FAST LIKELIHOOD COMPUTATION METHODS FOR CONTINUOUS MIXTURE DENSITIES IN LARGE VOCABULARY SPEECH RECOGNITION

Stefan Ortmanns, Hermann Ney and Thorsten Firzlaff

Lehrstuhl für Informatik VI, RWTH Aachen – University of Technology,
D-52056 Aachen, Germany

ABSTRACT

This paper studies algorithms for reducing the computational effort of the mixture density calculations in HMM-based speech recognition systems. These likelihood calculations take about $70 - 85\%$ of the total recognition time in the RWTH system for large vocabulary continuous speech recognition. To reduce the computational cost of the likelihood calculations, we investigate several space partitioning methods. A detailed comparison of these techniques is given on the North American Business Corpus (NAB'94) for a 20 000-word task. As a result, the so-called projection search algorithm in combination with the VQ method reduces the cost of likelihood computation by a factor of about 8 with no significant loss in the word recognition accuracy.

1. INTRODUCTION

A computationally expensive operation in speech recognition systems is the computation of the mixture densities of the hidden Markov models (HMMs). Typically, for large vocabulary continuous speech recognition tasks with a very large number of mixture densities, the computation of the likelihoods (or strictly speaking log-likelihoods) needs more than 75% of the overall recognition effort. Therefore, we investigate in this paper techniques for reducing the computational cost of the mixture density (or state likelihood) calculations. The fast log-likelihood computation techniques are integrated in the time-synchronous beam search algorithm where the search algorithm is based on a tree-organized pronunciation lexicon in connection with a bigram language model. For efficiency reasons, look-ahead pruning techniques are used during the search process as described in [11].

In this paper, we present two efficient likelihood computation techniques which are based on space partitioning techniques. In particular, we describe a fast likelihood computation algorithm using the k-dimensional binary search tree [1, 6, 7]. In addition, we present a fast log-likelihood calculation technique which is similar to the nearest neighbor search method as described in [9]. Unlike the k-dimensional binary search tree method, this method is based on dynamic partitioning of the search space. The basic idea of the so-called projection search technique is to find all prototype vectors within a hypercube centered at a given acoustic observation vector. A further reduction of the computational effort can be achieved by integrating the projection search technique into other fast log-likelihood computation methods. To this purpose, we combine the projection search technique with two well-known fast log-likelihood computation methods, namely the Hamming distance approximation (HDA) [2] and a vector quantization (VQ) method for mixture density preselection [4, 8].

The organization of this paper is as follows. In Section 2, we briefly describe the task of log-likelihood calculations using Laplacian mixture densities. In Section 3, we review the fast log-likelihood computation technique which is based on a k-dimensional binary search tree. Further, we present a *dynamic* space partitioning technique for fast log-likelihood calculation. In Section 4, we give experimental results on the North American Business Corpus (Nov.'94) for a 20 000-word task.

2. LAPLACIAN MIXTURE DENSITIES

The emission probability of an HMM state s can be expressed as the weighted sum of prototype densities:

$$p(x|s) \quad = \quad \sum_{l=1}^{L(s)} p(l|s) \cdot p(x|s,l) \ ,$$

where the term $p(l|s)$ denotes the mixture weight of the lth mixture density component. When using the maximum approximation in connection with Laplacian mixture densities [10], the negative log-likelihood for a given observation vector $x \in \mathbb{R}^D$ is given by:

$$-\log p(x|s) \quad = \quad \min_l \left\{ -\log p(l|s) + \sum_{d=1}^{D} \frac{|x_d - \mu_{lsd}|}{\sigma_{lsd}} + \sum_{d=1}^{D} \log(2\,\sigma_{lsd}) \right\} .$$

where μ_{lsd} is the dth component of the prototype vector of the component density l of HMM state s. σ_{lsd} denotes the deviation of the dth component of density l. The task is now to find the density l with a minimal negative log-likelihood with respect to the acoustic observation vector x for each HMM state (or mixture). In a straightforward implementation, this so-called nearest neighbor search requires the calculation of $L(s)$ weighted distances per state. For instance, in our recognition experiments (20 000-word NAB-task) we have used about 290 000 Laplacian mixture densities per

gender with a single pooled vector of absolute deviations. In this case, the likelihood computation takes about 85% of the total recognition time. In the following section, we present two fast likelihood computation methods which are based on space partitioning techniques.

