Gaussian Selection with Non-Overlapping Clusters for ASR in Embedded Devices

Jussi Leppänen and Imre Kiss
Multimedia Technologies Laboratory, Nokia Research Center
Tampere, Finland

ABSTRACT

In this paper we propose a memory efficient version of the Gaussian Selection (GS) scheme, which is used for speeding up the likelihood calculations of an ASR system. The memory savings are achieved by using non-overlapping (disjoint) clusters instead of the overlapping clusters normally used in GS. As we will show, the new scheme achieves 66% computational savings with a relative increase in word error rate (WER) of 4%. We will also show, that combining the new GS scheme with frame rate reduction and feature masking provides further savings in computation. 75% (4% increase in WER) and 68% (3.5% increase in WER) savings were obtained by adding frame rate reduction and feature masking, respectively.

1. INTRODUCTION

As voice user interface technology is maturing, it is becoming a more and more important input/output method for small, embedded devices. Using a voice user interface is especially convenient when the device is being used in situations where normal input methods are not available.

For embedded devices, low memory and computational complexity implementations of the ASR algorithms is very important. Even though the computational power of embedded devices in rising constantly, cost will always be an important factor in designing mass-market products. Moreover, there will always be an increasing amount of applications competing for the same computational resources as the voice UI.

In a HMM based speech recognizer more than half of the computational time can be spent in calculating the density likelihoods. Thus, any decrease in density calculation time will have an effect on the overall speed of the recognition algorithm. Numerous efficient algorithms have been proposed that address this problem. Using vector or scalar quantization of the acoustic model parameters, for example, allows for the acoustic models to be stored in a smaller amount of memory and for faster likelihood calculation without affecting recognition performance [1,2]. In [3], several techniques (feature component masking, variable rate updating of feature components and density pruning) for reduced complexity likelihood calculation are proposed. In [4], Gaussian Selection (GS) is used to select a shortlist of Gaussians for which to calculate accurate likelihoods, thus reducing computation.

In this paper, we look at a few of the above methods for speeding up the process of calculating the density likelihoods. More, specifically, we examine GS, for which we present here a memory efficient implementation. We also look at how GS performs in combination with frame rate reduction and feature vector masking.

The rest of the paper is organized as follows. First, in Section 2.1, we will review GS. Then, in Section 2.2, we will introduce the proposed memory efficient GS implementation. Section 3 we look at two methods, frame rate reduction and feature vector masking, which can also be used to reduce the computational complexity of the likelihood computations. In Section 4 we show the results of recognition experiments done using the original GS and the proposed GS method. In addition we show how frame rate reduction and feature masking work together with GS. Conclusions are then finally drawn in Section 5.

2. ALGORITHM DESCRIPTIONS

2.1 Gaussian Selection

GS was first introduced by Bocchieri in [4] and is used to limit the number of likelihood calculations needed during decoding. The motivation in GS is that the likelihood of a feature vector can be approximated accurately only when it does not land on the tail of a Gaussian density [4]. Also, when the feature vector does land on the tail of a Gaussian density, the likelihood will be small, and thus it won't contribute much to the state score. This implies that it would be beneficial to determine quickly the subset of Gaussians that the feature vector is not an outlier to, before the actual likelihood calculation. The likelihoods of these Gaussians would then be calculated and the likelihoods of the rest of the Gaussians would be set to some small constant.

Gaussian densities are first grouped together into overlapping neighborhoods. These neighborhoods are created by first applying k-means clustering on the densities. The distance measure used in the clustering is a weighted Euclidian distance metric:

$$\partial(\mu_i,\mu_j)=\frac{1}{d}\sum_{k=1}^{d}\left\{w(k)\left(\mu_i(k)-\mu_j(k)\right)\right\}^2 \qquad (1)$$

where d is the dimensionality of the feature vector, $\mu_i(k)$ is the k^{th} component of the mean of the i^{th} Gaussian density and w(k) is a weight equal to the k^{th} diagonal element of the inverse square root of the average of the covariances of the Gaussian in the acoustic model set.

