FAST LIKELIHOOD COMPUTATION USING HIERARCHICAL GAUSSIAN SHORTLISTS

Xin Lei, Arindam Mandal, Jing Zheng

Speech Technology and Research Laboratory
SRI International, Menlo Park, CA 94025 USA
{leixin,arindam,zj}@speech.sri.com

ABSTRACT

We investigate the use of hierarchical Gaussian shortlists to speed up Gaussian likelihood computation. This approach is a combination of hierarchical Gaussian selection and standard Gaussian shortlists. First, all the Gaussians are clustered hierarchically. Then, for the Gaussians in each level of the hierarchy, shortlists are trained to reduce likelihood computation at the corresponding level. This approach enables a hierarchical coarse-to-fine control of the Gaussian likelihood computation. The proposed approach is evaluated in computing the high-dimensional posteriors for feature space Minimum Phone Error (fMPE) front end and also in Viterbi search. Experimental results show that the performance of the proposed approach is superior to using only hierarchical Gaussian selection or standard Gaussian shortlists.

Index Terms: Fast likelihood computation, Gaussian shortlist, hierarchical clustering, fMPE.

1. INTRODUCTION

For hidden Markov model (HMM) based large vocabulary automatic speech recognition (ASR) systems, Gaussian likelihood computation is typically the most computationally intensive operation in decoding. It takes about 30-70% of the total recognition time [1]. Besides Viterbi search, the more recently proposed feature space Minimum Phone Error (fMPE) processing [2] also requires computing the likelihoods of a large number of Gaussians. Therefore, fast Gaussian likelihood computation is essential to achieve good real-time performance for a state-of-the-art ASR system.

Various techniques have been proposed to speed up the Gaussian likelihood computation in ASR, as reviewed in [3]. Some techniques exploit modern hardware-based parallel computation, such as using SIMD (Single Instruction Multiple Data) instructions or even using graphics processors [4]. Other approaches seek optimized algorithms to reduce computation at four different levels [5]: frame-level, Gaussian mixture model (GMM)-level, Gaussian-level, and element-level. Frame-level algorithms use a criterion to decide whether the likelihood computation can be skipped for each frame. The authors in [6] also proposed a maximum probability increase estimation algorithm to reduce computation by exploiting the Gaussian likelihood information from previous frames. At the GMM-level, context-independent (CI) HMM based GMM selection [7] was proposed to reduce the number of GMMs for evaluation. At the Gaussian-level, vector quantization (VQ)-based Gaussian selection techniques [8, 1] have been widely used to identify a subset of the most likely Gaussians for evaluation at each frame. Finally, at the element-level, techniques such as partial distance elimination (PDE) [9] can be used to stop the likelihood computation before evaluating all the dimensions of a Gaussian when the accumulated Mahalanobis distance is larger than a dynamic threshold.

In this paper, we propose the use of *hierarchical Gaussian shortlists* (HGS) to improve the performance of standard VQ-based Gaussian selection. A Gaussian shortlist is a subset of Gaussian distributions expected to have high likelihood values for a given acoustic feature vector. We first perform hierarchical Gaussian clustering to merge all the Gaussian component distributions into a number of indexing clusters. Further clustering can be performed on the indexing clusters to form a hierarchical clustering tree. Then, Gaussian shortlists are trained for the Gaussians in each level of the hierarchy to save computation in the corresponding level. Our HGS approach can be seen as a combination of hierarchical Gaussian selection and VQ-based Gaussian shortlists. It enables a hierarchical coarse-to-fine control of the Gaussian likelihood computation. By using a simple two-level hierarchy and clustering Gaussians within the HMM states, HGS can reduce likelihood computation at both GMM-level and Gaussian-level.

The rest of this paper is organized as follows. In Section 2 we briefly review the background on likelihood computation and fMPE feature processing. In Section 3 we propose HGS for fast likelihood computation. In Section 4 and Section 5, we describe the use of HGS in fMPE processing and Viterbi search. Section 6 shows the experimental results in SRI's real-time Dynaspeak™ speech recognition system. Section 7 concludes and discusses future work.

