SPEEDING UP THE SCORE COMPUTATION OF HMM SPEECH REGOGNIZERS WITH THE BUCKET VORONOI INTERSECTION ALGORITHM

J. Fritsch, I. Rogina, T. Sloboda, A. Waibel
{*fritsch,rogina,sloboda,waibel*}*@ira.uka.de*

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

ABSTRACT

With increasing sizes of speech databases, speech recognizers with huge parameter spaces have become trainable. However, the time and memory requirements for high accuracy realtime speaker-independent continuous speech recognition will probably not be met by the available hardware for a reasonable price for the next few years. This paper describes the application of the Bucket Voronoi Intersection algorithm to the JANUS-2 speech recognizer, which reduces the time for the computation of HMM emission probabilities with large Gaussian mixtures by 50% to 80%.

1. INTRODUCTION

Although the computation of Gaussians is only a part (for very large vocabularies, even a small part) of the overall run time, speeding it up does reduce the reaction time of the recognizer, and especially the time for training significantly. When computing the log probability of a Gaussian mixture, many speech recognizer do not use all Gaussians but only the top n. We have found that in our system using only the one Gaussian with the highest probability is almost as good as using the sum of more Gaussians. We have also found that using the Euclidean distance instead of the Mahalanobis distance for finding that most probable Gaussian does not decrease recognition accuracy too much. This reduces the computation of an HMM emission probability to a two part process: First, find the centroid that has the smallest Euclidean distance to the current speech sample, and second, compute the value of the Gaussian (multiplied with its mixture weight) for that centroid. So instead of computing n Gaussians, where n is the size of the mixture, we only have to compute one Gaussian plus we have to run an algorithm for finding the closest centroid. For this we use the Bucket Voronoi Intersection (BVI) algorithm [1]. It was introduced for high speed vector quantization of low-dimensional vectors. However, we have found that it is still good enough for 16-dimensional speech vectors. In this paper we describe experiments in which we have investigated the effect of the BVI-algorithm on the run-time behavior and the recognition accuracy of the JANUS-2 speech recognizer [2, 3].

2. THE BUCKET VORONOI INTERSECTION ALGORITHM

For a detailed discourse on the Bucket Voronoi Intersection (BVI) algorithm see [1].

All points in the feature space having the same nearest-neighbor codebook vector define a Voronoi region. The set of all Voronoi regions constitutes a disjoint partitioning of the feature space. The aim of the algorithm is to approximate this partitioning with a top-down tree search.

The principle behind it is a binary tree. Each node of the tree represents a hyperplane in the feature space. When classifying a sample vector, the tree is descended from the root down to a leaf. At every node, a decision is made to descend into the left or the right successor node, depending on the sample vector being on the left or on the right side of the current hyperplane. So every step down the tree reduces the size of the search space.

When the tree descending algorithm has finally reached a leaf node, there will be only a few codebook vectors left whose Voronoi region is intersecting with the remaining search space, which is called a bucket. The set of all buckets constitutes a disjoint partitioning of the feature space. Depending on how deep we descend the tree, we get different buckets and a different partitioning of the feature space. In higher levels of the tree we get larger buckets, which contain more Voronoi regions.

Fig.1 illustrates the Bucket Voronoi Intersection algorithm in the case of a 2-dimensional feature space. The space is partitioned into disjoint regions by a search tree of depth $d = 2$ and the Voronoi partitioning of the codevectors. The intersection of the two partitions define the BVI lists that have to be stored at the leafs of the search tree.