3. LIKELIHOOD COMPUTATION ALGORITHMS

3.1. k-d Tree Method

In this section, we present a static space partitioning technique which is based on a k-dimensional binary search tree, called k-d tree. The k-d tree is a data structure which partitions space using hyperplanes where the hyperplanes are perpendicular to the coordinate axes [1]. To organize the $L(s)$ densities of a HMM state s as a k-d tree (with $k = D$), we construct the tree in a similar way as proposed in [6]: A density (or strictly speaking the prototype vector of a density) is chosen to be the root node. Densities located on one side of a hyperplane passing through the root node are added to the left child and the densities on the other side are added to the right child. This process is applied recursively on the left and right child nodes until all densities are assigned to a tree node. Thus, the complete tree partitions the space into hyper-rectangular regions. However, the described tree generation process generally leads to an unbalanced tree. To generate a balanced tree, each tree node is assigned to the median of densities, i.e. the median of prototype vector components passing this specific node [6]. Fig. 1 shows an example of the space partitioning using a k-d tree ($k = 2$). To determine the density with the shortest distance to an observed vector, the search in the k-d tree is organized as follows: The k coordinates of the observed vector are used to find the hyper-rectangular regions in which the vector is located. All prototype vectors within these regions are collected. Thus, the likelihood computation is only performed for these vectors.

3.2. Projection Search Algorithm

Unlike the k-d tree approach, the projection search algorithm (PSA) is based on *dynamic* space partitioning. The idea of the algorithm is to find all prototype vectors located inside a hypercube centered at a given acoustic observation vector. The prototype vectors within the hypercube can be determined as follows: First, we find the prototype vectors that are between a pair of parallel hyperplanes. These vectors are then added to a list of 'candidates'. Note, the planes are orthogonal to the first coordinate axis and are located on either side of the observation vector at a distance ϵ. Next, we reduce the list of candidates by removing vectors that are not located between a second pair of parallel planes being orthogonal to the first planes and so on. These slicing process results in a D-dimensional cube containing all prototype vectors that are closest to the observed vector within a distance of ϵ. For $D = 3$, the slicing process is illustrated in Fig. 2 [9]. To determine points within the hypercube, we start with a list containing all prototypes that are sandwiched between the two parallel planes H_{X_1} and H_{X_2}. The planes are located on either side of the observation vector x at a distance of ϵ. Then, we reduce

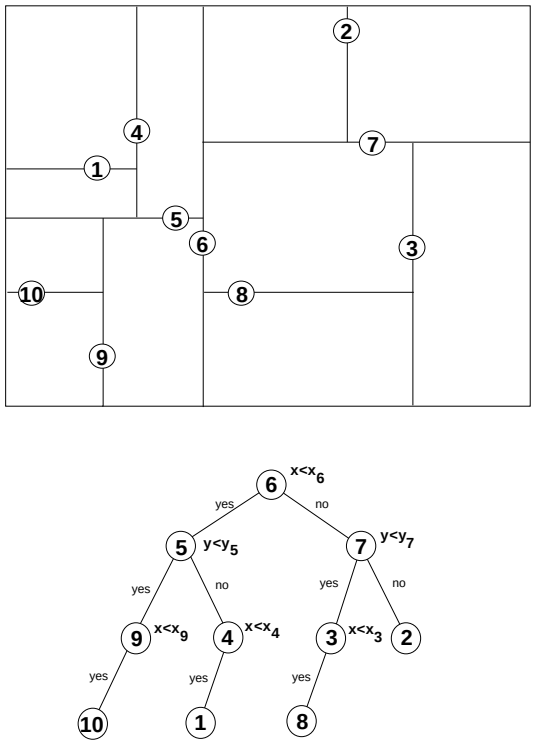

Figure 1. Space partitioning using a k-d tree (with $k = 2$); a) space tesselation, b) associated k-d tree.

