STATE-BASED GAUSSIAN SELECTION IN LARGE VOCABULARY CONTINUOUS SPEECH RECOGNITION USING HMMS

M.J.F.Gales, K.M.Knill & S.J.Young

CUED/F-INFENG/TR 284

January 1997

Cambridge University Engineering Department
Trumpington Street
Cambridge CB2 1PZ
England

Email: {mjfg,sjy}@eng.cam.ac.uk

Abstract

This paper investigates the use of Gaussian Selection (GS) to increase the speed of a large vocabulary speech recognition system. Typically 30-70% of the computational time of a HMM-based speech recogniser is spent calculating probabilities. The aim of GS is to reduce this load by dividing the acoustic space into a set of clusters and associating a "short-list" of Gaussians with each of these clusters. Any Gaussian not in the short-list is simply approximated. This paper examines new techniques for obtaining "good" short-lists. All the new schemes make use of state information, specifically which state each of the components belongs to. In this way a maximum number of components per state may be specified, hence reducing the size of the short-list. The first technique introduced is a simple extension of the standard GS one, which uses this state information. Then, more complex schemes based on maximising the likelihood of the training data are proposed. These new approaches are compared with the standard GS scheme on a large vocabulary speech recognition task. On this task, the use of state information reduced the percentage of Gaussians computed to 10-15%, compared with 20-30% for the standard GS scheme, with little degradation in performance.

1 Introduction

High accuracy large vocabulary continuous speech recognition (LVCSR) HMM-based systems have been developed in recent years. These systems tend to operate at several times real-time and this is not practical for most applications. Techniques are therefore required to reduce the decoding time to faster than real-time, while maintaining, or staying close to, the same level of accuracy. A significant contribution (between 30 to 70%) to the computational load of LVCSR systems comes from the state likelihood calculation. This is a result of the need to use multiple components in a state. Anywhere from 8 to 64 components are typically used and each component requires evaluating separately in order to determine the overall likelihood. Gaussian Selection (GS) was first proposed by Bocchieri [3] in order to reduce this computation, thus reducing decode time. This paper attempts to refine the GS technique in order to make further reductions.

The motivation behind GS is as follows. If an input vector is an outlier with respect to a component distribution, i.e. it lies on the tail of the distribution, then the likelihood of that component producing the input vector is very small. This results in the likelihood of the input frame per-component within a state having a large dynamic range, with one or two components tending to dominate the state likelihood for a particular input vector. Hence, the state likelihood could be computed solely from these components without a noticeable loss in accuracy. GS methods attempt to efficiently select these components, or a subset containing them, at each frame.

In the method proposed by Bocchieri [3], the acoustic space is divided up during training into a set of vector quantised regions. Each component is then assigned to one or more VQ codewords. During recognition, the input speech vector is vector quantised. Only the likelihood of the input frames given components associated with the VQ codeword corresponding to the observation vector are computed exactly and the remaining likelihoods are approximated. In Bocchieri's work, the VQ codebooks were generated by clustering the Gaussian components. An alternative data-driven approach was proposed by Murveit et al [8] where the codebooks were generated by clustering training data.

Alternative approaches to reducing the state likelihood computation have recently been proposed. These approaches perform a direct search during recognition to determine which components are close to the observation vector. Only the likelihood of the input frames given these components are fully computed. Unfortunately however, for many HMM systems the computation required to perform the search can match that of the state likelihood computation. Hence, these methods tend to suit systems that have either a single shared covariance [2], or a large number of components per state [5].

This paper investigates the GS approach proposed by Bocchieri and focuses on methods for selecting the "best" short-list of Gaussians for each codeword. Section 2 reviews the standard GS scheme of Bocchieri, how the Gaussian codewords are generated and performance assessed. The following section describes how state information may be incorporated into the selection process. Section 4 details a refinement of this based on a maximum likelihood approach to GS. These schemes are evaluated on a standard large vocabulary speech recognition task, the unlimited vocabulary Hub 1 task from the 1994 ARPA evaluation [11].

2 Gaussian Selection

The first task in Gaussian selection is to generate a set of codewords, or clusters. Each of these codewords will then have an associated "short-list" of Gaussians which should be calculated when an input vector falls within that codeword cluster.. The decrease in computational load is dependent on the size of this "short-list".

2.1 Gaussian Cluster Generation

For the work presented here a single codebook was used and the codewords were generated by clustering Gaussians during training as follows. A weighted Euclidean distance between the means

was used to calculate the distance between the ith and jth Gaussians.

$$\delta(\mu_i, \mu_j) = \frac{1}{K} \sum_{k=1}^{K} \{w(k)(\mu_i(k) - \mu_j(k))\}^2 \tag{1}$$

where K is the feature vector dimension, $\mu_i(k)$ is the kth component of vector μ_i, and $w(k)$ is equal to the inverse square root of the kth diagonal element of the average of the covariances of the Gaussian set $\mathcal{N}(\mu_m, \mathbf{\Sigma}_m), m = 1, \ldots, M$, where $\mathbf{\Sigma}_m$ is the diagonal covariance of the mth Gaussian component. Using this metric, a clustering procedure based on the Linde-Buzo-Gray [7] algorithm was applied to minimise the average (per Gaussian component) distortion, δ_{avg},

$$\delta_{avg} = \frac{1}{M} \sum_{m=1}^{M} \left\{ \min_{\phi=1}^{\Phi} \delta(\mu_m, \mathbf{c}_\phi) \right\} \tag{2}$$

where M is the number of Gaussian components in the model set, Φ is the number of clusters (pre-defined), and $\mathbf{c}_\phi$ is the centre (codeword) of the ϕth cluster χ_ϕ,

$$\mathbf{c}_\phi = \frac{1}{size(\chi_\phi)} \sum_{m \in \chi_\phi} \mu_m, \phi = 1, \ldots, \Phi \tag{3}$$

