A Basis Representation of Constrained MLLR Transforms for Robust Adaptation

Daniel Povey, Kaisheng Yao

Microsoft, One Microsoft Way, Redmond, WA 98052, USA

Abstract

Constrained Maximum Likelihood Linear Regression (CMLLR) is a speaker adaptation method for speech recognition that can be realized as a feature-space transformation. In its original form it does not work well when the amount of speech available for adaptation is less than about five seconds, because of the difficulty of robustly estimating the parameters of the transformation matrix. In this paper we describe a basis representation of the CMLLR transformation matrix, in which the variation between speakers is concentrated in the leading coefficients. When adapting to a speaker, we can select a variable number of coefficients to estimate depending on the amount of adaptation data available, and assign a zero value to the remaining coefficients. We obtain improved performance when the amount of adaptation data is limited, while retaining the same asymptotic performance as conventional CMLLR. We demonstrate that our method performs better than the popular existing approaches, and is more efficient than conventional CMLLR estimation.

Keywords: Speech Recognition, Speaker Adaptation

1. Introduction

Constrained Maximum Likelihood Linear Regression (CMLLR) [3, 5, 4] is a speaker adaptation method for speech recognition. Although originally presented as a model-space transformation, it can be described more com-

Email addresses: dpovey@microsoft.com (Daniel Povey), kaisheny@microsoft.com (Kaisheng Yao)

pactly as an affine transformation of the speech feature vectors, of the form:

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

where $\mathbf{x} \in \mathbb{R}^D$ is the feature vector, and $\mathbf{A}^{(s)}$ and $\mathbf{b}^{(s)}$ are transformation parameters specific to speaker s. We will write this here in the more convenient form

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

where $\mathbf{x}^{+} = \left[\mathbf{x}^T\ ,\ 1\right]^T$, and $\mathbf{W}^{(s)} = \left[\mathbf{A}^{(s)}\ ;\ \mathbf{b}^{(s)}\right]$. Because it can be represented as a feature-space transform, CMLLR is also known as feature-space MLLR (fMLLR).

CMLLR is typically estimated in a Maximum Likelihood fashion based either on known transcripts of a speaker's utterances (so-called supervised adaptation) or on the text output from a speech recognition system (unsupervised adaptation). An EM algorithm is normally used to estimate $\mathbf{W}^{(s)}$; we will summarize this in Section 2. However, when less than about five seconds of adaptation data is available this does not always lead to improvements in Word Error Rate (WER). This is presumably because the estimate of $\mathbf{W}^{(s)}$ is too noisy and does not generalize well to test data.

Various methods have been proposed to improve CMLLR estimation for limited adaptation data; we discuss these further in Section 3. These include the use of block-diagonal and diagonal forms of the matrix $\mathbf{A}$ [4], the use of Bayesian priors ("fMAPLR") [11, 8], and representing $\mathbf{W}$ in a smaller dimension using a basis [16, 15]. In [15] it was found that it is important to train such a basis using a Maximum Likelihood criterion (rather than the PCA scheme used in [16]).

We describe the key ideas behind our approach in Section 4. Our method is a basis method; the general idea is the same as [16] but we have solved a number of problems to make it practical and efficient. In particular, we are able to use a PCA-like algorithm to train the basis in a Maximum Likelihood fashion. This differs from the method used in [16] by the use of preconditioning, which enables it to more closely approximate Maximum Likelihood. In addition we allow the number of basis elements used to vary per speaker. This enables us to improve performance across all lengths of adaptation data. The preconditioning also helps us when we need to optimize the speaker-specific coefficients that describe the adaptation matrix, because it approximately decorrelates the coefficients in the objective function and this makes the optimization of the speaker-specific coefficients converge faster.

We describe our approach as an algorithm in Section 5. This includes a pre-computation phase in which we use training data to compute an appropriate set of basis matrices $\{\mathbf{W}_n, 1 \leq n \leq D(D+1)\}$, and a test-time computation in which we compute the speaker-specific adaptation matrix $\mathbf{W}^{(s)}$ as a weighted sum of a subset of the basis matrices $\{\mathbf{W}_n, 1 \leq n \leq N(s)\}$, where $N(s)$ is the number of basis elements we choose to keep. Rather than introduce a complicated method based on an information criterion, we simply set $N(s)$ proportional to the number of frames of adaptation data. The idea is that the more data we have, the more parameters we can afford to estimate. We present two iterative methods of optimizing the speaker-specific coefficients in test time. In both cases, on each iteration we choose a direction to search in and then perform a line search in that direction. Our simpler method searches in the gradient direction; the alternative method uses conjugate gradients.

In Section 6 we describe our experiments. We compare our method with standard CMLLR, diagonal and block-diagonal CMLLR, and fMAPLR. Our experiments show a clear advantage of our technique over the baselines we tested. We conclude in Section 7. We believe that our method may be usefully deployed wherever CMLLR adaptation is used. This is because it gives good WER improvements when the amount of speaker adaptation data is small, while giving the same performance as the baseline for large amounts of adaptation data. In this respect it is quite similar to fMAPLR. However, our experiments show that our method gives about twice the improvement of fMAPLR, taking standard CMLLR as the baseline. In addition it is more efficient in the update phase than either CMLLR or fMAPLR.

We note that in this paper we have devoted a lot of space to a detailed description of the steps of our algorithm, and relatively little to its derivation, or to justification of the exact approximations we made. The main aim of our paper is to facilitate easy reproduction; there are many aspects of our method which could have been done differently, but time did not permit an exhaustive investigation of all the possibilities. We have also limited our experimental baselines to normal CMLLR, its block diagonal forms, and fMAPLR. We have not implemented the methods described in [16, 15] which are closely related to the methods we describe here. One reason is that those methods were not described in a level of detail that allows for very easy reproduction. Certainly we cannot claim to have fully explored the solution space for robust CMLLR parameter estimation. Our objectives are to communicate the key ideas behind our method, to describe it in detail sufficient to allow others to

reproduce it, and to demonstrate that it is better than the commonly used baselines.

2. Constrained MLLR

CMLLR was originally described in [3] as a model-space transformation. Using a dash to distinguish the quantities used in the original paper, the model-space transformation is from generic to speaker-specific parameters:

$$\hat{\boldsymbol{\mu}}_{jm} = \mathbf{A}'\boldsymbol{\mu}_{jm} + \mathbf{b}' \quad (3)$$

$$\hat{\boldsymbol{\Sigma}}_{jm} = \mathbf{A}'\boldsymbol{\Sigma}_{jm}\mathbf{A}'^{T}. \quad (4)$$

Making the substitutions $\mathbf{A} = \mathbf{A}'^{-1}$ and $\mathbf{b} = -\mathbf{A}'^{-1}\mathbf{b}'$ to obtain the opposite transformation from speaker-specific to generic features, we obtain the form given in (1). When writing it as a feature transformation, we also have to include a log determinant term $\log|\det \mathbf{A}|$ in the likelihood function, which arises from the following equality:

$$\begin{aligned} & \log \mathcal{N}(\mathbf{x}\,;\ \hat{\boldsymbol{\mu}}_{jm}, \hat{\boldsymbol{\Sigma}}_{jm}) \\ = \ & \log|\det \mathbf{A}| + \log \mathcal{N}\left(\mathbf{A}\mathbf{x} + \mathbf{b}\,;\ \mu_{jm}, \boldsymbol{\Sigma}_{jm}\right). \end{aligned} \quad (5)$$

It is straightforward to show this by writing out the Gaussian likelihood functions.

