FMPE: DISCRIMINATIVELY TRAINED FEATURES FOR SPEECH RECOGNITION

Daniel Povey, Brian Kingsbury, Lidia Mangu, George Saon, Hagen Soltau, Geoffrey Zweig [*]

IBM T.J. Watson Research Center, NY; {dpovey,bedk,mangu,gsaon,hsoltau,gzweig}@us.ibm.com

ABSTRACT

MPE (Minimum Phone Error) is a previously introduced technique for discriminative training of HMM parameters. fMPE applies the same objective function to the features, transforming the data with a kernel-like method and training millions of parameters, comparable to the size of the acoustic model. Despite the large number of parameters, fMPE is robust to over-training. The method is to train a matrix projecting from posteriors of Gaussians to a normal size feature space, and then to add the projected features to normal features such as PLP. The matrix is trained from a zero start using a linear method. Sparsity of posteriors ensures speed in both training and test time. The basic technique gives similar improvements to MPE (around 10% relative); MPE on top of fMPE results in error rates up to 6.5% relative better than MPE alone, or more if multiple layers of transform are trained.

1. INTRODUCTION

This article introduces fMPE, a method of discriminatively training features. The MPE objective function is reviewed in Section 2; Sections 3 and 4 describe fMPE; Section 5 discusses some issues relating to its use. Datasets and experimental conditions are described in Section 6, and experimental results are presented in Sections 7, 8, 9 and 10. Conclusions are given in Section 11.

2. MINIMUM PHONE ERROR (MPE)

The Minimum Phone Error (MPE) objective function for discriminative training of acoustic models was previously described in [1, 2]. The basic notion is the same as other discriminative objective functions such as MMI, i.e. training the acoustic parameters by forcing the acoustic model to recognize the training data correctly.

The MPE criterion is an average of the transcription accuracies of all possible sentences s, weighted by the probability of s given the model:

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

[*]Thanks to Michiel Bacchiani, Upendra Chaudhari, Olivier Siohan and Bhuvana Ramabhadran for help with the Callcenter, Malach and Broadcast News experiments.

where $P_{\lambda}^{\kappa}(s|\mathcal{O}_r)$ is defined as the scaled posterior sentence probability $\frac{p_{\lambda}(\mathcal{O}_r|s)^{\kappa}P(s)^{\kappa}}{\sum_u p_{\lambda}(\mathcal{O}_r|u)^{\kappa}P(u)^{\kappa}}$ of the hypothesized sentence s, where λ is the model parameters and $\mathcal{O}_r$ the r'th sequence of acoustic data.

The function $\mathrm{A}(s, s_r)$ is a "raw phone accuracy" of s given s_r, which equals the number of phones in the reference transcription s_r for file r, minus the number of phone errors.

3. FMPE

3.1. Overview of fMPE

fMPE is a form of discriminative training that optimizes the same objective function as MPE, but does so by modifying the features. The basic transformation is:

$$\mathbf{y}_t = \mathbf{x}_t + \mathbf{M}\mathbf{h}_t, \qquad (2)$$

where $\mathbf{x}_t$ are the original features on time t and $\mathbf{y}_t$ the modified features. $\mathbf{h}_t$ are high dimensional features calculated at each frame t, which may be a function of the original features $\mathbf{x}_t$. These are projected down with the matrix $\mathbf{M}$. If very high dimensions are used, it is important that the features $\mathbf{h}_t$ are sparse, i.e. very few of the elements of the vector are nonzero on each time frame. This means that very few of the rows of $\mathbf{M}$ have to be accessed on each time frame. The reason for adding the original features $\mathbf{x}_t$ is that it solves the problem of initializing the training algorithm with something reasonable. The matrix $\mathbf{M}$ can be trained from a zero start.

3.2. High-dimensional feature generation

The first stage of fMPE is to transform the features into a very high dimensional space. This is done as follows for most experiments described here. A set of Gaussians is created by likelihood-based clustering of the Gaussians in the acoustic model to an appropriate size (up to 100,000 in experiments reported here). On each frame, the Gaussian likelihoods are evaluated with no priors, and a vector of posteriors $\mathbf{h}_t$ is formed. This can be done very quickly (e.g. less than 0.1xRT) by further clustering the Gaussians to, say, 2000 cluster centers and only evaluating the 100 most likely clusters based on the cluster-center's likelihood [3]. A key feature is that the vector $\mathbf{h}_t$ is sparse, i.e. only certain elements on each time t differ significantly from zero; this greatly speeds up the computation.

3.3. Acoustic context expansion

The vector is further expanded with left and right acoustic context. The following is a typical configuration used: If the central (current) frame is at position 0, vectors are appended which are the average of the posterior vector at positions 1 and 2, at positions 3, 4 and 5, and at positions 6, 7, 8 and 9. The same is done to the left (positions -1 and -2, etc) so that the final vector is of size 700,000 if there were 100,000 Gaussians. Sparse vector routines are used for speed.

