

# Feature Pruning for Low-Power ASR Systems in Clean and Noisy Environments

Xiao Li and Jeff Bilmes

**Abstract**—Likelihood evaluation can substantially affect the total computational load for continuous hidden Markov model (HMM)-based speech-recognition systems with small vocabularies. This letter presents *feature pruning*, a simple yet effective technique to reduce computation and, hence, power consumption of likelihood evaluation. Our technique, under certain conditions, only evaluates the likelihoods of a fraction of feature elements and approximates those of the remaining (pruned) ones by a simple function. 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 1.3–1.8 and reduce its power consumption by 27%–43% for various recognition tasks, including those in noisy environments.

**Index Terms**—Gaussian evaluation, high speed, low power, speech recognition.

## I. INTRODUCTION

ISOLATED word and digit recognition has seen increased use in mobile devices (e.g., cell-phones and hand-held computers) since they provide a convenient mobile human-computer interface. Most contemporary automatic speech recognition (ASR) systems, however, use continuous-density hidden Markov models (CHMMs). Such systems can be computationally expensive and energy demanding, posing challenges for real-time processing and extended battery life for such embedded systems.

When CHMMs are used in small-vocabulary ASR applications, the state likelihood evaluation can dominate the operational load by taking up to 96% of the total computation [1]. 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 the Gaussians. This suggests two ways to improve efficiency: reducing the number of Gaussians and reducing their dimensionality.

The number of Gaussian evaluations can, in fact, be reduced in systems that employ tying since, once evaluated, the Gaussian score can be cached and utilized multiple times [2]–[6]. Alternatively, Gaussians can be pruned on the fly using Gaussian selection [1], [7] or nearest-neighbor approximation [8]. On the other hand, various feature selection methods

[9]–[11] tackle the dimensionality problem by selecting only the feature elements with the highest discriminative power. Additionally, feature elements can also be selectively computed online. Reference [12] presented an incremental feature pruning method applied to discrete-mixture models, and reference [13] proposed a partial distance elimination technique associated with feature element reordering.

Sharing the basic approach of [12] and [13], we presented in [14] our feature pruning algorithm and preliminary results. In this letter, we focus on how the Gaussian likelihoods are conditionally approximated and how the feature elements are clustered to maximize the amount of feature pruning while maintaining recognition performance. We present our results for isolated word recognition and digit recognition tasks, since likelihood evaluation is the computational bottleneck of such systems and since they are of interest for portable, low-power, and embedded applications. Furthermore, we investigate the degree to which our technique is noise robust (also relevant to portable devices) by evaluating on a variety of noisy speech corpora.

## II. FEATURE PRUNING

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

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

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

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

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

where  $O_k$  (resp.  $\mu_{i,k}$ ,  $\Sigma_{i,k}$ ) are the projection of the observation (resp. mean, variance) onto the  $k$ th subspace.

If any of the factors in (2) are very small, the full-space likelihood will also likely be small. Specifically, let  $m_{i,k}$  denote the maximum value of the  $k$ th subspace Gaussian, and define  $m_i = \prod_k m_{i,k}$ , which is the maximum value of the full-space Gaussian. If  $\mathcal{N}(O_1; \mu_{i,1}, \Sigma_{i,1}) \ll m_{i,1}$ , then  $\mathcal{N}(O; \mu_i, \Sigma_i) \ll m_i$ , and hence, the  $i$ th component will likely 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

Manuscript received June 10, 2004; revised October 21, 2004. This work was funded by the National Science Foundation under Grant ITR/RC-0086032. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Dominic K. C. Ho.

The authors are with the Electrical Engineering Department, University of Washington, Seattle, WA 98195 USA (e-mail: lixiao@ee.washington.edu; bilmes@ee.washington.edu).

Digital Object Identifier 10.1109/LSP.2005.847858

case when  $\mathcal{N}(O_1; \mu_{i,1}, \Sigma_{i,1})$  is large enough; otherwise, we approximate the score of the entire  $i$ th component (similar to [13]) by a simple function.

Note that the conditional independence properties made by the diagonal covariance Gaussian model does not imply that the likelihoods of the subspaces are also independent—in other words, given two nonindependent random variables  $A$  and  $B$  and a function that factorizes as  $f(A, B) = f_1(A)f_2(B)$ , it is not necessarily the case that  $f_1(A)$  is independent of  $f_2(B)$ . Indeed, we have found this to be true in our data, where in our experiments, there exists a correlation between subspace Gaussian *likelihoods*. Therefore, we choose to predict the full-space likelihood from a subspace likelihood.

