A Comparative Study of Gaussian Selection Methods in Large Vocabulary Continuous Speech Recognition[*]

Dirk Gehrig

interACT, Universität Karlsruhe (TH), Germany
dgehrig@ira.uka.de

Thomas Schaaf

interACT, Carnegie Mellon University, USA
tschaaf@cs.cmu.edu

Abstract

Gaussian mixture models are the most popular probability density used in automatic speech recognition. During decoding, often many Gaussians are evaluated. Only a small number of Gaussians contributes significantly to probability. Several promising methods to select relevant Gaussians are known. These methods have different properties in terms of required memory, overhead and quality of selected Gaussians. Projection search, bucket box intersection, and Gaussian clustering are investigated in a broadcast news system with focus on adaptation (MLLR).

Index Terms: speech recognition, LVCSR, Gaussian selection, speaker adaptation, MLLR.

1. Introduction

The most popular probability density functions used in speech recognition are Gaussian mixtures. To compute the likelihood for an active HMM-state during decoding the associated Gaussian mixture (GM) has to be evaluated. Only a relatively small number of Gaussians that is close to the observation x has a chance to significantly contribute to the likelihood. Even in a system with well-tuned beams, 30%-50% of the time can be spent in evaluating the acoustic model. To accelerate this part of decoding, several techniques have been developed to find the relevant Gaussians quickly. In many speech recognition systems adaptation techniques are used to improve performance. We compare three substantially different methods to find out how they are affected by adaptation: projection search, bucket box intersection, and clustering.

2. Fast Gaussian Selection Methods

Gaussian selection (GS) methods can be divided into two categories. The first category selects the most promising Gaussians based on a precalculated data structure (bucket box intersection and clustering), whereas the second category operates directly on the feature space (projection search). It is quite obvious that methods with precalculated data structures yield better results, but such data structures need to be updated after adaptation of the acoustic model.

2.1. Projection Search

Projection search [1] is based on a dynamic partitioning of the feature space and therefore needs almost no additional memory. The Gaussians that are located in a rectangular subspace of size 2ϵ around the feature vector are identified during decoding without any precalculation. This is done by limiting each dimension of the feature space by two hyperplanes orthogonal to the according coordinate axes. The hyperplanes have distance ϵ to the observation vector. According to Ortmanns [2], it is sufficient to limit only the first 4 to 7 dimensions of an LDA-transformed feature space. GS starts with the first dimension and proceeds iteratively to the next higher dimensions. If no Gaussian is left inside the hypercube at any point a back-off Gaussian has to be selected. In our implementation, the closest Gaussian according to the last examined dimension is used.

2.2. Bucket Box Intersection

The Bucket Box Intersection (BBI) algorithm [3] divides the feature space into 2^d rectangular regions (buckets). The regions are organized in a K-dimensional space partitioning tree (K-d tree) developed by Bentley [4]. At every non-terminal node, the current region is divided into two half-spaces by means of a hyperplane orthogonal to one of the K coordinate axes. Any vector x can now be located with respect to the dividing hyperplane by a single scalar comparison. A sequence of d scalar comparisons leads to the leaf containing the vector x. The GM computation can be restricted to the evaluation of the Gaussians in the bucket. The assignment of a Gaussian to a bucket is based on the rectangular box, which contains the part of a Gaussian, with a diagonal covariance matrix, that is above a certain threshold γ. Each Gaussian is assigned to every bucket, with which the Gaussian box intersects. If a bucket does not contain any Gaussian of a certain GM, the nearest Gaussian of the GM is assigned to the bucket.

2.3. Gaussian Clustering

The preprocessing of Gaussian clustering partitions the feature space into cells, and a centroid is calculated for each cell. During evaluation, only the Gaussians of one or a few cells with centroids closest to the feature vector are evaluated. This method was first published by Boccherie [5]; many systems use variations [6, 7, 8]. In addition to the standard k-means clustering, we also used k-means-like clustering with Kullback-Leibler (KL) divergence [7]. Instead of k-means clustering, top-down clustering could also be used [8]. In our approach, the clusters are disjoint and each Gaussian only belongs to the closest centroid, whereas other methods assign a Gaussian e. g. to all centroids that are closer than a certain threshold. If no Gaussian of a GM belongs to any of the $topN$ selected clusters during decoding of a HMM-state, a back-off value has to be used. We tested different approaches: a constant value,