3.4. Training the matrix

The matrix is trained by linear methods, because in such high dimensions accumulating squared statistics would be impractical. The update on each iteration is:

$$M_{ij} := M_{ij} + \nu_{ij} \frac{\partial \mathcal{F}}{\partial M_{ij}}, \tag{3}$$

i.e. gradient descent where the parameter-specific learning rates are:

$$\nu_{ij} = \frac{\sigma_i}{E(p_{ij} + n_{ij})}, \tag{4}$$

where p_{ij} and n_{ij} (see below) are the sum over time of the positive and negative contributions towards $\frac{\partial \mathcal{F}}{\partial M_{ij}}$, E is a constant that controls the overall learning rate and σ_i is the average standard deviation of Gaussians in the current HMM set in that dimension. Since $\frac{\partial \mathcal{F}}{\partial M_{ij}} = p_{ij} - n_{ij}$, the most each M_{ij} can change is $1/E$ standard deviations, and the most any given feature element y_{ti} can change is n/E standard deviations, where n is the number of acoustic contexts by which the vector H_t has been expanded (e.g. n =7).

It follows from Equation 2 that

$$\frac{\partial \mathcal{F}}{\partial M_{ij}} = \sum_{t=1}^{T} \frac{\partial \mathcal{F}}{\partial y_{ti}} h_{tj}, \tag{5}$$

where h_{tj} is the j'th dimension of $\mathbf{h}_t$ and y_{ti} is the i'th dimension of the transformed feature vector $\mathbf{y}_t$. The differential $\frac{\partial \mathcal{F}}{\partial M_{ij}}$ is broken into the positive and negative parts needed to set the learning rate in Equation 4:

$$p_{ij} = \textstyle\sum_{t=1}^{T} \max(\frac{\partial \mathcal{F}}{\partial y_{ti}} h_{tj}, 0) \tag{6}$$

$$n_{ij} = \textstyle\sum_{t=1}^{T} \max(-\frac{\partial \mathcal{F}}{\partial y_{ti}} h_{tj}, 0). \tag{7}$$

3.5. Smoothing of update

To prevent over-training of parameters that cannot be estimated robustly, a modification is made as follows, which amounts to using a slower learning rate for the elements that have sparse training data. Let the "count" c_{ij} be $\sum_{t=1}^{T} h_{tj}$, which is similar to the number of nonzero points available in estimating the differential $\frac{\partial \mathcal{F}}{\partial M_{ij}}$. This formula only makes sense if the high dimensional features h_{tj} are generally either zero or not far from one; another way to set c_{ij} is $(\sum_{t=1}^{T} |d_{ij}(t)|)^2 / \sum_{t=1}^{T} d_{ij}(t)^2$ where $d_{ij}(t) = \frac{\partial \mathcal{F}}{\partial y_{ti}} h_{tj}$, which is the number of points that would have the same expected ratio of squared sum of absolute values to sum-of-squares if it were Gaussian distributed with zero mean. These approaches gives similar counts. The count c_{ij} is used to work out the typical magnitude of a nonzero differential which is $(p_{ij} + n_{ij})/c_{ij}$. This is used to "pad" the differentials p_{ij} and n_{ij} with a number τ of typical imaginary observations prior to update, so $n_{ij} := n_{ij} + 0.5\tau(p_{ij} + n_{ij})/c_{ij}$, and $p_{ij} := p_{ij} + 0.5\tau(p_{ij} + n_{ij})/c_{ij}$. This slows down the learning rate (Equation 4) for parameters that have too few observations. Smoothing may slightly improve results, on the order of 0.1% absolute; generally this is done with $\tau \simeq 100$.

Some experiments reported here pad the two statistics with imaginary counts that are not equal, but have the same ratio as the overall statistics for the relevant cluster of Gaussians. However this does not make any clear difference to the WER so it is not described further.

4. CALCULATING THE DIFFERENTIAL

4.1. Direct differential

As mentioned in Section 3.4, a key quantity in fMPE training is $\frac{\partial \mathcal{F}}{\partial y_{ti}}$ which is the differential of the MPE function w.r.t. the i'th dimension of the transformed feature vector on time t.

