UNIVERSAL BACKGROUND MODEL BASED SPEECH RECOGNITION[1]

Daniel Povey and Stephen M. Chu
IBM T. J. Watson Research Center
Yorktown Heights, New York, USA
{dpovey, schu}@us.ibm.com

Balakrishnan Varadarajan[2]
Johns Hopkins University
Baltimore, Maryland, USA
bvarada2@jhu.edu

ABSTRACT

The *universal background model* (UBM) is an effective framework widely used in speaker recognition. But so far it has received little attention from the speech recognition field. In this work, we make a first attempt to apply the UBM to acoustic modeling in ASR. We propose a tree-based parameter estimation technique for UBMs, and describe a set of smoothing and pruning methods to facilitate learning. The proposed UBM approach is benchmarked on a state-of-the-art large-vocabulary continuous speech recognition platform on a broadcast transcription task. Preliminary experiments reported in this paper already show very exciting results.

Index Terms – UBM, universal background model, speech recognition, acoustic modeling.

1. INTRODUCTION

The *universal background model* (UBM) [1] is an effective framework that has found great success in speaker recognition. Conceptually, it is a large mixture of Gaussians that covers all speech, and in the context of speaker recognition, it is adapted to each speaker using a *maximum a posteriori* (MAP) scheme.

The UBM so far has received little attention from the automatic speech recognition (ASR) field. In this paper, we make a first attempt to apply UBM to acoustic modeling in ASR, and demonstrate substantial improvements at the *maximum likelihood* (ML) level. The basic idea is to adapt the UBM to each context-dependent phone rather than to each speaker. The context-dependent phones are not an unstructured collection of phones but are related via a tree structure, hence we devise a set of smoothing methods that can utilize this structure. Training is done through multiple iterations of EM [2] rather than just one as in [1]. Furthermore, in our best-performing system, a separate semi-tied covariance (STC) transform [3] is applied for each Gaussian in the UBM. One challenge in UBM-based speech recognition is the very large size of the resulting models. An entropy based pruning method similar to [5] is used to address the problem. The results in this paper should be considered preliminary, as we have not had time to explore the many design choices involved.

The remainder of this paper is organized as follows. In Section 2, we review the baseline UBM learning algorithm. The proposed tree-based parameter estimation technique is described in Section 3. Section 4 presents the experimental results, followed by conclusions in Section 5.

2. UNIVERSAL BACKGROUND MODELS

The UBM is a Gaussian Mixture Model (GMM) whose parameters consist of K weights w_k, means $\boldsymbol{\mu}_k$ and (diagonal) variances $\boldsymbol{\Sigma}_k$, which in a speaker identification context are MAP adapted to each speaker's data to create a GMM for that speaker. In the speech recognition context, let us consider that the speech is already split up into many speech classes $j = 1 \cdots J$ based on the tree-clustered context dependent phones, and our reference transcriptions have been Viterbi-aligned given some previously existing models, so that we have (zero-one) phone posteriors, $\gamma_j(t)$, so we can treat the set of frames for which $\gamma_j(t) = 1$ for some j as we would the data from a particular speaker.

It is helpful to consider a single iteration of standard EM update starting from the UBM. Define the Gaussian-specific posterior,

$$\gamma_{jk}(t) = \gamma_j(t) \frac{w_k g(\mathbf{x}(t) \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{k=1}^{K} w_k g(\mathbf{x}(t) \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}. \tag{1}$$

Define the count statistics $\gamma_{jk}(t)$ and the first and second-order statistics $\mathbf{x}_{jk}$ and $\mathbf{S}_{jk}$ as,

$$\gamma_{jk} = \sum_{t=1}^{T} \gamma_{jk}(t), x_{jk_i} = \sum_{t=1}^{T} \gamma_{jk}(t) x(t)_i, S_{jk_{ii}} = \sum_{t=1}^{T} \gamma_{jk}(t) x(t)_i^2, \tag{2}$$

where k is the Gaussian index in the UBM. The standard EM update for the speech-class specific mean μ_{jk} and Σ_{jk} would be,

$$w_{jk} := \frac{\gamma_{jk}}{\sum_{j=1}^{J} \gamma_{jk}}, \mu_{jk_i} := \frac{x_{jk_i}}{\gamma_{jk}}, \Sigma_{jk_{ii}} := \frac{S_{jk_{ii}}}{\gamma_{jk}} n - \mu_{jk_i}^2 \tag{3}$$