As mentioned, the conventional approach for estimating CMLLR transform parameters is an Expectation-Maximization (E-M) process. Normally only one or two iterations of E-M are required since it converges very fast. We assume we are given input features $\mathbf{x}(t)$ for speaker s; t is the time index. Sums over time in equations below are assumed to be over the frames for the given speaker s. In the expectation phase, we first compute the Gaussian level posteriors $\gamma_{jm}(t)$ for states j and Gaussian mixtures m. This would typically be done by forward-backward or Viterbi alignment of a HMM corresponding to a known or hypothesized text string. Writing out the auxiliary function derived in the normal way using Jensen's inequality, we have:

$$\mathcal{Q}(\mathbf{W}^{(s)}) = \sum_{j,m,t} \gamma_{j,m,t} \mathcal{N}(\mathbf{x}(t)\,;\ \hat{\boldsymbol{\mu}}_{jm}, \hat{\boldsymbol{\Sigma}}_{jm}). \quad (6)$$

By using (5) to express this as a feature-space transform, rearranging expressions to move the sums over j and m into equations for sufficient statistics,

and discarding terms constant in $\mathbf{W}^{(s)}$, we obtain the following auxiliary function [4]. For brevity we omit the superscript $\cdot^{(s)}$:

$$\mathcal{Q}(\mathbf{W}) = \beta \log|\det \mathbf{A}| + \operatorname{tr}(\mathbf{K}^T\mathbf{W}) + \sum_i \mathbf{r}_i^T \mathbf{G}_i \mathbf{r}_i, \tag{7}$$

where the sufficient statistics are $\mathbf{K} \in \mathbb{R}^{D\times D+1}$, $\beta \in \mathbb{R}$, and $\left\{\mathbf{G}_i \in \mathbb{R}^{D+1\times D+1}, 1 \le i \le D\right\}$, and the column vector $\mathbf{r}_i$ corresponds to the i'th row of $\mathbf{W}$. The sufficient statistics are defined as:

$$\mathbf{G}_i = \sum_t \sum_{j,m} \gamma_{jm}(t) \frac{1}{\sigma_{jm}^2(i)} \mathbf{x}(t)^+ \mathbf{x}(t)^{+T} \tag{8}$$

$$\mathbf{K} = \sum_t \sum_{j,m} \gamma_{jm}(t) \boldsymbol{\Sigma}_{jm}^{-1} \boldsymbol{\mu}_{jm} \mathbf{x}(t)^{+T} \tag{9}$$

$$\beta = \sum_t 1, \tag{10}$$

where (10) is just the number of frames for the speaker. These formulas rely on the assumption that the covariance matrices $\boldsymbol{\Sigma}_{jm}$ are diagonal; in the general case we would have matrices $\mathbf{G}_{ij}$, with $1 \le i, j \le D$ [14], or we could alternatively store per-Gaussian statistics.

Equation (7) is still not very easy to maximize, due to the presence of the log determinant term. The standard approach [5, 4] uses the fact that it is possible to solve it for a particular row $\mathbf{r}_i$ of $\mathbf{W}$ given fixed values for all the other rows. If this is done for each row i in turn, and the process is repeated, it converges reasonably fast: only about 10 or 20 repetitions are typically necessary.

3. Prior work on Robust CMLLR estimation

In our experience CMLLR gives little or no improvement when less than about 5 seconds of adaptation data is available (e.g. see our experiments in Section 6). Various methods have been proposed to adapt on small amounts of data. In [4] it was mentioned that diagonal or block-diagonal structures for $\mathbf{A}$ can be used to reduce the number of parameters to estimate. Because of their simplicity such methods are frequently used, and we use them as baselines here.

3.1. Bayesian techniques

Bayesian techniques have been investigated in [11] and [8], both under the name fMAPLR. The basic idea is to use the Maximum A Posteriori rule to choose the parameter, given a suitable prior, i.e. to maximize:

$$p(\mathbf{W}|\mathcal{X}) \propto p(\mathcal{X}|\mathbf{W})p(\mathbf{W}), \tag{11}$$

where $\mathcal{X}$ is the speech data, $p(\mathcal{X}|\mathbf{W})$ is the data likelihood (represented by (7)), and $p(\mathbf{W})$ is the prior likelihood. The two papers both used Gaussian priors over $p(\mathbf{W})$ (viewing the matrix as the vector of its concatenated rows) but they used different ways of compactly representing the prior covariance which is a matrix of dimension $D(D+1) \times D(D+1)$. In [8] a factor-analyzed form was used (i.e. the covariance matrix was a diagonal matrix plus the outer product of a rectangular matrix), and in [11] a diagonal matrix was used. In both cases the Maximum Likelihood estimates of the matrices $\mathbf{W}^{(s)}$ for a set of speakers were used as training data for the prior parameters (a simple "empirical Bayes" approach). The version of fMAPLR we used as a baseline here is a slight generalization of [11], in which we give the covariance of the prior a block-diagonal structure (one block for each row of $\mathbf{W}$) and also introduced a scaling factor on the log-prior term, which we tuned to optimize WER. We trained the prior only on speakers with a relatively large amount of adaptation data, as we found this worked best.

3.2. Basis methods

A basis representation of the CMLLR matrix was described in [16]. The idea is to represent $\mathbf{W}$ as a sum over basis matrices:

$$\mathbf{W}^{(s)} = \sum_{n=1}^{N} d_n^{(s)} \mathbf{W}_n, \tag{12}$$

where N is some basis size decided in advance with $1 \leq N < D(D+1)$ (e.g. N=200), $\mathbf{W}_n$ are the basis matrices, and $d_n^{(s)}$ are speaker-specific coefficients. This improved WER for small amounts of adaptation data, but the technique ultimately degraded performance relative to CMLLR as the amount of adaptation data became larger, due to the fixed basis size. In [15] the same idea was pursued further, and it was found that for best performance it was important to train the matrices $\mathbf{W}_n$ in a Maximum Likelihood fashion. For

this reason we have focused on Maximum Likelihood solutions for the basis matrices in our own work (the key difference is that our approach does not require us to decide the basis size in advance, so that we can choose an appropriate basis size for each speaker).

We note that basis representations of Maximum Likelihood Linear Regression (MLLR) have been proposed, such as Eigen-MLLR [2]. The problem with such approaches is that they are difficult to make very efficient, since they require the model's means to be transformed for each new speaker. This will typically dominate the computation time in cases where the amount of adaptation data is small. Another well-known method for fast adaptation is Eigenvoices [10], but methods of that type require a very large number of parameters to be learned in training time.

4. Key ideas of our approach

The basis representation we use is very similar to (12), except with an offset term and (more importantly) a basis size that varies per speaker:

$$\mathbf{W}^{(s)} = \mathbf{W}_0 + \sum_{n=1}^{N(s)} d_n^{(s)} \mathbf{W}_n, \tag{13}$$

where $0 \leq N(s) \leq D(D{+}1)$ and

$$\mathbf{W}_0 = [\mathbf{I}\,;\,\mathbf{0}]\,. \tag{14}$$

In our work we just set $N(s)$ to be proportional to the the amount of adaptation data (but not exceeding $D(D{+}1)$ which is the number of parameters in $\mathbf{W}$). Note that while this is a model selection problem, we have not compared against standard model selection methods such as the Bayesian Information Criterion (BIC) [12], or the Aikake Information Criterion (AIC) [1]. This is because, in our experience, for these kinds of problems, such methods do not show a clear advantage over simple count-based heuristics; in addition, they are much more complex to implement. Our approach to model selection was chosen for simplicity and speed.

