Technical Report

A method to add Gaussian mixture models

Peter Hall, Yulia Hicks, Tony Robinson

Technical Report 2005-03 Feb 2005

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: Department of Computer Science, University of Bath
Yulia Hicks, Department of Computer Science, University of Cardiff
Tony Robinson: Department of Mathematical Sciences, University of Bath

Feb 2005

Abstract

We provide a method for adding Gaussian Mixture Models (GMMs), so enabling incremental learning. Our method does not require past data, but instead operates on parameters used to describe GMMs. The number of components in the result is automatically determined. Empirical evidence is used to compare our method against alternative incremental learners for quality and efficiency, and we compare to batch methods too. We illustrate the potential utility of GMM addition in two novel applications: detecting user-defined low-level features in image, and the addition of Hidden Markov Models.

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 original 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 the combination of a pair of operand GMM to produce a third GMM which closely approximates that which would be constructed by a standard algorithm for fitting GMM to data, having the concatenation of data sets as input. That is, we specify an operator $\oplus$ such that

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

in which : is the concatenation operator. The utility of this is demonstrated in this paper using real applications that depend on the ability to update some GMM model as new data becomes available.

In general, addition operators are useful because they furnish *on-line* or *incremental* learning methods, allowing models to respond as new data arrives. This contrasts with *ab initio* or *batch* methods that learn a fixed model just once from a single set of training data. Our contribution is to show that incremental acquisition of GMMs is possible based on the GMM parameters alone, including unsupervised choice of the number of components.

Batch methods have received majority attention in the literature — both in GMM fitting, and more generally. They may be favored over incremental methods for several reasons, including: the difficulty of validating an incremental model (the distinction between training data and test data is blurred); the model obtained can depend on the order of new data; and the problem of "over-learning" in which models gradually become too general and therefore of reduced discriminative power. Furthermore, compared to incremental learners, batch learners tend to be simpler to understand and implement, are usually more robust, and produce more reliable models.

Yet incremental learners do offer advantages. Humans provide compelling testimony to the power of incremental learning, for people routinely re-categorise objects and events without worrying about the arguments used to favor batch learning. This observation partly provides motivation for research into incremental learning, but practical issues are the essential driving force. We observe two *in-principle* assumptions that under-pin batch methods: (1) the sample data fairly represent the underlying population; and (2) the population is static [1].

These assumptions cannot always be justified, and when violated incremental methods must be used. An obvious example arises when the training data is so large that one's computer is overwhelmed; the ability to construct a parametric model for data partitions then to add those parametric models provides a means to build a model for all the data that would otherwise not be possible. In Section 4 we provide two example applications where incremental learning is necessary. One is the user-driven construction of a classifier to detect features of interest in color images 4.1. The second is the construction of a model of the dynamics of a walking person; the model is used for video tracking and reconstructing people. In this case we extend our addition of GMMs to support the addition of Hidden Markov Models (HMMs) 4.2.

A standard approach to unsupervised batch learning is to use Expectation Maximization (EM) [4] to fit a sequence of GMMs, each with a specified number of components. The optimal model is selected from this candidate set using some penalty function, many of which exist including Akaike's Information Criterion [1], Minimum Description Length [16], Minimum Message Length [22], and Bayesian approaches [18].

Alternatives to the EM–penalty approach exist. Verbeek *et al.* [21] use a greedy algorithm to gradually increase the number of components in a GMM. In contrast, both Yang and Zwolinski [23], and Figueiredo and Jain [6] assume a GMM comprising too many components has been fitted to the data, and reduce the number by discarding "weak" components. Such methods can yield considerable increase in efficiency over standard EM fitting, (see Appendix). The latter alternatives are of specific interest here because we too merge the components of an over-fitted model.

An obvious approach to incremental GMM is to resample the existing models. This would allow a standard approach to generating a new model, but using resampled rather than original data. This approach is attractive in that it is simple to conceive and implement, and as our experiments show, can yield high quality results (see Section 3). However, adding GMMs by manipulating the components, as we advocate here, has advantages and Vasconcelos and Lippman [20] offer two: (1) Addition requires a structure of some kind to be cast over the components in the operand GMM. Vasconcelos and Lippman use this property to build a hierarchical description of data distribution doing so "layer by layer"; our method naturally yields a hierarchy and does so more efficiently than Vasconcelos and Lippman. Furthermore, we make use of the information in this structure in support of adding HMMs (Subsection 4.2. (2) The computational expense of resampling can be very high because a large number of resample points would be required. We provide empirical evidence (see Section 3) that demonstrates this.

The problem of GMM addition has been researched by others, typically where assumption (2) above is breached. Motivated by speech drifting in time, Gotoh [8] updates a GMM of speech using new data as it arrives. The number of Gaussian components is fixed for both operands and the result, whereas we allow the number to vary. Lu and Zhang [15] track a speaker's voice signal by "adding" GMMs. They first mix operand GMMs to make a single GMM that has too many components but which models the distribution. Next they add components in pair-wise fashion, which reduces their

[1]If we give all populations a potentially temporal dependence, then we need only assumption 1

total number, halting when an information-theoretic measure is not improved. We eschew pairwise addition of components not only because it is heuristic but more importantly, because greedy algorithms often yield sub-optimal results as a matter of principle. We agree with Vasconcelos and Lippman [20], who use a weighted sum of all original components, which tends towards a globally optimal result. Vasconcelos and Lippman use EM to find a set probabilities (weights) to combine N components into $M < N$ components; we also use weights although our approach is based on deterministic graph partitioning.

The reader should note that the weights apply to component parameters rather than to the density function (surface) associated with each component in the operand GMM. This is a crucial distinction, because adding parameters leads to a unimodal density, whereas the result of adding densities can be multi-modal We now explain our approach to merging multidimensional GMMs based upon the addition of component parameters.

2 A method to add GMMs

Given a pair of GMMs, $G[\boldsymbol{X}]$ and $G[\boldsymbol{Y}]$, our aim, as specified by Equation 1, is to define an operator $\oplus$ such that given a pair of operand GMMs, $G[\boldsymbol{X}]$ and $G[\boldsymbol{Y}]$, we have $G[\boldsymbol{X}] \oplus G[\boldsymbol{Y}] \approx G[\boldsymbol{X} : \boldsymbol{Y}]$ with : being data concatenation.

In laying out our requirements and assumptions it is convenient to write $G[\boldsymbol{Z}] = G[\boldsymbol{X}] \oplus G[\boldsymbol{Y}]$, but the reader should note that $\boldsymbol{Z}$ does not refer to a concatenation of the original data $\boldsymbol{X}$ and $\boldsymbol{Y}$, rather $\boldsymbol{Z}$ should be interpreted as *virtual data* that would be expected to yield $G[\boldsymbol{X}] \oplus G[\boldsymbol{Y}]$.

Suppose $G[\boldsymbol{X}]$ has N components and $G[\boldsymbol{Y}]$ has M components. We require $G[\boldsymbol{Z}]$ to possess an optimal number of components, K. We assume that:

- The population can be modeled by a GMM.
- There is no access to original data when defining addition.
- The optimal number of components is bounded, $1 \leq K \leq M + N$.
- Each operand $G[\boldsymbol{X}]$ fairly represents some part of the population density. We say that operand, $G[\boldsymbol{X}]$ *fairly samples some part of a population* if the components of the operand are a subset of the population's components, subject to a uniform scaling of priors. This assumption is basic not just to our approach but to any approach for adding GMMs, for if violated one might justifiably argue that the operand GMM is drawn from a different population density. We return to this issue when presenting results in Section 3.

