FAST GAUSSIAN LIKELIHOOD COMPUTATION BY MAXIMUM PROBABILITY INCREASE ESTIMATION FOR CONTINUOUS SPEECH RECOGNITION

Nicolás Morales[1], Liang Gu[2], Yuqing Gao[2]

[1]HCTLab. Universidad Autonoma de Madrid, SPAIN.
[2]IBM T.J. Watson Research Center, Yorktown Heights, USA.
nicolas.morales@uam.es, lianggu@us.ibm.com, yuqing@us.ibm.com

ABSTRACT

Speech signals are semi-stationary and speech features in neighboring frames are likely to share similar Gaussian distributions. A fast Gaussian computation algorithm is hence proposed to speed up the computation of the *N*-best posterior probabilities based on a large set of Gaussian distributions for the task of large vocabulary continuous speech recognition. The maximum probability increase between the current speech frame and a previous reference frame is estimated for all Gaussian distributions in order to reduce explicit computations of posteriors for a large number of Gaussians. The method was applied to the fMPE front-end of IBM's state-of-the-art speech recognizer resulting a decoding speed-up of 40% in probability computation for a loss-less mode and more than 55% in an approximated implementation, respectively.

Index Terms— Fast Gaussian computation, fMPE

1. INTRODUCTION

Gaussian likelihood computation may take up to 80% of total computation time in state-of-the-art Automatic Speech Recognition (ASR) systems. Such systems typically compute posterior probabilities of a large number of Gaussian distributions in the procedures of Viterbi search, model adaptation (MAP [1]), feature extraction (fMPE [2]), etc.

Several methods have been proposed to improve the speed of Gaussian likelihood computation in ASR. Some approaches make trade-offs between estimation accuracy and computational complexity, at the potential cost of lower recognition accuracy. For example, Gaussian clustering schemes [3] and tree-based search algorithms [4] [5] [6] quickly identify a subset of the most likely Gaussians for a particular frame. The computation of Gaussian posteriors can also be approximated by Hamming Distance Approximation [7]. Other approaches aim at improving computational speed without any loss of accuracy. A typical example is Partial Distance Elimination (PDE), where the likelihood computation process is terminated for a specific Gaussian mixture when its partial accumulated value is larger than the total posterior value of the current *N*-best candidate [8] [9]. Another approach via efficient programming is to use SIMD instructions that exploit the power of modern processors by parallelizing frequently repeated operations [10].

In this paper we propose *Maximum Probability Increase Estimation* (MPIE), a new method for fast Gaussian likelihood computation in ASR that reduces the number of Gaussian posteriors to be explicitly calculated and therefore enhances overall speech recognition speed without any accuracy loss. In particular, the Gaussian posteriors' information from previous speech frames is exploited in order to speed up likelihood computation for the current frame. Our approach is analogous to Gaussian clustering and tree-based pruning methods in the sense that computation is restricted to a selection of candidates. Moreover, it is risk-free and as accurate as the PDE approach in [9]. As a result, our approach can essentially enjoy the merits of both types of speed-up methods previously proposed.

The rest of this paper is organized as follows. In Section 2 we briefly describe fMPE feature extraction for which our method was applied. Section 3 presents the mathematical formulation of MPIE and Section 4 explains several implementation issues. In Section 5 we present experimental results using the latest version of IBM's large vocabulary continuous speech recognition system [11] and in Section 6 we discuss on results and future work.

2. FMPE FEATURE EXTRACTION

Feature space Minimum Phone Error (fMPE) [2] is a form of discriminative training recently proposed that uses the same objective function as MPE [12] in order to modify speech feature vectors during both ASR model training and ASR decoding.

Original feature vectors $\mathbf{x}_t$ for a frame t with dimension D are transformed as:

$$\mathbf{y}_t = \mathbf{x}_t + \mathbf{M} \cdot \mathbf{h}_t, \quad (1)$$

where $\mathbf{h}_t$ are high-dimensional features calculated from the original feature vector and $\mathbf{M}$ is a matrix that maps the high-dimensional space to the lower dimension of the original features.

