Estimating Constrained MLLR Parameters With Limited Adaptation Data

Daniel Povey and Kaisheng Yao

***Abstract*—Constrained MLLR (CMLLR) is a popular and effective adaptation technique in which a speaker-specific affine transform is applied to the speech features, mapping them into a "normalized" space in which speaker variation is reduced. For typical systems the CMLLR transformation matrix will have more than a thousand parameters, and about five seconds of speech is normally required for CMLLR to give improvements. In this paper we describe an effective technique for training the CMLLR transform on less data. This uses a basis representation of the CMLLR matrix, together with a novel optimization approach.**

I. Introduction

CONSTRAINED MLLR (CMLLR) [1], [2], also known as feature-space MLLR (fMLLR) is one of the most widely used methods of accounting for speaker variation in speech recognition. It is an affine transform on the features $\mathbf{x} \in \mathbb{R}^D$:

$$\mathbf{x} \rightarrow \mathbf{A}^{(s)}\mathbf{x} + \mathbf{b}^{(s)} \tag{1}$$

for speaker s, with parameters $\mathbf{A}^{(s)} \in \mathbb{R}^{D \times D}$ and $\mathbf{b}^{(s)} \in \mathbb{R}^D$. This may be written using the more compact notation

$$\mathbf{x} \rightarrow \mathbf{W}^{(s)}\mathbf{x}^+, \tag{2}$$

where $\mathbf{W}^{(s)} = \left[\mathbf{A}^{(s)} \; ; \; \mathbf{b}^{(s)}\right]$ and $\mathbf{x}^+ = \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}$. The matrix $\mathbf{W}^{(s)}$ is normally estimated in a Maximum Likelihood (ML) fashion on some adaptation data; in the usual scenario (unsupervised adaptation), this would be based on word labels generated from a speaker-independent decoding. In our experience, constrained MLLR needs about 5 seconds or so of adaptation data to robustly estimate the parameters, and with less adaptation data it can degrade performance.

In this paper we describe a new method which allows us to learn the CMLLR parameters more robustly from small amounts of data, while still giving the best possible performance for larger amounts of data. We represent the transformation using a basis:

$$\mathbf{W}^{(s)} = \mathbf{W}_0 + \sum_{b=1}^{D(D+1)} a_b^{(s)} \mathbf{W}_b, \tag{3}$$

where for a given speaker s we only allow a certain number of leading coefficients $a_b^{(s)}$ to be nonzero, i.e. $a_b^{(s)} = 0$ for $b > B(s)$, where $B(s)$ is the number of allowed parameters. For simplicity we take $\mathbf{W}_0$ to be $[\mathbf{I} \; ; \; \mathbf{0}]$. The basis $\{\mathbf{W}_b, 1 \leq b \leq D(D+1)\}$ needs to have the property that the directions with the most important variation are at the beginning. We will describe how we obtain such a basis and how we efficiently optimize the corresponding coefficients in test time.

In Section II we discuss prior work in this area. In Section III we discuss our approach in general terms and compare it with Bayesian methods. In Section IV we describe the mathematics of our approach in terms of a generic speaker-specific parameter. In Section V we provide update equations. In Section VI we describe our experimental results; in Section VII we conclude.

II. Prior work

Various approaches have been proposed to estimate CMLLR parameters from little data. Simple methods include limiting $\mathbf{A}^{(s)}$ to be a diagonal or block-diagonal matrix [2]. This is effective but is obviously not optimal.

Another alternative is fMAPLR [3], [4], which is based on picking the Maximum a Posteriori estimate of the parameter $\mathbf{W}^{(s)}$. For speed of update and ease of estimating the prior it is helpful to limit the prior to be the product of a separate Gaussian prior over each row of the transform. This is convenient to combine with the traditional row-by-row update formulas [2]. Both referenced papers [3], [4] used a diagonal Gaussian prior for each row. In [3] an "Empirical Bayes" approach was used: i.e., the prior was estimated from the ML estimates of the transforms for the training speakers; in [4] an E-M approach based on the Factor Analysis framework was used.

An earlier, similar scheme [5] smoothed the statistics accumulated for CMLLR with a certain number of frames' worth of "fake" statistics equivalent to expected data generated from the model itself. This is similar in spirit to the "relevance MAP" smoothing method used for model adaptation in HTK [6], and the update equations have the same form as fMAPLR. One thousand frames was found to work well.

In [7], the matrix $\mathbf{W}$ was represented as a weighted sum of basis matrices $\mathbf{W}_b$, with the coefficients (weights) being estimated for each speaker. This method has the advantage that it can model correlations between the rows, and it was demonstrated that this improves Word Error Rate (WER). This is a key motivation for our work here. However, the optimization in that method was quite difficult and the authors did not describe a single recipe that works for varying amounts of adaptation data; a fixed basis size was used.

The method we describe here grew out of work on differently structured models, namely Subspace Gaussian Mixture Models (SGMMs) which, most relevant to CMLLR estimation, contain a relatively small number (e.g. 1000) of tied full-covariance Gaussians [8], [9]. We previously described [10]

Microsoft Corporation, One Microsoft Way, Redmond, WA 98052, email: {dpovey,kaisheny}@microsoft.com

a version of this technique for SGMMs. The current paper represents a simplification of the mathematics of the original version, and also contains extensive experimental results with traditional diagonal-covariance models.

III. Discussion

Before introducing the specifics of our method, we discuss it as a generic parameter estimation problem, keeping equations to a minimum. At this level of detail, our approach would be applicable to any situation where we require a robust estimate of an unobservable variable, but where time constraints or the incorrectness of the model make it impractical to apply a traditional Bayesian approach.

In order to avoid introducing extra notation and to clarify the link with what comes later, we use notation specific to our method; however, at this stage the reader is encouraged to think of it as a generic problem. The problem in test time is to estimate a variable $\mathbf{w}^{(s)}$ for speaker s, given observations $\mathcal{O}^{(s)}$ (for us, this represents a variable-length sequence of feature vectors). We use the notation $\mathbf{w}^{(s)} = \mathrm{vec}\,(\mathbf{W}^{(s)^T})$ to forget the matrix structure of $\mathbf{W}^{(s)}$ and make the parameter a generic vector[1]. We assume the model and the transcripts used for supervision are given. Ideally, in test time we would like to use Bayes' rule as follows:

$$p(\mathbf{w}^{(s)}|\mathcal{O}^{(s)}) = \frac{p(\mathcal{O}^{(s)}|\mathbf{w}^{(s)})p(\mathbf{w}^{(s)})}{p(\mathcal{O}^{(s)})}. \quad (4)$$

We assume it is impractical to compute more than a point estimate of $p(\mathbf{w}^{(s)}|\mathcal{O}^{(s)})$. The denominator term $p(\mathcal{O}^{(s)})$ is irrelevant as it does not affect the estimated value of $p(\mathbf{w}^{(s)})$. If we keep just the term $p(\mathcal{O}^{(s)}|\mathbf{w}^{(s)})$ and maximize this, we get the Maximum Likelihood (ML) estimate of $p(\mathbf{w}^{(s)})$. The key extra part is the prior term $p(\mathbf{w}^{(s)})$. We assume that this is a Gaussian prior, but keep in mind that in high dimensions the key part is its covariance rather than its mean (which is much easier to estimate).

1) Simple Empirical Bayes: The parameter $\mathbf{w}^{(s)}$ is not an observed variable, so we are forced to rely on estimates of it to estimate the prior distribution. The most obvious solution is a simple "Empirical Bayes" approach where we compute the ML values of $\mathbf{w}^{(s)}$ for the training speakers and estimate the prior from these. The problem is that unless the training speakers all have a lot of data, the estimates of $\mathbf{w}^{(s)}$ will be dominated by estimation error and we will overestimate the variance of the prior. Also, if there are fewer training speakers than the dimension of $\mathbf{w}^{(s)}$, the estimated prior will be exactly zero in some directions.

