THE GENERATION AND USE OF REGRESSION CLASS TREES FOR MLLR ADAPTATION

M.J.F. Gales

CUED/F-INFENG/TR263

August 1996

Cambridge University Engineering Department
Trumpington Street
Cambridge CB2 1PZ
England

Email: mjfg@eng.cam.ac.uk

Abstract

Maximum likelihood linear regression (MLLR) is an adaptation technique suitable for both speaker and environmental model-based adaptation. The models are adapted using a set of linear transformations, estimated in a maximum likelihood fashion from the available adaptation data. As these transformations can capture general relationships between the original model set and the current speaker, or new acoustic environment, they can be effective in adapting all the HMM distributions with limited adaptation data. Two important decisions that must be made are (i) how to cluster components together, such that they all have a similar transformation matrix, and (ii) how many transformation matrices to generate for a given block of adaptation data. This paper addresses both problems. Firstly it describes two optimal clustering techniques, in the sense of maximising the likelihood of the adaptation data. The first assigns each component to one of the regression classes. This may be used to generate standard regression class trees. The second scheme performs a *fuzzy* assignment of base class to regression class, so the transformation associated with each component is a linear combination of a set of transformations. Secondly two schemes are examined which address the problem of how to determine the number of regression classes, transforms, for a given amount of adaptation data. Two schemes are examined here. A cross-validation scheme based on the auxiliary function of the adaptation data is described. Another scheme based on the use of iterative MLLR is also detailed. Both these schemes require no a-priori thresholding information. An initial evaluation of the techniques was performed using data from the ARPA 1994 test data. On this task, though "good" trees, in terms of the likelihood of the adaptation training data were generated, neither of the optimal clustering schemes yielded gains in recognition performance. The performance of the cross-validation scheme was found to be comparable to an empirically determined threshold scheme. The best performance was achieved using iterative MLLR, which outperformed both fixed classes and threshold based schemes.

Contents

1 Introduction

Current state-of-the-art speaker independent (SI) speech recognition systems are capable of achieving impressive performance in clean acoustic environments for speakers that are well represented in the training data. However, performance can be relatively poor for some speakers e.g. for non-native speakers using a system trained on speech from native speakers. Furthermore, the performance degrades, often dramatically, if there is some mismatch between the training and test data acoustic environments. For complex speech recognition systems a large amount of data is required to retrain the system for a particular speaker or acoustic environment. Hence, it is very desirable to be able to improve the performance of an existing system while only using a small amount of speaker-specific or environment-specific adaptation data.

Some environmental adaptation techniques require no speech data in the new acoustic environment to adapt the model parameters [3, 12], only noise samples. However these schemes make assumptions about the form of the acoustic environment. Other techniques can only update distributions for which observations occur in the adaptation data, such as those using maximum a-posteriori (MAP) estimation [5, 6]. These require a relatively large amount of adaptation data to be effective. Another approach is to estimate a set of transformations that can be applied to the model parameters. If these transformations can capture general relationships between the original model set and the current speaker or new acoustic environment, they can be effective in adapting all the HMM distributions. One such transformation approach is maximum likelihood linear regression (MLLR) [8, 9] which estimates a set of linear transformations for the mean parameters of a mixture Gaussian HMM system, such that the likelihood of the adaptation data is maximised. As many components are assumed to share the same transformation, it is possible to adapt all the components of recognition system with little data. It should be noted that while MLLR was initially developed for speaker adaptation, since it reduces the mismatch between a set of models and adaptation data it can also be used to perform environmental compensation by reducing a mismatch due to channel or additive noise effects.

In this paper two problems associated with MLLR adaptation are examined. The first problem is to decide how components should be clustered together, such that they all have a similar transformation matrix. The second problem is how to decide how many transformations to generate, given a particular set of adaptation data. Initially the paper describes the basic theory of maximum likelihood linear regression. The two clustering schemes considered, a *hard* clustering scheme and a *fuzzy* clustering scheme, are described. The use of cross-validation and iterative MLLR to determine the optimal number of transforms is then discussed. Finally some initial experiments with simple regression class generation are described and the various transformation selection techniques investigated.

2 Maximum Likelihood Linear Regression

MLLR is a technique for finding the optimal, in the sense of maximising the likelihood of the adaptation data, linear transformation of the model parameters to represent the adaptation data. In general there will be little adaptation data compared to the number of model parameters. Hence it is necessary to cluster model parameters together into *regression classes*. It is assumed that all components in a given regression class transform in a similar fashion.

2.1 Maximum likelihood linear regression

The new estimate of the mean, $\hat{\mu}$, is found by

$$\hat{\mu} = \hat{\mathbf{W}}\xi \tag{1}$$

where $\hat{\mathbf{W}}$ is the $n \times (n+1)$ transformation matrix (n is the dimensionality of the data) and ξ is the extended original mean vector

$$\xi = \begin{bmatrix} 1 & \mu_1 & \dots & \mu_n \end{bmatrix}^T \tag{2}$$

It is simple to see that

$$\hat{\mathbf{W}} = [\ \hat{\mathbf{b}} \quad \hat{\mathbf{A}}\ ] \tag{3}$$

where $\hat{\mathbf{b}}$ is a bias on the mean and $\hat{\mathbf{A}}$ is a transformation matrix, which may be full, block diagonal, or diagonal. The aim is to find the transformation $\hat{\mathbf{W}}$ that maximises the likelihood of the adaptation data. This optimisation was originally described in [7] and is described in appendix A.

2.2 Regression classes

As previously described all components associated with a particular regression class are assumed to transform in a similar fashion, ie $\hat{\mathbf{W}}$ is the same for all components. These regression classes may be pre-determined and fixed prior to adaptation. This will be referred to as *fixed* regression classes. Alternatively the regression classes may be determined dynamically according to the amount of adaptation data available using a *regression class tree*.

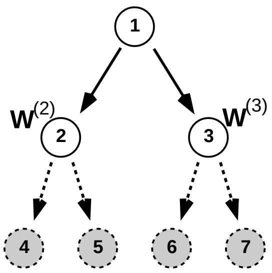

Figure 1: A binary four base class regression class tree

A simple binary regression class tree with four *base classes*[1] is shown in figure 1. A regression class tree, T, consists of a hierarchy of regression classes, $\{r_1, r_2, r_3\}$ and a set of base classes, $\{c_4, c_5, c_6, c_7\}$. These base classes may also be regression classes. When building a regression class tree, it is necessary to assign base classes to regression classes according to some appropriate criterion. Currently the regression class tree clustering is normally based on one of two schemes.

1. **Phonetic knowledge**. Here expert knowledge is used to decide which components are to be transformed together. For example, the components may be split according to broad classes, *nasal*, *glide*, or at a lower level into *phones*.

2. **Acoustic space**. Components are clustered according to how close they are in acoustic space, irrespective of which phone they belong to. This has the advantage of being a "data-driven" approach with no need for expert knowledge.

Little difference between the two techniques has been observed on relatively small tasks [7].

When using a regression class tree, there will typically be different amounts of adaptation data for each speaker. There needs to be some method of deciding how far down the regression class tree to go. For example, in the diagram a solid arrow and class indicates that there is "sufficient" data for a transformation matrix to be generated using the data associated with that class, a dotted line and class indicates insufficient data. Previously the concept of "sufficient" data has been determined by using an empirically set threshold for the minimum number of frames to robustly estimate a transformation [7]. Such a scheme is described below.

