, while in the *full-search* case it is the state output likelihoods which are scaled, leaving the transition probabilities unscaled. It seems that the former technique would be more likely to have the desired effect, by leaving realistic segmentations within words and phones while increasing the diversity of words matched to the data. Experiments have shown the *exact-match* style of alignment to give at least as good recognition results as *full-search*, at twice the speed and with simpler code.

4.1. Effect of probability scaling

In order to measure the effect of probability scaling, the average number of states with an occupancy greater than 0.01 was measured for both the numerator and denominator HMMs on four different systems: *full-search* and *exact-match*, both with and without probabilty scaling.

	Probability scale			
	1/12		1	
	num	den	num	den
Full-search	3.54	8.16	1.43	1.63
Exact-match	1.78	5.58	1.26	1.45

As expected, probability scaling significantly increased the number of states which matched each frame of the data. The *exact-match* implementation reduced the diversity of states by restricting the phone alignment and by keeping an unscaled alignment within phones.

5. LATTICE-BASED FRAMEWORK FOR OTHER DISCRIMINATIVE CRITERIA

.

A general formulation for the discriminative training criterion, proposed in [9], is:

$$F_D(\lambda) = \sum_{r=1}^{R} f\left(log \frac{P^\alpha(\mathcal{O}_r|M_{w_r})P^\alpha(w_r)}{\sum_{w\in\mathcal{W}_r} P^\alpha_\lambda(\mathcal{O}_r|\mathcal{M}_w)P^\alpha(w)} \right) \quad (2)$$

where the choice of the exponent α, the smoothing function f and the set $\mathcal{W}_r$ of word sequences for discrimination decide which criterion is represented. For conventional MMI training, we would choose $f(x) = x$, $\mathcal{W}_r$ = all word sequences (or the most likely ones), and $\alpha = 1$ (or ∞ for corrective training, in which we only consider the best recognised word sequence). For MCE training we would use $f(x) = -1/[1 + \exp(2\beta x)]$, α set to any value (empirically), and the correct word sequence w_r would be excluded from $\mathcal{W}_r$.

5.1. Excluding the correct sequence

In the work presented here, the denominator of the expression for $F_D(\lambda)$ is represented by the recognition model M_{rec} (or a recognition lattice derived from it). This works for the MMI case, in which we want to include the correct sequence; but it is not obvious how to exclude the correct sequence from the lattice if MCE training is desired. There are two ways in which the MCE effect could be obtained: firstly, by noticing that there is a simple relationship between the values of: $\frac{P^\alpha(\mathcal{O}_r|M_{w_r})P^\alpha(w_r)}{\sum_{w\in\mathcal{W}_r} P^\alpha_\lambda(\mathcal{O}_r|\mathcal{M}_w)P^\alpha(w)}$ (i.e, the numerator probability divided by the denominator probability), in the case where the correct sequence has and has not been excluded (it is simply the difference between $\frac{a}{b}$ and $\frac{a}{a+b}$. To obtain the MCE value we would just have to replace $f(x)$ with $f(\frac{1}{\frac{1}{x}-1})$. Alternatively, the correction could be done at the level of the accumulators (sums of input values), by using the difference between the numerator and denominator probabilities to calculate what proportion of the likelihood of the the denominator model M_{rec} is due to the correct word sequence; subtracting the accumulators corresponding to the correct word sequence, and scaling up the remainder. Experiments on Resource Management showed that the exclusion of the correct word sequence from the denominator model (while leaving $f(x) = x$, i.e not fully simulating MCE) makes essentially no difference to recognition accuracy; however, this may be due to the likelihood scaling, which ensures that even correctly recognised files contribute to discrimination (which they would not do to a significant extent in conventional MMI).