After clustering, the cluster centers are stored and a neighborhood of Gaussians is determined for each of them. The neighborhood of a cluster center comprises all Gaussians for which the following equation holds:

$$\frac{1}{d}\sum_{i=1}^{d}\frac{\left(c(i)-\mu_m(i)\right)^2}{U_{avg}(i)}\le\Theta \qquad (2)$$

where c(i) is the ith component of the cluster center, and $U_{avg}(i)$ is the i^{th} diagonal component of the average of the covariance matrices of the Gaussians in the model set. Θ is a threshold, which controls the size of the neighborhood. It is also possible to use state information during the neighborhood creation to obtain better performance, as described in [5]. In the experiments presented in this paper, however, state information is not used during clustering.

During decoding, after a feature vector is obtained, the cluster center that is closest to the feature vector is found Equation 1 is used for the distance calculation by replacing μ_j with the current feature vector. The likelihoods for the Gaussians that belong to the neighborhood of this cluster center are then evaluated. The likelihood of the other Gaussians is set to some small constant value. The number of likelihoods that are calculated for every frame is controlled by the threshold Θ (see Equation 2). The smaller the Θ value, the less likelihoods are calculated which results in reduced computational complexity. Setting Θ too low, will however, result in lower recognition accuracies as too few likelihoods are calculated.

2.2 Gaussian Selection with non-overlapping clusters

2.2.1 Motivation

While GS has been shown to reduce the computation needed for the calculation of the density probabilities significantly, its use in embedded devices might not be justified because of increased memory requirements. The increased memory footprint needed for GS is due to two factors. First, the cluster centers and the distance weight need to be stored. The memory needed for these is however usually negligible when compared to the memory needed to store the actual densities. The number of neighborhoods ranges usually from 64 to 512, while the number of densities in a triphone acoustic model set might be several tens of thousands. Another source for the increased memory footprint is due to the fact that the neighborhood member information needs to be stored. The memory needed for this information is quite high. Consider for example an acoustic model set with 25K densities and 256 neighborhoods. Since each density may belong to any of the neighborhoods, a 25,000 x 256 (1 bit elements) table that holds the neighborhood member information needs to be stored along with the models. This requires 800KB of memory. The size of a subspace distribution clustered acoustic model set which uses 4-bit quantization of the mean-variance value pairs requires about 600-700KB [1]. This means that the GS information would more than double the memory footprint of the acoustic models!

If, however, disjoint clusters were to be used instead of the overlapping neighborhoods, the cluster member information would require much less memory. This would be achieved by first arranging the densities in the memory according to the cluster membership information such that the densities belonging to the first cluster are placed first and so on. Now, only a table with as many elements as there are clusters would be needed. The elements in the table would represent the indices of the first density that belongs to the respective cluster.

As mentioned in [4], using disjoint clusters results in problems when the feature vector lands near the edge of a cluster. When this happens, the densities that are close to the feature vector but lie on the 'wrong side' of the cluster border are not evaluated. This problem can be mitigated by keeping the cluster sizes relatively small (smaller than the neighborhoods) and picking several clusters for evaluation instead of picking just the closest cluster.

Using disjoint clusters and picking more than one of them are the main ideas of the GS scheme proposed in this paper. This new scheme will be referred to as DCGS (disjoint cluster GS) for the rest of the paper. The clustering procedure and the cluster selection process done during decoding is explained next, in Sections 2.2.2 and 2.2.3, respectively.

2.2.2 Density clustering

In DCGS the densities are clustered into disjoint clusters using a binary divisive k-means clustering algorithm. The clustering is done such that every density is first placed in a single cluster whose mean is then calculated (average of the Gaussian means). The cluster is then split by perturbing the mean in opposite directions by a small amount and then reassigning the densities to the newly obtained means. K-means is then run for a few iterations and the clusters are split again. This procedure is repeated until a desired number of clusters is obtained. The distance metric used in the clustering is the same as is used in the original GS scheme (Equation 1).

During clustering a threshold was set such that any cluster, whose member count was below the aforementioned threshold, was not split. The use of the threshold resulted in the algorithm producing clusters with more even member counts than when the threshold was not used. This, in turn, means that the computational load is more predictable, as the number of clusters chosen for every frame is relatively constant (and every cluster has a similar number of cluster members).