Various types of composition of vector $\mathbf{h}_t$ have been proposed. In this paper, $\mathbf{h}_t$ consists of the concatenation of weighted offset features. The offset in each dimension is defined as the difference between the speech feature of the current frame and the mean of a Gaussian distribution in a set of L Gaussian multivariate functions that model the original feature space, weighted by the posterior of the Gaussian distribution. Additionally, the posterior probability score itself is assigned as an extra feature dimension. Consequently, the concatenation of such offset vectors for all Gaussians yields a feature vector $\mathbf{h}_t$ of dimension $R = L \cdot (D+1)$.

This work was conducted as part of an internship in IBM.

While significant improvement of ASR accuracy was achieved in large-vocabulary continuous ASR experiments [2], the substantial increase of computational complexity hence involved in front-end analysis during speech recognition remains a big concern, particularly for real-time client-based ASR applications where computational power is very limited. The problem may be alleviated by using very sparse $\mathbf{h}_t$ vectors, i.e. for each frame t only a few elements of $\mathbf{h}_t$ are non-zero. This may be achieved by setting all Gaussian posteriors to zero except the best C candidates [2]. In our implementation we set C=3 for optimal reduction of fMPE computational complexity.

3. MPIE FORMULATION

3.1. Basic principle of MPIE

In this paper Gaussian posteriors are denoted and computed as:

$$p_m(\mathbf{o}) = N(\mathbf{o};\boldsymbol{\mu}_m,\boldsymbol{\Sigma}_m) = \frac{1}{\sqrt{(2\pi)^D|\boldsymbol{\Sigma}_m|}}\cdot \exp\left(-\frac{1}{2}\cdot(\mathbf{o}-\boldsymbol{\mu}_m)^T\cdot\boldsymbol{\Sigma}_m^{-1}\cdot(\mathbf{o}-\boldsymbol{\mu}_m)\right), \quad (2)$$

where $\boldsymbol{\mu}_m$ and $\boldsymbol{\Sigma}_m$ are the mean vector and covariance matrix for Gaussian mixture m, respectively, and $\mathbf{o}$ is a speech feature vector with dimension D. When diagonal covariance matrixes are used, the logarithmic Gaussian posterior can be obtained as:

$$\log p_m(\mathbf{o}) = c_m - 0.5\cdot\sum_{d=1}^{D}(o_d-\mu_{m,d})^2\cdot v_{m,d} \quad (3)$$

where c_m is a constant for each Gaussian and $v_{m,d}$ are the diagonal elements of the inverse covariance matrix. Given two observations $\mathbf{o}_{t1}$ and $\mathbf{o}_{t2}$ from frames t_1 and t_2 respectively ($t_1 < t_2$), we may write an equivalent equation:

$$\log p_m(\mathbf{o}_{t2}) - \log p_m(\mathbf{o}_{t1}) = 0.5\cdot\sum_{d=1}^{D}\left[(o_{t1,d}-\mu_{m,d})^2-(o_{t2,d}-\mu_{m,d})^2\right]\cdot v_{m,d}; \quad (4)$$

$$\log p_m(\mathbf{o}_{t2}) = \log p_m(\mathbf{o}_{t1}) + \sum_{d=1}^{D}\left[A_d + B_d\cdot\mu_{m,d}\right]\cdot v_{m,d} = \log p_m(\mathbf{o}_{t1}) + \sum_{d=1}^{D}P_{m,d}(\mathbf{o}_{t1},\mathbf{o}_{t2}), \quad (5)$$

where $P_{m,d}(\mathbf{o}_{t1},\mathbf{o}_{t2})$ defines the change of likelihood for $\mathbf{o}_{t2}$ with respect to $\mathbf{o}_{t1}$ given Gaussian m and dimension d. We defined:

$$A_d = A(o_{t1,d},o_{t2,d}) = 0.5\cdot\left(o_{t1,d}^{\ 2}-o_{t2,d}^{\ 2}\right), \quad (6)$$

$$B_d = B(o_{t1,d},o_{t2,d}) = (o_{t2,d}-o_{t1,d}). \quad (7)$$

Vectors $\mathbf{A}=\{A_d\}$ and $\mathbf{B}=\{B_d\}$ are constant for all Gaussian mixtures and the corresponding computational cost is hence negligible compared to the cost of Eq. (3). Since Eq. (5) is equivalent to Eq. (3) and $\log p_m(\mathbf{o}_{t1})$ is known, it is possible to reduce the computational cost of $\log p_m(\mathbf{o}_{t2})$ if we devise a fast computation algorithm for $P_{m,d}(\mathbf{o}_{t1},\mathbf{o}_{t2})$. To achieve this goal, in MPIE, instead of using Eq. (5) for all Gaussians m, we estimate an upper bound of $P_{m,d}(\mathbf{o}_{t1},\mathbf{o}_{t2})$ for all Gaussians, corresponding to the maximum likelihood improvement that any Gaussian may get in frame $\mathbf{o}_{t2}$ compared to that in frame $\mathbf{o}_{t1}$. Specifically we define the upper bound as:

$$U = \sum_{d=1}^{D}U_d = \sum_{d=1}^{D}\left[A_d + B_d\cdot\mu_{\max/\min,d}\right]\cdot v_{\max,d}\ . \quad (8)$$

where U_d is the maximum possible probability improvement for dimension d. The improvement term in Eq. (8) was maximized by taking the maximum value of the inverse of the variance $v_{\max,d}$ for the whole set of Gaussians, and the maximum or minimum of the means values $\mu_{\max/\min,d}$, according to the sign of B_d. From Eq. (5):

$$\log p_m(\mathbf{o}_{t2}) \le \log p_m(\mathbf{o}_{t1}) + U\ , \quad (9)$$

As stated in [9], a good candidate to be the best scoring Gaussian for a particular frame $\mathbf{o}_{t2}$ is $p_{best\_t1}(\mathbf{o}_{t2})$, the best Gaussian of a neighboring previous frame $\mathbf{o}_{t1}$. In fMPE, we only search for the best C Gaussians and only if their posterior is larger than that of the best candidate plus the constant *prun*. Therefore, all mixtures m that fulfill the following inequation are irrelevant for frame t_2:

$$\log p_m(\mathbf{o}_{t1}) + \sum_{d=1}^{D}U_d < \log p_{best\_t1}(\mathbf{o}_{t2}) - prun \quad (10)$$

3.2. Partial MPIE (P-MPIE)

Estimation of U as in Eq. (8) is conservative (for the goal of removing a large number of Gaussians from the candidate list) and in some cases U may be so large that all Gaussians need to be evaluated explicitly. Typically, the contribution to the U function by different dimensions d is non-uniform; therefore, we propose to use the estimation for dimensions that contribute less, and perform explicit computation for those dimensions for which the estimation is large. In this fashion Eq. (10) is modified, so that mixtures that fulfill the following equation do not need to be recalculated:

$$\log p_m(\mathbf{o}_{t1}) + \sum_{d=1,d\in P}^{D}U_d + \sum_{d=1,d\notin P}^{D}P_{m,d}(\mathbf{o}_{t1},\mathbf{o}_{t2}) < \log p_{best\_t1}(\mathbf{o}_{t2}) - prun, \quad (11)$$

where

$$d\in P \quad iff \quad U_d \le imp\_thresh\ . \quad (12)$$

The elements $d\in P$ for which we estimate an upper bound are different for each frame, according to the relative variation of that particular frame compared to the reference frame, as well as the characteristics of the acoustic models.

4. IMPLEMENTATION ISSUES

A key element for a significant speed-up with the proposed approach is the minimization of the overhead cost. We found four sources of extra cost for our implementation:

a) Computation of vectors **A** and **B** as in Eqs. (6) and (7).

b) Computation of U function defined in Eq. (8).

c) Evaluation of the condition for mixture discarding, Eq. (11).

