XMLLR for Improved Speaker Adaptation in Speech Recognition

Daniel Povey, Hong-Kwang J. Kuo

IBM T.J. Watson Research Center, Yorktown Heights, NY, USA
{dpovey,hkuo}@us.ibm.com

Abstract

In this paper we describe a novel technique for adaptation of Gaussian means. The technique is related to Maximum Likelihood Linear Regression (MLLR), but we regress not on the mean itself but on a vector associated with each mean. These associated vectors are initialized by an ingenious technique based on eigen decomposition. As the only form of adaptation this technique outperforms MLLR, even with multiple regression classes and Speaker Adaptive Training (SAT). However, when combined with Constrained MLLR (CMLLR) and Vocal Tract Length Normalization (VTLN) the improvements disappear. The combination of two forms of SAT (CMLLR-SAT and MLLR-SAT) which we performed as a baseline is itself a useful result; we describe it more fully in a companion paper. XMLLR is an interesting approach which we hope may have utility in other contexts, for example in speaker identification.

Index Terms: speech recognition, speaker adaptation, MLLR

1. Introduction

In the popular speaker adaptation technique called Maximum Likelihood Linear Regression (MLLR) [1], the means are adapted per speaker using an affine transform:

$$\hat{\mu}_j^{(s)} = \mathbf{A}^{(s)}\mu_j + \mathbf{b}^{(s)}. \quad (1)$$

This is often combined with the use of regression classes [2], in which the transform parameters $\mathbf{A}^{(s)}$ and $\mathbf{b}^{(s)}$ depend on the particular Gaussian j; sometimes there are just two regression classes corresponding to silence and non-silence, sometimes a variable number based on a tree of clustered phones.

The technique we are proposing here we are calling eXtended MLLR (XMLLR); it is a mean adaptation technique where we do, per speaker:

$$\hat{\mu}_j^{(s)} = \mu_j^{(s)} + \mathbf{A}^{(s)}\mathbf{n}_j, \quad (2)$$

where $\mathbf{A}^{(s)}$ is a speaker-specific transformation and $\mathbf{n}_j$ are vectors which we associate with each Gaussian j. We compute these associated vectors $\mathbf{n}_j$ using an ingenious eigenvalue based method; this invites comparison with Eigenvoices [5] but our technique has a much higher ratio of speaker-specific parameters to global parameters. This technique is also similar to the Cluster Adaptive Training approach B-CAT [4], but the initialization technique and other details differ.

Section 2 describes the method by which we initialize the associated vectors; Section 3 describes how we further optimize them; Sections 4 and 5 gives experimental conditions and results, and Section 6 concludes.

2. Eigen decomposition method of initializing associated vectors

In this section we describe an eigenvalue based method of initializing the vectors $\mathbf{n}_j$. The method assumes we have a system already trained (i.e. the means and variances, but not the associated vectors), and it must have a number of Gaussians J less than about 10,000 so it we can fit in memory a matrix of that dimension. We can use this initialization to bootstrap a larger system.

Suppose the associated vectors $\mathbf{n}_j$ are of size E (say, $E = 80$) and we have a total of, say $J = 5000$ Gaussians $1 \leq j \leq J$ in our HMM set. Let $\mathbf{N}$ be a tall ($J \times E$) matrix, the j'th row of which is $\mathbf{n}_j$.

We can write the overall improvement in the auxiliary function that we obtain from our adaptation technique, as a function of $\mathbf{N}$, as follows. Firstly, let us compute the improvement in likelihood from speaker s and feature dimension $1 \leq d \leq D$ where D is the total feature dimension e.g. 40. In the following we use the fact that $\mathbf{N}\mathbf{a}_d^{(s)}$ is a column vector representing the change in the mean of Gaussians 1 through J for dimension d of speaker s; $\mathbf{a}_d^{(s)}$ is the d'th row of speaker transform $\mathbf{A}^{(s)}$. It is helpful to think of each pair (s, d) as like a separate single-dimensional "sub-speaker." Our auxiliary function improvement for "sub-speaker" (s, d) is:

$$f(s,d) = \mathbf{v}_d^{(s)T}\mathbf{N}\mathbf{a}_d^{(s)} - 0.5\mathbf{a}_d^{(s)T}\mathbf{N}^T\mathrm{diag}(\mathbf{w}_d^{(s)})\mathbf{N}\mathbf{a}_d^{(s)}, \quad (3)$$

where $\mathrm{diag}(\cdot)$ means making a matrix whose diagonal is the given vector, and we define

$$v_{d,j}^{(s)} = \sum_{t=1}^{T} \gamma_j^{(s)}(t)\frac{x_d^{(s)}(t) - \mu_{j,d}}{\sigma_{j,d}^2} \quad (4)$$

$$w_{d,j}^{(s)} = \sum_{t=1}^{T} \gamma_j^{(s)}(t)\frac{1}{\sigma_{j,d}^2}. \quad (5)$$

$v_{d,j}^{(s)}$ and $w_{d,j}^{(s)}$ represent the linear and quadratic terms in the objective function for the mean-offset of Gaussian j, speaker s and dimension d. We write $\gamma_j^{(s)}(t)$ and $\mathbf{x}^{(s)}(t)$ for the Gaussian posteriors and features. We can solve (3) for $\mathbf{a}_d^{(s)}$:

$$\mathbf{a}_d^{(s)} = \left(\mathbf{N}^T\mathrm{diag}(\mathbf{w}_d^{(s)})\mathbf{N}\right)^{-1}\mathbf{N}^T\mathbf{v}_d^{(s)}, \quad (6)$$

and substituting this into Equation 3 and simplifying we get:

$$f(s,d) = 0.5\mathbf{v}_d^{(s)T}\mathbf{N}\left(\mathbf{N}^T\mathrm{diag}(\mathbf{w}_d^{(s)})\mathbf{N}\right)^{-1}\mathbf{N}^T\mathbf{v}_d^{(s)}. \quad (7)$$

The overall objective function is a sum over all $f(s, d)$. This is starting to look like an eigenvalue problem but we first need to get rid of the diagonal matrix $\mathrm{diag}(\mathbf{w}_d^{(s)})$ in the middle. We will do this as best we can by using appropriate per-speaker, per-Gaussian and per-dimension constants to make the remaining elements as close to unity as possible, and then ignore them.

2.1. Approximating $\mathbf{w}_d^{(s)}$

In approximating $\mathbf{w}_d^{(s)}$ we use the following things: we compute the average per-dimension variance $\bar{\sigma}_d^2$ over all the Gaussians in our model; we also use the number of frames T_s for

each speaker, and the prior p_j of Gaussian j (these sum to one over all Gaussians); we also compute a constant $c_j = \frac{1}{D}\sum_{d=1}^{D}\frac{\bar{\sigma}_d^2}{\sigma_{j,d}^2}$ which is large if Gaussian j has smaller than average variances; we can then approximate $w_{d,j}^{(s)} \simeq \frac{1}{\bar{\sigma}_d^2} T_s p_j c_j$. We break down $\mathbf{w}_d^{(s)}$ into a "sub-speaker"-specific constant $k_d^{(s)}$ multiplied by an "average" value $\bar{\mathbf{w}}$, where

$$\mathbf{w}_d^{(s)} \simeq k_d^{(s)} \bar{\mathbf{w}} \quad (8)$$

$$k_d^{(s)} = T_s / \bar{\sigma}_d^2 \quad (9)$$

$$\bar{\mathbf{w}}_j = p_j \frac{1}{D}\sum_{d=1}^{D}\frac{\bar{\sigma}_d^2}{\sigma_{j,d}^2} \quad (10)$$

We can then approximate the objective function as:

$$f \simeq \sum_{s,d} \frac{0.5}{k_d^{(s)}} \mathbf{v}_d^{(s)^T} \mathbf{N} \left(\mathbf{N}^T \mathrm{diag}(\bar{\mathbf{w}}) \mathbf{N}\right)^{-1} \mathbf{N}^T \mathbf{v}_d^{(s)}. \quad (11)$$