Probably the key aspects of our work that distinguish it from [16, 15] are the use of a varying number of basis elements, and our approach to computing the basis matrices $\mathbf{W}_n$. This approach approximates Maximum Likelihood but is still efficient and is applicable when the basis size is to

be decided in test time. The preconditioning we use to accomplish this has the useful side effect that it speeds up the algorithms we use to learn the parameters $d_n^{(s)}$ in test time. Some of the ideas used here are derived from prior work described in [6], which describes an efficient method of updating the CMLLR transformation for a differently structured GMM-based system with full covariances.

Below we discuss the ideas behind various aspects of our algorithm. In Section 4.1 we discuss the preconditioning; in Section 4.2 we describe how we compute the basis matrices $\mathbf{W}_n$; in Section 4.3 we describe how we compute the coefficients $d_n^{(s)}$, and in Section 4.4 we describe our formula to compute $N(s)$.

4.1. Preconditioning

In many of our computations it is easiest to think of $\mathbf{W}$ as a vector rather than a matrix, so we define

$$\mathbf{w} = \mathrm{vec}\,(\mathbf{W}^T), \tag{15}$$

where the vec operator stacks the columns, so with the transpose, $\mathbf{w}$ is a row stack of $\mathbf{W}$; the transpose is useful later on. We will implicitly make use of (15), by making it apply to pairs $\mathbf{w}$ and $\mathbf{W}$ whenever they share the same subscripts, superscripts and other modifiers. Consider a second-order Taylor expansion of (7), taken around $\mathbf{w} = \mathbf{w}_0$. We write $\Delta\mathbf{w}$ for $(\mathbf{w}-\mathbf{w}_0)$. The approximation is written as follows (ignoring constant offsets):

$$\mathcal{Q}^{(s)}(\mathbf{w}) \simeq (\Delta\mathbf{w})^T\mathbf{p}^{(s)} - \tfrac{1}{2}(\Delta\mathbf{w})^T\mathbf{H}^{(s)}(\Delta\mathbf{w}). \tag{16}$$

The quantities $\mathbf{p}^{(s)}$ and $\mathbf{H}^{(s)}$ may be computed from the statistics $\mathbf{G}_i^{(s)}$, $\mathbf{K}^{(s)}$ and $\beta^{(s)}$. The idea is to precondition via a change of variables, such that when written in the new variable, the matrix corresponding to $\mathbf{H}^{(s)}$ has good condition number (i.e. it is close to the unit matrix times a constant). This is quite straightforward to do. First we define

$$\mathbf{H} = \frac{1}{\sum_s \beta^{(s)}} \sum_s \mathbf{H}^{(s)}, \tag{17}$$

so $\mathbf{H}$ is the average value of the $\mathbf{H}^{(s)}$ term (normalized by the number of frames). Note that this is a $D(D{+}1) \times D(D{+}1)$ matrix. We do the Cholesky decomposition

$$\mathbf{H} = \mathbf{C}\mathbf{C}^T, \tag{18}$$

with $\mathbf{C}$ a lower triangular matrix. We then perform a change of variables by defining

$$\hat{\mathbf{w}} = \mathbf{C}^T \mathbf{w}. \tag{19}$$

Thus we can rewrite (16) as

$$\hat{\mathcal{Q}}^{(s)}(\hat{\mathbf{w}}) = (\Delta\hat{\mathbf{w}})^T \hat{\mathbf{p}}^{(s)} - \tfrac{1}{2}(\Delta\hat{\mathbf{w}})^T \hat{\mathbf{H}}^{(s)} (\Delta\hat{\mathbf{w}}), \tag{20}$$

(where $\Delta\hat{\mathbf{w}} \equiv \hat{\mathbf{w}} - \hat{\mathbf{w}}_0$ and $\hat{\mathbf{w}}_0 = \mathbf{C}^T \mathbf{w}_0$), with

$$\hat{\mathbf{H}}^{(s)} = \mathbf{C}^{-1} \mathbf{H}^{(s)} \mathbf{C}^{-T} \tag{21}$$

$$\hat{\mathbf{p}}^{(s)} = \mathbf{C}^{-1} \mathbf{p}^{(s)}. \tag{22}$$

Thus the new quadratic term averages to the unit matrix (i.e. $\hat{\mathbf{H}} = \mathbf{I}$).

4.2. Basis computation

The basis computation also relies on the Taylor approximation of (16). We additionally make the assumption that $\mathbf{H}^{(s)} \simeq \beta^{(s)} \mathbf{H}$ (equivalent to $\hat{\mathbf{H}}^{(s)} \simeq \beta^{(s)} \mathbf{I}$). This is reasonable as long as all the speakers are sufficiently similar. This assumption is necessary in order to reduce the problem to a Principal Components Analysis (PCA) problem, which is tractable. These approximations may seem quite crude, but the key is that they make the basis computation fast. In [15] more exact methods were considered, but they are not applicable if the basis size is to be decided in test time.

Under these assumptions it is easy to compute a Maximum Likelihood solution for the basis matrices. The way we formulate the problem is to ask for a sequence $\{\mathbf{W}_n,\ 1 \leq n \leq D(D{+}1)\}$, such that whatever basis size $1 \leq N \leq D(D{+}1)$ we choose, the training data likelihood (subject to our assumptions and approximations) is maximized if we choose the first N matrices as our basis. Let us consider the problem in its vector form (i.e. in terms of $\mathbf{w}_n$). In order to ensure good conditioning of the auxiliary function (7) when written in terms of the coefficients d_n, we insist that the transformed form of the vectors (i.e. $\hat{\mathbf{w}}_n$) form an orthonormal set. It is not hard to show that this is without loss of generality. We will consider some arbitrary but fixed basis size $1 \leq N \leq D(D{+}1)$, and write $\hat{\mathbf{w}}$ as a sum over basis elements, i.e.:

$$\hat{\mathbf{w}} = \hat{\mathbf{w}}_0 + \sum_{n=1}^{N} d_n \hat{\mathbf{w}}_n. \tag{23}$$

Here, $\hat{\mathbf{w}}_0$ is the transformed version of $\mathbf{W}_0$ as in (14). We have limited $\Delta\hat{\mathbf{w}} \equiv \hat{\mathbf{w}} - \hat{\mathbf{w}}_0$ to the subspace spanned by the vectors $\hat{\mathbf{w}}_n$. We will now describe how we compute the the basis elements $\hat{\mathbf{w}}_n$ in such a way that they approximately maximize the objective function for all basis sizes simultaneously.

We first write down the auxiliary function in $\hat{\mathbf{w}}$ without yet applying the subspace constraint of (23). Rewriting (20) using $\hat{\mathbf{H}}^{(s)} \simeq \beta^{(s)}\mathbf{I}$,

$$\hat{\mathcal{Q}}^{(s)}(\hat{\mathbf{w}}) \simeq (\Delta\hat{\mathbf{w}})^T\hat{\mathbf{p}}^{(s)} - \tfrac{1}{2}\beta^{(s)}(\Delta\hat{\mathbf{w}})^T(\Delta\hat{\mathbf{w}}). \tag{24}$$

It is easy to see that this is maximized by $\Delta\hat{\mathbf{w}}^{(s)} = \frac{1}{\beta^{(s)}}\hat{\mathbf{p}}^{(s)}$, and that the corresponding auxiliary function value is $\frac{1}{2\beta^{(s)}}\ \hat{\mathbf{p}}^{(s)\,T}\,\hat{\mathbf{p}}^{(s)}$. Defining

$$\hat{\mathbf{M}} = \sum_s \frac{1}{\beta^{(s)}}\hat{\mathbf{p}}^{(s)}\ \hat{\mathbf{p}}^{(s)\,T}, \tag{25}$$

we may write the total auxiliary function, summed over all speakers, as $\mathrm{tr}\,(\hat{\mathbf{M}})/2$. Suppose we write $\mathbf{X}_N = [\hat{\mathbf{w}}_1 \ldots \hat{\mathbf{w}}_N]$ to represent a basis of size N (recall that the $\hat{\mathbf{w}}_n$ are orthonormal). It is not hard to show that when limiting $\hat{\mathbf{w}}$ to the form (23), the corresponding total auxiliary function value is $\mathrm{tr}\,(\mathbf{X}_N^T\hat{\mathbf{M}}\mathbf{X}_N)/2$, and that the basis $\mathbf{X}_N$ that maximizes this can be obtained by doing an eigenvalue decomposition on $\hat{\mathbf{M}}$ and letting $\hat{\mathbf{w}}_n$ be the n'th eigenvector of $\hat{\mathbf{M}}$ (ordered from largest to smallest eigenvalue), so that whatever basis size N we choose, $\mathbf{X}_N$ always contains the top N eigenvectors.

