REVISITING SEMI-CONTINUOUS HIDDEN MARKOV MODELS

K. Riedhammer[1]*, T. Bocklet*[1]*, A. Ghoshal*[2,3]*, D. Povey*[4]

[1]Pattern Recognition Lab, University of Erlangen-Nuremberg, GERMANY
[2]Spoken Language Systems, Saarland University, GERMANY
[3]Centre for Speech Technology Research, University of Edinburgh, UK
[4]Microsoft Research, Redmond, WA, USA

korbinian.riedhammer@informatik.uni-erlangen.de

ABSTRACT

In the past decade, semi-continuous hidden Markov models (SC-HMMs) have not attracted much attention in the speech recognition community. Growing amounts of training data and increasing sophistication of model estimation led to the impression that continuous HMMs are the best choice of acoustic model. However, recent work on recognition of under-resourced languages faces the same old problem of estimating a large number of parameters from limited amounts of transcribed speech. This has led to a renewed interest in methods of reducing the number of parameters while maintaining or extending the modeling capabilities of continuous models. In this work, we compare classic and multiple-codebook semi-continuous models using diagonal and full covariance matrices with continuous HMMs and subspace Gaussian mixture models. Experiments using on the RM and WSJ corpora show that while a classical semi-continous system does not perform as well as a continuous one, multiple-codebook semi-continuous systems can perform better, particular when using full-covariance Gaussians.

Index Terms— automatic speech recognition, acoustic modeling

1. INTRODUCTION

In the past years, semi-continuous hidden Markov models (SC-HMMs) [1], in which the Gaussian means and variances are shared among all the HMM states and only the weights differ, have not attracted much attention in the automatic speech recognition (ASR) community. Most major frameworks (e.g. CMU SPHINX or HTK) have more or less retired semi-continuous models in favor of continuous models, which performed considerably better with the growing amount of available training data.

We were motivated by a number of factors to look again at semi-continuous models. One is that we are interested in applications where there is a small amount of training data (or a small amount of in-domain training data), and since in semi-continuous models the amount of state-specific parameters is relatively small, we felt that they might have an advantage there. We were also interested in improving the open-source speech recognition toolkit KALDI [2], and felt that since a multiple-codebook form of semi-continous models is, to our knowledge, currently being used in a state-of-the-art system [3], we should implement this type of model. We also wanted to experiment with the style of semi-continous system described in [4, 5], particularly the smoothing techniques, and see how these compare against a more conventional system. Lastly, we felt that it would be interesting to compare semi-continous models with Subspace Gaussian Mixture Models (SGMMs) [6], since there are obvious similarities.

In the experiments we report here we have not been able to match the performance of a standard diagonal GMM based system using a traditional semi-continuous system; however, using a two-level tree and multiple codebooks similar to [3], in one setup we were able to get better performance than a traditional system (although not as good as SGMMs). For both traditional and multiple-codebook configurations, we saw better results with full-covariance codebook Gaussians than with diagonal. We investigated smoothing the statistics for weight estimation using the decision tree, and saw only modest improvements. We note that the software used to do the experiments is part of the KALDI speech recognition toolkit [2] and is freely available for download, along with the example scripts to reproduce these results.

In Section 2 we describe the traditional and tied-mixture models, as well as the multiple-codebook variant of the tied system, and describe some details of our implementation, such as how we build the two-level tree. In Section 3 we describe two different forms of smoothing of the statistics, which we experiment with here. In Section 4 we describe the systems we used (we are using the Resource Management and Wall Street Journal databases); in Section 5 we give experimental results, and we summarize in Section 6.

2. ACOUSTIC MODELING

In this section, we summarize the different forms of acoustic model we use: the continuous, semi-continuous, multiple-codebook semi-continuous, and subspace forms of Gaussian Mixture Models. We also describe the phonetic decision-tree building process, which is necessary background for how we build the multiple-codebook systems.

2.1. Acoustic-Phonetic Decision Tree Building

The acoustic-phonetic decision tree provides a mapping from a phonetic context window (e.g. a triphone), together with a HMM-state index, to an emission probability density function (pdf). The statistics for the tree clustering consist of the sufficient statistics needed to estimate a Gaussian for each seen combination of (phonetic context window, HMM-state). These statistics are based on a Viterbi alignment of a previously built system. The roots of the decision trees can be configured; for the RM experiments, these correspond to the phones in the system, and on Wall Street Journal, they correspond to the "real phones", i.e. grouping different stress and position-marked versions of the same phone together.

