AFFINE TRANSFORMATIONS IN SPEAKER ADAPTATION – WHY SIMPLER IS BETTER

Trym Holter, Julien Epps, Arun Gopalakrishnan and Eric Choi

Motorola Australian Research Centre, Motorola Labs
{Trym.Holter, Julien.Epps, Arun.Gopal, Eric.Choi}@motorola.com

ABSTRACT: Speaker adaptation is an important technique that can compensate for the mismatch between training data and the vocal characteristics of an individual user in a speech recognition system, however this can come at the cost of increased computational complexity. This paper reports a detailed comparison of four different affine transformation configurations for speaker adaptation, and the evaluation of their recognition accuracy, complexity and memory requirements. Results of this comparison show that for optimum parameter choices, simpler transformation configurations are capable of producing accuracies close to those of the conventional full transformation, allowing the computational complexity to be reduced by one to two orders of magnitude.

1. INTRODUCTION

Most contemporary automatic speech recognition (ASR) systems comprise a set of hidden Markov models (HMMs), each modelling a specific acoustic unit, typically a word or a phone-like unit. The most important features of an HMM are the probability density functions (pdfs) that specify the state output distributions. These are typically Gaussian mixture models (GMMs), and the mean and covariance vectors of the individual Gaussians are the most crucial parameters. In some very low-complexity systems, the parameters of the acoustic models are estimated from speech data provided by the actual user, thus giving a set of speaker-dependent (SD) models. More typically they are created off-line, based on a large corpus of speech data from many speakers. These are known as speaker-independent (SI) models. However, even with a large SI training corpus, many potential users will experience that their speaker characteristics are not well covered by the SI training data, thus leading to poor ASR performance.

To mitigate this problem, speaker adaptation can be used to adjust the acoustic models to better fit the characteristics of the specific user, based on a relatively small amount of speaker specific adaptation data. One common approach to speaker adaptation is to apply affine transformations to the mean vectors of the Gaussian components that comprise the acoustic models. An established technique for the calculation of these transformations is maximum likelihood linear regression (MLLR) (Leggetter and Woodland, 1995). In order to reduce the requirements for large amounts of adaptation data, a (potentially large) number of these components can share the same transform.

Speaker adaptation is known to greatly improve the ASR performance for most speakers, however the problem of rapid, robust adaptation of a set of SI models to a new user remains a substantial one. The solution of this problem at a sufficiently low complexity for feasible implementation in hand-held devices is a further challenge of interest. This has motivated previous investigations into simpler affine transformation structures such as diagonal transforms (Leggetter and Woodland, 1994), which concluded that their performance cannot match that of a more conventional full transform.

In this paper, a new evaluation of the recognition accuracy, computational complexity and memory requirements of different affine transformation configurations is made. Section 2 discusses some different transformation types, section 3 outlines an efficient scheme for MLLR, before the complexities of the proposed approaches are discussed in section 4. Section 5 describes and discusses the experiments before some conclusions are drawn in section 6.

2. MLLR TRANSFORMATION ESTIMATION

The criterion function to be maximised in MLLR is given by (Leggetter and Woodland, 1995):

$$J = -\frac{1}{2}\sum_{t=1}^{T}\sum_{k\in\Omega}\gamma_t(k)\left(\mathbf{o}_t - \mathbf{A}\boldsymbol{\mu}_k - \mathbf{b}\right)'\mathbf{R}_k\left(\mathbf{o}_t - \mathbf{A}\boldsymbol{\mu}_k - \mathbf{b}\right) = -\frac{1}{2}\sum_{t=1}^{T}\sum_{k\in\Omega}\gamma_t(k)\left(\mathbf{o}_t - \mathbf{W}\hat{\boldsymbol{\mu}}_k\right)'\mathbf{R}_k\left(\mathbf{o}_t - \mathbf{W}\hat{\boldsymbol{\mu}}_k\right), \qquad (1)$$