We place no restrictions on the components within the operand GMMs, $G[\boldsymbol{X}]$ and $G[\boldsymbol{Y}]$, they may or may not "overlap".

There are three main steps to our approach:

1. **Concatenate** — a model with $M+N$ components is produced by trivially combining operand GMMs into a single model.
2. **Simplify** — a GMM with fewer components from the concatenated model by merging components using a weighted summation of their parameters. In fact this step produces a *candidate set* comprising several GMMs with varying numbers of components.
3. **Select** — the GMM that best explains the distribution using the fewest number of components is selected from the candidate set and output as the result.

We next explain the details of each step, in turn.

2.1 Concatenation

Concatenation simply concatenates the descriptions of each operand GMM, subject to an initial weighting of priors, into a single GMM. The motivation behind weighting is to preserve the form of the density function. The GMMs $G[\boldsymbol{X}]$ and $G[\boldsymbol{Y}]$ represent the distributions $p(\boldsymbol{x})$ and $q(\boldsymbol{x})$, respectively:

$$p(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i g(\boldsymbol{x}; \boldsymbol{\mu}_i, \boldsymbol{C}_i) \tag{2}$$

$$q(\boldsymbol{x}) = \sum_{j=1}^{M} \beta_j g(\boldsymbol{x}; \boldsymbol{\nu}_j, \boldsymbol{D}_j) \tag{3}$$

where $g(\boldsymbol{x}; \boldsymbol{\xi}, \boldsymbol{A})$ is a multivariate Gaussian component centered at $\boldsymbol{\xi}$ and covariance $\boldsymbol{A}$; α_i and β_j are priors on the components, and sum to one.

A weighted sum of these densities gives the target density

$$r(\boldsymbol{x}) = f_1 p(\boldsymbol{x}) + f_2 q(\boldsymbol{x}) \tag{4}$$

The weights f_1 and f_2 scale the component priors to give a concatenated GMM

$$\sum_{k=1}^{N+M} \gamma_k g(\boldsymbol{x}; \boldsymbol{\phi}_k, \boldsymbol{E}_k) \tag{5}$$

in which the first N components terms, specified by γ_k, $\boldsymbol{\phi}_k$ and $\boldsymbol{E}_k$, come from $G[\boldsymbol{X}]$. In particular these γ_k are exactly the α_i scaled by f_1, while the centers and covariances are simply copied from the source GMM. Analogous comments apply to the remaining $M - N$ components that come from $G[\boldsymbol{Y}]$, except now the priors are scaled by f_2.

Provided $f_1 + f_2 = 1$ the new priors γ_k are guaranteed to sum to one. We set these weights in proportion to the number of points in each operand GMM, hence

$$f_1 = \frac{|\boldsymbol{X}|}{|\boldsymbol{X}| + |\boldsymbol{Y}|} \tag{6}$$

$$f_2 = \frac{|\boldsymbol{Y}|}{|\boldsymbol{X}| + |\boldsymbol{Y}|} \tag{7}$$

This gives greater weight to the GMM with the greater number of points, which we assume we can be more confident about.

The concatenation is easy to generalize to more than two operand GMMs. Consequently our method is capable of adding any number of GMMs at once.

2.2 Simplification

The basic idea underlying simplification is to merge the components of an input GMM to yield a result with fewer components. The input GMM is expected to be the result of a concatenation, as described just above — but this is not necessary, the input GMM could be any model that may have been over-fitted to original data.

Relevant questions are: how should components be merged? which components should be merged? and how many components should there be in the result? Here we answer the first two questions, leaving optimal choice of result to the selection step in Subsection 2.3. We merge components by combining their parametric descriptions, not by adding the density functions. Our combination requires an array of weights w_{ij} that links the ith component in the result GMM to the j component in the input GMM in the sense that it specifies the fraction of mass that the jth component contributes to the ith.

Suppose the input GMM has N_{in} components, with priors δ_j, centers $\boldsymbol{\psi}_j$, and covariance matrices $\boldsymbol{E}_j$. The output GMM has N_{out} components, with priors η_i, centers $\boldsymbol{\theta}_i$, and covariance matrices $\boldsymbol{F}_i$. For any set of weights w_{ij} we have

$$\eta_i = \sum_{j=1}^{N_{in}} w_{ij}\delta_j \tag{8}$$

$$\boldsymbol{\theta}_i = \frac{1}{W_i}\sum_{j=1}^{N_{in}} w_{ij}\delta_i\boldsymbol{\psi}_j \tag{9}$$

$$\boldsymbol{F}_i = \frac{1}{W_i}\left(\sum_{j=1}^{N_{in}} w_{ij}\delta_j(\boldsymbol{E}_j + \boldsymbol{\psi}_j\boldsymbol{\psi}_j^T)\right) - \boldsymbol{\theta}_i\boldsymbol{\theta}_i^T \tag{10}$$

for $i = 1 \ldots N_{out}$. Having now a method to merge components we must determine which components to merge, that is determine the weights w_{ij}. In the past we have searched for these weight by minimizing the χ^2 distance between the input and output GMM [12]; Vasconcelos and Lippman [20] use an EM approach in which the "E" step is exactly equivalent to Equations 8 to 10, and the "M" step updates the w_{ij}. Here we advocate a method based on graph partitioning.

We begin by building a table of size N_{in}^2 that denotes the "overlap" between each pair of components in the input GMM; each table entry t_{kl} is the Chernoff bound [5] between the kth and lth component. The Chernoff bound gives an upper bound on the probability that a point randomly drawn from one component will belong to the second component. If this value is small, the components are well separated; if high the components overlap; if unity, the components are identical.

Thresholding such a table creates an adjacency matrix, with entries $a_{kl} = 1$ only if $t_{kl} >= \mathit{thresh}$ indicating that the kth and lth components should be merged. In this way thresholding partitions the total graph (in which all components are connected) into a set of forests. Each component in a forest must be merged with at least one other component in the same forest; no two components in different forests should be merged. Hence each forest defines an equivalence class $\mathcal{C}[i]$, and number of forests determines the number of components, N_{out}, in the simplified model. The weights are now particularly easy to set:

$$v_{ij} = \begin{cases} \frac{1}{|\mathcal{C}[i]|} & \text{if input component } j \text{ belongs to } \mathcal{C}[i] \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

$$w_{ij} = \frac{v_{ij}}{\sum_{j=1}^{N_{in}} v_{ij}} \tag{12}$$

We avoid the problem of choosing any one threshold by thresholding the table, t_{kl} on every distinct value in the table; thus generating a set of output models. Because we choose many thresholds we generate a *candidate set* of possible GMM, which are naturally structured into a DAG. The problem now is to select which of the candidate set is optimal.

2.3 Selection

Batch methods often fit a sequence of models using EM, choosing the optimal model from fitted GMMs later. This choice balances the quality of fit, using the log-likelihood of data, $L[\boldsymbol{X}] = \sum_{\boldsymbol{x}\in\boldsymbol{X}} log(p(\boldsymbol{x}))$ with some penalty term to limit model complexity. These methods vary from one another through the penalty term, several variations [1, 16, 22, 18] have already been mentioned in the Introduction.

One way to decide between models without using data is to use the Fuzzy Hyper Volume (FHV) [7], computed by summing the volume of each component; if $\boldsymbol{C}$ is the covariance of a components, then $|\boldsymbol{C}|^{1/2}$ is directly proportional to its volume. To see how FHV works consider two Gaussian components that overlap, and the Gaussian component that is the result of merging them — the volume of the result may be less than the sum of volumes of original components, FHV prefers the model of the lowest overall volume. We require a method for deciding which model to choose without reference to original data, so FHV is an obvious choice. However FHV compares poorly to methods that select the optimal model based on data, as our results in Appendix show. This leads us to consider resampling to select a GMM as practical alternative.