The splitting procedure can ask questions not only about the context phones, but also about the central phone and the HMM state; the questions about phones are derived from an automatic clustering procedure. The splitting procedure is greedy and optimizes the likelihood of the data given a single Gaussian in each tree leaf; this is subject to a fixed variance floor to avoid problems caused by singleton counts and numerical roundoff. We typically split up to a specified number of leaves and then cluster the resulting leaves as long as the likelihood loss due to combining them is less than the likelihood improvement on the final split of the tree-building process; and we restrict this clustering to disallow combining leaves across different subtrees (e.g., from different monophones). The post-clustering process typically reduces the number of leaves by around 20%.

2.2. Continuous Models

For continuous models, every leaf of the tree is assigned a separate Gaussian mixture model. Formally, the emission probability of tree leaf j is computed as

$$p(\mathbf{x}|j) = \sum_{i=1}^{N_j} c_{ji} \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_{ji}, \boldsymbol{\Sigma}_{ji}) \tag{1}$$

where N_j is the number of Gaussians assigned to j, and the $\boldsymbol{\mu}_{ji}$ and $\boldsymbol{\Sigma}_{ji}$ are the means and covariance matrices of the mixtures.

We initialize the mixtures with a single component each, and subsequently allocate more components by splitting components at every iteration until a pre-determined total number of components is reached. We allocate the Gaussians proportional to a small power (e.g. 0.2) of the state's occupation count.

2.3. Semi-continuous Models

The idea of semi-continuous models is to have a large number of Gaussians that are shared by every tree leaf using individual weights, thus the emission probability of tree leaf j can be computed as

$$p(\mathbf{x}|j) = \sum_{i=1}^{N} c_{ji} \, \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \tag{2}$$

where c_{ji} is the weight of mixture i for leaf j. As the means and covariances are shared, increasing the number of states only implies a small increase in number of parameters even for full-covariance Gaussians. Furthermore, the Gaussians need to be evaluated only once for each $\mathbf{x}$.

Another advantage is the initialization and use of the codebook. It can be initialized and adapted in a fully unsupervised manner using expectation maximization (EM), maximum a-posteriori (MAP), maximum likelihood linear regression (MLLR) and similar algorithms on untranscribed audio data.

For better performance, we initialize the codebook using the tree statistics collected on a prior phone alignment. For each tree leaf, we include the respective Gaussian as a component in the mixture. The components are merged and (if necessary) split to eliminate low count Gaussians and to match the desired number of components. In a final step, 5 EM iterations are used to improve the goodness of fit to the acoustic training data.

2.4. Multiple-Codebook Semi-continuous Models

A style of system that lies between the semi-continuous and continuous types of systems is one based on a two-level tree; this is described in [3] as a state-clustered tied-mixture (SCTM) system. The way we implement this is to first build the decision tree used for phonetic clustering to a relatively coarse level (e.g. 100 leaves). Each leaf of this tree corresponds to a codebook of Gaussians, i.e. the Gaussian parameters are tied at this level. The leaves of this tree are then split further to give a more fine-grained clustering (e.g. 2500 leaves), where for each new leaf we remember which codebook it corresponds to. The leaves of this second tree correspond to the actual context-dependent HMM states and contain the weights. For this type of system we do not apply any post-clustering.

The way we formalize this is that the context-dependent states j each have a corresponding codebook indexed k, and the function $m(j) \rightarrow k$ maps from the leaf index to the codebook index. The likelihood function is now

$$p(\mathbf{x}|j) = \sum_{i=1}^{N_{m(j)}} c_{ji} \, \mathcal{N}\left(\mathbf{x}; \boldsymbol{\mu}_{m(j),i}, \boldsymbol{\Sigma}_{m(j),i}\right) \tag{3}$$

where $\boldsymbol{\mu}_{m(j),i}$ is the i-th mean vector of the codebook associated with j.