The MAP adaptation scheme for the means and variances corresponds to adding a small number τ of statistics with mean and variance the same as the original Gaussian, to the statistics obtained from the data:

$$\mu_{jk_i} := \frac{x_{jk_i} + \tau \mu_{k_i}}{\gamma_{jk} + \tau}, \Sigma_{jk_{ii}} := \frac{S_{jk_i} + \tau(\mu_{k\,i}^2 + \Sigma_{k\,ii})}{\gamma_{jk} + \tau} - \mu_{jk_i}^2 \tag{4}$$

Various schemes have been used for the MAP adaptation of the weights for speaker identification, which we will not describe here.

If multiple iterations of EM are used as in [2], the Gaussian posteriors are derived from the previous iteration's models. So, on the second iteration, we define,

$$\gamma_{jk}(t) = \gamma_j(t) \frac{w_k g(\mathbf{x}(t) \mid \boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk})}{\sum_{k=1}^{K} w_k g(\mathbf{x}(t) \mid \boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk})} \tag{5}$$

[1] This work was funded in part by DARPA contract HR0011-06-2-0001.

[2] The author was at IBM T. J. Watson Research Center during this work.

3. TREE-BASED PARAMETER ESTIMATION

3.1. Global Tree

The form of MAP adaptation applied to our models is more complicated than the scheme described in the previous section.

First we obtain a single global tree to describe all of our phonetic states. The standard tree-clustering of the phones as used in our baseline system is a simple binary tree within each of the three states of each context-independent phone, where each non-leaf node in the tree represents a question about the acoustic context and each leaf node is a clustered HMM state. There is no second pass of clustering the resulting nodes, which would cause problems with the approach described here. We merge all of these individual trees by forming a node corresponding to each phone, with three children corresponding to each of the three states, and then forming a single parent node for all phones.

For parameter estimation, we need smoothing methods that utilize this tree structure. We will present our smoothing equations and describe some of the design goals used in devising these smoothing equations, which are loosely based on the Kneser-Ney equations as used for smoothing language models [4].

3.2. Design Goals

Taking and giving the same amount
If any amount of data is *taken away* from a particular state for use in smoothing, the same amount should be *given back* to it. This principle has various good effects. For instance, in mean smoothing, no data point *matters more* than any other in terms of its effect on the (weighted) mean of the entire model; in weight smoothing, it implies that the overall counts of a particular UBM Gaussian *k* tend to be preserved.

Limiting behavior
As the number of counts for any particular UBM Gaussian *k* in any particular state *j* becomes large, the weight and mean and variance value should approach the standard E-M value, regardless of the counts of other Gaussians within that state.

Behavior within the tree
In the mean and variance updates, if a particular leaf node in the tree has observations for a particular Gaussian *j* but belongs to a branch of the tree that is otherwise devoid of counts for *j*, the smoothing scheme should in essence smooth up to the closest parts of the tree where counts are available. It should not have the bad property that an isolated count within some section of the tree can manage to smooth *to itself* and in effect not get any smoothing. This constraint is why in the following equations we will sometimes use hard rather than soft count cutoffs.

3.3. Mean and Variance Smoothing

Our smoothing method for Gaussian parameters has an adjustable parameter τ_g (*g* refers to Gaussians; 20 is our default value), similar to the τ used in standard MAP.

The mean and variance smoothing is presented here as an operation on the counts γ_{jk} and the first and second order statistics $\mathbf{x}_{jk}$ and $\mathbf{S}_{jk}$. Let us say that we have numbered all of the nodes in our global clustering tree with the leaf nodes $1 \ldots J$ first, followed by the non-leaf nodes up to a total $N > J$, and that a node *j*'s parent $parent(j)$ is always numbered higher than *j*.

