DYNAMICALLY CONFIGURABLE ACOUSTIC MODELS FOR SPEECH RECOGNITION

Mei-Yuh Hwang and Xuedong Huang

Microsoft Research
1 Microsoft Way
Redmond, WA 98052, USA
mhwang,xueh@microsoft.com

ABSTRACT

Senones were introduced to share Hidden Markov model (HMM) parameters at a sub-phonetic level in [3] and decision trees were incorporated to predict unseen phonetic contexts in [4]. In this paper, we will describe two applications of the senonic decision tree in (1) dynamically downsizing a speech recognition system for small platforms and in (2) sharing the Gaussian covariances of continuous density HMMs (CHMMs). We experimented how to balance different parameters that can offer the best trade off between recognition accuracy and system size. The dynamically downsized system, without retraining, performed even better than the regular Baum-Welch [1] trained system. The shared covariance model provided as good a performance as the unshared full model and thus gave us the freedom to increase the number of Gaussian means to increase the accuracy of the model. Combining the downsizing and covariance sharing algorithms, a total of 8% error reduction was achieved over the Baum-Welch trained system with approximately the same parameter size.

1. INTRODUCTION

One of the challenges to practice speech-recognition research systems is how to utilize users' resources without accuracy degradation. To fit in the platforms most users (especially PC users) have, often a simplified acoustic/language model is provided instead, which results in higher recognition error rates in comparison with the research system. Due to the wide spectrum of users' platforms, one good compromise is to provide the best model and allow the user to dynamically decide how much resource he/she would like to allocate for the speech recognizer. The hierarchy in the senonic decision tree [4] provides a good downsizing architecture. Leaves of the same ancestor node represent Markov states of similar acoustic realization. Based on this principle, we first build the best acoustic model with as many parameters as possible. In other words, the tree is grown to the deepest until it cannot be well trained by the given training corpus. This usually takes a few experiments to plot the histogram of the error rate vs. parameter size on a development test set. We then cluster all the parameters under some pre-chosen ancestor node down to a smaller size, based on the criterion of minimizing the loss in the likelihood of generating the training data. Using the statistics embedded in the acoustic model, we don't need to refer to the original training speech. We will show that this decision-tree based downsizing algorithm actually provides slightly better performance (4%) than the full Baum-Welch trained system with the same reduced parameter size. This downsizing algorithm is not only fast and accurate, it does not require the training speech. Therefore it is particularly useful in real applications to provide the user with the flexibility of dynamically configuring the size of the speech recognizer and trading the accuracy with resource utilization.

While using CHMMs with Gaussian density functions, one should be careful about smoothing the covariances. The shared-Gaussian model [2] was attempting to smooth the covariance through sharing. When the covariance is too small, the model becomes tailored too much to the training data and thus is not robust with new test data. Here we propose to use the hierarchy of the senonic decision tree to guide the sharing of the covariances — only those covariances of leaves under the same pre-selected ancestor node can be tied. Using similar likelihood loss measurement, Gaussian covariances of the same pre-selected ancestor are clustered, while the Gaussian means are still separate. When the covariances are tied, not only we reduce the parameter size, the covariance also becomes more robust. Furthermore, after the covariance size is reduced, we have the luxury of increasing the number of Gaussian means to increase the detailness of the acoustic model. In our experiments, the decision-tree shared covariance model suffers no or very minor performance degradation compared with the unshared full-sized model. When more Gaussian means than covariances are allocated, the performance is improved slightly (4%) over the model which has about the same parameter size but has equal number of Gaussian means and covariances.

In Section 2, we will describe the downsizing algorithm and the clustering objective function in detail. Section 3 introduces the covariance sharing algorithm. Section 4 describes the experimental speech recognition tasks and results. Section 5 summarizes our key findings.

2. THE DOWNSIZING ALGORITHM

Assume the acoustic model of a speech recognizer utilizes the senonic decision-tree based CHMMs with Gaussian mixtures. Suppose there are n_1 senones (tree leaves) in the best system tuned on a particular training and development corpora. Let's call these senones the *deep* senones since they are located at the deepest level of the tree. Assume there are m_1 Gaussians used for each deep senone. Our goal is to reduce the number of Gaussians $(n_1 * m_1)$ to a smaller value so that the speech recognizer runs efficiently on a smaller platform.