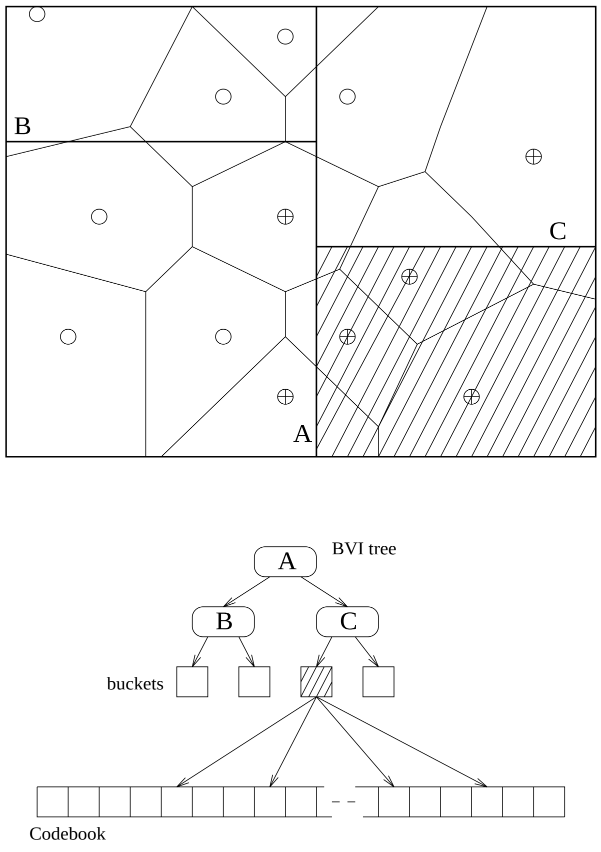

Fig. 1: Bucket Voronoi Intersection search

Given the tree with its 3 separating hyperplanes A, B and C, 2 scalar comparisons are sufficient to determine the bucket in which a test vector x is located. A linear search among the codevectors whose Voronoi regions intersect with the located bucket will give the correct nearest neighbor. Considering a test vector located in the striped bucket, the search complexity reduces from 13 (full search) to the 6 cross marked codevectors. (see Fig.1).

Although the bucket sizes decrease monotonically with increasing tree depth, there is no guarantee to reach the optimal case of having only one codebook vector per bucket. For that reason, there is a tradeoff between speed up and memory requirements of the tree. The time for traversing the tree is not the critical factor. Let d be the depth of the tree, b the average bucket size and n the codebook size, then we will have to compute d hyperplane comparisons, plus b Gaussians instead of n Gaussians. Since the BVI-algorithm only uses hyperplanes of the form $x_i = t$, deciding on which side of the hyperplane a vector $\mathbf{y}$ is located takes only one simple floating point comparison $y_i < t$. A full binary tree of depth d has 2^d leafs (buckets), so the memory requirements for storing the tree grow exponentially. Since we are usually using feature spaces that are at least 16-dimensional, the limit for the depth of the trees will be determined rather by the amount of available memory, than by the run-time requirements.

3. COMPUTING THE BVI-TREE

Since it is extremely expensive to compute the real boundaries of a high dimensional Voronoi region, we approximate the Voronoi region with a cuboid whose edges are parallel to the coordiates. These approximate regions generally overlap each other. The boundaries of the cuboids are determined by encoding a sufficiently large set of training vectors. (see Fig. 2).

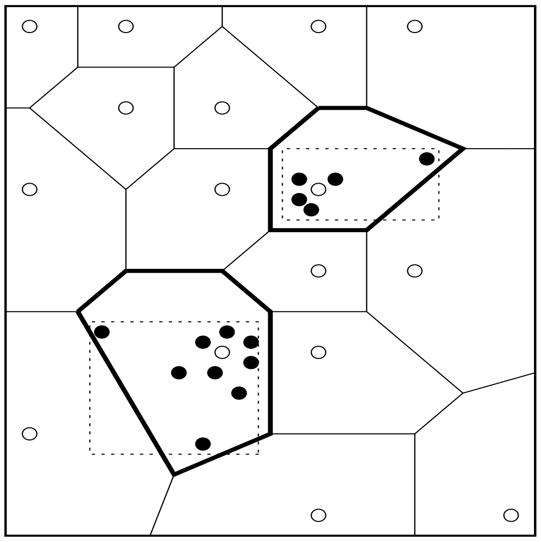

Fig. 2: approximated Voronoi regions

A cuboid-approximated Voronoi region is defined entirely on one side of a hyperplane if all the training vectors that fall into the region are on the same side of the hyperplane. With this approach we get a very simple decision rule

based on scalar comparisons, but we introduce a possible classification error (Fig. 2).
The error rate can be reduced by increasing the number of training vectors. The more vectors we use for training the more it is likely that the approximate cuboid of a Voronoi region will contain the entire region.