2) Advanced Empirical Bayes: On the other hand assume that we are willing to use a more complex approach to estimate the prior over $\mathbf{w}$. We still make a point estimate of the prior parameters, but now we integrate over $\mathbf{w}^{(s)}$ for the training speakers. The most plausible scenario is that we initialize the prior parameters to some default value giving a flat distribution, then estimate the posterior distribution of $\mathbf{w}^{(s)}$ for each training speaker, then estimate the prior from the posterior distributions of all the $\mathbf{w}^{(s)}$, and then iterate until convergence. We are integrating over the uncertainty in $\mathbf{w}^{(s)}$ for training speakers. The problem is that, due to model incorrectness, we do not know exactly what that uncertainty is. This relates to the acoustic model "scaling factor" (typically between 1/10 and 1/20) that is used in speech recognition systems. It is not clear whether we should apply this scaling factor to $p(\mathcal{O}|\mathbf{w}^{(s)})$ in our likelihood term. We could introduce an arbitrary factor and tune it to optimize performance, but we do not anticipate that this would lead to good results. The problem is that we are trying to tease out a small signal (the variance of $\mathbf{w}^{(s)}$) from a large amount of noise (the estimation error in $\mathbf{w}^{(s)}$), and if we are forced to guess at the variance of the noise we could get radically different estimates of the signal depending what guess we make (i.e. depending what scaling factor we use). This would make this type of method non-robust, aside from the considerable practical difficulties.

[1]The transpose is convenient at a later stage.

A. Use of a basis

In this paper we use a basis method. There is no prior term involved; instead we re-parameterize the problem using a basis. In the vector notation, we would write

$$\mathbf{w}^{(s)} = \mathbf{w}_0 + \sum_{b=1}^{B(s)} a_b^{(s)} \mathbf{w}_b, \quad (5)$$

where N is the dimension of $\mathbf{w}$. Then for each speaker we limit $\mathbf{w}^{(s)}$ to be a sum over the first $1 \leq B(s) \leq N$ basis elements, where $B(s)$ will be larger if more data is available for the speaker. This is no different from an affine change of variables to a vector $\mathbf{a}$, and then enforcing a special form of sparsity on the new variable $\mathbf{a}$. By making appropriate assumptions it is possible to obtain an ML solution for the basis vectors $\mathbf{w}_b$. Our solution is Maximum Likelihood in the sense that for any given basis size B we choose, our solution maximizes the likelihood of the training set when modeling it using only the first B coefficients. Essentially the problem reduces to Principal Components Analysis (PCA) on suitably transformed and scaled variables, and each vector $\mathbf{w}_b$ corresponds to an eigenvector of a symmetric matrix. We make the following assumptions in order to reduce the problem to PCA: that the likelihood function is approximately quadratic in $\mathbf{w}_b$, and that the quadratic part of the likelihood function is about the same for each speaker (up to a speaker-specific scaling factor that in our case corresponds to the utterance length).

B. Relationship to Bayesian methods

It is instructive to compare our basis method with the use of a Bayesian prior. Think of $\mathbf{w}_0$ as the mean of a prior, and the directions $\mathbf{w}_b$ as the eigenvectors of a covariance matrix. It is possible to show via suitable assumptions, that the use of a prior is equivalent to a speaker-specific scaling factor in each dimension of $\mathbf{a}$, i.e. that for each $a_b^{(s)}$ we would have a scaling factor between zero and one that scales the Maximum likelihood estimate. The scaling factor, say $0 \leq \lambda_i^{(s)} \leq 1$, would be speaker-dependent (given our assumptions, it would just depend on the amount of data available for the speaker)

and would decrease with increasing i. Very crudely, we can view the enforced sparsity on the vector $\mathbf{a}$ as a quantization of this scaling factor $\lambda_i^{(s)}$, so that the first $B(s)$ values are one and the rest are zero.

As far as the estimation of the basis is concerned, if we make similar assumptions within the Bayesian framework (i.e. about the quadratic nature of the likelihood function), we can recover quite similar equations to those we obtain for our basis method. A key difference is that a per-speaker scaling factor arises in our basis method (proportional to the amount of data for that speaker), and that factor does not initially arise within the Bayesian framework. However, if the Maximum Likelihood estimates of the parameters $\mathbf{w}^{(s)}$ are assumed to be dominated by estimation error in all dimensions, it is possible to recover this speaker-specific factor within a Bayesian approach. This would be done by choosing a very "tight" prior *a priori*, with variance equal to a small multiple of the inverse of the Fisher Information Matrix, estimating the *maximum a posteriori* (MAP) values of training parameters, and estimating our prior from these values.

A key practical difference between the basis method and a Bayesian approach is that in the basis method we keep only the eigenvectors of the prior covariance matrix, and ignore the eigenvalues. Let us assume that this prior covariance matrix is being estimated after a change of variables such that the Hessian of the likelihood function is proportional to the unit matrix (this corresponds to the calculation we use in the basis method). It is possible to justify throwing away the eigenvalues by pointing to the problem of bias in the estimation of the eigenvalues. The noise introduced into the prior covariance matrix due to estimation error in the values of $\mathbf{w}^{(s)}$ for training speakers, is proportional to the unit matrix[2]. Thus, it increases the eigenvalues of this matrix but in the limit of sufficient training data it has no effect on the eigenvectors. It may seem paradoxical from a parameter estimation point of view that we would not trust our estimates of the relatively small number of eigenvalues, and yet would trust the much larger number of parameters in the eigenvectors. There are two responses we can make to this. Firstly, we view it as a question of estimation bias rather than of variance in the estimate. Secondly, we do not need very accurate estimates of the eigenvectors at all. Suppose we have somehow obtained the "true" value of the covariance matrix, and we swap the last and the second-to-last eigenvector. The eigenvectors become completely inaccurate, and yet the effect on the covariance matrix will be tiny because those eigenvectors probably have quite similar eigenvalues. It is the leading few eigenvectors that are most important, and these can be more accurately estimated because in those directions the signal dominates the noise.

The closest Bayesian equivalent of our basis method would be as follows: transform the variable to make the average negated Hessian proportional to unity, and then estimate the prior covariance matrix in some Empirical Bayes fashion. Then keep only its (ordered) eigenvectors and replace its eigenvalues with some set of eigenvalues decided upon *a priori*, for instance eigenvalues λ_i proportional to $1/i$. We have not experimented with these types of approaches.

[2]We can justify this claim by reference to identities on the Fisher Information Matrix, which for well-behaved distributions can be defined in two different ways.

C. Impact on optimization

Up till now, our discussion has treated $\mathbf{w}^{(s)}$ as a generic parameter without reference to its structure as a transformation matrix. We have not considered how we would optimize the likelihood function in test time, subject either to a prior or to the basis constraint. Recalling that $\mathbf{w} = \mathrm{vec}\,(\mathbf{W}^T)$, the familiar approach [2] updates each row of $\mathbf{W}$ in turn. This does not interact well with a basis method, since in a basis method all the rows change at once when we update the coefficients. It could also converge slowly when using a conventional prior, if the prior introduced strong correlations between the rows of $\mathbf{W}$.

In our work, we optimize $\mathbf{w}^{(s)}$ by gradient descent, but crucially we do this via the coefficients $a_b^{(s)}$, and the Hessian with respect to these is proportional to the unit matrix. This makes the gradient descent converge very fast. Additionally we experiment with conjugate gradient methods, but since the problem is already so well conditioned this gives little additional benefit.

D. Summary

In this section we have explained how our basis method relates to Bayesian approaches and discussed why it solves certain problems that arise when attempting to apply Bayesian methods to this type of parameter estimation problem. We do not believe that our basis method is an optimal solution– in fact, there are some obvious "more Bayesian" extensions of it that would probably do better. The difficulty is that after deciding on an approximate Bayesian approach, one is faced with an almost infinite array of choices. Conversely, the principles behind our basis method are very simple and most aspects of the algorithm flow naturally from a few assumptions. In the experiments we report here, we have limited the Bayesian baselines to the fMAPLR approach that has previously been investigated in the literature; we consider a more thorough investigation of the possible Bayesian alternatives to be beyond the scope of this paper. We also do not experiment with the technique of [7] since it was described only in general terms, and in its original form it does not appear to be a viable method if the utterance length of test data varies widely.

IV. Our Method

In this section we describe our method with more mathematical details, but still at a relatively generic level (i.e. treating $\mathbf{w}^{(s)}$ as a generic vector).