Resampling to select a GMM is not the same resampling as a means for adding GMMs: here we only choose between models, we do not create them. Resampling to select is faster, and more likely to result in a unbiased model that fits well (see results in Section 3). Resampling to select can be used to estimate an error bound on the penalty term, whatever penalty method is used. This is useful because all penalty methods exhibit a bias toward either over-fitting or under-fitting (see Appendix) and knowledge of the error bound can be used to militate against a known bias. Assuming a penalty method that is biased to over fitting, repeated resampling is succinctly presented in algorithmic form:

1. Repeat some small number of times (we use 10 times)
2. Resample from the source GMM
3. Compute a penalty for each GMM in the simplified set
4. Compute the mean and standard deviation of penalties for each model. Keep the best mean penalty and its associated standard deviation.
5. Identify the subset of models that lie within three standard deviations of the mean.
6. Select from the subset that model with the smallest number of components
7. If more than one model remains, select that with the best mean penalty.

We now describe experiments to compare our method of adding GMMs with possible alternatives.

3 Experiments

Our experiments set out to compare alternative methods for adding GMM in terms of quality and efficiency. *Efficiency* is just the number of CPU seconds taken to add GMMs. Our experiments were conducted on an Intel Pentium 4 running at 1.7GHz, with 768Mb of physical memory. *Quality*, relative to a ground-truth, has three parts: (1) The *accuracy* of fit, measuring how closely the density function of the ground truth is fitted for which we used the earth movers distance (EMD) [19]. Accurate fits can be produced using models with too many components, and although EMD takes this into account we choose to measure bias and the second term in our assessment of GMM quality.

(2) The *bias* is defined as the difference between the number of components in the fitted model and the number of components in the ground truth. A positive bias indicates an over-fitted model, a negative bias an under-fitted model. When averaged over many trials a zero bias cannot be taken to imply that the result GMM is always correct, but rather that an over-fitting GMM is as likely as an under-fitted GMM — the actual GMM may depart radically from the ground truth. Thus we require (3) the *PCM* — the probability of a correct model. We define a GMM as correct, if its EMD is less than a threshold (we used 1/100) and it has a bias of exactly 0.

We tested six methods of adding GMMs, that can be arranged into three groups. The first group is our method for addition, comprising two "flavors" depending on whether repeated-resampling or FHV was used to select from amongst the candidate GMMs produced by simplification.

The second group uses Vasconcelos and Lippman (V&L) [20] to simplify. As previously mentioned V&L provide a method very similar to our own, which motivates he comparison. In Section 2.2 we explained that V&L use EM to find the weights w_{ij}, whereas we use deterministic graph partitioning. However, another difference is that they produce a single GMM, whereas we produce several GMMs within a hierarchy. To bring parity between their method and ours we simply use V&L several times, each time specifying a different number of components in the output GMM. (V&L are quiet about how to initialize their EM search; we use K-means based on component centers, then merge the closest components.) In this way we generate a candidate set for V&L simplification, which is then open to selection using either repeated-resampling or FHV. Thus we created two flavors of the V&L approach.

The final group of methods are the resample-and-fit methods. These rely on the simple strategy of resampling from the operand GMM, followed by standard batch fitting. A preliminary experiment (see Appendix) led us to conclude that amongst the EM-penalty methods for batch fitting, the penalty due to Robert's *et al*(RBC) [18] is to be preferred because it tends to produce the most accurate GMM, albeit with a slight positive bias. Comparison with an EM-penalty method is relevant because our method also conforms to the "select amongst generated models" paradigm, and in fact uses standard penalties associated with EM fitting to do so. We also used Figueiredo and Jain (F&J) [6]; these authors kindly donated their code. Like us, F&J simplify an over-fitted model, but their method has selection automatically built in. They require an upper limit on the number of components. Rather than choose some large upper limit we choose a smallish number (10, in fact), which is then increased in steps of 1 until a GMM with the same number of components is selected twice; the GMM with the smaller RBC penalty is retained as the result GMM. Experience shows this method produces the highest quality batch models (see Appendix).

Intuitively, the quality of GMMs produced by any given addition method is likely to depend on the quality of the GMMs that are added. In turn, these depend on the number of points used in their initial construction. Furthermore, the number of *resampled points* used in resample-and-fit methods and for model selection by repeated-resampling is also expected to impact upon the quality of output models. This same number of points will also effect efficiency. We ensured that the total number of points that went into building an initial GMM was exactly matched by the number of points used in any resampling step. The number of original points is, therefore, an important experimental parameter and is in fact our independent variable.

In summary, we set out to compare the following methods for adding GMM (1) our method with repeated-resampling selection, (2) our method with FHV selection, (3) V&L with repeated-resampling selection, (4) V&L method with FHV selection, (5) resample-and-fit using F&J, (6) resample-and-fit using EM with RBC penalty. in terms of quality (EMD, bias, probability of a correct model), and efficiency. The number of points used to build the operand GMM is a free variable.

3.1 Experimental Method

Our method was based upon a single framework that could be easily controlled by the experimenter to set different experimental environments and to emulate different real-world conditions. The idea underlying our general framework is simple: generate a GMM ; divide it parts; add the parts; compare the result with the ground-truth. Repeat this process for a set number of trials so that averages, *etc*, can be computed; we used 50 trials. The need to compare different addition methods, under different sets of conditions, requires us to be a little more formal and so we introduce the following definitions.

We generated a *ground-truth GMM*, without using sample data but which is instead selected randomly, subject to limits set by experimenter, as explained below. The ground-truth GMM was divided into *division GMMs*, again without using samples, again randomly and again subject to limits set by the experimenter (see below). Any given division GMM need not have all the components of the ground truth within it, in fact a division GMM will usually represent some restricted region of the ground-truth. However, the components of division GMM are exact copies of their counterparts within the ground-truth GMM, only the priors on the division GMM components will differ because these must sum to unity. Division GMM can be added to obtain the ground-truth exactly. In this sense the division GMM are perfect.

To better represent the GMM dealt with in practice the division GMM were sampled, to represent real data perhaps created by different laboratories, or at different times. These *original samples* were input to the batch fitting method of F&J to create *operand GMMs* of the kind one might have in a practical situation. Although each operand GMM has a corresponding division GMM it is not the case they will be identical. Operand GMMs will usually represent that fraction of the ground truth covered by the corresponding division GMM less accurately — and there is not even a guarantee that they will have the same number of components for we place no restrictions at all on the batch fitting process. We added the operand GMMs to produce a *result GMM* which was compared to the ground truth.

To allow the number of points to be a free variable, we sampled the division GMMs (and by implication a given ground-truth GMM) to obtain sets of original samples with 100, 200, 300,. . ., 900, 1000 points. In this way we generated sets of operand GMMs, each having a particular number of original samples associated with it. Consequently the result GMM also form a set that can be indexed by the number of original samples. Thus for any given ground-truth GMM obtained a sequence of measures for quality and efficiency as a function of the number of original samples. Running several trials allow useful us to estimate useful statistics such as the average EMD, bias, efficiency and (of course) the probability of producing a correct model. We ran 50 trials because previous experience demonstrated that these statistics converge quite rapidly and change little after about 30 or 40 trials.

