

Efficient greedy learning of Gaussian mixture models

Jakob Verbeek, Nikos Vlassis, Ben Krose

▶ **To cite this version:**

Jakob Verbeek, Nikos Vlassis, Ben Krose. Efficient greedy learning of Gaussian mixture models. Neural Computation, 2003, 15 (2), pp.469-485. 10.1162/089976603762553004 . inria-00321487

HAL Id: inria-00321487
https://inria.hal.science/inria-00321487

Submitted on 16 Feb 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Published in Neural Computation 15(2), pages 469-485, 2003.

Efficient Greedy Learning of Gaussian Mixture Models

J.J. Verbeek N. Vlassis B. Kröse
Informatics Institute, University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
E-mail: {jverbeek,vlassis, krose}@science.uva.nl

Abstract

This paper concerns the greedy learning of Gaussian mixtures. In the greedy approach, mixture components are inserted into the mixture one after the other. We propose a heuristic for searching for the optimal component to insert. In a randomized manner a set of candidate new components is generated. For each of these candidates we find the locally optimal new component. The best local optimum is then inserted into the existing mixture. The resulting algorithm resolves the sensitivity to initialization of state-of-the-art methods, like EM, and has running time linear in the number of data points and quadratic in the (final) number of mixture components. Due to its greedy nature the algorithm can be particularly useful when the optimal number of mixture components is unknown. Experimental results comparing the proposed algorithm to other methods on density estimation and texture segmentation are provided.
Keywords: unsupervised learning, finite mixture models, Gaussian mixtures, EM algorithm, greedy learning.

1 Introduction

This paper concerns the learning (fitting the parameters of) a mixture of Gaussian distributions (McLachlan and Peel, 2000). Mixture models form an expressive class of models for density estimation. Applications in a wide range of fields have emerged in the past decades. They are used for density estimation in 'unsupervised' problems, for clustering purposes, for estimating class-conditional densities in supervised learning settings, in situations where the data is partially supervised and in situations where some observations have 'missing values'.

The most popular algorithm to learn mixture models is the Expectation-Maximization (EM) algorithm (Dempster et al., 1977). For a given finite data set $\mathbf{X}_n$ of n observations and an initial mixture f^0, the algorithm provides a means to generate a sequence of mixture models $\{f^i\}$ with non-decreasing log-likelihood on $\mathbf{X}_n$. The EM algorithm is known to converge to a locally optimal solution. However, convergence to a globally optimal solution is not guaranteed. The log-likelihood of the given data set under the found mixture distribution is highly dependent on the initial mixture f^0.

The standard procedure to overcome the high dependence on initialization is to start the EM algorithm for several random initializations and use the mixture yielding maximum likelihood on the data. In this paper we present a method to learn finite Gaussian mixtures, based on the approximation result of Li and Barron (Li and Barron, 2000), which states that we can 'quickly' approximate any target density with a finite mixture model, as compared to the best we can do with *any* (possibly non-finite) mixture from the same component class. This result also holds if we learn the mixture models in a greedy manner: start with one component and optimally add new components one after the other. However locating the optimal new component boils down to finding the global maximum of a log-likelihood surface. In (Li, 1999) it is proposed to grid the parameter space to locate the global maximum, but this is unfeasible when learning mixtures in high dimensional spaces.

Here, we propose a search heuristic to locate the global maximum. Our search procedure selects the best member from a set of candidate new components. The set of candidate components is 'dynamic' in the sense that it depends on the mixture learned so far. Our search procedure for a new component has running time $O(n)$. For learning a k component mixture this results in a run time of $O(nk^2)$ if the complete mixture is updated with EM after each insertion. If the mixture is not updated between the component insertions (the approximation result still holds in this case) the run time would be $O(nk)$.

The paper is organized as follows: In Section 2 we recapitulate the definition and EM-learning of Gaussian mixtures. Section 3 discusses greedy learning of Gaussian mixtures. Our component insertion procedure is discussed in Section 4, which forms the core of the paper. Then, in Section 5, we present experimental results on two tasks: modeling of artificially generated data drawn from several types of Gaussian mixtures and texture segmentation. The experiments compare the performance of the new algorithm with the performance of several other methods to learn Gaussian mixtures. Section 6 ends the paper with conclusions and a discussion.