A simple top-down scheme starts the search at the root node (assuming that the root node has sufficient data) and then:

[1] A base class is the lowest clustering of components that may be independently transformed. In the limit, of course, there is a separate base class per-Gaussian. However this requires a vast amount of adaptation data to be used.

For a node A
if A has children B and C
if B has sufficient data
search through node B
else
let A be the regression class for base classes in B
if C has sufficient data
search through node C
else
let A be the regression class for base classes in C
else
let A be the regression class for base classes in B and C

This will be referred to as the *threshold* stopping criterion.

3 Optimal Clustering Schemes

Neither of the clustering techniques described earlier are necessarily optimal, in the sense of maximising the likelihood of the adaptation data. The first scheme using phonetic knowledge assumes that all components of a particular phonetic unit all transform in a similar fashion. Alternatively in the acoustic space scheme, the components that are "close" in acoustic space transform the same. It is felt that neither is necessarily true. To achieve an optimal[2] clustering it is necessary to find

$$\hat{T} = \arg\max_{T} \left\{ \sum_{s=1}^{S} \mathcal{Q}^{(s)}(\mathcal{M}, \hat{\mathcal{M}}|T) \right\} \tag{4}$$

where

$$\begin{aligned} &\mathcal{Q}^{(s)}(\mathcal{M}, \hat{\mathcal{M}}|T) = \\ &\quad K_1 - \frac{1}{2}\mathcal{L}(\mathbf{O}_T|\mathcal{M}) \sum_{m=1}^{M} \sum_{\tau=1}^{T} L_m(\tau) \left[K_m + \log(|\mathbf{\Sigma}_m|) + (\mathbf{o}(\tau) - \hat{\mu}_m)^T \mathbf{\Sigma}_m^{-1} (\mathbf{o}(\tau) - \hat{\mu}_m) \right] \end{aligned} \tag{5}$$

and $\hat{\mu}_m$ is the new estimate of the mean obtained using the transformation obtained using regression tree T. Though it is not normally feasible to obtain the globally optimal tree, it is possible to guarantee that at each split a locally optimal split is made. Moreover, schemes based on this approach are only guaranteed to obtain a local maximum auxiliary function value at each split. Two possible schemes are described below.

3.1 Hard clustering

Hard clustering requires a unique assignment of a base class to one regression class. This is the standard MLLR scheme, so the new mean is found using equation 1. The problem is how to optimally assign a particular base class, c, to one of the possible regression classes. Consider the regression class tree shown in figure 1. There are two transformations, $\mathbf{W}^{(2)}$ and $\mathbf{W}^{(3)}$ associated with regression classes r_2 and r_3 respectively, and an initial assignment of base classes c_4 and c_5 to regression class r_2, and c_6 and c_7 to regression class r_3. The aim is to rearrange the base class to regression class assignment to maximise the likelihood of the training adaptation data given the current estimate of transforms $\mathbf{W}^{(2)}$ and $\mathbf{W}^{(3)}$.

[2]Wherever the term optimal is used in this paper it means optimal in the sense of maximising the likelihood of the adaptation data.

To optimise the hard clustering, an initial clustering is assumed, typically from acoustic clustering. For each base class, c, the assignment to regression class r, with associated transformation $\mathbf{W}^{(r)}$, is determined by

$$\hat{r}^{(c)} = \arg\max_{r} \left\{ \sum_{s=1}^{S} \mathcal{Q}_c^{(s)}(\mathcal{M}, \hat{\mathcal{M}}|r) \right\} \tag{6}$$

where

$$\begin{aligned} &\mathcal{Q}_c^{(s)}(\mathcal{M}, \hat{\mathcal{M}}|r) = \\ &\quad K_1 - \frac{1}{2}\mathcal{L}(\mathbf{O}_T^{(s)}|\mathcal{M}) \sum_{m=1}^{M_c} \sum_{\tau=1}^{T^{(s)}} L_m(\tau) \left[K_m + \log(|\mathbf{\Sigma}_m|) + (\mathbf{o}(\tau) - \hat{\mu}_m^{(r)})^T \mathbf{\Sigma}_m^{-1} (\mathbf{o}(\tau) - \hat{\mu}_m^{(r)}) \right] \end{aligned} \tag{7}$$

and

$$\hat{\mu}^{(r)} = \mathbf{W}^{(r)}\xi = \mathbf{A}^{(r)}\mu + \mathbf{b}^{(r)} \tag{8}$$

and $\hat{r}$ is the new estimated regression class for base class c. This is performed for all base classes.

This process is guaranteed not to decrease the overall auxiliary function value, as each base class will only swap regression class if it increases the auxiliary function. If the regression class has changed for any of the base classes, it is necessary to re-estimate the transforms, $\mathbf{W}^{(r)}$. This is performed using the standard MLLR estimation formulae described in appendix A and is again guaranteed to increase the auxiliary function. This process may then be repeated until none of the base classes changes regression class. At each iteration the auxiliary function value is guaranteed to increase, eventually arriving at a local maximum.

3.2 Fuzzy clustering

A hard clustering scheme does not make optimal use of the transformation matrices generated. Where few transformations are generated, it is unlikely that all base classes will be exclusively assigned to one transform. Hence the use of a *fuzzy clustering* scheme is proposed. Here, each base class makes use of all the transforms available, using a weighted sum to form the complete transformation.

For a fuzzy grouping the following transformation of the mean is used

$$\hat{\mu} = \left[\sum_{p=1}^{P} \gamma_p \mathbf{W}^{(p)} \right] \xi = \sum_{p=1}^{P} \gamma_p \hat{\mu}^{(p)} \tag{9}$$

This is a simple extension of the hard clustering scheme where

$$\gamma_p = \begin{cases} 1, & \text{base class in question is in regression class } p \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

An example of fuzzy clustering is shown in figure 2. Again, a solid arrow and class indicates that there is sufficient data for a transformation matrix to be generated using the data associated with that class, a dotted line and class indicates insufficient data. There are thus three transformation generated, $\mathbf{W}^{(2)}$, $\mathbf{W}^{(3)}$, and $\mathbf{W}^{(4)}$. It is assumed that there are sufficient frames assigned to each of the regression base classes to estimate the smoothing parameters, so each of the base classes has three "weights" associated with it.

Once the transforms are generated the regression class tree disappears and may be replaced by figure 3. For very small regression class trees and fixed regression classes this scheme is fine. However, as larger regression class trees are used the description of the regression class tree becomes very complicated. The weights are a function of all the transformations generated. Thus for each possible combination of transforms a different set of weights must be generated For example in the 3-level binary regression tree illustrated in figure 2 there are 16 possible transformation sets for each

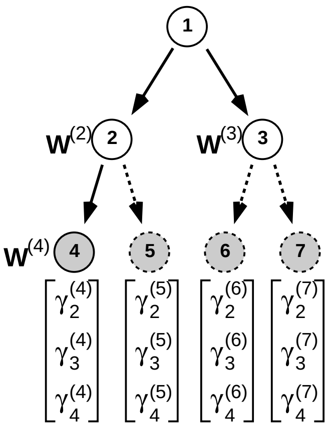

Figure 2: A binary four base class regression class tree with smoothing parameters

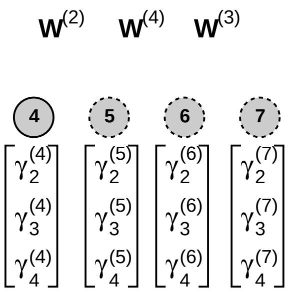

Figure 3: A flattened regression class tree with smoothing parameters

of the base classes. These must all be stored and different transforms calculated for each possible set of weights if multiple iterations of fuzzy clustering are used. When there are no constraints on the sets of transforms that may be used this will involve a very large computational overhead. It is therefore preferable to reduce the range of possible weights. The simplest constraint is to allow a base class to only use a limited number of transforms selected in the tree, or to use pre-defined regression classes not a regression class tree[3].