2. BACKGROUND

2.1. Likelihood Computation

For a feature vector x_t, the likelihood of an N-dimensional Gaussian distribution with mean μ and covariance Σ is computed as

$$p(x_t|\mu,\Sigma) = \frac{1}{(2\pi)^{\frac{N}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x_t-\mu)^T\Sigma^{-1}(x_t-\mu)\right). \quad (1)$$

In most speech recognition systems, diagonal covariance and log likelihood are commonly used for computational efficiency and avoiding numerical issues. Let the diagonal covariance be $\Sigma = diag(\sigma_1^2, \sigma_2^2, \cdots, \sigma_N^2)$, the log likelihood becomes

$$\log p(x_t|\mu,\Sigma) = -\frac{1}{2}\sum_{i=1}^{N}\log(2\pi\sigma_i^2) - \frac{1}{2}\sum_{i=1}^{N}\frac{(x_t(i)-\mu_i)^2}{\sigma_i^2} \quad (2)$$

The first item is not related to the feature vector and can be precomputed before decoding. The second item can be further decomposed into a dot-product format, which can be used to compute likelihoods of multiple frames over multiple Gaussians as a matrix-matrix multiplication [10].

2.2. fMPE Feature Processing

The feature space discriminative training technique fMPE was proposed in [2]. It adopts the same objective function as MPE to transform the feature vectors in both training and decoding.

Let x_t denote the original feature vector at time t, the fMPE transformed feature vector is

$$y_t = x_t + Mh_t, \tag{3}$$

where h_t is a high-dimensional posterior probability vector, and M is a matrix mapping the posterior vector onto a lower-dimensional feature space. The projection matrix M is trained to optimize the MPE criterion. The posterior vector h_t is computed first by evaluating the likelihood of the original feature vector along a large set of Gaussians (e.g., all the Gaussians in the acoustic model) with no priors. Then for each frame, the posteriors of the contextual frames are also computed and concatenated with the specified frame to form the final posterior vector.

3. HIERARCHICAL GAUSSIAN SHORTLIST

3.1. Gaussian Shortlists

To prevent a large number of unnecessary Gaussian computations, we use state-based Gaussian shortlists similar to [11, 1]. In our implementation, first, VQ is used to subdivide the acoustic space into a number of VQ regions. The VQ codebooks are organized as a tree to quickly locate the VQ region in which a given input feature vector falls. Then, one list of Gaussians is created for each combination (v, s) of VQ region v and state s. The lists are created empirically, by considering a sufficiently large amount of speech data. For each acoustic observation, every Gaussian distribution is evaluated. Those distributions whose likelihoods are within a predetermined fraction of the most likely Gaussian are added to the list for the corresponding pair (v, s). This is the main difference from the Gaussian selection algorithm proposed by Bocchieri [8], where the shortlist for each VQ region is determined by looking only at the centroid of that region.

3.2. Hierarchical Shortlists

Standard Gaussian shortlists can reduce computation at the Gaussian-level. To reduce the computation at higher levels (e.g., GMM-level) and allow a coarse-to-fine control of the likelihood computation, we propose to use hierarchical Gaussian shortlists.

First, hierarchical Gaussian clustering is performed to cluster all the Gaussian distributions into K clusters/Gaussians, which we refer to as *indexing Gaussians*. Each cluster is represented by a single Gaussian distribution. The clustering criterion used in this work is an entropy-based measure as described in [11]. Further clustering can be performed on the indexing Gaussians to form a hierarchical clustering tree. Then, for each level in the hierarchy, Gaussian shortlists are trained to save computation in the corresponding level.

In this work, we use a simple two-layer shortlist as illustrated in Figure 1. Each of the K indexing Gaussians in the upper layer is associated with a cluster of Gaussians in the lower layer. All the indexing Gaussians can be seen as being in a single pseudo cluster. In addition to the standard shortlists for each cluster in the lower layer, we can also build shortlists for the pseudo cluster for each VQ region. The same VQ tree is shared for both shortlists. For a given feature vector, we first locate the VQ region, then use the shortlist in the upper layer to find the most probable clusters, and finally use the lower-layer shortlists to evaluate the most probable Gaussians in each probable cluster. The use of such hierarchical shortlists allow us to limit likelihood computation to a small fraction of Gaussian distributions, resulting in significant computational savings.

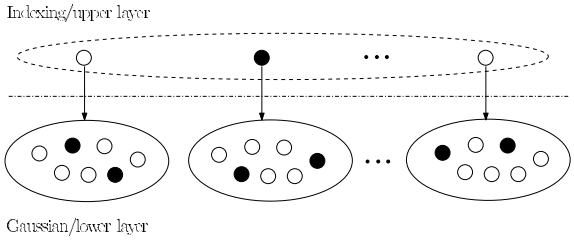

Fig. 1. A two-layer shortlist. The black circles denote active Gaussians in the shortlist for a given feature vector.

4. HGS FOR FMPE PROCESSING

Although fMPE brings significant recognition accuracy improvement [2], it is computationally expensive in its naïve implementation, especially for a real-time ASR system on a portable device.