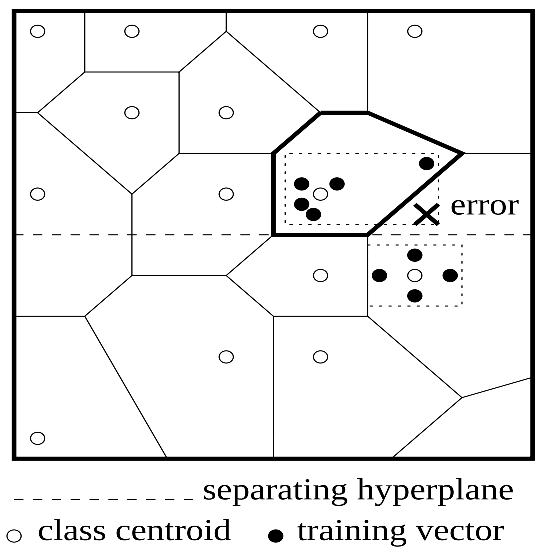

Fig. 3: classification error in approximated Voronoi regions

Fig. 4 shows the average classification error rate, depending on the number of training vectors and the depth of the BVI-trees.
In our experiments, we have found that a low classification error rate for the nearest neighbor is not important for a good speech recognition accuracy (see Fig. 5)
The objective of a good BVI-tree is to have as few Voronoi regions in every bucket as possible. The average size of a bucket decreases with the depth of the tree, while the memory requirements and error rate grow exponentially, limiting the tree size. We have conducted experiments with trees up to a depth of 12. Fig 6 shows the average bucket size depending on the depth of the BVI-tree.

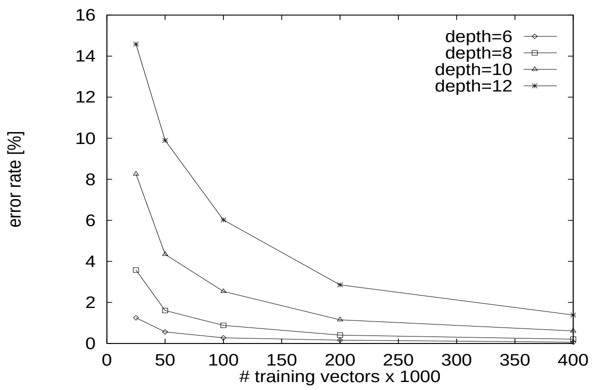

Fig. 4: average classification error rate

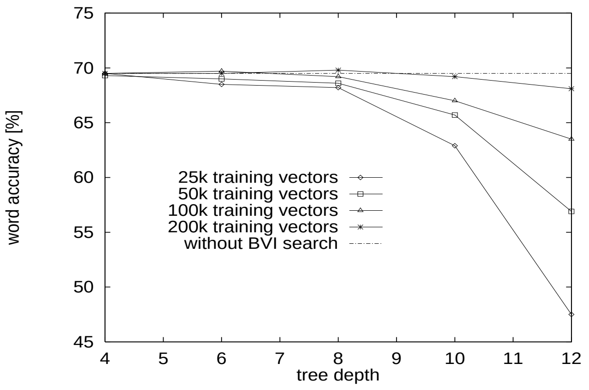

Fig. 5: recognition accuracy using BVI-search

4. RUNTIME BEHAVIOR

The speedup in the HMM-emission probability computation can be approximated by the average mixture size divided by the average bucket size. Of course, the relative speedup for the entire system is smaller. Fig. 7 shows the speedup of the score computation mechanism for training and testing sessions with the JANUS-2 speech recognizer. We were using equally sized codebooks containing 50 vectors with 16 coefficients.