In all schemes, it is necessary to assign each observation vector (i.e. speech input vector) to one of the clusters during recognition in order to select the Gaussian short-list to evaluate. The appropriate cluster is selected by determining the cluster centroid, $\mathbf{c}_\phi$, which minimises the weighted Euclidean distance to the observation vector, $\mathbf{o}(t)$, at time t,

$$\chi_t = \arg \min_{\chi_\phi} \delta(\mathbf{o}(t), \mathbf{c}_\phi) \tag{4}$$

In some of the Gaussian selection schemes proposed here it is also necessary to assign the training data to one of the clusters. Although the same cluster selection scheme may be used, there is also the option of using a softmax assignment. Hence for generality when describing the various Gaussian selection schemes, the cluster selection will be described in terms of a "probability", $p(\chi_\phi|\mathbf{o}(t))$. However, in all experiments described in this work a hard assignment as determined by equation 4 was actually used.

2.2 Standard Gaussian Selection

If a Gaussian short-list for cluster χ_ϕ, ν_ϕ, was generated solely from the Gaussians used in the cluster generation, the clusters would be disjoint. When used in recognition, errors would then be likely since the likelihood of some Gaussians close to the input feature vector but not in the selected cluster would be excluded from the full computation. To avoid this, clusters share Gaussians as follows. Given a threshold $\Theta \geq 0$, an input feature vector, $\mathbf{o}$, is said to fall on the tail of the mth Gaussian if

$$\frac{1}{K} \sum_{k=1}^{K} \frac{(o(k) - \mu_m(k))^2}{\sigma_m^2(k)} \gg 1 \tag{5}$$

where $\sigma_m^2(k)$ is the ith diagonal element of $\mathbf{\Sigma}_m$. Thus, if the cluster centroid is taken to be a typical input feature vector the Gaussian short-list, ν_ϕ, of codeword χ_ϕ, can be defined as consisting of all Gaussians such that [3]

$$G(m) \in \nu_\phi \text{ iff } \frac{1}{K} \sum_{k=1}^{k} \frac{(c_\phi(k) - \mu_m(k))^2}{\sigma_{avg}^2(k)} \leq \Theta \tag{6}$$

where $\sigma_{avg}^2(i)$ is the ith diagonal element of the matrix of the average covariance of all Gaussians in the system and $G(m)$ indicates component m of the standard recognition system.

An alternative method for determining which Gaussians are to be in the short-list is to use

$$G(m) \in \nu_\phi \text{ iff } \frac{1}{K} \sum_{k=1}^{K} \frac{(c_\phi(k) - \mu_m(k))^2}{\sqrt{(\sigma_{avg}^2(k)\sigma_m^2(k))}} \leq \Theta \tag{7}$$

where $\sigma^2_{avg}(k)$ is the kth diagonal element of the average covariance of all the components in the HMM set, and $\sigma^2_m(k)$ the same for the mth component. The advantage of this measure is that components with large variances are encouraged to be assigned to a cluster and this has been shown to perform better than the scheme described by equation 6 [6] [1]. This scheme, which is referred to as *Standard Gaussian Selection* (SGS), is used here to provide a performance baseline.

2.3 Gaussian Selection Performance

The performance of the various Gaussian selection schemes is assessed in terms of both the recognition performance and the reduction in the number of Gaussian components calculated. This reduction, the computation fraction, C, is defined as

$$C = \frac{G_{new} + VQ_{comp}}{G_{full}} \tag{8}$$

where G_{new}, G_{full} are the average number of Gaussians calculated per frame in the GS and full system respectively, and VQ_{comp} the number of computations required to calculate the VQ index. The latter is dependent on the size of the codebook and search technique used. A number of techniques exist to efficiently select the codeword (such as [10]) so the contribution of VQ_{comp} to the computation fraction is minimal. For these reasons, in this paper performance is assessed only in terms of the number of Gaussians calculated compared to the standard recognition system, that is $VQ_{comp} = 0$ in equation 8.

3 State-based Gaussian Selection

The choice of Θ in equation 7 controls the average size of the Gaussian short-list. Efficiency improves with reductions in Θ because fewer Gaussian likelihoods have to be computed during recognition. However, there is a trade-off with the recognition accuracy. For states with at least one component assigned to a selected cluster, errors can occur if some components that make a significant contribution to a state likelihood are not contained in that cluster. Further errors can occur due to 'state flooring' which occurs when no components from a state are assigned to the selected cluster the state likelihood is simply given a discrete approximate value. Since it is possible for an input vector to be an outlier with respect to all the component distributions of a state on the optimal path, the state flooring chosen can be crucial to maintaining accuracy.

