THE BUCKET BOX INTERSECTION (BBI) ALGORITHM FOR FAST APPROXIMATIVE EVALUATION OF DIAGONAL MIXTURE GAUSSIANS

J. Fritsch, I. Rogina
fritsch,rogina@ira.uka.de

Interactive Systems Laboratories
University of Karlsruhe — Germany
Carnegie Mellon University — USA

ABSTRACT

Today, most of the state-of-the-art speech recognizers are based on Hidden Markov modeling. Using semi-continuous or continuous density Hidden Markov Models, the computation of emission probabilities requires the evaluation of mixture Gaussian probability density functions. Since it is very expensive to evaluate all the Gaussians of the mixture density codebook, many recognizers only compute the M most significant Gaussians ($M = 1, \ldots, 8$). This paper presents an alternative approach to approximate mixture Gaussians with diagonal covariance matrices, based on a binary feature space partitioning tree. The proposed algorithm is experimentally evaluated in the context of large vocabulary, speaker independent, spontaneous speech recognition using the JANUS-2 speech recognizer. In the case of mixtures with 50 Gaussians, we achieve a speedup of 2-5 in the computation of HMM emission probabilities, without affecting the accuracy of the system.

1. INTRODUCTION

To approximate the log probability of a Gaussian mixture density using only the top-M Gaussians, one has to keep a sorted list of the M most significant Gaussians during the VQ operation. This approach requires the partial computation of squared Mahalanobis distances for all the densities in the codebook. To reduce the computational load, some speech recognizers use Euclidean instead of Mahalanobis distances or restrict the computation of the emission probability to the one Gaussian with the highest value ($M = 1$). In [1] we presented an application of the BVI algorithm [2] that allows for fast nearest-neighbor search when seeking the top-1 Gaussian. However, we found that restricting the probability computation to the evaluation of the Gaussian with the highest value will not be appropriate for most tasks and will result in a lower recognition accuracy. To avoid a performance degradation, one is forced to either evaluate all the Gaussians in the mixture or, at least, search for the Gaussians, that give a significant contribution to the mixture probability. Evaluation of all the Gaussians is mostly omitted, since it slows down the score computation considerably, while most of the Gaussians do not contribute significantly anyway.

In this paper, we present an extension of the BVI algorithm, the Bucket Box Intersection (BBI) algorithm, that allows for the fast approximative evaluation of a mixture Gaussian density, ensuring an approximation error less than a given threshold T. Instead of scanning all the Gaussians in the codebook to find the set of the M most significant Gaussians in the mixture, the BBI algorithm uses a pre-computed binary decision tree to dynamically determine this set. The algorithm is found to be superior in time complexity to both the top-1 and top-all algorithms with a word accuracy that is still higher than that achieved by restricting the mixture to the top-1 Gaussian.

2. GAUSSIAN BOXING

Consider the log of a single multivariate Gaussian pdf with diagonal covariance $\Sigma = \mathbf{I}\sigma, \quad \sigma = (\sigma_1^2, \ldots, \sigma_K^2)$:

$$\log N(\mathbf{x}, \mu, \Sigma) = -\frac{1}{2}[\log((2\pi)^K \prod_{j=1}^{K} \sigma_j^2) + \sum_{j=1}^{K} \frac{(x_j - \mu_j)^2}{\sigma_j^2}]$$

The region in K-dimensional space, where this function gives higher logprobs than a given absolute threshold T is a hyperellipsoid with axes parallel to the coordinate axes. Given a threshold T, we can compute a box with boundary hyperplanes orthogonal to the coordinate axes that completely includes the Gaussian hyperellipsoid. We call this box the Gaussian box associated with T (see Fig. 1).

