QUICK FMLLR FOR SPEAKER ADAPTATION IN SPEECH RECOGNITION[1]

Balakrishnan Varadarajan[2]

Johns Hopkins University
Baltimore, Maryland, USA
bvarada2@jhu.edu

Daniel Povey and Stephen M. Chu

IBM T. J. Watson Research Center
Yorktown Heights, New York, USA
{dpovey, schu}@us.ibm.com

ABSTRACT

Feature space *Maximum Likelihood Linear Regression* (fMLLR) is a widely used technique for speaker adaptation in HMM-based speech recognition. However, in extremely resource constrained systems the time required to perform the sufficient statistics accumulation for fMLLR adaptation can be considerable. In this paper we describe a novel method that can lead to significant reduction in the time taken for statistics accumulation while preserving the adaptation gains. The proposed *Quick* fMLLR (Q-fMLLR) algorithm is implemented in a state-of-the-art large-vocabulary continuous speech recognition system, and evaluated on a broadcast transcription task. We present results both in terms of the average likelihood after adaptation and the character error rate. It is shown that Q-fMLLR attains the performance of regular fMLLR with a fraction of the computation.

Index Terms **–** MLLR, feature space MLLR, constrained MLLR, fMLLR, speaker adaptation.

1. INTRODUCTION

Maximum likelihood linear regression (MLLR) and feature space MLLR (fMLLR), also known as constrained MLLR, are commonly used speaker adaptation techniques [1][6]. However, in extremely resource constrained systems the time required to accumulate the statistics for fMLLR, which is O(d^3) per frame where d is the feature dimension, can be significant. In this paper, we describe a method that can reduce this time to O(d^2).

In fMLLR, the auxiliary function we optimize has three parts: a linear term, a quadratic term, and a log determinant. Most of the time taken to compute the fMLLR transformation comes from computing statistics for the quadratic term. The fast algorithm proposed in this work reduces the computation by storing an approximation to the quadratic term, and uses this to optimize an auxiliary function that has the same local gradient as the objective function but an approximated Hessian. A similar concept was used in [2] in the context of extending fMLLR to the case of full covariance Gaussians.

The proposed technique is based on projecting the (diagonal) precision matrix into a subspace for the purpose of accumulating the quadratic statistics. In this paper we describe two variants of our technique. The first is not guaranteed to increase the objective function on any given iteration but will approach a local maximum after many iterations. The second variant of the method is guaranteed to improve the objective function on every iteration, but at the price of slower convergence.

The remainder of this paper is organized as follows. In Section 2, we review the baseline fMLLR algorithm and the row-by-row computation. The proposed quick fMLLR algorithm is described in Section 3. Section 4 presents the experimental results, followed by conclusions in Section 5.

2. BASELINE FMLLR

2.1. Basic Formulation

In the classic formulation of fMLLR, the adapted features for a given speaker are computed through the affine transform

$$\hat{\mathbf{x}}^{(t)} = \mathbf{W}\boldsymbol{\xi}^{(t)}, \tag{1}$$

where $\boldsymbol{\xi} = [1 \quad \mathbf{x}^T]^T$ is the input vector extended with an extra element equal to unity, and $\mathbf{W}$ is a d by $d+1$ matrix, which can be viewed as consisting of a square matrix $\mathbf{A}$ and a bias term $\mathbf{b}$, $\mathbf{W} = [\mathbf{b} \quad \mathbf{A}]$.

The objective function consists of the log likelihood of the transformed data given our models, plus the log determinant $\log|\det \mathbf{A}|$. The log likelihood of the transformed data given the models is a complicated function to optimize, but through E-M, we get an auxiliary function which is simply a sum of quadratic functions of the rows of $\mathbf{W}$. The auxiliary function is as follows:

$$F(\mathbf{W}) = \log|\det \mathbf{A}| + \sum_{i=1}^{d} \mathbf{w}_i^T \mathbf{k}_i - 0.5\mathbf{w}_i^T \mathbf{G}_i \mathbf{w}_i, \tag{2}$$