When fuzzy clustering is used there are far more parameters in the regression class tree. However these parameters are learnt from the training data, not at run-time. It is therefore valid to compare systems with fuzzy and hard clustering.

The estimation task is to find the ML estimate of the set of weights associated with each base class, c, $\gamma^{(c)}$. This weights vector is obtained as

$$\hat{\gamma}^{(c)} = \arg\max_{\gamma} \left\{ \sum_{s=1}^{S} \mathcal{Q}_c^{(s)}(\mathcal{M}, \hat{\mathcal{M}}|\gamma) \right\} \tag{11}$$

where

$$\mathcal{Q}_c^{(s)}(\mathcal{M}, \hat{\mathcal{M}}|\gamma) = \tag{12}$$
$$K_1 - \frac{1}{2}\mathcal{L}(\mathbf{O}_T^{(s)}|\mathcal{M}) \sum_{m=1}^{M_c} \sum_{\tau=1}^{T^{(s)}} L_m(\tau) \left[K_m + \log(|\mathbf{\Sigma}_m|) + (\mathbf{o}(\tau) - \hat{\mu}_m)^T \mathbf{\Sigma}_m^{-1} (\mathbf{o}(\tau) - \hat{\mu}_m) \right]$$

and

$$\hat{\mu} = \sum_{p=1}^{P} \gamma^{(c)} \mathbf{W}^{(p)} \xi \tag{13}$$

Note the transformation weights are not constrained to be positive, nor are they required to sum to one. In appendix C, the expression for the ML estimate of the weight vector $\gamma^{(c)}$ is obtained and is found to be a unique solution. For the same reason as the hard clustering scheme, the transformations may be re-estimated using the new weights vectors. The standard formulae are no longer applicable, but in appendix B it is shown how to obtain an ML estimate of a set of transforms given a set of weights. This whole process may then be repeated until the base class weights remain unaltered or a maximum number of iterations performed.

3.3 Practical considerations

There are a number of factors which need to be considered when building trees to maximise the auxiliary function given in equation 4.

1. **Number of base classes**. As previously described the base classes are the smallest collection of components that may be adapted independently. In the limit each base class will consist of an individual component. It is preferable to have as many base classes as possible in the system as this allows the greatest flexibility in building the regression class trees. However as the number of base classes increases, so the amount of tree-building data required increases in order to make a robust estimate of the base class to regression class assignment.

2. **Number of speakers**. It is, of course, desirable to have as many speakers as possible in the tree-building data. However, this must be offset by the need to robustly estimate the transformation matrices for each speaker.

3. **Depth of tree**. Both the above aspects become more important as the depth of tree, or number of tree nodes, increases. The deeper the tree the larger the amount of tree-building data required.

[3] If no constraints are placed on the fuzzy clustering then this is a globally optimal regression tree.

4. **Transformation form**. There are many forms that the transformations can take, for example diagonal, block diagonal or full [10]. The optimal regression tree will be dependent on the type of transform being considered. The more complex the transformation, the longer it takes to train the regression tree. In addition, more data from each speaker is required to robustly estimate the transform.

4 Run-Time Fuzzy Clustering

The use of fuzzy clustering during the generation of regression class trees has been described. An alternative approach is to calculate a set of weights for a particular speaker given a set of transforms calculated on the adaptation data for that speaker. This has the advantage that a specific fuzzy clustering is performed for each speaker, rather than averaged over all training speakers. However, it is now also necessary to ensure that the set of weights for each speaker is robustly estimated. This again may be achieved using a regression class tree. Note less data is required to achieve this than to estimate the transforms themselves. This is a disadvantage of the run-time fuzzy clustering as the depth to which it is sensible to go in the regression class tree is a function of the amount of adaptation data. This is in contrast to using fuzzy clustering to generate the original regression class tree, where each of the base classes has a different weight vector associated with it. In the same way as the global class weight estimation, this may be iterated with new transforms estimated, then new weights estimated. In practice this was found to give yield difference in performance and is not considered in this work.

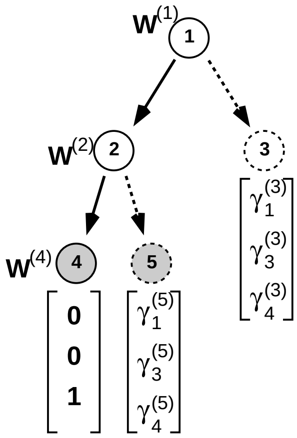

Figure 4: A binary four base class regression class tree with smoothing parameters

Figure 4 show the use of a regression tree in such a case. Here a dotted line indicates that there is insufficient data to calculate a transform, but sufficient to calculate a set of weights. It is interesting to note that the weights associated with base-class 4 are $\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$ as there is a specific transformation matrix associated with that base class and the transforms are not iteratively estimated.

5 Stopping Criterion

If the regression classes are not fixed a-priori, it is necessary to choose an appropriate stopping criterion. When building regression class trees using either phonetic knowledge or acoustic clustering it is possible to build very large trees with base classes consisting of individual components. As previously mentioned, there is usually insufficient data to estimate a transform for each component. According to the CART literature [1], the choice of stopping criterion is very important. Previous work [7] has used a minimum number of frames (see section 2.2). This requires the "learning" of an appropriate threshold, which varies according to the nature of the transformation. Two alternative schemes are described here. The first scheme uses cross-validation techniques [1] to decide on the regression classes to use. The second is primarily geared to static unsupervised adaptation tasks and uses an iterative MLLR scheme.