where T is total number of feature vectors in an adaptation data set, Ω is the set of Gaussians within a regression class, $\mathbf{o}_t$ is the feature vector at time *t*, $\gamma_t(k)$ is the posterior probability of $\mathbf{o}_t$ occupying the *k*-th Gaussian at time *t*, $\boldsymbol{\mu}_k$ is the mean vector of the *k*-th Gaussian and $\mathbf{R}_k$ is the corresponding diagonal covariance matrix. A is the rotation matrix of the transform and $\mathbf{b}$ is the bias vector. To simplify notation we let $\mathbf{W} = [\mathbf{A}\ \mathbf{b}]$, $\hat{\boldsymbol{\mu}}' = [\boldsymbol{\mu}'\ 1]$, $\bar{\gamma}_k = \sum_{t=1}^{T}\gamma_t(k)$, and $\bar{\mathbf{o}}_k = \sum_{t=1}^{T}\gamma_t(k)\mathbf{o}_t$.

A set of adaptation algorithms of different complexity can now be found by introducing different constraints on the affine transformation defined by A and $\mathbf{b}$. In the following sub-sections we will discuss four different variations, with the A matrix taking the structure of a full, block-diagonal, diagonal, and an identity matrix.

2.1 Full Rotation Matrix

With a full rotation matrix A, the solution to eq. 1 is given by (Leggetter and Woodland, 1995):

$$\mathbf{W}_i = \left[\sum_{k\in\Omega}\mathbf{R}_k(i)\bar{\mathbf{o}}_k(i)\hat{\boldsymbol{\mu}}_k'\right]\left[\sum_{k\in\Omega}\bar{\gamma}_k\mathbf{R}_k(i)\hat{\boldsymbol{\mu}}_k\hat{\boldsymbol{\mu}}_k'\right]^{-1} = \mathbf{z}_i\left[\mathbf{G}_i\right]^{-1},\ i = 1,\ldots,D, \tag{2}$$

where $\boldsymbol{W}_i$ is the i'th row of $\boldsymbol{W}$, and D is the feature vector dimension. We thus have to solve D systems of $D+1$ unknowns for each of the transformations.

2.2 Block-Diagonal Rotation Matrix

If we let $\boldsymbol{A}$ be block-diagonal with L blocks, the solution to eq. 1 is given by:

$$\mathbf{W}_i^{(l)} = \left[\sum_{k\in\Omega}\mathbf{R}_k^{(l)}(i)\bar{\mathbf{o}}_k^{(l)}(i)\hat{\boldsymbol{\mu}}_k^{(l)'}\right]\left[\sum_{k\in\Omega}\bar{\gamma}_k\mathbf{R}_k^{(l)}(i)\hat{\boldsymbol{\mu}}_k^{(l)}\hat{\boldsymbol{\mu}}_k^{(l)'}\right]^{-1} = \mathbf{z}_i^{(l)}\left[\mathbf{G}_i^{(l)}\right]^{-1},\ i = 1,\ldots,D_l \quad \text{and} \quad l = 1,\ldots,L, \tag{3}$$

where $\boldsymbol{W}_i^{(l)}$ is the i'th row of $\boldsymbol{W}^{(l)}$. $\boldsymbol{W}^{(l)}$ is the transform associated with the *l*'th stream of the feature vector, with a feature vector dimension of D_l. We thus have to solve D equation systems of D_l+1 unknowns for each of the transformations.

2.3 Diagonal Rotation Matrix

If we let $\boldsymbol{A}$ be a diagonal matrix, we can find the solution to eq. 1 using a similar approach to the one in (Myrvoll *et al.*, 2001):

$$\begin{array}{lclcl}\sum_{k\in\Omega}\bar{\gamma}_k\mathbf{R}_k(diag(\boldsymbol{\mu}_k))^2\mathbf{a} & + & \sum_{k\in\Omega}\bar{\gamma}_k\mathbf{R}_k diag(\boldsymbol{\mu}_k)\mathbf{b} & = & \sum_{k\in\Omega}\mathbf{R}_k diag(\boldsymbol{\mu}_k)\bar{\mathbf{o}}_k \\ \sum_{k\in\Omega}\bar{\gamma}_k\mathbf{R}_k diag(\boldsymbol{\mu}_k)\mathbf{a} & + & \sum_{k\in\Omega}\bar{\gamma}_k\mathbf{R}_k\mathbf{b} & = & \sum_{k\in\Omega}\mathbf{R}_k\bar{\mathbf{o}}_k,\end{array} \Leftrightarrow \begin{bmatrix}\mathbf{E} & \mathbf{G}\\ \mathbf{G} & \mathbf{H}\end{bmatrix}\begin{bmatrix}\mathbf{a}\\ \mathbf{b}\end{bmatrix} = \begin{bmatrix}\mathbf{z}_u\\ \mathbf{z}_l\end{bmatrix} \tag{4}$$