Directly differentiating the MPE objective function can be done via the following equation. Defining the log likelihood of Gaussian m of state s on time t as l_{smt},

$$\frac{\partial \mathcal{F}}{\partial y_{ti}}^{\text{direct}} = \textstyle\sum_{s=1}^{S} \sum_{m=1}^{M_s} \frac{\partial \mathcal{F}}{\partial l_{smt}} \frac{\partial l_{smt}}{\partial y_{ti}}. \tag{8}$$

The first factor $\frac{\partial \mathcal{F}}{\partial l_{smt}}$ is already calculated in normal MPE training [1, 2]; it equals $\sum_{q=1}^{Q} \kappa \gamma_q^{\text{MPE}} \gamma_{qsm}(t)$ where κ is the probability scale, $\kappa \gamma_q^{\text{MPE}}$ is the differential of $\mathcal{F}$ w.r.t. the log likelihood of the q'th phone arc, and $\gamma_{qsm}(t)$ is the Gaussian occupation probability within the phone arc. The second factor $\frac{\partial l_{smt}}{\partial y_{ti}}$ equals $\frac{\mu_{smi} - y_{ti}}{\sigma^2_{smi}}$. Note that the positive and negative γ_q^{MPE} (and the positive and negative l_{smt}) should sum to zero on each time t, and where for numerical or pruning reasons they did not they were re-balanced, for experiments reported here.

4.2. Indirect differential

Equation 8 is unsatisfactory because it takes no account of the fact that the same features are used to train as well as test the model, and the features will affect the HMM parameters. When using Equation 8 for the differential, it was found that much of the WER improvement was lost as soon as the same features were used to to retrain the models (with ML training). For this reason, the differential is augmented with a term that reflects changes in the models. The statistics used for normal MPE training are used to calculate $\frac{\partial \mathcal{F}}{\partial \mu_{smi}}$ and $\frac{\partial \mathcal{F}}{\partial \sigma^2_{smi}}$, i.e. the differential of the objective function w.r.t. the model means and variances (see Section 4.3). This allows us to calculate the part of the differential that is mediated by changes in the Gaussians:

$$\frac{\partial \mathcal{F}}{\partial y_{ti}}^{\text{indirect}} = \sum_{s=1}^{S} \sum_{m=1}^{M_s} \frac{\gamma_{sm}(t)}{\gamma_{sm}} \left(\frac{\partial \mathcal{F}}{\partial \mu_{smi}} + 2 \frac{\partial \mathcal{F}}{\partial \sigma^2_{smi}} (y_{ti} - \mu_{smi}) \right) \qquad (9)$$

where $\gamma_{sm}(t)$ is the ML occupation probability as used in standard forward-backward training; γ_{sm} is the same thing summed over all the training data. The final differential that is used is:

$$\frac{\partial \mathcal{F}}{\partial y_{ti}} = \frac{\partial \mathcal{F}}{\partial y_{ti}}^{\text{direct}} + \frac{\partial \mathcal{F}}{\partial y_{ti}}^{\text{indirect}}. \qquad (10)$$

Note that Equation 9 is based on assumptions that are not quite met. The fMPE differential of Equation 8 and the MPE differentials $\frac{\partial \mathcal{F}}{\partial \mu_{smi}}$ etc are the differentials around the current acoustic parameters and features. The current acoustic parameters λ were generated from statistics obtained by aligning previous models, say λ^{prev}. Ideally, Equation 9 should refer to these previously obtained occupation probabilities $\gamma_{sm}(t)^{\text{prev}}$ and $\gamma_{sm}^{\text{prev}}$. For convenience this is not done.

4.3. Model parameter differentials

In order to calculate the indirect differential, the quantities $\frac{\partial \mathcal{F}}{\partial \mu_{smi}}$ and $\frac{\partial \mathcal{F}}{\partial \sigma^2_{smi}}$ are are obtained from normal MPE statistics [1, 2] as follows:

$$\frac{\partial \mathcal{F}}{\partial \mu_{smi}} = \frac{\kappa}{\sigma^2_{smi}} \left(\theta^{\text{num}}_{smi}(\mathcal{O}) - \theta^{\text{den}}_{smi}(\mathcal{O}) - \mu_{smi}(\gamma^{\text{num}}_{sm} - \gamma^{\text{den}}_{sm}) \right), \qquad (11)$$

where μ_{smi} and σ^2_{smi} are the mean and variance in the Gaussians used for the alignment, and $\theta^{\text{num}}_{smi}(\mathcal{O})$ and $\gamma^{\text{num}}_{smi}$ etc are the sum-of-data and count MPE statistics.