To downsize the acoustic model, suppose we would like to re-select the set of senonic decision tree nodes that correspond to n_2 $(\leq n_1)$ senones as the new leaves. Let's call this set of leaves the *shallow* senones, as they are closer to the root. For every

0-7803-4428-6/98 $10.00 © 1998 IEEE

shallow senone s, there associates a set of deep senones who are descendants of s in the tree. Let's donate $A(s)$ = {Gaussians of all the deep senones who are descendants of s}.

We would like to downsize the system by trimming the decision tree to the shallow-senone level and for every shallow senone s, we need to decide the best m_2 Gaussians, where $m_2 \leq |A(s)|$, without involving the training speech. Our algorithm is to merge the $|A(s)|$ deep Gaussians into m_2 Gaussians for every shallow senone s. While merging Gaussians, the objective of minimizing the loss in likelihood is always followed. Since the deep Gaussian parameters are trained in a maximum likelihood (ML) fashion, for every deep Gaussian $G_1 = N(\mu_1, \Sigma_1)$, the Baum-Welch ML estimate is:

$$\begin{aligned} \mu_1 &= \sum_{\mathbf{x}\in\mathbf{X}} \gamma(\mathbf{x})\mathbf{x} \;/\; a \\ \Sigma_1 &= \sum_{\mathbf{x}\in\mathbf{X}} \gamma(\mathbf{x})(\mathbf{x}-\mu)(\mathbf{x}-\mu)^t \;/\; a \end{aligned} \tag{1}$$

where $\mathbf{X}$ = {speech data aligned to Gaussian G_1 with occupancy count $\gamma(\mathbf{x})$ for each data $\mathbf{x}$}, $a = \sum_{\mathbf{x}\in\mathbf{X}} \gamma(\mathbf{x})$ = total occupancy of Gaussian G_1 in the training data. The likelihood of generating the set of data $\mathbf{X}$ is thus:

$$\begin{aligned} L_{\mathbf{X}}(\mathbf{X}|G_1) &= \log \prod_{\mathbf{x}\in\mathbf{X}} N(\mathbf{x};\mu_1,\Sigma_1)^{\gamma(\mathbf{x})} \\ &= -a/2\{d\log 2\pi + \log|\Sigma_1| + d\} \end{aligned}$$

where d is the dimensionality of the feature stream $\mathbf{x}$. Similarly for any Gaussian G_2, in $A(s)$, associated with data $\mathbf{Y}$,

$$L_{\mathbf{Y}}(\mathbf{Y}|G_2) = -b/2\{d\log 2\pi + \log|\Sigma_2| + d\}$$

When both sets of data $\mathbf{X}$ and $\mathbf{Y}$ are modeled by the same Gaussian $G = N(\mu, \Sigma)$, setting the derivative of the Q function in [1] with respect to G to zero gives ML estimate of the following, *assuming the data alignment is fixed*:

$$Q(G) = \sum_{\mathbf{x}\in\mathbf{X}} \gamma(\mathbf{x})\log N(\mathbf{x};\mu,\Sigma) + \sum_{\mathbf{y}\in\mathbf{Y}} \gamma(\mathbf{y})\log N(\mathbf{y};\mu,\Sigma)$$

$$= \sum_{\mathbf{x}} \gamma(\mathbf{x})\{d\log 2\pi + log|\Sigma| + (\mathbf{x}-\mu)^t\Sigma^{-1}(\mathbf{x}-\mu)\}$$

$$+ \sum_{\mathbf{y}} \gamma(\mathbf{y})\{d\log 2\pi + log|\Sigma| + (\mathbf{y}-\mu)^t\Sigma^{-1}(\mathbf{y}-\mu)\}$$

$$\begin{aligned} \Longrightarrow \quad \mu &= \frac{a}{a+b}\mu_1 + \frac{b}{a+b}\mu_2 \\ \Sigma &= \frac{a}{a+b}\{\Sigma_1 + (\mu_1-\mu)(\mu_1-\mu)^t\} \\ &\quad + \frac{b}{a+b}\{\Sigma_2 + (\mu_2-\mu)(\mu_2-\mu)^t\} \end{aligned} \tag{2}$$

Thus the likelihood loss of generating data $\mathbf{X}$ and $\mathbf{Y}$ after G_1 and G_2 are merged (tied) is

$$2\Delta_{1+2} = (a+b)\log|\Sigma| - a\log|\Sigma_1| - b\log|\Sigma_2| \tag{3}$$

The two Gaussians in $A(s)$ that give least likelihood loss are merged to reduce the number of Gaussians left by 1. Notice the new Gaussian G now has a total occupancy of $a+b$ and its aligned data include $\mathbf{X}\bigcup\mathbf{Y}$. The raw data $\mathbf{X}$ and $\mathbf{Y}$ do not need to be available. Only the occupancy count and the Gaussian mean and covariance need to be recorded for each remaining Gaussian. This process continues until m_2 Gaussians are left for every shallow senone s. The mixture weights of these m_2 Gaussians for shallow senone s are the normalized occupancy.