2.2.3 Density selection

As mentioned before, since we are using disjoint clusters, more than one cluster needs to be selected for which to calculate the density probabilities. There are at least two ways of doing this. One possible way is to use a threshold-based selection such that all clusters are selected whose distance to the current feature vector is less than a certain threshold. This method is referred to as DCGS-T. Another way is to pick the N clusters that are closest to the current feature vector. This will be referred to as DCGS-N. The distance measure used here is also the one in Equation 1.

Notice that the number of likelihood calculations done per frame is controlled by either the N value (in DCGS-N) or the distance threshold (in DCGS-T). The N value and the threshold affect only the number of clusters that are chosen for likelihood calculation. Thus, it is easy to change them to increase or decrease the number of likelihood calculations done per frame, even on the fly, during decoding. This is not the case, however, in the original GS scheme, where neighborhoods are used. There the amount of likelihood calculations is controlled by the Θ value, which controls the size of the neighborhoods. This means that, when the Θ value is changed, the neighborhood members need to be calculated again to reflect the new Θ value.

3. FRAME RATE REDUCTION AND FEATURE VECTOR MASKING

In this section we describe two methods, frame rate reduction and feature vector masking, which also address the problem of costly density likelihood computation [3]. These methods have been found to decrease the computational complexity of the likelihood computation.

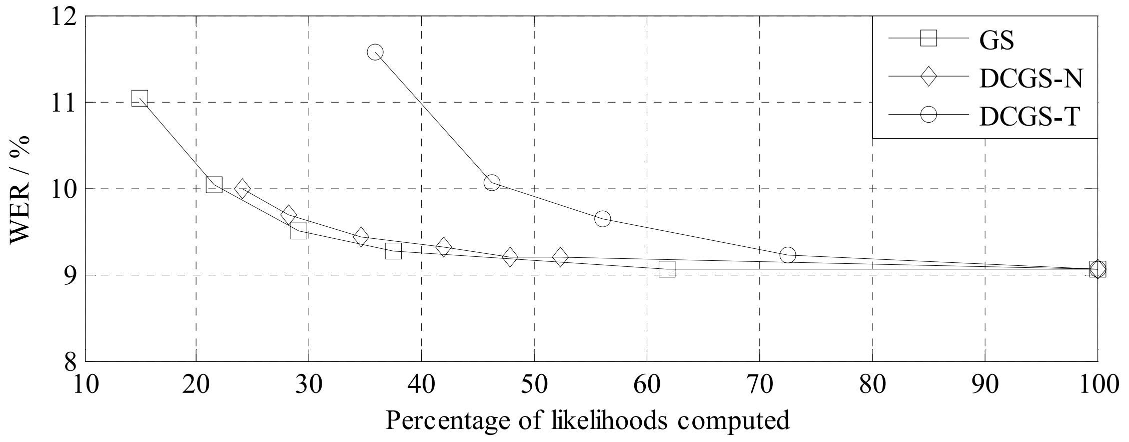

Figure 1 Word error rate vs. the percentage of likelihoods calculated per frame for GS, DCGS-N and DCGS-T.

3.1 Frame rate reduction

Frame rate reduction [3] is a simple and effective way of reducing the computational complexity of the density likelihood calculations. When frame rate reduction is used, the likelihoods are calculated e.g., for every other frame and then used again for the following frame. The motivation behind this is the assumption that consecutive feature vectors do not differ very much from each other. Thus, the density likelihoods for consecutive frames will be similar. It is also possible to calculate the likelihoods only for every third, fourth, etc. frame. The recognition accuracy will, however, drop quite fast if the likelihoods are calculated for less than every third or fourth frame [3].

3.2 Feature vector masking

The idea behind feature masking is that feature components contribute differently to the density likelihoods and the overall recognition performance [3]. It turns out that some components can be left out or masked altogether without affecting the recognition performance. The computational complexity is affected as the density likelihoods are calculated based on only the non-masked components. The masks can be determined, for example, by masking each feature component separately and checking the recognition performance for each such mask. The mask that is to be used is then created by combining the single component masks that affected the recognition performance the least.