5.2. Probability scaling

As mentioned above, all probabilities were scaled down by the inverse of normal language model scale, to increase the diversity of states. In the *full-search* implementation, the state output likelihoods were scaled; in the *exact-match* implementation, the segment likelihoods were scaled. The language model itself was scaled back down to its unscaled form. The method of scaling employed in the *exact-match* experiments is equivalent to the the general formulation in Equation 2, if we assume that the language model probabilities $P(w)$ in that equation are scaled relative to the acoustic probabilities, as is normally done in speech recognition systems. The scaling employed in the *full-search* implementation, on the other hand, is not equivalent to the scaling in Equation 2, since the alignments within word sequences are affected.

6. CHANGES TO PARAMETERS

For the HMM set trained with 65 hours of data that had the best performance on the eval97sub test data (*exact-match*, E=halfmax, 4th iteration), the parameters were compared to the starting HMM model to see what kinds of differences were present. The difference in the means (normalised by the standard deviation) was approximately Gaussian, 90% within ± 0.25. The relative change in the variance was also approximately Gaussian, 90% within $\pm 25\%$. The log change in the weights was 90% within ± 0.7. Almost all of the self-transitions had decreased (on average by around 20%); the other transitions had increased by about the same factor, but this may be an artefact from the way probability scaling is done. Most of the distributions had become more compact (as measured by the sum of logs of variances), but this may arise from the update equations, which change the variance in linear rather than log space; the distribution of log changes to the variance therefore has a longer tail on the left.

As training progressed, the parameters became progressively more different from the starting parameters, but the differences between subsequent iterations decreased slightly.

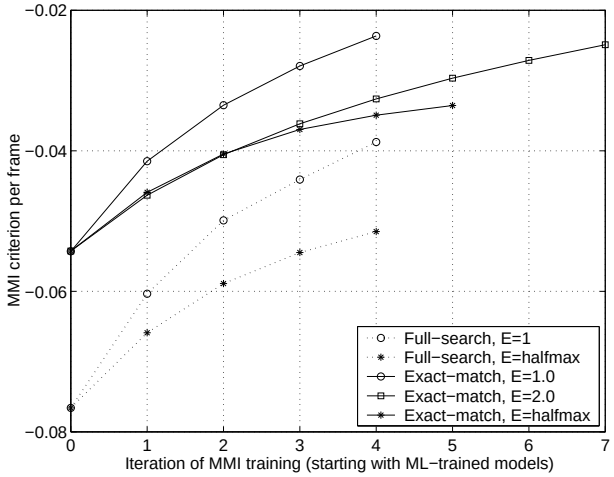

Figure 1: MMI criterion optimisation on Switchboard

7. EXPERIMENTS

7.1. Function-optimisation/generalisation tradeoff

The experiments in Figure 2 explore the effect of two different update options on test set accuracy and criterion optimisation. One is the *full-search/exact-match* choice of alignment method, which shows no important difference in recognition accuracy; however, *exact-match* (which is described in detail above) is recommended on account of its greater simplicity and speed. The other parameter explored is the value of E, a constant used to set a floor on D, the smoothing constant in the Gaussian update. The options tried are $E = 1$, $E = 2$, and $E = \mathrm{halfmax}$ (defined above). The results show that with less smoothing ($E = 1$) the criterion is optimised faster, but the models suffer from overtraining and generalise poorly to the test data. The more smoothed update rule ($E = \mathrm{halfmax}$) optimises the criterion slower but fares better on recognition. This shows that optimising the MMI criterion (even with probability scaling) is not sufficient to generate good models. An alternative, and more elegant, solution to the problem is to define a different criterion which is a weighted sum of the ML criterion and the MMI criterion. An experiment (not shown on the graph) was performed to optimise the criterion: $F_{MMI} + 0.25F_{ML}$. The setup was: *exact-match*, and $E = 1$ for fast optimisation. The error rate consistently decreased, and stabilised at 43.7% error on the 5th iteration. The models had changed very little relative to the original ones (90% of means within 0.1 standard deviations of the originals, compared to 0.25 for similarly performing MMI-trained models). The optimisation was performed as for normal MMI optimisation, except that all numerator-model accumulators were scaled by 1.25 prior to optimisation. This shows that an interpolated criterion (or H-criterion [**?**]) may be a more effective, and more principled, way to control generalisation than interfering with the function-optimisation routines, but it has not yet been thoroughly explored.