In our baseline implementation of computing fMPE posterior vectors, we adopt the hierarchical Gaussian selection method as in [2]. All the Gaussians are clustered into K indexing Gaussians. For each feature vector x_t, the likelihoods of the indexing Gaussians are computed and the top M clusters are selected. The likelihoods of all the Gaussians within these M clusters are then evaluated. If we reduce the number of selected clusters M, the speed becomes faster but accuracy begins to degrade as described in Section 6.

Standard shortlists can be used by treating each cluster of Gaussians as a state. However, one issue of using standard shortlist in fMPE is that at each frame, the feature vector still needs to be evaluated for all the indexing Gaussians. If the number of indexing Gaussians K is large, e.g., 2000 as in [2], a large number of Gaussians (at least K) need be evaluated for every frame. To reduce the computation at the indexing level, we propose to use HGS for both indexing layer and lower Gaussian layer as shown in Figure 1.

During fMPE processing, at each frame t, a VQ region is first located according to the original feature vector x_t. When evaluating the likelihood for an indexing Gaussian, we skip the computation if it is not in the upper-layer shortlist. Finally, for each of the top M clusters, we evaluate only the shortlist of Gaussians for this VQ region and cluster pair.

5. HGS FOR VITERBI SEARCH

In Viterbi search, the likelihoods of a large number of active HMM states are evaluated at each frame. We can also use HGS to reduce the likelihood computation in Viterbi search similarly.

The VQ codebook is trained with the fMPE transformed features y_t. We start with clustering all the Gaussians in each tied-state of the acoustic model into a single Gaussian and form a two-layer hierarchy as in Figure 1. This clustering is different from the clustering in fMPE, where the Gaussians may be clustered in a more general way such as k-means. Then, shortlists are trained for the Gaussians in each level of the hierarchy. Finally in decoding, the shortlists in the indexing layer are used to locate the most probable states for evaluation at each frame. For these states, the lower-layer shortlists are

used to select the subset of Gaussians for evaluation within the state. For the remaining states that are not in the indexing-layer shortlist, we can prune these states by assigning a very small probability to the state likelihood. The pruning/skipping strategy works well in fMPE processing. However, the recognition accuracy is much more sensitive to the states pruning in Viterbi search. There are two options to reduce likelihood computation for these states: 1) approximate the state likelihood by the likelihood of the corresponding indexing Gaussian; 2) use a more aggressive shortlist for these states to minimize the computation.

In essence, the indexing layer shortlists of HGS are similar to the CI-HMM based GMM selection [7] except here the selection is done based on prior knowledge learned from the training phase. Therefore, the GMM selection in HGS requires minimal computation at run-time. However, the selection might be less accurate. This may be compensated by using the aggressive lower-layer shortlists instead of pruning the GMMs.

6. EXPERIMENTAL RESULTS

Experiments are performed in SRI's state-of-the-art real-time Dynaspeak™ speech recognition engine in a speech-to-speech translation system [12]. Two test sets are used in the experiments: the offline and online English test sets in the July 2007 DARPA Transtac evaluation. The offline test set contains around 2000 seconds of speech with 568 utterances. The online test set has about 1500 seconds of speech with 403 utterances.

6.1. Baseline System

The front end uses a 16kHz sampling rate with 10ms frame advance rate. Standard Mel frequency cepstral coefficients (MFCC) with first, second and third order derivatives are extracted. Heteroscedastic linear discriminant analysis (HLDA) is then applied to reduce the dimensionality of the feature vector to 39. The features are also transformed with fMPE and adapted with feature space maximum likelihood linear regression (fMLLR). The MPE-trained acoustic model contains 7000 crossword triphone states clustered by decision trees, with 16 Gaussians per state. The decoding vocabulary covers 30K unique words. We use a weighted finite-state transducer (WFST) based static decoding graph constructed in a procedure similar to that described in [13]. A pruned trigram is used for the first pass decoding and generating word lattices. A pruned 4-gram is used to perform fast rescoring of the lattices. We perform recognition experiments on a Windows XP desktop with a 2.4GHz Intel Core™2 CPU and 2G RAM.

Our fMPE implementation follows the original paper [2]. Only posterior features are used, without the offset features as used in later fMPE papers. The offset features can reduce the dimensionality of the posterior vector h_t; however, it introduces more non-zero elements in h_t and may slow down the matrix-vector multiplication of Mh_t. For the fMPE transform, we have around 30K Gaussians for computing the high-dimensional posterior vector. Hierarchical clustering is used to group the 30K Gaussians into 929 indexing clusters, with 32 Gaussians per cluster. The baseline system selects the top M clusters for the feature vector at every frame. The word error rate (WER) results and real-time factors of fMPE processing for different M values are listed in Table 1. Although $M = 100$ is used in fMPE training, reducing M to 50 clusters in decoding does not degrade the recognition performance on either test set. We use $M = 50$ in all the following experiments.

top-M	july07-offline		july07-online	
	WER	RealTime	WER	RealTime
100	13.0	0.091	4.7	0.074
50	13.0	0.075	4.7	0.060
20	13.1	0.056	4.7	0.044

Table 1. Baseline WER (%) and fMPE real-time factors.