4. EXPERIMENTS

4.1 Experimental setup

The performance of the proposed DCGS scheme was tested on a medium vocabulary continuous speech recognition task. The task vocabulary was around 1000 words. The acoustic models used in the experiments were standard decision tree state-tied 3-state triphone HMMs with 16 densities per state. The total number of densities in the set was 26K. The models were trained on an in-house training set containing continuous speech (US English). For the GS experiments, the densities were clustered into 128 neighborhoods (GS) or 150 clusters (DCGS). These cluster and neighborhood counts were found to work best in previous experiments, not presented here. The language model used here was a bigram language model.

The front-end used in the experiments was based on FFT-derived Mel cepstral coefficients and their first and second order derivatives (39 components in total). Recursive mean removal was applied on all components of the resulting feature vectors, and the variance of the energy component and its derivatives was normalized to unity [6].

4.2 Experimental Results

Figure 1, shows the word error rate achieved in the recognition experiments for the original GS scheme as well as for the proposed DCGS-N and DCGS-T schemes. The results for the original GS scheme were obtained by using 1.1, 1.3, 1.5, 1.7 and 2.3 for the Θ values. For DCGS-N, the N-values used were 32, 40, 48, 56, 64 and 72. The distance thresholds used in DCGS-T were 80, 90, 100 and 120. As it can be seen, the GS and DCGS-N schemes perform nicely and also very much alike, with respect to word error rate and computational savings. DCGS-N gives a word error rate of 9.43% at 34.6% of likelihoods computed while GS gives approximately the same word error rate at 29% of likelihoods calculated. DCGS-T, however, does not perform very well. The word error rate increases quite fast as the density threshold is tightened.

4.2.1 Frame rate reduction

To see whether frame rate reduction could be used in conjunction with GS to provide further savings in computational complexity, the following tests were performed. First, the recognizer was run without GS, but with the density frame rate set to 2, which meant that the density likelihoods were calculated for every other frame and reused for the next frame. The word error rate in this experiment turned out to be 9.29%. By looking at Figure 1, it can be seen that, the same number of likelihood computations can be achieved by using GS such that the word error rate is around 9.2%. So, based on word error rate and computational complexity it would seem that using GS is a slightly better option than using frame rate reduction.

However, things look a bit different when both GS and frame rate reduction are applied simultaneously. Figure 2, shows the word error rates for the DCGS-N scheme (DCGS-N) and the

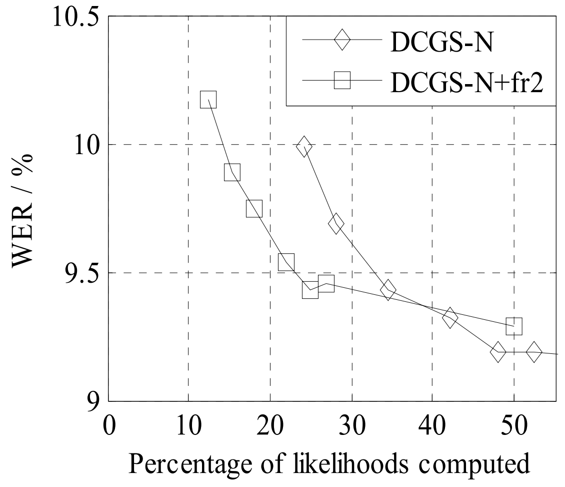

Figure 2 Word error rate vs. the percentage of likelihoods calculated per frame for DCGS-N and DCGS-N combined with frame rate reduction.

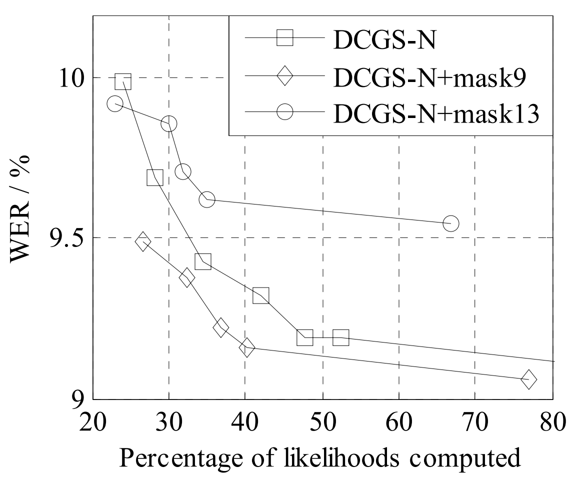

Figure 3 Word error rate vs. the percentage of likelihoods calculated per frame for DCGS-N and DCGS-N combined with feature masking.