where $\mathbf{w}_i$ are the transposed rows of $\mathbf{W}$. The $d+1$ dimensional vectors $\mathbf{k}_i$ and the $d+1$ by $d+1$ matrices $\mathbf{G}_i$ (for $1 \le i \le d$) are the sufficient statistics which are accumulated as follows:

$$\mathbf{k}_i = \sum_{t=1}^{T}\sum_{m=1}^{M} \frac{1}{\sigma_i^{2(m)}} \gamma_m^{(t)} \mu_i^{(m)} \boldsymbol{\xi}^{(t)} \tag{3}$$

$$\mathbf{G}_i = \sum_{t=1}^{T}\sum_{m=1}^{M} \frac{1}{\sigma_i^{2(m)}} \gamma_m^{(t)} \boldsymbol{\xi}^{(t)} \boldsymbol{\xi}^{(t)T} \tag{4}$$

where $\gamma_m^{(t)}$ are the Gaussian occupation probabilities, and T is the number of frames in the data.

[1] This work was funded in part by DARPA contract HR0011-06-2-0001.

[2] The author was at IBM T. J. Watson Research Center during this work.

2.1. Row-By-Row Iterative fMLLR Computation

The matrix **W** can be estimated through maximization of the auxiliary function in equation (2) using an iterative update [1]. It exploits the fact that the determinant of a matrix equals the dot product of any given row of the matrix with the corresponding row of cofactors. To update the i^{th} row of the transform, we let the column vector $\mathbf{c}_i$ equal the transpose of the i^{th} row of the cofactors of $\mathbf{A}$, appended to a zero in the first dimension to make a vector of size $d+1$, so that the determinant $|\det \mathbf{A}|$ can be represented as a function of $\mathbf{w}_i$ by $\mathbf{w}_i^T \mathbf{c}_i$. Thus we can optimize the function,

$$\beta \log(|\mathbf{w}_i^T \mathbf{c}_i|) + \mathbf{w}_i^T \mathbf{k}_i - 0.5 \mathbf{w}_i^T \mathbf{G}_i \mathbf{w}_i, \quad (5)$$

where β=T is the number of frames. The matrix of cofactors of $\mathbf{A}$ equals $\det(\mathbf{A})(\mathbf{A}^{-1})^T$, and the value of $\mathbf{w}_i$ that maximizes the expression in equation (5) is not affected by any scalar factor in $\mathbf{c}_i$. We can more easily let $\mathbf{c}_i$ equal the i^{th} column of the current value of $\mathbf{A}^{-1}$ (appended to a zero to make a $d+1$ dimensional column vector) and thus avoid any numerical problems that could arise when the determinant is very large or small. Let $f = \mathbf{w}_i^T \mathbf{c}_i$ and differentiate (5) w.r.t. $\mathbf{w}_i$, we see that the maximum occurs at $\mathbf{w}_i = \mathbf{G}_i^{-1}((\beta / f)\mathbf{c}_i + \mathbf{k}_i)$. Substituting this expression for $\mathbf{w}_i$ into the definition of f and multiplying by f, we get,

$$f^2 - f \mathbf{c}_i^T \mathbf{G}_i^{-1} \mathbf{k}_i - \beta \mathbf{c}_i^T \mathbf{G}_i^{-1} \mathbf{c}_i = 0, \quad (6)$$

in which f can be readily solved for using the quadratic formula. (Note that in [2], the factor β is mistakenly omitted).

The value of the auxiliary function in (5) can be used to test which solution to the quadratic equation is better. Alternatively, one can just take the positive square root for more convenience, essentially constraining **A** to have all positive eigenvalues, which is almost always taken anyway. The procedure is iterative. Starting from the baseline transform where $\mathbf{A} = \mathbf{I}, \mathbf{b} = 0$, we apply the update to each row in turn and continue iterating until the change in the auxiliary function is smaller than a threshold, or for a fixed number of iterations.