A. Preconditioning

The first stage is to precondition the problem so that the quadratic part of the auxiliary function is approximately proportional to the unit matrix. We define

$$\mathbf{H} \equiv -E\left[\left.\frac{\partial^2 \log p(\mathbf{x})}{\partial \mathbf{w}^2}\right|_{\mathbf{w}=\mathbf{w}_0}\right] \tag{6}$$

where the expectation is taken over acoustic vectors $\mathbf{x}$ generated from the model, and $\mathbf{w}_0$ is the "default" or mean value of the parameter $\mathbf{w}$; in our case it corresponds to the matrix $\mathbf{W}_0 = [\mathbf{I}\,;\,\mathbf{0}]$. In fact, in our computation of $\mathbf{H}$ we approximate the likelihood function by the auxiliary function, but the difference is small (and using the auxiliary function is more appropriate anyway for a single iteration of update). We can compute $\mathbf{H}$ from the Hidden Markov Model (HMM) parameters (including occupation probabilities for particular states). Next we do the Cholesky decomposition

$$\mathbf{H} = \mathbf{C}\mathbf{C}^T \tag{7}$$

and define a transformed parameter

$$\hat{\mathbf{w}} = \mathbf{C}^T\mathbf{w}\ . \tag{8}$$

We can then show that

$$E\left[\left.\frac{\partial^2 \log p(\mathbf{x})}{\partial \hat{\mathbf{w}}^2}\right|_{\hat{\mathbf{w}}=\hat{\mathbf{w}}_0}\right] = -\mathbf{I} \tag{9}$$

i.e. the Hessian w.r.t. $\hat{\mathbf{w}}$ is now unit (in expectation, per frame of data). Our problem is now well conditioned. Note that it is only well conditioned in an averaged sense over speakers, since Equation (9) is an expectation and does not imply an exact equality for a particular sequence of features.

B. Basis computation

The next stage is to compute the basis, i.e. the set of vectors $\mathbf{w}_b$ for $1 \leq b \leq N$ where N is the dimension of $\mathbf{w}$. For a testing speaker s we will choose some basis size $B(s)$ and limit $\mathbf{w}^{(s)} - \mathbf{w}_0$ to be a combination of $\mathbf{w}_b$ for $1 \leq b \leq B(s)$. There is a natural Maximum Likelihood solution to the problem of basis estimation, once we assume the likelihood function is quadratic and approximate the Hessian by its expected value. We assume in the text below that we are trying to solve the basis learning problem for some fixed basis size B with $1 \leq B \leq N$, but in fact this naturally leads to a solution valid simultaneously for all basis sizes. We are solving for a matrix $\mathbf{X}_B$ with B columns $\mathbf{w}_1, \ldots, \mathbf{w}_B$. It is easiest to solve this initially in the pre-conditioned space, so we solve for $\hat{\mathbf{X}}_B = \mathbf{C}^T\mathbf{X}_B$, transformed as in (8). We can assume without loss of generality that the columns of $\hat{\mathbf{X}}_B$ are orthonormal:

$$\hat{\mathbf{X}}_B^T\hat{\mathbf{X}}_B = \mathbf{I}\ . \tag{10}$$

This is true because non-singular linear transformations of the columns of $\hat{\mathbf{X}}_B$ do not affect the subspace spanned by them, hence do not affect the objective function, and a solution with linearly dependent columns can always be turned into one with linearly independent columns without decreasing the objective function value (using $B \leq N$). It follows that given some $\hat{\mathbf{X}}_B$ we can find one satisfying (10) with the same or better objective function value.

For a speaker s, let us write the number of frames of speech observations as $\beta^{(s)}$ (this is the conventional notation used for the statistics accumulation in Constrained MLLR [2]). Without knowing anything about the speaker other than $\beta^{(s)}$, we can write as follows, where $\mathcal{O}^{(s)}$ is the sequence of frames $\mathbf{x}_1^{(s)}, \ldots, \mathbf{x}_{\beta^{(s)}}^{(s)}$ for the speaker:

$$E\left[\left.\frac{\partial^2 \log p(\mathcal{O}^{(s)})}{\partial\ \hat{\mathbf{w}}^{(s)\,2}}\right|_{\hat{\mathbf{w}}^{(s)}=\hat{\mathbf{w}}_0}\right] = -\beta^{(s)}\mathbf{I}\ . \tag{11}$$

We then assume this is true not just as an expectation, but as an equality, and in fact for a sufficient number of frames it will be a close enough approximation. Next we define

$$\hat{\mathbf{p}}^{(s)} = \left.\frac{\partial \log p}{\partial \hat{\mathbf{w}}^{(s)}}\right|_{\hat{\mathbf{w}}^{(s)}=\hat{\mathbf{w}}_0} \tag{12}$$

which is the gradient of the log-likelihood function with respect to $\hat{\mathbf{w}}^{(s)}$, taken at the "default value" $\hat{\mathbf{w}}_0 = \mathbf{C}^T\mathbf{w}_0$. Using a quadratic approximation to the likelihood function with (11) as an approximation to the quadratic term,

$$p(\mathcal{O}^{(s)}|\hat{\mathbf{w}}) \simeq C + (\hat{\mathbf{w}} - \hat{\mathbf{w}}_0)\cdot\hat{\mathbf{p}}^{(s)} - \frac{(\hat{\mathbf{w}} - \hat{\mathbf{w}}_0)^T(\hat{\mathbf{w}} - \hat{\mathbf{w}}_0)}{2\beta^{(s)}}. \tag{13}$$

Constraining the value of $\hat{\mathbf{w}} - \hat{\mathbf{w}}_0$ to the basis $\hat{\mathbf{X}}_B$, and using (10), it is not hard to show that this is maximized by:

$$\hat{\mathbf{w}}^{(s)} = \hat{\mathbf{w}}_0 + \frac{1}{\beta^{(s)}}\hat{\mathbf{X}}_B\hat{\mathbf{X}}_B^T\hat{\mathbf{p}}^{(s)}. \tag{14}$$

For brevity, let us write $\hat{\mathbf{q}}^{(s)}$ for the second term of (14), which is the difference $\hat{\mathbf{w}}^{(s)} - \hat{\mathbf{w}}_0$. We can write the change in our approximated log-likelihood of (13) as:

$$\begin{aligned}\Delta p &\simeq \hat{\mathbf{p}}^{(s)}\cdot\hat{\mathbf{q}}^{(s)} - \frac{1}{2\beta^{(s)}}\ \hat{\mathbf{q}}^{(s)\,T}\,\hat{\mathbf{q}}^{(s)} \\ &= \frac{1}{2\beta^{(s)}}\ \hat{\mathbf{p}}^{(s)\,T}\,\hat{\mathbf{X}}_B\hat{\mathbf{X}}_B^T\hat{\mathbf{p}}^{(s)}\end{aligned} \tag{15}$$

(using (10)), and we can recognize the task of maximizing (15) as a Principal Components Analysis (PCA) problem on the following matrix:

$$\hat{\mathbf{M}} = \sum_s \frac{1}{\beta^{(s)}}\hat{\mathbf{p}}^{(s)}\ \hat{\mathbf{p}}^{(s)\,T}\ . \tag{16}$$

This can be solved by computing the eigenvectors of $\hat{\mathbf{M}}$ in order from greatest to least eigenvalue, setting $\hat{\mathbf{X}}$ to the matrix whose columns are these eigenvectors, and making $\hat{\mathbf{X}}_B$ the matrix containing the first B columns of $\hat{\mathbf{X}}$. Thus, we naturally obtain a solution that is simultaneously valid for all basis sizes. Note that the scatter of the ML-estimated parameters would have an extra factor of $\beta^{(s)}$ on the bottom of (16); we referred to this issue in Section III.

C. Optimization in test time

After computing this basis we automatically have a fast way to obtain the parameter $\mathbf{w}^{(s)}$ for a given speaker; note that we refer now to maximizing an auxiliary function rather than a quadratic approximation to one. If we can efficiently compute the gradient of the auxiliary function $\mathcal{Q}(\mathbf{w}^{(s)})$, then we can optimize it as follows. Suppose for test speaker s we limit the basis size to $B(s)$. Then let $\mathbf{a}^{(s)} \in \mathbb{R}^{B(s)}$, and we will use

$$\mathbf{w}^{(s)} = \mathbf{X}_{B(s)}\mathbf{a}^{(s)}. \tag{17}$$

Note that this equation is written in terms of the original, un-transformed parameter $\mathbf{w}^{(s)}$. The likelihood function expressed as a function of $\mathbf{a}^{(s)}$ is automatically well conditioned; it is easy to show that the quadratic term in the approximated likelihood function is $-\frac{1}{2}\beta^{(s)}(\mathbf{a}^{(s)} \cdot \mathbf{a}^{(s)})$. Thus, the gradient direction on $\mathbf{a}^{(s)}$ will automatically be a good direction to search in, and all we have to do is to compute the gradient w.r.t. $\mathbf{a}^{(s)}$, take the optimal step size in that direction, and repeat. Using this approach, we automatically get the benefit of the matrix $\mathbf{C}$ used for preconditioning, but we do not need to refer to it in test time since the information we need is implicit in $\mathbf{X}$.