S_2	july07-offline		july07-online	
	WER	RealTime	WER	RealTime
10	12.8	0.056	4.7	0.045
5	12.7	0.048	4.6	0.038
2	12.9	0.042	5.0	0.032

Table 2. WER (%) and real-time factors of fMPE with lower-layer shortlists. M is set to 50.

To speed up the likelihood computation at the element-level, we use Intel's Streaming SIMD Extensions (SSE) intrinsics to parallelize the computation in Equation 2. Four dimensions in the multi-dimensional Gaussian are processed at the same time in one operation. The addresses of the feature, mean and variance arrays need to be aligned at 16-byte boundaries to speed up the data loading into the 128-bit registers.

6.2. HGS for fMPE

Since the baseline fMPE implementation consumes noticeable CPU time and may consume several times more on a portable small platform, we first use standard Gaussian shortlists to speed up the posterior vector computation. A 1024-codeword VQ codebook is used. The shortlists are trained on 2 hours of speech data. In decoding, the top M clusters are chosen at each frame. Then according to the VQ-partition of the feature vector and the cluster index, only the shortlist of Gaussians are evaluated. The length of the shortlist (number of Gaussians in the shortlist) for the lower-layer shortlists is denoted as S_2. We run experiments with different shortlist lengths. Results are shown in Table 2. As we can see, the WER begins to degrade when the shortlist length is reduced to 2. Interestingly, we find the use of shortlists with length of 5 actually improves WER slightly on both test sets with a significant speedup.

Table 1 and Table 2 show that neither hierarchical clustering nor lower-layer shortlists can further reduce the fMPE posterior computation without sacrificing the WER performance. We then apply hierarchical Gaussian shortlists in both layers. In the following experiments, we fix the lower-layer shortlist size S_2 to be 5 and try different shortlist sizes S_1 for the indexing layer. The results are listed in Table 3. As we can see, by using a shortlist of length 100 in the indexing layer, the real time factor is reduced from 0.048 to 0.029 on the offline test set and similarly on the online test set, without degrading the recognition accuracy.

Lastly, as shown in Table 3, our efficient SIMD-based programming improves the fMPE speed by another 20% relative. Overall, we speed up fMPE processing by more than 3 times.

6.3. HGS for Viterbi Search

We also try HGS to speed up the Gaussian evaluation in Viterbi search. In the experiments, we find the recognition accuracy de-

S_1	july07-offline		july07-online	
	WER	RealTime	WER	RealTime
200	12.8	0.032	4.9	0.028
100	12.7	0.029	4.7	0.025
50	13.1	0.026	4.8	0.022
+ SIMD				
100	12.7	0.023	4.7	0.020

Table 3. WER (%) and real-time factors of fMPE with two-layer shortlists. M is set to 50, and S_2 is set to 5.

p_1	p_2	p_3	july07-offline		july07-online	
			WER	RealTime	WER	RealTime
-	-	-	12.7	0.353	4.7	0.186
-	0.999	-	12.7	0.278	4.7	0.145
-	0.96	-	12.8	0.261	5.0	0.137
-	0.92	-	13.1	0.251	5.2	0.132
0.90	0.999	0.94	12.6	0.263	4.8	0.136
0.90	**0.999**	**0.90**	12.6	**0.257**	4.9	**0.133**
0.90	0.999	0.85	12.9	0.251	4.9	0.130
0.94	0.999	0.90	12.5	0.264	4.9	0.135
0.82	0.999	0.90	12.8	0.253	5.0	0.130

Table 4. WER (%) and real-time factors of Viterbi decoding with two-layer shortlists. "-" denotes no shortlist is used (all Gaussians are evaluated).

grades significantly if the upper-layer shortlists are used to prune states or if the state likelihoods are approximated by the likelihood of the corresponding indexing Gaussian. Therefore, we choose to use three sets of shortlists: one set for the indexing layer to select important states, one set for the selected states in the lower layer, and another set of aggressive lower-layer shortlists with very short length for the states that are not selected. In addition, we have used a different algorithm to train the shortlist by choosing the Gaussians covering the top p percent of the probability mass of having high likelihood values in evaluation, instead of choosing only the top n most likely Gaussians. This typically results in longer but more accurate shortlists. The percentage used for the upper-layer shortlists is denoted as p_1, p_2 for the lower-layer shortlists of the selected states, and p_3 for the aggressive lower-layer shortlists for the states that are not selected.

