Department of Computer Science

Technical Report

A Method to add Gaussian Mixture Models

Peter Hall and Yulia Hicks

Technical Report 2004-03 **April 2004**

ISSN 1740-9497

Contact Address:
Department of Computer Science
University of Bath
Bath, BA2 7AY
United Kingdom
URL: http://www.cs.bath.ac.uk

ISSN 1740-9497

A method to add Gaussian mixture models

Peter Hall and Yulia Hicks *†

April 5, 2004

Abstract

We provide a method for the addition of Gaussian Mixture Models (GMMs), so enabling GMMs to be incrementally learned over time. Our method does not require past data, neither does it resample, but operates on parameters used to describe the input GMMs; we provide an information theoretic criterion to automically recommend the number of components in the result. Empirical evidence illustrates the performance of our method, and demonstrate its utility with a taxonomic application. We conclude that the contributions of our method are (1) to provide a framework of greater generality for incremental GMM than currently available, uniquely allowing as we do "forgetting", and (2) the automated choice for the optimal number of components without reference to original data.

Keywords

Unsupervised data analysis, Gaussian mixture models,
Incremental learning, Optimal number of components.

1 Introduction

This paper describes a method to "add" a pair of Gaussian mixture models (GMMs) to produce a third, without reference to sample data. To make matters more concrete we write $G[\boldsymbol{X}]$, where $\boldsymbol{X}$ is a data set and $G[.]$ is the associated GMM constructed from that data. By "the addition of GMMs" we mean to combine a pair of operand GMM to produce a third GMM which closely approximates that which would be constructed by a standard algorithm having the concatenation of data sets as input. That is, we specify an operator $\oplus$ for which

$$G[\boldsymbol{X}] \oplus G[\boldsymbol{Y}] \approx G[\boldsymbol{X} : \boldsymbol{Y}] \tag{1}$$

In which : is the concatenation operator. Our contributions are (1) a framework for addition that is more general than described elsewhere, and (2) a method for atomically selecting the optimal number of Gaussian components in the result.

Addition operators furnish on-line learning methods, allowing data models to respond as new data arrives; that is, they provide an incremental learning ability. This is in contrast with *ab initio* or *batch* methods which learn just once. Despite the fact that *ab initio* methods tacitly assume training data to be complete and static, they are often favoured over incremental methods for several reasons, amongst which are: the difficulty of validating an incremental model (the distinction between training data and test data is blurred); the model obtained can depend on the temporal order in which update data appears; and the problem of "over-learning" in which models gradually become too general and therefore of reduced discriminative power.

Our method for the addition of GMMs is unique amongst other such methods in that it allows "forgetting", which combats over-learning. To do this we are able to "subtract" one Gaussian component from another during the course of addition; this has the same effect as removing data points from a component. It is in this respect we claim a framework which is uniquely general. Empirical testing has shown that our method yields a GMM which can be expected to converge to some underlying population distribution, independent of temporal data order. This testing used a distance measure between GMMs that can be computed in closed form, and so we do not confuse training data and test data.

We assume each operand GMM ($G[X]$ and $G[Y]$)is a good representation of the data used to build it, and that these data are drawn from the same underlying population distribution of which $[X : Y]$

*Peter Hall: Department of Computer Science, Univeristy of Bath
†Yulia Hicks, Department of Computer Science, Univeristy of Cardiff

is representative subset. We do not require that data used to build an operand GMM are uniformly drawn from the population distribution; each operand GMM could contain data generated from a single component only. Indeed, the case of non-uniform data distribution is a strong motivator for incremental methods.

Incremental GMM is useful whenever training data is insufficient. A typical symptom would be that a significant fraction of input data is poorly classified by the current GMM. Insufficiency may be forced through pragmatic constraints — machine capacity may force the exclusion of some data. Alternatively a suitable range of data was unavailable at the time the original GMM was constructed: for example, taxonomic data for a yet unknown species does not exist. We can also imagine scenarios in which insufficiency means the GMM lack specificity. For example, a GMM might be constructed for a recognition task of some kind, typically each component of the GMM represents an average (an average face, say, to be specific).

Incremental GMM is beginning to be used in applications, the greater majority coming from the speech community, not least because a speaker's voice changes over time and, hence, so should the representation. Gotoh [9] updates a GMM of speech using new data as it arrives; doing in a principled manner and like us does not require previous data to be stored, but the number of Gaussian components if fixed. Lu and Zhang [12] track a speaker's voice signal by "adding" GMMs. They use a method which is very similar to our own, at least in the early stages (see 2.1). They (like us) mix operand GMMs to make a single GMM that has too many components but which models the distribution. Next (unlike us) they add components in pair-wise fashion, which reduces their total number, halting when an information-theoretic measure is not improved. This strategy is liable to the temporal orderering problem mentioned above, depicted in Figure 1. The underlying problem is a tacit assumption: that GMM components are independent, which breaks down as components cluster tightly, and start to "overlap".

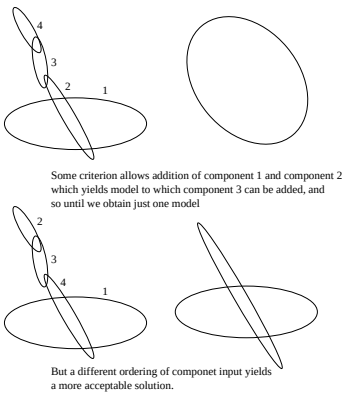

Figure 1: The ordering problem with incremental GMM; very different results can be obtain, depending on the order in which components are added.

The problem of incremental GMM has been studied in more general settings — the approach we adopt in this paper. Verbeek *et al.* [20] use a greedy algorithm to gradually increase the number of components in a GMM. In contrast, both Yang and Zwolinski [22], and Figueiredo and Jain [7] assume a GMM comprising too many components has been fitted to the data, and reduce the number by discarding "weak" components. These methods can yield considerable increase in efficiency over standard EM fitting, but all require access to past data, which is one important way we differ. Roberts *et al.* [15] use Reversible Jump Markov Chain Monte Carlo to minimise the Kullback-Liebler measure between source and target distributions; the target comprising a weighted sum of component posteriors in the source; the form of the kernels is unspecified but includes Gaussians, raw data is not required, but the number of underlying components remains constant.

The work that resembles our own most closely is due to Vasconcelos and Lippman [19], who use EM to find a set of weights (probabilities) used to combine N components into $M < N$ components. One important difference is that Vasconcelos and Lippman assume independence between GMM components. We have already observed this can lead to sub-optimal output, and this tendency is especially manifest when components "overlap". We have found it desirable to lift this assumption and thereby recognise that dependency does exist between components. To do so we need weights which can rise above unity and fall below zero. The negative weights can be interpreted as the "forgetting" we mentioned earlier, because they have the effect of removing points from a component An alternative interpreation is that of "inhibition". Weights abouve unity suggest a kind of re-enforcement. Whatever the interpretation,

it remains true we need weights beyong the $[0, 1]$ interval. A second important difference is that we provide a mechanism for determining the optimal number of components, requiring neither past data nor resampling.