We have slightly glossed over the issue of whether it is the likelihood function or the auxiliary function that is well-conditioned. In practice it is almost the same thing for speech recognition tasks. In Section IV-A we assume it is the likelihood function that we condition well, but we actually compute $\mathbf{H}$ as the Hessian of the auxiliary function since this is easier to compute.

D. Application to Constrained MLLR estimation

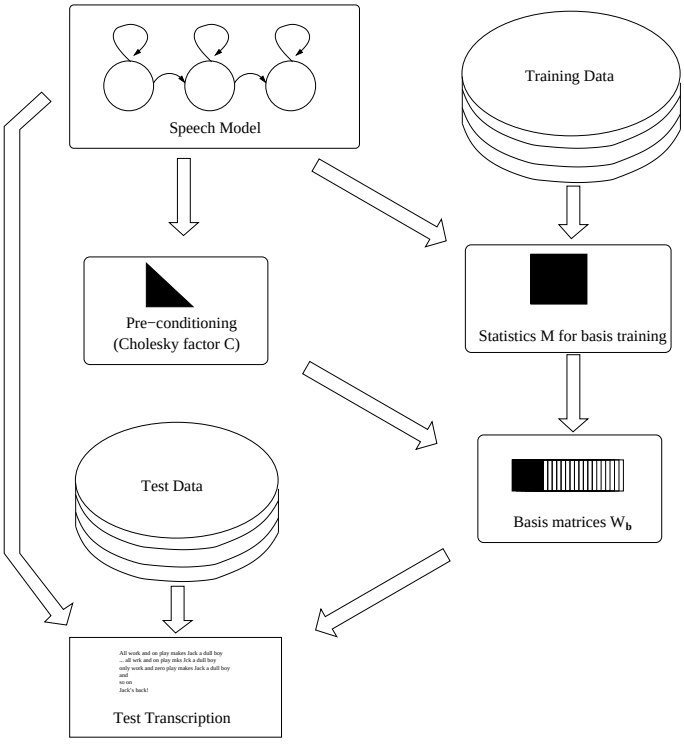

Fig. 1. Overall process

Turning now to the application of this method to Constrained MLLR estimation, Figure 1 describes the overall training and testing process. This assumes that we have already built a conventional HMM-GMM system. We compute the Cholesky matrix $\mathbf{C}$ (this can be done directly from the model), and then we compute the matrix $\mathbf{M}$ in order to train the basis (this is the same as $\hat{\mathbf{M}}$ but in un-transformed coordinates). From $\mathbf{M}$ and $\mathbf{C}$ we can compute the matrix $\mathbf{X}$, each column of which is (the vectorized form of) one of the basis matrices $\mathbf{W}_b$ for $1 \leq b \leq D(D+1)$, where D is the feature dimension (e.g. 39). In test time, for a particular speaker s we decide the basis size $B(s)$ and then compute our coefficients $\mathbf{a}^{(s)} = a_1^{(s)}, \ldots, a_{B(s)}^{(s)}$. These coefficients are optimized by repeated line search in the gradient direction.

E. Conjugate gradient modification

As an improvement to the basic optimization framework of line search in the gradient direction, we can use directions determined by the conjugate gradient method [11]. This is not necessary but it can speed up convergence. Conjugate gradient methods are applicable to the situation where one can measure the gradient and then do line search in some chosen direction, and this is exactly the situation we have here. These methods are very simple to use, since the step direction is just a linear combination of the gradient direction with the previous iteration's step direction, with an interpolation factor that depends only on the current and previous iteration's gradient and step directions. Various formulae are available to determine this factor; we choose a popular approach which is the Polak-Ribière formula with "restarting" (see [11]).

V. Details of our proposed method

A. Statistics and auxiliary function

The statistics required for CMLLR estimation can be stored in various forms, and which one is optimal depends on the system architecture and a tradeoff between memory and efficiency [2]. For conventionally structured diagonal systems, the most convenient statistics are generally as follows: for dimension $1 \leq i \leq D$,

$$\mathbf{G}_i = \sum_t \sum_{j,m} \gamma_{jm}(t) \frac{1}{\sigma_{jm}^2(i)} \mathbf{x}(t)^+ \mathbf{x}(t)^{+T} \tag{18}$$

$$\mathbf{K} = \sum_t \sum_{j,m} \gamma_{jm}(t) \boldsymbol{\Sigma}_{jm}^{-1} \boldsymbol{\mu}_{jm} \mathbf{x}(t)^{+T} \tag{19}$$

$$\beta = \sum_t 1 \tag{20}$$

where $\sum_t$ is a sum over the frames for the speaker in question and $\gamma_{jm}(t)$ is the posterior of Gaussian mixture index m of state j at time t, derived from the forward-backward algorithm or Viterbi alignment. The Gaussian means and (diagonal) variances are $\boldsymbol{\mu}_{jm}$ and $\boldsymbol{\Sigma}_{jm}$, with the diagonal elements of the variance being written $\sigma_{jm}^2(i)$. We will put speaker superscripts (s) on these statistics where necessary for clarity.

The auxiliary function can be written as follows:

$$\mathcal{Q}(\mathbf{W}) = \mathrm{tr}\,(\mathbf{W}^T \mathbf{K}) + \beta \log |\det \mathbf{A}| - \frac{1}{2} \sum_i \mathbf{w}_i^T \mathbf{G}_i \mathbf{w}_i \tag{21}$$

where $\mathbf{A}$ is the first D columns of $\mathbf{W}$, and $\mathbf{w}_i$ is the i'th row of $\mathbf{W}$ (transposed). We aim only to find a local maximum of this function close to the default value $\mathbf{W}_0$; finding a global maximum is hard, and we do not believe it would lead to improved performance in terms of error rates.

For background: the CMLLR update that is generally used [2] is based on optimizing each row of $\mathbf{W}$ while holding all the others fixed. The only interaction between the rows arises from the log determinant term of (21), and given the

other rows it is possible to compute a particular row's value in closed form. For a number of iterations, this is done for each row in turn; ten iterations is usually sufficient.

B. Cholesky matrix computation

In this section we describe how we obtain the Cholesky matrix $\mathbf{C}$ of (7) that we use for preconditioning. We first compute the matrix $\mathbf{H}$ of (6). As discussed, we approximate the likelihood function by the auxiliary function (21), and we take the derivative at the value $\mathbf{W}^{(s)} = \mathbf{W}_0 = [\mathbf{I}\ ;\ \mathbf{0}]$. With $\mathbf{W}^{(s)}$ fixed, the second derivative of (21) depends only on the quantities $\mathbf{G}_i$, and we need the expected values of $\mathbf{G}_i$ for a single frame of data. Let these be written as $\bar{\mathbf{G}}_i$, for $1 \leq i \leq D$. Thus,

$$\bar{\mathbf{G}}_i^{(s)} = \sum_{j,m} P(j,m) \frac{1}{\sigma_{jm}^2(i)} \left(\boldsymbol{\mu}_{jm}^+ \boldsymbol{\mu}_{jm}^{+\,T} + \boldsymbol{\Sigma}_{jm}^{+0} \right), \quad (22)$$

where $P(j,m)$ is the occupation probability of Gaussian j of mixture m, with $\sum_{j,m} P(j,m) = 1$, and Σ_{jm}^{+0} (our own notation) means Σ_{jm} extended with an extra row and column with zero values. The occupation probabilities $P(j)$ for the states can be computed using occupation statistics from model training if available, or otherwise we can just use a flat prior $P(j) = 1/J$ (we did this). Then the Gaussian-level occupation probabilities are $P(j,m) = P(j)c_{jm}$ where c_{jm} are the mixture weights.