For the variance, let us first define the quantities S^{num}_{smi} and S^{den}_{smi} which are the variance of the numerator and denominator statistics around the current mean, so e.g.

$$S^{\text{num}}_{smi} = (\theta^{\text{num}}_{smi}(\mathcal{O}^2) - 2\theta^{\text{num}}_{smi}(\mathcal{O})\mu_{smi} + \gamma^{\text{num}}_{sm}\mu^2_{smi})/\gamma^{\text{num}}_{sm}, \qquad (12)$$

where $\theta^{\text{num}}_{smi}(\mathcal{O}^2)$ are the sum-of-squared-data statistics. The differential w.r.t the variance is then

$$\frac{\partial \mathcal{F}}{\partial \sigma^2_{smi}} = \frac{\kappa \gamma^{\text{num}}_{sm}}{2} (S^{\text{num}}_{smi} \sigma^{-4}_{smi} - \sigma^{-2}_{smi}) - \frac{\kappa \gamma^{\text{den}}_{sm}}{2} (S^{\text{den}}_{smi} \sigma^{-4}_{smi} - \sigma^{-2}_{smi}). \qquad (13)$$

4.4. Checks

A useful check that no implementation errors have been made is that adding a small quantity to all the features in some dimension should not affect the MPE objective function, as long as it is done in both training and test. This implies that

$$\sum_{t=1}^{T} \frac{\partial \mathcal{F}}{\partial y_{ti}}^{\text{direct}} + \sum_{t=1}^{T} \frac{\partial \mathcal{F}}{\partial y_{ti}}^{\text{indirect}} = 0, \qquad (14)$$

where the summation $\sum_{t=1}^{T}$ is over all training data. The two terms in the above equation generally cancel out to within a margin of, say 1% of the absolute values of the two terms. Discrepancies are due to the assumptions made in Equation 9 not being met. A similar metric relating to a linear scaling of each dimension can be more sensitive to problems but should cancel to within a few percent:

$$\sum_{t=1}^{T} y_{ti} \frac{\partial \mathcal{F}}{\partial y_{ti}}^{\text{direct}} + \sum_{t=1}^{T} y_{ti} \frac{\partial \mathcal{F}}{\partial y_{ti}}^{\text{indirect}} = 0. \qquad (15)$$

5. OVERVIEW AND GENERAL CONSIDERATIONS IN FMPE TRAINING

5.1. Overview

Procedurally, each iteration of fMPE training involves three passes over the data: one to accumulate normal MPE statistics; a second to accumulate fMPE statistics (chiefly the quantities n_{ij} and p_{ij}), and a third pass to do an ML update with the newly transformed data. All three passes start with the same HMMs; for simplicity, in these experiments the third pass aligns with the newly transformed features rather than doing single-pass retraining from the old to the new features. Naturally, on the $n+1$'th iteration the updated HMMs from the n'th iteration will be used to align the data and the first two passes will use the transformed features from the n'th iteration's update. Convergence speed is similar to MPE, so three or four iterations may give most of the improvement. However, fMPE seems to be more robust to overtraining, i.e. the error rate does not tend to start rising after after a few iterations.

5.2. Dimension of high-dimensional features

Experiments on Callcenter data suggest that it is probably good to use as high a dimension as possible until there is insufficient data for each parameter and data-learning becomes an issue. This is why the very high dimension of

$100{,}000 \times 7$ contexts was used in CTS experiments reported here. The overhead in testing is very small - about 0.1 to 0.2xRT. Much of the improvement in WER can be obtained with a smaller dimension and no acoustic context. Early experiments used state posteriors rather than Gaussian posteriors; no clear evidence is available as to their relative usefulness but Gaussian posteriors are more convenient.

5.3. Offset features

A different kind of high-dimensional feature that has proved useful is the 'offset' feature. This is obtained by including in the feature vector the offset of the observed feature vector from each Gaussian's mean, scaled by its posterior. If there are N diagonal Gaussians over d-dimensional input features, the vector $\mathbf{h}_t$ will have dimension $N(d+1)$. Each Gaussian n has a posterior $0 \leq p_t(n) \leq 1$ on time t. The offset features will be, for each dimension $1 \leq d \leq D$, $p_t(n)\frac{x_t(d)-\mu_n(d)}{\sigma_n(d)}$, where $\mu_n(d)$ and $\sigma_n(d)$ are the mean and standard deviation of the n'th Gaussian (the normalization is to ensure that the offset features all have zero mean and approximately the same dynamic range). The $d+1'th$ feature for each Gaussian is the posterior itself scaled by a constant: $Sp_t(n)$, where the scaling factor S is 5.0 for experiments reported here. This is to prevent the posteriors being swamped by the much larger number of offsets; it has the effect of giving the posteriors a faster learning rate. With offset features, it is especially important for efficiency reasons to prune the Gaussian posteriors heavily to keep the number of nonzero features few in number. For experiments reported here, only the top two posteriors are kept (and renormalized to sum to one).

5.4. Typical criterion improvements

In fMPE, the improvement in MPE criterion (expressed relative to the number of phones in the correct transcription) tends to be smaller than in MPE training: around 2-3% absolute, e.g. rising from 0.70 to 0.725, compared with perhaps 6% in MPE training. However the observed WER improvements on test data are not much smaller than the criterion improvement (say, around 2%); also in fMPE training a greater proportion of the training data criterion improvement is seen when the MPE criterion is measured on unseen data, as compared with MPE training. Note that the MPE criterion is a kind of smoothed error rate so the comparison with WER makes sense.

