TRANSFORMATION SMOOTHING FOR SPEAKER AND ENVIRONMENTAL ADAPTATION

M.J.F.Gales

Cambridge University Engineering Department,
Cambridge, CB2 1PZ, UK
mjfg@eng.cam.ac.uk

ABSTRACT

Recently there has been much work done on how to transform HMMs, trained typically in a speaker-independent fashion on clean training data, to be more representative of data from a particular speaker or acoustic environment. These transforms are trained on a small amount of training data, so large numbers of components are required to share the same transform. Normally, each component is constrained to only use one transform. This paper examines how to optimally, in a maximum likelihood sense, assign components to transforms and allow each component, or component grouping, to make use of many transformations. The theory for obtaining both "weights" for each transform and transforms given a set of weights is given. The techniques are evaluated on both speaker and environmental adaptation tasks.

1. INTRODUCTION

There has been much work done on how to improve speech recognition performance in new acoustic environments or for new speakers. A currently popular technique is to transform the clean, speaker-independent, model parameters [5, 3, 4] to be more representative of the new environment or speaker. These transforms may be linear [4] or non-linear [1, 3]. Furthermore they may either be based on some assumptions about the acoustic environment [3], or by training a set of transforms on some adaptation data [5, 4].

When modifying the models using adaptation data, there is generally too little data to independently adapt each component so many components share the same transform. Typically each Gaussian component of an HMM-based speech recognition system will be associated with one particular transform and its parameters modified accordingly. This paper examines the problem of how to use many transforms, or "experts", to yield an improved adaptation scheme. The task is to combine these multiple experts, in this case transforms of the mean for model compensation, in a sensible fashion. A simple linear combination scheme is proposed here, where the final estimate is a linear combination of the output of each of the experts. This is referred to as *transformation smoothing*. The problem of estimating the weight for each expert is approached using maximum likelihood (ML) estimation. In addition, techniques for estimating the expert parameters given a set of weights will be described.

A simple example of combining experts has previously been proposed [1] where a particular set of components is transformed by a combination of a linear and a non-linear transformation. Each set of components uses one of these combined transforms for adaptation. This concept is extended here to allow each set of components to use an arbitrary number of linear and non-linear transforms for adaptation.

Preliminary experiments on transformation smoothing are detailed. Two smoothing approaches are examined. First, the technique is used to generate a "soft" regression tree for speaker adaptation, so that components make use of all the available transforms. The scheme is contrasted with standard "hard" regression trees. Second the use of smoothing at run-time is examined.

2. TRANSFORMATION SMOOTHING

The scheme chosen here for combining the experts is a linear one. Only adaptation of the means of the Gaussian components will be considered, thus

$$\hat{\mu}^{(m)} = \sum_{r=1}^{R} \lambda_r \hat{\mu}^{(mr)} = \hat{\mathbf{M}}^{(m)}\lambda \tag{1}$$

where $\hat{\mu}^{(mr)} = \mathcal{F}_r(\mu^{(m)})$, is the estimate of the mean generated from the r^{th} expert, λ is the $R \times 1$ weights vector and

$$\hat{\mathbf{M}}^{(m)} = \left[\begin{array}{ccc} \hat{\mu}^{(m1)} & \dots & \hat{\mu}^{(mR)} \end{array} \right] \tag{2}$$

Given this transformation scheme it will be necessary to estimate the weights vector for a set of transforms, or the set of transforms given the weight vectors. Typically there is insufficient data to estimate either experts or weights for each individual component. Therefore Gaussian components are grouped into *base classes*, all components of which are assumed to be transformed in a similar way, ie the weight vector is the same.

2.1. Estimating the Weights

The task is to find the ML estimate of the elements of the weights vector λ, given a set of experts, either linear or non-linear. The standard auxiliary function is used to optimise this, given the component alignment of each observation with a particular state. Details of the derivation are in [2]. The weights vector for base class j, $\lambda^{(j)}$ is

$$\lambda^{(j)} = \mathbf{Z}^{(j)-1}\mathbf{v}^{(j)} \tag{3}$$

where row r of the $R \times R$ matrix, $\mathbf{Z}^{(j)}$ is defined as

$$\mathbf{z}_r^{(j)} = \sum_{m=1}^{M^{(j)}} \sum_{\tau=1}^{T} \gamma_m(\tau) \hat{\mu}^{(mr)T} \mathbf{\Sigma}^{(m)-1} \hat{\mathbf{M}}^{(m)} \tag{4}$$