We are computing the second derivative of (21), and since this is for just one frame of data, in expectation, we take β to be 1 and $\mathbf{G}_i = \bar{\mathbf{G}}_i$. There are two terms in the Hessian, one arising from the log determinant term in (21) and one involving $\mathbf{G}_i$. The part arising from the log determinant is:

$$\frac{\partial^2}{\partial w_{ij} \partial w_{kl}} \log(|\det \mathbf{A}|) \Big|_{\mathbf{A}=\mathbf{I}} = -\delta(i,l)\delta(j,k), \quad (23)$$

so each element of the matrix is only correlated with the corresponding transposed element. We can view this as a matrix $\mathbf{H}^{(1)} \in \mathbb{R}^{D(D+1) \times D(D+1)}$, with elements

$$h^{(1)}_{((i-1)(D+1)+j),((k-1)(D+1)+l)} = \delta(i,l)\delta(j,k). \quad (24)$$

for $1 \leq i,j,k,l \leq D$, where the elements of $\mathbf{H}^{(1)}$ that are not covered by this formula are equal to zero. This can be computed as follows: first set $\mathbf{H}^{(1)} \leftarrow \mathbf{0}$ and then for $1 \leq i,j \leq D$, set $h^{(1)}_{((i-1)(D+1)+j),((j-1)(D+1)+i)} \leftarrow 1$.

The part of the Hessian arising from the term in (21) involving $\mathbf{G}_i$, can be written as the block-diagonal matrix:

$$\mathbf{H}^{(2)} = \mathrm{diag}\,(\bar{\mathbf{G}}_1, \bar{\mathbf{G}}_2, \ldots, \bar{\mathbf{G}}_D) \quad (25)$$

where we interpret diag here in the obvious way[3]. So our total expected per-frame negated Hessian is

$$\mathbf{H} = \mathbf{H}^{(1)} + \mathbf{H}^{(2)}. \quad (26)$$

This is a sparse matrix but its Cholesky is not sparse and it has a rather difficult structure so it is best treated as dense; computing its Cholesky will not dominate the pre-computation phase anyway. We then do Cholesky decomposition as in (7) to get $\mathbf{C}$.

[3]The reason for the transpose in $\mathbf{w} = \mathrm{vec}\,(\mathbf{W}^T)$ is to get this block diagonal structure.

C. Computing the basis

It is easiest to compute the matrix $\hat{\mathbf{M}}$ of (16) by first computing $\mathbf{M}$ in the original space (i.e. no multiplication by the Cholesky) and then transforming it. Analogous to (12), we define in the original space:

$$\mathbf{p}^{(s)} = \frac{\partial \mathcal{Q}}{\partial \mathbf{w}^{(s)}} \Big|_{\mathbf{w}^{(s)}=\mathbf{w}_0} \quad (27)$$

$$= \mathrm{vec} \left(\left(\mathbf{K}^{(s)} + \beta^{(s)} [\mathbf{I}\ ;\ \mathbf{0}] - \begin{bmatrix} \mathbf{g}_{11}^{(s)} \\ \vdots \\ \mathbf{g}_{DD}^{(s)} \end{bmatrix} \right)^T \right) \quad (28)$$

where by $\mathbf{g}_{ii}^{(s)}$ we mean the i'th row (or column) of $\mathbf{G}_i^{(s)}$. Note that this is a specialization of a more general expression we introduce later as (34); here, we use $\mathbf{W} = [\mathbf{I}\ ;\ \mathbf{0}]$ to simplify. Next we accumulate the quantity:

$$\mathbf{M} = \sum_s \frac{1}{\beta^{(s)}} \mathbf{p}^{(s)}\ \mathbf{p}^{(s)\,T}. \quad (29)$$

Using $\hat{\mathbf{w}} = \mathbf{C}^T \mathbf{w}$, and given that gradients always transform with the inverse transpose of the transformation on the vector, we have $\hat{\mathbf{p}} = \mathbf{C}^{-1}\mathbf{p}$, so $\hat{\mathbf{M}} = \mathbf{C}^{-1}\mathbf{M}\mathbf{C}^{-T}$. It is typically more robust to use Singular Value Decomposition (SVD) rather than eigenvalue decomposition, so we do the SVD:

$$\hat{\mathbf{M}} = \mathbf{U}\mathbf{L}\mathbf{V}^T, \quad (30)$$

in which we require that the diagonal matrix $\mathbf{L}$ be sorted from greatest to least value (if not, we need to sort this and the corresponding columns of $\mathbf{U}$). Since $\hat{\mathbf{M}}$ is positive semidefinite we can show that the columns of $\mathbf{U}$ are eigenvectors of $\hat{\mathbf{M}}$ and the diagonal elements of $\mathbf{L}$ are the corresponding eigenvalues. The basis matrices $\mathbf{W}_b$ for $1 \leq b \leq D(D+1)$ are given by the identity

$$\mathbf{C}^{-T}\mathbf{u}_b = \mathrm{vec}\,(\mathbf{W}_b^T), \quad (31)$$

where $\mathbf{u}_b$ is the b'th column of $\mathbf{U}$. This completes the pre-computation phase. We only need to keep the basis matrices $\mathbf{W}_b$.

D. Test-time: optimizing the matrix

In test time we do not explicitly work with or store the coefficients $a_b^{(s)}$ as it is more convenient to work directly with the matrix $\mathbf{W}^{(s)}$. The computation on $a_b^{(s)}$ is implicit. We use the statistics from (18) to (20), and we require some initial value of $\mathbf{W}^{(s)}$ to start from. This initial value may be the default value $\mathbf{W}_0 = [\mathbf{I}\ ;\ \mathbf{0}]$, or in an iterative or on-line setting it may be some previously optimized value. In the text below we will write $\mathbf{W}^{(s)}$ simply as $\mathbf{W}$ for brevity.

To choose the basis size for speaker s, for our experiments here we used the formula:

$$B(s) = \min(\lfloor \eta \beta^{(s)} \rfloor, D(D+1)), \quad (32)$$

where the constant η (e.g. $\eta = 0.2$) represents the number of parameters we introduce for each new frame of speech data. This formula means that the number of parameters rises

linearly with the amount of training data, up to the maximum possible value.

The phases 1), 2), 3) and 4) below should be done repeatedly, for iterations $1 \leq q \leq Q$. Between $Q = 5$ and $Q = 20$ iterations is a reasonable number.

1) Check auxiliary function: On each iteration of update it makes sense for debugging purposes to check the auxiliary function of (21). This is bound to increase (or stay the same) from iteration to iteration and a decrease indicates an error.

2) Compute auxiliary function gradient: To compute the auxiliary function gradient $\mathbf{P} \in \mathbb{R}^{D \times (D+1)}$, we first compute a matrix $\mathbf{S} \in \mathbb{R}^{D \times (D+1)}$ which is the contribution to the gradient from the quadratic term. Defining $\mathbf{s}_i$ and $\mathbf{w}_i$ as the i'th rows of $\mathbf{S}$ and $\mathbf{W}$ respectively, viewed as column vectors, we do:

$$\mathbf{s}_i = \mathbf{G}_i \mathbf{w}_i \tag{33}$$

$$\mathbf{P} = \beta \left[\mathbf{A}^{-T};\ \mathbf{0} \right] + \mathbf{K} - \mathbf{S} \tag{34}$$

where $\mathbf{A}$ corresponds to the first D columns of $\mathbf{W}$. From this we compute the gradient w.r.t. the retained coefficients $a_b^{(s)}$ for $1 \leq b \leq B(s)$. This is a vector $\mathbf{d} \in \mathbb{R}^{B(s)}$, with elements

$$d_b = \mathrm{tr}\,(\mathbf{W}_b^T \mathbf{P}). \tag{35}$$

Each d_b can be computed in $O(D^2)$.

3) Compute step direction: The next phase is to compute the step direction $\mathbf{\Delta} \in \mathbb{R}^{D \times D+1}$. If we are not doing the conjugate-gradient modification, this is quite simple and we set:

$$\mathbf{\Delta} = \sum_{b=1}^{B(s)} d_b \mathbf{W}_b. \tag{36}$$

This is equivalent to the gradient direction on the coefficients $a_i^{(s)}$.