5.1 Cross-validation

A simple V-fold cross-validation scheme is considered here. Let the adaptation data for a given speaker, $\mathbf{O}$, be divided into V subsets, with the proviso that all the data from one utterance is placed in the same subset[4], $\mathbf{O}_v$, $v = 1, \ldots, V$. Define the v^{th} learning sample as

$$\mathbf{O}^{(v)} = \mathbf{O} - \mathbf{O}_v \tag{14}$$

A simple top-down scheme starts the search at the root node with $\mathbf{W}_p^{(v)} = \mathbf{I}$[5] and then:

For a node A

    generate V transforms $\mathbf{W}_a^{(v)}$ *using* $\mathbf{O}^{(v)}$ *for A*

    if $\left(\sum_{v=1}^{V} \mathcal{G}(\mathbf{O}_v|\mathcal{M}, \mathbf{W}_a^{(v)}) > \sum_{v=1}^{V} \mathcal{G}(\mathbf{O}_v|\mathcal{M}, \mathbf{W}_p^{(v)})\right)$

        if A has children B and C

            search through node B with $\mathbf{W}_p^{(v)} = \mathbf{W}_a^{(v)}$

            search through node C with $\mathbf{W}_p^{(v)} = \mathbf{W}_a^{(v)}$

        else

            let A be the regression class for base classes in A

    else

        let parent of A be the regression class for base classes in A

where

$$\mathcal{G}(\mathbf{O}_v|\mathcal{M}, \mathbf{W}_a^{(v)}) = \sum_{c=1}^{C_a} \sum_{m=1}^{M_c} \sum_{\tau=1}^{T_v} L_m(\tau) \left(K_m - \frac{1}{2} \left(\mathbf{o}(\tau) - \hat{\mu}_m\right)^T \mathbf{\Sigma}_m^{-1} \left(\mathbf{o}(\tau) - \hat{\mu}_m\right) \right) \tag{15}$$

and

$$\hat{\mu}_m = \mathbf{W}_a^{(v)} \xi_m \tag{16}$$

This should generate an "honest" set of regression classes [1]. However there are practical difficulties. It is preferable to make V as large as possible, as when generating $\mathbf{W}_a^{(v)}$ only $(V-1)/V$ of the total adaptation data is available for estimating the transform. However, the upper limit on V is the number of sentences, or sub-sentence blocks, in the adaptation data. As V increases there is an approximately linear increase in both computation and memory requirements, as V transforms must be generated at each node. If full transformation matrices are used this can be computationally very expensive. Furthermore, the statistics required to calculate equation 15 is the sum and sum squared of the observations for each subset for each component when calculated directly. This load can be reduced by storing statistics at the base class level [4].

The use of cross validation techniques greatly increases the computational and memory requirements for adaptation. As such it is most suited to static adaptation tasks, though may be used for incremental tasks.

[4] This is preferable due to the correlation between frames. It is feasible to break the data into blocks, smaller than utterances, but this has not been examined.

[5] Here $\mathbf{I}$ is the extended identity transformation, thus $\mathbf{I}\xi = \mu$.

5.2 Iterative MLLR

An alternative scheme for selecting regression classes is to use iterative MLLR. Originally the technique was used to provide improved transcriptions for adaptation [14]. Here it is used to make the best use of the adaptation data and regression class tree. It relies on the fact that when transformations are not robustly estimated, they will "learn" the transcription of the adaptation data. By always using the the best, latest, transcription, it is possible to overtrain[6] the transforms without degrading performance.

The procedure is:

1. Starting at the root node of the regression class tree and the speaker-independent transcription for the current adaptation data.

2. Given the current hypothesised transcription and regression class tree estimate a new set of transforms.

3. Generate a new hypothesised transcription given the current transformation set and adaptation data.

4. If the new hypothesised transcription is different to the previous transcription go down one level of the regression class tree and goto step 2.

This scheme is guaranteed to overtrain the regression transforms. The main problem with the scheme occurs when the tree is very unbalanced, ie some regression classes at a particular level of the regression class tree have very little adaptation data assigned to them compared to others in that level. This results in some transforms being non-robustly estimated very early in the iterative MLLR process, which may be detrimental to the overall performance of the scheme.

This scheme is only applicable to unsupervised adaptation tasks. Furthermore, as multiple iterations are required it is more suited to static adaptation tasks than incremental adaptation schemes.

6 Results

6.1 Recognition system

The baseline system used for the recognition task was a gender-independent cross-word-triphone mixture-Gaussian tied-state HMM system. This was the same as the "HMM-1" model set used in the HTK 1994 ARPA evaluation system [13]. The speech was parameterised into 12 MFCCs, C_1 to C_{12}, along with normalised log-energy and the first and second differentials of these parameters. This yielded a 39-dimensional feature vector. The acoustic training data consisted of 36493 sentences from the SI-284 WSJ0 and WSJ1 sets, and the LIMSI 1993 WSJ lexicon and phone set were used. The standard HTK system was trained using decision-tree-based state clustering [15] to define 6399 speech states. A 12 component mixture Gaussian distribution was then trained for each tied state, a total of about 6 million parameters. For the H1 task a 65k word list and dictionary was used with the trigram language model described in [13]. All decoding used a dynamic-network decoder [11] which can either operate in a single-pass or rescore pre-computed word lattices.

For the secondary channel experiments a PLP version of the standard MFCC models were built using single-pass retraining [3] on the secondary channel training data.

The training data used to generate the regression class trees were also based on the SI-284 WSJ0 and WSJ1 sets, or subsets thereof. Where a subset of the speakers were used these were selected at random. Where a subset of the sentences for each speaker was used, these were selected as the

[6] Here the term overtraining means that the transforms will not generalise. Thus the transforms will map an observed component to be "close" the adaptation data for that component, but any unobserved components in the same regression class will be poorly transformed. The effect of this is that the likelihoods for the observed components become very large compared to the unobserved components. This means that the adaptation transcription will be "learnt" by the transforms.

first n sentences from each speaker. All trees were generated using a block diagonal transformation with individual blocks for the static, delta and delta-delta parameters. The initial regression trees were generated using clustering in acoustic space. For all experiments a set of 750 base classes, excluding silence, were used.

All recognition tests were carried out on the 1994 ARPA Hub 1 and S5 evaluation data. The H1 task is an unlimited vocabulary task with approximately 15 sentences per speaker. The data was recorded in a clean[7] environment. The S5 task is an unknown microphone task with a 5k word vocabulary. All the experiments described here were carried out in an unsupervised static mode with the SI recognition transcription used for the initial alignments. This was not acceptable for the actual evaluation, but was felt to allow better contrasts as the same alignments can be used for all schemes.

6.2 Clustering results

All the experiments described in this section only consider the initial split from a single, global, regression class to two regression classes. This is felt to be the split which should have greatest affect on the recognition performance.

Iteration	Aux. Func.	Swap (%)
0	-61.542	—
1	-61.508	5.7
2	-61.493	9.3
3	-61.486	11.5
4	-61.480	12.9
5	-61.475	14.1
9	-61.463	17.2

Table 1: Swap (%) and auxiliary function value for hard clustering with 284 speakers and 25 sentences per speaker (run to convergence) on the training data

Table 1 shows the auxiliary function and the number of base classes that swapped regression class at each stage for a hard clustering scheme. Here the clustering was run to convergence, where a total of 17.2% of the base classes have changed transform. However the increase in auxiliary function value was small, an average increase of 0.079 in auxiliary function value (or 1.082 times higher).

Clustering	Aux. Func.
—	-65.349
Global	-63.228
Acoustic	-62.995
Hard	-62.910
Hard (con)	-62.892
Fuzzy	-62.801

Table 2: Auxiliary function value on the H1 evaluation data using hard and fuzzy clustering schemes for the first level regression class tree split

The results shown in table 1 are from the training data. Table 2 shows the performance, in terms of average auxiliary function, on an unseen test set, the 1994 H1 eval data. *Global* and *Acoustic*

[7] Here the term "clean" refers to the training and test conditions being from the same microphone type with a high signal-to-noise ratio.

are the performance of an acoustically clustered regression tree generating 1 and 2 transforms, respectively. For all the optimal schemes 25 sentences from all 284 speakers were used to generate the regression class trees. Normally five clustering iterations were used. The table shows the gain in using the standard two transform system over a single transform. Surprisingly there is only a very small gain in auxiliary function, 0.223. The gain obtained using hard clustering is a further 0.085 (similar to the gains on the training data). By running this clustering to convergence a gain of 0.103 was obtained, *Hard (con)*. This increase in auxiliary function values was observed for all the speakers in the data. Greater gains were obtained using the fuzzy clustering scheme, *Fuzzy*. Here the auxiliary function rose by 0.194, getting towards the gains obtained by using two transforms instead of one. This shows that extensive use was being made of the fuzzy clustering.