the list by eliminating prototypes that are *not* between the planes H_{Y_1} and H_{Y_2} which are perpendicular to H_{X_1} and H_{X_2}. After repeating this slicing step for the planes H_{Z_1} and H_{Z_2}, the list includes all prototypes within the hypercube of edge size 2ϵ. Finally, all prototype vectors within the hypercube are then evaluated in the likelihood computation routine. It should be mentioned that the computation of the list of candidates can be done in an efficient way [9]. In a preprocessing step, each coordinate of the prototype densities is sorted so that binary search can be performed coordinate-wise to find the prototypes between a pair of parallel planes. For practical aspects, we consider only the first, say 7 vector components instead of all components of the prototype vectors after LDA transformation. Moreover, only the first component of all prototype vectors is stored in an ordered set. A further speedup can be achieved by state flooring [8]. If no prototype (or density) from a certain state belongs to the hypercube, an approximative log-likelihood score will be assigned to this specific state. Note, state flooring will be applied in all methods presented in this paper.

3.3. Hybrid Techniques

Due to the simplicity of the projection search algorithm and for further speeding up the log-likelihood calculations, we have combined the projection search with the following well-known fast log-likelihood techniques:

- preselection (VQ) method [4],
- Hamming distance approximation (HDA) [2].

The idea of these so-called hybrid fast log-likelihood techniques can be viewed as a *two*-step selection process.

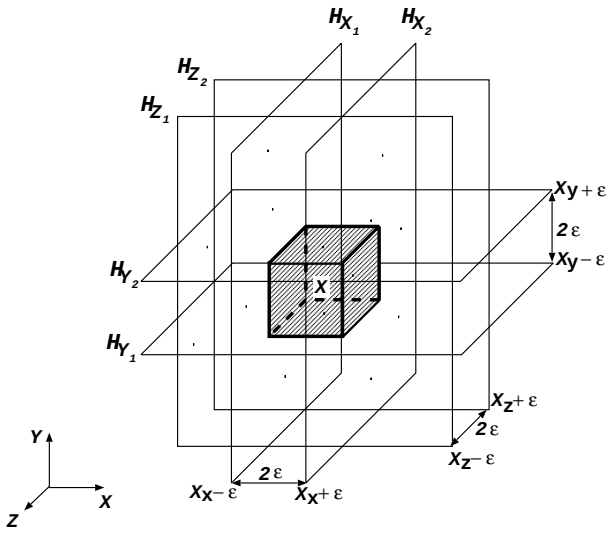

Figure 2. Illustration of the slicing process using the projection search algorithm.

In a first step for example, we use the VQ method to get a coarse preselection of prototypes. Then, in a second step, the list of candidates can be further confined by applying the projection search algorithm on the preselected prototype vectors.

Preselection VQ Method

To reduce the memory requirement depending on the size of the VQ cells (or codebooks), the preselection VQ method proposed in [4] has been slightly modified. Instead of selecting the cell with the closest distance to the observation vector for the log-likelihood calculation, we determine now the first, say 3 or 5 closest cells to the observation vector. After merging these n closest cells, the exhaustive log-likelihood calculation can be performed. Thus, this strategy allows the construction of smaller VQ cells.

HDA Method

The combination of the projection search with the HDA method [2] works in a similar way as described before. When considering the l_1 norm, the distance $d(x,y)$ between vectors x and y can be defined as

$$\begin{aligned} d(x,y) &= \sum_{d=1}^{D} |x_d - y_d| \\ &= \sum_{d=1}^{D} \Bigg[|x_d| + |y_d| \\ &\qquad - \left\{ \begin{array}{ll} 2\min(|x_d|,|y_d|) & , \quad x_d \cdot y_d > 0 \\ 0 & , \quad \text{otherwise} \end{array} \right\} \Bigg] \\ &= \|x\|_1 + \|y\|_1 - 2 \cdot \sum_{x_d \cdot y_d > 0} \min(|x_d|,|y_d|)\,. \end{aligned}$$