With the conjugate-gradient modification, we need a linear combination of the formula above and the previous step direction. We write the preconditioned gradient direction as $\mathbf{\nabla}$:

$$\mathbf{\nabla} = \sum_{b=1}^{B(s)} d_b \mathbf{W}_b. \tag{37}$$

If q=1 (we are on the first iteration), we set:

$$\mathbf{\Delta} = \mathbf{\nabla} \tag{38}$$

$$\bar{\mathbf{\nabla}} = \mathbf{\nabla}, \tag{39}$$

where we use the notation $\bar{\mathbf{\nabla}}$ for the "preconditioned" gradient direction $\mathbf{\nabla}$ from the previous iteration. Otherwise ($q > 1$), we use the Polak-Ribière formula "with restarting" [11]:

$$b = \max\left(0, \frac{\mathrm{tr}\,(\mathbf{\nabla}^T \mathbf{\nabla}) - \mathrm{tr}\,(\mathbf{\nabla}^T \bar{\mathbf{\nabla}})}{\mathrm{tr}\,(\bar{\mathbf{\nabla}}^T \bar{\mathbf{\nabla}})}\right) \tag{40}$$

$$\mathbf{\Delta} = \mathbf{\nabla} + b\mathbf{\Delta} \tag{41}$$

$$\bar{\mathbf{\nabla}} = \mathbf{\nabla}. \tag{42}$$

The variables $\bar{\mathbf{\nabla}}$ and $\mathbf{\Delta}$ must be retained between iterations.

4) Compute optimal step size: We will take a step $\mathbf{W} \leftarrow \mathbf{W} + k\mathbf{\Delta}$ for a positive scalar k. This is computed iteratively starting from $k = 0$, via Newton's method. Note that if the Hessian $\mathbf{H}$ is accurate for the current speaker's data we expect the final value of k to be about 1. If it has a very different order of magnitude, this may indicate an error.

We first compute the scalars:

$$m = \mathrm{tr}\,(\mathbf{\Delta}\mathbf{K}^T) - \mathrm{tr}\,(\mathbf{\Delta}\mathbf{S}^T) \tag{43}$$

$$n = \sum_{i=1}^{D} \boldsymbol{\delta}_i^T \mathbf{G}_i \boldsymbol{\delta}_i, \tag{44}$$

where $\boldsymbol{\delta}_i$ is the i'th row of $\mathbf{\Delta}$, viewed as a column vector. We are maximizing the auxiliary function

$$\mathcal{Q}(k) = \beta \log |\det(\mathbf{A} + k\mathbf{\Delta}_{1:D})| + k\,m - \frac{1}{2}k^2 n \tag{45}$$

where $\mathbf{\Delta}_{1:D}$ represents the first D columns of $\mathbf{\Delta}$. For each iteration $1 \leq r \leq R$ (we used $R = 3$), we compute the quantities $d_1 \equiv \frac{\partial \mathcal{Q}(k)}{\partial k}$ and $d_2 \equiv \frac{\partial^2 \mathcal{Q}(k)}{\partial k^2}$ as follows:

$$\mathbf{N} = (\mathbf{A} + k\mathbf{\Delta}_{1:D})^{-1} \mathbf{\Delta}_{1:D} \tag{46}$$

$$d_1 = \beta \mathrm{tr}\,(\mathbf{N}) + m - kn \tag{47}$$

$$d_2 = -\beta \mathrm{tr}\,(\mathbf{N}\mathbf{N}) - n. \tag{48}$$

We then generate the candidate new value of k,

$$\hat{k} = k - d_1/d_2. \tag{49}$$

We evaluate (45) for k and $\hat{k}$. If $\mathcal{Q}(\hat{k}) < \mathcal{Q}(k)$, then we have overshot and we need to reduce the step size by setting $\hat{k} \leftarrow (\hat{k} + k)/2$. This should happen rarely, and if it does happen we should keep checking that the auxiliary function has not decreased and keep setting $\hat{k} \leftarrow (\hat{k}+k)/2$ until $\mathcal{Q}(\hat{k}) \geq \mathcal{Q}(k)$. When this terminates set set $k \leftarrow \hat{k}$. If we halve the step size more than, say, ten times then the auxiliary function decrease may have been due to roundoff error and we may already have converged, so in this case we would just leave k unchanged and terminate the loop over r.

VI. Experimental Results

A. System description

1) Common system features: We used two systems to evaluate our technique. Both use cross-word triphone context dependency, with standard phone context clustering and three-state left-to-right HMMs. The features are 13-dimensional MFCC + Δ,$\Delta\Delta$ and $\Delta\Delta\Delta$ features, reduced in dimension with HLDA. The frame shift is 10ms. We use cepstral mean normalization; in test time this is applied in an on-line fashion. We do not use VLTN. We train gender-independent models and further train these for four more E-M iterations on only male or female data, to obtain gender-specific models, before discriminative training. The gender-independent models are used in the first pass of decoding to obtain the supervision hypothesis and the corresponding phone-level alignment.

2) Interactive Voice Response (IVR) system: Our first system is a speaker-independent interactive voice response (IVR) system, geared towards telephone applications with short utterances. The training data consists of 7500 hours of speech, mostly voice search data recorded over the telephone but also read speech. This is doubled to 15 000 hours of speech by duplicating the data and applying cepstral mean normalization either at a per-utterance or per-session level. The average length of training utterances is 5.3 seconds. The feature dimension after HLDA is 36. Each of the three (GI, male and female) models has 9116 clustered states with 46 Gaussians per state on average. The models were trained with Minimum Classification Error (MCE) [12]. The test set contains three subsets consisting of digits, city-state pairs and stock names, totaling 20 hours of speech, with 21K words and an average utterance length of 3.4 seconds. Such a large test set is necessary because the WERs in this domain are very low, particularly for digit strings.

3) Extended Voice Mail (EVM) system: This is an LVCSR system that is tested mostly on longer utterances. The system architecture is as before but with 33 dimensional feature vectors and MMIE [13]. The number of clustered states is 10144, with 47 Gaussians per state on average. The training data consists of 1700 hours of read speech, with an average utterance length of 5.3 seconds, plus 130 hours of voicemail recordings with an average utterance length of 35.3 seconds; the statistics from the voicemail are scaled up by a factor of 20 in training. We present results averaged over five different sources of voicemail test data, totaling 507 utterances with 4 hours of speech in total. The average length of the test utterances is 28.4 seconds.

TABLE I
BASELINE WERS AND WORD COUNTS, BY UTTERANCE DURATION BINS

	IVR				EVM			
Min Duration	1.2	2.9	4.7	6.4	2.8	20.2	37.6	72.3
Max Duration	2.9	4.7	6.4	15.1	20.2	37.6	72.3	176.5
Mean Duration	2.2	3.8	5.4	7.7	11.6	28.1	51	101
#Words	29K	22K	30K	12K	7K	11K	12K	6.5K
%WER (Unadapted)	5.64	1.63	0.65	0.98	32.8	31.4	33.3	35.8

4) Data subsets: In order to see how our technique performs on different utterance lengths, we broke up the IVR and EVM test sets into four subsets each, corresponding to different duration bins. The bins were generated by initializing them as equal-sized duration intervals and then merging bins as necessary to have a specified minimum number of words in each. Table I describes these bins, along with the baseline (unadapted) Word Error Rates in each bin. There is a very large variation in Word Error Rates because different bins are dominated by different types of test data. In particular, the longer IVR bins are dominated by digits which have very low error rates. We will also show results separately for IVR digits; this is a 10.5 hour test set with average utterance duration of 5.5 seconds, and the baseline (unadapted) Word Error Rate is 0.85%.

B. Details of CMLLR implementation

Our CMLLR statistics accumulation has an unusual feature: to speed up the search, our decoder uses context-independent (CI) phone models derived from merging the context dependent models, and these CI models are also used in parallel with the context-dependent models when accumulating CMLLR statistics. We assume that to a first approximation, this is equivalent to doubling the statistics; our total count $\beta^{(s)}$ will be twice what one would normally expect. For better comparison with more normal setups, in the experimental section we have adjusted all figures that refer indirectly to data counts (e.g. proportionality constants and minimum data counts), by a factor of two to remove this effect. Therefore, this feature of our system can safely be ignored when reading the rest of this section.