2.2. Eigenvalue solution for N

We can then solve for $\mathbf{N}$ by using the substitution

$$\mathbf{O} = \mathrm{diag}(\bar{\mathbf{w}})^{0.5} \mathbf{N} \quad (12)$$

The objective function then becomes:

$$f \simeq \sum_{s,d} \frac{0.5}{k_d^{(s)}} \mathbf{v}_d^{(s)^T} \mathrm{diag}(\bar{\mathbf{w}})^{-0.5} \mathbf{O} \left(\mathbf{O}^T \mathbf{O}\right) \mathbf{O}^T \mathrm{diag}(\bar{\mathbf{w}})^{-0.5} \mathbf{v}_d^{(s)}. \quad (13)$$

We can make the arbitrary stipulation that $\mathbf{O}^T\mathbf{O} = I$ (we can show that it does not matter what this is as long as it is non-singular). Then we can see that the columns of $\mathbf{O}$ are the principal eigenvectors of $\mathbf{M}$, where

$$\mathbf{M} = \sum_{s=1}^{S}\sum_{d=1}^{D} \frac{0.5}{k_d^{(s)}} \mathrm{diag}(\bar{\mathbf{w}})^{-0.5} \mathbf{v}_d^{(s)} \mathbf{v}_d^{(s)^T} \mathrm{diag}(\bar{\mathbf{w}})^{-0.5}. \quad (14)$$

We can then set $\mathbf{N} = \mathrm{diag}(\bar{\mathbf{w}})^{-0.5}\mathbf{O}$.

2.3. Computation for initialization of XMLLR

For a small system (we use 5000 Gaussians for the initialization) we can compute the matrix $\mathbf{M}$ quite efficiently. For each speaker we accumulate statistics as for ML training (count and mean; variance is not needed); from this we can get $\mathbf{v}_d^{(s)}$ for this speaker and each dimension and accumulate its weighted outer product; that is, accumulate the sum $\sum_d^{(s)} 1/k_d^{(s)} \mathbf{v}_d^{(s)} \mathbf{v}_d^{(s)^T}$. We take into account the sparseness of $\mathbf{v}_d^{(s)}$ when accumulating the outer product and iterate over a list of nonzero indexes. We do this in parallel for different blocks of data, sum up the different parts and then scale by $\bar{\mathbf{w}}^{-0.5}$ prior to finding the top eigenvectors. Doing a singular value decomposition (SVD) on $\mathbf{M}$ for dimension 5000 should take about 1 hour at 1GFLOP assuming SVD takes $O(30n^3)$ operations; in fact we use a very fast method which we will not describe here.

3. Iterative optimization in XMLLR

After initialization of the associated vectors $\mathbf{n}_j$ we start an iterative process where we first compute the per-speaker matrices $\mathbf{A}^{(s)}$, then recompute the per-Gaussian associated vectors $\mathbf{n}_j$, and then re-estimate the Gaussian mixture weights and parameters μ_j and Σ_j^2; we do each of these three steps for 4 iterations in experiments reported here. If we want to convert to a larger size system, at that point we use the per-speaker matrices $\mathbf{A}^{(s)}$ to compute the associated vectors $\mathbf{n}_j$ for the larger system and start 4 iterations of the same process on the larger system. This requires that the smaller and larger system share the same features.

3.1. Computing per-speaker matrices $\mathbf{A}^{(s)}$

The computation of per-speaker adaptation matrices $\mathbf{A}^{(s)}$ is similar to the MLLR computation [1]. We first adapt the means using any previous speaker adaptation matrix used and any current associated vectors, and compute the Gaussian posteriors. Then we compute per-speaker statistics as for ML training but omitting the variance, i.e. we accumulate counts $\gamma_j^{(s)}$, and data sums $\mathbf{x}_j^{(s)}$ From these statistics we accumulate for each feature dimension d a vector $\mathbf{v}_d^{(s)}$ and matrix $\mathbf{M}_d^{(s)}$ that appear in our objective function:

$$f(s,d) = C + \mathbf{a}_d^{(s)^T} \mathbf{v}_d^{(s)} - 0.5 \mathbf{a}_d^{(s)^T} \mathbf{M}_d^{(s)} \mathbf{a}_d^{(s)} \quad (15)$$

$$\mathbf{v}_d^{(s)} = \sum_{t=1}^{T_s}\sum_{j=1}^{J} \gamma_j^{(s)}(t) \frac{\mathbf{x}_d^{(s)}(t) - \mu_{j,d}}{\sigma_{j,d}^2} \mathbf{n}_j \quad (16)$$

$$\mathbf{M}_d^{(s)} = \sum_{t=1}^{T_s}\sum_{j=1}^{J} \frac{\gamma_j^{(s)}(t)}{\sigma_{j,d}^2} \mathbf{n}_j \mathbf{n}_j^T \quad (17)$$

We then solve $\mathbf{a}_d^{(s)} = \mathbf{v}_d^{(s)} \mathbf{M}_d^{(s)^{-1}}$. We also enforce a per-dimension of c_{min} frames; by default we use 10 for this. Thus, if necessary we truncate the dimension of $\mathbf{v}_d^{(s)}$ and $\mathbf{M}_d^{(s)}$ to T/c_{min} and use zero for any remaining dimensions in $\mathbf{a}_d^{(s)}$. This makes sense because the dimensions of $\mathbf{n}_j$ are sorted by eigenvalue.

3.2. Re-computing associated vectors $\mathbf{n}_j$

3.2.1. Full computation

The most exact (but expensive) form of the computation of the associated vectors $\mathbf{n}_j$ is similar in requirements to Speaker Adaptive Training [6] but in the dimension E which in our case is 80. Like the computation above it requires computing vectors $\mathbf{v}_j$ and matrices $\mathbf{M}_j$ which form an auxiliary function $f(j)$ as follows:

$$f(j) = \mathbf{v}_j^T \mathbf{n}_j - 0.5 \mathbf{n}_j^T \mathbf{M}_j \mathbf{n}_j \quad (18)$$

$$\mathbf{v}_j = \sum_{s,d}\sum_{t=1}^{T_s} \gamma_j^{(s)}(t) \frac{x_d^{(s)}(t) - \mu_{j,d}}{\sigma_{j,d}^2} \mathbf{a}_d^{(s)} \quad (19)$$

$$\mathbf{M}_j = \sum_{s,d}\sum_{t=1}^{T_s} \frac{\gamma_j^{(s)}(t)}{\sigma_{j,d}^2} \mathbf{a}_d^{(s)} \mathbf{a}_d^{(s)^T}. \quad (20)$$

The solution is then $\mathbf{n}_j = \mathbf{v}_j \mathbf{M}_j^{-1}$. As for the per-speaker matrices, the Gaussian posteriors $\gamma_j^{(s)}(t)$ must be obtained using the current adapted form of the Gaussians.

3.2.2. Quick computation

We also used a quick form of the computation of the associated vectors $\mathbf{n}_j$, which we found to be as effective as the full form. This relies on storing only the diagonal of the quadratic term $\mathbf{M}_j$ for most Gaussians j, but storing the full $\mathbf{M}_j$ for a subset $\mathcal{K}$ of Gaussians (we use 1 in E of the total). We then do a diagonal update of $\mathbf{n}_j$, but using dimension specific learning rate factors $\mathbf{l} = l_e, 1 \leq e \leq E$ which are learned from the subset for which we stored the full matrix. These should be between zero and 1 (in fact we enforce $0.1 \leq l_e \leq 1$). We formulate the computation as an update to $\mathbf{n}_j$. The computation becomes (note the dependence now on the adapted mean $\hat{\mu}_j^{(s)}$):