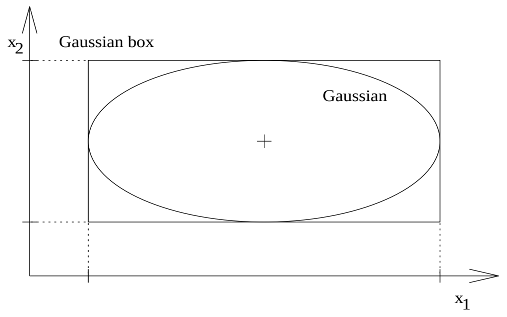

Fig. 1: Gaussian boxing

The projection interval $[a_j, b_j]$ of a Gaussian box to coordinate j is given by

$$[a_j, b_j] = \mu_j \pm \sqrt{-2\sigma_j^2[T + \frac{1}{2}\log((2\pi)^K)\prod_{j=1}^{K}\sigma_j^2]}$$

T must not be chosen greater than the maximum value of the Gaussian, otherwise the argument of the square root gets negative.

3. THE BUCKET BOX INTERSECTION ALGORITHM

Consider the multivariate Gaussian mixture $p(\mathbf{x}|\omega)$ with diagonal covariances for class ω

$$p(x|\omega) = \sum_{i=1}^{N} c_i N(x, \mu_{\omega i}, \sigma_{\omega i})$$

with the vector of mixture coefficients $(c_1, \ldots, c_N)$ satisfying the constraints $c_j \geq 0$ and $\sum_{j=1}^{K} c_j = 1$. By restricting the evaluation of a Gaussian to the region of a Gaussian box with threshold T, we introduce an approximation error smaller than T. Using Gaussian boxes with threshold T for all Gaussians in the mixture and restricting the evaluation of the mixture to the Gaussians with boxes that contain the current argument vector $\mathbf{x}$, we also introduce an overall error of less than T, since the mixture weights add up to one.
To find the Gaussian boxes containing a vector $\mathbf{x}$ we use a K-dimensional space partitioning tree (K-d tree) developed by Bentley [3].
A K-d tree is a generalization of the simple one-dimensional binary tree. At every nonterminal 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 $\mathbf{x}$ can now be located with respect to the dividing hyperplane by a single scalar comparison. Given a K-d tree of depth d and a K-dimensional vector $\mathbf{x}$, a sequence of d scalar comparisons leads to the leaf (called bucket) containing the vector $\mathbf{x}$. Such a K-d tree partitions the K-dimensional space into 2^d disjoint rectangular regions (buckets) and allows fast identification of the bucket containing a given vector.
Having localized a given vector $\mathbf{x}$ in one of the K-d tree's buckets, we can restrict the Gaussian mixture computation to the evaluation of the Gaussians whose boxes are intersecting with the current bucket. Depending on the position and the form of the Gaussians, the threshold T and the tree depth d, the bucket-box-intersection (BBI) list associated with each bucket will contain fewer Gaussians to evaluate than the complete mixture, reducing the computational load considerably. Fig. 2 illustrates the K-d tree partitioning and the BBI lists in the 2-dimensional space.

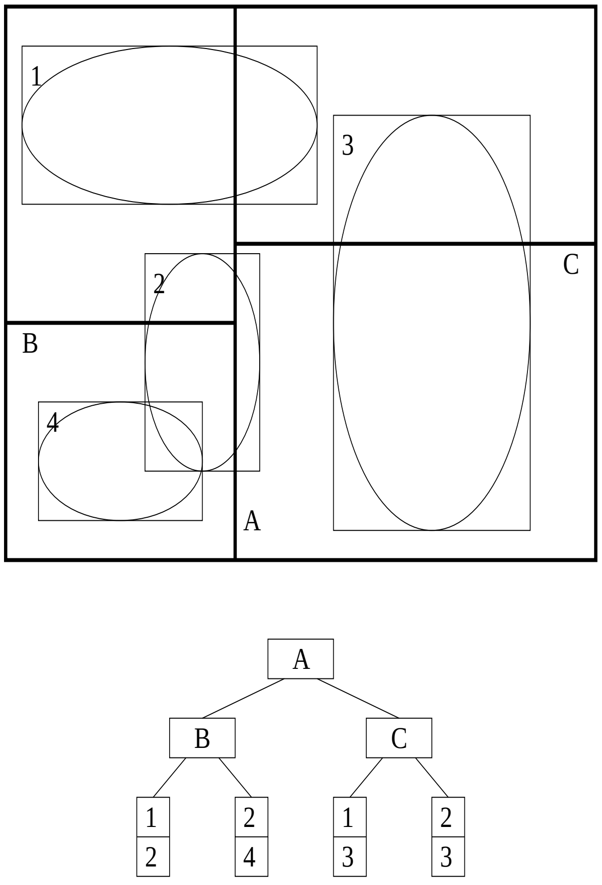

Fig. 2: Bucket Box Intersection (BBI)