2 Gaussian mixtures and the EM algorithm

A Gaussian mixture (GM) is defined as a convex combination of Gaussian densities. A Gaussian density in a d-dimensional space, characterized by its mean $\mathbf{m} \in \mathbb{R}^d$ and $d \times d$ covariance matrix $\mathbf{C}$, is defined as:

$$\phi(\mathbf{x};\theta) = (2\pi)^{-d/2} \det(\mathbf{C})^{-1/2} \exp\left(-(\mathbf{x}-\mathbf{m})^\top \mathbf{C}^{-1}(\mathbf{x}-\mathbf{m})/2\right), \tag{1}$$

where θ denotes the parameters $\mathbf{m}$ and $\mathbf{C}$. A k component GM is then defined as:

$$f_k(\mathbf{x}) = \sum_{j=1}^{k} \pi_j \phi(\mathbf{x};\theta_j), \quad \text{with} \quad \sum_{j=1}^{k} \pi_j = 1 \quad \text{and for} \quad j \in \{1,\ldots,k\} : \pi_j \geq 0. \tag{2}$$

The π_j are called the mixing weights and $\phi(\mathbf{x};\theta_j)$ the components of the mixture. As compared to non-parametric density estimation, mixture density estimation (or: semi-parametric density estimation) allows for efficient evaluation of the density, since only relatively few density functions have to be evaluated. Furthermore, by using few mixture components a smoothness constraint on the resulting density is enforced, allowing for robust density estimates.

The well known Expectation-Maximization (EM) algorithm (Dempster et al., 1977) enables us to update the parameters of a given k-component mixture with respect to a data set $\mathbf{X}_n = \{\mathbf{x}_1,\ldots,\mathbf{x}_n\}$ with all $\mathbf{x}_i \in \mathbb{R}^d$, such that the likelihood of $\mathbf{X}_n$ is never smaller under the new mixture. The updates for a GM can be accomplished by iterative application of the following equations for all components $j \in \{1,\ldots,k\}$:

$$P(j \mid \mathbf{x}_i) := \pi_j \phi(\mathbf{x}_i;\theta_j)/f_k(\mathbf{x}_i), \tag{3}$$

$$\pi_j := \sum_{i=1}^{n} P(j \mid \mathbf{x}_i)/n, \tag{4}$$

$$\mathbf{m}_j := \sum_{i=1}^{n} P(j \mid \mathbf{x}_i)\mathbf{x}_i/(n\pi_j), \tag{5}$$

$$\mathbf{C}_j := \sum_{i=1}^{n} P(j \mid \mathbf{x}_i)(\mathbf{x}_i-\mathbf{m}_j)(\mathbf{x}_i-\mathbf{m}_j)^\top/(n\pi_j). \tag{6}$$

As already mentioned in the previous section, the EM algorithm is not guaranteed to lead us to the best solution, where 'best' means the solution yielding maximal likelihood on $\mathbf{X}_n$ among all maxima of the likelihood. (We implicitly assume throughout that the likelihood is bounded, by restricting the parameter space, and hence the maximum likelihood estimator is known to exist (Lindsay, 1983).) The *good* thing is that *if* we are 'close' to the global optimum of the parameter space, then it is very likely that by using EM we obtain the globally optimal solution.

3 Greedy learning of Gaussian mixtures

In this section we discuss the greedy approach to learning GMs. The basic idea is simple: Instead of starting with a (random) configuration of all components and improve upon this configuration with EM, we *build* the mixture component-wise. We start with the optimal one-component mixture, whose parameters are trivially computed. Then we start repeating two steps until a stopping criterion is met. The steps are: (i) insert a new component and (ii) apply EM until convergence. The stopping criterion can implement the choice for a pre-specified number of components or it can be any model complexity selection criterion (like Minimum Description Length, Akaike Information Criterion, Cross Validation, etc.).

3.1 Motivation

The intuitive motivation for the greedy approach is that the optimal component insertion problem involves a factor k fewer parameters than the original problem of finding a near optimal start configuration, we therefore expect it to be an easier problem. Of course, the intuitive assumption is that we can find the optimal (with respect to log-likelihood) $(k+1)$-component mixture by local search if we start the local search from the mixture obtained by inserting optimally a new component in the optimal k-component mixture. We recently applied a similar strategy to the k-means clustering problem (Likas et al., 2002), for which encouraging results were obtained.