In the next section (Section 2) we describe our approach to the addition of GMMs. Following this we present a series of experiments and applications (Section 3) designed to demonstrate the utility and limitations of our method. We conclude (Section 4) that the two most important issues when adding GMM are: (1) they way in which tightly clustered components are handled — this is where the "component independence" assumption begins to break down, and (2) the choice of the optimal number of components; we address both issues.

2 Adding a pair GMMs

We now describe how to add GMMs. We begin by writing a data set as $\boldsymbol{X} = \{\boldsymbol{x}_i : i = 1 \dots N_X\}$, N_X is the size of the data set, and each datum is n-dimensional, so $x_i \in \Re^n$. A normal distribution is of Gaussian form:

$$g(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{C}) = \frac{1}{(2\pi)^{n/2}|\boldsymbol{C}|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{C}^{-1}(\boldsymbol{x} - \boldsymbol{\mu})\right) \tag{2}$$

in which $\boldsymbol{x}$ is any point in $\Re^n$. The scalar function $g(\boldsymbol{x})$ is the likelihood of $\boldsymbol{x}$. The normalising factor in front of the Gaussian ensures that $\int_{-\infty}^{\infty} g(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{C}) = 1$.

An N-component Gaussian mixture model (GMM) comprises a parameter set $\{(\alpha_i, \boldsymbol{\mu}_i, \boldsymbol{C}_i) : i = 1 \dots N\}$, and can be constructed from the data $\boldsymbol{X}$. The GMM specifies a probability distribution given by

$$p(\boldsymbol{x}; G) = \sum_{i=1}^{N} \alpha_i g(\boldsymbol{x}; \boldsymbol{\mu}_i, \boldsymbol{C}_i) \tag{3}$$

The scalar function $p(\boldsymbol{x}; G)$ is the likelihood of $\boldsymbol{x}$ with respect to the GMM; for notational convenience we write $p(\boldsymbol{x})$. The weights, α_i are the prior probabilities associated with each GMM component. The priors are such that $\sum_{i=1}^{N} \alpha_i = 1$. The centroid (mean) of the ith component is $\boldsymbol{\mu}_i$ and $\boldsymbol{C}_i$ is its covariance matrix. A random point $\boldsymbol{x}$ can be classified as belonging to some GMM component or other, an aspect we largely ignore in this paper. We construct operand GMMs using Expectation Maximisation [6].

When adding two GMMs, under the assumptions mentioned above (that the GMM are a fair representation of the data used to construct them, and that all data in all GMMs are drawn from a single underlying population distribution) we proceed as follows. Let one operand GMM have N components, the other M. First we *concatenate* the operand GMM into a single GMM of $N + M$ components. Next we *simplify* this large GMM into one with $K \leq (N + M)$ components; this is done by computing a weighted sum of existing components. Finally we *select* a K that is optimal by appealing to an information theoretic criterion. We consider each step in turn.

2.1 Concatenation

Suppose we are given a pair of GMMs; an N-component GMM made from a data set having N_x points, and an M-component GMM constructed using a data set having N_y points. The GMMs represent the distributions $p(\boldsymbol{x})$ and $q(\boldsymbol{x})$, respectively:

$$p(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i g(\boldsymbol{x}; \boldsymbol{\mu}_i, \boldsymbol{C}_i) \tag{4}$$

$$q(\boldsymbol{x}) = \sum_{j=1}^{M} \beta_i g(\boldsymbol{x}; \boldsymbol{\nu}_j, \boldsymbol{D}_j) \tag{5}$$

We have changed the names given to the various parameters describing the second GMM for notational convenience. Now consider a weighted sum of these distributions:

$$\begin{aligned} r(\boldsymbol{x}) &= f_1 p(\boldsymbol{x}) + f_2 q(\boldsymbol{x}) \\ &= f_1 \sum_{i=1}^{N} \alpha_i g(\boldsymbol{x}; \boldsymbol{\mu}_i, \boldsymbol{C}_i) + f_2 \sum_{j=1}^{M} \beta_i g(\boldsymbol{x}; \boldsymbol{\nu}_j, \boldsymbol{D}_j) \\ &= \sum_{k=1}^{N+M} \gamma_k g(\boldsymbol{x}; \boldsymbol{m}_k, \boldsymbol{E}_k) \end{aligned} \tag{6}$$

We have *concatenated* the GMMs in the sense of concatenating the sets of their descriptions — subject to scaling the priors: $\gamma_k = f_1\alpha_i$ or $f_2\beta_j$, as appropriate. We insist on the constraint $f_1 + f_2 = 1$.

One obvious way to choose the weights is to set $f_1 = N_x/(N_x + N_y)$. In practice we found is better to premise the f on some measure of confidence, such as Minimum Description Length, which reflects the relative confidence we can place in each operand GMM (see Subsection 3.3). We note in passing that this concatenation is easily generalised to more than two operand GMMs, and that the possibility of using negative fractions remains open (we associate subtraction with "forgetting"). In any case we should ensure that $r(\boldsymbol{x}) \geq= 0$ for all $\boldsymbol{x}$.

This concatenated description of the data distribution is accurate, we assume, but possibly not useful, because there may be too many components; there may even repeated components. One possible consequence of this is a diminution of classification performance: "overlapping" components that should represent the same class may be differentiated. In any event the description is less parsimonious that it could be. Therefore we need to reduce the number of components needed, that is to *simplify* the description. A method for doing this without reference to any of the original data is now described.

2.2 Simplification

In this section we consider the problem of reducing an N component GMM to an M component GMM, $M \leq N$. We call this problem *simplification*. It is motivated by the need to simplify the result of concatenating two GMMs (subsection 2.1). Our approach to simplification depends on adding Gaussian descriptions.

We must highlight the distinction between adding a Gaussian functions, and adding descriptions of Gaussian functions. The former case will generally result in a multi-modal function, and is readily expressed in standard form (as in Equation 3). In contrast, adding descriptions of Gaussians results a new description — a single Gaussian, which is unimodal. If the Gaussians each represent the same distribution, then the single description which is their addition is to be preferred for this will represent the same distribution with fewer descriptive terms; in that sense we have simplified the description.

Suppose we wish to add N Gaussians to make M, that is to simplify and N-component GMM into an M-component GMM. Call the *description* sets G and H, respectively. Elements of G are tuples $< \alpha, \boldsymbol{\mu}, \boldsymbol{C} >$, elements of H are tuples $< \beta, \boldsymbol{\nu}, \boldsymbol{D} >$. Suppose further we are given a matrix of NM scalars w_{ij}, used to specify the contribution that the ith component from G makes to the $j$the component from H. The descriptions that result from such an addition are given by

$$\beta_j = \sum_{i=1}^{N} w_{ij}\alpha_i \tag{7}$$

$$\boldsymbol{\nu}_j = \frac{1}{\beta_j} \sum_{i=1}^{N} w_{ij}\alpha_i \boldsymbol{\mu}_i \tag{8}$$

$$\boldsymbol{D}_j = \frac{1}{\beta_j} \left(\sum_{i=1}^{N} w_{ij}\alpha_i (\boldsymbol{C}_i + \boldsymbol{\mu}\boldsymbol{\mu}^T) \right) - \boldsymbol{\nu}\boldsymbol{\nu}^T \tag{9}$$

This produces a new GMM, of $M \leq N$ components, and hence represents a simplification. This new GMM describes a new distribution that approximates that given by concatenation in Equation 6.