The predicted MPE criterion improvement calculated from the gradient $\frac{\partial \mathcal{F}}{\partial M_{ij}}$ and the change in each M_{ij} tends on the first iteration to be around 0.05 to 0.10 after dividing by the number of phones in the reference transcript; this corresponds to around 6% to 12% predicted improvement in phone error rate. This predicted improvement may decrease by as much as half on the second iteration and will decrease further thereafter. The observed improvement in the MPE objective function tends to be in the region of half the predicted improvement.

5.5. Suggested learning rates

The value of the learning rate E is one of the more critical parameters. The optimal value of E will tend to increase as the number of contexts increases because the summed value of the elements of $\mathbf{h}_t$ is increasing. It is suggested to fix an amount of predicted MPE criterion improvement to aim for on the first iteration (say 0.06) and calculate the E value corresponding to this; and to use the same E value for subsequent iterations. One measure of whether the learning rate is too fast or too slow is the number of matrix parameters $\mathbf{M}_{ij}$ that change sign. For the best values of E (in terms of WER on test data), the proportion of the parameters $\mathbf{M}_{ij}$ that changes sign seems to be around 10-15% on the second iteration, decreasing to around 5-10% on subsequent iterations.

5.6. Testing for data learning

This section demonstrates a convenient way to test how much of the improvement in the MPE criterion derives from data learning and how much should generalize to other data. Note that this is for interest only and is not necessary in order to implement fMPE.

The update procedure improves the MPE criterion on the training data; only some of this improvement in criterion would generalize to unseen data. Determining how much of the improvement is due to data learning is easy given the nature of the update equations. If the basic update described above is used (no τ involved), the learning rule for each parameter is gradient descent for each parameter with the learning rate more or less inversely proportional to the amount of training data available.

If the training data is imagined divided up into blocks of a certain size, and each block contributes a differential w.r.t. each parameter that is some common value plus a random block-specific term that is drawn from a zero-mean, i.i.d. distribution, and if a subset of the differentials from these blocks are used to train, it can be shown fairly easily that the expected criterion improvement due to data-learning is constant independent of the number of blocks used, but the "good" improvement rises linearly.

Let us suppose that the differential from each block $b = 1 \ldots B$ equals a constant amount k (the expected differential for that block size) plus a random amount r_b. If the learning rate is l/B, the improvement in criterion is $(l/B)(\sum_{b=1\ldots B} k + r_b)^2$. r_b is drawn from an i.i.d. distribution with mean zero, and it works out that the expected improvement in criterion is $l(v + Bk^2)$, where v is the variance of the distribution from which the r_b are drawn. So the expected criterion improvement lv due to overtraining is independent of the size of the training dataset, whereas the real improvement lBk^2 rises linearly.

This leads to a rule exemplified as follows: If the update is done with 0.2 the amount of data and the improvement in criterion is 0.37 what it was before, it works out that the proportion of the original improvement that was due to overtraining is $\frac{0.37-0.2}{1-0.2} = 0.21$. The predicted proportion of improvement due to overtraining can be as high as 0.5.

Note that these calculations refer to the absolute MPE criterion, not normalized by the number of correct phones.

5.7. Typical learning rates, and acoustic scaling

When early fMPE experiments were performed (including those for systems submitted for the RT-04 evaluation), it was believed that there was a danger that the fMPE transform might attempt to generally strengthen or weaken the acoustic model relative to the LM. In order to prevent this from happening, the differential of the MPE criterion w.r.t a scaling of all the acoustic likelihoods was calculated and the acoustic and LM weights were tuned until this was close to zero. This explains why there is a lot of variation in the acoustic and language model weights used (Table 6.3). Further experiments have shown that this consideration is not important and simply using the same acoustic weight as MPE (e.g. $\kappa = 0.1$) and the same language model weight can give better results. A more complete investigaton of the effect of the acoustic weight on fMPE has not been attempted.

6. EXPERIMENTAL CONDITIONS

6.1. Datasets

Experiments are reported here on four datasets: CTS (conversational telephone speech), Broadcast News, Callcenter and Malach. For CTS, the training data was 2300 hours of Switchboard, Call Home English, and (mostly) Fisher data. The amount of data remaining after segmentation, and removing segments containing only one word or OOV's or which have alignment problems, was around 2100 hours. The Broadcast News training data was the 140 hours of fully transcribed data from 1996 and 1997; the length was 133 hours after segmentation and cleaning. Callcenter data was data recorded from an internal IBM technical call center; the amount was 300h of data (raw), or 200 (segmented and cleaned). The Malach data consists of transcriptions of testimonies of Holocaust survivors, collected by the Shoah Foundation. This data mostly consists of heavily accented speech by elderly speakers, leading to high error rates. The length of the training data was 300h (raw), and only 95h after cleaning because of problems with the transcripts.