where $diag(\boldsymbol{\mu}_k)$ is a diagonal matrix with the elements of the mean vector $\boldsymbol{\mu}_k$ on the diagonal and $\mathbf{a}$ is a vector with the elements from the diagonal of the rotation matrix $\mathbf{A}$.

The solution for **a** and **b** is given by:

$$\begin{aligned}\mathbf{a} &= (\mathbf{G} - \mathbf{H}\mathbf{G}^{-1}\mathbf{E})^{-1}(\mathbf{z}_l - \mathbf{H}\mathbf{G}^{-1}\mathbf{z}_u)\\ \mathbf{b} &= \mathbf{G}^{-1}(\mathbf{z}_u - \mathbf{E}\mathbf{a}).\end{aligned} \tag{5}$$

Here, **E**, **G**, and **H** are all diagonal matrices, so that the equation system is easily solved element by element, with no need for matrix inversions.

2.4 Identity Rotation Matrix

If we let $\boldsymbol{A}$ be an identity matrix, the solution to eq. 1 can be further simplified. In this case the solution can be written:

$$\mathbf{b} = \left[\sum_{k\in\Omega} \bar{\gamma}_k \mathbf{R}_k\right]^{-1} \left[\sum_{k\in\Omega} \mathbf{R}_k \left(\bar{\mathbf{o}}_k - \bar{\gamma}_k \boldsymbol{\mu}_k\right)\right] = \mathbf{H}^{-1}\mathbf{z}\,. \tag{6}$$

The equation system described here is again easily solved element by element.

3. AN EFFICIENT HIERARCHICAL TRANSFORMATION SCHEME

The first step in a transformation based adaptation scheme is to align the adaptation data to the acoustic models, using either the forward/backward or Viterbi algorithm. As a result, the occupation count ($\bar{\gamma}_k$) and the weighted sum of the feature vectors aligned to a particular Gaussian ($\bar{\mathbf{o}}_k$) is found for each of the Gaussians in the model set. We call these entities the occupation statistics.

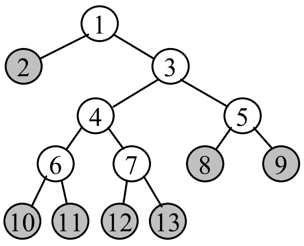

Figure 1. An example regression tree structure for grouping the Gaussian components of acoustic models. The grey nodes are base regression classes.

Given the occupation statistics, the next question is how to cluster the Gaussians into the regression classes used for adaptation. A common approach is to organise the Gaussians in a tree structure, a so-called regression class tree (Gales, 1996). A simple binary regression class tree is shown in figure 1. The tree contains a hierarchy of regression classes *1 – 13*. Classes *2* and *8-13* are also *base classes*, i.e., they have the individual Gaussians in the model set as members. The idea behind a tree of this kind is that on a certain level in the tree, all Gaussians belonging to the same node will have similar acoustic properties, and thus could share a transformation. Classes without transformations by default inherit the transformation of the parent class.

In the suggested approach, we first explicitly determine which regression classes should be used for transformation with a particular adaptation data set. The following criteria must be fulfilled, in order for a regression class to be used for transformation estimation:

a) The occupation count must exceed a threshold.
b) The number of member Gaussians with a non-zero occupation count must exceed a threshold.
c) At least one of the child nodes must fail one or both of the previous criteria.

Criterion (b) above is not used in other implementations that have been reported, but is used here in order to improve the numerical properties of the system. Without this criterion, the equation systems described in section 2 will often be ill-conditioned, and thus a numerically robust and computationally expensive solution will be required.

With all the above in place, the final step is to calculate the transformations for the appropriate regression classes in the tree. For this, we traverse the tree in a depth first, post-order, non-recursive fashion. This ensures that for every node that is being processed, all its child nodes will already have been processed. For the example tree shown in figure 1, such a node sequence is *2, 10, 11, 6, 12, 13, 7, 4, 8, 9, 5, 3,* and *1*.