[*]This work has been supported by the Deutsche Forschungsgemeinschaft (DFG) within Collaborative Research Center 588 "Humanoid Robots – Learning and Cooperating Multimodal Robots."

the value of the $(topN + 1)$st centroid Gaussian and the value of the Gaussian closest to the first centroid, which can be preselected. We found that the back-off scheme is crucial to achieve a good performance with a small number of selected Gaussians. Therefore, in the experiments a back-off Gaussian is used. When performing speaker adaptation the cluster centroids, the assignment of the Gaussians to the centroids and the back-offs should also be updated. Since the update of the assignment and the back-offs is expensive in time, we only recalculate the centroids based on the transformed Gaussians and the assignment of the initial unadapted acoustic model.

2.4. Measurements

For the evaluation of the GS methods, we measured different values of each system. Word Error Rate (WER) was measured using sclite from NIST's Speech Recognition Scoring Toolkit. To measure time complexity, we counted the average number of evaluated Gaussians Per Frame normalized by the total number of Gaussians in the acoustic model (GPF). This also indicates the quality of the selected Gaussians. Methods or settings with a smaller WER and the same GPF select better Gaussians. Since the overhead for finding the nearest Gaussians is not taken into account by GPF, we also measured the Real-Time Factor (RTF) of the speech recognizer without start-up time. The RTF is normalized by the length of the decoded audio data.

3. Baseline System

The baseline system is a Modern Standard Arabic (MSA) Broadcast News (BN) recognizer based on the Janus Recognition Toolkit (JRTk) [9] and the single pass Ibis decoder [10]. First, automatic segmentation followed by clustering of the speech segments found is performed. The same signal processing as [11] is used: 13 MFCC coefficients, cluster based cepstral mean and variance normalization, concatenation of +/-7 MFCC frames and linear discriminant analysis (LDA) reduce the dimensionality to 42. On top of the LDA transform, a single semi-tied covariance (STC) [12] matrix is used. Two context-dependent fully continuous acoustic models each with 3000 Codebooks and a total of 160k Gaussians were trained on 85h of audio (FBIS + TDT4 distributed by LDC) using maximum likelihood criteria. The first model is a speaker independent model used in the first decoding pass (SI-pass), the model used in the second decoding pass (SA-pass) is a speaker-adaptive trained one (SAT). At each iteration a single constrained MLLR (cMLLR) per speaker is estimated [13]. The SI-pass hypotheses are used to adapt the SAT-model of the SA-pass. For each speaker cluster, one cMLLR transform is estimated; depending on the amount of data, multiple MLLR transformations are used to adapt the means [13, 14]. With this adapted model, the final hypotheses are decoded.

Both decoding passes use a 3-gram language model (test set perplexity 372) trained on the MSA Gigaword corpus from LDC with a vocabulary of 115k words beeing used. The evaluation of the GM models uses a top-1 approximation and the beams of both passes are optimized for speed without loss in WER. The experiments are performed on the DARPA RT04f evaluation set; Table 1 shows the WER for both passes, the average GPF, the RTF and how much time was spent to compute the acoustic scores (AS-Time). AS-Time allows to estimate how much speedup can be achieved by reducing the score computation. It must be taken into account, however, that the number of active states, the number (and quality) of Gaussians to be evaluated, and the number of language model queries affect each other during decoding. As a consequence the RTF may increase if the pruning is too tight.

RT04f	WER	GPF	RTF	AS-Time
SI-pass	35.7%	58%	4.4	37%
SA-pass	29.4%	43%	2.5	46%

Table 1: Performance measures for baseline system

4. Experiments

To compare the different GS methods, the parameter spaces of the methods were explored extensively for SI-pass and SA-pass. In this paper, we focus on experiments that explore the behavior of the SA-pass after adaptation and relate it to the behavior of the SI-pass. To achieve a common ground for adaptation, the same hypotheses of the baseline SI-pass are used during adaptation instead of the hypotheses from a fast system. In our experiments, we found that if we sacrificed some WER (1-2%) in the SI-pass to obtain the hypotheses quickly the WER of the SA-pass is unaltered. However, if the WER of the SI-pass degrades too much, this also affects the SA-pass: first the system becomes slower and then WER starts to increase.

In the experiments, we consider a difference in WER of up to 0.5% absolute (2% relative) as not significant compared to the baseline system.