Notice the above clustering is guided by the hierarchy of the senonic decision tree between the deep senones and the shallow senones. Arbitrary clustering across far-away relatives in the tree is discouraged because they represent quite different acoustic realizations. In Section 4 we will show that this reduced model, without re-training, is slightly better than the standard ML trained model which has n_2 senones with m_2 Gaussians per senone. Further Baum-Welch training on the auto-reduced model does not improve the performance as the model is already saturated at a local optimal point. This once more demonstrates the efficiency and accuracy of the downsizing algorithm.

3. THE SHARED COVARIANCE MODEL

One way to smooth the covariances in the Gaussian model is to share them in similar acoustic contexts so that there are enough data to train the shared covariance. When two Gaussians $G_1 = N(\mu_1, \Sigma_1)$ and $G_2 = N(\mu_2, \Sigma_2)$ decide to tie their covariances to Σ_{1+2}, the Q function becomes,

$$Q(\mu_1, \mu_2, \Sigma_{1+2}) =$$

$$\sum_{\mathbf{x}} \gamma(\mathbf{x})\{d\log 2\pi + log|\Sigma_{1+2}| + (\mathbf{x}-\mu_1)^t\Sigma_{1+2}^{-1}(\mathbf{x}-\mu_1)$$

$$+ \sum_{\mathbf{y}} \gamma(\mathbf{y})\{d\log 2\pi + log|\Sigma_{1+2}| + (\mathbf{y}-\mu_2)^t\Sigma_{1+2}^{-1}(\mathbf{y}-\mu_2)\}$$

$\partial Q/\partial\mu_1 = 0$ gives the same estimate as Formula (1) for μ_1. Similarly for μ_2. $\partial Q/\partial\Sigma_{1+2} = 0$ gives

$$\Sigma_{1+2} = \frac{a}{a+b}\Sigma_1 + \frac{b}{a+b}\Sigma_2$$

Again the likelihood loss is computed as Formula (3). The two Gaussians which yield the least likelihood loss after their covariances are tied are chosen to share their covariance. This process repeats as in Section 2. The senonic decision tree is used again to guide the constraint of the clustering: covariances from far-away relative nodes are not allowed to be merged since they are modeling quite different acoustics.

Notice a few differences with Section 2.

- Σ_{1+2} here is not affected by the means while Formula (2) does.
- At the end of the covariance clustering, there are still $n_1 * m_1$ Gaussians with $n_1 * m_1$ means and $n_2 * m_2$ covariances. Neither the Gaussian mean nor the Gaussian mixture weight is changed.
- For further covariance-only merges to proceed, the merged count $a+b$ has to be recorded along with the merged covariance Σ_{1+2}. This count is used purely for the computation of the merge. It is not related to the Gaussian mixture weight. To see this, suppose a third Gaussian G_3 is joining the same

covariance with G_1 and G_2. Setting the derivative of the Q function to zero gives

$$\Sigma_{(1+2)+3} = \frac{(a+b)}{(a+b)+c}\Sigma_{(1+2)} + \frac{c}{(a+b)+c}\Sigma_3$$

Therefore the following notation can summarize the above clustering procedures:

For the merge of the entire Gaussian, including the mean, the covariance, and the mixture weight,

$$(a, \mu_1, \Sigma_1) + (b, \mu_2, \Sigma_2) \Longrightarrow (a+b, \frac{a}{a+b}\mu_1 + \frac{b}{a+b}\mu_2,$$

$$\frac{a}{a+b}\{\Sigma_1+(\mu_1-\mu)(\mu_1-\mu)^t\}+\frac{b}{a+b}\{\Sigma_2+(\mu_2-\mu)(\mu_2-\mu)^t\})$$

For the merge of the covariance only,

$$(a, \Sigma_1) + (b, \Sigma_2) \Longrightarrow (a+b, \frac{a}{a+b}\Sigma_1 + \frac{b}{a+b}\Sigma_2)$$