It is worth noting that Equations 7 to 9 ostensibly require all components to be of the same dimension. However, it is possible to consider each GMM component as an eigenmodel, with the advantage of being able to add components of differing dimension [10].

The weights w_{ij} are subject to constraints:

$$\sum_{j=1}^{N} w_{ij} = 1 \tag{10}$$

$$0 < \sum_{i=1}^{N} w_{ij}\alpha_i < 1 \tag{11}$$

the first acts to "preserve number", the second ensures valid priors in the simplified GMM. To see preservation of number, observe that we can interpret the w_{ij} as the re-distribution of a single datum, from the source components into the target. some w_{ij} can be negative, meaning that some fraction of a datum is removed from a source component — equivalent to "forgetting". This interpretation is given credence from observations when adding eigenmodels [10]. Let us explain: an eigenmodel is specified by the number of points used to make it, the mean, the eigenvectors, and eigenvalues: $\Omega = (N, \boldsymbol{\mu}, \boldsymbol{U}, \boldsymbol{\Lambda})$. It is possible to both add and subtract eigenmodels descriptions [10], so that if $\Omega_3 = \Omega_1 + \Omega_2$, then $\Omega_2 = \Omega_3 - \Omega_1$. To "add" is to concatenate data sets, to "subtract" is to remove (forget) data points. Subtraction can be made explicit by [10], but we have found we can "subtract" just by setting the number of points in an eigenmodel to be negative.

The principal difficulty associated with simplification is finding weights, w_{ij} that minimise some distance measure between two distributions: that which results from concatentation (Equation 6) (in this section the GMM with N components) and that specified by the simplified GMM (the GMM with M components). We need a suitable objective function, and a search mechanism. We consider these in turn.

We need an objective function, that is some distance measure between distributions. Such measures include the Kullback-Liebler measure used by Roberts *et al.* [15], and mutual information used by Yang and Zwolinski [22]. We use χ^2 distance because it is easily computable in closed form, as we now explain. For notational convenience the distribution specified by G is called $p(\boldsymbol{x})$ here, and that specified by H is called $q(\boldsymbol{x})$. These have exactly the same form as Equations 4 and 5, but of course represent possibley different distributions. The signed distance between the distributions at $\boldsymbol{x}$ is

$$\begin{aligned}
&d(\boldsymbol{x}; G, H) = p(\boldsymbol{x}) - q(\boldsymbol{x}) \\
&= \sum_{i=1}^{N} \alpha_i g(\boldsymbol{x}; \boldsymbol{\mu}_i, \boldsymbol{C}_i) - \sum_{j=1}^{M} \beta_j g(\boldsymbol{x}; \boldsymbol{\nu}_j, \boldsymbol{D}_j) \\
&= \sum_{j=1}^{M} \beta_j \sum_{i=1}^{N} \alpha_i g(\boldsymbol{x}; \boldsymbol{\mu}_i, \boldsymbol{C}_i) - \sum_{i=1}^{N} \alpha_i \sum_{j=1}^{M} \beta_j g(\boldsymbol{x}; \boldsymbol{\nu}_j, \boldsymbol{D}_j) \\
&= \sum_{i=1}^{N} \sum_{j=1}^{M} \alpha_i \beta_j (g(\boldsymbol{x}; \boldsymbol{\mu}_i, \boldsymbol{C}_i) - g(\boldsymbol{x}; \boldsymbol{\nu}_j, \boldsymbol{D}_j))
\end{aligned} \tag{12}$$

For ease of notation, denote the ith Gaussian component of G by g_i, and the jth Gaussian component of H by h_j; the g and h are functions of the independent variable $\boldsymbol{x}$. The difference is now written

$$d(\boldsymbol{x}; G, H) = \sum_{i=1}^{N} \sum_{j=1}^{M} \alpha_i \beta_j (g_i - h_j) \tag{13}$$

The squared difference at any point is

$$\begin{aligned}
&d^2(\boldsymbol{x}; G, H) = \sum_{i=1}^{N} \sum_{j=1}^{M} \alpha_i \beta_j (g_i - h_j) \sum_{k=1}^{N} \sum_{l=1}^{M} \alpha_k \beta_l (g_k - h_l) \\
&= \sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{k=1}^{N} \sum_{l=1}^{M} \alpha_i \beta_j \alpha_k \beta_l (g_i - h_j)(g_k - h_l) \\
&= \sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{k=1}^{N} \sum_{l=1}^{M} \alpha_i \beta_j \alpha_k \beta_l F_{ijkl}
\end{aligned} \tag{14}$$

Where $F_{ijkl} = (g_i g_k - g_i h_l - h_j g_k + h_j h_l)$. Hence the sum of squared differences, χ^2, is given by

$$\begin{aligned}
&\chi^2(G, H) \\
&= \int_{-\infty}^{\infty} \sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{k=1}^{N} \sum_{l=1}^{M} \alpha_i \beta_j \alpha_k \beta_l F_{ijkl} d\boldsymbol{x} \\
&= \sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{k=1}^{N} \sum_{l=1}^{M} \alpha_i \beta_j \alpha_k \beta_l \int_{-\infty}^{\infty} F_{ijkl} d\boldsymbol{x}
\end{aligned} \tag{15}$$

Here, we understand the integral to be over all space — a volume integral — this interpretation holds throughout this paper. Now $\int_{-\infty}^{\infty} F_{ijkl} d\boldsymbol{x} = \int_{-\infty}^{\infty} (g_i g_k - g_i h_l - h_j g_k - h_j h_l) d\boldsymbol{x}$ is the sum of four integrals of Gaussian products, each of the form

$$s = \int_{-\infty}^{\infty} g(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{C}) g(\boldsymbol{x}; \boldsymbol{\nu}, \boldsymbol{D}) d\boldsymbol{x} \tag{16}$$

This integral is computable in closed form because the product of two Gaussians is another Gaussian. To be precise

$$\begin{aligned}
& \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{C}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right) \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\nu})^T \boldsymbol{D}^{-1}(\boldsymbol{x}-\boldsymbol{\nu})\right) \\
= \quad & \exp\left(-\frac{1}{2}\gamma\right) \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{m})^T \boldsymbol{E}^{-1}(\boldsymbol{x}-\boldsymbol{m})\right)
\end{aligned} \tag{17}$$

in which

$$\boldsymbol{E} = (\boldsymbol{C}^{-1} + \boldsymbol{D}^{-1}) \tag{18}$$

$$\boldsymbol{m} = \boldsymbol{E}\left(\boldsymbol{C}^{-1}\boldsymbol{\mu} + \boldsymbol{D}^{-1}\boldsymbol{\nu}\right) \tag{19}$$

$$\gamma = (\boldsymbol{\mu}-\boldsymbol{\nu})^T (\boldsymbol{C}+\boldsymbol{D})^{-1} (\boldsymbol{\mu}-\boldsymbol{\nu}) \tag{20}$$

which can be verified by substitution. We can therefore continue evaluating the integral product, Equation 16, by substitution of the above result. This yields an integral of standard form with solution

$$s = \frac{1}{(2\pi)^{n/2}} \frac{\exp\left(-\frac{1}{2}\gamma\right)}{(|\boldsymbol{C}||\boldsymbol{D}||\boldsymbol{E}|)^{1/2}} \tag{21}$$

Hence we can compute χ^2 distance as given in Equation 15 in closed form. To strike a practical note; a naive algorithm to compute χ^2 may be of quartic complexity — a practical algorithm exploits symmetries and is of quadratic complexity.