A good cluster assignment is therefore one which assigns all, or most of, the components that contribute significantly to state likelihoods for that region of acoustic space, while assigning as few non-contributing components as possible. It is not possible to satisfy this "good" clustering requirement with the previous schemes, as no account was made of which state the component belonged to. The concept of retaining the component to state information is therefore introduced (based on preliminary work in [6]). This scheme will be referred to as *State-Based Gaussian Selection* (SBGS). SBGS allows a maximum number of components associated with each state to be defined using the following selection routine

$$G(jm) \in \nu_\phi \text{ iff } G(jm) \in \arg\min_m(n) \{D(G(jm), \mathbf{c}_\phi)\} \tag{9}$$

where $G(jm)$ is the component m of state j, $D(G(jm), \mathbf{c}_\phi)$ represents the "distance" from $G(jm)$ to the cluster centroid $\mathbf{c}_\phi$ and for this work is the same measure as used in equation 7, and $\arg\min_m(n)\{\}$ represents selecting the minimum n components according to the selected distance measure. There is also an added constraint that $D(G(jm), \mathbf{c}_\phi) \leq \Theta$.

It is preferable to assign more components to the states which are seen often in the training data and fewer to those seen less. Assuming that the components with data often assigned to a particular cluster will sit nearer to the cluster center, those states with components near to the center of cluster need to be modelled more accurately. This requirement is most easily achieved using a "multi-ring" approach. Thus, if the closest component of state j to the cluster centroid $\mathbf{c}_\phi$

[1] Equation 7 shows the distance measure used for this work and the work described in [6] (termed the class-weighted distance measure). The equation given in [6] is incorrect.

is less than threshold Θ_1 then the maximum number of components in the short-list associated with that state is N_1. A second threshold Θ_2, where $\Theta > \Theta_2 > \Theta_1$, is then used to give a maximum number of components for states whose closest component falls between Θ_1 and Θ_2. This is simply described by the following algorithm for selecting the maximum number of components n in equation 9

$$
\begin{aligned}
&\text{if } (\min_m \{D(G(jm), \mathbf{c}_\phi)\} \leq \Theta_1) \text{ then} \\
&\qquad n = N_1 \\
&\text{else if } (\min_m \{D(G(jm), \mathbf{c}_\phi)\} \leq \Theta_2) \text{ then} \\
&\qquad n = N_2 \\
&\text{else} \\
&\qquad n = 0
\end{aligned}
$$

where $N_1 \geq N_2 \geq 0$, with the constraint throughout that for all selected components $D(G(jm), \mathbf{c}_\phi) \leq \Theta_2$. Thus Θ_2 controls which states are to be "floored" and Θ_1 controls which states are more accurately modelled. This is a simple two-ring approach, though of course more complex functions may be used.

4 Maximum Likelihood Gaussian Selection

4.1 Selection Process

Gaussian selection aims to select the set of Gaussians that will best model the data that will be assigned to a particular cluster during recognition. The Gaussian selection techniques described above base this selection purely on distance from the cluster centroid. This is clearly sub-optimal since it requires two crude assumptions to be made. First, the centroid is assumed to be representative of the acoustic realisations of each Gaussian associated with that cluster. However, since Gaussians from many different phonetic contexts will usually be assigned to the same cluster, the associated data frames may differ substantially. Thus, the cluster centroid may be a poor representation of the data from a particular context. Second, the choice of Gaussians takes no account of their ability to model the data. For example the actual distribution may be bimodal, with one peak near the center of the cluster and the second further out. Simply choosing the closest components to the cluster center will not well model the data in this case.

To overcome these problems an alternative scheme, *Maximum Likelihood Gaussian Selection* (MLGS) is proposed. The aim of this scheme is to select the set of Gaussians that minimise the difference in likelihood of the training data between the standard and Gaussian selected systems[2]. Thus, the Gaussian short-list for state j and cluster χ_ϕ, $\hat{\nu}_\phi$, is selected according to

$$
\hat{\nu}_\phi = \arg\min_{\nu_\phi} \left\{ \sum_{t=1}^{T} \gamma_j(t) p(\chi_\phi|\mathbf{o}(t)) \log \left(\sum_{m=1}^{M_j} w_{jm} \mathcal{N}(\mathbf{o}(t); \mu_{jm}, \mathbf{\Sigma}_{jm}) \right) - \sum_{t=1}^{T} \gamma_j(t) p(\chi_\phi|\mathbf{o}(t)) \log \left(\sum_{G(jm) \in \nu_\phi} w_{jm} \mathcal{N}(\mathbf{o}(t); \mu_{jm}, \mathbf{\Sigma}_{jm}) \right) \right\} \quad (10)
$$

where M_j is the number of components in state j and

$$
\gamma_j(t) = p(q_j(t)|\mathbf{O}_T) \quad (11)
$$

where $q_j(t)$ denotes the occupation of state j at time t. This probability is computed from the system with no Gaussian selection. Hence the basic criterion for selecting components for the short-list is to choose the set that minimises the loss in likelihood between the standard no Gaussian selection system and the new Gaussian selection system with the assumption that the frame-state alignments do not significantly alter.

[2] These are not really likelihoods, but auxiliary function values. It is therefore assumed for this work that the frame-state alignment will be the same for both the full and Gaussian selected scheme.

Given that components may only be left out, the Gaussian short-list that maximises the probability of the training data, will be the one that minimises equation 10[3]. To find the "true" best set of components requires checking through all possible combinations of the required number of Gaussians. The ordering of the groups of components is then based on the likelihood scores. This may be done, but for many systems quickly becomes impractical[4]. Appendix A describes an algorithm which closely approximates this desired scheme.