Two recent theoretical results provide complementary motivation. The Kullback-Leibler (KL) divergence between two densities f and g is defined as:

$$D(f \parallel g) = \int_{\Omega} f(x) \log \left[f(x)/g(x) \right] \mathrm{d}x, \tag{7}$$

where Ω is the domain of the densities f and g, see (Cover and Thomas, 1991) for details. In (Li and Barron, 2000) it is shown that for an *arbitrary* probability density function f there exists a sequence $\{f_i\}$ of finite mixtures such that $f_k(\mathbf{x}) = \sum_{i=1}^{k} \pi_i \phi(\mathbf{x}; \theta_i)$ achieves Kullback-Leibler (KL) divergence $D(f \parallel f_k) \leq D(f \parallel g_P) + c/k$ for every $g_P = \int \phi(\mathbf{x}; \theta) P(d\theta)$. Hence, the difference in KL divergence achievable by k-component mixtures and the KL divergence achievable by any (possibly non-finite) mixture from the same family of components tends to zero with a rate of c/k, where c is a constant not dependent on k but only on the component family. Furthermore, it is shown in (Li and Barron, 2000) that this bound is achievable by employing the greedy scheme discussed in Section 3.2. This tells us that we can 'quickly' approximate any density by the greedy procedure. Therefore, we might expect the results of the greedy procedure—as compared to methods that initialize all-at-once—to differ more when fitting mixtures with many components.

The sequence of mixtures generated by the greedy learning method can conveniently be used to guide a model selection process in the case of an unknown number of components. Recently a result of 'almost' concavity of the log-likelihood of a data set under the maximum-likelihood k-component mixture, as function of the number of components k was presented (Cadez and Smyth, 2000). The result states that the first order Taylor approximation of the log-likelihood of the maximum likelihood models as function of k is concave under very general conditions. Hence, if we use a penalized log-likelihood model selection criterion based on a penalty term which is concave or linear in k then the penalized log-likelihood is almost concave. This implies that if the 'almost' concave turns out to be concave then there is only one peak in the penalized log-likelihood and hence this peak can be relatively easily identified, e.g. with a greedy construction method.

3.2 A General scheme for greedy learning of mixtures

Let $\mathcal{L}(\mathbf{X}_n, f_k) = \sum_{i=1}^{n} \log f_k(\mathbf{x}_i)$ (we will just write $\mathcal{L}_k$ if no confusion arises) denote the log-likelihood of the data set $\mathbf{X}_n$ under the k-component mixture f_k. The greedy learning procedure outlined above can be summarized as follows:

1. Compute the optimal (yielding maximal log-likelihood) one-component mixture f_1. Set k:=1.

2. Find the optimal new component $\phi(\mathbf{x};\theta^*)$ and corresponding mixing weight α^*:

$$\{\theta^*,\alpha^*\} = \arg\max_{\{\theta,\alpha\}} \sum_{i=1}^{n} \log\left[(1-\alpha)f_k(\mathbf{x}_i) + \alpha\phi(\mathbf{x}_i;\theta)\right] \tag{8}$$

   while keeping f_k fixed.

3. Set $f_{k+1}(\mathbf{x}) := (1-\alpha^*)f_k(\mathbf{x}) + \alpha^*\phi(\mathbf{x};\theta^*)$ and $k := k+1$;

4. Update f_k using EM until convergence. [optional]

5. If a stopping criterion is met then quit, else go to step 2.

Clearly, the crucial step is the component insertion (step 2), which we will be the topic of the next section. Let us first make a few remarks before we turn to the next section: (i) The stopping criterion in step 5 can be used to force the algorithm to find a mixture of a pre-specified number of components. Of course, step 5 may also implement any kind of model selection criterion. (ii) Note that step 4 may also be implemented by other algorithms than EM. Furthermore, step 4 is not needed to obtain the approximation result of (Li and Barron, 2000). (iii) One may note similarity to the Vertex Direction Method (VDM) (Böhning, 1995; Lindsay, 1983; Wu, 1978) to learn Gaussian Mixtures. Indeed, VDM may be regarded as a specific choice for implementing the search needed for step 2, as we explain in the discussion in Section 6.

4 Efficient search for new components

Suppose we have obtained a k-component mixture f_k, the quest is to find the component characterized by equation (8). It is easily shown that if we fix f_k and θ then $\mathcal{L}_{k+1}$ is concave as function of α only, allowing efficient optimization. For example, in (Vlassis and Likas, 2002) a second order Taylor approximation was used. However, $\mathcal{L}_{k+1}$ as function of θ can have multiple maxima. Hence, we have to perform a global search among the new components in order to identify the optimum since no constructive method to find the global maximum is known.

In our new algorithm, the global search for the optimal new component is achieved by starting km 'partial' EM searches from initial parameter pairs $\{(\theta_i,\alpha_i)\}, 1 \le i \le km$. By 'partial' EM search we mean that we fix f_k and optimize $\mathcal{L}_{k+1}$ over θ and α only. The use of partial EM searches is dictated by the general scheme above, for which the results of (Li and Barron, 2000) hold. One may wonder why we would use such partial searches, since we might as well optimize over all parameters of the resulting $(k+1)$-component mixture. The answer is that if we would update all parameters, then the number of computations needed in each of the km individual searches would be $O(nk)$ instead of $O(n)$.