Having specified the objective function it remains to propose an algorithm by which the weights w_{ij} are found. We have found a two stage approach to be useful. The first stage initialises the weights, and the second stage "polishes" the solution.

The set an initial value on the weights we begin by considering a fully connected graph in which each of the N components in the source GMM is a graph node, and each edge is labelled with the Chernoff bound between the linked components. The Chernoff bound provides some measure of the extent to which the two components are likely to be confused; it is 1 if they are identical, and tends to 0 as the difference between the components rises. Clearly, we do not assume independent components. We iteratively remove the edge with the smallest label in the graph, until the graph comprises M forests. The Chernoff bound from the ith source component to the newly formed jth component is set as the initial weight w_{ij}. The weights are normalised so that those with a common source sum to unity: $\sum_i w_{ij} = 1$. The components within each cluster are then added (equations 7 to 9) to give a new GMM, a first guess at the target distribution, and hence an initial distance measure. We continue searching using a simple downhill method until χ^2 is minimised.

Why use the Chernoff bound rather than χ^2 to initiate a search? We found that for well separated components almost any measure would do and little or no search was required. However, in cases where the components were close, the weights computed as described give a good starting points for a simple gradient descent search. This was especially noticeable when in a pair of "overlapping" components was "fat", and the other "thin". An explanation is that the Chernoff bound places an upper bound on mis-classification rate and so in some sense measures "overlap" more effectively than χ^2 between individual components. However, the Chernoff bound between a pair of distributions is not easy to compute, which explains why we do not use it as the basis of an objective function.

When searching we must take the constraints into account. The constraint that $\sum_i w_{ij} = 1$ over all j seems to give a problem, because it imposes a dependency between weight values. To remove this dependency observe that each row of weights lies on a $(M-1)$ dimensional hyperplane which is embedded in M dimensional space. In fact, this hyperplane is fully specified by the unit vector $\boldsymbol{u}_M = [1, \ldots 1]^T/\sqrt{(M)}$; its direction is normal to the hyperplane, and its tip specifies a point in the hyperplane. We compute the eigenvectors of $\boldsymbol{u}\boldsymbol{u}^T$ to find a set of basis vectors in the hyperplane. Each row of initial weights is projected onto that hyperplane, thus reducing the degrees of freedom for the overall problem from NM to $N(M-1)$.

Consider now the second constraint is $\forall j : \beta_j = \sum_{i=1}^{N} w_{ij}\alpha_i \in [0, 1]$. From a geometric point of view a column-vector of the weights must project onto the row-vector α in the same direction, and the sum of such projections must total unity. In N-dimensional space the constraint can be interpreted as providing the central axis of a hyper-conic of some kind. The shape of the hyper-conic is unknown to us. Each column of weights gives rise to one such hyper-conic, which must then be projected into the $N(M-1)$ dimensional search space; unfortunately we have been unable to characterise the geometry of the solution space in any greater detail. We therefore simply search over the hyperplane, which guarantees the first constraint. At each candidate point we compute the priors and check them against the second constraint. If it succeeds we continue to compute the χ_2 distance as already described, otherwise we set $\chi^2 = \infty$.

When a (locally) minimal χ^2 is found we project back into the full space and so obtain final weights w_{ij} which are used to add the components of G, and hence obtain a final simplified version of the GMM, which we call H.

2.3 Selecting the optimal number of components

We now turn our attention to selecting the optimal number of components, without appeal to either original data or resampling. Our scheme is analogous to methods of selecting the optimal number in *ab initio* GMM: several GMM which differ in the number of components are computed, the selected model has the smallest penalised log-likelihood. The difference is that we have to estimate log-likelihood from the GMM descriptors, we have no data to use.

A *penalised* log-likelihood has the general form

$$-L(\boldsymbol{X}|G) + \rho(\boldsymbol{X}, G) \tag{22}$$

in which $L(.)$ is the log-likelihood of data $\boldsymbol{X}$ given model G, and $\rho(.,.)$ is some penalty measure. Given a probability function $p(\boldsymbol{x})$, the log-likelihood is defined as

$$L(\boldsymbol{X}|p) = \sum_{i=1}^{|\boldsymbol{X}|} \log(p(\boldsymbol{x}_i)) \tag{23}$$

in our case the $p(.)$ is a GMM, and so is the form given by Equation 3. The penalty $\rho(.)$ has several proposed forms, including Minimum description length (MDL) [14], Minimum message length (MML) [21], ICOMP (also called CAICF) [3], and Akaike's information criterion (AIC) [2]. Extensive empirical studies we have carried out lead us to prefer the Bayesian method [16] for *ab initio* GMM, for it shows the least bias and greatest efficiency.

The only method we know of in the literature which does not depend on data is Fuzzy hypervolume(FHV) [8]. At first glance this looks attractive, but our studies show it to give poor results (high bias, low efficiency) and we prefer not to use it. Instead we take as our starting point the work of Smyth [18], who uses a cross-validation method to determine the optimal number of components (our experimental studies show it compares with AIC). Smyth observes that the *expectation* value of the log-likelihood *per datum* drawn from a distribution $p(\boldsymbol{x})$, with respect to a second distribution $q(\boldsymbol{x})$ is

$$E[L(\boldsymbol{X})] = \int_{-\infty}^{\infty} p(\boldsymbol{x}) \log\left(q(\boldsymbol{x})\right) d\boldsymbol{x} \tag{24}$$

$$\tag{25}$$

In our case $p(\boldsymbol{x})$ is the distribution that has been simplified, and $q(\boldsymbol{x})$ is its simplified version; recall

both are sums of Gaussians:

$$\begin{aligned} p(\boldsymbol{x}) &= \sum_{i=1}^{N} \alpha_i g(\boldsymbol{x}; \boldsymbol{\mu}_i, \boldsymbol{C}_i) \\ q(\boldsymbol{x}) &= \sum_{j=1}^{M} \beta_j g(\boldsymbol{x}; \boldsymbol{\nu}_j, \boldsymbol{D}_j) \end{aligned}$$