Similar to the traditional semi-continuous codebook initialization, we start from the tree statistics and distribute the Gaussians to the respective codebooks using the tree mapping to obtain preliminary codebooks. The target size N_k of codebook k is determined with respect to the occupancies of the respective leaves as

$$N_k = N_0 + \frac{\sum_{l \in \{m(l)=k\}} \text{occ}(l)}{\sum_r \sum_{t \in \{m(t)=r\}} \text{occ}(l)} \left(N - K \cdot N_0\right) \tag{4}$$

where N_0 is a minimum number of Gaussians per codebooks (e.g., 3), N is the total number of Gaussians, K the number of codebooks, and $\text{occ}(j)$ is the occupancy of tree leaf j. The target sizes of the codebooks are again enforced by either splitting or merging the components.

2.5. Subspace Gaussian Mixture Models

The idea of subspace Gaussian mixture models (SGMM) is, similar to semi-continuous models, to reduce the number of parameters by selecting the Gaussians from a subspace spanned by a universal background model (UBM) and state specific transformations. The SGMM emission pdfs can be computed as

$$p(\mathbf{x}|j) = \sum_{i=1}^{N} c_{ji} \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_{ji}, \boldsymbol{\Sigma}_i) \tag{5}$$

$$\boldsymbol{\mu}_{ji} = \mathbf{M}_i \mathbf{v}_j \tag{6}$$

$$c_{ji} = \frac{\exp \mathbf{w}_i^T \mathbf{v}_j}{\sum_l^N \exp \mathbf{w}_l^T \mathbf{v}_j} \tag{7}$$

where the covariance matrices $\boldsymbol{\Sigma}_i$ are shared between all leaves j. The weights w_{ji} and means $\boldsymbol{\mu}_{ji}$ are derived from $\mathbf{v}_j$ together with $\mathbf{M}_i$ and $\mathbf{w}_i$. The term "subspace" indicates that the parameters of the mixtures are limited to a subspace of the entire space of parameters of the underlying codebook. A detailed description and derivation of the accumulation and update formulas can be found in [6]. Note that the experiments in this article are without adaptation.

3. SMOOTHING TECHNIQUES FOR SEMI-CONTINUOUS MODELS

3.1. Intra-Iteration Smoothing

Although an increasing number of tree leaves does not imply a huge increase in parameters, the estimation of weights at the individual leaves suffer from data fragmentation, especially when training models with small amounts of data. The *intra-iteration* smoothing is motivated by the fact that tree leaves that share the root are similar, thus, if the sufficient statistics of the weights of a leaf j are very small, they should be interpolated with the statistics of closely related leaf nodes. To do so, we first propagate all sufficient statistics up the tree so that the statistics of any node is the sum of its children's statistics. Second, the statistics of each node and leaf are interpolated top-down with their parent's statistics using an interpolation weight τ:

$$\hat{\gamma}_{ji} \leftarrow \underbrace{\left(\gamma_{ji} + \frac{\tau}{\left(\sum_k \gamma_{p(j),k}\right) + \epsilon}\,\gamma_{p(j),i}\right)}_{:=\bar{\gamma}_{ji}} \cdot \underbrace{\frac{\sum_k \gamma_{jk}}{\sum_k \bar{\gamma}_{jk}}}_{\text{normalization}}, \qquad (8)$$

where γ_{ji} is the occupancy of the i component in tree leaf j and $p(j)$ is the parent node of j. For two-level tree based models, the propagation and interpolation cannot be applied across different codebooks. A similar interpolation scheme without normalization was used in [4, 5] along with a different tree structure.

3.2. Inter-Iteration Smoothing

Another typical problem that arises when training semi-continuous models is that the weights tend to converge to a poor local optimum over the iterations. Our intuition is that the weights tend to overfit to the Gaussian parameters in such a way that the model quickly converges with the GMM parameters very close to the initialization point. To try to counteract this effect, we smooth the newly estimated parameters $\Theta^{(i+1)}$ with the ones from the prior iteration using an interpolation weight ρ

$$\hat{\Theta}^{(i)} \leftarrow (1-\rho)\,\Theta^{(i)} + \rho\,\Theta^{(i-1)} \qquad (9)$$

which leads to an exponentially decreasing impact of the initial parameter set. We implemented this for all the parameters but found it only helped when applied to just the weights, which is consistent with our interpretation that stopping the weights from overfitting too fast allows the Gaussian parameters to move farther away from the initialization than they otherwise would. We note that other practitioners have also used mechanisms that prevent the weights from changing too fast and have found that this improved results[1].

4. SYSTEM DESCRIPTIONS

The experiments presented in this article used the DARPA Resource Management Continuous Speech Corpus (RM) [7] and the Wall Street Journal Corpus [8].