The entire update described here is also part of an E-M procedure, and if desired, can be applied for multiple iterations. This is done by computing the Gaussian occupancies $\gamma_m^{(t)}$ using the current transformed features $\hat{\mathbf{x}}^{(t)}$ and repeating all the steps described above. Note that equations (3) and (4) still refer to the original features rather than the transformed features.

3. Q-FMLLR

The proposed *quick*-fMLLR (Q-fMLLR) technique targets the large computational load associated with accumulating the matrices $\mathbf{G}_i$. The basic idea is that if the diagonal precisions (inverse variances) were to lie in a smaller dimensional subspace than the feature dimension, fewer statistics could be stored because $\mathbf{G}_i$ would be linearly dependent. Therefore, we find the least significant dimensions in which the precisions vary and reject them, and store a smaller number of matrices $\hat{\mathbf{G}}_i$ from which we can approximately reconstruct the matrices $\mathbf{G}_i$. In addition, we need to be able to find the *exact* local gradient of the objective function so that we will be optimizing a weak-sense auxiliary function for our objective function [3]. So, instead of storing $\mathbf{k}_i$, we store a set of different statistics, $\mathbf{d}_i$, which represent the local gradient of the objective function, and later use these to reconstruct $\mathbf{k}_i$.

3.1. Subspace Representation of Precisions

Rationale

In Q-fMLLR we intend to reject the least important dimensions in which the precisions vary. A reasonable method of doing this is to ask how we can get the least change in our models, as measured by the K-L divergence of the changed model from the original model, from rejecting a particular subspace of precisions. For simplicity and efficiency, we intend to simply remove any amount of the precision in the rejected subspace rather than find the optimal value in the accepted subspace.

For an individual Gaussian m in a Gaussian mixture, let's write $p_i^{(m)} = 1/\sigma_i^{2(m)}$ and $\mathbf{p}_m = [p_1^{(m)} \cdots p_d^{(m)}]^T$. If Gaussian m has probability w_m in the mixture, the increase in the K-L divergence from changing p_i^m by a small value δ will be $0.5 w_m \delta^2 / p_i^{(m)2}$. If we first pre-scale the precisions so that they are on average approximately equal to 1, the loss will be around $0.5 w_m \delta^2$, or just the weight times half the squared distance, which means that it is appropriate to do PCA on the pre-scaled precision vectors p_m. We introduce a weight in the PCA to both take account of the Gaussian mixture weights and scale the precision elements of each Gaussian to make them average to unity.

Method

The objective is to project p_m from a d-dimensional space to an n-dimensional subspace with an n by d matrix **P**, and project back to get the approximated precisions with a d by n matrix **Q**. To obtain **P** and **Q**, we first compute the average variance s_i in each dimension, and then compute the variance-normalized scatter matrix **S** with

$$S_{ij} = \sum_{m=1}^{M} v_m s_i p_i^{(m)} s_j p_j^{(m)} \Big/ \sum_{m=1}^{M} v_m, \quad (7)$$

where v_m are *modified* weights,

$$v_m = w_m \left(d \Big/ \sum_{i=1}^{d} s_j p_j^{(m)} \right)^2. \quad (8)$$

For a HMM, we just use the mixture weights within the individual states and assume all states are equally likely.

The projections $\hat{\mathbf{P}}$ and $\hat{\mathbf{Q}}$ within the space where the precisions are normalized by s_i are computed as follows: let the first row of $\hat{\mathbf{P}}$ be the principal eigenvector of **S**, the second row be the eigenvector with the next largest eigenvalue, and so on. $\hat{\mathbf{Q}}$ is the transpose of $\hat{\mathbf{P}}$. And finally, **P** and **Q** are computed as

$$\mathbf{P}_{ij} = \hat{\mathbf{P}}_{ij} s_j, \mathbf{Q}_{ij} = \hat{\mathbf{Q}}_{ij} / s_i. \quad (9)$$