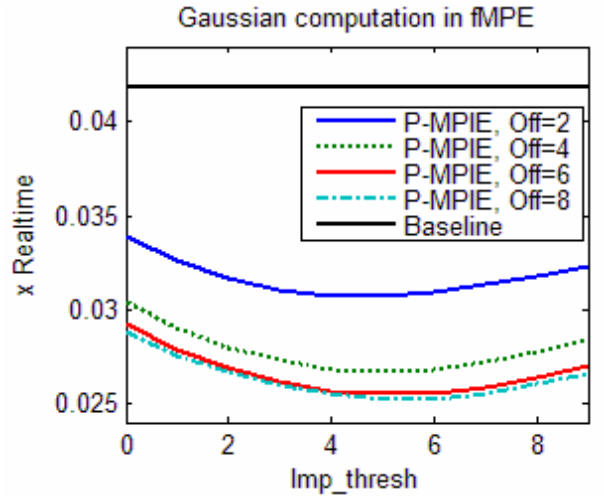

Figure 1. Cost of Gaussian computation in the fMPE module for the baseline and P-MPIE using fixed values of *offset* for full re-computation of Gaussians.

d) Break-up of computation of posteriors for those Gaussians that need to be re-evaluated.

The impact of a) and b) is very small because these are performed only once for each frame and the cost is negligible compared to that of computing Gaussian posteriors. Also, the evaluation of the condition for discarding mixtures is small because it is done once for each Gaussian.

Break-up of the computation of posteriors was an important problem. When using the P-MPIE approach, the contribution to the posterior of the elements of the feature vector $d \notin P$ is done in an initial stage and that for $d \in P$ is done later. In order to avoid the introduction of *if* conditions we used index mapping, resulting in very little overhead cost. Interestingly, we observed that explicit computation using Eq. (5) is faster than with Eq. (3), but this should be tested in different platforms and compilers before conclusions may be extracted.

Additionally we performed several other code optimizations; loop unrolling was used in all cases where the number of iterations is fixed (e.g. an operation repeated for the number of dimensions D of the feature vector). The whole code was profiled and some of the most costly operations (fMPE feature extraction and Viterbi search) were parallelized for nearly 50% speed-ups in two-core CPUs. Assembly code instructions were also employed for Gaussian likelihood computation.

5. EXPERIMENTAL RESULTS

In this section we show experimental results for several variants of MPIE for the computation of posteriors in fMPE. We use 1024 Gaussian mixtures and tests are for 1800 spontaneous utterances with 30k-word English vocabulary. Frame windows are 25 ms long with a 10 ms shift and preprocessing consists of MFCC feature extraction (24 features) followed by LDA.

5.1. Loss-less implementation

Here we compare speed of the baseline system (computing all posteriors for all frames) to that of Partial-MPIE. Parameter *imp_thresh* needs to be adjusted for a convenient trade-off, in order to assure a sufficiently high number of dimensions included in the estimation of U, but keeping the estimation small enough. Additionally, a criterion has to be defined so as to determine when to perform a complete computation of posteriors (after a few frames the reference and current observations are not similar anymore and MPIE is less effective).

Method →	Baseline	Fixed	DBL	ND
Speed-Up →	0 %	39.76 %	41.06 %	41.07 %

Table 1. Relative speed-up for three different criteria for full re-computation. *Imp_thresh* = 4.

In Figure 1 we show results for a fixed number of frames (*offset*) after which full computation is performed (e.g *offset*=4 means that full computation as in Eq. (3) is done every 4 frames and for all other frames P-MPIE is used).