Before presenting results we should explain the process of producing random ground-truth GMM, and random division GMM, not only because this provides relevant detail, but also because it leads to the idea of *robustness*.

3.1.1 Picking and dividing a ground-truth GMM

We randomly picked a ground truth GMM from a *problem-space* defined by the number of components $\mathcal{N}$, the *dispersion* $d = [r, dr] \in \Re^2$ and *component shape* $s = [v, dv, de] \in \Re^3$. The effect of the number of components is obvious. Dispersion specifies the expected Euclidean distance r, with standard deviation dr, between component centers and the origin — directions of centers were randomly chosen. Component shape specifies the expected Fuzzy Hyper Volume v, its standard deviation dv and the standard deviation in the ratio of lengths of eigenvectors of a component's covariance matrix, de. Component orientation was randomly selected. Priors on each component were random variates from a uniform distribution, scaled to sum to unity. The parameters of this problem-space allowed the experimenter to exercise control over the complexity of the ground truth GMM. Unless stated otherwise, all experiments in this paper were fixed at $d = [3, 1]$ and $s = [1, 0.5, 0.25]$. We restricted choice on the number of components to $\mathcal{N} = [1, 2, 3, 4, 5]$.

The ground truth was divided by distributing its components amongst a set of division GMMs. The same component could be attributed to more than one division, and any given division could possibly have a unique set of components. The only rule we enforced was that the union of components across all divisions formed a covering of the ground truth components. Unless stated otherwise, the number of division GMMs was randomly chosen from $1, 2, 3, 4, 5$ and the number of components assigned to any division was randomly chosen to lie between 1 and the number of components in the ground truth. Because each component prior is directly proportional to the number of points expected in that component, the sum of ground truth priors of components assigned to a particular division GMM is directly proportional to the number of points expected in that division. We scaled these numbers across the division GMM so that they summed to unity, to obtain *division priors*.

A specific example will help make matters concrete. Consider the a ground truth GMM comprising three components A, B, C, with priors $\alpha_1, \alpha_2, \alpha_3$ respectively. The ground truth is divided into two GMMs. The first division GMM uses components A and B, the second uses components B and C. Recall that component location and covariance are copied exactly from the ground truth into the division GMM, but the priors may change. The first division GMM is given priors $\beta_1 = \alpha_1/(\alpha_1 + \alpha_2)$ and $\beta_2 = \alpha_2/(\alpha_1 + \alpha_2)$. Likewise, the second division GMM has priors $\gamma_1 = \alpha_2/(\alpha_2 + \alpha_3)$ and $\gamma_2 = \alpha_3/(\alpha_2 + \alpha_3)$ on its components, which are identical otherwise. Because in any GMM the ith prior α_i is the fraction of points one expects to observe in the ith component, it follows that the denominators $q_1 = (\alpha_1 + \alpha_2)$ and $q_2 = (\alpha_2 + \alpha_3)$ are the expected number of points in each division GMM. Hence $p_1 = q_1/(q_1 + q_2)$ and $p_2 = q_2/(q_1 + q_2)$ are the division priors.

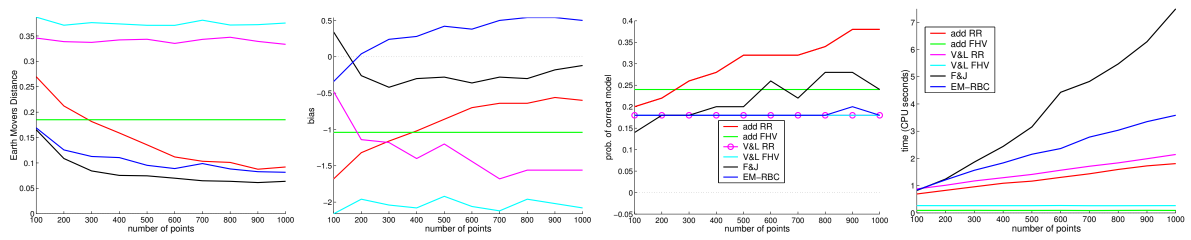

Figure 1: The ideal case in which operand GMMs and division GMMs are identical. EMD, the probability of forming the correct model, bias and time are all shown as functions of the number of points used in resampling. Each method is color coded as legends indicate — legends were omitted where they interfered with data, but colors are consistent across each plot. Note that both V&L methods coincide in terms of the probability of producing a correct model.

3.1.2 The importance of division priors and robustness

Division priors are important, as we explain through the use of a thought experiment. Consider a limiting case in which each ground-truth component is contained in exactly one division GMM. In this case the priors on the ground-truth components and the division priors are identical. Let us allow division GMM and operand GMM to be identical. Now suppose each operand GMM was independently produced, perhaps by different laboratories. In this case there is no guarantee that

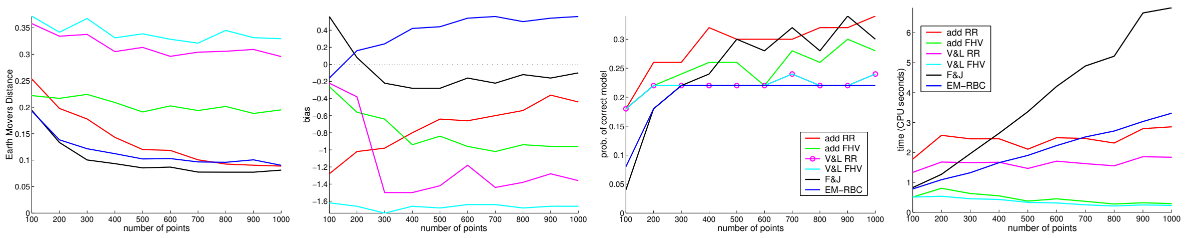

Figure 2: The case in which operand GMMs are sampled from division GMMs in direct proportion to the division priors. EMD, the probability of forming the correct model, bias and time are all shown as functions of the number of points used in resampling. Each method is color coded as legends indicate — legends were omitted where they interfered with data, but colors are consistent across each plot.

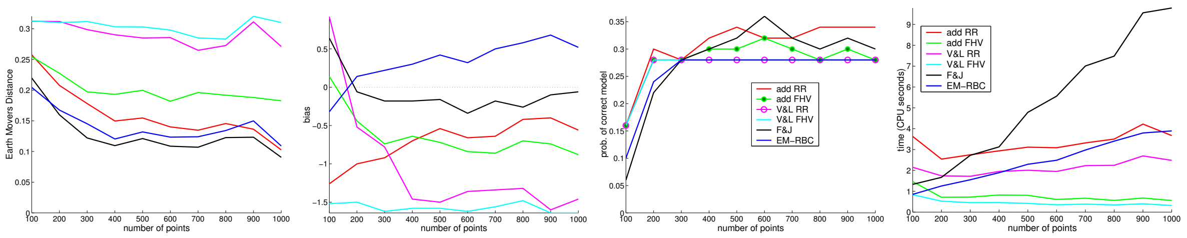

Figure 3: The case in which operand GMMs are sampled from division GMMs in direct proportion to *randomized* division priors. Color legends carry across plots. Note that both V&L methods are coincident for the probability of being correct, and that resampling-and-fit using EM-penalty is also coincident with them after 300 points.

the ratio of numbers of points used to create the operand GMM will match the ratio of the division priors. Consequently the priors in the result GMM (the added operand GMM) may not match priors in the ground-truth — so even if the result and ground-truth are identical in all other respects the density functions they model will differ.

The observation that the ratio of division priors should ideally be the ratio of the number of points in each division is true of all methods to add GMMs. As our thought experiment demonstrates, practice may depart from the ideal. Consequently *robustness* to variations in the division priors is an important issue.