and element r of the $R \times 1$ vector $\mathbf{v}^{(j)}$ is defined as

$$v_r^{(j)} = \sum_{m=1}^{M^{(j)}} \hat{\mu}^{(mr)T} \mathbf{\Sigma}^{(m)-1} \left(\sum_{\tau=1}^{T} \gamma_m(\tau) \mathbf{o}(\tau) \right) \tag{5}$$

with $\gamma_m(\tau) = p(q_m(\tau)|\mathbf{o}_T)$, $q_m(\tau)$ denotes occupation of component m at time τ and $M^{(j)}$ is the number of components associated with base class j. Notice this has made no assumption about the nature of the transformation, solely that each expert generates an estimate of the new mean.

2.2. Estimating the Transforms

Given a set of weights it is possible to train both linear and non-linear transformations.

Linear Transformations : Consider the general linear mean transformation, maximum likelihood linear regression (MLLR) [4], combined with smoothing

$$\hat{\mu}^{(m)}=\sum_{r=1}^{R}\lambda_r^{(b_m)}\mathbf{W}^{(r)}\xi^{(m)}=\sum_{r=1}^{R}\lambda_r^{(b_m)}(\mathbf{A}^{(r)}\mu^{(m)}+\mathbf{b}^{(r)}) \quad (6)$$

where $\xi^{(m)}$ is the extended mean vector. In [2], it is shown that all the linear transforms may be simultaneously optimised by solving the following expression

$$\begin{bmatrix} \mathbf{z}_i^{(1)T} \\ \vdots \\ \mathbf{z}_i^{(R)T} \end{bmatrix} = \begin{bmatrix} \mathbf{G}^{(11i)} & \cdots & \mathbf{G}^{(1Ri)} \\ \vdots & \ddots & \vdots \\ \mathbf{G}^{(R1i)} & \cdots & \mathbf{G}^{(RRi)} \end{bmatrix} \begin{bmatrix} \mathbf{w}_i^{(1)T} \\ \vdots \\ \mathbf{w}_i^{(R)T} \end{bmatrix} \quad (7)$$

where

$$\mathbf{Z}^{(r)} = \sum_{m=1}^{M}\sum_{\tau=1}^{T}\gamma_m(\tau)\mathbf{\Sigma}^{(m)-1}\mathbf{o}(\tau)\lambda_r^{(b_m)}\xi^{(m)T} \quad (8)$$

$$g_{pq}^{(kli)} = \sum_{m=1}^{M}\lambda_k^{(b_m)}\lambda_l^{(b_m)}v_{ii}^{(m)}d_{pq}^{(m)} \quad (9)$$

$$\mathbf{V}^{(m)} = \sum_{\tau=1}^{T}\gamma_m(\tau)\mathbf{\Sigma}^{(m)-1} \quad (10)$$

$\mathbf{D}^{(m)} = \xi^{(m)}\xi^{(m)T}$ and component m belongs to base class b_m. It is simple to see that when "hard" weights are used, ie only one element of the weight vector is non-zero and equal to one, then equation 7 simplifies to the standard MLLR estimation equation.

Non-Linear Transformations : A simple closed form expression is not normally possible when training non-linear transforms. Instead an iterative procedure must be used whereby each transform is individually optimised. Typically, the training of the non-linear transform is based on the generalised EM algorithm [1]. However, now instead of training on $\mathbf{o}(\tau)$, the transform $\mathcal{F}_r(\mu)$ is trained on $\mathbf{o}^{(m\overline{r})}(\tau)$ for those frames assigned to component m, where

$$\mathbf{o}^{(m\overline{r})}(\tau) = \mathbf{o}(\tau) - \sum_{p\neq r}\lambda_p^{(b_m)}\hat{\mu}^{(mp)} \quad (11)$$

The same routine may be used to train the linear transformations [2]. Thus instead of requiring the inverse of an $(R\times(n+1))$ by $(R\times(n+1))$ matrix, it is an iterative solution where each iteration involves R inversions of $(n+1)$ by $(n+1)$ matrices.

If linear and non-linear transformations are to be combined then an iterative procedure is required. The linear transforms may be optimised given the current non-linear transforms and so on.