DCGS-N scheme and frame rate reduction (DCGS-N+fr2). Note that in the figure the percentages of likelihoods calculated are relative to the case where no GS or frame rate reduction is used. Thus, when only frame rate reduction is enabled, the percentage is 50. From the figure, it is evident that frame rate reduction provides additional savings in computation. The same word error rate (~9.4%) is achieved by the DCGS-N scheme at around 34% of likelihoods evaluated as for the DCGS-N+frame rate reduction scheme at 25% likelihoods evaluated.

4.2.2 Feature masking

The performance of feature masking was tested together with the DCGS-N scheme. Two different feature masks were tested, one with 9 components masked and another one with 13 components masked. The results are presented in Figure 3. The masking was done such that it was not applied to the cluster selection process. Only the density calculation was affected. From the figure, it can be seen, that when combining with DCGS, a 9-feature mask brings further savings in likelihood calculation, while the 13-feature mask does not. For the 13-component mask, the word error rate is already relatively high (9.55%) before applying DCGS-N. Note that the likelihood percentages in Figure 3 have the savings from the feature masking included in them. For example, setting N to 64 results in 47.8% of the likelihoods to be computed, but when 9 out of the 39 components are masked the equivalent percentage is (47.8%*30/39=) 36.8%.

5. CONCLUSIONS

In this paper, we examined the performance of a memory efficient Gaussian Selection algorithm intended for use in embedded ASR systems. The proposed algorithm performed nearly at the same level on a medium vocabulary continuous speech recognition task as the original Gaussian Selection algorithm but with significantly reduced memory requirements. The proposed algorithm was able to obtain a 66% complexity reduction in likelihood computation with only a 4.1% relative increase in word error rate. When applying frame rate reduction in addition to the proposed GS scheme, a 75% complexity reduction was obtained with the same relative increase in word error rate. Combining the proposed GS scheme with feature masking also provided additional savings. A complexity reduction of 68% was achieved with a 3.5% relative increase in word error rate.

6. ACKNOWLEDGEMENTS

This work has partially been funded by the European Union under the integrated project TC-STAR - Technology and Corpora for Speech-to-Speech Translation - (IST-2002-FP6-506738, http://www.tc-star.org).

7. REFERENCES

[1] E. Bocchieri, and B. Mak, “Subspace Distribution Clustering for Continuous Observation Density Hidden Markov Models,” in Proceedings of the 5th European Conference on Speech Communication Technology, vol. 1, pp.107-110, 1997.

[2] M. Vasilache, “Speech Recognition Using HMMs with Quantized Parameters,” in Proceedings of Eurospeech 2001, Aalborg, Denmark, vol. 2, pp. 1265-1268, 2001.

[3] I. Kiss, and M. Vasilache, “Low Complexity Techniques for Embedded ASR Systems,” in Proceedings of ICSLP 2002, Denver, Colorado, USA, pp. 1593-1596, 2002.

[4] E. Bocchieri, “Vector Quantization for Efficient computation of continuous density likelihoods,” in Proceedings of ICASSP 1993, Minneapolis, MN, USA, vol. 2, pp. II-692 – II-695, 1993.

[5] M. J. F. Gales, K. M. Knill, and S. J. Young, “State-Based Gaussian Selection in Large Vocabulary Continuous Speech Recognition Using HMM’s,” *IEEE Transactions on Speech and Audio Processing*, vol. 7, no. 2, March 1999.

[6] O. Viikki, D. Bye, and K. Laurila, “A Recursive Feature Vector Normalization Approach for Robust Speech Recognition in Noise,” in Proceedings of ICASSP 1998, Seattle, Washington, USA, pp. 1692-1695, 1998.