An obvious route to follow is to make use of the Taylor expansion for $\log(x+1)$:

$$\log(x+1) = \log(2) + \sum_{i=1}^{\infty} \frac{-1^{i+1}(x-1)^i}{i2^i}$$

valid for $x \in (-1, 3]$ [13]. This can be used to give the result is

$$\begin{aligned} E[L] = \quad & \log(2) + \\ & \sum_{i=1}^{\infty} \frac{-1^{i+1}}{i2^i} \int_{-\infty}^{\infty} p(\boldsymbol{x}) \left(q(\boldsymbol{x}) - 2\right)^i d\boldsymbol{x} \end{aligned}$$

If we now make use of the binomial expansion we obtain

$$\begin{aligned} E[L] = \quad & \log(2) + \\ & \sum_{i=1}^{\infty} \frac{-1^{i+1}}{i2^i} \sum_{j=0}^{i} \left(\begin{array}{c} i \\ j \end{array} \right) (-2)^j \int_{-\infty}^{\infty} p(\boldsymbol{x}) q(\boldsymbol{x})^{(i-j)} d\boldsymbol{x} \end{aligned}$$

each term of which, in principle, is computable in closed form as we now explain. We can compute $\int pq^n$ because we can compute $\int pq$. Let the set of Gaussian components that make $p(.)$ be denoted by G, and those making q be denoted by H. As earlier, each element of G and H is a description of a Gaussian. Now consider the Cartesian product $F_1 = G \times H$, each element in the new set, F_1 is the product of a pair of Gaussians, which by Equation 17 is just another Gaussian, the description of which we can compute in closed form. We can continue recursively and compute $F_{i+1} = F_i \times H$ in exactly the same way. Further, each element in each F_i can be integrated analytically,in closed form, because each is of standard form; the sum of all element-integrals is exactly $\int pq^n$.

Therefore, in principle, we need only compute Taylor terms until some error bound is satisfied. Unfortunately this algorithm has exponential complexity because the size of F_i is $|G||H|^i$. We have found discarding very small elements of F_i to be effective in controlling this growth. However, we also found that (1) convergence is slow so that the number of Taylor terms can be large, and (2) rounding errors can lead to divergence. Both of these problems are worse when a Gaussian has low prior, so that is has broad, flat distribution. We concluded that using Taylor expansion of log is an ineffective approach to our problem of computing $\int p \log(q)$.

Our approach is instead based on the following observation:

$$\frac{d}{d\eta} \int_{-\infty}^{\infty} p(\boldsymbol{x}) q(\boldsymbol{x})^\eta d\boldsymbol{x} = \int_{-\infty}^{\infty} p(\boldsymbol{x}) \log(q(\boldsymbol{x})) q(\boldsymbol{x})^\eta d\boldsymbol{x} \tag{26}$$

Hence

$$\frac{d}{d\eta} \int_{-\infty}^{\infty} p(\boldsymbol{x}) q(\boldsymbol{x})^\eta d\boldsymbol{x} \Big|_{\eta=0} = \int_{-\infty}^{\infty} p(\boldsymbol{x}) \log(q(\boldsymbol{x})) d\boldsymbol{x} \tag{27}$$

It would seem we could now base a solution on the expansion of $(1+x)^\eta$, and use finite-differencing to estimate the slope, but this suffers from the same problems as the log-based approach. Therefore, we fit a function that we can differentiate analytically to the first few terms of

$$f(\eta) = \int_{-\infty}^{\infty} p(\boldsymbol{x}) q(\boldsymbol{x})^\eta d\boldsymbol{x} \tag{28}$$

at integer values of η from 0 to some convenient value, typically about 40 or so terms, which can be computed on a practical basis. Next we fit a sum of exponentials:

$$y(\eta) = \sum_{i=1}^{K} a_i \exp(-b_i \eta) \tag{29}$$

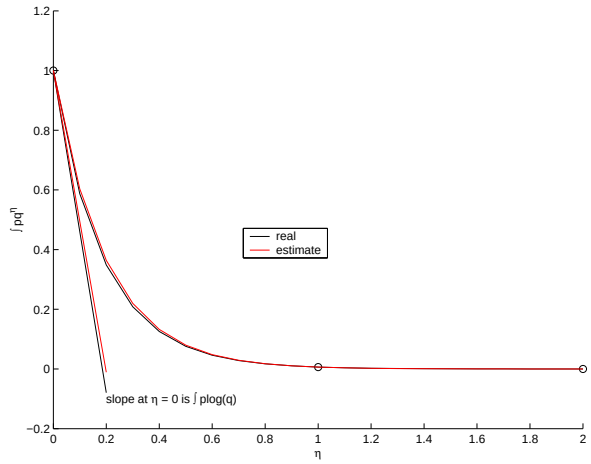

Figure 2: Showing the computation of $\int p \log(q)$: The function $f(\eta) = \int pq^\eta$ is sampled at integer values of η; such samples are shown as black circles. The solid curve shows the function at non-integer values, it was computed directly from the (2D) distributions p and q by spatial sampling at regular intervals. The dotted curve shows the fitted curve, a sum of exponentials. The $\int p \log(q)$ is the slope at $\eta = 0$, the direct (spatially sampled) value shown in black, the estimated value shown in red.

and search for the coefficients a_k and b_k so as to give a least-squares fit between the $y(\eta)$ and $f(\eta)$. It is advisable to select several random starting points, for the solution space is very "bumpy". We iterate K from 1 to 5, fitting several randomly-initiated curves for each and selecting the curve which best fits the data in the least-squares sense. The slope of the fitted function, at 0, is

$$\int p \log(q) \approx \sigma = \sum_{i=1}^{K} a_i b_i \tag{30}$$

$$\tag{31}$$

which approximates the expected log-likelihood *per datum*, and which can be computed in an acceptable time (given that small terms in the F_i are discarded). Figure 2 illustrates the process.

We are now in a position to estimate a penalty. We assume we know the total number of data points N_d that contribute to the model, and the number of free parameters N_f in the GMM, and compute

$$E[L] = \sigma N_d + \frac{1}{2} N_f \log(N_d) \tag{32}$$

we use $N_f = k(n+2)(n+1)/2 - 1$, where k is the number of components in the GMM, and n is the dimension of the space in which it is embedded. This form of penalised log-likelihood function is analogous to MDL, the Bayesian approach [16] we prefer is not available to use because its penalty term uses the data samples.

3 Experiments

We now present a series of empirical tests designed to measure the behaviour of both simplification and relative entropy estimation, which are the most important subcomponents of GMM addition. We also show empirical studies regarding GMM addition as an overall process. Unless stated otherwise, all GMMs use two-dimensional data. When we constructed a random GMM of N components we selected parameters for each component independently, sampled from these components, and fitted a GMM *ab initio*, determining the number of components, M automatically. We insisted that the distribution of samples be "separable", by which mean $M = N$. This allows components to "overlap" so much that they are effectively a single component.