For both data sets, we first train an initial monophone continuous system with about 1000 diagonal Gaussians (122 states) on a small data subset; with the final alignments, we train an initial triphone continuous system with about 10000 Gaussians (1600 states). The alignments of this triphone system are used as a basis for the actual system training.

[1] Jasha Droppo, personal communication.

Note that this baseline may be computed on any other (reduced) data set and has the sole purpose of providing better-than-linear initial alignments to speed up the convergence of the models. The details of the model training can also be found in the KALDI recipes.

For the acoustic frontend, we extract 13 Mel-frequency cepstral coefficients (MFCCs), apply cepstral mean normalization, and compute deltas and delta-deltas. For decoding, we use the language models supplied with those corpora; for RM this is a word-pair grammar, and for WSJ we use the bigram version of the language models supplied with the data set.

4.1. Resource Management

For the RM data set, we train on the speaker independent training and development set (about 4000 utterances) and test on the six DARPA test runs Mar and Oct '87, Feb and Oct '89, Feb '91 and Sep '92 (about 1500 utterances total). The parameters are tuned to this joint test set since there was not enough data to hold out a development set and still get meaningful results. We use the following system identifiers in Table 1:

- *cont*: continuous triphone system using 9011 diagonal covariance Gaussians in 1480 tree leaves
- *semi*: semi-continuous triphone system using 768 full covariance Gaussians in 2500 tree leaves, no smoothing (explanations later).
- *2lvl*: two-level tree based semi-continuous triphone system using 3072 full covariance Gaussians in 208 codebooks ($4/14.7/141$ min/average/max components), 2500 tree leaves, $\tau = 35$ and $\rho = 0.2$.
- *sgmm*: subspace Gaussian mixture model triphone system using a 400 full-covariance Gaussian background model and 2016 tree leaves.

4.2. WSJ

Due to time constraints, we trained on half of the SI-84 training set using settings similar to the RM experiments; we tested on the eval '92 and eval '93 data sets. As the classic semi-continuous system did not yield good performance on the RM data both in terms of run-time and performance, we omitted it for the WSJ experiments. Results corresponding to the following systems are presented in Table 3:

- *cont*: continuous triphone system using 10000 diagonal covariance Gaussians in 1576 tree leaves.
- *2lvl*: same configuration as for RM (no further tuning).
- *sgmm*: subspace Gaussian mixture model triphone system using a 400 full-covariance Gaussian background model and 2361 tree leaves.

5. RESULTS

5.1. RM

The results on the different test sets on the RM data are displayed in Table 1. The classic semi-continuous system shows the worst performance of the four, which serves to confirm the conventional wisdom about the superiority of continous systems. The two-level tree based multiple-codebook system performance lies in between the continuous and SGMM system.

Keeping the number of codebooks and leaves as well as the smoothing parameters τ and ρ of *2lvl* constant, we experiment with

	ma87	oc87	fe89	oc89	fe91	se92	*avg*
cont	1.08	2.48	2.69	3.46	2.66	5.90	3.38
semi	1.80	3.19	4.72	4.62	4.15	6.88	4.66
2lvl	0.48	1.70	2.46	3.35	1.89	5.31	2.90
sgmm	0.48	2.20	2.62	2.50	1.93	5.12	2.78

Table 1. Detailed recognition results in % WER for the six DARPA test sets using different acoustic models on the RM data set.

covariance	Gaussians	none	inter	intra	both
full	1024	3.70	3.64	3.79	3.74
full	3072	3.01	3.01	3.02	2.90
diagonal	3072	4.13	4.15	4.25	4.35
diagonal	9216	3.22	3.09	3.28	3.20

Table 2. Average % WER of the multiple-codebook semi-continuous model using different numbers diagonal and full covariance Gaussians, and different smoothings on the RM data. Settings are 208 codebooks, 2500 context dependent states; $\tau = 35$ and $\rho = 0.2$ if active.

the number of Gaussians and type of covariance; the results are displayed in Table 2. Interestingly, the full covariances make a strong difference: Using the same number of Gaussians, full covariances lead to a significant improvement. On the other hand, a substantial increase of the number of diagonal Gaussians leads to a similar performance as the regular continuous system. The effect of the two smoothing methods using τ and ρ is not clear from Table 2, i.e. the results are not always consistent. While the inter-iteration smoothing with ρ helps in most cases, the intra-iteration smoothing coefficient τ shows an inconsistent effect. In the future we may abandon this type of smoothing or investigate other smoothing methods.