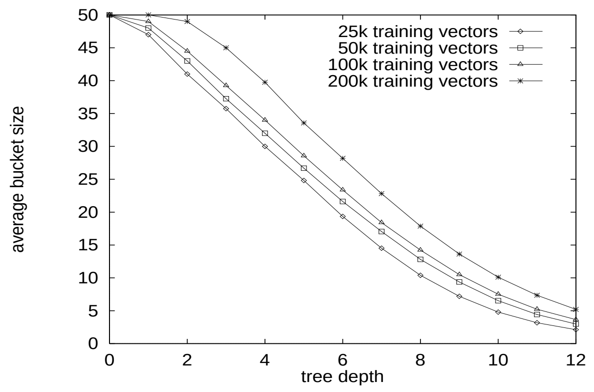

Fig. 6: average bucket size depending on tree depth

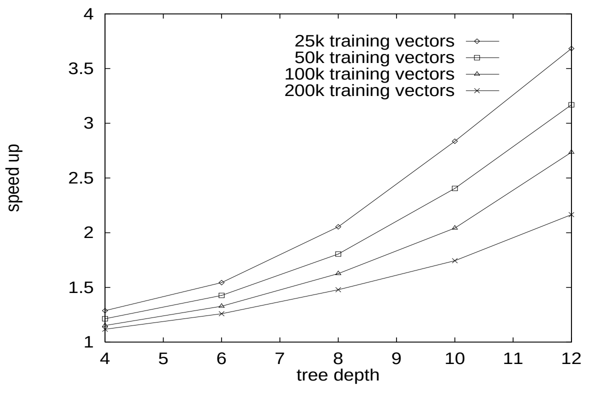

Fig. 7: speedup of BVI-score computation

5. RECOGNITION ACCURACY

We have found that the recognition accuracy of the speech recognizer does not suffer from the possible classification errors of the BVI-algorithm if the training vector set used for determining the Voronoi projections is large enough ($>$ 100000 vectors). Fig. 5 shows the word accuracy on the German Spontaneous Scheduling Task (GSST) [3, 2] for different amounts of training data for the BVI-algorithm.

6. LARGER CODEBOOKS

Experiments by [1] with different sized codebooks showed, that the BVI algorithm's performance improves with increasing codebook size and/or decreasing dimensionality. Until now, we were almost exclusively working with codebooks of size $N = 50$, consisting of 16-dimensional melscale vectors. To improve the speedup, we have generated new codebooks of size $N = 1024$ (same dimension) for the JANUS-2 speech recognizer and tested the BVI algorithm again. Fig. 8 shows the average bucket size of the new BVI trees, depending on the tree depth.

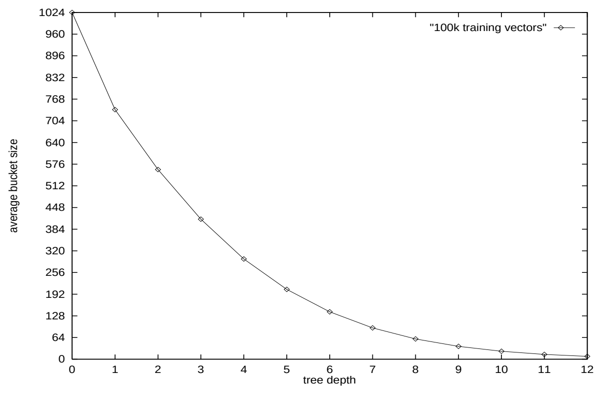

Fig. 8: average bucket size for search trees generated from large codebooks

Compared with the results for smaller codebooks (see Fig. 6), the new BVI search trees for the large codebooks offer a tremendously higher speedup. This suggests, that one should use codebooks as large as reasonable for the specific application, to obtain the highest speedup from the BVI algorithm.

7. CONCLUSIONS

We presented first results of our ongoing research on speeding up the computation of HMM emission probabilities with the BVI-algorithm. Although the algorithm was developed for rather low dimensional data compression applications, we succesfully integrated this fast vector quantization method into a HMM speech recognizer.

ACKNOWLEDGMENTS

This research was partly funded by grant 413-4001-01IV101S3 from the German Ministry of Science and Technologie (BMBF) as a part of the VERBMOBIL project. The views and conclusions contained in this document are those of the authors.

REFERENCES

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

[2] 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.

[3] 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.

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

[5] Cheng D. Y., Gersho A., Ramamurthi B. and Shoham Y.: *Fast search algorithms for vector quantization and pattern matching*, Proceedings of the IEEE ICASSP 1984, vol 1. Mar. 1984, pp. 9.11.1-9.11.4.