Clustering	Avg. Prob.	Error Rate (%)
—	-68.789	9.20
Global	-66.762	8.30
Acoustic	-66.536	8.21
Hard	-66.451	8.22
Fuzzy	-66.348	8.23

Table 3: Error Rate (%) on the H1 evaluation data using hard and fuzzy clustering for the first level regression class tree split

Having established that the use of either fuzzy or hard clustering satisfied the criterion for a "good" clustering scheme, the actual recognition performance on the 1994 ARPA H1 evaluation test set was examined. Table 3 shows the performance of the various schemes and the average probability per frames of the test data. The performance was not as expected. Comparing the single transform standard system and the two transform standard system, there was a marked increase in probability of the test data (not surprising as this was the adaptation data). However, only a slight increase in recognition performance was obtained. This was surprising as the transforms generated were felt to be well trained as the minimum number of frames assigned to the transform was over 3300 and a block diagonal transform was being used.

The performance of the hard clustering scheme was also surprising. The average probability of the test data increased, again not surprising as the auxiliary function on the adaptation data was greater. However the recognition performance is no better, 8.22% compared to 8.21%, though the probabilities are higher as expected. Similarly the fuzzy clustering scheme, though increasing the auxiliary function, gave no gains in recognition performance. From these results the use of various clustering criteria may alter the auxiliary function value, it has little affect on the recognition performance.

Clustering	Error Rate (%)	
	H1 Dev	H1 Eval
—	9.57	9.20
Acoustic	8.39	8.21
Run-Time	8.27	8.21

Table 4: Error Rate (%) on 1994 H1 development and evaluation data using run-time fuzzy clustering

Run-time fuzzy clustering using the original first level acoustic clustering and the standard acoustic regression class tree to determine on the number of weight vector to generate, was performed. The thresholds used to determine the number of weight vectors were "reasonable" values determined on other test sets. The results are shown in table 4 On the H1 development data a slight gain in performance was observed. However, on the evaluation data no gain in performance was seen.

6.3 Stopping criterion results

For all the experiments in this section an acoustically clustered regression class tree was used. For the S5 unknown microphone task an additional binary split in the regression class tree was added, so that silence could be adapted as a separate regression class from the speech (provided sufficient data, where appropriate, was available).

Stopping Criterion	Reg. Classes		Error Rate (%)	
	H1 Dev	H1 Eval	H1 Dev	H1 Eval
—	—		9.57	9.20
Fixed	1		8.49	8.30
	2		8.39	8.21
	4		8.59	8.11
Threshold	10.05	11.00	8.63	8.02
X-val.	11.95	13.25	8.72	8.07

Table 5: Error Rate (%) on 1994 H1 development and evaluation data using fixed regression classes and threshold and cross validation stopping criteria

Various stopping criteria were considered and the results are shown in table 5. *Fixed* indicates using a fixed number of classes. These fixed regression classes were selected by using one level of the acoustically clustered regression class tree. Thus four regression classes corresponds to the third level of the regression class tree. The threshold stopping criterion, *Threshold*, used a minimum number of frames per transform constraint, the threshold for these experiments were "reasonable" values derived from previous experiments. Finally the cross-validation criterion, *X-val*, used cross validation with a 6-fold cross-validation scheme.

The performance on the H1 development data was not as expected. The best performance was obtained using the 2 fixed regression classes. The performance of both the thresholding scheme and the cross-validation scheme on this test set were similar, but 3-4% worse than the 2 fixed regression class case. On the H1 evaluation data, the recognition performance increased as the number of fixed classes increased. It is interesting that despite both tasks being very similar and the amount of adaptation data approximately the same the best fixed class schemes were very different for the evaluation and development data. This shows one of the problems of using MLLR, that the optimisation criterion that of ML is not directly related to error rate for the speech recognition task. The performance of the cross-validation scheme was comparable with that of the empirically set threshold scheme and yielded a similar average number of transformations. However the advantage of the cross-validation scheme is that there is no need to empirically estimate new thresholds as the task or transformation type varies.

The use of a 6-fold cross-validation scheme substantially increased the computational overhead of calculating the regression classes and transformation matrices. This overhead can be reduced by reducing the number of cross-validation sets. Table 6 shows the performance of two different cross-validation schemes. In terms of the number of regression classes generated, the 2-fold cross-validation scheme had significantly fewer regression classes than the 6-fold case. This was not surprising as only approximately 50% of the adaptation data was available in the 2-fold case when performing cross-validation compared with approximately 83% in the 6-fold case. On the H1 development data, in contrast to the evaluation data, improved performance was obtained with the 2-fold scheme. As seen in table 5, the performance on the development data peaked at two regression classes, dramatically fewer than the evaluation data. Hence, by ensuring that fewer, more robustly estimated, transforms are generated the 2-fold scheme improved performance. On

Cross-Validation Sets	Reg. Classes		Error Rate (%)	
	H1 Dev	H1 Eval	H1 Dev	H1 Eval
2	6.30	7.35	8.38	8.16
6	11.95	13.25	8.72	8.07

Table 6: Error Rate on the 1994 H1 development and evaluation data using 2-fold and 6-fold cross-validation

the other hand the 6-fold scheme performed marginally better on the evaluation data where a large number of regression classes were useful.

Stopping Criterion	Reg. Classes	Error Rate (%)
—	—	8.95
Fixed	1+sil	7.27
	2+sil	7.33
	4+sil	7.18
Threshold	5.05	7.24
X-val.	4.75	7.18

Table 7: Error Rate (%) on 1994 S5 evaluation data using fixed regression classes and threshold and cross validation stopping criteria

Table 7 shows the performance of the various stopping criteria on the S5 evaluation data. The problems of selecting the "best" set of regression classes is again illustrated with the use of two speech regression classes giving slightly worse performance than the one and four regression class cases. The performance of the thresholding and 6-fold cross-validation schemes were again similar.

Reg. Classes (+sil for S5)	Error Rate (%)		
	H1 Dev	H1 Eval	S5 Eval
—	9.57	9.20	8.95
1	8.49	8.30	7.27
2	8.23	8.18	7.03
4	8.12	8.04	6.99
8	7.92	7.94	6.93
16	7.95	7.95	—
32	7.93	—	—

Table 8: Error Rate (%) using the iterative MLLR stopping criterion on the 1994 H1 development, H1 evaluation and S5 evaluation data

Table 8 shows the performance of the iterative MLLR scheme. In all cases the transcriptions converged and in most cases the recognition performance increased as the number of levels was increased. There was a slight degradation in performance, an additional one or two word errors, on the H1 development and evaluation data. This technique does not guarantee to decrease the word error rate. Comparing the performance figures in table 8 with those of table 5 and table 7 shows that the use of iterative MLLR gave better performance figures than standard MLLR. Furthermore it may be used with no threshold information in choosing the regression classes to use. However this technique is computationally expensive. At each level it is necessary to recognise all the adaptation data. This load is greatly reduced by using lattice to constrain the search space. Furthermore at each iteration the pruning becomes more effective as the transforms start to learn the transcription.