Each partial search starts with a different initial configuration. After these multiple partial searches we end up with km 'candidate' new components together with their mixing weights. We pick that candidate component $\phi(\mathbf{x};\hat{\theta})$ that maximizes the likelihood when mixed into the previous mixture by a factor $\hat{\alpha}$ as in (8). Then, in step 3 of the general algorithm, in place of the (unknown) optimal parameter pair (θ^*,α^*) we use $(\hat{\theta},\hat{\alpha})$.

Note that a simple uniform quantization of the parameter space, as implemented for VDM for mixtures of univariate Gaussian distributions in (DerSimonian, 1986), is not feasible in general since the number of quantization levels grows exponentially with the dimension of the parameter space. In the following section we discuss the step 2 as proposed in (Vlassis and Likas, 2002) and its drawbacks. In Section 4.2 we present our improved search procedure.

4.1 Every data point generates a candidate

In (Vlassis and Likas, 2002) (we will from now on refer to this method as VL) it is proposed to use n candidate components. Every data point is the mean of a corresponding candidate. All candidates have the same covariance matrix $\sigma^2\mathbf{I}$, where σ is taken at a theoretically justifiable

value. For each candidate component the mixing weight α is set to the mixing weight maximizing the second order Taylor approximation of $\mathcal{L}_{k+1}$ around $\alpha = 1/2$. The candidate yielding highest log-likelihood when inserted in the existing mixture f_k is selected. The selected component is then updated using partial EM steps before it is actually inserted into f_k to give f_{k+1}. Note that for every component insertion that is performed, all point-candidates are considered. There are two main drawbacks to the aforementioned method:

1. Using n candidates in each step results in a time complexity $O(n^2)$ for the search which is unacceptable in most applications. $O(n^2)$ computations are needed since the likelihood of every data point under every candidate component has to be evaluated.

2. By using fixed small (referring to the size of the determinant of the covariance matrix) candidate components, the method keeps inserting small components in high density areas while in low density areas the density is modeled poorly. We observed this experimentally. Larger components that give greater improvement of the mixture are not among the candidates, nor are they found by the EM procedure following the component insertion.

Working on improvements of the above algorithm, we noticed that both better and faster performance could be achieved by using a smaller set of candidates that are tailored to the existing mixture. In the next section we discuss how we can generate the candidates in a data- and mixture-dependent manner.

4.2 Generating few good candidates

Based on our experience with the above method, we propose a new search heuristic for finding the optimal new mixture component. Two observations motivate the new search method:

1. The size (i.e. the determinant of the covariance matrix) of the new component should in general be smaller than the size of the existing components of the mixture.

2. As the existing mixture f_k contains more components, the search for the optimal new component should become more thorough.

In our search strategy we account for both observations by (i) setting the size of the initial candidates to a value related to (and in general smaller than) the size of the components in the existing mixture and (ii) increasing the number of candidates linearly with k.

For each insertion problem, our method constructs a fixed number of candidates per existing mixture component. Based on the posterior distributions, we partition the data set $\mathbf{X}_n$ in k disjoint subsets $A_i = \{\mathbf{x} \in \mathbf{X}_n : i = \arg\max_j \{P(j \mid \mathbf{x})\}\}$. If one is using EM for step 4 (of the general scheme given above), then the posteriors $P(i \mid \mathbf{x})$ are available directly since we have already computed them for the EM updates of the k-component mixture. For each set A_i $(i = 1, \ldots, k)$ we construct m candidate components. In our experiments we used $m = 10$ candidates but more can be used, trading run-time for better performance. In fact, like in (Smola and Schölkopf, 2000), we can compute confidence bounds of the type: *"With probability at least $1 - \delta$ the best split among m uniformly randomly selected splits is among the best ϵ fraction of all splits, if $m \geq \lceil \log \delta / \log(1 - \epsilon) \rceil$."*

To generate new candidates from A_i, we pick uniformly random two data points $\mathbf{x}_l$ and $\mathbf{x}_r$ in A_i. Then, we partition A_i into two disjoint subsets A_{il} and A_{ir}. For elements of A_{il} the point $\mathbf{x}_l$ is closer than $\mathbf{x}_r$ and vice versa for A_{ir}. The mean and covariance of the sets A_{il} and A_{ir} are used as parameters for two candidate components. The initial mixing weights for candidates generated from A_i are set to $\pi_i/2$. To obtain the next two candidates we draw new $\mathbf{x}_l$ and $\mathbf{x}_r$, until the desired number of candidates is reached.