From the equations in section 2, we see that for each of the transformation types, there are several matrices and vectors based on the occupation statistics and model parameters that are required for the transformation estimation. We call these the *intermediate matrices/vectors*. For the full transformation these are $\{\mathbf{z}_i, \mathbf{G}_i\}$, for a block-diagonal transformation $\{\mathbf{z}_i^{(l)}, \mathbf{G}_i^{(l)}\}$, for a diagonal transformation $\{\mathbf{E}, \mathbf{G}, \mathbf{H}, \mathbf{z}_u, \mathbf{z}_l\}$, and for the bias-only transformation $\{\mathbf{H}, \mathbf{z}\}$. By studying equations 2 - 6, we can see that the value of each of these matrices/vectors in a given node is simply the sum of the corresponding entities in all its child nodes. Thus, by traversing the tree in the suggested fashion, we ensure that the number of nodes that hold the intermediate matrices/vectors in memory at any instant of time is a maximum of one node per level in the tree. As soon as a node has been processed, the intermediate matrices/vectors are propagated up the tree to the parent node, unless it has already been established that no nodes further up the tree will require a transformation to be calculated. In any case, the memory used for these matrices/vectors in the current node can be freed.

4. COMPLEXITY ANALYSIS

As evident from the discussion in the previous two sections, the various constraints put on the affine transformation can lead to very different computational load. In this section we will analyse each of the four different transformation types discussed with respect to computational complexity and memory requirements.

There are three main components to the analysis. First of all the calculation of the occupation statistics will require a considerable amount of resources. We will assume here that the Viterbi algorithm is applied for this purpose. Note that the complexity of this procedure is not affected by the transformation configuration. The second component is the calculation, storage and propagation of the intermediate matrices/vectors. Obviously, the choice of transform type will influence the resources required for this to a large degree. The final component is the calculation and storage of the transformations themselves. Again, the transform type will influence the complexity.

The following notation is used for the remainder of this section:

- D: feature vector dimension
- L: number of blocks in the block-diagonal transformations (assumed equal-sized blocks)
- P: number of pdfs in the HMM set
- N: depth of the regression tree
- K: number of transforms used for a given amount of data
- T: number of adaptation data frames
- M: average number of components in the observation density GMMs
- S: average number of active states in the Viterbi-based statistics collection

Table 1 shows the dominant terms contributing to the number of multiplications/additions, as well as the number of elements that require storage during the transformation estimations. These numbers do not show an exact picture, but aim to give an understanding of the relationship between the complexities of the different parts of the algorithm. Note that the results shown in section 5 build on the underlying and more comprehensive analyses.

This table tells us that the computational complexity associated with the Viterbi algorithm will typically be smaller than the two other components if full transformation matrices are used, but this is gradually changed when the structure of the matrices are simplified, in particular because no matrix inversions are needed. With the simple transforms, the Viterbi algorithm will dominate the complexity. The Viterbi algorithm dominates the memory requirements in all the instances, but again the simpler transformations will lead to a reduction compared to the full and block-diagonal transformations. Note however that these effects are partially offset by the fact that when the transformation structure is being constrained, it is required to increase the number of transforms to maintain the performance.

Table 1. The dominant terms with respect to the number of multiplications and additions (#m/a) and elements requiring storage (#elm) for different transformation types.

	Full		Block-diagonal		Diagonal		Identity	
	#m/a	#elm	#m/a	#elm	#m/a	#elm	#m/a	#elm
Viterbi alignment	TSMD	*PD*	TSMD	*PD*	TSMD	*PD*	TSMD	*PD*
Intermediate matrices/vectors	PD^3	ND^2	$P(D/L)^3$	$N(D/L)^2$	PD	*ND*	PD	*ND*
Calculation of transformations	KD^4	KD^2	$K(D/L)^4$	$K(D/L)^2$	KD	*KD*	KD	*KD*

5. EXPERIMENTS

The experiments were performed on the 5000-word Wall Street Journal (WSJ) 1993 Spoke 3 task of non-native speakers. The adaptation was done in unsupervised mode with both monophone and tied-state triphone models. These models have 1920 and 10336 Gaussian components, respectively. Feature vectors comprising 12 MFCCs and normalised log-energy, augmented by the delta and delta-

delta parameters, were extracted for each 10 ms frame of speech. Cepstral mean subtraction was also performed. The grammar used was the back-off bigram from MIT Lincoln Laboratories.