$$
\begin{aligned}
\mathbf{v}_j &= \sum_{s=1}^{S}\sum_{d=1}^{D}\sum_{t=1}^{T_s}\gamma_j^{(s)}(t)\frac{\mathbf{x}_d^{(s)}(t)-\hat{\mu}_j^{(s)}(d)}{\sigma_{j,d}^2}\mathbf{a}_d^{(s)} \\
\mathbf{M}_j &= \sum_{s=1}^{S}\sum_{d=1}^{D}\sum_{t=1}^{T_s}\frac{\gamma_j^{(s)}(t)}{\sigma_{j,d}^2}\mathbf{a}_d^{(s)}\mathbf{a}_d^{(s)^T} \\
w_e &= \sum_{j\in\mathcal{K}} v_{j,e}\frac{v_{j,e}}{M_{j\,e,e}} \\
N_{e,f} &= \sum_{j\in\mathcal{K}} M_{j\,e,f}\frac{v_{j,e}}{M_{j\,e,e}}\frac{v_{j,f}}{M_{j\,f,f}} \\
\mathbf{l} &= \mathbf{w}\mathbf{N}^{-1} \\
n_{j,e} &:= n_{j,e} + l_e\frac{v_{j,e}}{M_{j\,e,e}}
\end{aligned}
$$

3.2.3. Gaussian update

We also update the Gaussian parameters using a form of Speaker Adaptive Training which is appropriate for our technique. Essentially the update consists of viewing the speaker adaptive modification to our Gaussian mean, as a modification of the opposite sign to our features. Writing the speaker-specific counts and mean and variance statistics as $\gamma_j^{(s)}$, $\mathbf{x}_j^{(s)}$ and $\mathbf{S}_j^{(s)}$, and if the speaker adaptive update to the mean is $\mathbf{m}_j^{(s)} = \hat{\mu}_j^{(s)} - \mu_j = \mathbf{A}^{(s)}\mathbf{n}_j$, then

$$\gamma_j = \textstyle\sum_{s=1}^{S}\gamma_j^{(s)} \quad (21)$$

$$\mathbf{x}_j = \textstyle\sum_{s=1}^{S}\mathbf{x}_j^{(s)} - \gamma_j^{(s)}\mathbf{m}_j^{(s)} \quad (22)$$

$$\mathbf{S}_j = \textstyle\sum_{s=1}^{S}\mathbf{S}_j^{(s)} - 2\mathbf{x}_j^{(s)}\mathbf{m}_j^{(s)} + \mathbf{m}_j^{(s)}\mathbf{m}_j^{(s)^T}. \quad (23)$$

Since we have diagonal Gaussians we only store the diagonal of the variance statistics $\mathbf{S}_j$. This approach requires storing two sets of Gaussian statistics in memory, one used to make speaker statistics and one used to store total statistics.

4. Experimental Setup

Our experiments use an English broadcast news transcription system. The acoustic models are trained on 50 hours of audio obtained by sampling entire shows from the 1996 and 1997 English Broadcast News Speech corpora. The recognition features are 40-d vectors computed via an LDA+MLLT projection of 9 spliced frames of 13-d PLP features. Features for the speaker-independent (SI) system are mean-normalized, while features for the speaker-adapted (SA) system are mean- and variance-normalized. In both cases, normalization statistics are accumulated per speaker. The LDA+MLLT transform is computed by interleaving semi-tied covariance estimation using a single, global class and HMM parameter estimation during system training to diagonalize the initial LDA projection. In the training of the speaker-adapted system we use both vocal tract length normalization (VTLN) and Constrained MLLR (CMLLR). The acoustic models are trained using maximum likelihood estimation.

There are either two or four decoding passes, each using adaptation based on the transcripts from the previous stage. In the SI systems we have a speaker-independent (SI) pass (cepstral mean subtraction only), and then a pass including MLLR or XMLLR. In the SA systems we have a SI pass, use this output compute VTLN warp factors for the VTLN pass, then a VTLN+CMLLR pass, then VTLN+CMLLR+(MLLR or XMLLR). In general all four passes use different HMMs (the exception is the last two, which are the same on iteration zero of XMLLR or in MLLR decoding without SAT).

In both the SI and SA cases we show experiments on a small model with 700 quinphone context-dependent states and 5k Gaussian mixture components, and a larger system with 1000 states and 30k Gaussians (SI) or 3000 states and 50k Gaussians (SA).