Thus, with the help of the preconditioning and some additional approximations, we have reduced the difficult Maximum Likelihood problem considered in [15] to a much easier problem similar to Principal Components Analysis (PCA). After computing the vectors $\hat{\mathbf{w}}_n$ we can reverse the co-ordinate changes and un-stack the matrix columns to obtain the set $\{\mathbf{W}_n, 1 \leq n \leq D(D{+}1)\}$.

In order to get to this point, we have had to make the simplifying assumption that the Hessians $\mathbf{H}^{(s)}$ for speakers s are all the same, up to a constant factor. In fact, later on we make a further approximation when computing the average Hessian, by assuming model correctness (this is for convenience; we could easily compute the exact average Hessian with an extra pass over the training data). The reason we feel confident in making these approximations is that the Hessian is just going into a PCA type of computation where we compute an ordered list of eigenvectors. These types of computation are not very sensitive to approximations in the matrix we are decomposing, because the more important directions of variation will still be represented by

the leading eigenvectors; the effect of the approximations will generally be to mix up the eigenvectors with similar eigenvalues (and hence similar rank ordering), which does not have much effect on the final objective function.

4.3. Test time computation

In test time, the optimization problem is as follows: we are given the speaker-specific statistics $\mathbf{K}^{(s)}$, $\mathbf{G}_i^{(s)}$ and $\beta^{(s)}$, the basis $\{\mathbf{W}_n, 1 \leq n \leq D(D+1)\}$ and a basis size $0 \leq N(s) \leq D(D+1)$ and we need to estimate the speaker-specific coefficients $d_n^{(s)}$ of (13).

Our update is iterative. On each iteration $1 \leq q \leq Q$ (where e.g. $Q = 10$ is a typical number of iterations), we compute the gradient of the auxiliary function (7) w.r.t. the coefficients $\{d_n^{(s)}, 1 \leq n \leq N(s)\}$ and select a search direction. In the basic version of our method the search direction is just the gradient direction; we also describe a modification based on the conjugate gradient method. Then we do a line search in that direction; our line search is iterative and based on Newton's method (in one dimension). We will show that both our basic method and the conjugate gradient modification converge very fast.

4.4. Choosing the basis size

The number of coefficients $d_n^{(s)}$ to use was determined by the following simple formula:

$$N(s) = \min(\lfloor \eta \beta^{(s)} \rfloor, D(D+1)), \tag{26}$$

where η is a constant set by hand (e.g. $\eta = 0.2$) that determines how many parameters to add for each new frame of data. The basic justification for this formula is that the more data we have, the more parameters we can robustly estimate, and (26) is the simplest formula that captures this heuristic ($D(D+1)$ is the size of the parameter space).

5. Our Algorithm

In this section we describe the computations involved in our approach, in a format conducive to replicating our results. In Section 5.1 we describe how we compute the preconditioner, i.e. the cholesky matrix $\mathbf{C}$. In Section 5.2 we describe how we compute the basis matrices $\mathbf{W}_n$. In Section 5.3 we describe how we optimize the coefficients $d_n^{(s)}$ in test time.

5.1. Training time: computing the preconditioner

In order to compute the basis matrices we first need to compute the Cholesky factor $\mathbf{C}$ of (18). In order to do that we need to compute the average per-frame negated Hessian $\mathbf{H}$ of (17). It is possible to do so directly from (17), given the quantities $\mathbf{H}^{(s)}$ which may be computed from per-speaker adaptation statistics. In fact, for convenience and to avoid an extra pass over the data we did this a different way, via an approximation in which we assume model correctness. We compute the expected values of the $\mathbf{G}$ statistics of (8), as follows:

$$\bar{\mathbf{G}}_i \equiv \frac{1}{\sum_s \beta^{(s)}} \mathbf{G}_i^{(s)} \tag{27}$$

$$\simeq \sum_{j,m} P(j,m) \frac{1}{\sigma_{jm}^2(i)} \left(\boldsymbol{\mu}_{jm}^+ \, \boldsymbol{\mu}_{jm}^{+\,T} + \boldsymbol{\Sigma}_{jm}^{+0} \right), \tag{28}$$

where $P(j,m)$ is the occupation probability for mixture m of state j, $\boldsymbol{\mu}_{jm}^+ = [\boldsymbol{\mu}_{jm} \ , \ 1]$, and the notation $\boldsymbol{\Sigma}_{jm}^{+0}$ means the variance $\boldsymbol{\Sigma}_{jm}$ extended with an extra row and column equal to zero. The quantity $P(j,m)$ may be computed as $P(j,m) = P(j)c_{jm}$ where $P(j)$ is the occupation probability of state j, and c_{jm} are the Gaussian mixture weights; however, we just used $P(j) = 1/J$ where J is the number of states.

Looking at the auxiliary function (7), we see that there are two terms that contribute to the second derivative w.r.t. $\mathbf{w}$: the log determinant term and the term involving the matrices $\mathbf{G}_i$. We will write

$$\mathbf{H} = \mathbf{H}^{(1)} + \mathbf{H}^{(2)}, \tag{29}$$

where $\mathbf{H}^{(1)}$ arises from the log determinant term and has elements given by:

$$h^{(1)}_{((i-1)(D+1)+j),((k-1)(D+1)+l)} = \delta(i,l)\delta(j,k). \tag{30}$$

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 $\mathbf{H}^{(1)} := \mathbf{0}$ and then[1] for $1 \leq i,j \leq D$, set $h^{(1)}_{((i-1)(D+1)+j),((j-1)(D+1)+i)} := 1$. The second term $\mathbf{H}^{(2)}$ is given by:

$$\mathbf{H}^{(2)} = \mathrm{diag}\,(\bar{\mathbf{G}}_1, \bar{\mathbf{G}}_2, \ldots, \bar{\mathbf{G}}_D), \tag{31}$$

[1]To minimize the probability of error: in zero-based indexing, the procedure becomes: for $0 \leq i,j < D$, set $h^{(1)}_{(i(D+1)+j),(j(D+1)+i)} := 1$.

where we approximate $\bar{\mathbf{G}}_i$ using (28), and we interpret diag here in the obvious way. The reason for the transpose in $\mathbf{w} = \mathrm{vec}\,(\mathbf{W}^T)$ is to get this block diagonal structure. After obtaining $\mathbf{H}$ from (29) we compute $\mathbf{C}$ as in (18).