For practical applications, however, a simpler implementation strategy is desirable. If it is assumed that for a state-cluster pairing, the associated data is well modelled by the n components from that state with the highest likelihood, then the selection process becomes

$$G(jm) \in \nu_\phi \text{ iff } G(jm) \in \arg\max_m(n) \left\{ \sum_{t=1}^{T} \gamma_j(t) p(\chi_\phi|\mathbf{o}(t)) \log\left(w_{jm}\mathcal{N}(\mathbf{o}(t);\mu_{jm},\mathbf{\Sigma}_{jm})\right) \right\} \quad (12)$$

Although this approximate scheme is only guaranteed to yield the maximum likelihood solution when a single component is to be selected, it is computationally very efficient. It also requires relatively little memory, since it is only necessary to store the probability of the state-cluster pairing for each component-cluster pairing. It is this implementation of maximum likelihood Gaussian selection that is used in this work and is referred to as MLGS.

A simple alternative scheme, not based on likelihoods, which attempts to improve the modelling of the data from the state-cluster pairing is to use the occupancy counts. Here

$$G(jm) \in \nu_\phi \text{ iff } G(jm) \in \arg\max_m(n) \left\{ \sum_{t=1}^{T} \gamma_{jm}(t) p(\chi_\phi|\mathbf{o}(t)) \right\} \quad (13)$$

where

$$\gamma_{jm}(t) = p(q_{jm}(t)|\mathbf{o}_T) \quad (14)$$

and $q_{jm}(t)$ indicates "being in" component m of state j at time t. Again this probability is calculated using the standard, no Gaussian selection, system. This selects the set of components with the greatest occupancy for that cluster-state pairing. The example of bimodal distributions is therefore well handled as the components "nearest" the two maxima should have higher occupancy counts. Unfortunately in this scheme the data from components with low occupancy counts is completely ignored when selecting the components for the short-list. This will be referred to as *Occupancy Gaussian Selection* (OGS).

4.2 Handling Limited Training Data

It has so far been implicitly assumed that there is an infinite amount of training data. Thus all state-cluster pairs that are ever feasible will have non-zero occupancies. Of course in reality this is not the case. For most large vocabulary systems the number of components and states is selected to make best use of the available training data. It is therefore very unlikely that all feasible pairings will be observed. However what can be stated is that the most likely pairings are liable to occur in the training data.

In most systems the vast majority of the occupancies for these state-cluster pairings will be zero. Additionally many of these pairings will have very low counts making them very poor estimates of the true[5] data associated with the state-cluster pairing. Hence basing a selection process on such state-cluster pairings may result in a "poor" short-list. However the problem of robust estimates is a standard one in speech recognition and many of the standard techniques used for robustly building large vocabulary systems may be used here. Appendix B describes how decision trees may be used for this task.

---

[3] This assumes that a well trained initial model set is used.

[4] Suppose that 4 components are to be selected from each 12 component state. In this case, there are 495 $(\frac{12!}{4!8!})$ combinations to consider for each state-cluster pairing. This also assumes that it is known a-priori the number of components required for the particular pairing.

[5] In the sense of the data associated with an infinitely large training database.

In the work presented here, however, a simple backing-off scheme is used whereby each state-cluster pairing is only used without any modification if there is "sufficient data". At this most specific level, the context-dependent level, the determination of "sufficient data" is simply based on a minimum occupancy measure, l_d. For those states exceeding this threshold the selection process is the same as equation 12. Unfortunately many state-cluster pairings will have low occupancy counts and for these pairings it is necessary to back-off to the context-independent level. If all contexts of each phone have the same HMM topology, then each state may be separately considered at the context-independent level. For MLGS the short-list selection process is then based on

$$G(jm) \in \nu_\phi \text{ iff } G(jm) \in \arg\max_m(n) \left\{ \sum_{k \in \mathbf{s}(j)} \sum_{t=1}^{T} \gamma_k(t) p(\chi_\phi|\mathbf{o}(t)) \log\left(w_{jm}\mathcal{N}(\mathbf{o}(t); \mu_{jm}, \mathbf{\Sigma}_{jm})\right) \right\} \tag{15}$$

where $\mathbf{s}(j)$ indicates the list of all states associated with the same monophone as state j and in the same state position. Again a minimum occupancy threshold, l_i, may be used to determine whether there is sufficient data at this stage. If there is still insufficient data it is necessary to back-off to the cluster level. Here all data associated with a particular cluster is used in the selection process. At this stage there is little useful temporal information in the state position, so all the states are combined together. The selection process now becomes

$$G(jm) \in \nu_\phi \text{ iff } G(jm) \in \arg\max_m(n) \left\{ \sum_{t=1}^{T} p(\chi_\phi|\mathbf{o}(t)) \log\left(w_{jm}\mathcal{N}(\mathbf{o}(t); \mu_{jm}, \mathbf{\Sigma}_{jm})\right) \right\} \tag{16}$$

The short-list generated using equation 16 is the same as that which would be produced if SBGS were used with a likelihood distance measure. It is not possible to sensibly back-off any further.

This simple backing-off scheme has been described in terms of the MLGS scheme. It is not appropriate for the OGS scheme as it would be necessary to select which of the components in the state to assign the occupancy of the additional backed-off data. This may be done, using either of the assignment schemes described in appendix A, but is not considered here. In this work OGS is only used to select components at the context-dependent level, all other selection is based on the MLGS scheme.