Our framework enables robustness to be investigated, as follows. In normal practice we sample from division GMMs such that the ratio of numbers of points is in harmony with the ratio of division priors. We can emulate a range of envisaged practical circumstances by varying the division priors before sampling, and hence before the fitting of operand GMMs. Clearly, we require the varied division priors to sum to unity so that the total number of sample points is invariant. We "jittered" the division priors with Gaussian noise, and randomly chose them from a uniform distribution on $[0.1, 1]$.

3.2 Results

We now present results for a series of different cases to compare different methods. We begin with the "ideal" case in which operand GMMs and division GMMs are identical. Although not achievable in practice this is interesting as a limiting case. Results are shown in Figure 1. In this case the resample-and-fit methods are identical with batch methods.

Next weare cases in which the operand GMMs are batch fitting to samples drawn from the division GMMs; the number of points in each division GMM is directly proportional to the correct division prior. The actual number of points is determined by scaling by the total number of points (recall 100 to 1000, in steps of 100). Results are shown in Figure 2. In this case, too, the resample-and-fit methods are identical with batch methods.

Last is the case where the division priors were not honored when sampling division GMMs. Although we made extensive tests when "jittering" the division priors, here we present results for the only the randomly selected division priors. This is justifiable as not only are these results characteristic of other randomized results but, intuitively, randomly varied division priors represent the most demanding case. Results are presented in Figure 3.

3.3 Discussion of experimental results

We consider the EMD between the result GMM and the ground-truth. Generally this is smallest for the resample-and-fit methods, with F&J giving the best performance. Our method for addition, when repeated resampling is used for selection, is competitive, especially as the number of points rises. The EMD for these "better" methods rises slights as experimental conditions move away from the ideal. In fact, the "random" case appears to be no worse than the "in ratio" case. This suggests that dependencies that are introduced when operand GMMs overlap are effective in producing robust behavior. The surprise is that EMD for both V&L flavors falls slightly under the same transition of conditions; this is a counter-intuitive result for which we can offer no explanation.

Looking at bias, one observes that F&J offers the least bias results overall. Again our method with repeated resampling competes when the number of samples is higher, but in this can not so effectively as for EMD.

Recall that the curves presented are averaged over several trial, and that PCM is the intersection of all those models with an EMD less than 1/100 and a bias of zero. Our method with repeated-resampling consistently produces the highest probability of producing the correct model (PCM). the resample-and-fit using F&J competes as the number of samples grows, and as conditions move away from the ideal. This role reversal is best explained by supposing that our method produces a greater spread in results, and this is in fact the case, as Figure 4 shows.

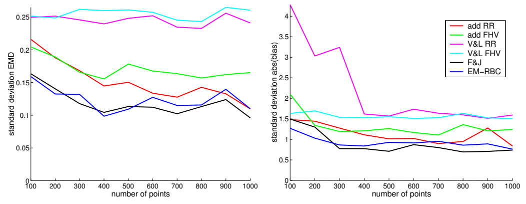

Figure 4: Standard deviation for EMD (left) and bias as a function of the number of original samples for the "random" case.

F&J is the most inefficient method, those that select using FHV are most efficient. Our method with repeated resampling is of modest efficiency. Unsurprisingly, all methods that reply on the number of original samples show a rise in time consumed as the number of samples rises. The surprise here is that EM based resample-and-fit can be more efficient that our method that relies on repeated resampling. This is possibly because EM arrives quickly at an extremity in its search space, giving it the possibility of early termination; our method iterates a fixed number of times.

3.4 Conclusions from experimental results

We conclude that when all experimental data is taken in account, our method for adding GMMs is to be preferred over the V&L approach. In particular it offers the greater probability of producing a correct model in a more efficient manner. Confidence in this conclusion is strengthened in view of the following. Dispersion, here set at $[3, 1]$ generates a mixture of difficult cases in which components overlap to great extent, and simpler cases in which components are relatively independent. When we set a larger dispersion, $[8, 1]$ we found greater advantage to our method.

Varying the division priors has less effect than might be expected. We explain this by appeal to the fact that operand GMMs often overlap, which generates dependencies between them. Note that operand GMMs could overlap even if the ground truth components are independent, as just mentioned.

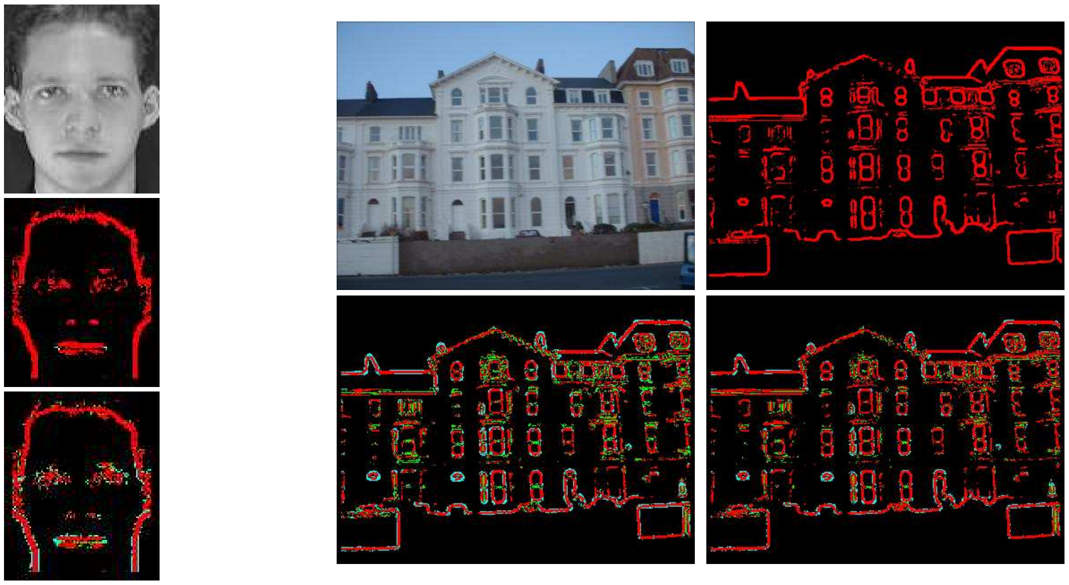

Figure 5: Incremental learning applied to low-level feature classification using an additive GMM classifier. Features to be classified are "edge" (red) "ridge" (green), "corner" (cyan), or "none" (black). Left column, from top: face picture; "face" classifier only; combined classier. Right columns, top-right to bottom-left: houses picture; houses with "face" classifer; houses with "house" classifier only; combined classifier result.

These experimental data presented were harvested for problems in $\Re^2$. However, the comparison of batch methods in the Appendix shows that quality measures change little as dimension changes. However, questions remain over time efficiency in relation to changes in dimension and in the number of components fitted. We answer these questions in a set of supplementary experiments, also in the Appendix, concluding that a change in dimension has little impact on V&L and our methods. Oddly, the time taken to fit using F&J falls as dimension rises. The effect of the number of components is more clear — a rise in the time taken to find a model.

4 Applications

Experimenting under controlled conditions is important, but equally important are the applications to which this new method can be put to use. In this section we describe two novel applications enhanced by the addition of GMMs.

4.1 User-specified classification of low-level features

We have developed a method for detecting user-specified low-level features in full color digital images. Details of the method and its applications are fully described elsewhere [9, 10]. Here we concentrate on machine learning aspects, but must briefly describe our approach because it motivates incremental learning.