The fastest configuration allows for a speed-up close to 40% compared to the baseline. The ideal value of *imp_thresh* is around 5.00 and depends on the set of Gaussians used, but not significantly on the type of data or the *offset* parameter. The ideal value of *offset* in this experiment is *offset*=6; full re-computation every 60 ms. In many cases this is excessive for the assumption of similarity of frames, however, the cost of full recalculation is large, thus favoring larger values of *offset*. This ideal value is nevertheless dependent on the speech characteristics (typically style, but other factors like noise also affect). In order to overcome the inconvenience of tuning it to specific data we propose the following two criteria for re-computation of all Gaussians for a given frame:

a) Difference between likelihood (*DBL*) of the best Gaussian in the reference frame and in the current frame is larger than a threshold.

b) Number of dimensions (*ND*) for which inequation (12) holds is smaller than a threshold.

Both approaches are less affected by the characteristics of data than setting a fixed *offset* and therefore are more efficient and easier to tune. In Table 1 we show speed-ups for the 3 approaches. In addition to the aforementioned tuning advantage the two non-fixed approaches perform faster.

When implemented in our latest IBM large vocabulary recognition engine on Windows XP, the proposed MPIE approach obtained an overall speed-up of 2.6% and 20%, when the graph search beam factor was set as 13 and 10, respectively. In the future we will further implement MPIE in the Viterbi decoder to achieve more significant improvements.

5.2. Approximated solution

The proposed method is conservative because it finds an upper bound for the likelihood improvement and assumes that all Gaussians may get such improvement. The maximum improvement is computed using the means value $\mu_{\max/\min,d}$, corresponding to outlier Gaussians that in most frames will have small likelihoods. Similarly, $v_{\max,d}$ corresponds to the narrowest Gaussians, also scoring badly in most cases. However, the estimation is applied to all Gaussians, incurring in significant overestimations. In order to reduce this effect we may assume some level of risk by multiplying the value of maximum improvement by:

$$Risk = Risk\_factor \cdot count\left(d \in P\right) / D\,, \qquad (13)$$

where $0 \le Risk\_factor \le 1$.

Imp_thresh	Risk factor	Errors Gauss. (%)	Rel. WER increase (%)	Speed-up (%)
4	0.02	0.20	0.000	40.23
	0.1	0.16	0.000	40.19
	0.5	0.00	0.000	40.14
12	0.02	1.04	0.038	49.12
	0.1	0.49	0.025	49.02
	0.5	0.04	0.000	46.64
20	0.02	2.10	0.075	54.11
	0.1	0.58	0.025	53.42
	0.5	0.02	0.000	42.75
28	0.02	3.82	0.100	57.23
	0.1	0.54	0.050	55.58
	0.5	0.02	0.000	35.90
36	0.02	5.76	0.138	59.55
	0.1	0.79	0.038	56.44
	0.5	0.02	0.000	30.97

Table 2. Speed-up and errors made for different values of the *risk_factor* and *imp_thresh* in the approximated implementation. Highlighted is a good trade-off.

In Table 2 we show relative speed-ups, percentage of errors in Gaussian classification (any error in the first C=3 Gaussians) and the relative impact in ASR with this setup. It is possible to obtain speed-ups over 55% relative to the baseline, with less than 1% errors in the selection of the best C candidates (and less than 0.04% relative increase in Word Error Rate).

6. DISCUSSION AND FUTURE WORK

The proposed approach has proved to be a very effective method for loss-less fast Gaussian computation. Even in the case of the approximated implementation, the impact in WER is so small that it could be assumed for very significant speed-ups. Our approach introduces extra operations in the Gaussian computation algorithm only at the level of Gaussian mixtures, while PDC [9] introduces these at the level of dimensions of the feature vector, resulting in larger overhead cost. PDC on the contrary is more effective for the purpose of removing operations and therefore obtains significant speed-ups too (we implemented PDC in its most efficient version, using *best mixture prediction* and *feature component reordering* for a speed-up of 45.47%). The combination of PDC and MPIE is under investigation and is likely to get further computational benefits, though PDC should not be used on the reference frames for which we need to know the posteriors for all Gaussians.

