Fast Likelihood Computation Techniques in Nearest-Neighbor Based Search for Continuous Speech Recognition

Bryan L. Pellom, Ruhi Sarikaya, and John H. L. Hansen

Abstract—This paper describes two effective algorithms that reduce the computational complexity of state likelihood computation in mixture-based Gaussian speech recognition systems. We consider a baseline recognition system that uses nearest-neighbor search and partial distance elimination (PDE) to compute state likelihoods. The first algorithm exploits the high dependence exhibited among subsequent feature vectors to predict the best scoring mixture for each state. The method, termed best mixture prediction (BMP), leads to further speed improvement in the PDE technique. The second technique, termed feature component reordering (FCR), takes advantage of the variable contribution levels made to the final distortion score for each dimension of the feature and mean space vectors. The combination of two techniques with PDE reduces the computational time for likelihood computation by 29.8% over baseline likelihood computation. The algorithms are shown to yield the same accuracy level without further memory requirements for the November 1992 ARPA Wall Street Journal (WSJ) task.

I. INTRODUCTION

In recent years, the majority of LVCSR systems developed have been based on continuous density hidden Markov models (CDHMMs). However, these systems typically operate slower than real-time, which renders them impractical for some applications (e.g., dialogue systems, dictation). One of the computationally most expensive steps in speech recognition systems based on CDHMMs is the state likelihood computation. Typically, these computations take up a major proportion (30–70%) of the overall recognition time. This is due to the multiple number of Gaussian mixtures used to model each state (4–64). In addition, the Gaussians associated with each active state must be evaluated for each frame of data. There are a number of techniques which attempt to reduce the computational requirements of likelihood calculation, while maintaining comparable recognition performance. One set of techniques reduce the dimension in the feature vector using linear discriminant analysis (LDA) [8]. Another class of techniques rely on Gaussian clustering for fast likelihood calculation [3], [4]. An elegant technique has been proposed by Seide [1] where the likelihood computation is formulated as a nearest-neighbor tree search. Some of the space partitioning techniques using a k-dimensional binary search tree have been used to improve likelihood computations [5], [7]. However, these methods can lead to degradation in recognition accuracy and an increase in memory requirements.

II. NEAREST-NEIGHBOR APPROXIMATION

The likelihood of an HMM state \( \lambda_s \) for a given input feature vector \( \mathbf{x}_t \) can be expressed as a weighted sum of likelihoods from individual Gaussian densities

\[
p(x_t | \lambda_s) = \sum_{m=1}^{M_s} w_m p(x_t | \mathcal{N}(\mu_m, \Sigma_m))
\]

where \( M_s \) and \( w_m \) represent the number of multivariate Gaussians, mixture weight, mean, covariance matrix, and feature vector dimension for the \( m \)th density in state \( \lambda_s \).

In practice, modeling states with diagonal covariances is preferred over full covariance matrices for computational and data sparsity reasons. Therefore, (1) becomes

\[
p(x_t | \lambda_s) = \sum_{m=1}^{M_s} \frac{w_m}{(2\pi)^{J/2} |\Sigma_m|^{1/2}} \exp\left(-\frac{1}{2} (x_t - \mu_m)^T \Sigma_m^{-1} (x_t - \mu_m)\right)
\]

where \( M_s, w_m, \mu_m, \Sigma_m \) and \( J \) represent the number of multivariate Gaussians, mixture weight, mean, covariance matrix, and feature vector dimension for the \( m \)th density in state \( \lambda_s \). In practice, modeling states with diagonal covariances is preferred over full covariance matrices for computational and data sparsity reasons. Therefore, (1) becomes

\[
p(x_t | \lambda_s) = \sum_{m=1}^{M_s} \frac{w_m}{(2\pi)^{J/2} \prod_{j=1}^{J} \sigma_m^2(j)} \exp\left(-\frac{1}{2} \sum_{j=1}^{J} \frac{(x_t(j) - \mu_m(j))^2}{\sigma_m^2(j)}\right)
\]

where \( C_m \) is a constant for each density. Since all terms that comprise \( C_m \) are known prior to recognition, it can be precomputed.

Computation of \( p(x_t | \lambda_s) \) is expensive due to the \( J \) multiplications, \( J \) divisions, and \( M_s \) exponential operations required in (2). One common technique for reducing the computational overhead is to use a nearest-neighbor [1] approximation of the likelihood. Instead of computing the likelihood by summing across all mixtures, the nearest mixture probability is taken as the state probability. In the log-domain, the nearest-neighbor approximation is given by

\[
\log(p(x_t | \lambda_s)) \approx \max_{1 \leq m \leq M} \left\{ \log(C_m) - \frac{1}{2} \sum_{j=1}^{J} \frac{(x_t(j) - \mu_m(j))^2}{\sigma_m^2(j)} \right\}
\]

Manuscript received July 5, 2000. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. A. S. Spanias.