The assumption is now that the correction term $2 \cdot \sum_{x_d \cdot y_d > 0} \min(|x_d|,|y_d|)$ is approximated by

$$\frac{2}{D} \min(\|x\|_1, \|y\|_1) \sum_{x_d \cdot y_d > 0} 1\,.$$

So, we have the approximation $d(x,y) \approx d(x,y)_{HDA}$ with

$$d(x,y)_{HDA} = \|x\|_1 + \|y\|_1 - \frac{2}{D} \min(\|x\|_1, \|y\|_1) \sum_{x_d \cdot y_d > 0} 1\,.$$

Note that this estimation can be efficiently derived by computing the Hamming distance of the two vectors x and y [2].

4. EXPERIMENTAL ANALYSIS

4.1. Test Conditions

The experimental tests were carried out on the ARPA North American Business (NAB'94) H1 development corpus comprising 310 sentences with a total of 7387 words spoken by 10 male and 10 female speakers. We used a 20 000-word vocabulary and a bigram language model with a perplexity (PP) of 198.4 [12]. 199 of the spoken words were out-of-vocabulary words. The training of the emission probability distributions of the underlying hidden Markov models was performed on the WSJ0 and WSJ1 training data as described in [5]. We used about 290 000 Laplacian mixture densities with a single pooled vector of absolute deviations per gender. The prototype vectors consist of 42 LDA-transformed filter bank coefficients. The experiments were performed on a SGI workstation with a R5000 processor (3.4 SpecInt95). In all experiments, we have used the word conditioned tree search method combined with a bigram language model look-ahead pruning technique [11].

4.2. Results

Table 1 summarizes the recognition results. The Table shows the effort of log-likelihood computation in terms of the effective number of computed densities per mixture (N_{eff}) and the required CPU time [%] for various log-likelihood computation methods. In addition, the search space (average number of active states, arcs and trees per time frame), the recognition errors and the real time factor are also given. In an initial experiment, we performed a test without fast likelihood computation. On average, about 90 densities per mixture were computed, which leads to a real time factor of 33 for the beam search. Next, a series of experiments was run to study the effect of the fast log-likelihood calculation techniques on the search effort. The results are shown in Table 1. It can be seen that the HDA method reduces the effort of the log-likelihood calculations by approximately 40%. The result for the HDA is not as good as reported in [2] because some implementational tricks have not been considered in this work [3]. Then, we have tested the k-d tree method. The k-d tree method works slightly better than the HDA method. Considering the results of the projection search algorithm, the overall recognition time was more than halved as compared to the baseline experiment. Finally, we have tested the VQ method. In an informal experiment, the size of the VQ cells and the number of cells evaluating in the log-likelihood procedure are adjusted beforehand. In the reported results, we have used three cells for the evaluation in the log-likelihood procedure. The size of each cell can be expressed by the overlapping factor which was 10.4. In total 512 VQ cells were used. For this conditions,

Table 1. Effect of various fast log-likelihood calculation methods on the overall recognition effort and recognition results for a 20 000-word task using a bigram language model (NAB'94 H1 development corpus: 20 speakers, 310 sentences, 7 387 spoken words; SGI workstation with a R5000 processor (3.4 SpecInt95)).

Method	Likelihood calculation		Search space			Recognition errors [%]		Real-time
	N_{eff}	CPU-time [%]	states	arcs	trees	del / ins	WER	factor
baseline	88	100.0	3312	936	13	2.5 / 2.6	16.5	32.7
Hamming distance approx. (HDA)	11	59.5	3175	904	13	2.5 / 2.5	16.4	22.2
k-d tree	30	50,7	3047	875	13	2.4 / 2.7	16.5	19.4
projection search (PSA)	10	24.6	2903	840	13	2.3 / 2.7	16.5	11.8
preselection method (VQ)	11	17.6	3132	894	13	2.4 / 2.6	16.5	10.0
PSA & HDA	4	39.4	2642	781	12	2.4 / 2.7	16.5	16.2
PSA & VQ	7	12.0	2703	798	12	2.3 / 2.7	16.6	7.5

the VQ method leads to a speedup factor of 5.7 of the CPU-time required for the log-likelihood evaluation. A further reduction has been achieved by combining the VQ method with the projection search algorithm. The time for the computation of the likelihood can be reduced by a factor of 8.3 with virtually no loss in recognition accuracy.