The initial candidates can be replaced easily by candidates that yield higher likelihood when mixed into the existing mixture. To obtain these better candidates we perform the partial EM searches, starting from the initial candidates. Each iteration of the km partial updates takes $O(mnk)$ computations, since we have to evaluate the likelihood of each datum under each of the

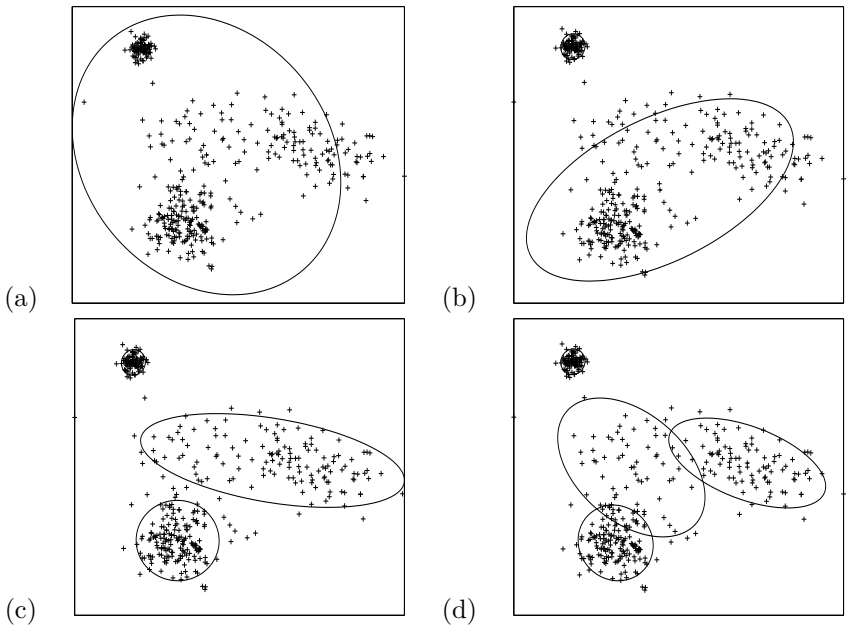


Figure 1: Images (a)-(d) illustrate the construction of a 4-component mixture distribution.

mk candidate components. In the following section we discuss how we can reduce the amount of computations needed by a factor k resulting in only $O(mn)$ computations to perform one iteration of the partial updates.

We stop the partial updates if the change in log-likelihood of the resulting $(k+1)$-component mixtures drops below some threshold or if some maximal number of iterations is reached. After these partial updates we set the new component $\phi(\mathbf{x};\theta_{k+1})$ as the candidate that maximizes the log-likelihood when mixed into the existing mixture with its corresponding mixing weight.

An example is given in Figure 1, which depicts the evolution of a solution for artificially generated data. The mixtures $f_1,\ldots,f_4$ are depicted, each component is shown as an ellipse which has the eigenvectors of the covariance matrix as axes and radii of twice the square root of the corresponding eigenvalue.

4.3 Speeding up the partial EM searches

In order to achieve $O(mn)$ time complexity for the partial EM searches initiated at the km initial candidates, we use a lower bound on the true log-likelihood instead of the true log-likelihood itself. The lower-bound is obtained by assuming that the support (non-zero density points in $\mathbf{X}_n$) for a candidate component $\phi(\mathbf{x},\mathbf{m}_{k+1},\mathbf{C}_{k+1})$ originating from existing component i is fully contained in A_i. The lower-bound, termed negative free-energy (Neal and Hinton, 1998), is detailed in Appendix A. Optimizing the lower-bound allows us to base the partial updates for a candidate from A_i purely on the data in A_i. The update equations are:

$$p(k+1 \mid \mathbf{x}_j) = \frac{\alpha\phi(\mathbf{x}_j;\mathbf{m}_{k+1},\mathbf{C}_{k+1})}{(1-\alpha)f_k(\mathbf{x}_j)+\alpha\phi(\mathbf{x}_j;\mathbf{m}_{k+1},\mathbf{C}_{k+1})} \tag{9}$$

$$\mathbf{m}_{k+1} = \frac{\sum_{j\in A_i} p(k+1 \mid \mathbf{x}_j)\mathbf{x}_j}{\sum_{j\in A_i} p(k+1 \mid \mathbf{x}_j)} \tag{10}$$