First row positivity

It is important later for our "safe" version of the algorithm to make sure that the first row of **P** and the first column of **Q** have all positive elements. If they both have all negative elements we reverse

the sign. There is no other case that can occur because the first eigenvector of $\mathbf{S}$ must have all positive or all negative elements, which we show as follows. The principal eigenvector $\mathbf{x}$ of $\mathbf{S}$ is the unit vector such that

$$\mathbf{x}^T\mathbf{S}\mathbf{x} = \sum_{m=1}^{M} w_m \left(\sum_i x_i s_i p_i^{(m)} \right)^2 \Big/ \sum_{m=1}^{M} w_m \tag{10}$$

is largest. If $\mathbf{x}$ had elements of different signs, it is easy to see that we could make this expression larger by changing its elements to be all positive (or negative), which would not affect the length of $\mathbf{x}$. Note that this depends on the fact that $s_i p_i^{(m)}$ and w_m are all positive.

3.2. Statistics Accumulation

In Q-fMLLR, instead of computing statistics $\mathbf{k}_i$ and $\mathbf{G}_i$ as in Equations 3 and 4, the statistics are computed as follows. First, $\mathbf{k}_i$ is replaced with $\mathbf{d}_i$, which has the same dimensions as $\mathbf{k}_i$ but represents the gradient of the objective function around the current point rather than the linear term of its quadratic part. Second, the d matrices $\mathbf{G}_i$ are replaced with the n matrices $\hat{\mathbf{G}}_i$, with $n \le d$. Note that when $n = d$, this is equivalent to normal fMLLR.

$$\mathbf{d}_i = \sum_{t=1}^{T} \sum_{m=1}^{M} \frac{1}{\sigma_i^{2(m)}} \gamma_m^{(t)} (\mu_i^{(m)} - \hat{x}_i^{(t)}) \xi^{(t)} \tag{11}$$

$$\hat{\mathbf{G}}_i = \sum_{t=1}^{T} \sum_{m=1}^{M} \left(\sum_{j=1}^{d} \frac{P_{ij}}{\sigma_j^{2(m)}} \right) \gamma_m^{(t)} \xi^{(t)} \xi^{(t)T} \tag{12}$$

Note that $\hat{x}_i^{(t)}$ refers to the current transformed feature, which on the first iteration of EM would be the same as the untransformed feature $x_i^{(t)}$ but would change if more than one iteration of EM is performed. As with normal fMLLR, the Gaussian occupation probabilities $\gamma_m^{(t)}$ are also dependent on the previous iteration's transformed features in the multiple iteration case.

We also investigated a *safe* version of Q-fMLLR that is guaranteed to increase the objective function on each iteration, although as we will see, it converges much more slowly. In this version we make sure that the elements of the reconstructed precision $\mathbf{QPp}_m$ are always greater than or equal to the real precision by adding some minimal amount of the first element in the compressed vector space. (Recall that the first row of $\mathbf{P}$ and the first column of $\mathbf{Q}$ are always positive.) Thus, the expression $\sum P_{ij} / \sigma_j^{2(m)}$ in equation (12) is modified for the case when $i = 0$, to add this minimal amount.

3.3. Update

In the update phase of the Q-fMLLR, we first reconstruct the approximated $\mathbf{G}$ and $\mathbf{k}$ statistics as used in normal fMLLR:

$$\mathbf{G}_i = \sum_j \mathbf{Q}_{ij} \hat{\mathbf{G}}_j \tag{13}$$

$$\mathbf{k}_i = \mathbf{d}_i + \mathbf{G}_i \mathbf{w}_i \tag{14}$$

where $\mathbf{w}_i$ is the transpose of the i^{th} row of the fMLLR transformation matrix that was used while accumulating statistics (this would just be the default matrix where $\mathbf{A} = \mathbf{I}, \mathbf{b} = 0$ on the first iteration). The rest of the computation is the same as normal fMLLR. Note that it is possible, although in practice very unlikely, for the matrices $\mathbf{G}_i$ to not be positive definite, which can cause an attempt to take the square root of a negative number while solving the quadratic equation in the row-by-row update. This can only occur if $1 < n < d$. In our implementation, the default matrix is returned when this condition is detected. Theoretically this condition could also lead to exceedingly large determinants of $\mathbf{A}$, which nonetheless can also be detected and rejected. In fact, our baseline fMLLR computation already rejects determinants with absolute values outside the range 0.1 and 1000 so no additional checks are added.