As a baseline for XMLLR because it includes SAT-like elements we compare against (MLLR-)SAT, which we implement as ML-like SAT as described in a companion paper [7].

The language model is a small (3.3M n-grams) interpolated back-off 4-gram model smoothed using modified Kneser-Ney smoothing. It was trained on a collection of 335M words from various public data sources, mostly released by LDC. The vocabulary size is 84K words. The test set consists of the dev04 English Broadcast News development set, a total of 2.06 hours.

5. Experimental Results

Iteration	WER		
	XMLLR	MLLR	MLLR+rtree
(None)	41.1%	41.1%	41.1%
0	37.9%	38.6%	38.4%
1	36.8%		37.1%
2	36.8%		37.0%
3	36.4%		37.1%
4	36.3%		37.2%
(all)			36.6%

Table 1: Speaker Independent, small (5k Gaussian)

Iteration	WER		
	XMLLR	MLLR	MLLR+rtree
(None)	34.1%	34.1%	34.4%
0	n/a	31.7%	31.9%
1	30.5%		30.9%
2	30.1%		30.9%
3	29.8%		30.9%
4	29.7%		30.7%
(all)			30.2%

Table 2: Speaker Independent, normal (30k Gaussian)

Iteration	WER	
	XMLLR	MLLR+rtree
(None)	33.9%	33.9%
0	33.3%	33.2%
1	33.1%	33.1%
2	33.0%	33.0%
3	32.9%	32.8%
4	32.7%	32.8%

Table 3: Speaker Adapted, small (5k Gaussian)

In these experiments, we show results without MLLR or XMLLR in rows marked "(None)", and with XMLLR and MLLR on various iterations of training. XMLLR iteration zero is after initialization of the associated vectors; following iteration numbers are iterations of updating everything (transforms, associated vectors using the quick approach, Gaussian parameters). MLLR iteration zero is a system without (MLLR-)SAT, and following iterations are iterations of SAT; iteration "(all)" means doing a system build from scratch with SAT; details on SAT training are in [7].

Iteration	WER	
	XMLLR	MLLR+rtree
(None)	26.0%	26.0%
0	n/a	25.3%
1	25.0%	25.2%
2	25.0%	25.2%
3	25.0%	25.1%
4	25.0%	25.1%
(all)		24.9%

Table 4: Speaker Adapted, normal (50k Gaussian)

Iteration	WER (XMLLR)		
	Full	Quick	...and no SAT
(None)	41.1%	41.1%	41.1%
0	37.9%	37.9%	37.9%
1	36.7%	36.8%	37.7%
2	36.7%	36.8%	37.6%
3	36.5%	36.4%	37.7%
4	36.3%	36.3%	37.5%

Table 5: Building style: (SI, small: 5k Gaussian)

Tables 1 and 2 show our technique on small and larger size speaker independent models. Comparing against MLLR without SAT, the improvements are quite impressive: 1.9% and 2.2% absolute for the small and large system (iteration 4). However our gains are reduced if we compare against systems with SAT: 0.3% and 0.5% respectively if we compare against systems trained from scratch with SAT. On the SA system (Tables 3 and 4), even more of our gains disappear: we get 0.5% and 0.3% respectively versus MLLR with no SAT, and 0.1% and -0.1% versus the best available SAT numbers.

Table 5 compares different methods of system building for XMLLR. We see that the quick update for the associated vectors gives about the same results as the full update, and that if we omit the SAT element (i.e. the fact that we train the Gaussian parameters jointly with the adaptation parameters) we lose 1.2% absolute. We tried modifying from 10 the per-dimension minimum count used in test time; there were small improvements (0.1% to 0.2%) from changing this to the 100 to 200 range, but these may not be significant. We tried combining, in test-time, the learned associated vectors with other predictors. In one experiment we simply appended the mean to the learned associated vectors but this gave a 0.3% degradation.