Another possibility to improve performance would be to group Gaussians according to their means and/or covariance values and perform P-MPIE independently for each group, thus obtaining more accurate estimations of MPIE.

Finally, an important topic that has not been addressed yet is the existence of very small values of diagonal covariance elements in Gaussian mixtures. The estimation in Eq. (8) uses $v_{\max,d}$, corresponding to the narrowest Gaussian of the set for each dimension. In our system no constraints were imposed on the narrowness of Gaussians and as a result, the relation between maximum and minimum values for different dimensions is typically over 10 and for 6 of the D=40 dimensions, it is over 50. This is an obvious cause for overestimation with MPIE and so, in the future we will study the possibility of imposing constraints on minimum narrowness of Gaussians, which would allow for significant improvements in speed at little or no accuracy costs.

7. CONCLUSIONS

In this paper we presented MPIE, a new algorithm for fast computation of the best candidates in a set of Gaussians. This is a novel idea that exploits information on Gaussian likelihood on previous frames in order to reduce the number of operations required for the current frame, by taking advantage of the semi-stationary nature of speech signals. The proposed implementation achieves 40% speed-up over the baseline in fMPE for a problem of large vocabulary continuous speech recognition in loss-less mode. We also presented a modification that allows for further speed improvement at a very moderate accuracy cost and proposed several other modifications for future work.

8. REFERENCES

[1] J.L. Gauvain and C.H. Lee, "Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains," *IEEE Trans. Speech and Audio Processing*, vol. 2, issue 2, pp. 291-298. April, 1994.

[2] D. Povey, B. Kingsbury, L. Mangu, G. Saon, H. Soltau and G. Zweig, "fMPE: Discriminatively trained features for speech recognition," *Proc. DARPA EARS RT-04 Workshop*, Paper no. 35. Palisades, NY. November, 2004.

[3] E. Bochieri, "Vector quantization for the efficient computation of continuous density likelihoods," *Proc. ICASSP'93*, vol. 2, pp. 692-695. April, 1993.

[4] F. Seide, "Fast likelihood computation for continuous-mixture densities using a tree-based nearest neighbor search," *Proc. EuroSpeech'95*, pp. 1079-1082. September, 1995.

[5] S. Ortmanns, T. Firzlaff and H. Ney, "Fast likelihood computation methods for continuous mixture densities in large vocabulary speech recognition," *Proc. EuroSpeech'97*, pp. 139-142. September, 1997.

[6] A. Chan, M. Ravishankar and A.I. Rudnicky. "On improvements to CI-based GMM selection," *Proc. Eurospeech'95*, pp. 565-568. September, 2005.

[7] P. Beyerlein and M. Ullrich, "Hamming distance approximation for a fast log-likelihood computation for mixture densities," *Proc. EuroSpeech'95*, pp. 1083-1086. September, 1995.

[8] L. Fissore, P. Laface, P. Massafra and F. Ravera, "Analysis and improvements of the partial distance search algorithm," *Proc. ICASSP'93*, vol. 2, pp. 315-318. April, 1993.

[9] B. Pellom, R. Surikaya and J.H.L. Hansen, "Fast likelihood computation techniques in nearest-neighbor based search for continuous speech recognition," *IEEE Signal Processing Letters*, vol. 8, issue 8, pp. 221-224. August, 2001.

[10] K. You, Y. Lee and W. Sung, "Mobile CPU based optimization of fast likelihood computation for continuous speech recognition," *Proc. ICASSP'07*, vol. 4, pp. 985-989. May, 2007.

[11] H. Soltau, G. Saon, B. Kingsbury, J. Kuo, L. Mangu, D. Povey and G. Zweig, "The IBM 2006 Gale Arabic ASR system," *Proc. ICASSP'07*, vol. 4, pp. 349-352. April, 2007.

[12] D. Povey and P. Woodland, "Minimum phone error and I-smoothing for improved discriminative training," *Proc. ICASSP'02*, vol. 1, pp. 105–108. May, 2002.