The adaptation and test data were partitioned into development and evaluation sets, where the development set was used to decide on optimum parameter settings for each transformation configuration, including the regression tree size and the occupation count thresholds. The optimum settings found for these are shown in table 2. The results reflect that when the transformation structure is being constrained, the optimal number of transforms will indeed increase due to the larger tree sizes and decreased occupation count thresholds.

Table 2. Optimal parameter settings found from development set experiments.

	Number of leaf nodes		**Occupation count threshold**	
Type	***Monophone***	***Triphone***	***Monophone***	***Triphone***
Full	64	128	500	750
Block diagonal	128	128	250	250
Diagonal	1024	1024	40	30
Bias only	1920	4096	10	10

The evaluation set was used to perform comparisons between the optimised configurations. Adaptation data comprised sets of 1, 3, 5, 10, 20 and 40 utterances, and testing was performed on a separate set of 40 utterances. The results were averaged across 10 speakers. The results from these experiments are shown in figure 2, and confirm that the performance of each configuration is remarkably similar. Recognition accuracy varies between different transform configurations by less than 1.5% and 2% for monophones and triphones, respectively. From these results, it is evident that simpler transform configurations such as diagonal and bias only transforms can achieve close to the same recognition accuracy as a conventional full transform if the number of transformations is increased to a sufficient level.

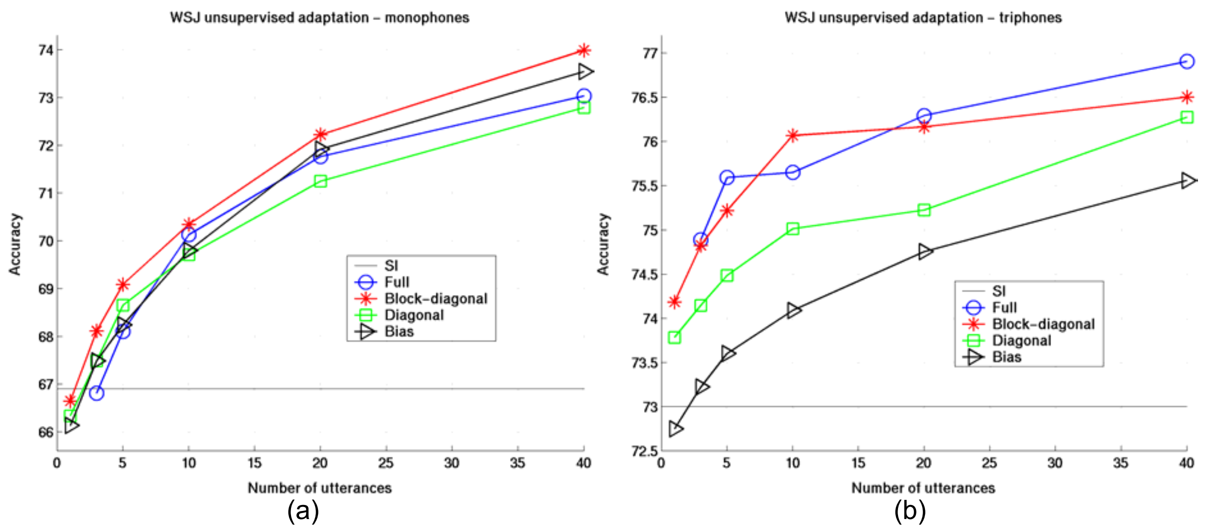

Figure 2. Average recognition accuracy vs. number of utterances with monophone (a) and triphone (b) models for different transform types.

Figure 3 shows the results from a complexity analysis of the triphone experiments. It shows that despite the increased number of transformations, a clear advantage is maintained for the simple transformations in terms of the number of additions and multiplications. In fact, the computational complexity is almost entirely dominated by the Viterbi alignment for the diagonal and bias transforms.