The smoothing is an operation done separately for each Gaussian *k* in the UBM. For a particular *k*, we first initialize the statistics γ_{jk}, $\mathbf{x}_{jk}$ and $\mathbf{S}_{jk}$ such that the leaf nodes have the statistics accumulated from the data, and the non-leaf nodes initially have zero statistics. Let $move(f, i \to j)$ denote the operation of moving a count *f* of statistics from node *i* to node *j* (defined if $\gamma_{ik} \geq f$), which is carried out by:

$$\begin{aligned} \mathbf{x}_{jk} &:= \mathbf{x}_{jk} + (f/\gamma_{ik})\mathbf{x}_{ik} \\ \mathbf{S}_{jk} &:= \mathbf{S}_{jk} + (f/\gamma_{ik})\mathbf{S}_{ik} \\ \gamma_{jk} &:= \gamma_{jk} + f \end{aligned} \tag{6}$$

$$\begin{aligned} \mathbf{x}_{ik} &:= ((\gamma_{ik} - f)/\gamma_{ik})\mathbf{x}_{ik} \\ \mathbf{S}_{ik} &:= ((\gamma_{ik} - f)/\gamma_{ik})\mathbf{S}_{ik} \\ \gamma_{ik} &:= \gamma_{ik} - f \end{aligned} \tag{7}$$

First, the data is distributed up the tree. For each node $j = 1 \ldots N-1$, we first estimate the amount to discount, $d(j)$. At leaf nodes ($j \leq J$), we use a *soft* rule: $d(j) := \gamma_{jk}\tau/(\gamma_{jk} + \tau)$; and at non-leaf nodes, we use a *hard* rule: $d(j) := \min(\gamma_{jk}, \tau)$. We then perform the operation $move(d(j), j \to parent(j))$. If at the top of the tree we have zero counts, we simply put some default statistics with zero mean and unit variance there. This should not happen unless there is a problem with the initial UBM.

We then distribute data down the tree, taking back the same amount of (now averaged) statistics that were previously removed: for $j = N-1 \ldots 1$, do $move(d(j), parent(j) \to j)$. While smoothing down the tree we refrain from removing data from the intermediate nodes by skipping equation (7). This makes no difference to the final statistics at the leaf nodes. The means and variances at the leaf nodes are then estimated using the normal formulae for EM, from the smoothed statistics. For any leaf nodes that have zero counts, we estimate their means and variances from the statistics at the closest parent node that does not have a zero count.

Note that unlike the baseline MAP smoothing, this discount scheme does not ever refer to the original Gaussian parameters μ_k and Σ_k, instead smoothing back to discounted statistics from other speech classes.

3.4. Weight Smoothing

The weight smoothing method has two τ values, a smaller τ_s and a larger τ_l. Similar to the Gaussian update, the method operates in two phases: counts move up the tree in one and move down in the other. However, instead of performing a separate operation per UBM Gaussian index *k* as in the Gaussian smoothing, here a single operation accounts for all the weights.

For data preparation, we initialize the counts γ_{jk} at the leaf nodes $j = 1 \ldots J$, and set zero counts $\gamma_{jk} = 0$ for non-leaf nodes $j = J+1 \ldots N$. In phase one, for each tree node $j = 1 \ldots N$, we first work out the total amount to discount from node *j*,

$$d(j) = \sum_{k=1}^{K} \min(\tau_s, \gamma_{jk}) \tag{8}$$

using the smaller of the two τ values. Then a proportion $p(j)$ is removed from any counts up to a maximum of the larger τ value, where $p(j)$ is computed to achieve the desired discount $d(j)$:

$$p(j) = \sum_{k=1}^{K} \min(\tau_s, \gamma_{jk}) \Big/ \sum_{k=1}^{K} \min(\tau_l, \gamma_{jk}) \,, \tag{9}$$

and the smoothing formulae are,

$$\begin{gathered} \gamma_{parent(j),k} := \gamma_{parent(j),k} + p(j)\min(\tau_l, \gamma_{jk}) \\ \gamma_{jk} := \gamma_{jk} - p(j)\min(\tau_l, \gamma_{jk}) \end{gathered} \tag{10}$$