5.2. Training time: computing the basis

After computing the preconditioner $\mathbf{C}$, we need to compute the basis matrices $\mathbf{W}_n$. These are computed via PCA on the matrix $\hat{\mathbf{M}} \in \mathbb{R}^{D(D+1)\times D(D+1)}$ defined in (25) as a weighted scatter of auxiliary function gradients. In fact we find it more convenient to compute a matrix $\mathbf{M}$ defined in terms of the un-transformed gradients $\mathbf{p}^{(s)}$, and convert it into the transformed space afterwards. Thus, we compute for each speaker s in the training set, the quantity:

$$\mathbf{p}^{(s)} = \left.\frac{\partial \mathcal{Q}}{\partial \mathbf{w}^{(s)}}\right|_{\mathbf{w}^{(s)}=\mathbf{w}_0} \tag{32}$$

$$= \mathrm{vec}\left(\left(\mathbf{K}^{(s)} + \beta^{(s)}\,[\mathbf{I}\ ;\ \mathbf{0}] - \begin{bmatrix} \mathbf{g}_{11}^{(s)} \\ \vdots \\ \mathbf{g}_{DD}^{(s)} \end{bmatrix}\right)^T\right) \tag{33}$$

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 use later as (38); here, we use $\mathbf{W}_0 = [\mathbf{I}\ ;\ \mathbf{0}]$ to simplify. We then accumulate the quantity $\mathbf{M}$ as:

$$\mathbf{M} = \sum_s \frac{1}{\beta^{(s)}} \mathbf{p}^{(s)}\ \mathbf{p}^{(s)\,T}, \tag{34}$$

and then compute $\hat{\mathbf{M}} = \mathbf{C}^{-1}\mathbf{M}\mathbf{C}^{-T}$. As mentioned, the basis vectors $\mathbf{w}_n$ correspond to the eigenvectors of $\hat{\mathbf{M}}$, but the Singular Value Decomposition (SVD) is more convenient to implement in software, so we do the decomposition:

$$\hat{\mathbf{M}} = \mathbf{U}\mathbf{L}\mathbf{V}^T \tag{35}$$

in which we require that the diagonal matrix $\mathbf{L}$ be sorted from greatest to least value. Since $\hat{\mathbf{M}}$ is positive semi-definite, we have that $\hat{\mathbf{M}} = \mathbf{U}\mathbf{L}\mathbf{U}^T$; the columns of $\mathbf{U}$ are the eigenvectors of $\hat{\mathbf{M}}$ and the diagonal elements of $\mathbf{L}$ are the corresponding eigenvalues. The basis matrices $\{\mathbf{W}_n, 1 \le n \le D(D+1)\}$ are given by:

$$\mathrm{vec}\,(\mathbf{W}_n^T) = \mathbf{C}^{-T}\mathbf{u}_n, \tag{36}$$

where $\mathbf{u}_n$ is the n'th column of $\mathbf{U}$. The only quantities that we need to retain from training time are these basis matrices.

5.3. Test time: optimizing the coefficients

Here we describe the update phase of one E-M iteration in test time. The statistics accumulation and the overall E-M process are the same as previously described, e.g. [4]. We describe the update for a generic speaker s, and we will drop the superscript $\cdot^{(s)}$ in this section to avoid cluttering the equations. The situation is as follows: we are given the speaker's statistics $\mathbf{K}$ and $\{\mathbf{G}_i, 1 \leq i \leq D\}$, and the data-count β: see Eqs. (8) to (10). We have the stored basis matrices $\{\mathbf{W}_n, 1 \leq n \leq D(D+1)\}$. We are also given an initial version of the speaker's adaptation matrix $\mathbf{W}$: if this is the first E-M iteration this will equal $\mathbf{W}_0 = [\mathbf{I}\ ;\ \mathbf{0}]$, but otherwise it will be the quantity estimated on the previous iteration. We use (26) to compute the speaker-specific basis size $N(s)$.

The process we describe here is an iterative optimization of the coefficients $d_n^{(s)}$. For iterations $1 \leq q \leq Q$ we compute the gradient w.r.t. the coefficients, compute the search direction (which may be the gradient direction or may be determined by the conjugate gradient method), and then iteratively compute an optimal step-size in that direction using Newton's method. However, in our algorithm the quantities $d_n^{(s)}$ become implicit and we deal directly with the matrix $\mathbf{W}$.

We now describe the algorithm. The four phases below are to be done in turn for each iteration $1 \leq q \leq Q$.

5.3.1. Check auxiliary function

For diagnostic purposes, on each iteration q we compute the auxiliary function (7). If this decreases on any iteration, it indicates an implementation error (very small decreases may be due to roundoff).

5.3.2. Compute auxiliary function gradient

Next we compute a quantity $\mathbf{P} \in \mathbb{R}^{(D \times (D+1))}$ which is the derivative of (7) w.r.t. $\mathbf{W}$. First we compute $\mathbf{S} \in \mathbb{R}^{(D \times (D+1))}$ which is the contribution of the quadratic term of (7) to the derivative, as follows:

$$\mathbf{s}_i := \mathbf{G}_i \mathbf{r}_i, \tag{37}$$

where $\mathbf{s}_i$ and $\mathbf{r}_i$ are the i'th rows of $\mathbf{S}$ and $\mathbf{W}$ respectively, viewed as column vectors. We can then compute:

$$\mathbf{P} := \beta \left[\mathbf{A}^{-T} \; ; \; 0\right] + \mathbf{K} - \mathbf{S}, \tag{38}$$

where $\mathbf{A}$ corresponds to the first D columns of $\mathbf{W}$.

5.3.3. Compute step direction

The step direction, represented as a change in $\mathbf{W}$, is a matrix $\mathbf{\Delta} \in \mathbb{R}^{D\times(D+1)}$. If using the simple gradient-based method, this is computed as:

$$\mathbf{\Delta} := \sum_{n=1}^{N(s)} \mathrm{tr}\,(\mathbf{W}_n^T \mathbf{P}) \mathbf{W}_n. \tag{39}$$

The reader may notice that $\mathbf{\Delta}$ does not equal the gradient of the auxiliary function (or we would have $\mathbf{\Delta} = \mathbf{P}$), but when the equations are written in terms of the coefficients $d_n^{(s)}$ it is equal to the gradient; alternately, it may be viewed as a "preconditioned" gradient.

If we are using the conjugate gradient modification, we use the symbol $\mathbf{D}$ for the (preconditioned) gradient, and $\mathbf{\Delta}$ for the step direction. In the conjugate gradient version of the algorithm, we do:

$$\mathbf{D} := \sum_{n=1}^{N(s)} \mathrm{tr}\,(\mathbf{W}_n^T \mathbf{P}) \mathbf{W}_n. \tag{40}$$

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

$$\mathbf{\Delta} := \mathbf{D} \tag{41}$$

$$\bar{\mathbf{D}} := \mathbf{D}, \tag{42}$$

where the variable $\bar{\mathbf{D}}$ is used to store the gradient direction from the previous iteration, as required by the conjugate gradient method. Otherwise ($q > 1$) we make the following application of the Polak-Ribière formula with "restarting" (see [13]):

$$b := \max\left(0, \frac{\mathrm{tr}\,(\mathbf{D}^T\mathbf{D}) - \mathrm{tr}\,(\mathbf{D}^T\bar{\mathbf{D}})}{\mathrm{tr}\,(\bar{\mathbf{D}}^T\bar{\mathbf{D}})}\right) \tag{43}$$

$$\mathbf{\Delta} := \mathbf{D} + b\mathbf{\Delta} \tag{44}$$

$$\bar{\mathbf{D}} := \mathbf{D}. \tag{45}$$

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

5.3.4. Compute optimal step size, and update matrix

We will take a step $\mathbf{W} := \mathbf{W} + k\mathbf{\Delta}$ for a positive scalar k. Here we describe how we optimize k. We use Newton's method, and check on each iteration that the auxiliary function has not decreased. Note that if the preconditioning is accurate we expect $k \simeq 1$, at least in an order-of-magnitude sense, so that $k \ll 1$ or $k \gg 1$ may indicate problems.