The improvements from XMLLR are associated with an increase in test-time likelihood. On the small SI system (Table 1) on iteration zero (just initialized) we reduced the dimension E from 80 to 40; this degraded WER from 37.9% to 38.3% which it still better than non regression tree MLLR with about the same number of per-speaker parameters (at 38.6%). The unadapted test likelihood was -54.78 per frame; non regression tree MLLR adapted was -52.82, and XMLLR adapted (E=40) was -52.59. For comparison the best XMLLR result in Table 1 had likelihood of -51.19 and the best regression-tree MLLR SAT result had -51.28. All these likelihood results track WER.

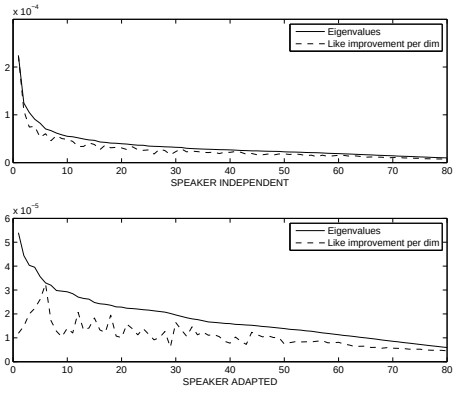

Figure 1: Eigenvalues of $\mathbf{M}$ and likelihood improvement

Figure 1 shows the eigenvalues of the matrix $\mathbf{M}$, and also the objective function improvement we get from adaptation, using each dimension of the computed associated vectors separately; the improvement is computed using the initially estimated associated vectors, and scaled arbitrarily for display. Any differences are due to approximations we made in the eigenvalue initialization. The correspondence seems quite close for the SI system; however, it breaks down for the SA system. We found a correspondence between lower-than-expected objective function improvement and uneven distribution of associated vectors, meaning that for the problematic dimensions most of the Gaussians had very small associated vector values in that dimension and just a few had large values. This appears to be due to certain Gaussians having very unevenly distributed counts among speakers, which breaks our assumptions. We were able to correct this discrepancy by iteratively identifying Gaussians with larger than average factor vectors $\mathbf{n}_j$ and decreasing their weight $\bar{w}_j^{-0.5}$ by a factor weakly dependent on the excess. This improved the objective function and WER on the first few iterations, but they both appeared to converge to the same point regardless of the initialization used (results not shown).

6. Conclusions

We have introduced a form of adaptation called eXtended MLLR (XMLLR), which is a modified form of MLLR in which we regress on vectors associated with each mean. XMLLR appears to be a more powerful form of adaptation than MLLR, giving large improvements against an MLLR baseline; however, when we implemented Speaker Adaptive Training (SAT) for use with MLLR and combined these techniques with other forms of adaptation the improvements versus MLLR disappeared. Nevertheless we hope that XMLLR may prove to be useful, perhaps in a different configuration or for use in speaker identification.

7. Acknowledgements

This work was funded by DARPA contract HR0011-06-2-0001.

8. References

[1] Leggetter, C. J. and Woodland, P. C., "Maximum Likelihood Linear Regression for Speaker Adaptation of Continuous Density Hidden Markov Models," Computer Speech and Language, v. 9, pp. 171-185, 1995.

[2] Gales M.J.F., "The generation and use of regression class trees for MLLR adaptation," Technical Report CUED/F-INFENG/TR263, Cambridge University, 1996.

[3] Gales, M. J. F., "Maximum Likelihood Linear Transformations for HMM-based Speech Recognition," Computer Speech and Language, v. 12, 1998.

[4] Gales, M. J. F, "Multiple-cluster adaptive training schemes," Proc. ICASSP, 2001.

[5] Kuhn R., Junqua J.-C., Nguyen, P., Niedzielski, N., "Rapid Speaker Adaptation in Eigenvoice Space," IEEE Transactions on Speech and Audio Processing, v. 8, no. 6, Nov. 2000.

[6] Anastasakos, T., McDonough, J., Schwartz R. and Makhoul, J., "A Compact Model for Speaker-Adaptive Training," Proc. ICSLP, 1996.

[7] Povey, D., Kuo, H-K. J. and Soltau H., "Fast Speaker Adaptive Training for Speech Recognition," submitted to: Interspeech, 2008.