Users are free to specify classes of low-level features by providing examples. The user decides how many classes and which examples to use. We chose “edge”, “corner” and “ridge” classes but others are possible, such as “t-junction”. Each example feature is enclosed within a circular window that is centered on the pixel around which the feature exists, and is of a particular scale determined by the window radius. The system computes a feature vector in $\Re^9$ for each window, the vector is orientation independent, scale independent, and robust to lighting variations (see [9, 10]). A GMM is constructed for each class, embedded in feature space of $\Re^9$. Given a new pixel, located at $\boldsymbol{x}$, which may be in a new image, this classifier assigns a posterior probability vector $\boldsymbol{p}(k|\boldsymbol{x})$; each element being the probability that the pixel is the center of a window of class k.

The problem is that it is impossible to specify a representative set of training examples in advance and therefore impossible to use batch learning to train the classifier. There are two reasons. (1) We the programmers cannot dictate to users what they wish for, and we cannot ask a nominal user to guarantee a set that is fully representative of each class. Incremental learning relieves the user of responsibility because it allows them to build up examples as they find them. (2) Any given image is unlikely to contain a fully representative set of each chosen class.

We allow users to train a classifier specifically for a given image, later adding that specific classifier into a general classifier accumulated over many images. This approach is much more comfortable for users than specifying a full set over many images before building a classifier. The obvious alternative is to keep all the examples from all images. This allows the user interface to retain its “comfortable, one image at a time” approach but is inefficient in both memory consumed to store the data, and in the time needed to process it.

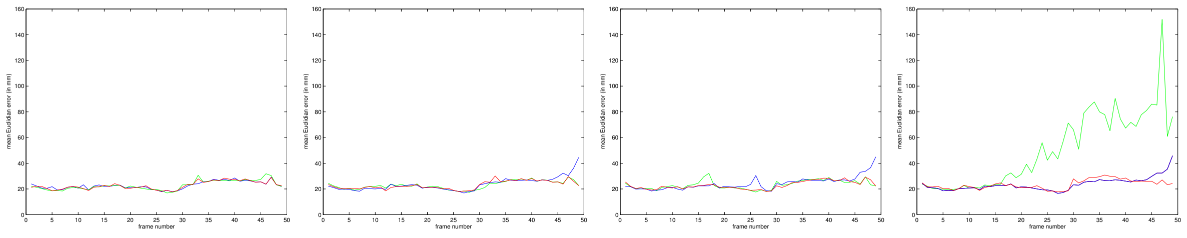

Figure 6: The mean tracking error (in mm) of the 3 models on the same video sequence. The original HMM errors are displayed in blue and green, the added HMM error is displayed in red. The errors are calculated for 256, 128, 64 and 32 samples in the modified CONDENSATION algorithm(displayed in left to right).

Figure 5 shows example results from our incremental classifier. We first trained using a monochrome face, using windows of fixed radius of 3 pixels. We picked a few example features, built an image-specific classifier using batch F&J, then classified the whole image. We repeated this, accumulating data along the way, until we were satisfied there would be no improvement in performance. We now have a "face" classifier, although we were disappointed that corners of eyes did not register as corners — there were too few example corners in the face to get a good classification of them.

We next applied the "face" classifier to a full-color image of houses. As expected not all features were satisfactorily detected. We therefore built a "house" classifier, using the same training procedures as before. We were satisfied with the performance of the house classifier. To obtain a general classifier we added the "face" and "house" classifiers. We then reclassified both images, which resulted in a much improved performance — especially in the face image where corners of eyes were now detected. This process could, of course, be iteratively applied to many images.

4.2 Adding hidden Markov models

Our approach to addition of GMMs supports the addition of HMMs [14], where each HMM state is represented by a single Gaussian distribution. This especially useful when the system under study has dynamic properties — we were motivated by the problem of tracking walking people seen in monocular video, in which case the HMM states represent distinct body "poses" and the HMM transition probabilities represent the transition probabilities from pose to pose. The walking pattern of a person may change over time, and we would like to update the HMMs as the time progresses. Methods for incremental update of HMMs have been developed in the past for speech recognition applications [8]; however, the number of the states in the HMMs had to stay fixed. Our method permits the number of the states in the HMMs to change over time and thus offers extra flexibility in the model.

Adding HMMs is possible because, fortunately, the matrix of weights w_{ij} we produce to add GMMs is exactly what is needed to add HMMs too. Suppose we have two HMMs: $\lambda_1 = \{\pi^1, \mathbf{A}^1, \mathbf{B}^1\}$ consisting of M states, and $\lambda_2 = \{\pi^2, \mathbf{A}^2, \mathbf{B}^2\}$ consisting of N states, where π^i are the initial state probability vectors, $\mathbf{A}^i$ are the state transition probability matrices, and $\mathbf{B}^i$ are the distributions representing the states. Let the result of the addition be a HMM $\lambda_3 = \{\pi^3, \mathbf{A}^3, \mathbf{B}^3\}$, consisting of K states.

To add λ_1 and λ_2 we firstly add the two underlying observation distributions $\mathbf{B}^1$ and $\mathbf{B}^2$ using our method for adding GMMs to estimate the combined distribution and then use the obtained matrix w_{ij} to estimate the new transition matrix and the new initial state probability vector.

The first stage of estimating the probability transition matrix of HMM λ_3 is to combine the two original transition matrices $\mathbf{A}^1$ of size MxM and $\mathbf{A}^2$ of size NxN into a single matrix $\mathbf{A}^c$ of size nxn, where $n = N + M$, as detailed below.

$$\mathbf{A^c} = \begin{pmatrix} a_{11} & \cdots & a_{1M} & \cdots & a_{1M+N} \\ a_{21} & \cdots & a_{2M} & \cdots & a_{2M+N} \\ \vdots & \ddots & \vdots & \ddots & \\ a_{M1} & \cdots & a_{M} & \cdots & a_{MM+N} \\ \vdots & \ddots & \vdots & \ddots & \\ a_{M+N1} & \cdots & a_{M+NM} & \cdots & a_{M+NM+N} \end{pmatrix}$$

$$\text{where } a_{ij} = \begin{cases} \{A^1\}_{ij} & \text{if } i \leq M \text{ and } j \leq M \\ \{A^2\}_{i-M,j-M} & \text{if } i > M \text{ and } j > M \\ 0 & \text{otherwise} \end{cases}$$

The elements of the probability transition matrix $\mathbf{A}^3$ are obtained using Equation 13, where α_i are the component priors, a_{ij} are the elements of the matrix $\mathbf{A}^c$ and w_{ij} are our weights.

$$\{\mathbf{A}^3\}_{ij} = \frac{\sum_{l=1}^{n}\sum_{k=1}^{n}\alpha_k w_{ki} w_{lj} a_{kl}}{\sum_{j=1}^{K}\sum_{l=1}^{n}\sum_{k=1}^{n}\alpha_k w_{ki} w_{lj} a_{kl}} \tag{13}$$

Finally, we estimate the initial state probability vector π^3 of λ_3. To obtain its values we first concatenate the two initial state probability vectors π^1 and π^2 into a single vector π^c and then update the vector elements in the following way.

$$\pi_i^3 = \frac{\sum_{k=1}^{n} w_{ki}\pi_k^c}{\sum_{i=1}^{K}\sum_{k=1}^{n} w_{ki}\pi_k^c} \tag{14}$$

We have tested our algorithm in a number of experiments, which showed that the HMMs produced by our algorithm are comparable to the HMMs trained using a batch method given all the data at once [14]. In addition, in [13] we applied our algorithm to add two HMMs trained on real human motion data and used the resulting model to track moving articulated human figure in synthetic video sequences. We showed that the model produced by our algorithm was tracking the human figure at least as well as any of the two HMMs used in the addition operation, and, under some conditions, even better. The experiments are described in detail below.