All our adaptation is done on a per utterance basis, i.e. we assume each utterance corresponds to a unique speaker. We apply the CMLLR estimation (in both the baselines and our technique) in a segment-wise online fashion. That is, we divide the utterances up into segments and the CMLLR estimation for each segment only sees the statistics for that segment and preceding segments. For the statistics accumulation, the CMLLR transform estimated for the previous segment is used for within-phone alignment (the phone-level segmentation is fixed by the first pass decoder). The segmenter is tuned to give about three segments per utterance. Assuming equal length segments, for an utterance of length T the number of frames used to estimate the CMLLR would be $T/3$, $2T/3$ and T for the three segments respectively. The average is $2/3T$, so we can approximate the effect by a correction factor of $2/3$ on our utterance lengths. However, the utterance lengths we quote are the real lengths.

All of our CMLLR implementations are set to only adapt if statistics corresponding to more than 1.5 seconds of speech are available. In our shortest subset (left column of Table I), only 1% of utterances are shorter than this, so this will have only a small effect on our results.

We only do one iteration of E-M in the sense that for each segment we only do the statistics accumulation and update once, but because of the on-line aspect the effect is somewhat similar to multiple iterations. For all experiments, including our own method, we used ten iterations of optimization in the update phase.

C. Baselines

The baseline adaptation strategies we compare against are "full CMLLR", which refers to CMLLR estimated in the standard way, "block-diagonal CMLLR", in which the matrix $\mathbf{A}$ is block diagonal with three equal-sized blocks, and "diagonal CMLLR" where $\mathbf{A}$ is diagonal. The use of the block-diagonal structure originated with systems that use MFCC+Δ+$\Delta\Delta$ coefficients, but with HLDA features there is no special meaning to the three blocks. We also compare with fMAPLR [3], for which we use a separate full-covariance Gaussian for each row; we extend it by putting a scale on the log prior term and tuning the scale. For the IVR system our fMAPLR is applied to block diagonal CMLLR with 3 blocks,

and for the EVM system we use a full matrix; this choice was made to optimize WER under fMAPLR estimation. The prior used for fMAPLR in the 3-block case is a Gaussian modeling the non-zero part of each row, which is a vector of dimension $D/3 + 1$.

Because the estimation of the prior in fMAPLR does not seem to work well with short training utterances, we trained the fMAPLR prior only on relatively long utterances. For the IVR system we used 4000 utterances containing digits, since these are well matched to the test set and are longer than the other types of test data. These utterances were not in the test set. We used the same utterances to train the basis matrices for our method. For the EVM system, we used the 130 hours of voicemail training data to train the fMAPLR prior and the basis matrices. We always treat each utterance as a separate speaker. For training the fMAPLR prior and the basis, we used the same feature extraction (with on-line CMN) that is used in test time.

D. Technical issues and tuning

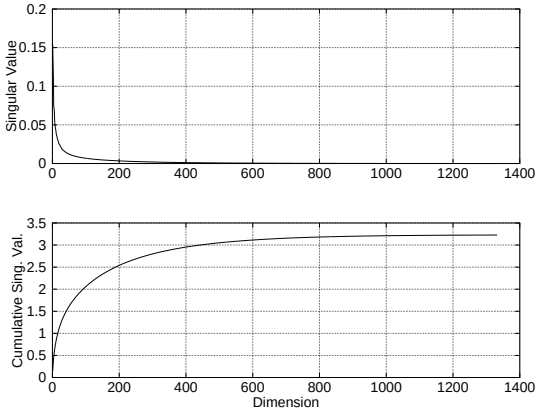

Fig. 2. Singular values for basis training: IVR

Figure 2 displays the sorted singular values of the matrix $\hat{\mathbf{M}}$ that was accumulated for training the basis for the IVR system. The singular values were divided by twice the number of frames in these utterances to get a per-frame log likelihood quantity; the factor of $\frac{1}{2}$ comes from Equation (15). We display the absolute and cumulative singular values. From the cumulative plot, it is clear that about half the "energy" is in the first 75 or so singular values. The total cumulative value (about 3.25) is roughly comparable to the amount of log-likelihood improvement we would get on these utterances without any dimensionality reduction. A similar plot for the EVM data (not shown) has even more energy concentrated in the first singular values.

Figure 3 shows the convergence of the iterative update phase for three randomly chosen utterances, against iteration q. We show this for both the baseline and conjugate-gradient updates. It can be seen that the conjugate method converges faster than the baseline method, with about the same auxiliary function after 5 iterations that the baseline reaches after 10. Word Error Rates were also measured for the baseline and

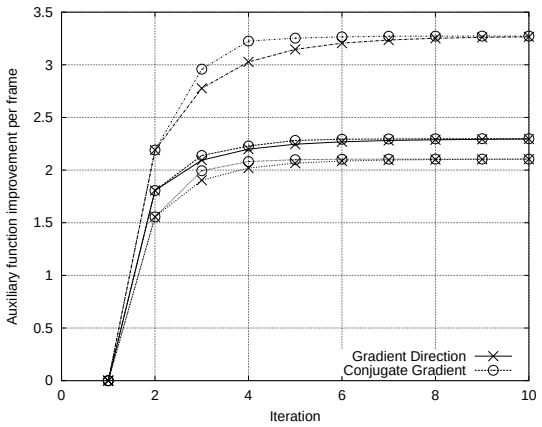

Fig. 3. Convergence of update phase: three randomly chosen utterances

Conjugate Gradient methods, for various numbers of iterations, and these broadly reflect the auxiliary function behavior visible in the graph. Results in this paper are with the baseline (not Conjugate Gradient) method with $Q = 10$ iterations, and $R = 3$ inner iterations.

TABLE II
TIME TAKEN IN UPDATE PHASE (%CPU)

	Proposed Method	Baseline		
		Full	3-Block	Diagonal
IVR (Digits)	5.75%	35.1%	1.84%	0.45%
EVM (Voicemail)	0.10%	0.68%	0.03%	0.01%

Table II shows the time taken in the update phase of the different CMLLR update methods, as a percentage of total CPU (the time taken for accumulation is the same for all methods in our code, since we always accumulate the same statistics). We do not show the fMAPLR based methods since the time taken is the same as the corresponding baseline CMLLR update. The time taken is a much larger percentage of total CPU for the digits task than for voicemail, because of differences in search speed and utterance length. The average length of the digits and voicemail utterances we used for this test was 5.8 seconds and 35.4 seconds respectively. For these amounts of data our method is faster in test time than the traditional CMLLR update. We expect our method to get closer to the speed of the traditional approach as the amount of adaptation data becomes very large, since both methods are $O(D^4)$ per iteration in this case.

Figure 4 relates to tuning the proportionality factor η used to control the per-speaker basis size $B(s)$, on IVR digits. The best factor is $\eta = 0.2$. Figure 5 shows the same for EVM, but this time we compare with tuning a fixed basis size B that does not depend on the utterance length. Again the best factor is $\eta = 0.2$, and this is better than any fixed basis size. The fact that the same value of η works well in domains with very different average utterance lengths confirms to us that our linear rule of (32) is reasonable. On IVR we did not see any clear difference between choosing a fixed versus adaptive basis size, probably because the range of utterance durations is much smaller. Figure 6 investigates tuning the scale on the log

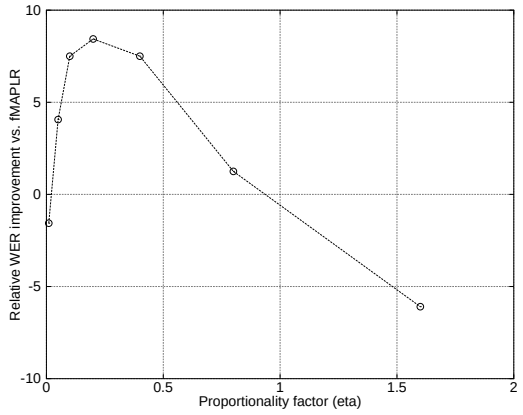

Fig. 4. Tuning proportionality factor η: IVR digits

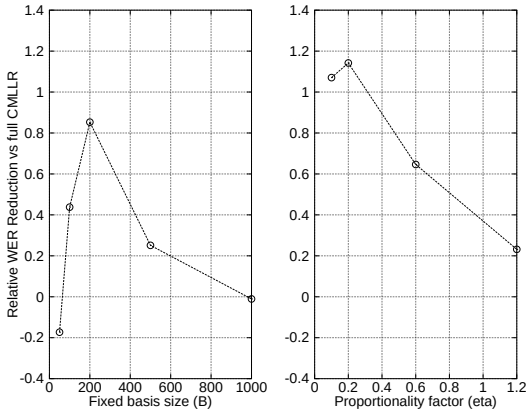

Fig. 5. Adaptive vs. fixed basis size: EVM

prior in fMAPLR, for EVM. The optimal scale appears to be about 2.5, and we used this for EVM (for IVR, the optimized scale was 5).