4. EXPERIMENTS

4.1. Task

All experiments were carried out on the 1994 ARPA Hub 1 unlimited vocabulary speaker-independent task, with the official 65k trigram language model supplied by ARPA. The acoustic training data used was the SI-284 WSJ0 and WSJ1 set. Speech was represented by MFCCs with the 1st-order and 2nd-order differences. Each utterance was normalized by its estimated speech mean and silence mean. Gender-*in*dependent Gaussian acoustic models with diagonal covariances were used throughout all the experiments reported here. Due to time constraint, only results on the H1 DEV94 set is reported. This set consisted of 10 male speakers and 10 female speakers, each uttered about 15 sentences which summed up to 310 utterances in total. Simple word alignment was used to compute the word error rate, which included insertion errors. The phonetic pronunciation dictionary was derived from Carnegie-Mellon University.

4.2. The Downsizing Algorithm

The baseline system we built consisted of 6400 context-dependent (cd) senones, with 12 Gaussians per senone. The regular procedure to train an $n_1 * m_1$ context-dependent model[1] is to

- Use the LBG clustering algorithm [5] on the Markov state segmentation created by an existing model to generate the initial seed for the context-independent (ci) model.
- Use Baum-Welch or Viterbi algorithm to train the ML ci model.
- Train the cd model from the trained ci model.

The word error rate on H1-DEV94 was shown in the first row of Table 1. The second row of Table 1, also trained through Baum-Welch regularly, indicated a 16% error rate increase when the parameter size is reduced by a factor of 3.2.

The third row of Table 1 shows the error rate of using the downsizing algorithm described in Section 2. This system was generated from the 6400*12 system without re-training. It actually performed slightly better than the standard 3000*8 system. This small difference may be insignificant and may due to the fact that the standard 3000*8 system was trained from an inferior starting point [2]. However, it certainly demonstrated that the dynamic downsizing algorithm was accurate and efficient. Further training on the auto-downsized model did not provide further improvement. This once more demonstrated the accuracy of the the downsizing algorithm.

system	H1DEV94	% error increase
6400*12 (ML trained)	8.95%	baseline
3000*8 (ML trained)	10.37%	+16%
3000*8 (auto-downsized)	9.94%	+11%

Table 1: Word error rates of ML trained models vs. auto-downsized models.

4.3. The Shared Covariance Model

To demonstrate that the shared-covariance model does not degrade the performance while saving parameter size, we first downsized the 6400*12 system to 4000*9 Gaussians with the algorithm described in Section 2. The performance of the auto-downsized 4000*9 model is shown in the first row of Table 2. To share the covariance, we set the shallow senone $n_2 = 1500$ and $m_2 = 8$. That is, the 4000*9 = 36k covariances are clustered to 12k covariances, with the guidance of the senonic decision tree hierarchy. The shared covariance model performed almost as well as the unshared model, as row 2 of Table 2 indicates. The loss in likelihood after merge is also shown in the table. To sum up, from downsizing 6400*12 Gaussians to 4000*9 Gaussians, to 1500*8 covariances, the total loss in likelihood is 1.98e+7.

Notice the shared covariance model in Table 2 consisted of 36k Gaussian means and 12k Gaussian diagonal covariances, which ends up $48d$ k parameters plus 36k Gaussian mixture weights, where d is dimensionality of the speech feature vector. The 3000*8 auto-downsized model described in Section 4.2 also had a parameter size of $48d$ k for the Gaussians, but a smaller number of mixture weights (24k). The increased Gaussian means and mixture weights explained the 4% superiority of the shared covariance model. In Table 2, we also compared the likelihood loss of the auto-downsized 3000*8 model. Again the larger likelihood loss (2.60e+7) explained the less accuracy of the model. Therefore, despite that the 4% improvement is small, we are confident that the shared covariance model is superior to the standard model.

Compared the regular ML trained 3000*8 model vs. the 1500*8 shared covariance model, the latter outperformed the former by 8% with minimal memory increase (12000 mixture weights).

One interesting question is if the shared covariance can be applied to the highly accurate 6400*12 model to improve the performance. Unfortunately, we didn't find further meaningful improvement by increasing the number of Gaussian means, as Figure

[1] We will use the notation $n_1 * m_1$ to represent n_1 cd senones, with m_1 Gaussians per senone from now on.