The WER and real-time results of the Viterbi search are shown in Table 4. For rows 3-5 in the table, only a single set of lower-layer shortlists is used. All the experiments use SIMD-based likelihood computation. The best setting for HGS is: $p_1 = 0.90$, $p_2 = 0.999$, $p_3 = 0.90$. At the same accuracy level, the real-time factor is improved by 27% over the baseline without shortlists, and 7.6% over using only lower-layer shortlists on the offline test set.

7. CONCLUSIONS AND FUTURE WORK

We have presented an approach of using hierarchical Gaussian shortlists for fast Gaussian likelihood computation. Compared to the standard Gaussian shortlists, the hierarchical shortlists can reduce computation at higher indexing levels. These shortlists are trained to minimize the probability loss. In the decoding phase, a VQ tree is used to quickly quantize the feature vectors for each frame and select the shortlist for each level. The proposed HGS technique is applied to speed up the high-dimensional posterior vector computation in fMPE feature extraction and the likelihood computation in Viterbi search. Significant speed improvements are achieved without apparent accuracy degradation. Future work includes investigating HGS with multiple levels of Gaussian clustering.

8. ACKNOWLEDGEMENTS

The authors thank Dimitra Vergyri for her help in building the baseline English system. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. NBCHD040058. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA.

9. REFERENCES

[1] M. J. F. Gales, K. M. Knill, and S. J. Young, "State-based Gaussian selection in large vocabulary continuous speech recognition using HMMs," *IEEE Trans. on Speech and Audio Processing*, vol. 7, pp. 152–161, 1999.

[2] D. Povey, B. Kingsbury, L. Mangu, G. Saon, H. Soltau, and G. Zweig, "fMPE: Discriminatively trained features for speech recognition," in *Proc. ICASSP*, 2005, pp. 961–964.

[3] J. Cai, G. Bouselmi, Y. Laprie, and J.-P. Haten, "Efficient likelihood evaluation and dynamic Gaussian selection for HMM-based speech recognition," *Computer Speech and Language*, vol. 23, pp. 147–164, 2009.

[4] P. R. Dixon, T. Oonishi, and S. Furui, "Fast acoustic computations using graphics processors," in *Proc. ICASSP*, 2009, pp. 4321–4324.

[5] A. Chan, J. Sherwani, M. Ravishankar, and A. Rudnicky, "Four-layer categorization scheme of fast GMM computation techniques in large vocabulary continuous speech recognition systems," in *Proc. Interspeech*, 2004, pp. 689–692.

[6] N. Morales, L. Gu, and Y. Gao, "Fast Gaussian likelihood computation by maximum probability increase estimation for continuous speech recognition," in *Proc. ICASSP*, 2008, pp. 4453–4456.

[7] A. Lee, T. Kawahara, and K. Shikano, "Gaussian mixture selection using context-independent HMM," in *Proc. ICASSP*, 2001, pp. 69–72.

[8] E. Bocchieri, "Vector quantization for the efficient computation of continuous density likelihood," in *Proc. ICASSP*, 1993, pp. 692–695.

[9] C.-D. Bei and R. Gray, "An improvement of the minimum distortion encoding algorithm for vector quantization," *IEEE Trans. on Communications*, vol. 33, pp. 1132–1131, 1985.

[10] M. Saraclar, M. Riley, E. Bocchieri, and V. Goffin, "Towards automatic close captioning: low latency real-time broadcast news transcription," in *Proc. ICSLP*, 2002, pp. 1741–1744.

[11] V. V. Digalakis, P. Monaco, and H. Murveit, "Genones: generalized mixture tying in continuous hidden Markov model-based speech recognizers," *IEEE Trans. on Speech and Audio Processing*, vol. 4, pp. 281–289, 1996.

[12] M. Akbacak, H. Franco, M. Frandsen, S. Hasan, H. Jameel, A. Kathol, S. Khadivi, X. Lei, A. Mandal, S. Mansour, K. Precoda, C. Richey, D. Vergyri, W. Wang, M. Yang, and J. Zheng, "Recent advances in SRI's IraqComm™ Iraqi Arabic-English speech-to-speech translation system," in *Proc. ICASSP*, 2009, pp. 4809–4812.

[13] M. Mohri, F. Pereira, and M. Riley, "Weighted finite-state transducers in speech recognition," *Computer Speech and Language*, vol. 16, pp. 69–88, 2002.