4. OPTIMIZING THE SEARCH TREE

We are interested in a K-d search tree that minimizes the expected number of bucket-box intersections given the Gaussian boxes for an error threshold T. This minimization process corresponds to a global optimization problem involving all the parameters in all nonterminal nodes of the tree which is extremely complex and unfeasible in practice. Therefore, it becomes necessary to locally optimize the parameters of the nonterminal nodes independently, starting at the root node. For this purpose, we propose a modification of the Generalized Optimization Criterion (GOC). See [2] for a detailed description of this criterion. The non-terminal nodes in the tree are processed top-down, starting at the root node. The following steps are taken to compute a locally optimized division hyperplane within a specific node of the tree:

1) Determine the Gaussian boxes that are intersecting with the hyperrectangular region corresponding to the current node. This region is implicitly represented by the set of division hyperplanes of the nodes above the current node, up to the root node.
2) For all coordinate axes x_i: label the boundaries of the projected Gaussian boxes with L(ower) and U(pper)

and sort them along the axis. Hypothesize a division hyperplane $x_i = h$ that has an equal number of left sided L labels and right sided R labels. By doing so, we focus on building up a balanced binary tree, that has an equal number of bucket box intersections in its leaf nodes. Next, compute the number C_i of intersecting Gaussian boxes that are split by the hyperplane.

3) Choose the hyperplane with minimum number of splits C_i as the current nodes division hyperplane.

Finally, having reached the terminal nodes (leafs), a BBI list has to be created for every leaf, containing references to the intersecting Gaussians. By increasing the error threshold T, search trees with a lower number of expected bucket-box intersections may be created, since the Gaussian boxes will be smaller. This will reduce the computational load of evaluating the Gaussian mixture at the cost of higher approximation error rates that directly lead to a degradation of the recognizers word accuracy. There is a tradeoff between fast and accurate score computation that has to be considered in adjusting the two parameters of the BBI algorithm, namely the threshold T used in Gaussian boxing and the depth of the search tree.
To reduce the average approximation error, we also used the following error-correcting postprocessing or tuning step after the optimization of the search tree was finished. A large number of representative training examples (feature vectors), taken from the recognizers training database, is used to verify the accuracy of the approximated scores, as computed by the BBI search tree. The training vectors are classified into one of the tree leafs, using the first part of the BBI algorithm (top-down tree search). Each of the leafs is then processed independently as follows, using the set of training vectors that were classified to the specific leaf:
For every Gaussian in the codebook, we compute an average contribution to the mixture by computing the correct (top-all) scores of the leaf's training vectors. After sorting the Gaussians by decreasing average contributions we can verify, if there is a particular Gaussian with high average contribution that is missing in the leaf's bucket box intersection list. If so, we can replace a low contributional Gaussian by the missing one, which will decrease the average approximation error while leaving the speed up unchanged.

5. EXPERIMENTS

We have conducted experiments with the JANUS-2 speech recognizer [4] on the German Spontaneous Scheduling Task , using the proposed algorithm for the approximation of mixture Gaussian emission probabilities. In our experiments, we were using a continuous-density HMM system with 3300 mixtures and more than 1300 equally sized codebooks, containing 50 vectors with 16 LDA-transformed melscale coefficients. In computing the multi-dimensional Gaussian boxes, we obtained better results by using a relative threshold R, instead of the proposed absolute threshold T (see Fig. 3).