7.2. Other parameters

There were other parameters whose effect was not well explored, due to time constraints. One of these is the probability scale used; the inverse of the normal language model scale seemed a natural choice, and has been used in experiments reported here. Preliminary experiments showed it to work well. Another parameter was the model size. It is generally found that discriminative training works well with a smaller model size, but this was not varied.

8. CONCLUSIONS

In order to implement MMI on a large-scale system, three major issues have been addressed: speed, optimisation, and generalisation. The first issue, speed of implementation, concerns the Forward-Backward phase of the calculation. The speed has been increased by using word nets to constrain the search, removing redundancy from the nets wherever possible, and constraining the search by using their time information. The second issue is optimisation of the MMI criterion. The Extended Baum-Welch equations have been used here, with modifications; the most important one is to set the smoothing constants in a different way. Since the best performance with MMI is not acheived at the optimum of the MMI criterion but at an intermediate stage of training, the optimisation must not only approach the optimum of the objective function but approach it in a smooth and recognition-friendly way. The third issue is generalisation. The approach taken was to weight all log likelihoods down when calculating word-level occupancies as in [9]. The result of all these changes was that MMI training made a significant increase in Switchboard accuracy (2-3% absolute), relative to ML-trained starting models.

An outstanding problem with this kind of MMI training is that there is not an obvious relationship between the value of the criterion and the recognition performance on the test set– e.g, values of smoothing constants which optimise the criterion well result in poor recognition. It would seem desirable to find a criterion which would correlate better with test set recognition performance.

9. REFERENCES

[1] Gopalakrishnan P.S., Kanevsky D., Nadas A. & Nahamoo D. (1991) An Inequality for Rational Functions with Applications

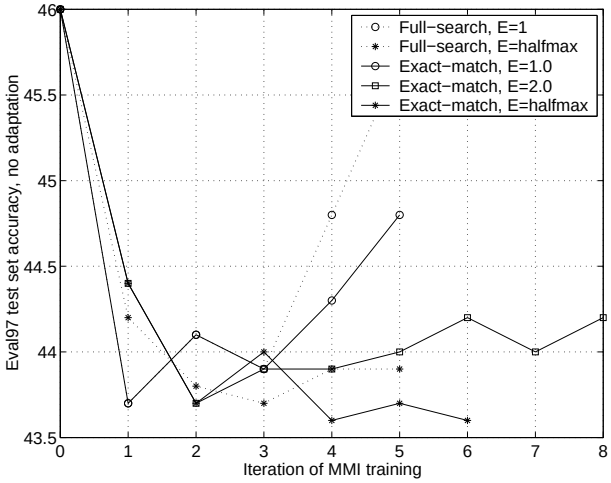

Figure 2: Accuracy, training Switchboard 65h subset without adaptation

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] Gopalakrishnan P.S., Kanevsky D., Nadas A., Nahamoo D., Picheney M.A. (1988) *Decoder Selection Based on Cross-Entropies*, ICASSP'98 Vol. 1 pp. 20-23.

[6] Povey D. & Woodland P.C. (1999) *Frame Discrimination Training of HMMs for Large Vocabulary Speech Recognition*, ICASSP'99

[7] 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.

[8] 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.

[9] 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.

[10] Schluter& Macherey, "Comparison of Discriminative Training Criteria", *Proc. ICASSP'98*, vol 1, pp. 493-6

[11] Merialdo B., 1988. "Phonetic recognition using hidden Markov models and maximum mutual information training." *Proc. ICASSP, New York, 1998, Vol. 1, pp. 111-114*