As implied above, our technique is applicable at multiple ( $K$ ) levels, where the approximation scheme activates only at the level when the previously partially computed component score falls below a threshold. We let the indices  $1 \dots K$  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  $\ln \mathcal{N}(O; \mu_i, \Sigma_i)$

```

1:  $a \leftarrow \ln \mathcal{N}(O_1; \mu_{i,1}, \Sigma_{i,1})$ ;
2: for  $k = 2 \dots K$  do
3:   if  $a > t_{k-1}$  then
4:     compute  $\ln \mathcal{N}(O_k; \mu_{i,k}, \Sigma_{i,k})$ ;
5:      $a \leftarrow a + \ln \mathcal{N}(O_k; \mu_{i,k}, \Sigma_{i,k})$ ;
6:   else
7:      $a \leftarrow a + s_k$ ; break;
8:   end if
9: end for
10: return  $a$ ;

```

where  $t_k$ ,  $k = 1 \dots K - 1$ , are a set of thresholds, and  $s_k$  are a set of constants determined using least-mean-square (LMS) estimation from the  $\ln \mathcal{N}(O_k; \mu_{i,k}, \Sigma_{i,k})$  values of the training data; consequently, only a fraction of observation vectors are precisely computed to full dimensionality. Note that the additional operations introduced by this algorithm, namely, the comparison operations, are taken into account in our CPU time and power simulations in Section IV.

### III. CLUSTERING OF FEATURE ELEMENTS

For recognizers that use both static and dynamic features, an obvious way to cluster feature elements is to allocate static features to the first group, their deltas to the second, and double deltas to the last (when available). The advantage is that it simplifies the incorporation of our scheme into existing ASR systems. The disadvantage, however, is that it is not guaranteed to give an optimal reduction in computation. Therefore, we utilize a greedy data-driven approach to derive a better clustering.

Our underlying goal is to order the clusters such that the amount of pruning is maximized while recognition accuracy is maintained. Empirically, we have found that *recognition accuracy* strongly depends on the threshold values  $t_k$ , regardless of the clustering. In other words, given  $K$  and a set of thresholds,

we observe only mildly different accuracy rates for different clustering schemes. This is not unexpected, since the thresholds determine how often approximation takes place. The clusterings do have a small impact on accuracy, however, since it can affect the predictability of the likelihood scores and, hence, the effectiveness of the approximation.

Moreover, the clusterings have a great impact on the reduction in computation and power consumption; different clustering schemes cause different numbers of feature subsets to be pruned under the same thresholds. Therefore, we desire a clustering such that pruning will remove as many subsets as possible before the critical point  $t_k$  for stage  $k$ , where recognition accuracy begins precipitously dropping from the baseline. Accordingly, the first subset of feature elements to be computed should have the maximum probability of having a score below  $t_1$ , 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  $Pr\{\ln \mathcal{N}(O_1; \mu_{i,1}, \Sigma_{i,1}) < t_1\}$ , which can be estimated nonparametrically using a large amount of data obtained by running inference. Of course, this implies an intractable optimal clustering problem, so we instead have developed the following greedy strategy.

- 1) Repeat several 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  $E[c_1]$ . (Note that the optimal choice of  $K$  is beyond the scope of this letter, but see [12].)
- 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 (of any Gaussian component) falls below  $E[c_1]/d$ , an estimate of the average per-feature-element threshold. Select the  $d$  feature elements with the highest counts to the first subset.
- 3) Repeat steps 1) and 2) to get thresholds  $E[c_k]$ ,  $k = 2 \dots K - 1$ , and the corresponding feature element subsets. Note that at stage  $k$ , we select  $d$  feature elements from the remaining candidates based on the per-feature-element threshold  $E[c_k]/(k \cdot d)$ .

Note that our clustering procedure bears a resemblance to discriminative feature selection [11], [15], [16], where typically a fixed global subset of features is chosen that has the best influence on accuracy. Here, we share this goal, but we have the added task of reducing evaluation cost and the added characteristic that features are chosen dynamically, depending on the partial likelihoods of earlier evaluated feature clusters.

### IV. EXPERIMENTS AND RESULTS

We evaluated the speedup and power reduction of our scheme on two databases: NYNEX PhoneBook [17], an isolated word corpus, and Aurora 2.0 [18], a continuous digit recognition