Results on the CTS system are reported here on the RT-03 test set. Broadcast News results are reported on the eval97 test set (3 hours). Callcenter results are reported on 6 hours of test data from the same source as the training data. Malach results are reported on a test set consisting of 1 hour of interviews (this is referred to elsewhere as "test set 2").

	Features		
CTS(SI)	PLP+LDA+MLLT, 40dim		
CTS(SD)	PLP+LDA+MLLT, 39dim		
Callcenter(SI)	PLP+LDA+MLLT, 40dim		
Malach(SI)	MFCC+LDA+MLLT, 60dim		
Malach(SD)	MFCC+LDA+MLLT, 60dim		
Broadcast News	MFCC+LDA+MLLT, 60dim		
	Adaptation		
CTS(SI)	CMS+CVN (per-side)		
CTS(SD)	...+VTLN+fMLLR		
Callcenter(SI)	CMS (per-conversation)		
Malach(SI)	CMS (per-segment)		
Malach(SD)	...+VTLN+fMLLR		
Broadcast News	CMS (per-cluster)		
	#States	#Gauss	Phn context
CTS(SI)	8K	150K	5-phone xwrd
CTS(SD)	22K	850K	7-phone xwrd
Callcenter(SI)	4K	97K	11-phn left-xwrd
Malach(SI)	4K	89K	3-phone xwrd
Malach(SD)	4K	80K	3-phone xwrd
Broadcast News	8K	128K	3-phone xwrd

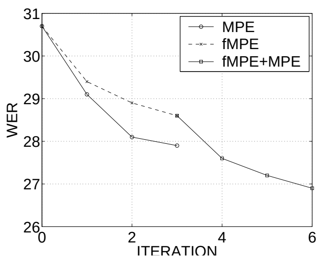
Table 1. System setups: features, adaptation etc.

6.2. Baseline system setups

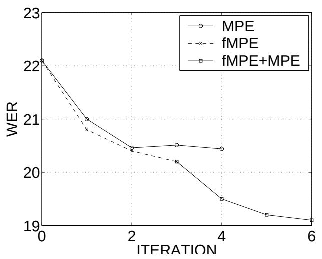
Table 6.2 gives details of the system setups for the various systems on which results are reported. Unless otherwise stated, MPE was done with an acoustic weight of 0.1, an LM weight of 1.9 (i.e 19 times the acoustic weight) and E=2.0. The setup for MPE is largely as described in [1]; however a fourth set of statistics (corresponding to the denominator statistics in MMI training) is also accumulated so that I-smoothing can back off to an MMI rather than an ML estimate. Unless otherwise stated, lattices are generated with either a unigram or highly pruned bigram language model. In some experiments, the statistics for MPE training are averaged over several acoustic and LM scales close to the baseline values of 0.1 and 1.9, e.g. in adapted experiments on CTS MPE experiments used scales of 0.10 and 0.16 (acoustic), and 1.0 and 1.6 (LM); four combinations. After each iteration of MPE udpate, variances are floored to the 20th percentile of the cumulative distribution of variances in each dimension [2].

	Acwt	LMwt	#Gauss	#cxts	E
CTS(SI)	0.15	1.25	100K	5	0.96
CTS(SD)	0.1	1.25	64K	7	1.44
Callctr	0.175	1.2	32K	7,9	1.44,1.66
Mal(SI)	0.1	1.9	32K	9	1.66
Mal(SD)	0.1	1.9	32K	9	2.0
BN(SI)	0.1	1.0	750	9	n/a

Table 2. fMPE training setups

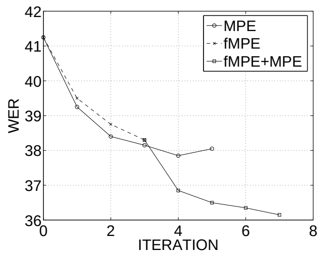


(a) CTS SI

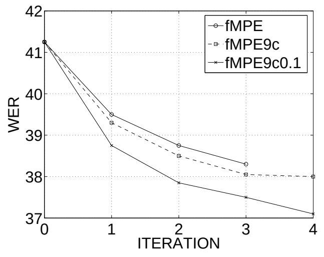


(b) CTS Adapted

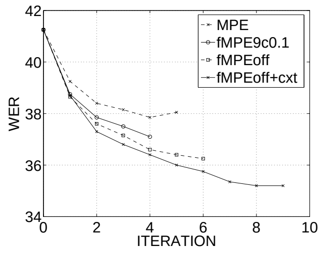
Fig. 1. MPE and fMPE results on CTS



(a) Basic Experiments



(b) Further tuning



(c) Testing offset features

Fig. 2. MPE and fMPE results on Callcenter data

6.3. fMPE system setups

Table 6.3 shows the settings of various fMPE-related parameters for the various systems trained. These are only for the 'primary' fMPE experiments; contrasting results with different settings are also presented. The '#Gauss' column is the number of Gaussians evaluated to obtain the high dimensional features, and not the number of Gaussians in the models. The low number of Gaussians for the BN system reflects the fact that only 'offset' features were used (see Sections 5.3 and 9).