In phase two, we go in the reverse order $j = N-1\ldots1$, and take back the same amount of counts that was given up to each parent. Again, we do not destroy statistics as we go down the tree,

$$\gamma_{jk} := \gamma_{jk} + \gamma_{parent(j),k} d(j) \Big/ \sum_{k=1}^{K} \gamma_{parent(j),k} \tag{11}$$

Finally, using the smoothed counts we compute the weights in the normal way,

$$w_{jk} := \gamma_{jk} \Big/ \sum_{j=1}^{J} \gamma_{jk} \tag{12}$$

3.5. Gaussian Pruning

Without pruning, the resulting models would be extremely large. We use the global tree mentioned above to prune the Gaussian parameters in a similar way to the entropy-based pruning of language models [4], by attempting to minimize the loss in average likelihood of the data given the pruned model. The method is parameterized by a penalty p (e.g. 10^{-7}), the maximum loss in likelihood allowable to delete one Gaussian. The pruning algorithm operates for each UBM Gaussian k separately. First, using the smoothing methods above we compute μ_{jk}, Σ_{jk} ,and w_{jk} for all $j = 1\ldots J$; and we also compute occupation probabilities p_j for all the speech states j using observation counts.

As in the smoothing algorithms, we have counts γ_{jk} and statistics $\mathbf{x}_{jk}$ and $\mathbf{S}_{jk}$ associated with each node in the tree. We initialize these to zero for the non-leaf nodes $j = J+1\ldots N$, and for the leaf nodes $j = 1\ldots J$,

$$\lambda_{jk} = p_j w_{jk}, \quad \mathbf{x}_{jk} = \lambda_{jk}\boldsymbol{\mu}_{jk}, \quad S_{jk_{ii}} = \lambda_{jk}(\Sigma_{jk_{ii}} + \mu^2_{jk_i}) \tag{13}$$

This algorithm relies on the notion of moving statistics around the tree, much as in the smoothing algorithms in the previous sections. We use the likelihood penalty incurred by combing the statistics from two nodes j_1 and j_2; this is a positive number which we compute as follows:

$$\begin{aligned} p(j_1, j_2) = -0.5\sum_{i=1}^{D} \lambda_{j_1k} \log\left(\frac{S_{j_1k_{ii}}}{\lambda_{j_1k}} - \left(\frac{x_{j_1k_i}}{\lambda_{j_1k}}\right)^2\right) \\ + \lambda_{j_2k} \log\left(\frac{S_{j_2k_{ii}}}{\lambda_{j_2k}} - \left(\frac{x_{j_2k_i}}{\lambda_{j_2k}}\right)^2\right) \\ + (\lambda_{j_1k} + \lambda_{j_2k}) \log\left(\frac{S_{j_1k_{ii}} + S_{j_2k_{ii}}}{\lambda_{j_1k} + \lambda_{j_2k}} - \left(\frac{x_{j_1k_i} + x_{j_2k_i}}{\lambda_{j_1k} + \lambda_{j_2k}}\right)^2\right) \end{aligned} \tag{14}$$

The pruning algorithm is iterative, as follows (for a particular k): (1) Until convergence is reached: *a.* For each non-leaf node j that has zero count λ_{jk} and has more than one child with nonzero count, Move the statistics of the child with the smallest count up to the parent. *b.* For each non-leaf node, work out the nonzero child with the smallest penalty for combining with its parent. If this penalty is smaller than p, combine the child with the parent by moving the child's statistics up to the parent. (2) Until convergence is reached: For any parent node that has exactly one child with zero count, move the parent's statistics down to the child. (3) Finally: For all nodes with nonzero statistics, turn the statistics back into Gaussian parameters:

$$\mu_{jk} := \frac{\mathbf{x}_{jk}}{\lambda_{jk}}, \Sigma_{jk_{ii}} := \frac{S_{jk_{ii}}}{\lambda_{jk}} - \mu^2_{jk_i} \tag{15}$$

This is a greedy algorithm which may not be optimal. We use the pruned parameters by going up the tree from a leaf node until we find estimated parameters, and using those.