The direct comparison of the iterative MLLR scheme with the the other schemes is not completely fair as all the other schemes used the transcriptions from the SI recognition run. Whereas the iterative MLLR scheme used the transcription from the previous run.

7 Conclusions

This report has addressed two problems in the use of maximum likelihood linear regression. The first problem examined was that of generating regression class trees. Two new techniques for generating optimal, in the sense of maximising the likelihood of the clustering data, regression class trees are described. This regression class tree building criterion is matched to the criterion used to generate the MLLR transforms in contrast to previous techniques which used phonetic knowledge or acoustic domain clustering. The first scheme, *hard* clustering, assigns each base class to one of the possible regression classes. A *fuzzy* clustering scheme is also described. In this, each base class uses a linear combination of all MLLR transforms generated for a particular set of adaptation data to generate an optimal transform. Both these schemes were evaluated on very simple regression class trees where only a single binary split was considered. The techniques described resulted in better regression class trees, in the sense that they yielded higher probabilities for the adaptation data than the standard acoustically clustered regression class tree. Unfortunately there was no reduction in word error rate, even for the more complex fuzzy weightings. This is disappointing, though consistent with the findings in CART literature, where performance has been shown to be to be consistent over a wide range of reasonable splitting criteria [1].

The second problem addressed was how to select appropriate regression classes given a regression class tree and set of adaptation data. The standard techniques for selecting regression classes is for them to be either pre-defined regression classes, *fixed*, or a minimum number of frames assigned per regression class, *threshold*. These standard schemes were found to give highly variable performance over the test sets examined. A cross-validation scheme and an iterative MLLR scheme were compared with these standard schemes. The use of cross-validation gave similar performance to the thresholding scheme without the need to empirically determine the thresholds, though the recognition performance gains were still inconsistent over the test sets. The lowest error rates on all the test sets were obtained with the iterative MLLR scheme. This again requires no thresholds to be used. However it does require multiple recognition runs on the adaptation data and is only applicable to unsupervised adaptation tasks.

A Maximum Likelihood Estimation of "Hard" Transformations

This appendix describes how ML estimates of the transformations are obtained when "hard", standard, clustering schemes are used in the regression tree.

In order to solve this maximisation problem an *Expectation-Maximisation* (EM) technique [2] is used. The standard auxiliary function $\mathcal{Q}(\mathcal{M}, \hat{\mathcal{M}})$ is adopted,

$$\mathcal{Q}(\mathcal{M}, \hat{\mathcal{M}}) = \tag{17}$$
$$K_1 - \frac{1}{2}\mathcal{L}(\mathbf{O}_T|\mathcal{M}) \sum_{m=1}^{M} \sum_{\tau=1}^{T} L_m(\tau) \left[K_m + \log(|\boldsymbol{\Sigma}_m|) + (\mathbf{o}(\tau) - \hat{\mu}_m)^T \boldsymbol{\Sigma}_m^{-1}(\mathbf{o}(\tau) - \hat{\mu}_m)\right]$$

where K_1 is a constant dependent only on the transition probabilities, K_m is the normalisation constant associated with Gaussian m, $\mathbf{O}_T = \{\mathbf{o}(1), \ldots, \mathbf{o}(T)\}$ is the adaptation data and

$$L_m(\tau) = p(q_m(\tau)|\mathcal{M}, \mathbf{O}_T) \tag{18}$$

where $q_m(\tau)$ indicates Gaussian m at time τ. Increasing the value of this auxiliary function is guaranteed to increase the likelihood of the adaptation data.

Given that a particular transformation $\hat{\mathbf{W}}^{(r)}$ is used to transform all components associated with regression class r, then $\hat{\mathbf{W}}^{(r)}$ may be found by solving

$$\sum_{\tau=1}^{T} \sum_{c=1}^{C_r} \sum_{m=1}^{M_c} L_m(\tau)\boldsymbol{\Sigma}_m^{-1}\mathbf{o}(\tau)\xi_m^T = \sum_{\tau=1}^{T} \sum_{c=1}^{C_r} \sum_{m=1}^{M_c} L_m(\tau)\boldsymbol{\Sigma}_m^{-1}\hat{\mathbf{W}}^{(r)}\xi_m\xi_m^T \tag{19}$$

For the full covariance matrix case the solution is computationally very expensive [4], however, for the diagonal covariance matrix case a closed-form solution is computationally feasible [8].

The left-hand side of equation 19 is independent of the transformation matrix and will be referred to as $\mathbf{Z}$, where

$$\mathbf{Z} = \sum_{c=1}^{C_r} \sum_{m=1}^{M_c} \sum_{\tau=1}^{T} L_m(\tau)\boldsymbol{\Sigma}_m^{-1}\mathbf{o}(\tau)\xi_m^T \tag{20}$$

A new variable $\mathbf{G}^{(i)}$ is defined with elements

$$g_{jq}^{(i)} = \sum_{c=1}^{C_r} \sum_{m=1}^{M_c} v_{ii}^{(m)} d_{jq}^{(m)} \tag{21}$$

where

$$\mathbf{V}^{(m)} = \sum_{\tau=1}^{T} L_m(\tau)\boldsymbol{\Sigma}_m^{-1} \tag{22}$$

and

$$\mathbf{D}^{(m)} = \xi_m \xi_m^T \tag{23}$$

$\hat{\mathbf{W}}^{(r)}$ is calculated using

$$\hat{\mathbf{w}}_i^{(r)T} = \mathbf{G}^{(i)-1}\mathbf{z}_i^T \tag{24}$$

where $\hat{\mathbf{w}}_i^{(r)}$ is the i^{th} vector of $\hat{\mathbf{W}}^{(r)}$ and $\mathbf{z}_i$ is the i^{th} vector of $\mathbf{Z}$.

B Maximum Likelihood Estimation of "Fuzzy" Transformations

This appendix describes how ML estimates of the transformations are obtained when "fuzzy" clustering schemes are used in the regression tree.