4.3 Number of Gaussians Required

The SBGS scheme uses a varying number of Gaussians for each state depending on how "close" the nearest component for a state was to the cluster centroid. A similar scheme is used in MLGS. When considering the total likelihood of the training data, rather than one state at a time, it is logical to assign more components to commonly occurring states than less commonly occurring ones. This may be calculated more explicitly in the MLGS scheme as the state-cluster occupancy counts may be calculated. So now the number of components may be directly related to the state-cluster occupancies. However there is a problem with this approach. For state-component pairings that seldom occur the occupancy counts will be very "noisy" and as such a poor measure of which states to "floor" completely. For the experiments carried out here state flooring was determined simply by how "close" a Gaussian was to the cluster centroid.

For the MLGS scheme with backing-off described here the maximum number of Gaussians was determined in the following way. For each state-cluster pairing, state j and cluster χ_ϕ,

$$\begin{aligned}
&\text{if } \left(\textstyle\sum_{t=1}^{T} \gamma_j(t) p(\chi_\phi|\mathbf{o}(t)) > l_d\right) \text{ then} \\
&\qquad n = N_1 \\
&\text{else if } \left(\textstyle\sum_{k \in \mathbf{s}(j)} \sum_{t=1}^{T} \gamma_k(t) p(\chi_\phi|\mathbf{o}(t)) > l_i\right) \text{ then} \\
&\qquad n = N_2 \\
&\text{else if } \left(\min_m \{D(G(jm), \mathbf{c}_\phi)\} \leq \Theta\right) \\
&\qquad n = N_3 \\
&\text{else} \\
&\qquad n = 0
\end{aligned}$$

This is a very simple scheme for selecting the number of Gaussians required. If more complex schemes are used, such as those described in appendix A, it is possible to explicitly calculate the contribution of each component to the likelihood score of the training data.

In common with all other GS schemes, there is the question of how to deal with the floored states and, to a lesser extent, the floored components. For all the experiments carried out in this paper a simple fixed value was used for all the floored components. To obtain the contribution of each floored component to the state likelihood, this value is scaled by the component weight. Hence the likelihood of a floored state is the same fixed value. The floor value used in this work was not optimised for the specific task chosen.

5 Experiments and Results

5.1 System Descriptions

The HTK standard large vocabulary speech recognition system was used to compare the performance of the various Gaussian selection schemes. This was configured as a gender-independent cross-word-triphone mixture-Gaussian tied-state HMM system identical to the "HMM-1" model set used in the HTK 1994 ARPA evaluation system [11]. The acoustic training data consisted of 36493 sentences from the SI-284 WSJ0 and WSJ1 sets, and the LIMSI 1993 WSJ lexicon and phone set were used. The system was trained using decision-tree-based state-clustering [12] to define 6399 speech states. A 12 component mixture Gaussian distribution was then trained for each tied state to give a total of about 6 million parameters. The speech data was parameterised into 12 MFCCs, C_1 to C_{12}, along with normalised log-energy and the first and second differentials of these parameters. This yielded a 39-dimensional feature vector to which cepstral mean normalisation was applied.

The data used for evaluation were the ARPA 1994 H1 development and evaluation test sets. The 1994 ARPA H1 task is an unlimited vocabulary task recorded in a clean[6] environment. A 65k word list and dictionary were used with a trigram language model as described in [11]. All decoding used a dynamic-network decoder [9] which can either operate in a single-pass or rescore pre-computed word lattices. For efficiency reasons, the results presented here are based on lattice rescores. This means that the percentage of Gaussians calculated is higher than would be the case if a full recognition run was used.

5.2 Results

Selection scheme	Thresholds				Average Likelihood (change per frame)
	Θ_1	N_1	Θ_2	N_2	
—	—	—	—	—	0.000
SGS	1.9	—	—	—	-0.021
SBGS	1.3	5	1.9	1	-0.826
	1.3	4	1.9	1	-1.125

Table 1: ARPA 1994 H1 task forced alignment likelihood change on the development data with 512 codewords

Initially performance was assessed by calculating the forced alignment likelihoods using the recognition from the standard system[7]. This allows the measure in equation 10 to be directly calculated.

[6] Here the term "clean" refers to the training and test conditions being from the same microphone type with a high signal-to-noise ratio.

[7] Strictly the state boundaries positions should be fixed in the alignment task. For the results in this work the state boundaries are allowed to move, so the figures given will be a slight overestimate of the "true" numbers. This is felt to be a second-order affect and is not expected to alter the relative values significantly.

Table 1 shows the results for SGS and SBGS schemes. The top line represents the performance of the standard no Gaussian selection system. Using SGS there is little degradation in likelihood score. This indicates that the use of tail thresholding schemes yield a good choice of Gaussians. Using SBGS reduced the likelihood score.

Selection scheme	Thresholds				H1 Dev		H1 Eval	
	Θ_1	N_1	Θ_2	N_2	%Gauss	%Error	%Gauss	%Error
—	—	—	—	—	100.0	9.57	100.0	9.20
SGS	1.9	—	—	—	36.0	9.72	35.6	9.27
SBGS	1.3	4	1.9	1	15.9	9.95	15.8	9.85
	1.3	5	1.9	1	17.6	9.79	17.4	9.58

Table 2: ARPA 1994 H1 Task using SBGS recognition performance

The figures in table 1 indicate that SBGS removes some of the Gaussians that should be calculated, but gives no indication of how this affects performance. Table 2 shows recognition rate and percentage of Gaussians calculated. As expected the SGS scheme showed little degradation in performance, however 36% of the Gaussians were required to be calculated. Despite the drop in likelihood score, the degradation in recognition performance for the SBGS schemes was small. On the evaluation task using a two ring approach with 5 components in the inner ring, calculating only 17.4% of the Gaussians resulted in only a slight degradation of 3.3% in relative performance.