3. IMPLEMENTATION OPTIONS

3.1. Regression Tree Training

One particular implementation of transformation smoothing is to generate an optimal, in the sense of maximising the likelihood, regression class tree for use with MLLR [4]. As the amount of adaptation data is unknown when deciding on the base classes a regression class tree may be used to group these base classes into hierarchical regression classes. The regression tree and base classes are usually determined using expert knowledge or assuming that components that are "close" in acoustic space belong to the same base class. Figure 1 shows a simple regression class

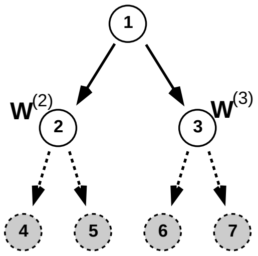

Figure 1: A binary regression class tree

tree. This is a binary tree, with two regression classes, r_2 and r_3 and associated transforms $\mathbf{W}^{(2)}$ and $\mathbf{W}^{(3)}$. Each of the four base classes (c_4, c_5, c_6, c_7) is assigned in a "hard" fashion to one of regression classes.

Transformation smoothing allows two improvements over the standard regression class tree. First, there is no "hard" allocation of base class to one particular regression class, each base class can make use of all available transforms. Second, the allocation of base classes to regression classes is based on optimising the likelihood of some regression-tree building data, rather than the standard "closeness" in acoustic space measure. Each base class will now have a weight vector associated with it.

The training of a simple two regression class tree is as follows. The tree is initialised using standard acoustic clustering schemes. For each speaker in the training data transforms $\mathbf{W}^{(2)}$ and $\mathbf{W}^{(3)}$ are trained. Using these transforms the weight vector for each of the base classes is learnt over all speakers. Given these new weight vectors new transforms are learnt and the whole process may be repeated. At each stage the likelihood of the training data is guaranteed not to decrease. Thus the transforms for each speaker are determined using equation 7, with the summation over time running to $T^{(s)}$ for speaker s and the "weights" of the tree are determined by

$$\hat{\lambda}^{(j)} = \arg\max_{\lambda\in\mathbf{\Lambda}}\left\{\sum_{s=1}^{S}\mathcal{Q}_j^{(s)}(\mathcal{M},\hat{\mathcal{M}}|\lambda)\right\} \quad (12)$$

where[1]

$$\mathcal{Q}_j^{(s)}(\mathcal{M},\hat{\mathcal{M}}|\lambda) = (-) \quad (13)$$
$$\sum_{m=1}^{M^{(j)}}\sum_{\tau=1}^{T^{(s)}}\gamma_m(\tau)\left(\mathbf{o}(\tau)-\hat{\mu}^{(mr_s)}\right)^T\mathbf{\Sigma}^{(m)-1}\left(\mathbf{o}(\tau)-\hat{\mu}^{(mr_s)}\right)$$

and $\hat{\mu}^{(mr_s)} = \mathcal{F}_{r_s}(\mu^{(m)})$, r_s indicates transformation r of speaker s. Equations 4 and 5 become

$$\mathbf{z}_r^{(j)} = \sum_{s=1}^{S}\sum_{m=1}^{M^{(j)}}\sum_{\tau=1}^{T^{(s)}}\gamma_m(\tau)\hat{\mu}^{(mr_s)T}\mathbf{\Sigma}^{(m)-1}\hat{\mathbf{M}}^{(m_s)} \quad (14)$$

[1] Ignoring all terms that are independent of the regression tree.

and

$$v_r^{(j)} = \sum_{s=1}^{S} \sum_{m=1}^{M^{(j)}} \hat{\mu}^{(mr_s)T} \mathbf{\Sigma}^{(m)-1} \left(\sum_{\tau=1}^{T^{(s)}} \gamma_m(\tau) \mathbf{o}(\tau) \right) \quad (15)$$

The obvious problem with the use of soft weights is that as the amount of data, hence use of regression class tree, is not known a-priori it is necessary to learn weights for every single allowable combination of transforms that could be generated from the tree. One way around this problem is to restrict the number of transforms that can be used at each level. There are a variety of implementations that could be used. For this paper only a simple one level binary tree is considered, so the problem does not arise.

It is not necessary to generate "soft" weights in order to obtain regression trees based on maximise the likelihood. "Hard" weights may be learnt in a similar fashion [2]. Here the transform that yields the highest average auxiliary function value is chosen. Thus

$$\hat{r}^{(j)} = \arg\max_{r \in R} \left\{ \sum_{s=1}^{S} \mathcal{Q}_j^{(s)}(\mathcal{M}, \hat{\mathcal{M}}|r) \right\} \quad (16)$$

where $\mathcal{Q}_j^{(s)}(\mathcal{M}, \hat{\mathcal{M}}|r)$ has a similar form as equation 13. This is again guaranteed to increase the likelihood of the adaptation at each iteration, though is very susceptible to ending up at local maxima. However, "hard" weights have the advantage that they allow a standard tree to be built in a top down fashion.