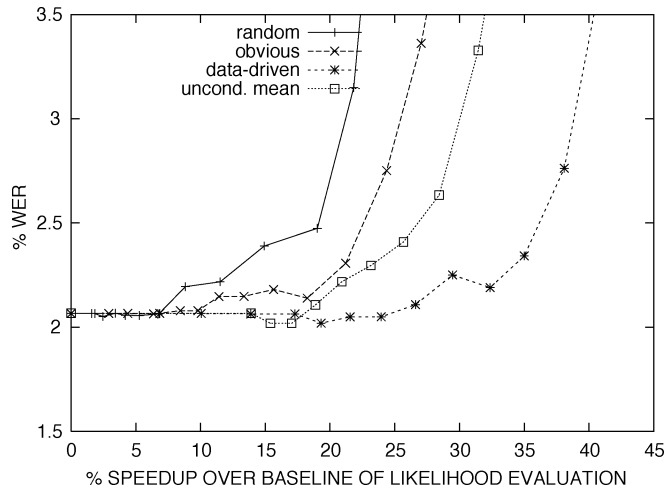


Fig. 1. Speedup achieved by feature pruning on PhoneBook. Thresholds were adjusted to obtain  $x$ -axis speedups and the corresponding WERs.

database with various noise conditions.<sup>1</sup> Note that the overall speedup and power reduction depend on the ratio of likelihood evaluation to search and other factors such as the vocabulary size. In these two applications, the search space and vocabulary size are relatively small—therefore, likelihood evaluation is indeed the computational bottleneck.

#### A. Isolated Word Recognition on PhoneBook

We used 42 continuous HMMs as an acoustic model with 165 states. 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 eight test sets each with a different 75-word vocabulary. The final word error rate (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 and over 80% with beam pruning applied.

We compared the speedup of feature pruning using our data-driven approach with that of the “obviously” grouped one mentioned above and a random grouping. We used  $K = 2$  in all cases. As shown in Fig. 1, the data-driven approach outperforms the other two by achieving a speedup of 1.35, with less than a 0.5% absolute increase in WER. To demonstrate the role of our approximations, we also compared this result with a scheme where the second part of a Gaussian likelihood is not approximated, as we propose, but rather is evaluated at the unconditional mean of the corresponding subspace Gaussian (using our best clustering). This scheme is called “uncond. mean” in the figure, and it performs worse than using average likelihoods. This is explained in the discussion in Section II.

To estimate the power consumed by the likelihood evaluation, we utilized Wattch, a framework for architecture-level power dissipation analysis [19]. We compiled the baseline system and the systems integrated with different amounts of feature

<sup>1</sup>In this work, we define

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

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

TABLE I  
BASELINE ACCURACIES USING MVA FEATURES

set	clean	noisy tests	most noisy test
testa	99.03%	83.46%	22.33%
testb	98.86%	83.43%	23.06%
testc	99.04%	83.06%	20.49%

TABLE II  
MAXIMUM SPEEDUP AND POWER REDUCTION ACHIEVED BY FEATURE PRUNING WITH A 0.5% DECREASE FROM BASELINE ACCURACIES

set	speedup / power reduction		
	clean	noisy tests	most noisy test
testa	1.78 / 42.5%	1.69 / 39.4%	1.76 / 41.8%
testb	1.80 / 43.0%	1.70 / 40.1%	1.74 / 40.5%
testc	1.77 / 42.6%	1.68 / 39.5%	1.79 / 42.0%

pruning, all targeted for the PISA architecture [20]. Power simulation was then launched, and cycle-level resource expenses were obtained, though only the power dissipation of likelihood evaluation was extracted in this letter. Since this process is quite time consuming, only subsets of the original test sets were used in our simulations. Feature pruning with data-driven clustering results in a power reduction of 27%, with again less than 0.5% absolute increase/decrease in WER/accuracy.

#### B. Aurora Digit Recognition

Aurora 2.0 is an English digit recognition corpus ideal to investigate noise-robust techniques for speech recognition. To create a state-of-the-art baseline, we used 39-dimensional features (MFCCs, log energy, and their deltas and second deltas), followed by mean subtraction, variance normalization, and ARMA filtering (MVA) post-processing [21]. MVA-processed features are highly effective for digit recognition under different noise conditions and are very simple to implement. We only studied the mismatched training/testing case, where CHMMs are trained using only clean speech and evaluated using clean, noisy (signal-to-noise ratio (SNR) from 20 dB to 0 dB) and most-noisy (SNR  $-5$  dB) speech. The baseline word accuracies using MVA features for the mismatched case are shown in Table I, where noisy test accuracy is an average of the 20 dB, 15 dB, 10 dB, 5 dB, and 0 dB SNR test sets.

Feature pruning was then applied to each test set, where  $K = 3$ . Table II reports the speedup factors and power reductions achieved by feature pruning for different test sets under different noise conditions. The results have a very similar trend as those in PhoneBook experiments. Here, we only report the speedup and power reduction when a 0.5% absolute decrease in word accuracy is allowed. As shown in the table, feature pruning achieves a speedup of up to 1.8 and power reduction over 40% for digit recognition in quiet environments. The speedup and power reduction remains significant when the SNR decreases. In the most adverse environments (SNR  $= -5$  dB), feature pruning still shows very good performance.