$$\mathbf{C}_{k+1} = \frac{\sum_{j\in A_i} p(k+1 \mid \mathbf{x}_j)(\mathbf{x}_j-\mathbf{m}_{k+1})(\mathbf{x}_j-\mathbf{m}_{k+1})^\top}{\sum_{j\in A_i} p(k+1 \mid \mathbf{x}_j)} \tag{11}$$

$$\alpha = \frac{\sum_{j\in A_i} p(k+1 \mid \mathbf{x}_j)}{N} \tag{12}$$

4.4 Time Complexity Analysis

As mentioned in the introduction, the total run time of the algorithm is $O(k^2 n)$ (or $O(kmn)$ if $k < m$). This is due to the updates of the mixtures f_i, which cost $O(ni)$ computations each if we use EM for these updates. Therefore, summing over all mixtures we get $O(\sum_{i=1}^{k} ni) \in O(nk^2)$. This is a factor k slower than the standard EM procedure. The run times of our experiments confirm this analysis, the new method performed on average about $k/2$ times slower than standard EM. However, if we would use EM to learn a *sequence* of mixtures consisting of $1, \ldots, k$ components then the run time would also be $O(nk^2)$.

Note that if we do not use the aforementioned speed-up, the run time would become $O(k^2 mn)$ since in that case the component allocation step for the $i+1$-st mixture component would cost $O(imn)$. This is a factor m more than the preceding EM steps, hence the total run-time goes up by a factor m.

5 Experimental Results

In this section we present results of two experiments. The experiments compare the results as obtained with k-means initialized EM, the method of VL discussed in Section 4.1 , Split and Merge EM (SMEM) (Ueda et al., 2000) and the new greedy approach presented here. The SMEM algorithm checks after convergence of EM if the log-likelihood can be improved by splitting one mixture component in two and merging two other mixture components. To initialize SMEM we also used the k-means initialization. The k-means initialization makes use of the results of applying the k-means or Generalized Lloyd Algorithm to the data (Gersho and Gray, 1992). The means are initialized as the centroids of the Voronoi Regions (VR) found, and the covariance matrices as the covariance matrix corresponding to the data in the VRs. The mixing weights were initialized as the proportion of the data present in each VR. The centroids of the k-means algorithm were initialized as a uniformly random selected subset of the data.

5.1 Artificial Data

In this experiment we generated data sets of 400 points in $\mathbb{R}^{\mathrm{d}}$ with $d \in \{2, 5\}$. The data was drawn from a Gaussian mixture of $k \in \{4, 6, 8, 10\}$ components. The separation of the components was chosen $c \in \{1, 2, 3, 4\}$. A separation of c means (Dasgupta, 1999):

$$\forall_{i \neq j}: \quad \| \boldsymbol{\mu}_i - \boldsymbol{\mu}_j \|^2 \geq c \cdot \max_{\{i,j\}} \{\mathrm{trace}(\mathbf{C}_i), \mathrm{trace}(\mathbf{C}_j)\}. \tag{13}$$

For each mixture configuration we generated 50 data sets. We allowed a maximum eccentricity (i.e. largest singular value of the covariance matrix over the smallest) of 15 for each component. Also, for each generated mixture, we generated a test set of 200 points not presented to the mixture learning algorithm. In the comparison below, we evaluate the difference in log-likelihood of the test sets under the mixtures provided by the different methods.

In Figure 2 we give show experimental results comparing our greedy approach to VL, SMEM and several runs of normal EM initialized with k-means. The average of the difference in log-likelihood on the test set is given for different characteristics of the generating mixture. The results show that the greedy algorithm performs comparable to SMEM and generally outperforms VL. Both greedy methods and SMEM outperform the k-means initialization by far.

5.2 Texture segmentation

In this experiment the task is to cluster a set of $16 \times 16 = 256$ pixel image patches. The patches are extracted from 256×256 Brodatz texture images, four of these textures are shown in Figure 3. Since different patches from the same texture differ, roughly speaking, from each other only in limited number of degrees of freedom (translation, rotation, scaling and lightness), patches from the same texture are assumed to form clusters in the 256 dimensional patch-space.