5.2. WSJ

The results on the different test sets on the WSJ data are displayed in Table 1. Using the parameters calibrated on the RM test data, the *2lvl* system performance is not as good as the continuous or GMM systems, but still in a similar range while using about the same number of leaves but about a third of the Gaussians. Once the number of Gaussians and leaves, as well as the smoothing parameters are tuned on the development set, we expect the error rates to be in between the continuous and SGMM systems, as seen on the RM data.

	eval '92	eval '93	*avg*
cont	12.75	17.10	14.39
2lvl/none	12.92	17.85	14.79
2lvl/intra	13.03	17.56	14.61
2lvl/inter	12.80	17.59	14.61
2lvl/both	13.01	17.59	14.75
sgmm	11.72	14.25	12.68

Table 3. Detailed recognition results in % WER for the '92 and '93 test sets using different acoustic models on the WSJ data; the *2lvl* system was trained with and without the different interpolations.

6. SUMMARY

In this article, we compared continuous and SGMM models with two types of semi-continuous hidden Markov models, one using a single codebook and the other using multiple codebooks based on a two-level phonetic decision tree, similar to [3]. Using the single-codebook system we were not able to obtain competitive results, but using a multiple-codebook system we were able to get better results than a traditional system, for one database (RM). Interestingly, for both the single-codebook or multiple-codebook systems, we saw better results with full rather than diagonal covariances.

Our best multiple-codebook results reported here incorporate two different forms of smoothing, one using the decision tree to interpolate count statistics among closely related leaves, and one smoothing the weights across iterations to encourage convergence to a better local optimum. In future work we intend to further investigate these smoothing procedures and clarify how important they are.

If we continue to see good results from these types of systems, we may investigate ways to speed up the Gaussian computation for the full-covariance multiple-codebook tied mixture systems (e.g. using diagonal Gaussians in a pre-pruning phase, as in SGMMs). To get state-of-the-art error rates we would also have to implement linear-transform estimation (e.g. Constrained MLLR for adaptation) for these types of models. We also intend to investigate the interaction of these methods with discriminative training and system combination; and we will consider extending the SGMM framework to a two-level architecture similar to the one we used here.

7. REFERENCES

[1] X.D. Huang and M.A. Jack, "Semi-continuous hidden Markov models for speech signals," *Computer Speech and Language*, vol. 3, no. 3, pp. 239–251, 1989.

[2] D. Povey, A. Ghoshal, et al., "The Kaldi Speech Recognition Toolkit," in *Proc. IEEE Workshop on Speech Recognition and Understanding (ASRU)*, 2011.

[3] R. Prasad, S. Matsoukas, C.L. Kao, J.Z. Ma, D.X. Xu, T. Colthurst, O. Kimball, R. Schwartz, J.L. Gauvain, L. Lamel, et al., "The 2004 BBN/LIMSI 20xRT English conversational telephone speech recognition system," in *Ninth European Conference on Speech Communication and Technology*, 2005.

[4] E.G. Schukat-Talamazzini, T. Kuhn, and H. Niemann, "Speech Recognition for Spoken Dialog Systems," in *Progress and Prospects of Speech Research and Technology in Proc. Artificial Intelligence*, H. Niemann, R. De Mori, and G. Hanrieder, Eds., 1994, pp. 110–120.

[5] E.G. Schukat-Talamazzini, *Automatische Spracherkennung – Grundlagen, statistische Modelle und effiziente Algorithmen*, Vieweg, Braunschweig, Germany, 1995.

[6] D. Povey, L. Burget, M. Agarwal, P. Akyazi, F. Kai, A. Ghoshal, O. Glembek, N. Goel, M. Karafiát, A. Rastrow, R. Rose, P. Schwarz, and S. Thomas, "The subspace Gaussian mixture model - A structured model for speech recognition," *Computer Speech and Language*, vol. 25, pp. 404–439, 2011.

[7] P. Price, W.M. Fisher, J. Bernstein, and D.S. Pallett, "Resource Management RM1 2.0," Linguistic Data Consortium, Philadelphia, USA, 1993.

[8] J. Garofalo et al., "CSR-I,II (WSJ0,1) Complete," Linguistic Data Consortium, Philadelphia, USA, 2007.