## V. CONCLUSION

Approaches such as Gaussian selection and feature element reordering [13] can reduce computation effectively. Our approach to low-complexity pruned Gaussian evaluation reduces

computation from a different perspective. It utilizes thresholds and returns approximate likelihood values, and the feature elements are clustered to maximize the amount of such pruning. Experiments show that our technique achieves a speedup of 1.3–1.8 and a power reduction of 27%–43% in the likelihood evaluation of two different recognition tasks. Furthermore, this technique is robust to different types and levels of noise.

#### ACKNOWLEDGMENT

The authors would like to thank J. Malkin for reading early drafts of this letter.

#### REFERENCES

- [1] E. Bocchieri, "Vector quantization for the efficient computation of continuous density likelihoods," in *Proc. ICASSP*, vol. 2, 1993, pp. 692–695.
- [2] K. F. Lee, S. Hayanuzu, 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.
- [3] S. J. Young and P. C. Woodland, "The use of state tying in continuous speech recognition," in *Proc. Eurospeech*, 1993, pp. 2203–2206.
- [4] X. Huang and M. Jack, "Semi-continuous hidden Markov models in isolated word recognition," in *Proc. 9th Int. Conf. Pattern Recognition*, vol. 1, Nov. 1988, pp. 406–408.
- [5] J. R. Bellegarda and D. Nahamoo, "Tied mixture continuous parameter modeling for speech recognition," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 38, no. 12, pp. 2033–2045, Dec. 1990.
- [6] E. Bocchieri and B. K. Mak, "Subspace distribution clustering hidden Markov model," *IEEE Trans. Speech Audio Process.*, vol. 9, no. 3, pp. 264–275, Mar. 2001.
- [7] 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. Speech Audio Process.*, vol. 7, no. 2, pp. 152–161, Mar. 1999.
- [8] F. Seide, "Fast likelihood computation for continuous-mixture densities using a tree-based nearest neighbor search," in *Proc. Eurospeech*, 1995, pp. 1079–1082.
- [9] E. Lleida and C. Nadeu, "Principal and discriminant component analysis for feature selection in isolated word recognition," in *Signal Processing V: Theories and Applications*. New York: Elsevier, 1990, pp. 1251–1254.
- [10] E. Bocchieri and J. Wilpon, "Discriminative feature selection for speech recognition," *Comput. Speech Lang.*, pp. 229–246, 1993.
- [11] J. Nouza, "Feature selection methods for hidden Markov model-based speech recognition," in *Proc. Int. Conf. Pattern Recognition*, vol. 2, 1996, pp. 186–190.
- [12] V. Digalakis, S. Tsakalidis, C. Harizakis, and L. Neumeyer, "Efficient speech recognition using subvector quantization and discrete-mixture HMMs," *Comput. Speech Lang.*, vol. 14, pp. 33–46, 2000.
- [13] B. Pellom, R. Sarikaya, and J. H. L. Hansen, "Fast likelihood computation techniques in nearest-neighbor based search for continuous speech recognition," *IEEE Signal Process. Lett.*, vol. 8, no. 8, pp. 221–224, Aug. 2001.
- [14] X. Li and J. Bilmes, "Feature pruning in likelihood evaluation of HMM-based speech recognition," in *Proc. IEEE ASRU Workshop*, 2003.
- [15] L. Jiang and X. Huang, "Acoustic feature selection using speech recognizers," in *Proc. IEEE Autom. Speech Recognition Understanding*, 1999.
- [16] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learning Res.*, vol. 3, pp. 1157–1182, 2003.
- [17] J. F. Pitrelli, C. Fong, and H. C. Leung, "PhoneBook: a phonetically-rich isolated-word telephone-speech database," in *Proc. ICASSP*, vol. 1, 1995, pp. 101–104.
- [18] H. G. Hirsch and D. Pearce, "The AURORA experimental framework for the performance evaluations of speech recognition systems under noisy conditions," in *Proc. ISCA ITRW ASR*, 2000, pp. 181–188.
- [19] D. Brooks, V. Tiwari, and D. Martonos, "Wattch: a framework for architecture level power analysis and optimizations," in *Proc. Int. Symp. Comput. Arch.*, 2000, pp. 83–94.
- [20] D. Burger and T. M. Austin, "The SimpleScalar Tool Set, Version 2.0," Univ. of Wisconsin-Madison, Tech. Rep. 1342, 1997.
- [21] C.-P. Chen, J. Bilmes, and K. Kirchhoff, "Low-resource noise-robust feature post-processing on AURORA 2.0," in *Proc. ICSLP*, 2002, pp. 2445–2448.