3.4. Multi-Iteration Q-fMLLR

As discussed earlier, the fMLLR computation is part of an E-M process and multiple iterations can help (as we shall see in the results section). Multiple iterations are even more important with the Q-fMLLR computation because each iteration only partially optimizes the full auxiliary function that we would be using in fMLLR. There is however a risk of instability because the method uses an estimate of the second gradient of the objective function. Close to the maximum, if the estimated second gradient in any particular direction is too small by more than a factor of two then instability will arise. This problem can be addressed as follows.

For each speaker, a factor f is introduced to scale the reconstructed second gradient matrices $\mathbf{G}_i$, thus equation (13) becomes

$$\mathbf{G}_i = f \sum_j \mathbf{Q}_{ji} \hat{\mathbf{G}}_j \,. \tag{15}$$

On the first iteration of computation, the factor is initialized to $f = 1.0$. If the objective function, which is simply the likelihood of the transformed data given the HMM plus the determinant of the transform times the number of frames, is observed to decrease on a subsequent iteration, then f is doubled and the iterative process continues. Though less than optimal, e.g., it might be better to backtrack to the previous iteration and apply the larger f at that point, this method is straightforward and effective. All multi-iteration Q-fMLLR results presented here use this technique. To give the reader an idea of how often it is necessary to change f, in the experiment shown below with $d = 1$ and with six iterations of update, f was increased to 2.0 in 57 of the 245 speakers at the end of the computation, and in only one case did it reach 4.0.

4. EXPERIMENTS

4.1. Experimental Setup

The Q-fMLLR algorithm is implemented in a variant of the IBM Mandarin broadcast transcription system [4].

The system uses continuous mixture density HMMs with context-dependent states conditioned on cross-word quinphone context. Two sets of acoustic models are built with *maximum likelihood* training on 1,321 hours of broadcast news and broadcast conversation speech released by LDC for the DARPA GALE pro-

gram: (1) a speaker independent (SI) model with 10K quinphone states and 300K Gaussian densities, and (2) a speaker adaptive (SA) model with 15K states and 500K Gaussians. The input audio is sampled at 16 KHz and coded using 13-dememsional PLP features with a 25ms window and 10ms frame-shift; nine consecutive frames are spliced and projected to 40 dimensions using LDA and *maximum likelihood linear transform* (MLLT). *Vocal tract length normalization* (VTLN) is applied in SA training.

The language model is built by interpolating 20 back-off 4-gram models using modified Kneser-Ney smoothing. The interpolation weights are chosen to optimize the perplexity of a 364K held-out set. In total, 5GB of text data is used in training. The final language model has 6.1M n-grams and a vocabulary of 107K words.

During decoding, an initial transcription of the input data is first generated using the SI model. The output is then used to carry out unsupervised Q-fMLLR, followed by a second-pass decoding using the SA model.

4.2. Experimental Results

Adaptation and decoding experiments were carried out on the *dev*'07 test set defined by the GALE consortium. The set is composed of 2 hours and 32 minutes of Mandarin broadcast speech collected from various TV stations in mainland China, Taiwan, and Hong Kong. There are 44.6K characters in the reference.

The recognition results measured by character error rate (CER) are summarized in Table 1. The result using the SA system but no fMLLR transformation is 18.6%. The baseline results using various iterations of full fMLLR computation are given in the last row of the table, i.e. for n=40. As we can see, even down to n=1 to the CER is essentially unchanged using the standard Q-fMLLR method versus using the full fMLLR computation. However the *safe* Q-fMLLR method degrades quickly as the dimensions are reduced. Because it is hard to see any changes in the CER, we show the objective function (likelihood) in Fig. 1. On the first iteration, setting n=1 vs. n=40 reduces the likelihood by 0.26, which is 6.7% of the total objective function improvement. By the sixth iteration, the n=1 approximation reduces the likelihood by 0.05 which is only 1.3% of the total change in likelihood.