The random GMM were generated by choosing random priors, means, and covariance matrices. The means were randomly distributed in a hypercube of edge d. A low value of d increased the likelihood of "overlapping" components making separation more difficult; simplification, and selecting the correct number of components became correspondingly more difficult tasks. A high value of d reverses this effect by allowing the components to spread out from one another. Thus d is a parameter that gives some indication as to the difficulty of the task. We found that $d = 10$ gave a good mix of "difficult" and "easy" problems, and so set d to this value for every experiment, unless otherwise indicated.

3.1 Efficiency, bias, and accuracy of simplification

One measure of the accuracy of simplification is the χ^2 distance between the estimated GMM and some ground truth. Alternatively, we can consider a performance measure, in our case the fraction of data whose classification data agreed with a ground-truth classification. χ is available between GMMS with differing numbers of components, but the fraction of correct classifications requires that each GMM have the same number of components.

Computing the fraction of data correctly classified (in the sense of agreeing with the ground-truth classifier) was a little more involved. We classified the original data used during construction against the ground-truth GMM, the control GMM, and the simplified GMM. To estimate the fractional agreement between a pair of GMMs we computed the number of number of points belonging to class i in one GMM, and class j in the other, in effect measuring the degree of intersection between a class pair. Class pairs with the largest intersections were deemed to correspond. Hence the fractional agreement was taken as the total fraction of points in such intersections. We estimated the fractional agreement with the control GMM against the ground-truth GMM, and also the simplified GMM against the ground-truth GMM.

We randomly sampled 1000 points from some ground truth GMM of N components. We fitted a control GMM of N components using EM. We fitted a second GMM with $2N$ components, which was simplifies to N components. We measured the χ^2 distance between each computed GMM and the ground truth. We also measured the fraction of correctly classified points between each computed GMM and the ground truth. We repeated this over 100 trials for $N = 1$ to 5 inclusive. Results from this experiment can be seen in Figure 3 and Figure 4.

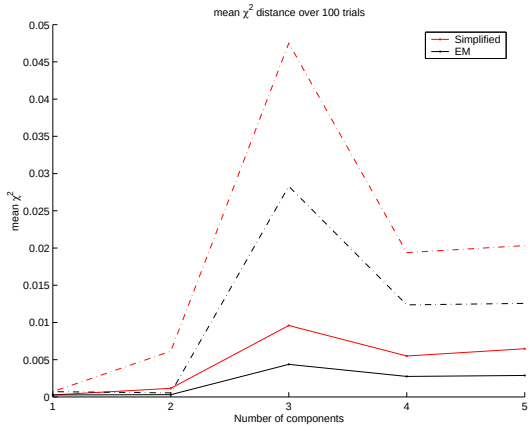

Figure 3: The mean χ^2 distance, over 100 trials, for simplified and *ab initio* GMMs. The broken lines mark one standard deviation from the mean; the mean is bounded below by the abscissa.

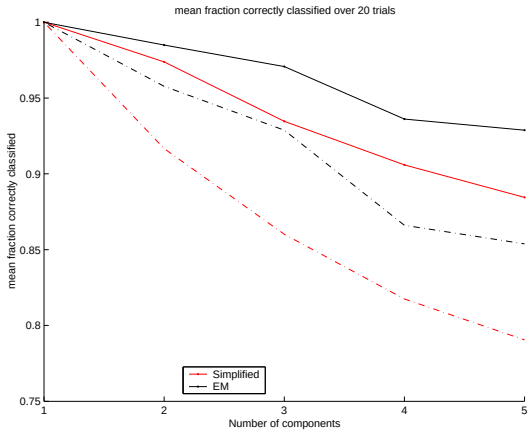

Figure 4: The fractional number of points correctly classified, over 100 trials, for simplified and *ab initio* GMMs. The broken lines mark one standard deviation from the mean. The mean is bounded below by the abscissa.

The results indicate that simplified GMM are usually further from the ground truth than *ab initio*

GMM, and that the classification performance for simplified GMM is general lower than for *ab initio* GMM. Given the fact that the simplified GMM are the consequence two search processes this result is not surprising. Even so, each mean for the simplified GMM is everywhere within one standard deviation of the mean for the *ab initio* GMM, and the correct classification rate can be expected to remain above 90% for simplified GMM

3.2 The accuracy of the relative entropy estimate

The aim here is to measure the accuracy of the $\int p \log(q)$ estimate; that is, relative entropy per datum, ρ. To do so we generated a pair of GMMs, one of N components at random, and a second of $M \leq N$ components, via an EM fit. We made sure that the randomly generated has separable components. We then estimated relative entropy; first by regular sampling to give a "real" value ρ_{real}, and second using the approximation given in Subsection 2.3 to give ρ_{est}. We varied the number of components, N and $M \leq N$ to lie between 1 and 5, and for each (M, N) pair we ran 10 trials and computed bias of the relative entropy estimate compared to the real relative entropy, at each (N, M) pair as

$$(\rho_{est}(N, M) - \rho_{real}(N, M))/|\rho_{real}(N, M)|$$

Finally we computed the relative error as a function of $N - M$, the difference in the number of components. Figure 5 shows that the error decreases as the number of components rise.

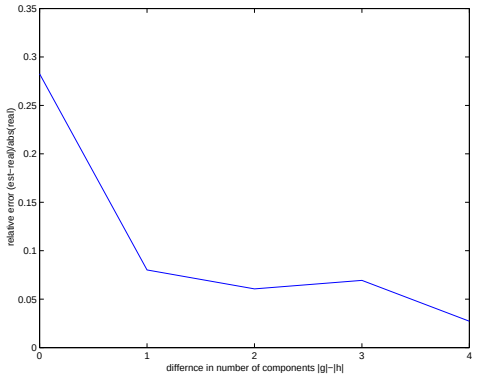

Figure 5: The relative entropy estimates error as function of the difference in the number of components.

The experiment shows that accuracy of the estimate decreases as NM rises. In practice we have found that the estimate gives a working approximation, despite the fall in accuracy.

3.3 Measures on the performance of GMM addition

GMM addition relies on both simplification and the relative entropy estimate; in this section we measure the overall process. We are interested in the bias, efficiency, and convergence properties of addition. Bias is defined as the expected difference between the ground truth number of components and the estimated number. We follow Baxter and Oliver [1] in defining efficiency as the expected minimum number of samples required to estimate the correct of components. We define convergence using the MDL of sample points: a GMM fitted with more samples should have an MDL per sample closer to the ground truth. We use MDL because the additive method uses MDL to select the number of components.

To measure bias we generated an N component ground truth GMM, which was separable, using 1000 points. We used these point to compute a control GMM, estimating the number of components automatically [16]. The data set was partitioned into 2 sets of equal size, and a GMM computed for each. This pair of GMMs was added. We performed 20 trials for each value of $N = 1 \ldots 5$. Figure 6 presents the results of the above experiment. The additive method shows the same general trend as the *ab initio* methods, and in these experiments produces a better bias as N rises.