The differing fMPE setups do not result from tuning to different conditions but reflect the fact that what was considered the "best" setup was in a state of flux.

7. CONVERSATIONAL TELEPHONE SPEECH (CTS) EXPERIMENTS

In Figure 1(a) and (b), results for MPE training and fMPE followed by MPE are shown for CTS in both SI and adapted conditions. These experiments were done in preparation for IBM's submission to the NIST RT-04 (Rich Transcription 2004) evaluation [4]. Testing is on RT-03. The poorer results on the SI task may be explained by the fact that in the SI setup, only 1/5 of the training data was used to train the fMPE transform; and because of poorer settings of the acoustic weight and number of contexts. For the adapted system, the final fMPE+MPE number, at 19.1%, is better by 1.3% than MPE alone; the corresponding improvement is 1.0% for the SI system.

For the RT-04 evaluation, a system with 0.4% better WER than the final adapted fMPE+MPE number was obtained. To do this, the fMPE features were used to train from scratch a small 5-phone context system. Then, a second layer of fMPE transform ("iterated fMPE") was trained on the small system using 1/4 the data, with 25K Gaussians $\times$ 7 contexts. This doubly transformed data was used to further train the original 7-phone context fMPE models (20.2% $\rightarrow$ 19.4%), after which MPE training was done ($\rightarrow$ 18.7%). This is 1.7% better than the best models with MPE alone. The final transcriptions submitted included other features such as cross-adaptation, MLLR, LM rescoring and consensus. The 10xRT system had 13.0% WER on Dev-04, and 16.1% on RT-03 with 12.4% on the Fisher portion only.

8. CALLCENTER EXPERIMENTS

Figure 2(a) shows fMPE and MPE experiments on data recorded from an IBM computer support call center. No adaptation is used. The basic fMPE+MPE results on callcenter data are an impressive 5.2% better than the ML baseline and 1.8% better than MPE alone. Figure 2(b) shows further tuning of the fMPE setup. fMPE9c is with nine rather than seven acoustic contexts, which are more compact than the original seven contexts and are the averages of sets of frames (0), (1), (2,3), (4,5), (6,7,8), (-1) etc. (The original seven contexts started as (0), (1,2), (3,4,5) etc.). fMPE9c0.1 changes the acoustic and LM weight from 0.175 and 1.2, to be the same as the normal MPE acoustic and LM weights, i.e. 0.1 and 1.9. The lowered acoustic weight probably leads to more robust statistics and better generalization to unseen data.

Figure 2(c) shows some further improved results with so-called 'offset' features, as described above in Section 5.3. From top to bottom, the lines are: baseline MPE, tuned 'vanilla' fMPE (fMPE9c0.1), 'offset' fMPE (fMPEoff), and 'offset' fMPE with training of the context expansion (fMPE-off+cxt). The 'offset' fMPE features (Section 5.3) also use nine contexts, and use E=9 and only 1024 Gaussians; the number of parameters is roughly the same as before because each Gaussian now leads to 41 times as many parameters in the matrix.

The bottom plot (fMPEoff+cxt) shows results obtained with offset features but when the context expansion is also trained. This gives more than 6% absolute improvement over the ML baseline. Briefly, the setup is as follows. It requires a rearrangement of the calculation for efficiency, and two layers of projection from high dimensional features down to dimension d, after which the original features are added. First, the unexpanded posteriors are multiplied by a matrix to be trained that has $9d$ columns (corresponding to the 9 contexts). Thus, for the current system with basic features of size $1024 \times 41 \simeq 42K$, $\mathbf{M}$ is of size $42K \times 9d$ rather than $(9 \times 42K) \times d$ as it would be with the previous method of context expansion. The feature vector of size $9d$ is then spliced together across $\pm$ 40 frames and collapsed down to size d by a projection which is initialized to have the same effect as the normal context expansion described previously. This 'collapsing' projection is constrained so that a particular output dimension (i.e. from $1 \ldots d$) is a linear combination of the $(2{\times}40 + 1) \times 9$ contexts and frame-offsets of only that same dimension. It thus has $9d(2{\times}40 + 1)$ parameters. Both matrices are trained by gradient descent with parameter-specific learning rates set as before, except that the collapsing layer does not require the variance compensation factor σ_i in the learning speed. The gradient descent procedure is used with a modification to avoid instabilities as follows: if more than 10% of the matrix parameters within a set of meaningfully related matrix parameters (e.g. rows, columns, etc.) change sign, the learning rate for that set is reduced until the number changing sign equals 10%. This is done from the second iteration, being irrelevant on the first. For the third iteration and onwards, the percentage allowed to change sign within any group is 5%. The E on the first iteration is set so as to give a predetermined amount of criterion improvement (e.g. 0.06 for the main matrix, 0.0075 for the collapsing projection), and the same E used to initialize the learning rates on subsequent iterations. Note that the first iteration of training for the collapsing layer is the second iteration of fMPE since its gradient on the first iteration of fMPE will be zero.