Again, the aim here is to maximise the transformations, given a set of adaptation data and weights. In order to re-estimate transform $\mathbf{W}^{(j)}$, rewriting equation 5[8]

$$\mathcal{Q}(\mathcal{M},\hat{\mathcal{M}}) = K_1 - \frac{1}{2}\mathcal{L}(\mathbf{O}_T|\mathcal{M})\sum_{r=1}^{R}\sum_{c=1}^{C_r}\sum_{m=1}^{M_c}\sum_{\tau=1}^{T} L_m(\tau)\left[K_m + \log(|\mathbf{\Sigma}_m|) + \mathbf{y}(\tau)^T\mathbf{\Sigma}_m^{-1}\mathbf{y}(\tau)\right] \quad (25)$$

where

$$\mathbf{y}(\tau) = \left(\mathbf{o}(\tau) - \sum_{p\neq j}\hat{\gamma}_p^{(c)}\hat{\mu}_m^{(p)}\right) - \mathbf{W}^{(j)}\gamma_j^{(c)}\xi_m \quad (26)$$

By rewriting in this form the transform is obtained in exactly the same way as the standard case. To calculate $\mathbf{W}^{(r)}$ the following accumulates are used

$$\mathbf{Z}^{(j)} = \sum_{r=1}^{R}\sum_{c=1}^{C_r}\sum_{m=1}^{M_c}\sum_{\tau=1}^{T} L_m(\tau)\mathbf{\Sigma}^{(m)-1}\mathbf{o}^{(\bar{j})}(\tau)\xi_m^{(j)T} \quad (27)$$

where

$$\mathbf{o}^{(\bar{j})}(\tau) = \mathbf{o}(\tau) - \sum_{p\neq j}\hat{\gamma}_p^{(c)}\hat{\mu}_m^{(p)} \quad (28)$$

and

$$\xi_m^{(j)} = \gamma_j^{(c)}\xi_m \quad (29)$$

A new variable $\mathbf{G}^{(ji)}$ is defined with elements

$$g_{sq}^{(ji)} = \sum_{r=1}^{R}\sum_{c=1}^{C_p}\sum_{m=1}^{M_c}\gamma_j^{(c)2}v_{ii}^{(m)}d_{pq}^{(m)} \quad (30)$$

where

$$\mathbf{V}^{(m)} = \sum_{\tau=1}^{T} L_m(\tau)\mathbf{\Sigma}_m^{-1} \quad (31)$$

and

$$\mathbf{D}^{(m)} = \xi_m\xi_m^T \quad (32)$$

$\hat{\mathbf{W}}^{(j)}$ is calculated using

$$\hat{\mathbf{w}}_i^{(j)T} = \mathbf{G}^{(ji)-1}\mathbf{z}_i^{(j)T} \quad (33)$$

where $\hat{\mathbf{w}}_i^{(j)}$ is the i^{th} vector of $\hat{\mathbf{W}}^{(j)}$ and $\mathbf{z}_i$ is the i^{th} vector of $\mathbf{Z}$. This shows that, given a set of weights, the transforms may be individually optimised. This has optimised an individual transformation given all other transforms are constant. This is not true as all transforms will be dependent, through the weights, on each other. If optimised in this fashion it is necessary to run a

[8] The summation over all speakers is left out for clarity.

few iterations updating each transformation individually. This is not as computationally expensive as it first appears as $\mathbf{G}^{(ji)}$ is independent of the transforms, so need only be inverted once.

Alternatively all the transforms may be estimated in a single step. Again rewriting the expression yields the following equality when considering $\mathbf{W}^{(j)}$

$$\sum_{r=1}^{R}\sum_{c=1}^{C_r}\sum_{m=1}^{M_c}\sum_{\tau=1}^{T} L_m(\tau)\mathbf{\Sigma}_m^{-1}\mathbf{o}(\tau)\xi_m^{(j)T} = \sum_{\tau=1}^{T}\sum_{r=1}^{R}\sum_{c=1}^{C_r}\sum_{m=1}^{M_c} L_m(\tau)\mathbf{\Sigma}_m^{-1}\left[\sum_{p=1}^{R}\gamma_p^{(c)}\hat{\mathbf{W}}^{(p)}\right]\xi_m\xi_m^{(j)T} \tag{34}$$

Only considering the diagonal covariance case, the following equation must be satisfied

$$\mathbf{z}_i^{(j)T} = \sum_{r=1}^{R}\mathbf{G}^{(rji)}\hat{\mathbf{w}}_i^{(r)T} \tag{35}$$

where

$$g_{pq}^{(kji)} = \sum_{r=1}^{R}\sum_{c=1}^{C_r}\sum_{m=1}^{M_c}\gamma_k^{(c)}\gamma_j^{(c)}v_{ii}^{(m)}d_{pq}^{(m)} \tag{36}$$

This may be written as

$$\begin{bmatrix}\mathbf{z}_i^{(1)T}\\ \vdots\\ \mathbf{z}_i^{(R)T}\end{bmatrix} = \begin{bmatrix}\mathbf{G}^{(11i)} & \dots & \mathbf{G}^{(1Ri)}\\ \vdots & \ddots & \vdots\\ \mathbf{G}^{(R1i)} & \dots & \mathbf{G}^{(RRi)}\end{bmatrix}\begin{bmatrix}\hat{\mathbf{w}}_i^{(1)T}\\ \vdots\\ \hat{\mathbf{w}}_i^{(R)T}\end{bmatrix} \tag{37}$$

This now requires the inversion of a $(R \times (n+1))^2$ matrix for each of the n dimensions. This may be compared with the individual optimisation case which involves inverting R sets of $(n+1)^2$ matrices for each of the dimensions. As inversion is an $\mathcal{O}(n^3)$ operation the increase in compute time is considerable, $\mathcal{O}(R^2)$.

C Maximum Likelihood Estimation of "Fuzzy" Weights

This appendix describes how ML estimates of the weights are obtained when "fuzzy" clustering schemes are used in the regression tree.

The aim is obtain the best set of weights, in a ML sense, for the given set of adaptation data and regression transforms. It is assumed that a transformation matrix has been obtained for each base class, as determined by the regression classes. All that remains is to find the weights associated with, say a particular base class (though this may be a regression class). From equation 9

$$\frac{\partial\mathcal{Q}(\mathcal{M},\hat{\mathcal{M}})}{\partial\hat{\gamma}_i^{(j)}} = -\frac{\partial}{\partial\hat{\gamma}_i^{(j)}}\left(\frac{1}{2}\mathcal{L}(\mathbf{O}_T|\mathcal{M})\sum_{r=1}^{R}\sum_{c=1}^{C_r}\sum_{m=1}^{M_c}\sum_{\tau=1}^{T} L_m(\tau)(\mathbf{o}(\tau)-\hat{\mu}_m)^T\mathbf{\Sigma}_m^{-1}(\mathbf{o}(\tau)-\hat{\mu}_m)\right) \tag{38}$$

Letting

$$\hat{\mu}_m^{(r)} = \hat{\mathbf{W}}^{(r)}\xi_m \tag{39}$$

and rewriting

$$\mathbf{o}(\tau)-\hat{\mu}_m = \left(\mathbf{o}(\tau)-\sum_{p\neq r}\hat{\gamma}_p^{(c)}\hat{\mu}_m^{(p)}\right) - \hat{\gamma}_r^{(c)}\hat{\mu}_m^{(r)} \tag{40}$$