The authors are with the Center for Speech and Language Research, University of Colorado, Boulder, CO 80309 USA (e-mail: pellom@csle.colorado.edu). Publisher Item Identifier S 1070-9908(01)05236-1.
III. PARTIAL DISTANCE ELIMINATION

Given this framework, we can think of the nearest-neighbor search problem as a vector quantizer (VQ) codebook search problem, where the densities in that state are the codewords and the distortion measure is given on the right side of (3). Equation (3) involves joint maximization of the mixture constant and minimization of the weighted Euclidean distance given by the second term, over the densities belonging to that state. If we denote \( \log(p(x_t | \lambda_s)) \) as \( D(x_t | g(C_m, \mu_{m^{th}}, \sigma_m)) = D(x_t | y_m) \), then we can rewrite (3) as

\[
D(x_t | y_m) = C_m - \sum_{j=1}^{J}(x_t(j) - \mu_m(j))^2 \cdot \left( \frac{1}{2\sigma_m^2(j)} \right)
\]

where \( C_m \) is log(\( C_m \)). Note that in general, division is computationally more expensive than multiplication. Therefore, we can load the inverse variance at run time and use multiplication instead of division. Contrary to the usual VQ search problems which minimize some type of distortion measure to find the best codeword, we maximize the distortion even though we perform minimization with the second term on the right side of the equation. Now, PDE can be used. Note that the weighted (with variance) squared error is a separable measure, and \( D(x_t | y_m) \) can be evaluated component-wise. Therefore, we can decide if a codeword will have minimal distortion without always computing over the range of \( J \) feature vector elements. Before finishing the computation of a complete distortion for any \( j < J \), if the accumulated distortion for the first \( j \) components of the input vector is smaller than the highest distortion \( D_{\text{max}} \) yet found in the search, that codeword is eliminated. This early test will reduce the more expensive multiplications at the expense of comparisons for each component in the feature vector. As more mixtures are computed, a better estimate of \( D_{\text{max}} \) will be obtained, and more mixtures will be eliminated without exhausting all the components in the feature vector. Note that the first mixture must be computed over all \( J \) components to get an initial estimate of \( D_{\text{max}} \).

IV. PROPOSED ALGORITHMS

Algorithms proposed in this paper are formulated to more efficiently calculate the probability given in (3). One unique aspect of the following techniques is that they require almost no memory and modifications in the underlying recognition system. Furthermore, they introduce no compromise between degraded recognition accuracy versus computational savings. The speed improvement resulting from these techniques comes at no additional cost. Finally, rather than being an alternative to previously proposed techniques [7], [5], these methods complement past approaches for further overall speed improvement.

A. Best Mixture Prediction

Obviously, the efficiency of the PDE technique heavily depends on how quickly a high estimate of \( D_{\text{max}} \) is obtained. This can easily be accomplished by exploiting the high dependence between consecutive feature vectors. Let \( m^{t-1} \) be the best match for the previous frame for state \( s \)

\[
m^{t-1} = \arg \max_{1 \leq i \leq M_s} D(x_{t-1} | y_m).
\]

Because of overlapping frames during feature extraction, \( x_t \) and \( x_{t-1} \) are usually similar. Therefore, we expect \( D(x_{t-1} | y_{m^{t-1}}) \) to be close to \( D(x_t | y_{m^{t}}) \). Therefore, predicting the previous best match, Gaussian as the current best match and computing its distortion first results in a high \( D_{\text{max}} \) immediately speeds up the elimination process for the best match search.

In order to test the validity of this assumption, we used the entire 410 sentence test set from the WSJ0 database and counted how many times densities are repetitively selected for each state as the best match across time. The average number of likelihood calculations for each sentence is 0.67 million. The average frequency of picking the current best mixture in the current frame as the best mixture in the next frame is 0.48 million. Therefore 71.6% of the time, Gaussian densities selected at time \( t-1 \) are selected as the best Gaussians at time \( t \).

B. Feature Component Reordering (FCR)

The second algorithm, feature component reordering (FCR) complements PDE and BMP techniques. By analyzing the components of the feature vectors and the densities, we observed that the contribution of some of the components are heavier than others. The idea in this technique is to reorder the components in such a way that components resulting in higher distortion are computed first followed by the components contributing less. Let \( j \rightarrow c[k] \) be a mapping of the location of component \( j \) in the vectors into a new location, \( c[k] \). Therefore, the distortion as represented by (4) becomes

\[
D(x_t | y_m) = C_m - \sum_{j=1}^{J}(x_t(c[j]) - \mu_m(c[j]))^2 \cdot \left( \frac{1}{2\sigma_m^2(c[j])} \right).
\]

The mapping \( c[k] \) can be learned from a portion of the development test set offline and stays fixed during recognition.