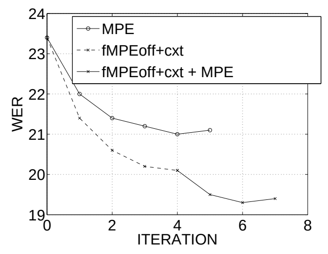
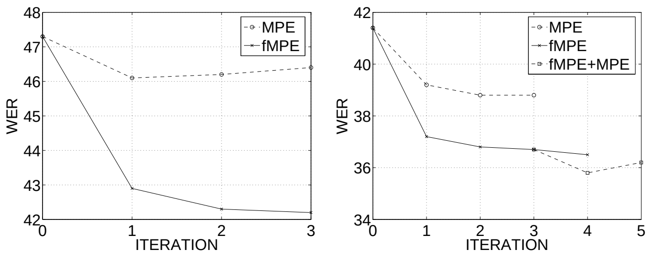
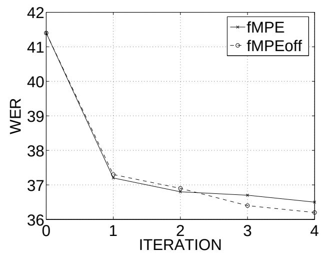


Fig. 3. MPE and fMPE results on Broadcast News (eval97).



(a) Speaker Independent (b) Speaker Adapted



(c) Adapted, offset features

Fig. 4. MPE and fMPE results on Malach data

9. BROADCAST NEWS EXPERIMENTS

The best setup mentioned above for Callcenter experiments in Figure 2(c), i.e. offset features and context expansion, has also been tried on a Broadcast News training setup (Figure 3). fMPE+MPE gives about 1.6% improvement over MPE alone. The setup will not be described in detail, but note that the MPE baseline has 0.5% advantage over fMPE because it combines statistics from two sets of lattices derived from decoding with word-based and phone-based language models, which the fMPE experiments do not.

10. MALACH EXPERIMENTS

The experiments reported in Figure 4 confirm the observation from Callcenter data that fMPE, if trained with the right parameters, can give better results than MPE alone. In this case, the difference is very dramatic. Speaker independent experiments (Figure 4(a)) show 3.2% improvement over MPE, and an enormous 5% absolute improvement over the ML baseline. Figure 4(b) gives speaker adapted numbers with a very impressive 3.0% improvement over MPE and 5.6% over the ML number. Figure 4(c) gives results on the speaker adapted features, duplicating the fMPE experiment with 'offset features' shown in Figure 2(c). Note that this is not the very best setup shown in Figure 2(c) which trained the context expansion. The improvement over the normal fMPE features is only 0.3%, but this is still useful because 'offset features' are much more efficient to test with (fewer Gaussians are evaluated).

11. CONCLUSION

fMPE is a novel and effective way to apply discriminative training to features rather than models. It made a significant contribution to IBM's submission to the RT-04 evaluation, and experiments on three other corpora demonstrate the robust nature of the improvements from fMPE and show that further improvements can be obtained by tuning the training setup.

fMPE is an extremely flexible framework, as shown for instance by the success of 'offset' features. This and other improvements to the fMPE setup came too late to make it into IBM's RT-04 submission but would doubtless have improved results. The feature-based nature of fMPE makes possible things that are not possible with normal discriminative training, such as building a system on the new features and iterating the process of fMPE training (as was done for the evaluation system); or applying projections both before and after adaptation.

12. REFERENCES

[1] D. Povey and P.C. Woodland, "Minimum Phone Error and I-smoothing for Improved Discriminative Training," in *ICASSP*, 2002.

[2] D. Povey, *Discriminative Training for Large Vocabulary Speech Recognition*, Ph.D. thesis, Cambridge University, 2004.

[3] G. Saon, G. Zweig, B. Kingsbury, L. Mangu, and U. Chaudhari, "An Architecture for Rapid Decoding of Large Vocabulary Conversational Speech," in *Eurospeech*, 2002.

[4] H. Soltau, B. Kingsbury, L. Mangu, D. Povey, G. Saon, and G. Zweig, "The IBM 2004 Conversational Telephony System for Rich Transcription in EARS," in *ICASSP*, 2005.