To estimate the relative importance of the smoothing and pruning steps, we computed the likelihood loss due to each compared to a normal EM update. A small variance floor was used in the EM baseline for the Gaussian update to avoid infinities due to single counts. The loss accrued from Gaussian smoothing, weight smoothing, and Gaussian pruning was (in natural log) 0.17, 0.004, and 0.06, respectively. This indicates that the smoothed weights are very close to their EM values, thus any adjustment to the weight smoothing formulae is not likely to have much effect.

The UBM structure gives us a convenient way to quickly evaluate the resulting Gaussian mixtures by using a subset of UBM indices k. On each frame we first evaluate all the Gaussians in the original UBM and pick the top n Gaussians, e.g. for n = 4; we take the union of this, and all Gaussians within a certain likelihood threshold from the top, which threshold is set to 4 also. The optimal parameter settings have not been investigated.

3.6. Semi-tied Covariance

Semi-tied covariance (STC) is used in our UBM implementation. A different STC class is used for each UBM Gaussian k. The STC computation is standard and not specific to the UBM framework, but the UBM framework does provide a convenient way to define the STC classes. In our experiments, the STC estimation is carried out on the second and fourth iterations of update. Note that the semi-tied covariance is not applied to the original UBM which is only used to pick the top Gaussian indexes on each frame for pruning, as described above. Because only a small number of Gaussian indexes are used, the semi-tied covariance computation is efficient since we only have to transform the features separately for each of those indexes. However we cannot claim that there is a particularly favorable combination between multiple STC classes and UBMs from a word error rate point of view since the improvements we report from multiple STC classes are similar to those reported in [7] in a normal system.

STC introduces additional complications for fMLLR and MLLR adaptation. In principle, we could use the technique we previously introduced in [5] for full covariance adaptation. However, this would require too much memory in the UBM case. For simplicity, in our speaker-adapted UBM experiments, the fMLLR and MLLR adaptation matrices are derived from a baseline system using intermediate transcriptions generated from an un-adapted UBM-based decoding. This was done whether or not the UBM-based model had semi-tied covariance.

4. EXPERIMENTS

4.1. Experimental Setup

The UBM is implemented in a variant of the IBM Mandarin broadcast transcription system [6].

The acoustic models are continuous mixture density HMMs with context-dependent states conditioned on cross-word quinphone context, built with *maximum likelihood* training on 1,321 hours of broadcast news and broadcast conversation speech released by LDC for the DARPA GALE program. The input audio is sampled at 16 KHz and coded using 13-dememsional PLP features with a 25ms window and 10ms frame-shift; nine consecutive frames are spliced and projected to 40 dimensions using LDA and *maximum likelihood linear transform* (MLLT). *Vocal tract length normalization* (VTLN) is applied. The systems have 15K states; the baseline fMLLR build (SAT-trained) system had 500K Gaussians which was our normal setup but our VTLN only baseline had 1M Gaussians which was about 0.8% better than with 500K Gaussians. The UBM based systems have the same number of states and K=1000, and after pruning at the threshold of 10^{-7} have about the same number of parameters as a system with 2M Gaussians. We smooth with $\tau_g = 20, \tau_s = 2.5, \tau_l = 3.5$. The UBMs were initialized by k-means likelihood-based clustering of the Gaussians in a baseline model, followed by 1 iteration of E-M on the data. UBM-based training was for 7 iterations, iterations 2 and 4 being devoted to STC estimation.