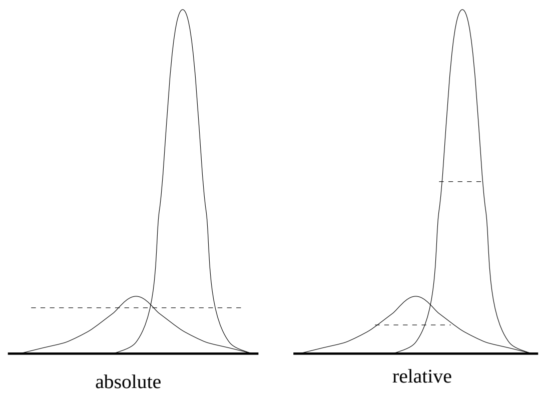

Fig. 3: Absolute vs. Relative Gaussian boxing

The projection intervals $[a_j, b_j]$ of the resulting relative Gaussian box are given by

$$[a_j, b_j] = \mu_j \pm \sqrt{-2\sigma_j^2 \log(R)}, \quad 0 \leq R \leq 1$$

Relative thresholds are motivated by the observation that the maxima of the Gaussians in our codebooks differ by more than 7 orders of magnitude. Using an absolute threshold to determine the boxes, we put emphasis on the absolute approximation error, resulting in inaccurate modelling of the low-probability regions of the mixture. By using relative thresholds, we do not have to compute a BBI search tree for all the mixture density distributions, but only for each of the 1300 codebooks, because, in contrast to absolute Gaussian boxes, relative boxes are independent of the mixture coefficient. The time for the computation of the search trees and their storage requirements can be neglected, compared to the running time and storage requirements of the whole system (see Fig. 4).

depth	R	time (min)	storage (MBytes)
8	0.3	42	9.3
8	0.5	27	6.1
8	0.7	15	3.8
6	0.3	11	2.6
6	0.5	7	1.8
6	0.7	4	1.2
4	0.3	3	0.6
4	0.5	2	0.4
4	0.7	1	0.3

Fig. 4: K-d trees computation time and storage requirements

Fig 5. shows the speedup of systems that use the proposed algorithm with different relative thresholds R and different tree depths. The speedup is computed by dividing the time required for the evaluation of all the Gaussians in the mixture by the average time required for the evaluation of the Gaussians in the leafs of the search tree.