Vlassis & Likas

d=2	c=1	2	3	4	d=5	c=1	2	3	4
k=4	0.03	0.00	0.01	0.01	k=4	0.05	0.02	-0.01	0.04
6	0.01	0.03	0.05	0.02	6	0.04	0.04	0.13	0.13
8	0.00	0.04	0.06	0.03	8	0.03	0.07	0.13	0.04
10	0.01	0.06	0.07	0.03	10	0.04	0.05	0.11	0.16

Best of k runs of k-means initialization EM

d=2	c=1	2	3	4	d=5	c=1	2	3	4
k=4	0.03	0.14	0.13	0.43	k=4	0.02	0.28	0.36	0.37
6	0.03	0.16	0.34	0.52	6	0.09	0.35	0.48	0.54
8	0.02	0.23	0.38	0.55	8	0.12	0.45	0.75	0.82
10	0.06	0.27	0.45	0.62	10	0.13	0.49	0.75	0.83

SMEM

d=2	c=1	2	3	4	d=5	c=1	2	3	4
k=4	-0.01	0.00	0.00	0.00	k=4	-0.02	0.01	0.00	0.00
6	-0.02	0.00	0.02	0.02	6	-0.06	-0.01	-0.01	0.00
8	-0.02	0.00	0.00	0.03	8	-0.05	-0.04	0.00	0.02
10	-0.00	0.00	0.03	0.04	10	-0.01	0.04	0.01	0.05

Distance to generating mixture

d=2	c=1	2	3	4	d=5	c=1	2	3	4
k=4	0.04	0.03	0.03	0.02	k=4	0.16	0.13	0.14	0.11
6	0.07	0.06	0.05	0.04	6	0.28	0.22	0.19	0.18
8	0.10	0.07	0.07	0.09	8	0.45	0.33	0.32	0.42
10	0.13	0.12	0.10	0.12	10	0.58	0.50	0.45	0.51

Figure 2: Detailed exposition of log-likelihood differences. The upper three tables give the log-likelihood of a method subtracted from the log-likelihood obtained using our greedy approach. The bottom table gives averages of the log-likelihood under the generating mixture minus the log-likelihood given by our method.

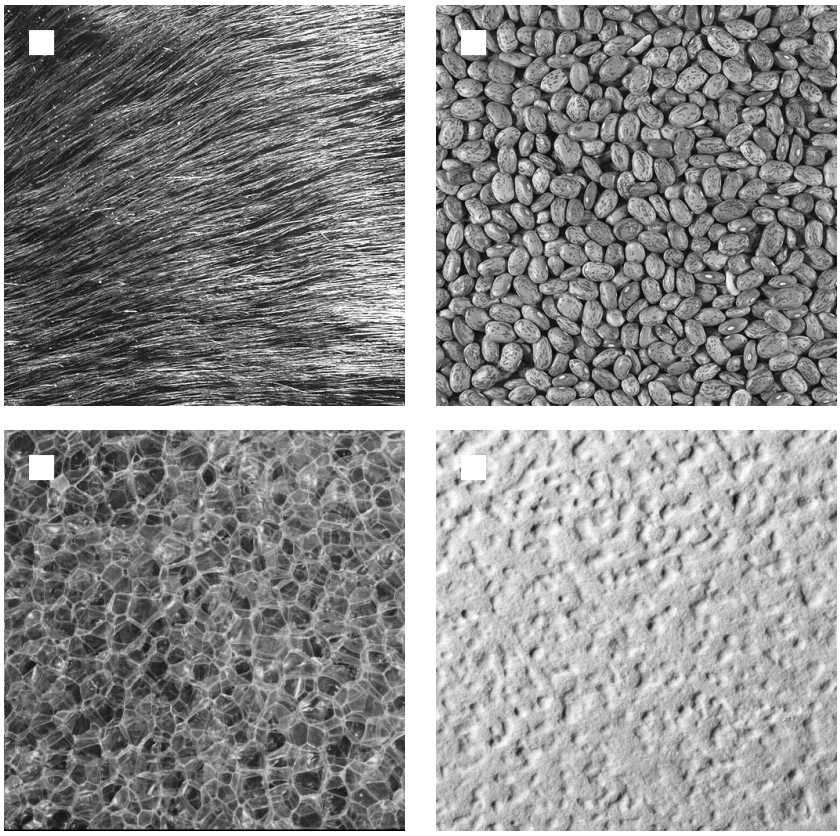


Figure 3: Several Brodatz textures, the white squares indicate the patch size.

k	2	3	4	5	6
greedy	0.20	0.34	0.48	**0.53**	**0.61**
k-means	**0.19**	0.46	0.74	0.80	0.94
SMEM	0.21	**0.22**	**0.36**	0.56	0.68
VL	0.93	1.52	2.00	2.29	2.58
uniform	1	1.59	2	2.32	2.58

Figure 4: Average conditional entropy for different values of k for the greedy approach, k-means initialized EM and SMEM and VL. The bottom line gives the conditional entropy if the clusters are totally uninformative. Bold face is used to identify the method with minimum conditional entropy for each k.