Before the iterations of Newton's method, we compute the following scalar quantities:

$$b := \mathrm{tr}\,(\mathbf{\Delta}\mathbf{K}^T) - \mathrm{tr}\,(\mathbf{\Delta}\mathbf{S}^T) \tag{46}$$

$$c := \sum_{i=1}^{D} \boldsymbol{\delta}_i^T \mathbf{G}_i \boldsymbol{\delta}_i, \tag{47}$$

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

$$\mathcal{Q}^{(q)}(k) = \beta \log|\det(\mathbf{A} + k\mathbf{\Delta}_{1:D})| + k\,b - \tfrac{1}{2}k^2 c \tag{48}$$

which is (7) written in terms of k, where the superscript $\cdot^{(q)}$ is to remind us that the quantities we use to define this function depend on the iteration index q (we have omitted the iteration index q from variables used in the algorithm in order to make it clearer how to implement this method). Note that $\mathbf{\Delta}_{1:D}$ represents the first D columns of $\mathbf{\Delta}$. We compute k iteratively, starting from $k = 0$, for iterations $1 \le r \le R$ (we used $R = 3$ as this converges very fast). On each iteration r we compute the following quantities:

$$\mathbf{N} := (\mathbf{A} + k\mathbf{\Delta}_{1:D})^{-1}\mathbf{\Delta}_{1:D} \tag{49}$$

$$d_1 := \beta\mathrm{tr}\,(\mathbf{N}) + b - kc \tag{50}$$

$$d_2 := \min\left(-\beta\mathrm{tr}\,(\mathbf{N}\mathbf{N}) - c, -\frac{c}{10}\right) \tag{51}$$

where d_1 and d_2 are the first and second derivatives of (48) w.r.t. k. In the expression for d_2, the first term in the min expression corresponds to the actual second derivative and the second term is a floor to avoid the theoretical possibility of a positive second derivative which would lead to the wrong direction of update (in fact, in our experiments the second half of the min expression was never taken). We then generate a candidate new value of k:

$$\hat{k} := k - d_1/d_2. \tag{52}$$

We evaluate (48) for k and $\hat{k}$. If $\mathcal{Q}^{(q)}(\hat{k}) < \mathcal{Q}^{(q)}(k)$, then we have overshot and we need to reduce the step size by setting

$$\hat{k} := (\hat{k} + k)/2. \tag{53}$$

This should happen rarely; if it does, we keep checking the auxiliary function and while $\mathcal{Q}^{(q)}(\hat{k}) < \mathcal{Q}^{(q)}(k)$ we repeat (53). However, there should be a maximum number of repetitions of this (e.g. 10) because numerical roundoff can cause spurious failures of the auxiliary function check if we are close to convergence; if the maximum number is reached we leave k unchanged and terminate the loop over r. Otherwise, once the check succeeds we set $k := \hat{k}$ and continue the loop over r.

After computing k we take the step $\mathbf{W} := \mathbf{W} + k\mathbf{\Delta}$.

6. Experimental Results

6.1. System description

6.1.1. 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 MFCCs with Δ, $\Delta\Delta$ and $\Delta\Delta\Delta$ (i.e. deltas, accelerations and third derivatives), 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 Vocal Tract Length Normalization (VTLN), but rely instead on gender-dependent models. To obtain these we train gender-independent models and further train these for four more E-M iterations on only male or female data. After this we perform 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.

6.1.2. 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 on average 46 Gaussians per state. The models were trained with Minimum Classification Error (MCE) [9]. 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.

6.1.3. Enhanced 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 [17]. The number of clustered states is 10144, with on average 47 Gaussians per state. 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 weighted by a factor of 20 in training (equivalent to replicating the voicemail training data 20 times). We present test 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.

6.1.4. Data subsets

	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
#Utterances	12.6K	3.7K	3.5K	1.4K	210	138	81	23
Length (h)	7.8	3.9	5.2	2.9	0.68	1.08	1.14	0.64
%WER (Unadapted)	5.64	1.63	0.65	0.98	32.8	31.4	33.3	35.8

Table 1: Utterance duration bins: definition, length and baseline WERs

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 1 shows the minimum, maximum and mean utterance length for each of 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%.

6.2. Details of adaptation setup

Here we describe some details of our process for collecting CMLLR statistics and for speaker-adapted decoding; these apply both to our baseline and experimental methods. All our adaptation is done per utterance, i.e. we assume that each utterance corresponds to a unique speaker. This arises from the nature of the products that these test setups correspond to: the identity of a caller over the telephone will generally not be known with certainty. 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 from the previous segments 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-duration bin (left column of Table 1), 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 the baseline CMLLR optimization we used ten iterations in the update phase.

The only tunable parameters of our algorithm are the proportionality constant η, the number of outer iterations Q and the number of inner iterations R. Unless stated otherwise, we used $\eta = 0.2$, $Q = 10$ and $R = 3$ (and we used the gradient, not conjugate gradient, version of the test-time computation).

6.3. 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, and with HLDA features there is no special meaning to the three blocks; however, it seems to work in practice. We also compare with fMAPLR [11], 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. The prior used for fMAPLR in the 3-block case is a full-covariance Gaussian that models the non-zero part of each row, which is a vector of dimension $D/3 + 1$.

Because the estimation of the prior in fMAPLR did 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 cepstral mean normalization) that is used in test time.

6.4. Technical issues and tuning

Fig. 1 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; for an explanation of the factor of $\frac{1}{2}$ see the text after Eq. (25). We display the absolute and cumulative singular values. From

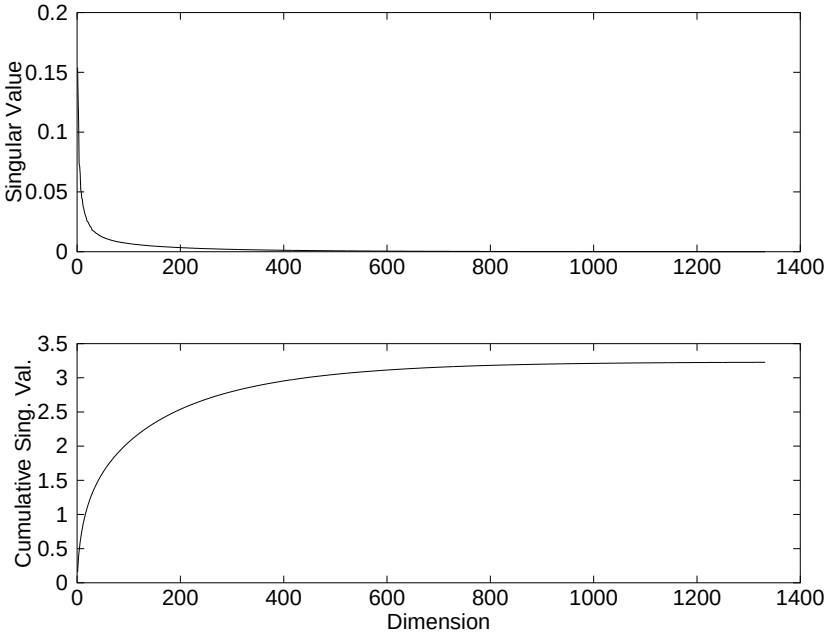

Figure 1: Singular values for basis training (IVR)

