Feature Pruning for Fast Likelihood Evaluation of Automatic Speech Recognition

Xiao Li and Jeff Bilmes

***Abstract*— This work presents *feature pruning*, a simple yet effective technique to reduce the likelihood computation in ASR systems that use continuous density HMMs. Our technique, under certain conditions, only evaluates the likelihoods of a fraction of feature elements, and approximates those of the remaining (pruned) ones by prediction. The order in which feature elements are evaluated is obtained by a data-driven approach to minimize computation. With this order, feature pruning can speed up the likelihood evaluation by a factor of 1.33 and reduce its power consumption by 27% for an isolated word recognition task. For a continuous speech recognition system using either monophone or triphone models, the speedup and power reduction of the likelihood evaluation are 1.50 and 35% respectively.**

***Index Terms*— Likelihood evaluation, pruning, speedup, power reduction, speech recognition**

I. Introduction

The proliferation of mobile devices has brought about a great need for a more convenient user interface. Voice control is an ideal method since it enables hand-free operations and eliminates conventional bulky keypads. However, most contemporary automatic speech recognition (ASR) systems use continuous hidden Markov models (CHMM). They can be computationally expensive and energy-hungry, posing potential problems for real-time processing and battery life.

In an ASR system using CHMMs, the state likelihood evaluation can dominate the operational load by taking up to 96% of the total computation for a typical small vocabulary application [1], and 30% to 70% for LVCSR tasks [2]. Since the observation distribution is typically represented by a Gaussian mixture, the computational complexity of the likelihood evaluation is proportional to the total number of Gaussians in the system and the dimensionality of these Gaussians. This suggests two potential research directions to improve efficiency: reducing the number of Gaussians and reducing their dimensionality.

The number of Gaussian evaluations can be reduced in a system that employs tying since, once evaluated, the Gaussian score can be stored and utilized multiple times. Tying has been extensively applied to different levels of the hierarchical structure of HMMs [3]–[8]. Alternatively, Gaussians can be pruned on the fly using Gaussian selection [1], [2], or nearest-neighbor approximation [9], where only the most relevant Gaussians are precisely computed. Taking the other direction, various feature selection methods [10]–[12] tackle the dimensionality problem by selecting only the feature elements with the highest discriminative power. Additionally, feature elements can also be selectively computed online. [13] presented an incremental feature pruning applied to discrete-mixture models, and [14] proposed a partial distance elimination technique associated with the nearest-neighbor based search.

Our work herein shares the basic approach of [13] and [14]. However, our technique is quite novel in that it is targeted for CHMM-based systems, and applies feature pruning to all Gaussian mixtures instead of only the nearest neighbors. Furthermore, subspace Gaussian likelihoods are approximated by prediction, and the ordering of feature element evaluation is driven by computation minimization.

This work was funded by NSF grant ITR/RC-0086032.

The authors are with the Electrical Engineering Department, University of Washington, Seattle

II. Feature Pruning

In continuous HMMs, the likelihood $b_j(O)$ of an observation vector O, given state j, is typically parameterized by a Gaussian mixture,

$$b_j(O) = \sum_{i \in M_j} w_i \mathcal{N}(O; \mu_i, \Sigma_i) \tag{1}$$

where M_j is the subset of Gaussians belonging to state j.

In a Gaussian mixture HMM with diagonal covariance matrices, it is implicitly assumed that feature subsets are conditionally independent given the state and the mixture component identity. This property allows the Gaussian over the full-space to be decomposed into K lower-dimensional Gaussians over orthogonal subspaces each with dimension d_k, $k = 1..K$ and $\sum_{k=1}^{K} d_k = D$. The i^{th} Gaussian component likelihood in Equation (1) then becomes

$$\mathcal{N}(O; \mu_i, \Sigma_i) = \prod_{k=1}^{K} \mathcal{N}(O_k; \mu_{i,k}, \Sigma_{i,k}) \tag{2}$$

where O_k, $\mu_{i,k}$ and $\Sigma_{i,k}$ are respectively the projection of the observation, mean and variance vector onto the k^{th} subspace.