The language model is built by interpolating 20 back-off 4-gram models using modified Kneser-Ney smoothing. The interpolation weights are chosen to optimize the perplexity of a 364K held-out set. In total, 5GB of text data is used in training. The final language model has 6.1M n-grams and 107k words. Recognition experiments were carried out on the *dev*'07 test set defined by the GALE consortium. The set is composed of 2 hours and 32 minutes of Mandarin broadcast speech collected from various TV stations in mainland China, Taiwan, and Hong Kong. There are 44.6K characters in the reference.

4.2. Experimental Results

The recognition results in character error rate (CER) are shown in Table 1 and Fig. 1. At the VTLN level we got 1.3% absolute improvement which was versus an unusually large system; at the fMLLR level we got 1.4% absolute improvement. Probably the "fairest" comparison in the table is between 17.9% (baseline) and 17.5% (UBM) at the VTLN level, which is without STC and where the UBM system is only 2 times bigger than the baseline (we do not believe that a larger baseline would give much improvement).

Table 1. On *dev*'07, UBM gives significant improvement in CER(%) compared with the baseline system.

Systems	Baseline	UBM	UBM w/ STC
VTLN	17.9	17.5	16.6
+fMLLR	16.6	--	15.0
+MLLR	16.2	--	14.8

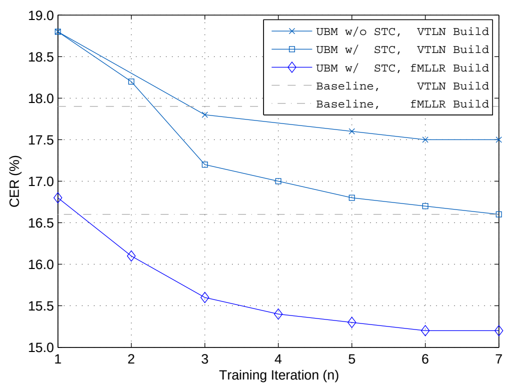

Fig. 1. UBM-based systems consistently outperform the non-UBM baseline at both the VTLN and fMLLR levels.

5. CONCLUSIONS

In this work, we make a first attempt to apply the UBM to acoustic modeling in ASR. We propose a tree-based parameter estimation technique for UBMs, and describe a set of smoothing and pruning methods to facilitate learning. The proposed UBM based approach is benchmarked on a state-of-the-art large-vocabulary continuous speech recognition platform on a broadcast transcription task. Our results at the ML level are substantially better than our previous state-of-the art system. It is possible that most or all of the improvement could have been obtained using a combination of multiple STC classes and more parameters, but nevertheless we believe that the extra structure of the UBM based approach makes it attractive as a starting point for other improvements; it is also very fast to train and test.

REFERENCES

[1] D. A. Reynolds, T. F. Quatieri, and R. Dunn, "Speaker verification using adapted Gaussian mixture models," *Digital Signal Processing*, vol. 10, no. 1-3, pp. 19-41, 2000.

[2] R. Vogt, J. Pelecanos, S. Sridharan, "Dependence of GMM adaptation on feature post-processing for speaker recognition," in *Proc. EUROSPEECH'03*, pp. 3013-3016, September 2003.

[3] M. J. F. Gales, "Semi-tied covariance matrices for hidden Markov models" *IEEE Transactions on Speech and Audio Processing*, vol. 7, no. 3, pp. 272 – 281, 1999.

[4] R. Kneser, H. Ney, "Improved backing-off for M-gram language modeling," in *Proc. ICASSP'95*, vol. 1, pp. 181-184, May 1995.

[5] D. Povey and G. Saon, "Feature and model space feature adaptation with full covariance Gaussians," in *Proc. ICSLP'06*, September 2006.

[6] S. M. Chu, H.-K. Kuo, Y. Y. Liu, Y. Qin, Q. Shi, and G. Zweig, "The IBM Mandarin broadcast speech transcription system," in *Proc. ICASSP'07*, vol. 2, pp. 345-348, May 2007.

[7] K. C. Sim, M. J. F. Gales: "Minimum phone error training of precision matrix models," *IEEE Transactions on Audio, Speech and language Processing*, vol. 14, no. 3, pp. 882-889, 2006.