It is simple to show that

$$\frac{\partial\mathcal{Q}(\mathcal{M},\hat{\mathcal{M}})}{\partial\hat{\gamma}_i^{(j)}} = \tag{41}$$

$$-\mathcal{L}(\mathbf{O}_T|\mathcal{M})\sum_{r=1}^{R}\sum_{c=1}^{C_r}\sum_{m=1}^{M_c}\sum_{\tau=1}^{T} L_m(\tau)\left(\hat{\gamma}_i^{(j)}\hat{\mu}_m^{(i)T}\mathbf{\Sigma}_m^{-1}\hat{\mu}_m^{(i)} - \hat{\mu}_m^{(i)T}\mathbf{\Sigma}_m^{-1}\left(\mathbf{o}(\tau)-\sum_{p\neq i}\hat{\gamma}_p^{(c)}\hat{\mu}_m^{(p)}\right)\right)$$

and rewriting slightly

$$\frac{\partial \mathcal{Q}(\mathcal{M},\hat{\mathcal{M}})}{\partial \hat{\gamma}_i^{(j)}} = -\mathcal{L}(\mathbf{O}_T|\mathcal{M}) \sum_{r=1}^{R}\sum_{c=1}^{C_r}\sum_{m=1}^{M_c}\sum_{\tau=1}^{T} L_m(\tau)\left(\hat{\mu}_m^{(i)T}\mathbf{\Sigma}_m^{-1}\sum_{p=1}^{R}\hat{\gamma}_p^{(c)}\hat{\mu}_m^{(p)} - \hat{\mu}_m^{(i)T}\mathbf{\Sigma}_m^{-1}\mathbf{o}(\tau)\right) \tag{42}$$

Defining a new $n \times R$ matrix $\hat{\mathbf{M}}_m$ where

$$\hat{\mathbf{M}}_m = \left[\begin{array}{ccc} \hat{\mu}_m^{(1)} & \ldots & \hat{\mu}_m^{(R)} \end{array}\right] \tag{43}$$

it is possible to write

$$\hat{\mu}_m = \hat{\mathbf{M}}_m\hat{\gamma}^{(c)} \tag{44}$$

where

$$\hat{\gamma}^{(c)} = \left[\begin{array}{ccc} \hat{\gamma}_1^{(c)} & \ldots & \hat{\gamma}_R^{(c)} \end{array}\right]^T \tag{45}$$

Equation 42 can be written as

$$\frac{\partial \mathcal{Q}(\mathcal{M},\hat{\mathcal{M}})}{\partial \hat{\gamma}_i^{(j)}} = -\mathcal{L}(\mathbf{O}_T|\mathcal{M}) \sum_{r=1}^{R}\sum_{c=1}^{C_r}\sum_{m=1}^{M_c}\sum_{\tau=1}^{T} L_m(\tau)\left(\hat{\mu}_m^{(i)T}\mathbf{\Sigma}_m^{-1}\hat{\mathbf{M}}_m\hat{\gamma}^{(j)} - \hat{\mu}_m^{(i)T}\mathbf{\Sigma}_m^{-1}\mathbf{o}(\tau)\right) \tag{46}$$

Equating equation 46 to zero[9]

$$\left[\sum_{r=1}^{R}\sum_{c=1}^{C_r}\sum_{m=1}^{M_c}\sum_{\tau=1}^{T} L_m(\tau)\hat{\mu}_m^{(i)T}\mathbf{\Sigma}_m^{-1}\hat{\mathbf{M}}_m\right]\hat{\gamma}^{(j)} = \sum_{r=1}^{R}\sum_{c=1}^{C_r}\sum_{m=1}^{M_c}\sum_{\tau=1}^{T} L_m(\tau)\hat{\mu}_m^{(i)T}\mathbf{\Sigma}_m^{-1}\mathbf{o}(\tau) \tag{47}$$

For the global maximum value this equality must hold for all elements of $\hat{\gamma}^{(m)}$. Expressing this in matrix notation

$$\mathbf{Z}^{(j)}\hat{\gamma}^{(j)} = \mathbf{v}^{(j)} \tag{48}$$

where $\mathbf{Z}^{(j)}$ is a $R \times R$ matrix and $\mathbf{v}^{(j)}$ is a $R \times 1$ vector defined by

$$\mathbf{z}_i^{(j)} = \sum_{r=1}^{R}\sum_{c=1}^{C_r}\sum_{m=1}^{M_c}\sum_{\tau=1}^{T} L_m(\tau)\hat{\mu}_m^{(i)T}\mathbf{\Sigma}_m^{-1}\hat{\mathbf{M}}_m \tag{49}$$

and

$$v_i^{(j)} = \sum_{r=1}^{R}\sum_{c=1}^{C_r}\sum_{m=1}^{M_c}\hat{\mu}_m^{(i)T}\mathbf{\Sigma}_m^{-1}\left(\sum_{\tau=1}^{T} L_m(\tau)\mathbf{o}(\tau)\right) \tag{50}$$

The ML estimate for $\hat{\gamma}^{(j)}$ is then given by

$$\hat{\gamma}^{(j)} = \mathbf{Z}^{(j)-1}\mathbf{v}^{(j)} \tag{51}$$

[9] The second derivative, $\frac{\partial^2 \mathcal{Q}(\mathcal{M},\hat{\mathcal{M}})}{\partial \hat{\gamma}_i^{(j)2}}$, is always negative indicating that the turning point is a maximum.

References

[1] L Breiman, J H Friedman, R A Olshen, and C J Stone. *Classification and Regression Trees.* Wadsworth, 1984.

[2] A P Dempster, N M Laird, and D B Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39:1–38, 1977.

[3] M J F Gales. *Model-Based Techniques for Noise Robust Speech Recognition.* PhD thesis, Cambridge University, 1996.

[4] M J F Gales and P C Woodland. Variance compensation within the MLLR framework. Technical Report CUED/F-INFENG/TR242, Cambridge University, 1996. Available via anonymous ftp from: svr-ftp.eng.cam.ac.uk.

[5] J L Gauvain and C H Lee. Maximum a-posteriori estimation for multivariate Gaussian mixture observations of Markov chains. *IEEE Transactions SAP*, 2:291–298, 1994.

[6] C H Lee, C H Lin, and B H Juang. A study of speaker adaptation of continuous density HMM parameters. In *Proceedings ICASSP*, pages 145–148, 1990.

[7] C J Leggetter. *Improved Acoustic Modelling for HMMs using Linear Transformations.* PhD thesis, Cambridge University, 1995.

[8] C J Leggetter and P C Woodland. Flexible speaker adaptation for large vocabulary speech recognition. In *Proceedings Eurospeech*, pages 1155–1158, 1995.

[9] C J Leggetter and P C Woodland. Maximum likelihood linear regression for speaker adaptation of continuous density HMMs. *Computer Speech and Language*, 9:171–186, 1995.

[10] L R Neumeyer, A Sankar, and V V Digalakis. A comparative study of speaker adaptation techniques. In *Proceedings Eurospeech*, pages 1127–1130, 1995.

[11] J J Odell, V Valtchev, P C Woodland, and S J Young. A one pass decoder design for large vocabulary recognition. In *Proceedings ARPA Workshop on Human Language Technology*, pages 405–410, 1994.

[12] A P Varga and R K Moore. Hidden Markov model decomposition of speech and noise. In *Proceedings ICASSP*, pages 845–848, 1990.

[13] P C Woodland, J J Odell, V Valtchev, and S J Young. The development of the 1994 HTK large vocabulary speech recognition system. In *Proceedings ARPA Workshop on Spoken Language Systems Technology*, pages 104–109, 1995.

[14] P C Woodland, D Pye, and M J F Gales. Iterative unsupervised adaptation using maximum likelihood linear regression. In *Proceedings ICSLP*, 1996.

[15] S J Young, J J Odell, and P C Woodland. Tree-based state tying for high accuracy acoustic modelling. In *Proceedings ARPA Workshop on Human Language Technology*, pages 307–312, 1994.