In terms of memory, the picture is slightly different, as the differences between the transform types are much smaller. The bias transformation is actually found to have larger memory requirements than all the other configurations in some instances. The block-diagonal transform is found to be the most efficient in terms of memory requirements. This is because the block-diagonal structure used here exploits the fact that there is little cross-correlation between the streams (in this case, between the static, delta, and delta-delta parameters) of the feature vector, thus reducing the overall number of transformation parameters required. The diagonal and bias transforms fail to exploit the correlation *within* a stream, thus increasing the number of transformation parameters required in these cases.

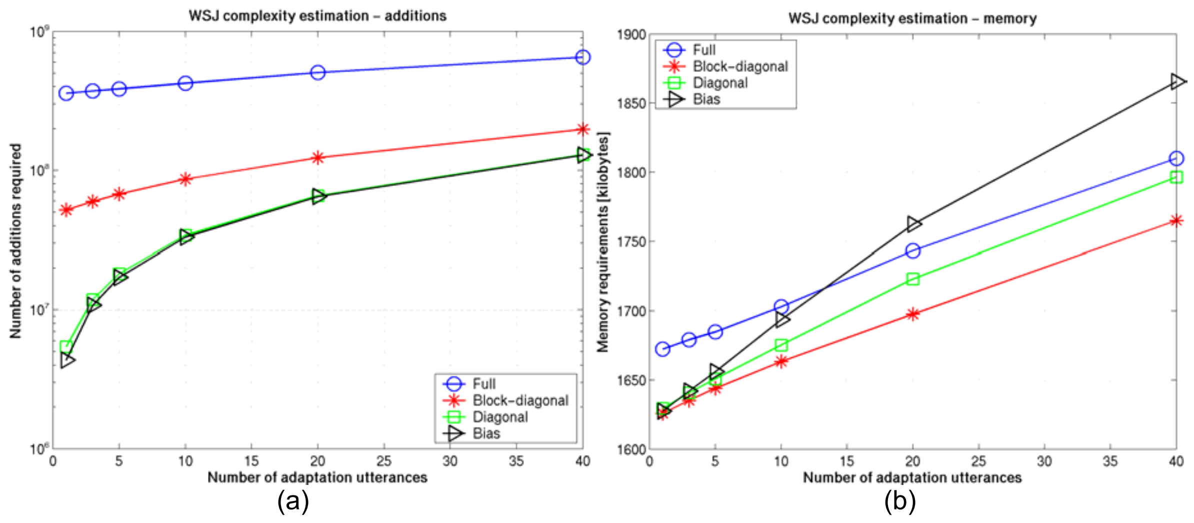

Figure 3. The number of additions (a) and the memory requirements (b) for transformation estimation with triphone models for different transform types.

In contrast to previous findings (Leggetter and Woodland, 1994), the diagonal transform was found to allow the computational complexity to be reduced by one to two orders of magnitude relative to conventional methods, while preserving a similar recognition accuracy. This is also accompanied by a reduction in the memory requirements. A further disadvantage of the full and block-diagonal configurations is the difficulty of implementing accurate matrix inversions in fixed-point arithmetic.

6. CONCLUSION

Results of the performance tests show that with optimum parameter choices, low-complexity transformation configurations are capable of producing accuracies virtually identical to those of high-complexity transformation configurations. Thus, computational complexity can be reduced by one to two orders of magnitude relative to more conventional speaker adaptation methods, while preserving the same recognition accuracy.

ACKNOWLEDGEMENTS

The authors wish to express their thanks to Dr M. Thomson for his guidance and helpful discussions.

REFERENCES

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

Leggetter, C. J., and Woodland, P. C., "Speaker adaptation of HMMs using linear regression" *Technical Report, CUED/F-INFENG/TR.181, Cambridge University Engineering Department*, 1994.

Leggetter, C. J., and Woodland, P. C., "Maximum likelihood linear regression for speaker adaptation of continuous density HMMs," *Computer Speech and Language*, vol. 9, 1995, pp. 171-185.

Myrvoll, T. A., Paliwal, K. K., and Svendsen, T., "Fast adaptation using constrained affine transformations with hierarchical priors", in *Proc. Eurospeech*, vol. 2, 2001, pp. 1233-1236.