The training data set represents 3-dimensional articulated walking human motion consisting of approximately 900 51-dimensional vectors obtained using an optical motion capture system, where each vector represents the (x,y,z) positions of the 17 markers placed on a human body during the motion capture process. The origin of the coordinate system was associated with the body of a person. Before performing any experiments, we reduced the dimensionality of the eigenspace of all data from 51 to 3 to reduce the running time of the experiments. The remaining 3 dimensions accounted for over 95% of the total eigenenergy. We divided the data representing a number of cycles of walking motion performed by the same person into two data sets, trained a HMM on each of the datasets and then added the two HMMs using our method. The number of clusters in each of the HMMs was chosen as 12 rather than automatically to reduce the running time.

The resulting HMM had 17 clusters, hence many of the original clusters merged since the data was strongly overlapping However, since we chose the number of the clusters as 12 for the original HMMs ourselves rather than automatically, a larger number of clusters (17) represents the underlying distribution better than the 12 clusters chosen by us. To prove this, we measured the likelihood of each of the HMMs representing the whole combined training set. The added HMMs had the highest value of likelihood.

In the further experiments, we used each of the above three HMMs (two original and the result of the addition operation) to track a human figure in the same synthesised video sequence using a different number of samples in the modified CONDENSATION algorithm [13]. The experiments showed, that when the number of the samples was higher (256), the tracking results had very similar precision. However, as the number of the samples reduced significantly, the added HMMs showed the best performance, as shown in Figure 6.

5 Concluding Remarks

We have shown it is possible to add GMMs efficiently and without recourse to original data. Our experiments show that we can produce high quality results efficiently, and our applications show its value in support of novel applications. In particular, we have shown that adding GMMs supports adding HMMs.

We believe that improving selection would contribute considerable further benefits to our method. This is because we have observed that the candidate set we produce during simplification nearly always includes an optimal model within it — the results would of course improve if we managed to pick it every time. Elsewhere [12] we have shown that if distributions $p(\boldsymbol{x})$ and $q(\boldsymbol{x})$ are represented by GMMs, then $\int p \log(q) d\boldsymbol{x}$ can be estimated without sampling by estimating the gradient $d(\int pq^{\eta} d\boldsymbol{x})/d\eta$ at $\eta = 0$ (integrated over all space); its sloth explains its absence from this paper. We suggest that a cascade of weak classifiers might address the problem more efficiently. For example, in [12] we show that the χ^2 distance $\int (p_{out}(\boldsymbol{x}) - p_{in}(\boldsymbol{x}))^2 d\boldsymbol{x}$ between the any pair of GMM can be efficiently computed in closed form. We have observed there is often a steep rise in this metric for poor models when compared to the concatenated GMM. This could be used to partition the candidate solution set, and so form one element of a weak classifier, alongside FHV and repeated resampling.

As Vasconcelos and Lippman point out [20], adding GMMs by computing weights allows a hierarchy to be constructed. They create a hierarchy by successive simplification, from N components to $M < N$ and then to $K < M$ and so on. Our method produces a hierarchy naturally as result of thresholding the table of Chernoff bounds, no addition is required. Hierarchies of GMMs might prove advantageous in several ways, most pertinently the addition of GMMs. Recalling that simplification nearly always produces an optimal answer in the candidate set, a hierarchy may militate against poor choices during selection because other options are retained at little cost. It would then be possible to add hierarchies until a clear "winner" emerges.

Hierarchies deserve greater attention because they offer a very flexible representation. For example, let us reconsider the application of tracking walking people in video. Suppose that the hierarchy maintained a set of labels to indicate the origin of the components at all levels of the hierarchy, one level of which is guaranteed to be the concatenated GMM. Should a new GMM be made available for a particular individual already within the hierarchy, then it might be advantageous use a restricted form of addition.

Another avenue of future work is to consider the possibility that negative weights w_{ij} might be used to merge components, and also weights that exceed unity. Our technical report [12] argues that such values are not only acceptable but can be meaningfully interpreted removing points from components, as "forgetting" and "re-enforcement", because the weights are equivalent to removing points from some components, adding them into other components. Forgetting information can be as useful, not least because it mitigates against over-learning.

We should also note that although the merging Equations 8 to 10 are pitched in terms of full rank matrices, we show elsewhere [11] how to efficiently add eigenmodels that are less than full rank. Indeed, Equations 8 to 10 are generalizations of [11], extended to any number of eigenmodels. Since a component of a GMM is an eigenmodel, we can conclude it is possible to add GMMs whose components differ in rank, even locally. Consequently, this paper provides a potentially very powerful method to build high dimensional manifolds that are increasingly common in computer vision applications.

Our overall conclusion is simple: adding GMMs without recourse to original data is possible, it can rapidly produce high-quality solutions, it supports real-world applications, and offers plenty of future work.

References

[1] H. Akaike, "Information theory and an extension of the maximum likelihood principle," in *Second International Symposium on Information Theory*. Budapest: Acadamiai Kiado, 1973,

pp. 267–281.

[2] J. Bezdeck, *Pattern recognition with fuzzy objective function algorithms.* New York: Plenum, 1981.

[3] H. Bozdogan, "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] 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.

[5] R. Duda, P. Hart, and D.G.Stork, *Pattern Classification 2nd Edition.* Wiley Interscience, 2000.

[6] 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.

[7] 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.

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

[9] P. Hall, J. Collomosse, and M. Owen, "A trainable low-level classifier," in *International Conference on Pattern Recognition*, 2004.

[10] P. Hall and M.Owen, "Learning to detect low-level features," in *British Machine Vision Conference*, 2004.

[11] 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.

[12] P. Hall and Y. Hicks, "A method to add gaussian mixture models," University of Bath, U.K., Tech. Rep. CSBU-2004–03, 2004.

[13] Y. Hicks, "Modelling and tracking of articulated human motion," Ph.D. dissertation, Cardiff University, 2003.

[14] Y. Hicks, P. Hall, and A. Marshall, "A method to add hidden markov models with application to learning articulated motion," in *British Machine Vision Conference*, R. Harvey and J.A.Bangham, Eds., September 2003, pp. 489–498.

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

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

[17] S. J. Roberts, "Parametric and non-parametrics unsupervised cluster analysis," *Pattern Recognition*, vol. 30, no. 2, pp. 261–272, 1997.

[18] 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.

[19] Y. Rubner, C. Tomasi, and L. Guibas, "A metric for distributions with applications to image databases," in *Proc. IEEE International Conference of Computer Vision*, 1998, pp. 59–66.

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

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

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

[23] 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.

A A comparison of batch fitting methods

In this experiment, which was preliminary to the main experiments in the text, we compared batch fitting methods using a "short circuited" version of the larger experiment described there. Here we did not divide the ground truth but sampled directly from it. Otherwise the overall experimental technique remains identical, as do experimental conditions (see Section 3).

As discussed in the Introduction, there are many approaches to batch fitting. We do not review them here, but rather compare selected methods with a view to deciding (1) which batch to use when fitting operand GMMs to sample sets, and (2) which penalty method to use for selection by repeated sampling.