3.2. Run-Time Smoothing

The previous section has described how a "soft" regression class tree may be generated for all speakers. An alternative approach is to calculate a set of weights for a speaker given a particular set of transforms calculated on the adaptation data for that speaker. This has the advantage that a specific transformation smoothing 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 tree. It is possible to go further down this weights tree, since less data is required to estimate the weights than to estimate the transforms themselves. However the need to robustly estimate the weights is a disadvantage of the run-time smoothing 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 smoothing 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 iterating was found to give little difference in performance and is not used in this work.

3.3. Environmental Smoothing

A common problem in speech recognition is dealing with varying acoustic environments. One approach to such situations is to generate a set of "transforms" for a set of acoustic conditions. At test time the most appropriate acoustic condition is selected. Normally it is not possible to build models of all acoustic environments so the resolution of environments may not be as fine as would be desirable. Instead of making a hard decision, selecting one environment, or performing some ad-hoc combination of environments, it is possible to smooth the environment transform in a ML fashion[2]. The optimisation is very similar to that of transformation smoothing, however now there is only one weight vector of dimensionality $E \times 1$ for the E acoustic environments. The environment weights, $\lambda^{(e)}$, are determined by

$$\lambda^{(e)} = \mathbf{Z}^{-1}\mathbf{v} \quad (17)$$

where row e of the $E \times E$ matrix, $\mathbf{Z}$ is defined as

$$\mathbf{z}_e = \sum_{m=1}^{M} \sum_{\tau=1}^{T} \gamma_m(\tau) \hat{\mu}^{(mr_e)T} \mathbf{\Sigma}^{(m)-1} \hat{\mathbf{M}}^{(me)} \quad (18)$$

$\hat{\mathbf{M}}^{(me)} = \left[\hat{\mu}^{(mr_1)} \dots \hat{\mu}^{(mr_E)}\right]$, element e of the $E \times 1$ vector v_i is defined as

$$v_e = \sum_{m=1}^{M} \hat{\mu}^{(mr_e)T} \mathbf{\Sigma}^{(m)-1} \left(\sum_{\tau=1}^{T} \gamma_m(\tau) \mathbf{o}(\tau) \right) \quad (19)$$

and $\hat{\mu}^{(mr_e)} = \mathcal{F}_{r_e}(\mu^{(m)})$, r_e indicates transformation r of acoustic environment e. The cost of calculating the weights is simply the inversion of an $e \times E$ matrix.

4. EXPERIMENTS

The HTK standard large vocabulary speech recognition system was used to compare the performance of the various Gaussian selection schemes. This was configured as a gender-independent cross-word-triphone mixture-Gaussian tied-state HMM system identical to the "HMM-1" model set used in the HTK 1994 ARPA evaluation system [7].

The data used for evaluation were the ARPA 1994 H1 development and evaluation test sets. This is an unlimited vocabulary task recorded in a clean environment. A 65k word list and dictionary were used with a trigram language model.

For the experiments here block-diagonal mean transformations were used, trained in a static unsupervised adaptation mode. A simple single-level tree, ie two regression classes, was trained with the Gaussian components split into 750 base classes determined by acoustic clustering.

4.1. Regression Tree Generation

A subset of the acoustic training data, 25 sentences from each of the 284 speakers, were used to train the regression tree ie the weight vectors for each base class. In addition to training a "soft" regression tree, a "hard" assignment of base classes to regression class was also performed. This is where [0,1] weights are learnt from the training data for each base class using equation 16. These trees were then used to generate the transforms for the test data.

Figure 2 shows the auxiliary function value for these two cases as a function of iteration during training of the trees. As expected in both cases the value always increased with iteration number. The "hard" weights were trained until convergence at which point 17% of the components had changed base-class and the average auxiliary function value per frame had increased from -61.54 to -61.46. The "soft" weights showed far greater increases in auxiliary function value, ending up at -61.39. This illustrates that extensive use is being made of the "soft" weights.

[2]For this work it is assumed that the environment transforms only relate to the means. Variance adaptation for each environment is not performed.