CONCLUSIONS

In this paper, we have investigated algorithms for fast log-likelihood calculation. The results are summarized as follows:

- We compared four different fast log-likelihood calculation techniques, namely the HDA method, k-d tree method, VQ method and the projection search method on the NAB'94 H1 development corpus. We found that projection search reduced the likelihood calculation effort by a factor of 4.1. The VQ method achieved a reduction of 5.7 resulting in an overall real time factor of 10.0 on a SGI Indy (R5000).
- Further, we have combined the VQ method with the projection search method. As a result, this hybrid method reduced the effort of likelihood computation by a factor of about 8. All in all, the total recognition time can be reduced by a factor of $4-5$ without affecting the word recognition error rate.

Acknowledgement. This research was partly funded by grant 01 IV 701 T4 from the German Ministry of Science and Technology (BMBF) as a part of the VERBMOBIL project. The views and conclusions contained in this document are those of the authors.

REFERENCES

[1] J.L. Bentley: Multidimensional Binary Search Tree used for Associative Searching. Communications of the ACM, 18(9), pp. 509-517, September 1975.

[2] P. Beyerlein, M. Ullrich: Hamming Distance Approximation for a Fast Log-Likelihood Computation for Mixture Densities. Proc. Europ. Conf. on Speech Communication and Technology, Madrid, Spain, pp. 1083-1086, September 1995.

[3] P. Beyerlein, M. Ullrich: Personal Communication. Philips Research Laboratories, May 1997, Aachen, Germany.

[4] E. Bocchieri: Vector Quantization for the Efficient Computation of Continuous Density Likelihoods. Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, Minneapolis, MN, Vol.II, pp. 692-695, April 1993.

[5] C. Dugast, R. Kneser, X. Aubert, S. Ortmanns, K. Beulen, H. Ney: Continuous Speech Recognition Tests and Results for the NAB'94 Corpus. Proc. ARPA Spoken Language Technology Workshop, Austin, TX, pp. 156-161, January 1995.

[6] J.H. Friedman, J.L. Bentley, R.A. Finkel: An Algorithm for Finding Best Matches in Logarithmic Expected Time. ACM Transactions on Mathematical Software, Vol. 3, No. 3, p. 209-226, September 1977.

[7] J. Fritsch, I. Rogina: The Bucket Box Intersection (BBI) Algorithm for Fast Approximative Evaluation of Diagonal Mixture Gaussians. Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, Atlanta, GA, pp. 837-840, May 1996.

[8] K.M. Knill, M.J.F. Gales, S. Young: Use of Gaussian Selection in Large Vocabulary Continuous Speech Recognition using HMMs. Proc. Int. Conf. on Spoken Language Processing, Philadelphia, PA, pp. 470-473, October 1996.

[9] S.A. Nene, S.K. Nayar: Closest Point Search in High Dimensions. Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR 96), San Francisco, CA, pp. 859-865, June 1996.

[10] H. Ney: Acoustic Modelling of Phoneme Units for Continuous Speech Recognition. Fifth European Signal Processing Conference, Barcelona, Spain, September 1990, pp. 65–72, in L. Torres, E. Masgrau, M.A. Lagunas (eds.): 'Signal Processing V: Theories and Applications', Elsevier Science Publishers B. V., 1990.

[11] S. Ortmanns, A. Eiden, H. Ney, N. Coenen: Look-Ahead Techniques for Fast Beam Search. Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, Munich, Vol. 3, pp. 1783-1786, April 1997.

[12] F. Wessel, S. Ortmanns, H. Ney: Implementation of Word Based Statistical Language Models. Proc. SQEL Workshop on Multi-Lingual Information Retrieval Dialogs, Pilsen, Czech Republic, pp. 55-59, April 1997.