Notice that if any one of these subspace Gaussian factors in Equation (2) is very small, the full-space Gaussian likelihood score will likely also be small. Specifically, let $m_{i,k}$ denote the maximum value of the k^{th} subspace Gaussian, and define $m_i = \prod_k m_k$, which is the maximum value of the full-space Gaussian. If $\mathcal{N}(O_1; \mu_{i,1}, \Sigma_{i,1}) << m_{i,1}$, then $\mathcal{N}(O; \mu_i, \Sigma_i) << m_i$ and hence the i^{th} component will have little contribution to $b_j(O)$. Therefore, rather than always computing the score of the full Gaussian component, we compute it only in the case when $\mathcal{N}(O_1; \mu_{i,1}, \Sigma_{i,1})$ is large enough. When this subspace Gaussian is small, since we then know the entire i^{th} component will have a low score, we simply approximate it using a computationally inexpensive function. Moreover, our technique is applicable at multiple

levels, where the approximation "kicks in" when the current partially computed component score falls below a threshold.

We let the indices $1..K$ indicate the order in which the subspace likelihoods are computed. The feature pruning algorithm is described as follows,

Input: observation O; $\{\mu_i, \Sigma_i\}$ of a Gaussian
Output: approximated $\ln\mathcal{N}(O;\mu_i,\Sigma_i)$

$a \leftarrow \ln\mathcal{N}(O_1;\mu_{i,1},\Sigma_{i,1})$;
for $k = 2..K$ **do**
  if $a > t_{k-1}$ **then**
    compute $\ln\mathcal{N}(O_k;\mu_{i,k},\Sigma_{i,k})$;
    $a \leftarrow a + \ln\mathcal{N}(O_k;\mu_{i,k},\Sigma_{i,k})$;
  else
    $a \leftarrow f_{k-1}(a)$;
    break;
  end if
end for
return a;

where t_k, $k = 1..K-1$, are a set of thresholds, and $f_k(\cdot)$ are a set of affine approximation functions whose coefficients can be determined using Least Mean Square (LMS) estimation from the training data. Consequently, only a fraction of observation vectors are precisely computed to full dimensionality. Some of the feature elements are ignored according to the likelihoods of already computed elements.

III. Clustering of Feature Elements

For recognizers which use both static and dynamic features, a natural way to cluster feature elements is to allocate static features to the first group, their deltas to the second and the double deltas the last, if available. The advantage is that the feature vectors are very likely to be represented in this order, which simplifies the incorporation of feature pruning into an existing ASR system. Furthermore, we empirically observe a certain degree of correlation between the likelihoods of static features and their dynamic counterparts. The predictability of the likelihoods is beneficial to the feature pruning technique.

This naturally-grouped scheme, however, is not guaranteed to give the best reduction in computation. Ideally the feature elements should be clustered in such a way that the likelihood computation can be reduced as much as possible, while the recognition accuracy is maintained. The complexity of trying every combination to find the best clustering scheme is factorial in the dimensionality of the feature vectors. Therefore, we propose a data-driven method to cluster the feature elements according to their statistical properties. This work only studies the case where feature subsets are of the same size.

First, we have found through empirical experimentation that the recognition accuracy depends heavily on the thresholds no matter how the feature elements are clustered. This makes sense since these thresholds determine how often the approximation takes place. Second, the clustering does have an impact on the recognition accuracy, but at a smaller scale, since it more or less affects the predictability of the likelihoods and hence the effectiveness of the approximation functions. Finally, the clustering has a great impact on the reduction in computation and power; different clustering schemes would have different numbers of feature subsets pruned under the same thresholding value. Therefore, we hope that a clustering will be found such that the pruning will remove as many feature elements as possible before the critical point (c_k for stage k) where recognition accuracy begins to drop from the baseline.