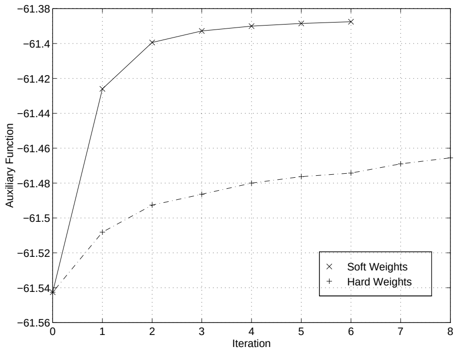

Figure 2: Auxiliary function value against iteration.

Cluster	Avg. Like.		Error Rate (%)	
	H1 Dev	H1 Eval	H1 Dev	H1 Eval
—	-68.732	-68.789	9.57	9.20
Global	-66.753	-66.762	8.49	8.30
Acoustic	-66.527	-66.536	8.39	8.21
Hard	-66.443	-66.451	8.20	8.22
Soft	-66.351	-66.348	8.29	8.23

Table 1: Error Rate (%) on the ARPA H1 data

Figure 2 illustrates that the standard acoustic clustering may not yield an optimal partitioning of the data. However it does not necessarily mean that the recognition performance improves. Table 1 shows the performance of various schemes: *Global* used a single transformation; *Acoustic* indicates the two-class baseline regression tree built clustering in acoustic space; *Hard* is the tree with [1,0] assignments of base classes; and *Soft* uses the weight vectors. From these results there is little difference in recognition performance between the various schemes. However, the likelihood of the test data and auxiliary function values were higher for the "soft" trees than the other schemes. This indicates that the use of transformation smoothing does, even in this simple two transform case, improve the ability of the regression tree to model the adaptation data.

Preliminary experiments on the ARPA 1994 S5 task, a 5K unknown microphone task, showed slight gains using the soft regression, 7.09%, compared to the standard acoustic regression tree 7.34% and the hard clustering 7.27% (see [6] for details of models used).

4.2. Run-Time Smoothing

An alternative to using smoothing when generating the regression tree is to use it during recognition. Here a set of transforms for a speaker are generated using the current regression class tree. These transforms are then smoothed. Table 2 shows the performance of such a scheme. The

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

Table 2: Error Rate (%) on 1994 H1 development and evaluation data using run-time fuzzy clustering

standard acoustic regression tree was used to generate the initial transforms. The thresholds for the smoothing weights estimates were empirically set on a similar task. Little gain in performance was again observed.

5. CONCLUSIONS

This paper has described how transforms, or experts, either linear or non-linear, may be combined in a maximum likelihood fashion. A simple linear combination is used and it is shown how the weights of the combination may be trained, given a set of transforms. Furthermore, if the weights are given the transforms may be trained. A variety of ways of using this transformation smoothing are described. It may be used to generate optimal regression trees. Instead of using ad hoc methods to generate a regression tree, weights are assigned to each base-class so that the training data likelihood is maximised. Alternatively, the weights may be optimised at run-time for the particular speaker, or it may be used to smooth acoustic environments. Preliminary experiments using this smoothing scheme to generate a simple "soft" regression class tree for speaker adaptation are described. The use of this soft tree improved the adaptation scheme in terms of the auxiliary function on the independent test data. In terms of recognition performance, though, there was little difference between this and the standard acoustic tree. The same was observed for the run-time smoothing.

Future work will involve more complex "soft" trees and environmental adaptation. In addition, the combination of both linear and non-linear transformations will be examined.

Acknowledgements

Mark Gales is funded as a Research Fellow at Emmanuel College, Cambridge.

REFERENCES

1. V Abrash, A Sankar, H Franco, and M Cohen. Acoustic adaptation using non-linear transformations of HMM parameters. In *Proceedings ICASSP*, pages 729–732, 1996.
2. M J F Gales. The generation and use of regression class trees for MLLR adaptation. Technical Report CUED/F-INFENG/TR263, Cambridge University, 1996. Available via anonymous ftp from: svr-ftp.eng.cam.ac.uk.
3. M J F Gales and S J Young. Robust speech recognition using parallel model combination. *IEEE Transactions Speech and Audio Processing*, 4:352–359, 1996.
4. 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.
5. A Sankar and C H Lee. A maximum likelihood approach to stochastic matching for robust speech recognition. *IEEE Transactions Speech and Audio Processing*, 4:190–202, 1996.
6. P C Woodland, M J F Gales, and D Pye. Improving environmental robustness in large vocabulary speech recognition. In *Proceedings ICASSP*, pages 65–68, 1996.
7. 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.