Codewords Φ	Thresh.		%States Assign.		
	l_d	l_i	$\lambda^{(d)}$	$\lambda^{(i)}$	$\lambda^{(\phi)}$
512	10.0	50.0	11.9	45.0	43.1
	20.0	50.0	8.2	48.4	43.4
	50.0	50.0	4.6	51.9	43.5

Table 3: ARPA 1994 H1 Task percentage states assigned at each level, $\lambda^{(d)}$ is the context dependent level, $\lambda^{(i)}$ is the context independent level and $\lambda^{(\phi)}$ is cluster level, with $\Theta = 1.9$

Before presenting recognition results for MLGS, it is interesting to note the affects that the various thresholds have on the number of Gaussians selected. Table 3 shows the percentages of states assigned to each group according to the value of l_d and l_i. Naturally as l_d increases the percentage assigned at the context dependent level decreased, this is approximately equivalent to reducing Θ.

Selection scheme	Max. Cmpts.			Average Likelihood
	N_1	N_2	N_3	(change per frame)
MLGS ($l_d = 10$)	12	2	1	-0.241
	6	2	1	-0.621
	4	2	1	-1.048
MLGS ($l_d = 20$)	6	2	1	-0.660
	6	1	1	-0.785
MLGS ($l_d = 50$)	6	2	1	-0.746
OGS ($l_d = 20$)	6	2	1	-0.555

Table 4: ARPA 1994 H1 task forced alignment probabilities on the development data for MLGS and OGS with 512 codewords and $l_i = 50$

As before the performance of the MLGS scheme was first assessed in terms of the likelihood reduction when forced aligning to the recognition transcriptions of the standard system. These

are illustrated in table 4. At the low occupancy count of $l_d = 10$ there is little degradation in likelihood for the 12 component case. However reducing the number of components does affect the likelihood score. It is interesting to compare the MLGS scheme with the OGS scheme. At the same thresholds there was a smaller reduction in likelihood using OGS than MLGS. This indicates that the simple decision rule used in MLGS is sub-optimal. Unfortunately the OGS scheme is only applicable prior to any backing-off, though this does indicate that the use of a more complex selection process such as that described in appendix A may be beneficial.

Selection scheme	Thresh.		H1 Dev		H1 Eval	
	l_d	l_i	%Gauss	%Error	%Gauss	%Error
MLGS ($N_2 = 2$)	10.0	50.0	15.4	9.78	15.2	9.35
	20.0	50.0	14.1	9.84	13.9	9.46
	50.0	50.0	12.6	10.11	12.5	9.41
MLGS ($N_2 = 1$)	20.0	50.0	11.4	10.02	11.2	9.49
OGS ($N_2 = 2$)	20.0	50.0	14.1	9.78	13.9	9.56

Table 5: ARPA 1994 H1 Task using MLGS with 512 codewords $N_1 = 6$, $N_3 = 1$ and $\Theta = 1.9$

From table 4 reducing the value of N_1 from 6 to 4 seriously affected the likelihood. N_1 was therefore set to 6 and various parameters investigated in terms of percentage of Gaussians calculated and word error rates. The results are shown in table 5. Comparing these performance figures with those in table 2 shows that similar performance may be obtained at a reduced number of Gaussians calculated. For example on the evaluation data a performance degradation of only around 3% may be achieved with under 14% of the Gaussians calculated, compared to over 17% with SBGS. It is also worth noting the performance when $N_2 = 1$ where the percentage of Gaussians calculated is reduced by around 20%. This is not surprising given that around 50% of the states (see table 3) are assigned to this context independent grouping. Hence reducing the components for this group significantly reduces the number of Gaussians, but at some cost to the recognition performance, the development data results are degraded by around 5% (or 2% relative to the $N_2 = 2$ case).

It is worth pointing out that the MLGS approach requires the setting of more variables in training than either SGS or SBGS. However, MLGS has the advantage that the choice of thresholds may be explicitly related to values measured from the training data. For the other schemes, less explicit methods have to be used.

Selection scheme	H1 Dev		H1 Eval	
	%Gauss	%Error	%Gauss	%Error
—	100.0	9.57	100.0	9.20
SGS	36.0	9.72	35.6	9.27
SBGS	17.6	9.79	17.4	9.58
MLGS	14.1	9.84	13.9	9.46
OGS	14.1	9.78	13.9	9.56

Table 6: Summary of comparative performance for the various selection schemes on the ARPA 1994 H1 Task

Finally, table 6 shows the performance of a series of comparable Gaussian selection schemes. All the state-based Gaussian schemes show significant reductions in the percentage of Gaussians calculated compared to the SGS scheme. The performance of the MLGS and OGS schemes show further improvements compared to the SBGS scheme.