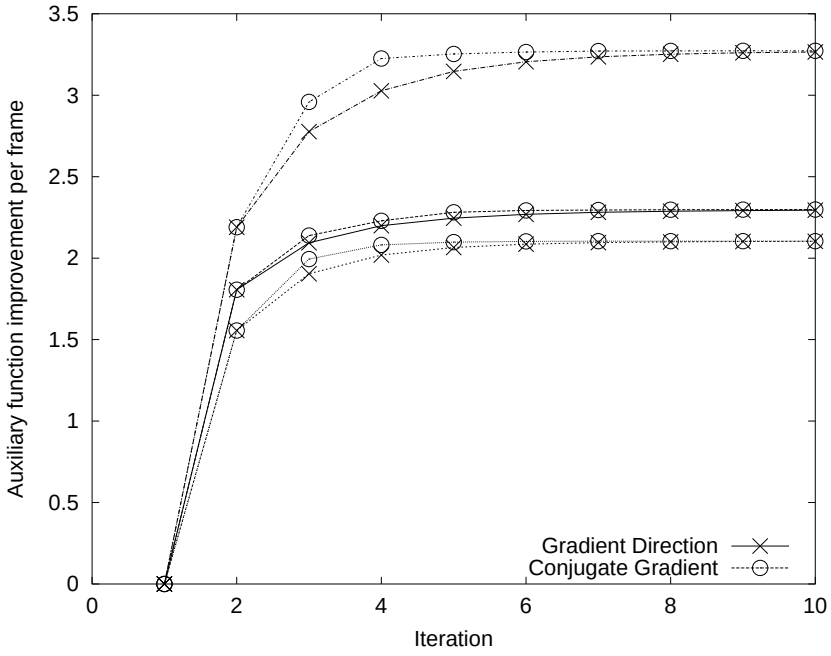

Figure 2: Convergence of update phase: three randomly chosen utterances

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.

Fig. 2 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 gradient 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 Conjugate Gradient methods, for various numbers of iterations, and these results (not shown) 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.

	Proposed	Baseline		
	Method	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 2: Time taken in CMLLR update phase (% of total CPU)

Table 2 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 lengths of the digits and voicemail utterances we used for this test were 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. The total real-time factor for IVR is 0.22 (i.e. faster than real time), and for EVM it is 1.33; these figures include two decoding passes.

Fig. 3 relates to tuning the proportionality factor η used to control the per-speaker basis size $N(s)$, on IVR digits. The best factor is $\eta = 0.2$. Fig. 4 shows the same for EVM, but this time we compare with tuning a fixed basis size N 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 (26) 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. Fig. 5 investigates tuning the scale on the log 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).

6.5. Main results

Our main results are in Fig. 6. This shows the relative WER improvement of various CMLLR adaptation strategies, compared with using no CMLLR