The first subset of feature elements to be computed should have the maximum probability of having a score below c_1, thereby increasing the chance that the remaining subspace likelihoods will be pruned. Specifically, we search for a set of feature elements that attempts to maximize $Pr\{\ln\mathcal{N}(O_1;\mu_{i,1},\Sigma_{i,1}) < c_1\}$, which can be estimated nonparametrically using a large amount of data. This implies an intractable optimal clustering problem, so we instead employ the following greedy algorithm:

1) Repeat several (5-10) times: randomly choose $d = D/K$ feature elements to be computed in the first group, and then find the critical threshold c_1 at which recognition accuracy begins to drop significantly using the pruning algorithm. Compute the average over these critical thresholds which we denote as $\mathrm{E}[c_1]$. (Note that the optimal choice of K is beyond the scope of this work, but see [13].)
2) Extract a subset of representative training data and compute the log likelihoods for each feature dimension over all Gaussian components and all data frames. For each of the D feature elements, count the number of instances for which the feature element log likelihood for a particular component fell below $\mathrm{E}[c_1]/d$, an estimate of the average per-feature-element threshold. Select the d feature elements with the highest counts to the first subset.
3) Repeat 1) and 2) excluding the already-selected feature elements to get c_2, ..., c_k and their corresponding feature element subsets.

IV. Experiments and Results

We ran two sets of experiments, one on an isolated word corpus and the other on a continuous speech recognition task. The speedup and power reduction of the likelihood evaluation were impressive for both tasks.[1]

A. Isolated word recognition on PhoneBook

The isolated word experiment was done on NYNEX Phonebook [15], a telephone-speech database. We used 42 continuous HMMs with 165 states altogether. Each state was represented by a Gaussian mixture comprised of 12 Gaussian components. MFCCs, log energy and their deltas were extracted, leading to 26 feature elements. The test was carried out on 8 test sets each with a different 75-word vocabulary. The

[1]In this work, we define

$$\%\ \text{speedup} = \frac{\text{baseline CPU time}}{\text{new CPU time}} \times 100\%$$

$$\%\ \text{power reduction} = (1 - \frac{\text{new CPU power}}{\text{baseline CPU power}}) \times 100\%$$

final WER was an average over them all. The baseline WER was 2.07%, and the likelihood evaluation was responsible for 70% of the total computation without beam pruning and over 80% when beam search was applied.

We compared the speedup of feature pruning using our data-driven approach with that of the naturally-grouped one and a randomly-grouped one. As shown in Figure 1, the data-driven approach outperforms the other two by achieving a speedup of 1.33, with only a 0.2% absolute increase in WER.

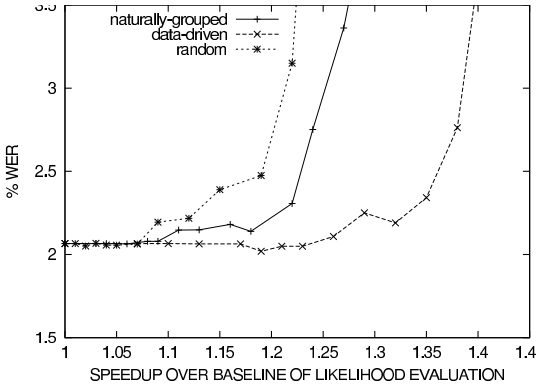

Fig. 1. Feature pruning on PhoneBook

B. Continuous speech word recognition on TIMIT

Our continuous speech word recognition system was based on TIMIT [16]. Our pruning technique was applied to both monophone and triphone systems. The monophone system used a simple bi-gram language model and 42 monophone HMMs with a total of 126 states as the acoustic model. Each state distribution was represented by a Gaussian mixture with 12 components. The feature elements were MFCCs, log energy and their delta and double deltas, leading to 39 dimensions. The baseline computed each observation vector to its full dimension for all 1512 Gaussian components. As for the triphone system, states were tied into 1356 distinct ones using a decision tree, and the system had a total of 8135 Gaussian components. For both systems, one-pass decoding was performed on the complete TIMIT test set consisting of 1344 utterances. The baseline word accuracy was 85.68% for the monophone system, and 88.20% for the triphone system.