[2] We learn that the initial model in CHMMs can make up to 5% difference in the local optimal performance.

system	H1DEV94	likelihood loss (in $\log_e$)	#parameters
4000*9 (auto-downsized)	9.48%	1.58e+7	72kd + 36k
4000*9 μ's 1500*8 Σ's	9.55%	0.40e+7	48kd + 36k
3000*8 (auto-downsized)	9.94%	2.60e+7	48kd + 24k

Table 2: Word error rates of the shared-covariance model (row 2). The number of parameters includes the Gaussian mean, diagonal covariance, and the mixture weight.

1 illustrates. Perhaps we have reached the upper bound of the parameter size based on the given training corpus, even for the Gaussian means. It could also be due to the less reliability the decision tree became as the tree grew deeper (the training data and the statistics used in the deeper nodes became sparser). However, we notice that when the parameter size is overwhelming as in the 12800*12 system, the shared covariance model is indeed more robust than the unshared full model. Therefore, we believe when more training data is available while memory is still an issue, the shared covariance model will play an important role of improving the performance [3].

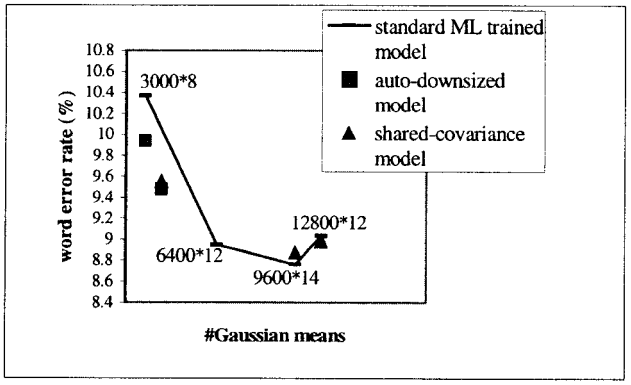

Figure 1: Word error rates vs. #Gaussian means

Another interesting experiment for the shared covariance model is to compare with the performance of the senone-dependent covariance model. The senone-dependent covariance model ties all the covariances of all the Gaussians corresponding to the same cd senone s:

$$\Sigma^{(s)} = \sum_{i=1}^{m_1} \frac{a_i}{\sum_{j=1}^{m_1} a_j} \Sigma_i^{(s)}$$

It has the advantage of saving $d(m_1 - 1)$ multiplications per senone while computing the Gaussian density value. Though indeed faster, it had more than 10% error increase as shown in Table 3. On the other hand, with approximately the same number of shared covariances, the likelihood based sharing had no degradation. This illustrates the importance of keeping the variance of the covariances within the same senone.

system	H1DEV94	% error increase
6400*12 (ML trained)	8.95%	–
6400*1 Σ's (senone-dep)	9.92%	+10.8%
535*12 Σ's (min likelihood loss)	8.96%	+0.1%

Table 3: Word error rates of the shared-covariance model (row 3) vs. that of the senone-dependent covariance model (row 2).

5. CONCLUSION

We propose an efficient and accurate algorithm for dynamically downsizing the acoustic model of a speech recognizer to fit in users' resource requirements. In addition, the shared covariance model can either reduce the memory requirement without an increase in the recognition error rate, or can enhance the performance by re-allocating the space to more Gaussian means and mixture weights. Both algorithms use likelihood loss and decision-tree hierarchy to guide the clustering. Both provide accurate models without re-training and need not to involve the raw training data. Combining both algorithms together yielded a system which had an 8% error reduction over the standard ML trained model, with minimal memory increase.

6. REFERENCES

[1] Baum, L. E. *An Inequality and Associated Maximization Technique in Statistical Estimation of Probabilistic Functions of Markov Processes.* **Inequalities**, vol. 3 (1972), pp. 1–8.

[2] Huang, X., Hwang, M., Jiang, L., and Mahajan, M. *Deleted Interpolation and Density Sharing for Continuous Hidden Markov Models.* in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. 1996.

[3] Hwang, M. and Huang, X. *Subphonetic Modeling with Markov States — Senone.* in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. vol. I, 1992, pp. 33–36.

[4] Hwang, M., Huang, X., and Alleva, F. *Predicting Unseen Triphones with Senones.* in: **IEEE International Conference on Acoustics, Speech, and Signal Processing**. vol. II, 1993, pp. 311–314.

[5] Linde, Y., Buzo, A., and Gray, R. *An Algorithm for Vector Quantizer Design.* **IEEE Transactions on Communication**, vol. COM-28 (1980), pp. 84–95.

[3] A variant of the described system can be downloaded from http://research.microsoft.com.