Our feature vector is composed of 12 mel scale frequency cepstrum coefficients (MFCCs) with their first-order and second-order differences. These features are augmented with energy and its first and second order differences resulting in a 39 dimensional feature vector. In Fig. 1, the solid plot shows the distortion contributed by individual elements of the feature vectors averaged over 30 test sentences during recognition. Note that after the first nine to ten sentences, the order of the vector elements is stabilized. The dashed curve shows the reordered distortion. It is interesting to note that energy, first and second cepstrums, and delta energy parameters ranked at the top of the reordered sequence indicating that they contribute to likelihood score computation more than others. This process speeds up PDE by eliminating mixture densities whose output probability is smaller than the current best.
V. DESCRIPTION OF RECOGNITION SYSTEM

The recognition task for the experiments used the WSJ0-Dev portion of the 1992 ARPA WSJ 5k vocabulary continuous speech recognition corpus. A cross-word gender-dependent system is used to build a model set using the complete SI-284 WSJ training set. From the signal processing side of the system, speech data, sampled 16 kHz are converted to 39 dimensional observation vectors. Acoustic modeling in our recognizer is based on gender dependent HMMs for base phones and triphones. All HMMs have three-state left-to-right topology with between six and 16 Gaussian mixtures per state. The lexicon for the recognizer uses the linear sequence of phones to represent the pronunciation for each word in the vocabulary. Since triphone acoustic models are used these base phone sequences are converted into triphone sequences by taking each base phone together with its left and right context base phones. The language model we used is a trigram language model. We use a single pass beam search Viterbi decoding.

VI. EVALUATIONS

Experimental results are summarized in Table I. There, we refer to “baseline” as the straightforward computation of the likelihood for each mixture in a state by using nearest-neighbor search without PDE. We assume this likelihood computation time to be 100% as the reference for the subsequent techniques. Note that the recognition accuracy is not listed either for the baseline as well as other techniques since it is maintained at its original level. The word error rate (WER) for the baseline as well as the proposed techniques is 11.8%. Incorporating PDE into the likelihood computation step lowered the computational time by 4.0%. The major improvement resulted from the FCR technique, which provides another 22.0% savings in computation. Finally, including BMP added 3.8%. Using all three techniques together reduces 29.8% of the time spent for likelihood computation. Note that FCR and BMP require the likelihood to be computed via a PDE technique, otherwise, they cannot be incorporated into the baseline scheme. At the bottom of Table I, we summarize the respective number of average multiplications and comparisons incurred for each technique. The new PDE technique trades off two multiplications for two comparisons. Note that as the number of computations decrease to compute the likelihood, the timing functions to measure the CPU clock begins to have more of an effect on computation time. For this

![Image of a graph showing cumulative distortion](image-url)

Fig. 1. The solid curve is the average distortion before reordering, and the dashed curve is the distortion of the reordered elements in descending order: [37, 1, 2, 38, 13, 25, 4, 36, 14, 3, 28, 16, 27, 8, 6, 5, 15, 29, 7, 11, 17, 12, 30, 20, 32, 10, 18, 9, 31, 39, 35, 19, 33, 34, 23, 21, 22, 36, 24].

<table>
<thead>
<tr>
<th>Likelikhood Computation Time (%)</th>
<th>Baseline</th>
<th>PDE</th>
<th>PDE + FCR</th>
<th>PDE + FCR + BMP</th>
</tr>
</thead>
<tbody>
<tr>
<td>100.0</td>
<td>96.0</td>
<td>74.0</td>
<td>70.2</td>
<td></td>
</tr>
<tr>
<td>Average # of multiplications for likelihood computation per sentence (million)</td>
<td>687.75</td>
<td>405.52</td>
<td>324.36</td>
<td>293.09</td>
</tr>
<tr>
<td>Average # of comparisons (1 float + 1 integer) for likelihood computation per sentence (million)</td>
<td>0.0</td>
<td>405.52</td>
<td>324.36</td>
<td>293.09</td>
</tr>
</tbody>
</table>
reason the average savings in multiplications and comparisons should have led to a greater decrease in likelihood computation time for BMP.

VII. CONCLUSION

In this study, two new techniques in a PDE framework are proposed to reduce computational time of the likelihood computation in nearest-neighbor based search for LVCSR. The algorithms are called BMP and FCR. The combination of these techniques with PDE reduces the computational complexity for likelihood computation by 29.8% over straightforward likelihood computation. The computational savings gained by these techniques come at no additional cost while complementing past approaches for fast likelihood computation.

REFERENCES