4.1. Projection Search

Projection search has two main parameters, which have to be tuned to achieve optimal performance: the number of evaluated dimensions N and the threshold ϵ. In our experiments, we varied both systematically and - as expected - found that with increasing N and decreasing ϵ the evaluated GPF drops. If the GPF falls below a certain value more or less independent of N and ϵ, the WER begins to increase quickly. Figure 1 shows this effect for the SA-pass, but the behavior is the same for both decoding passes. A small GPF with a low WER can be achieved by using a comparatively large ϵ and limiting an appropriate large number (e. g. 9) of dimensions.

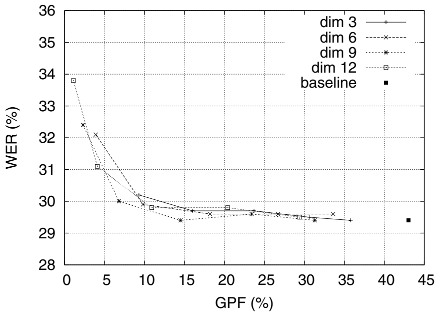
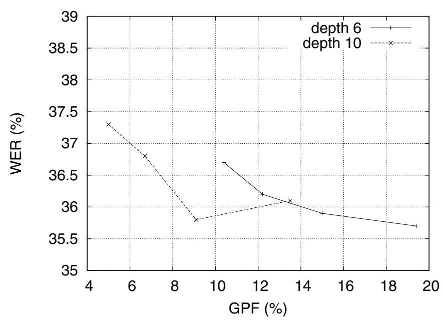


Figure 1: Projection search: Quality of Gaussians (SA-pass)

Unfortunately, the overhead of a large N and ϵ reduces the achievable speedup and can even lead to an increase of the RTF. The best trade-off for the SI-pass is to use only the first 3 dimensions, which reduce the RTF by 15% relative. A similar RTF re-



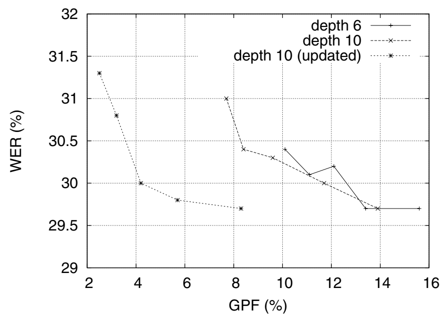


Figure 2: BBI: Quality of Gaussians (top SI-pass, bottom SA-pass)

duction (17%) can be achieved during the SA-pass, but here it is necessary to consider a slightly higher number of dimensions to achieve the same performance. This might be because after adaptation to the SAT acoustic model, more dimensions may be equally important. The experiment shows that projection search can be applied successfully in combination with adaptation. Since the overhead of this method depends on the number of bounded dimensions and the threshold ϵ only a small number of dimensions should be examined with a small threshold.

4.2. Bucket Box Intersection

The BBI algorithm has two parameters that can be tuned, the depth d of the tree and the threshold γ, which controls the size of the Gaussian boxes. With large boxes, it is more likely that a Gaussian is assigned to multiple leafs of the tree. We explored various depths, each with varying thresholds, and found out that for the SI-pass the best performance is achieved with deeper trees and a γ value for large boxes, and that for flat trees more GPFs are evaluated (see figure 2 top)

The situation changes if adaptation is applied in the SA-pass. Because cMLLR can be applied in the feature space to let the observation x look more like the model expects it to be, BBI is not affected. If the acoustic model changes due to MLLR adaptation, the BBI-tree needs to be changed. Since no algorithm is known to update the tree after adaptation, it is necessary to recompute the BBI-tree for each speaker. Often not much data per speaker is available for the BN-domain; i.e. automatic clustering found 34 speakers in the RT04f eval set. The time to build individual BBI-trees for each of them would add at least 50% of the time the baseline system needs for decoding. This overhead would make the overall system slower than the baseline, but if sufficient data of one speaker is available this is the best approach (see figure 2 bottom). Nevertheless, it is possible to speed up the SA-pass with a single precomputed BBI-tree. To make sure that an adapted Gaussian which contributes significantly to the acoustic score falls into the selected leaf, the Gaussian boxes should be large or the tree should be shallow. Figure 2 bottom demonstrates that both depths achieve the same WER for the same number of GPFs, i. e. a BBI-tree of depth 10 selects Gaussians with the same quality as the BBI-tree of depth 6. Therefore it makes no sense to use deeper BBI-trees in the SA-pass, since they have more overhead and need more memory. With a depth of 6 and a WER of 29.7%, we achieved a relative speedup of 19%.