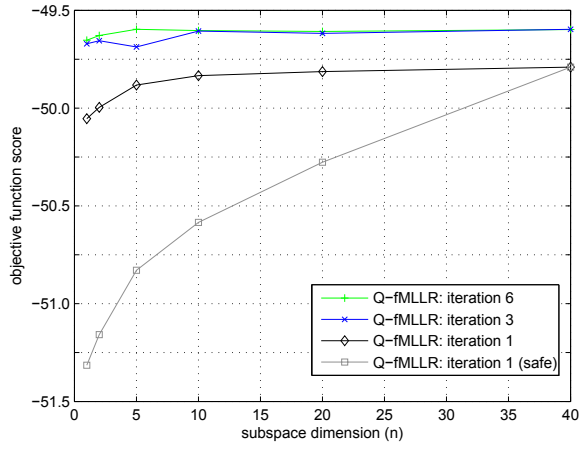

Fig. 1. As n increases, the objective function score of Q-fMLLR converges quickly to that of regular fMLLR ($n = 40$) except in the *safe* variant. Objective function before optimization was -53.75.

Table 1. Character error rates on dev'07, showing results after 1, 3, and 6 iterations of Q-fMLLR and one iteration of *safe* Q-fMLLR, with the subspace dimension n varying from 1 to 40.

n	Q-fMLLR:1	Q-fMLLR:3	Q-fMLLR:6	*safe* Q-fMLLR:1
1	16.8	16.5	16.4	17.5
2	16.8	16.5	16.4	17.4
5	16.7	16.5	16.4	17.3
10	16.7	16.4	16.4	17.2
20	16.7	16.4	16.4	17.0
40	16.7	16.4	16.3	16.7

The theoretical speedup from using n=1 versus the baseline fMLLR computation is about 40; the actual speedup observed in adaptation plus decoding was tiny because we tested on a very slow LVCSR system. The use we envisage for this approach is in much faster, small-vocabulary embedded speech recognizers.

5. CONCLUSIONS

We describe a novel method that can lead to significant reduction in the time taken for statistics accumulation in fMLLR while preserving the adaptation gains. The proposed Q-fMLLR algorithm is validated in a state-of-the-art large-vocabulary continuous speech recognition system. We show that Q-fMLLR attains the performance of regular fMLLR with a fraction of the computation. In fact, test results suggest that the extremely efficient 1-dimensional Q-fMLLR may be the best option as the loss incurred is likely to be negligible. These findings would have an especially significant impact in resource constrained systems, and may also be extended to efficient updates for systems with full covariance matrices [2] or multiple semi-tied covariance (STC) classes [5].

REFERENCES

[1] M.J.F Gales, "Maximum likelihood linear transformations for HMM-based speech recognition," *Computer Speech and Language*, vol. 12, pp. 75-98, 1998.

[2] D. Povey and G. Saon, "Feature and model space feature adaptation with full covariance Gaussians," in *Proc. ICSLP'06*, 2006.

[3] S. Wegmann, D. McAllaster, J. Orloff, and B. Peskin, "Speaker normalization on conversational telephone speech," in *Proc ICASSP'96*, vol. 1, pp. 339-343, May 1996.

[4] S. M. Chu, H.-K. Kuo, Y. Y. Liu, Y. Qin, Q. Shi, and G. Zweig, "The IBM Mandarin broadcast speech transcription system," in *Proc. ICASSP'07*, vol. 2, pp. 345-348, May 2007.

[5] M. J. F. Gales, "Semi-tied covariance matrices for Hidden Markov Models", in *IEEE Transactions on Speech and Audio Processing*, vol. 7, pp. 272-281, 1999.

[6] V. V. Digilakis, D. Rtischev and L. G. Neumeyer, "Speaker adaptation using constrained estimation of Gaussian mixtures", in *IEEE Transactions on Speech and Audio Processing*, vol. 3, pp. 357-366, 1995.