6 Conclusions

This paper has considered the problem of Gaussian selection. In particular it has addressed the problem of, having split the acoustic space into a set of clusters, how to select the "best" short-list of Gaussians to associate with each cluster. Various new selection schemes have been introduced based on the use of state information, specifically which state each of the components belongs to. First a simple extension of standard Gaussian selection was described, where a maximum number of components per-state constraint was applied. Then the more complex MLGS scheme was presented based on selecting Gaussians such that the likelihood of the training data is maximised. Techniques for handling limited training data were also described. The technique using state information were compared with the standard Gaussian selection scheme on a large vocabulary task, the 1994 ARPA Hub 1 task. On this task a substantial reduction in the computation of Gaussians was achieved. Despite the fact that lattice rescores were used, thus yielding a higher percentage of Gaussians calculated than if a full system was run, less than 15% of the Gaussians were required to be calculated for little degradation in performance.

In this paper only the simpler versions of maximum likelihood Gaussian selection have been examined. Future work will use decision trees to handle limited training data and more complex selection schemes.

7 Acknowledgements

Mark Gales is supported by a Research Fellowship from Emmanuel College, Cambridge. Kate Knill was funded by Hewlett-Packard Labs, Bristol, UK.

A A More Accurate Algorithm for MLGS

As described in section 4.1 the aim of MLGS is to select the Gaussians for the short-list associated with state j and cluster χ_ϕ according to

$$\hat{\nu}_\phi = \arg\max_{\nu_\phi} \left\{ \sum_{t=1}^{T} \gamma_j(t) p(\chi_\phi|\mathbf{o}(t)) \log \left(\sum_{G(jm)\in\nu_\phi} w_{jm} \mathcal{N}(\mathbf{o}(t); \mu_{jm}, \mathbf{\Sigma}_{jm}) \right) \right\} \tag{17}$$

where the maximisation is over all short-lists of the appropriate length. It is too expensive both computationally and in terms of memory requirements to investigate every possible component grouping within a state. However by using a scheme similar to the clustering scheme proposed by Chou [4] a relatively efficient selection scheme may be devised. Two additional assumptions are made. First, all the data from a particular component, according to the no Gaussian selection scheme, is re-assigned in a similar way in the system with Gaussian selection. Second the frame-state component alignment is not significantly altered. The selection process therefore becomes

$$\hat{\nu}_\phi = \arg\max_{\nu_\phi} \left\{ \sum_{G(jm)\in\nu_\phi} \sum_{t=1}^{T} \gamma_j(t) \gamma'_m(t) \log \left(w_{jm} \mathcal{N}(\mathbf{o}(t); \mu_{jm}, \mathbf{\Sigma}_{jm}) \right) \right\} \tag{18}$$

where $\gamma'_m(t)$ is the probability that the observation at time t was assigned to component m in the hypothesised short-list, subject to the previously stated constraint that all data from the same component is reassigned in a similar way.

The problem is how to assign all the data to one of the components in the short-list, i.e. find $\gamma'_m(t)$. As the data must be re-assigned on a component level, $\mathcal{D}_{jm\phi}$ will be used to represent all the data associated with component m of state j and cluster χ_ϕ, and $\gamma'_{jm\phi}(k)$ to represent the assignment of all the data from component k to component m in the short-list (see below). The selection process in equation 18 may then be rewritten as

$$\hat{\nu}_\phi = \arg\max_{\nu_\phi} \left\{ \sum_{G(jm)\in\nu_\phi} \sum_{G(jk)} \gamma'_{jm\phi}(k) \left[\mathcal{L}(\mathcal{D}_{jk\phi}; \mu_{jm}, \mathbf{\Sigma}_{jm}) + \sum_{t=1}^{T} L_{jk\phi}(t) \log(w_{jm}) \right] \right\} \tag{19}$$

where

$$\mathcal{L}(\mathcal{D}_{jk\phi};\mu_{jm},\mathbf{\Sigma}_{jm}) = -\frac{1}{2}\left[(K_{jm}+\mu_{jm}^T\mathbf{\Sigma}_{jm}^{-1}\mu_{jm})\sum_{t=1}^{T}L_{jk\phi}(t)+ \right. \tag{20}$$
$$\left. \mathrm{tr}\left(\mathbf{\Sigma}_{jm}^{-1}\sum_{t=1}^{T}L_{jk\phi}(t)\mathbf{o}(t)\mathbf{o}(t)^T\right)+\mu_{jm}^T\mathbf{\Sigma}_{jm}^{-1}\sum_{t=1}^{T}L_{jk\phi}(t)\mathbf{o}(t)\right]$$

and

$$L_{jk\phi}(t) = \gamma_{jk}(t)p(\chi_\phi|\mathbf{o}(t)) \tag{21}$$

K_{jm} is the standard normalising term associated with Gaussian $G(jm)$.

Two assignment schemes may be used. First a "hard" assignment may be used according to

$$\gamma'_{jm\phi}(k) = \begin{cases} 1, & \text{iff } G(jm) = \arg\max\limits_{G(jm)\in\nu_\phi}\left\{\mathcal{L}(\mathcal{D}_{jk\phi};\mu_{jm},\mathbf{\Sigma}_{jm})+\sum_{t=1}^{T}L_{jk\phi}(t)\log(w_{jm})\right\} \\ 0 & \text{otherwise} \end{cases} \tag{22}$$

An alternative is to use a soft assignment.

$$\gamma'_{jm\phi}(k) = \frac{\exp\left(\mathcal{L}(\mathcal{D}_{jk\phi};\mu_{jm},\mathbf{\Sigma}_{jm})+\sum_{t=1}^{T}L_{jk\phi}(t)\log(w_{jm})\right)}{\sum_{G(jm)\in\nu_\phi}\exp\left(\mathcal{L}(\mathcal{D}_{jk\phi};\mu_{jm},\mathbf{\Sigma}_{jm})+\sum_{t=1}^{T}L_{jk\phi}(t)\log(w_{jm})\right)} \tag{23}$$

This is analogous to Baum-Welch re-estimation in contrast to Viterbi re-estimation. Both these schemes re-assign data in an optimal ML sense.