Theoretically it doesn't seem reasonable to use BBI if adaptation is used at the same time, but it is possible to gain a speedup without significant loss of performance with the right choice of parameters.

4.3. Gaussian Clustering

Our clustering algorithm has got three parameters: the number of clusters N, the number of evaluated clusters $topN$ and the distance measure. If Euclidean distance is used for clustering, the same distance is used to find the $topN$ centroids. If KL-distance is used during clustering, the centroids are actual Gaussians with a diagonal covariance matrix and the likelihood is used to find the $topN$ centroids. We performed tests varying N and $topN$ using both distances. As expected fewer GPF can achieve the same WER with increasing N for both decoding passes. However, the overhead to find the $topN$ centroids increases with N and, for the SI-pass, 1024 clusters were optimal for both distances. With KL-clusters a speedup of 26% relative was achieved and an even better speedup (33%) for Euclidian distance, because it is less expensive to compute.

Similar to BBI, adaptation with cMLLR in the feature space is unproblematic. Theoretically, a single MLLR transform could also be used to transform the centroids. However, we did not investigate this approach, because we used multiple transformations. In our experiments with the SA-pass, we investigated if an update of the centroids is beneficial. The quality of the selected Gaussians improves after updating the centroids. In figure 3, the graph "Divergence (no update)" shows that the quality of the Gaussian selection is low compared to the "Divergence" with update. With the update, the same fast settings can be used as during the SI-pass and therefore a better speedup can be achieved. The figure also shows that by using the same number (1024) of clusters the quality of the selected Gaussians degrades consistently if Euclidian distance is used. However, the overhead is smaller and therefore the speedup is the same. Figure 3 also shows that with a larger (2048) number of clusters, the Euclidian distance improves the selection of Gaussians over the KL-distance. The smaller overhead of the Euclidian distance and a low GPF of 6% due to the 2048 clusters achieve the best speedup overall of 41% with a WER of 29.4%.

This shows that it is important to update the cluster centroids in the SA-pass, but it does not seem necessary to update the assignment of the Gaussians to the centroids or the assigned back-off Gaussians. It also shows that the simple Euclidean distance can be used with our preprocessing. However, this can vary depending on the type of features. If a tree-like structure such as a BBI-tree is used to find the $topN$ clusters, we expect the advantage of Euclidian distance to disappear. To compute a BBI-tree for only a small number of centroid Gaussians is not time-consuming and can be done after every adaptation. However, if a layered clustering is used, the same technique to update the centroids can be used recursively.

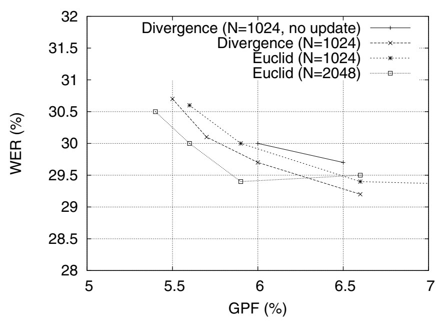


Figure 3: Clustering: Quality of Gaussians (SA-pass)

SI-pass			
	Proj. search	BBI	Clusters (Eucl.)
WER	35.8%	35.8%	35.5%
Speedup	15%	19%	33%
GPF	19%	9%	3%

SA-pass			
	Proj. search	BBI	Clusters (Eucl.)
WER	29.9%	29.7%	29.4%
Speedup	17%	19%	41%
GPF	13%	11%	6%
Adaptable	-	NO	YES

Table 2: Best speedup for the different Gaussian selection methods

4.4. Comparison

We are especially interested in speedup if adaptation (cMLLR, MLLR) is applied. Table 2 compares the performance of the different methods with the best parameter settings.

The advantage of projection search over BBI and Gaussian clustering is that it can be applied easily after adaptation, because it computes the selected Gaussians based exclusively on the current model. It achieves a consistent reduction of the RTF of about 15% relative for both decoding passes. However, the quality of the GS is not as high as for BBI or Gaussian clustering. In addition, after adaptation more dimensions need to be evaluated, which results in higher costs.