EM-penalty fitting is common place, and so provides an obvious alternative. There are, however, a variety of penalty terms. Those we tested were Minimum Message Length (MML) [22], Minimum Description Length (MDL) [16], Fuzzy Hyper Volume (FHV) [7], Evidence Density (ED) [17], Akaike's Information Criterion (AIC) [1], Robert's Bayesian Criterion (RBC) [18], The Partition Coefficient (PC) [2], ICOMP [3]. We also batch fitted GMM using the method of Figueiredo and Jain (F&J) [6].

Deliberately over-fitting a GMM to data provided us with the opportunity to include our simplify-and-select approach in this comparison of standard batch methods — the over-fitted model was substituted for a concatenated GMM. Likewise, we were able to use Vasconcelos and Lippman's (V&L) simplification method [20].

Results for all these different methods are presented in Figure 7. We found that the general pattern of these results obtains no matter what measure of difference was used. For example, Figure 8 uses the χ^2 distance between GMMs as a distance measure, computed on exactly the same data as for Figure 7.

Furthermore, we have conducted extensive tests, including iterating exhaustively over a large problem space (see Subsection 3.1) — an experiment that took several days to complete. We found no significant changes compared to the results presented here, but were able to examine results in detail by performing analysis over results by holding some variable, or set of variables, constant. As before we found no significant changes. We illustrate in Figure 9 shows this lack of effect, for a change of dimension, which shows we obtain the same general pattern of results even if we change the dimension of the problem.

We conclude that these the results presented are representative of a true pattern. The batch method of Figueiredo and Jain [6] is optimal for fitting operand GMMs, and that RBC is the preferred penalty for repeated resampling. Further, that testing using GMM in $\Re^2$ over randomized

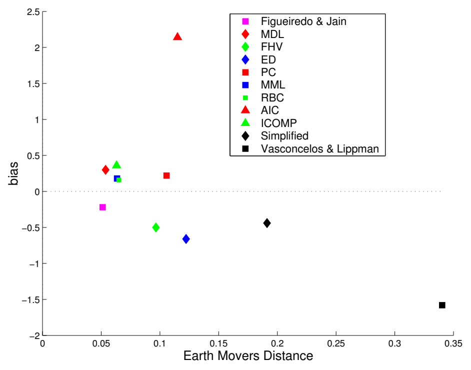

Figure 7: A scatter plot of earth movers distance and bias for GMM compared to a ground truth when fitted directly to data sampled from that ground truth.

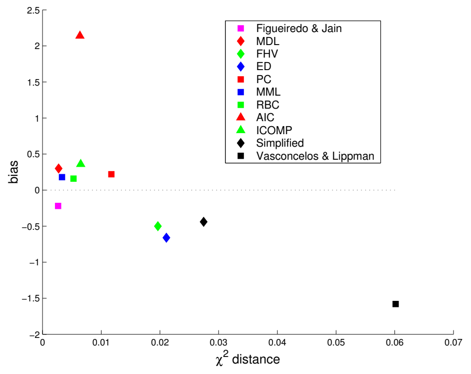

Figure 8: A scatter plot of χ^2 distance and bias) for GMM compared to a ground truth when fitted directly to data sampled from that ground truth. The symmetric difference was computed using a heuristic explained in the text of this appendix.

data sets is sufficient to characterize the different methods, and is considerably more efficient than exhaustive testing (an hour or so, rather than days).

A.1 Supplementary experiments

We conducted supplementary experiments to investigate the effect of changes of dimension and the number of components in the ground-truth on the efficiency of additive methods.

To observe the effect of dimension we used the simplest possible ground truth - a single component GMM, origin centered, unit covariance matrix. We produced such a ground truth GMM for each dimension, 2,3,4, 5. Being uninterested in the accuracy of the result GMM, it was sufficient that each ground truth GMM was input directly to the additive methods and the time taken to produce a result was measured. We repeated this process 50 times, using 100 resample points where necessary, and took the average time for each method and each dimension. Results are presented in Figure 10.

To investigate the effect of the number of components we again used circular components in $\Re^2$. In this case, however, we used 1 to 5 components inclusive, setting dispersion to $[1000, 0]$ to make sure components were well separated. As above and for exactly the same reasons, the ground-truth

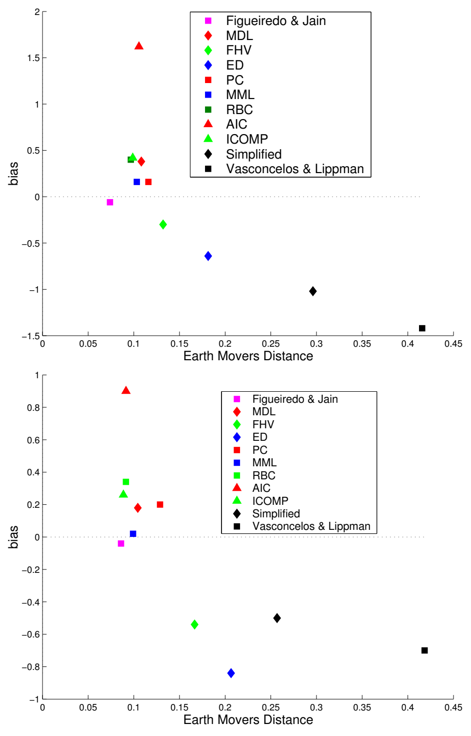

Figure 9: EMD and bias for GMM in $\Re^3$ (above) and $\Re^4$ (below).

was input directly to the additive methods, Results are presented in Figure 11.

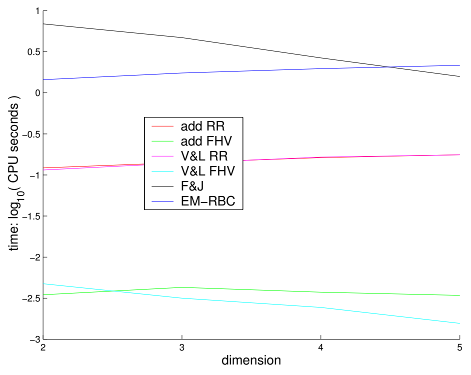

Figure 10: The effect of dimension on fitting efficiency. A $\log_{10}$ ordinate is used because of the wide range in times between the different methods.

These results show there is little change in time efficiency when dimension changes for simplify-and-select methods, but resample-and-fit methods become more efficient, which is most obvious for F&J. At this juncture we have no satisfactory explanation for this.

The effect of the number of components is clearer — a rise in all cases (after an initial fall for F&J).

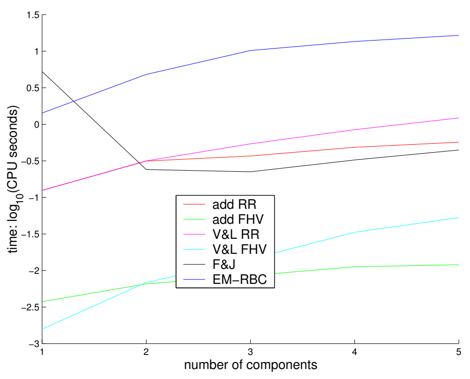

Figure 11: The effect of number of components on fitting efficiency. A $\log_{10}$ ordinate is used because of the wide range in times between the different methods.

This hints at the more general conclusion that the “complexity” of the underlying ground-truth affects the time taken to fit to it, which appeals to intuition — more difficult problems usually require more resources to solve. However, we have been unable to properly characterize the “complexity” of addition problems because the distribution of components in terms of their location, shape, size, orientation, dimension and so on all make up “complexity” — simple “overlap” measures are not sufficient, as this example show, because the components did not overlap. For the purposes of this paper we are content to ascribe the number of components to the term “complexity”, leaving the open the question of a more general characterization.