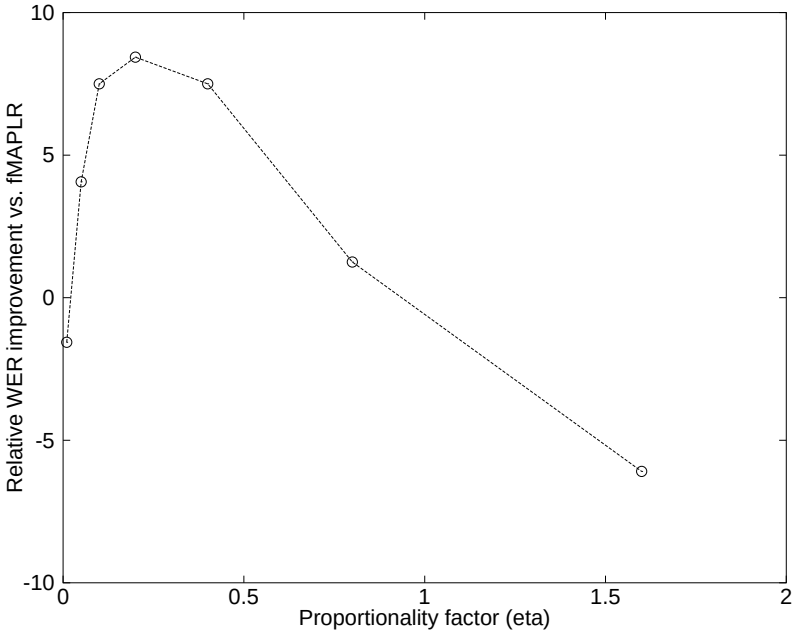

Figure 3: Tuning proportionality factor η: IVR digits

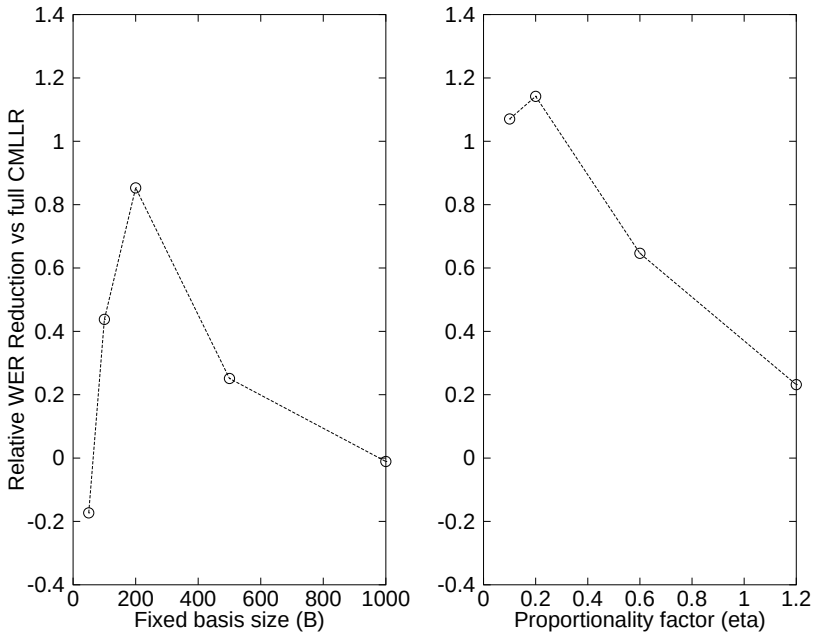

Figure 4: Adaptive vs. fixed basis size: EVM

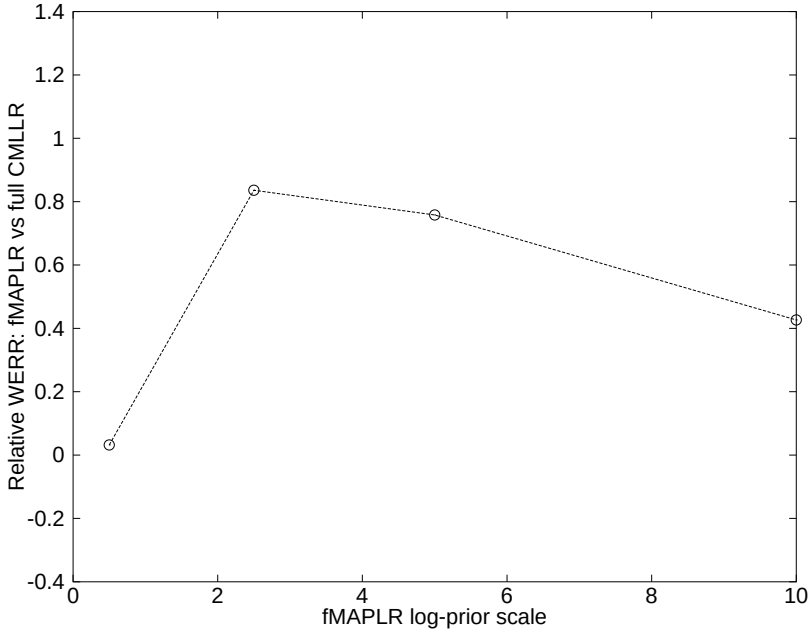

Figure 5: fMAPLR improvement vs. prior scaling factor: EVM

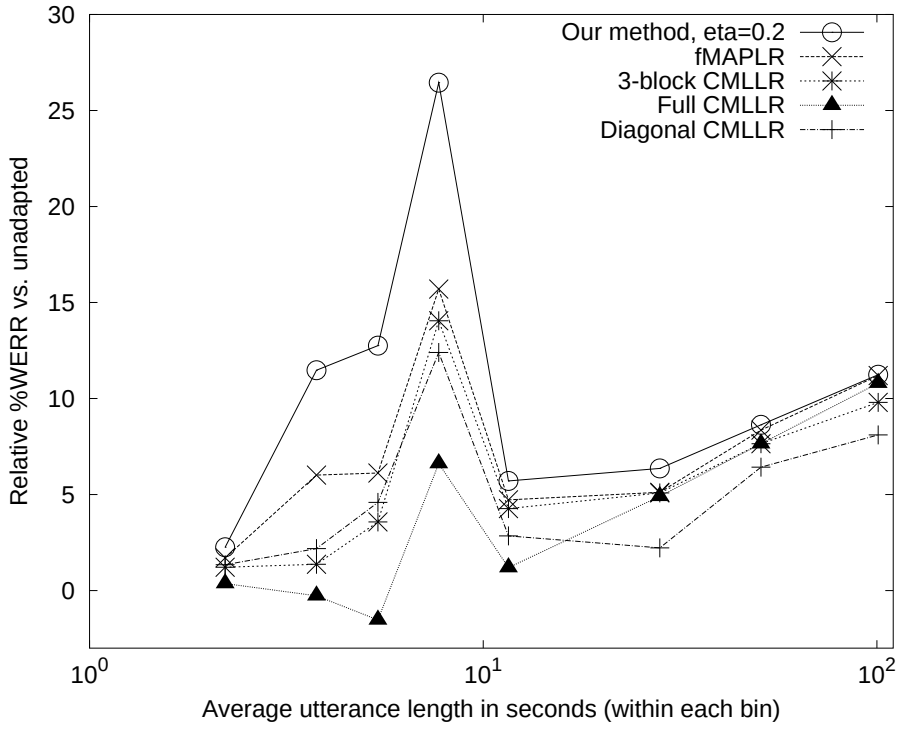

Figure 6: WER improvement from adaptation vs. utterance duration

adaptation, for the duration bins described in Table 1. 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". The line with full CMLLR (triangles) substantiates our claim in the introduction that CMLLR does not give improvements below about five seconds of data.

Note that the various bins of data are at very different absolute WERs (see Table 1, last row), and the relative improvements tend to be higher when the absolute WER is low. This explains the wide variations seen in Fig. 6. 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, for less than about 3 seconds of data (first bin), none of the adaptation methods give very much improvement. The improvement that our method gives 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 6.2, 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 when generalizing to systems in which the CMLLR is estimated after seeing the whole utterance. With this correction, we anticipate that the greatest improvement from our method will be between about 3 and 10 seconds of adaptation data.

Baseline	IVR, W per bin				EVM, W per bin			
CMLLR	2.56	2.83	3.89	3.81	3.87	1.78	1.21	0.39
fMAPLR	1.10	1.61	1.94	2.60	1.14	1.93	0.43	0.07
	Two-sided significance levels (if $\geq 85\%$)							
CMLLR	98.95%	99.53%	99.99%	99.99%	99.99%	92.5%	-	-
fMAPLR	-	89.26%	94.76%	99.07%	-	94.6%	-	-

Table 3: Significance results per bin, our method versus selected baselines (W statistic)

Significance tests per bin are given in Table 3, in terms of the statistic W of the matched-pair test of [7], which is a number of standard deviations; W is signed but is always positive here because our method was always better). We also provide significance levels, where relevant. The columns correspond to the duration intervals defined in the corresponding columns of Table 1, and also correspond to the points in Fig. 6, in order from left to right. These are pairwise significance tests, in which we compare our method to CMLLR and fMAPLR. If we pool the data for each task, for IVR and EVM respectively (the left four and right four bins of the table), the improvements versus CMLLR are significant, in the two-tailed sense, to 99.99% and 99.95%, and versus fMAPLR to 99.07% and 93.71%.

7. Conclusions

We have described a practical algorithm for estimating Constrained MLLR transforms robustly by training a set of basis matrices and, in test time, restricting the estimated matrix to a leading subset of the basis matrices. Stated very crudely, our method reduces the amount of adaptation data that is needed to obtain a substantial improvement, from about 10 seconds to about 3 seconds. We have shown that it gives significant improvement versus the next best baseline we tested, fMAPLR, and is faster than conventional CMLLR.

[1] Akaike, H., 2002. A new look at the statistical model identification. Automatic Control, IEEE Transactions on 19 (6), 716–723.

[2] Chen, K. T., Liau, W. W., Wang, H. M., Lee, L. S., 2000. Fast Speaker Adaptation Using Eigenspace-based Maximum Likelihood Linear Regression. In: Proc. ICSLP. Vol. 3. pp. 742–745.

[3] Digalakis, V., Rtischev, D., Neumeyer, L., 1995. Speaker adaptation using constrained estimation of Gaussian mixtures. IEEE Trans. on Speech and Audio Processing 3, 357–366.

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

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

[6] Ghoshal, A., Povey, D., Agarwal, M., Akyazi, P., Burget, L., Feng, K., Glembek, O., Goel, N., Karafiát, M., Rastrow, A., Rose, R. C., Schwarz, P., Thomas, S., 2010. A Novel Estimation of Feature-space MLLR for Full Covariance Models. In: Proc. ICASSP. pp. 4310–4313.

[7] Gillick, L., Cox, S., 1989. Some statistical issues in the comparison of speech recognition algorithms. In: Proc. ICASSP. Vol. 1. pp. 532–535.

[8] Huang, J., Marcheret, E., Visweswariah, K., 2005. Rapid Feature Space Speaker Adaptation for Multi-Stream HMM-Based Audio-Visual Speech Recognition. IEEE Int'l Conf on Multimedia and Expo, 338–341.

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

[10] Kuhn, R., Perronnin, F., Nguyen, P., Rigazzio, L., 2001. Very Fast Adaptation with a Compact Context-Dependent Eigenvoice Model. In: Proc. ICASSP. pp. 373–376.

[11] Lei, X., Hamaker, J., He, X., 2006. Robust Feature Space Adaptation for Telephony Speech Recognition. In: Proc. ICSLP. pp. 773–776.

[12] Schwarz, G., 1978. Estimating the Dimension of a Model. The Annals of Statistics 6 (2), 461–464.

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

[14] Sim, K. C., Gales, M. J. F., 2005. Adaptation of Precision Matrix Models on Large Vocabulary Continuous Speech Recognition. In: Proc. ICASSP. Vol. 1. pp. 97–100.

[15] Visweswariah, K., Goel, V., Gopinath, R., 2002. Maximum Likelihood Training Of Bases For Rapid Adaptation. Available online.

[16] Visweswariah, K., Goel, V., Gopinath, R., 2002. Structuring Linear Transforms for Adaptation Using Training Time Information. In: Proc. ICASSP. pp. 3238–3241.

[17] Woodland, P., Povey, D., 2000. Large Scale MMIE Training For Conversational Telephone Speech Recognition. In: Proc. Speech Transcription Workshop.