Using one global BBI-tree for the acoustic model allows to find the list of relevant Gaussians for a given feature x with the smallest overhead of all methods in the SA-pass. Since no method is known to adapt a BBI-tree, it seems necessary to recompute the tree after each adaptation. This is only suitable if the amount of audio per speaker to decode is large enough. If this is too time-intensive, an alternative is to use a precomputed BBI-tree with very robust settings. However, this limits the speedup that can be achieved. The number of evaluated GPF with such a BBI-tree is smaller than for projection search and the speedup is higher compared to projection search.

Gaussian clustering and the use of Gaussians of the $topN$ clusters gave the best speedup in both decoding passes compared to the other methods. One advantage of Gaussian clustering is the possibility to adapt the centroids quickly after transforming the acoustic model by MLLR. This allows to use the same fast settings as in the SI-pass and achieves the smallest number of GPFs while maintaining the WER of the baseline system.

5. Conclusion

We compared projection search, bucket box intersection and Gaussian clustering with a focus on adaptation and we found that all methods improve decoding speed regardless of the use of adaptation.

Overall, Gaussian clustering results in the best speedup compared to the other methods for both decoding passes. We described how to update the precomputed data structure quickly after adaptation and found this to be important to achieve improved performance.

Projection search presents no problems with adapted acoustic models and seems most suitable for systems with very tight memory constraints. For systems with large vocabularies, large language models, and large acoustic models, memory usually is not a constraint and therefore other methods are preferable.

A global BBI-tree finds relevant Gaussians with a small overhead. That is less effective after MLLR adaptation if the tree is not recomputed. Nevertheless, a single shallow unadapted tree with large boxes still yields a good speedup even after adaptation, but is not as effective as Gaussian clustering.

6. References

[1] S. Nene and S. Nayar. *Closest point search in high dimensions.* In Proc. CVPR '96, pp. 859-865, 1996.

[2] S. Ortmanns. *Effiziente Suchverfahren zur Erkennung kontinuirlich gesprochener Sprache*. PhD thesis, Rheinisch-Westfälischen Technischen Hochschule Aachen, Nov. 1998.

[3] M. Woszczyna and J. Fritsch. *Codebuchübergreifende bucket-box-intersection zur schnellen Berechnung von Emissionswahrscheinlichkeiten im Karlsruher VM-Erkenner.* Verbmobil, July 1997.

[4] J. L. Bentley. *Multidimensional binary search trees used for associative searching.* Commun. Ass. Comput. Mach., vol. 18, no. 9, pp. 509-517, Sept. 1975.

[5] E. Bocchieri. *Vector Quantization for the effizient computation of continuous density likelihoods.* In Proc. ICASSP '93, Minneapolis, 1993.

[6] K. Knill, M. Gales, S. Young. *Use of Gaussian Selection in Large Vocabulary Continuous Speech Recognition Using HMMs* . In Proc. ICSLP'96 Philadelphia, PA USA 1996.

[7] T. Watanabe, K. Shinoda, K. Takagi, K. Iso. *High Speed Speech Recognition Using Tree- Structured Probability Density Function.* In Proc. ICASSP'95, Detroit, USA, 1995.

[8] G. Saon, D. Povey, G. Zweig. *Anatomy of an Extremely Fast LVCSR Decoder*. In Proc. Interspeech/Eurospeech 2005, Lisbon, Portugal, 2005.

[9] M. Finke, P. Geutner, H. Hild, T. Kemp, K. Ries, and M. Westphal. *The Karlsruhe Verbmobil Speech Recognition Engine*. In Proc. ICASSP 97. München; Germany, 1997.

[10] H. Soltau, F. Metze, C. Fügen, and A. Waibel. *A Onepass Decoder Based on Polymorphic Linguistic Context Assignment*. In Proc. ASRU 2001. Madonna di Campiglio, Italy, 2001.

[11] H. Yu and A. Waibel. *Streaming the Front-End of a Speech Recognizer*. In Proc. ICSLP-2000. Beijing; China, 2000.

[12] M. Gales. *Semi-Tied Covariance Matrices for Hidden Markov Models*. IEEE Transactions on Speech and Audio Processing, Vol. 2, May 1999.

[13] M. Gales. *Maximum Likelihood Linear Transformations for HMM-based Speech Recognition*. Computer Speech and Language Vol. 12, 1998.

[14] C.J. Leggetter, P.C. Woodland. *Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models*. Computer Speech and Language Vol. 9, 1995.