B Handling Limited Training Data with Decision Trees

There are a variety of standard schemes to ensure that HMM parameters are robustly estimated. Many of these schemes may be applied to ensure robust short-list generation for Gaussian selection.

One currently popular scheme for robustly estimating parameters in speech recognition is to use a decision tree approach [1, 12]. This allows a smooth sharing of data between the various contexts of a particular phone, or even between phones. Assuming that the speech recogniser has been built using decision trees[8], the same trees may be used in the Gaussian selection process. To ensure the selection process is robust a minimum threshold is set. A simple top-down scheme starts the search at the root node (assuming that the root node has sufficient data) for each cluster and then:

For a node A
  if A has children B and C
    if $\sum_{Leaf_B}\sum_{t=1}^{T}\gamma_j(t)p(\chi_\phi|\mathbf{o}(t)) > Thresh$
      search through node B
    else
      let all components of subtree A be associated with all the states of subtree B
    if $\sum_{Leaf_C}\sum_{t=1}^{T}\gamma_j(t)p(\chi_\phi|\mathbf{o}(t)) > Thresh$
      search through node C
    else
      let all components of subtree A be associated with all the states of subtree C
  else
    let all components of subtree A be associated with all the states of subtree A

[8] If this is not the case a separate decision tree may be built.

where Leaf_B indicates all leaves of subtree B. Using this scheme the set of components to be associated with each state is easily derived. Depending on the selection process to be used determines how this information is to be used. For the standard MLGS scheme the following selection process is used.

$$G(jm) \in \nu_\phi \text{ iff } G(jm) \in \arg\max_m(n) \left\{ \sum_{k \in \mathbf{s}(j)} \sum_{t=1}^{T} \gamma_k(t) p(\chi_\phi | \mathbf{o}(t)) \log \left(w_{jm} \mathcal{N}(\mathbf{o}(t); \mu_{jm}, \mathbf{\Sigma}_{jm}) \right) \right\} \tag{24}$$

where $\mathbf{s}(j)$ is the list if all components associated with state j as determined by the decision tree. For the accurate MLGS scheme described in appendix A all components associated with state j must be assigned to one of the components in the hypothesised short-list, i.e. $\gamma'_{jm\phi}(k)$ is calculated for all components.

References

[1] L R Bahl, P V de Souza, P S Gopalkrishnan, D Nahamoo, and M A Picheny. Context dependent modelling of phones in continuous speech using decision trees. In *Proceedings DARPA Speech and Natural Language Processing Workshop*, pages 264–270, 1991.

[2] P. Beyerlein and M. Ullrich. Hamming distance approximation for a fast log-likelihood computation for mixture densities. In *Proc. Eurospeech*, pages 1083–1086, Madrid, 1995.

[3] E. Bocchieri. Vector quantization for efficient computation of continuous density likelihoods. In *Proc. ICASSP*, volume II, pages II–692–II–695, Minneapolis, 1993.

[4] P Chou. Optimal partitioning for classification and regression trees. *IEEE Trans PAMI*, 13:348, 1991.

[5] J. Fritsch and I. Rogina. The bucket box intersection (BBI) algorithm for fast approximate evaluation of diagonal mixture gaussians. In *Proc. ICASSP*, volume 2, pages II–273–II–276, Atlanta, 1996.

[6] K. M. Knill, M. J. F. Gales, and S.J. Young. Use of Gaussian selection in large vocabulary continuous speech recognition using HMMs. In *Proc. ICSLP*, volume 1, pages I–470–I–473, Philadelphia, 1996.

[7] Y. Linde, A. Buzo, and R. M. Gray. An algorithm for vector quantizer design. *IEEE Trans Comms*, COM-28(1):84–95, Jan 1980.

[8] H. Murveit, P. Monaco, V. Digalakis, and J. Butzberger. Techniques to achieve an accurate real-time large-vocabulary speech recognition system. In *Proc. ARPA Workshop on Human Language Technology*, pages 368–373, Plainsboro, N.J., Mar 1994.

[9] J J Odell, V Valtchev, P C Woodland, and S J Young. A one pass decoder design for large vocabulary recognition. In *Proceedings ARPA Workshop on Human Language Technology*, pages 405–410, 1994.

[10] G. Poggi. Fast algorithm for full-search VQ encoding. *Electronics Letters*, 29(12):1141–1142, June 1993.

[11] P C Woodland, J J Odell, V Valtchev, and S J Young. The development of the 1994 HTK large vocabulary speech recognition system. In *Proceedings ARPA Workshop on Spoken Language Systems Technology*, pages 104–109, 1995.

[12] S J Young, J J Odell, and P C Woodland. Tree-based state tying for high accuracy acoustic modelling. In *Proceedings ARPA Workshop on Human Language Technology*, pages 307–312, 1994.