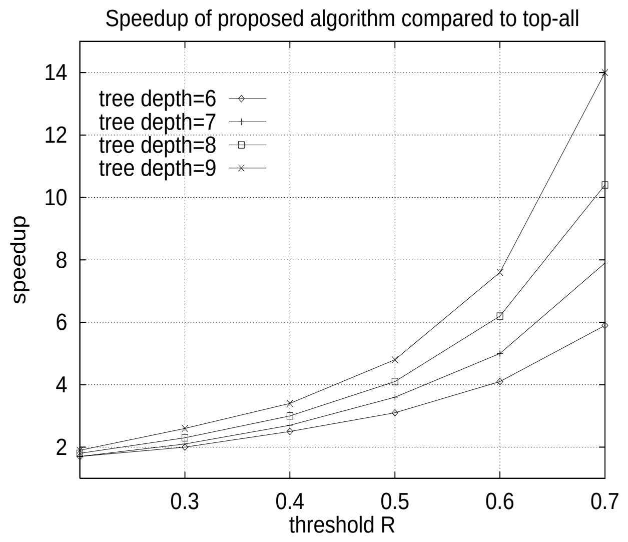

Fig. 5: Speedup of proposed algorithm

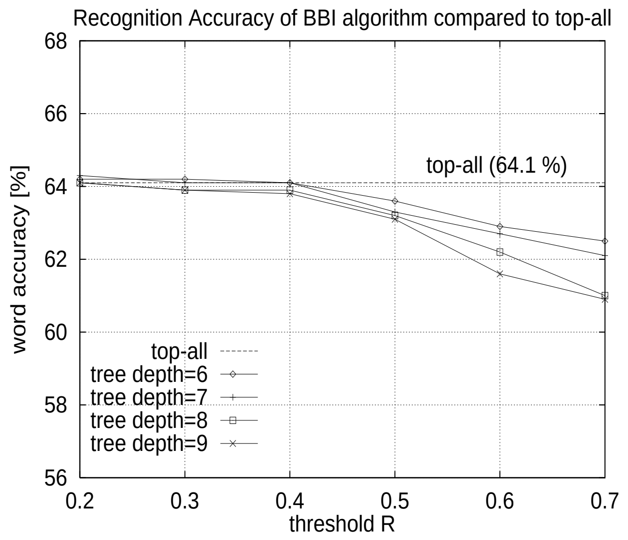

Fig. 6: Word Accuracy on German Spontaneous Scheduling Task

Fig 6. shows the word accuracy for several test runs with different relative thresholds R and different tree depths. The dotted line shows the accuracy of the top-all system that is evaluating all the Gaussians in the mixture. Up to a threshold of $R = 0.5$, the word accuracy is barely affected by the use of the proposed algorithm, sometimes being even higher than that of the top-all system.

6. CONCLUSIONS

In this paper, we have addressed the issue of fast approximative evaluation of mixture Gaussian pdf's with diagonal covariance matrices using the BBI algorithm. The proposed algorithm is studied in the context of speaker-independent, spontaneous speech recognition using semi-continuous or continuous density HMM's with mixture Gaussian density codebooks of size $N = 50$. It outperforms the top-M approaches to the fast approximation of mixture Gaussians, offering even higher speedups than reported here when using larger mixtures.

7. ACKNOWLEDGEMENTS

This research was partly funded by grant 413-4001-01IV1-1S3 from the German Ministry of Science and Technology (BMBF) as a part of the VERBMOBIL project. The authors wish to thank all members of the Interactive System Laboratories, especially A. Waibel for useful discussions and active support.

REFERENCES

[1] Fritsch, J.; Rogina, I.; Sloboda, T.; Waibel, A.: *Speeding up the Score Computation of HMM Speech Recognizers with the Bucket Voronoi Intersection Algorithm*, To appear in Proceedings of the EUROSPEECH, Madrid, Spain 1995.

[2] Ramasubramanian, V.; Paliwal, K. K.: *Fast K-dimensional Tree Algorithms for Nearest Neighbor Search with Application to Vector Quantization Encoding*, IEEE Transactions on Signal Processing, Vol. 40, No. 3, March 1992.

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

[4] M. Woszczyna, N. Aoki-Waibel, F.D. Buø, N. Coccaro, K. Horiguchi, T. Kemp, A. Lavie, A. McNair, T. Polzin, I. Rogina, C.P. Rose, T. Schultz, B. Suhm, M. Tomita, A. Waibel: *JANUS 93: Towards Spontaneous Speech Translation*, Proceedings of the ICASSP 1994, Adelaide, volume 1, pp 345-348.

[5] M.Woszczyna, N.Coccaro, A.Eisele, A.Lavie, A.McNair, T.Polzin, I.Rogina, C.P.Rose, T.Sloboda, M.Tomita, J.Tsutsumi, N.Aoki-Waibel, A.Waibel, W.Ward: *Recent Advances in Janus, a Speech to Speech Translation System*, Proceedings of the EUROSPEECH, Berlin, 1993.

THE BUCKET BOX INTERSECTION (BBI) ALGORITHM FOR FAST APPROXIMATIVE EVALUATION OF DIAGONAL MIXTURE GAUSSIANS

J. Fritsch, I. Rogina
fritsch,rogina@ira.uka.de
Interactive Systems Laboratories
University of Karlsruhe — Germany
Carnegie Mellon University — USA

Today, most of the state-of-the-art speech recognizers are based on Hidden Markov modeling. Using semi-continuous or continuous density Hidden Markov Models, the computation of emission probabilities requires the evaluation of mixture Gaussian probability density functions. Since it is very expensive to evaluate all the Gaussians of the mixture density codebook, many recognizers only compute the M most significant Gaussians ($M = 1, \ldots, 8$). This paper presents an alternative approach to approximate mixture Gaussians with diagonal covariance matrices, based on a binary feature space partitioning tree. The proposed algorithm is experimentally evaluated in the context of large vocabulary, speaker independent, spontaneous speech recognition using the JANUS-2 speech recognizer. In the case of mixtures with 50 Gaussians, we achieve a speedup of 2-5 in the computation of HMM emission probabilities, without affecting the accuracy of the system.