We used an iterative procedure to determine efficiency, and from there to measure accuracy. We began by drawing 50 samples from a separable ground truth GMM of N components. At each iteration we added the new samples to an accumulating sample set (initially the empty set) and fitted an *ab initio* control GMM, determining the number of components automatically [16]. Also, at each iteration, we fitted a second *ab initio* GMM, but only using the samples from that particular iteration. This new GMM was added to an accumulating GMM (initially empty). The iteration halted when both the

accumulator GMM and control GMM had the correct number of components. Results can be seen in Figure 7

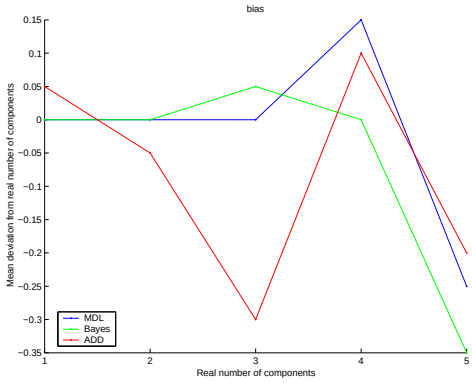

Figure 6: The bias for different numbers of ground-truth components, K; the mean of 20 trials for each K value is shown.

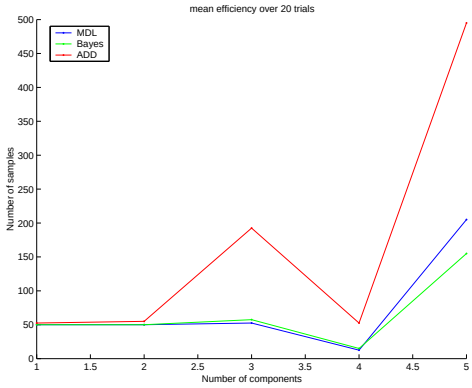

Figure 7: The efficiency of addition as a function of the number of ground-truth components.

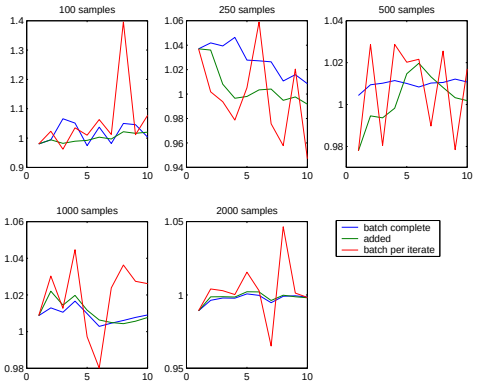

Figure 8: The progression relative MDL as a GMM is accumulated using GMMs built from 100, 250, 500, 1000, and 2000 points. In general as more points are used the "complete" and "accumulated" GMMs both tend to converge to the ideal value of 1; it is not surprising to find the "batch" version demonstrates superior convergence properties.

Finally we set out to test whether the result of accumulating GMMs tends toward the ground truth. Starting with a null accumulating GMM we iterated 10 times over the following steps: (1) randomly sample N_d points from a ground truth GMM of N components; (2) formed a new *ab initio* GMM from these samples; (3) add the "new" GMM onto the accumulator, using MDL as a basis to decided the concatenations fractions; (4) form an *ab initio* from *all* the points sampled in this and all previous iterations; (5) measure the MDL using the same data used to build the ground truth, for the

accumulated GMM, each new batch GMM, and the "complete" GMM formed at each iteration from all data.

Results in Figure 8 shows a typical set of curves for $N_d = \{100, 250, 500, 1000, 2000\}$, with a ground truth of 3 components, build from 1000 samples. The MDL measure shown is normalised by the MDL for the ground truth data (optimal "relative MDL" is 1, therefore). As mentioned these curves are typical. However, we should mention the fact that if a particularly poor GMM (that is, one that does not fairly represent the ground truth), then the system can diverge. This is because we assume from the outset (stated at the top the paper) that the operand GMMS are "fair" estimate of the population distribution, and hence so too is the result of our concatenation step. If this assumption does not hold, then we cannot give guarantees regarding a result. This is reasonable; of we input poor data we cannot expect a good result. Nonetheless, we found that using of MDL as a weight during concatenation down-plays the effect of a poor distribution, allowing us to violate our assumption to some degree.

3.4 An application in the taxonomy

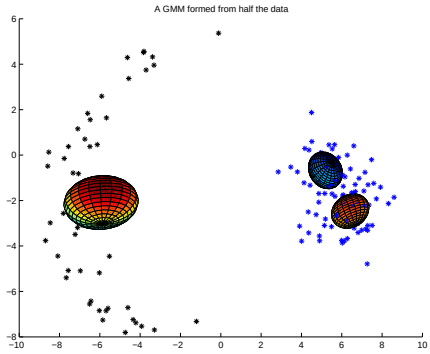

Figure 9: The data and GMM for 'taxonomist A'.

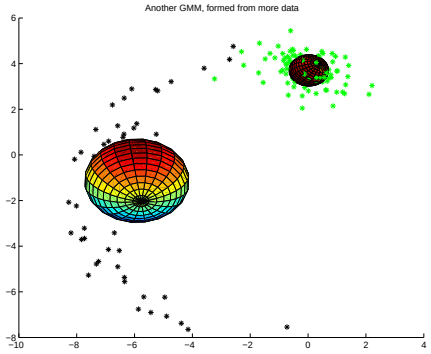

Figure 10: The data and GMM for 'taxonomist B'..

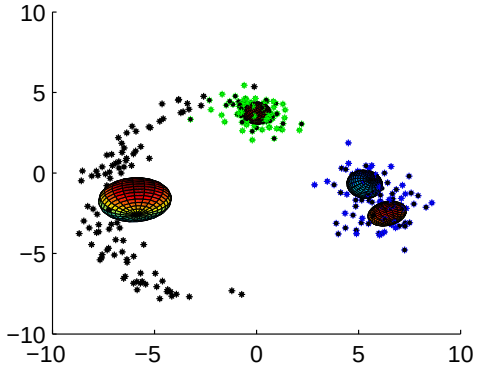

Figure 11: A control GMM made *ab initio* from the combined data.

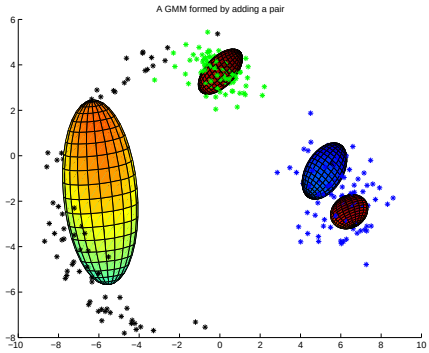

Figure 12: The result of adding the GMMs of taxonomists A and B, the raw data is shown as a reference only and is not used for the addition.