We divided the feature elements into three equally sized groups. As shown in Figure 2, for the monophone system, the data-driven approach works remarkably well by having a speedup over 1.50 with a 0.2% absolute decrease in accuracy. Neither the naturally-grouped one nor the random one can compete with it in terms of speedup. Similar results are observed in Figure 3 for the triphone system.

C. Power simulation

To estimate the power consumed by the likelihood evaluation, we utilized Wattch, a framework for architecture-level power dissipation analysis [17]. We compiled the baseline system and the systems integrated with different amount of feature pruning, all targeted for PISA architecture [18].

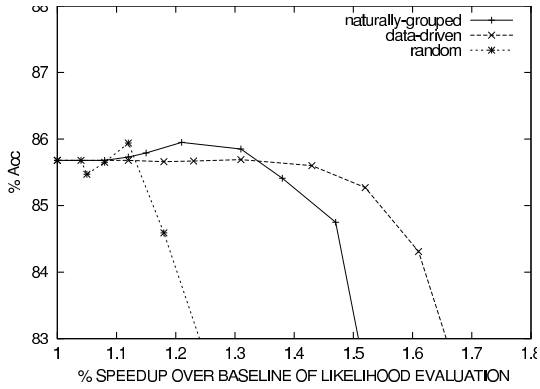

Fig. 2. Feature pruning on TIMIT monophone system

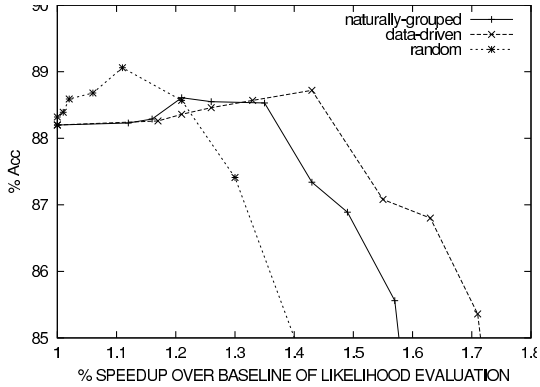

Fig. 3. Feature pruning on TIMIT triphone system

Power simulation was then launched and cycle-level resource expenses were obtained, though only the power dissipation of the likelihood evaluation was extracted in our work. Table I shows the percentage of power reduction in the likelihood evaluation for different clustering schemes on different tasks, where feature pruning with data-driven clustering results in a power reduction of 27% to 35% with again a 0.2% absolute increase/decrease in WER/accuracy. The overall reduction again depends on the ratio of likelihood evaluation to search and the other factors such as the vocabulary size.

TABLE I
POWER SAVINGS IN LIKELIHOOD EVALUATION

Clustering	PhoneBook	TIMIT mono.	TIMIT tri.
random	16%	12%	21%
sequential	21%	30%	29%
data-driven	27%	35%	34%

V. CONCLUSION AND DISCUSSION

In comparison to conventional efficient decoding techniques, our approach focuses on reducing likelihood computation and power consumption by partially computing the likelihood of a Gaussian component. It explores a data-driven approach to cluster feature elements to maximally reduce computation. Our technique achieves a speedup of 1.33 to 1.50 and a power reduction of 27% to 35% in the likelihood evaluation of various recognition tasks. In addition, our

technique does not require a complicated training procedure and is complementary to other fast search techniques, such as Gaussian selection and beam search. It is notable that there are many variations on our technique; the number of feature subsets can be optimized, and the thresholds can be customized for different phonemes or HMM states.