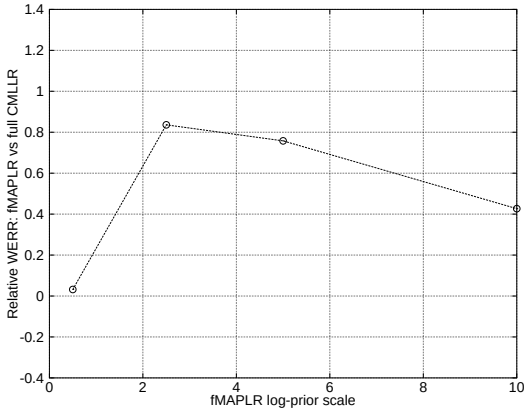

Fig. 6. fMAPLR improvement vs. prior scaling factor: EVM

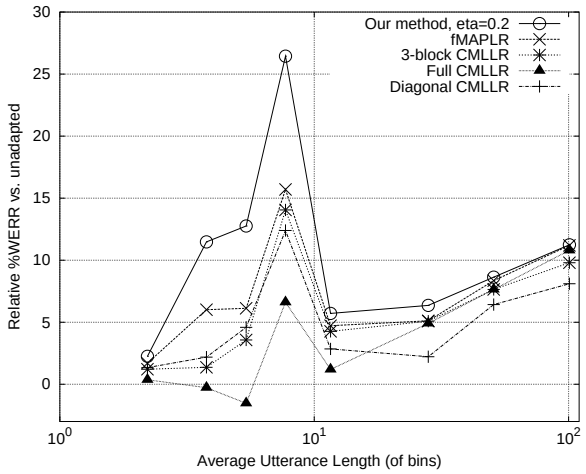

Fig. 7. WER improvement from adaptation vs. utterance duration

E. Main results

Our main results are in Figure 7. This shows the relative WER improvements of our technique versus no adaptation, along with the various baselines, for the duration bins described in Table I. It can be seen that our method is generally best, followed by fMAPLR, followed by standard CMLLR; depending on the utterance length, either full, block-diagonal or diagonal CMLLR was best. The points on the left half of the graph are from IVR, and those on the right half are from EVM. Note that in this graph, the fMAPLR line does not represent a single technique: on the left half, it is block diagonal, and on the right half it is the full transform. However, this does roughly represent "the best we can do with fMAPLR". Note that the line with full CMLLR (triangles) substantiates our claim in the introduction that CMLLR does not give improvements below about five seconds of data.

Figure 7 is a little hard to interpret because the various buckets are at very different absolute WERs (see Table I, last row), and the relative improvements are higher at very low WER. We display the same results in Figure 8, normalized so that the improvement from our technique is 100%. The trend becomes rather clearer here. Until about 20 seconds of adaptation data, our technique gives a substantial improvement over fMAPLR and the other baselines, and after that the differences are small. In addition, referring back to the Figure 7, for less than about 3 seconds of data none of the adaptation methods give very large improvements. The improvement from implementing our technique versus CMLLR is greatest between about 5 and 15 seconds of speech. For less than 20 seconds of speech our technique gives more than twice the improvement of fMAPLR, taking CMLLR as the baseline. For 20 seconds or more, the differences are small.

As mentioned in Section VI-B, the online manner in which the CMLLR transforms are estimated can be approximated by a factor of $2/3$ on the utterance length, so we should modify our statements above. With this correction, we anticipate that the greatest improvement from our method will be between about 3 and 10 seconds of adaptation data in a

batch-processing framework.

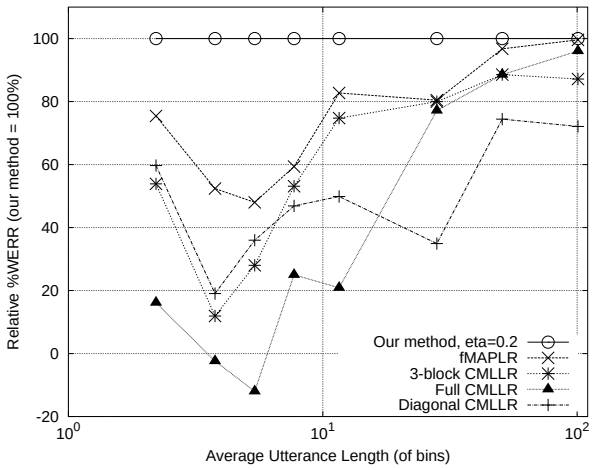

Fig. 8. WER reduction of baselines, relative to our method

VII. Conclusions

We have introduced a new method for estimating Constrained MLLR transforms from a limited amount of adaptation data. Our method is used in conjunction with a new update method for CMLLR matrices. This update method is faster than the traditional approach for the utterance lengths we tested on. We show that our method gives substantial WER improvements over CMLLR and fMAPLR for less than about 13 seconds of adaptation data. For less than about 2 seconds of data, adaptation with CMLLR does not help much even with our method. We anticipate that the benefits of our approach will be greatest in scenarios where there are between about 3 and 10 seconds of data to adapt on. Our approach appears to be at least as good as the baselines we tested for any amount of adaptation data.

References

[1] M. J. F. Gales and P. C. Woodland, "Mean and Variance Adaptation Within the MLLR Framework," *Computer Speech and Language*, vol. 10, pp. 249–264, 1996.

[2] M. Gales, "Maximum Likelihood Linear Transformations for HMM-based Speech Recognition," *Computer Speech and Language*, vol. 12, pp. 75–98, 1997.

[3] X. Lei, J. Hamaker, and X. He, "Robust Feature Space Adaptation for Telephony Speech Recognition," in *ICSLP*, 2006.

[4] J. Huang, E. Marcheret, and K. Visweswariah, "Rapid Feature Space Speaker Adaptation for Multi-Stream HMM-Based Audio-Visual Speech Recognition," *Multimedia and Expo, IEEE International Conference on*, pp. 338–341, 2005.

[5] Y. Li, H. Erdogan, Y. Gao, and E. Marcheret, "Incremental On-line Feature Space MLLR Adaptation for Telephony Speech Recognition," in *Interspeech*, 2002, pp. 1417–1420.

[6] S. Young *et al.*, *The HTK Book (for version 3.4)*. Cambridge University Engineering Department, 2009.

[7] K. Visweswariah, V. Goel, and R. Gopinath, "Structuring Linear Transforms for Adaptation Using Training Time Information," in *ICASSP*, 2002.

[8] D. Povey, L. Burget, M. Agarwal, P. Akyazi, K. Feng, A. Ghoshal, O. Glembek, N. K. Goel, M. Karafiát, A. Rastrow, R. C. Rose, P. Schwarz, and S. Thomas, "Subspace Gaussian Mixture Models for Speech Recognition," in *ICASSP*, 2010.

[9] D. Povey, "A Tutorial Introduction to Subspace Gaussian Mixture Models for Speech Recognition," Microsoft Research, Tech. Rep. MSR-TR-2009-111, 2009.

[10] A. Ghoshal, D. Povey, M. Agarwal, P. Akyazi, L. Burget, K. Feng, O. Glembek, N. Goel, M. Karafiát, A. Rastrow, R. C. Rose, P. Schwarz, and S. Thomas, "A Novel Estimation of Feature-space MLLR for Full Covariance Models," in *ICASSP*, 2010.

[11] J. R. Shewchuk, "An introduction to the conjugate gradient method without the agonizing pain," Carnegie Mellon University, Tech. Rep., 1994.

[12] B.-H. Juang, W. Chou, and C.-H. Lee, "Minimum classification error rate methods for speech recognition," *IEEE Trans. on Speech and Audio Processing*, vol. 5, pp. 257–265, 1997.

[13] P. Woodland and D. Povey, "Large Scale MMIE Training For Conversational Telephone Speech Recognition," in *Proc. Speech Transcription Workshop*, 2000.