We now present a particular application that is enabled by adding GMMs. We are in collaboration with taxonomists whose interests centre on diatoms; unicellular algae that live in almost any moist or aquatic environmen [17]. The problem for us is to design a classifier, despite that fact that not all species are known. It is true that classifiers for diatoms exist [5, 11], but these do not respond to novel species. Consider the scenario in which a taxonomist is presented with two classifiers, perhaps generated by different laboratories using some existing approaches (possibley different at each lab). The evidence from the classifiers is that there exist just two species of diatom, because these seem to be two clusters of components in each GMM. The taxonomist suspects that there are, in fact, three species. By adding the GMMs and determining the number of clusters, that taxonomist can hope to decide the number of species: two, or three.

We modelled the above scenario, knowing that there were in fact three species of diatoms. Taxonomist 'A' was given 150 datum from species 'a', and 'b', taxonomist 'B' was given 150 datum from species 'b' and 'c'. Each datum was a 30-component vector, comprising Fourier descriptors to represent

the shape of a particular diatom. The only pre-processing was to use Principal Component Analysis to reduce the data to just 3 dimensions. Figures 9 and 10 show the GMMs produced by taxonomist A and B, respectively. The GMM that results from their addition is shown in Figure 12, while an *ab initio* control GMM is in Figure 11. The two "total" GMMS are in broad agreement in that both provide evidence for the existence of three species. They differ in the number of and shape of components — this might be expected as the different searches fall into local minima.

4 Conclusion

We have provided a method for the addition of GMMs, including a change in the number of components and the selection of the optimal number of components. We require no reference to data, either original or sampled, and need to supplement the GMM descriptions with the number of datum only. The result of addition is a GMM that approximates an *ab initio* GMM founded on a concatenation of the original data. We have empirically tested our method, in detail and in whole.

There are several additions we could have made to this paper. We could have presented more experiments, but those we have presented reflect salient properties: details of simplification, of the $\int p \log(q)$ estimate, and of addition as a whole. We also demonstrated an application. We could have included additional applications, but with little gain. We could also have generalised the concatenation step (Subsection 2.1), but only at the cost of extending the paper unduly: such a generalisation potentially allows for many more applications, such as biasing a general model toward a specific (individual) model, but that is another paper.

As a result of this work we conclude the most difficult, but essential, issues relevant to incremental learning of GMM are (1) coping with "overlapping" components, and (2) choosing the optimal number of components in the final model. We contibute to issue (1) by allowing weights to take on any value — subject to constraints of the final model, and to issue (2) by fitting to a one-dimensional function (and therby avoiding the need to resample in some high dimensional sapce while estimating an integral). Further work is needed in noth areas — the search for a good set of weights can be slow, and the estimate of $\int p \log(q)$ can be innacurate to the point where one finds it a little surprising it works as well as it does.

From this work it is clear that *ab initio* methods are to be preferred, where they are available. However, the addition of GMMs is possible and can give acceptable results. The addition of GMMs has many applications, including the inclusion of new classes. By generalising the relative weights of complete GMM during concatentation, we allow the possibility of subtracting one GMM from another (that is, "forgetting") as well as strongly biasing output towards any given operand GMM (useful, perhaps, for emphasising data from an individual for biometeric analysis, say). Estimating $\int p \log(q)$ is difficult but very useful, it may provide a way to estimate other distances (e.g. Kullbeck-Lieblier) without resort to expensive resampling). It is also interesting to consider whether the weights used to sum component GMMs can find use in other applications; the addition of Markov Chains, perhaps.

Acknowledgements

Thanks to David Mann, Stephen Droop, and Micha Bayer at the Royal Botanical Gardens, Edinburgh.

References

[1] R. A. Baxter and J. J. Oliver, "Finding overlapping components with mml," *Statistics and Computing*, vol. 10, no. 1, pp. 5–16, 2000.

[2] H. Bozdogan, "Determining the number of component clusters in the standard multivariate normal mixture model using model-selection criteria," TR UIC/DQM/A83-1, Quantitative methods Dept., University of Illinois, Chicago, Illinois 60680, Tech. Rep., 1983.

[3] ——, "On the information-based measure of covariance complexity and its application to the evaluation of multivariate linear models," *Communications in Statistics: Theory and Methods (A)*, vol. 19, no. 1, pp. 221–278, 1990.

[4] ——, "Mixture-model cluster analysis using model selection criteria and a new informational measure of complexity," in *Proc. of the first US/Japan Conf. on the frontiers of statistical modelling: An informational approach.* Kluwer Academic Publishers, 1994, pp. 69–113.

[5] H. D. Buf and M. Bayer, Eds., *Automatic Diatom Identification*. Singapore: World Scientific Publishing Company, 2002, vol. 51.

[6] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society*, vol. 39, no. 1, pp. 1–38, 1977.

[7] M. Figueiredo and A. K. Jain, "Unsupervised learning of finite mixture models," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 24, no. 3, pp. 381–396, March 2002.

[8] I. Gath and B. Geva, "Unsupervised optimal fuzzy clusturing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 7, pp. 773–781, 1989.

[9] Y. Gotoh, "Incremental algorithms and map estimation: Efficient hmm learning of speech signals," Ph.D. dissertation, Brown University, 1996.

[10] P. Hall, D. Marshall, and R. Martin, "Merging and splitting eigenspaces," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 9, pp. 1042–1049, September 2000.

[11] Y.A. Hicks, *et al* "Modelling life cycle related and individual shape variation in biological specimins" in *Proc. British Machine Vision Conference*, Cardiff, Wales, pp 2–5, 2002.

[12] L. Lu and H.-J. Zhang, "Real-time unsupervised speaker change detection," in *International Conference on Pattern Recognition*, 2001.

[13] P. O'Neil, *Advanced Engineering Mathematics*. Brooks Cole, 1995.

[14] J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14, pp. 465–471, 1978.

[15] S. J. Roberts, C. Holmes, and D. Denison, "Minimum-entropy data partitioning using reversible jump markov chain monte carlo," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 8, pp. 909–914, 2001.

[16] S. J. Roberts, D. Husmeier, I. Rezek, and W. Penny, "Bayesian approaches to gaussian mixture modelling," *IEEE Transactions of Pattern Analysis and Machine Intelligence*, vol. 20, no. 11, pp. 1133–1142, 1998.

[17] F. Round, *The Diatoms*. Cambridge University Press, 1990.

[18] P. Smyth, "Model selection for probabilistic clustering using cross-validated likelihood," *Statistics and Computing*, vol. 10, no. 1, pp. 63–72, 2000.

[19] N. Vasconcelos and A. Lippman, "Learning mixture hierarchies," in *Neural Information Processing Sysytem 11*, Denver, Colorado, 1998, pp. 602–608.

[20] J. Verbeek, N. Vlassis, and B. Krose, "Efficient greedy learning of Gaussian mixtures," *Pattern Recognition*, 2002.

[21] C. Wallace and D. Boulton, "An information measure for classification," *Computer Journal*, vol. 11, no. 2, pp. 195–209, 1968.

[22] Z. R. Yang and M. Zwolinski, "Mutual information theory for adaptive mixture models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 4, pp. 396–403, 2001.