ACKNOWLEDGMENT

The authors thank Jonathan Malkin for reading early drafts of this work.

REFERENCES

[1] E. Bocchieri, "Vector quantization for the efficient computation of continuous density likelihoods," in *Proc. ICASSP*, vol. 2, 1993, pp. 692–695.

[2] M.J.F.Gales, K.M.Knill, and S.J.Young, "State based Gaussian selection in large vocabulary continuous speech recognition using HMMs," *IEEE Trans. on Speech and Audio Processing*, vol. 7, 1999.

[3] K.F.Lee, S.Hayamizu, H.W.Hon, C.Huang, J.Swartz, and R.Weide, "Allophone clustering for continuous speech recognition," in *Proc. ICASSP*, vol. 2, 1990, pp. 749–752.

[4] S.J.Young and P.C.Woodland, "The use of state tying in continuous speech recognition," in *Proc. Eurospeech*, 1993, pp. 2203–2206.

[5] X. Huang and M. Jack, "Semi-continuous hidden Markov models in isolated word recognition," in *9th International Conference on Pattern Recognition*, vol. 1, November 1988, pp. 406–408.

[6] J.R.Bellegarda and D.Nahamoo, "Tied mixture continuous parameter modeling for speech recognition," *IEEE Trans. on Acoustics, Speech and Signal Processing*, vol. 38, pp. 2033–2045, 1990.

[7] S.Takahashi and S.Sagayama, "Four-level tied-structure for efficient representation of acoustic modeling," in *Proc. ICASSP*, vol. 1, 1995, pp. 520–523.

[8] E.Bocchieri and B.K.Mak, "Subspace distribution clustering hidden Markov model," *IEEE Trans. on Speech and Audio Processing*, vol. 9, no. 3, pp. 264–275, 2001.

[9] F.Seide, "Fast likelihood computation for continuous-mixture densities using a tree-based nearest neighbor search," in *Proc. Eurospeech*, 1995, pp. 1079–1082.

[10] E. Lleida and C. Nadeu, "Principal and discriminant component analysis for feature selection in isolated word recognition," in *Signal Processing V: Theories and Applications*. Elsevier Sc. Publ. B.V., 1990, pp. 1251–1254.

[11] E. Bocchieri and J. Wilpon, "Discriminative feature selection for speech recognition," *Computer, Speech and Language*, pp. 229–246, 1993.

[12] J. Nouza, "Feature selection methods for hidden Markov model-based speech recognition," in *Proc. Intl. Conf. on Pattern Recognition*, vol. 2, 1996, pp. 186–190.

[13] V.Digalakis, S.Tsakalidis, C.Harizakis, and L.Neumeyer, "Efficient speech recognition using subvector quantization and discrete-mixture HMMs," *Computer Speech and Language*, vol. 14, pp. 33–46, 2000.

[14] B.Pellom, R.Sarikaya, and J.H.L.Hansen, "Fast likelihood computation techniques in nearest-neighbor based search for continuous speech recognition," *IEEE Signal Processing Letters*, vol. 8, no. 8, pp. 221–224, 2001.

[15] J.F.Pitrelli, C.Fong, and H.C.Leung, "Phonebook: A phonetically-rich isolated-word telephone-speech database," in *Proc. ICASSP*, 1995.

[16] V.W.Zue, S.Seneff, and J.Glass, "Speech database development at MIT: TIMIT and beyond," *Speech Communication*, vol. 9, no. 4, pp. 351–356, 1990.

[17] D.Brooks, V.Tiwari, and D.Martonos, "Wattch: A framework for architecture level power analysis and optimizations," in *Proc. Intl. Symp. on Computer Architecture*, 2000, pp. 83–94.

[18] D.Burger and T.M.Austin, "The SimpleScalar tool set, Version 2.0," Univ. of Wisconsin-Madison, Tech. Rep. 1342, 1997.