We conducted experiments where the number of textures from which patches were extracted ranged from $k = \{2, \ldots, 6\}$. For each value of k, we created 100 data sets of $500k$ patches each by picking up 500 patches at random from each of k randomly selected textures. The experiment comprised learning a Gaussian mixture model with as many components as different textures involved. We compared our greedy approach to the same methods as in the previous experiment.

To speed-up the experiment, we projected the data into a linear subspace with PCA. We used a 50 dimensional subspace, or to a lower than 50 dimensional space if that could capture at least 80% of the data variance. Once the mixture model was learned, we segmented the patches by assigning each patch to the mixture component that has the highest posterior probability for this patch. To evaluate the quality of the different segmentations discovered by the different learning methods, we considered how *informative* the found segmentations are. We measure how predictable the texture label of a patch is, given the cluster of the class. This can be done using the conditional entropy (Cover and Thomas, 1991) of the texture label given the cluster label. In Figure 4 the average conditional entropies (over 50 experiments) are given. The results of the greedy method and SMEM are roughly comparable, although the greedy approach was much faster. Both provide generally more informative segmentations than using the k-means initialization. Note that VL fails to produce informative clusters in this experiment. This is probably due to the high dimensionality of the data that renders spherical candidate components rather unfit.

6 Discussion and conclusions

Discussion Both VDM (Böhning, 1995; Lindsay, 1983; Wu, 1978) and the proposed method are instantiations of the more general greedy scheme given in section 3.2 (although VDM skips step 4 of the scheme). VDM makes use of the directional derivative $D_{f_k}(\phi)$ of the log-likelihood for the current mixture f_k, where

$$D_{f_k}(\phi) = \lim_{\alpha \to 0} [\mathcal{L}(\mathbf{X}_n, (1-\alpha) f_k + \alpha\phi) - \mathcal{L}(\mathbf{X}_n, f_k)]/\alpha.$$

VDM proceeds by picking the ϕ^* that maximizes $D_{f_k}(\phi)$ and inserting this in the mixture with a factor α^* such that $\alpha^* = \arg\max_\alpha \{\mathcal{L}(\mathbf{X}_n, (1-\alpha) f_k + \alpha\phi^*\}$. Using the directional derivative at the current mixture can be seen as an instantiation of step 2 of the general scheme. The optimization over both ϕ and α is replaced by (i) an optimization over $D_{f_k}(\phi)$ (typically implemented by griding the parameter space) and then (ii) an optimization over α. Note that by moving in the direction of maximum $D_{f_k}(\phi)$ does not guarantee that we move in the direction of maximum improvement of log-likelihood if we optimize over α subsequently.

Recently, several novel methods to learn mixtures (of Gaussian densities) were proposed among which we mention (Dasgupta, 1999; Figueiredo and Jain, 2002; Ueda et al., 2000). The first tries to overcome the difficulties of learning mixtures of Gaussian densities in high dimensional spaces. By projecting the data to a lower dimensional subspace (which is chosen uniformly randomly), finding the means in that space and then projecting them back, the problems of high dimensionality are reduced. The last two methods try to overcome the dependence of EM on the initial configuration

as does our method. In (Ueda et al., 2000) split and merge operations are applied to local optima solutions found by applying EM. The split and merge operations constitute jumps in the parameter space that allow the algorithm to jump from a local optimum to a better region in the parameter space. By then reapplying EM a better (hopefully global) optimum is found.

One may note that the latter method and the greedy approach are complementary, i.e. they can be combined. The split-and-merge operations can be incorporated in the greedy algorithm by checking after each component insertion whether a component i can be removed such that the resulting log-likelihood is greater than the likelihood before insertion. If so, the component i is removed and the algorithm continues. If not, the algorithm just proceeds as usual. However, in experimental studies on artificially generated data we found that this combination hardly gave improvement over the individual methods.

An important benefit of our new method over (Ueda et al., 2000) is that the new algorithm produces a sequence of mixtures that can be used to perform model complexity selection as the mixtures are learned. For example a kurtosis-based selection criterion, like the one in (Vlassis and Likas, 1999), can be used here. We also note that our algorithm executes faster than SMEM, which has a run time of $O(nk^3)$. In (Figueiredo and Jain, 2002) it is proposed to start with a large number of mixture components and to successively annihilate components with small mixing weights. This approach can be characterized as 'pruning' a given mixture, whereas our approach can be characterized as 'growing' a mixture. As compared to both these methods, ours does not need an initialization scheme, since the optimal one-component mixture can be found analytically.

Also Bayesian methods have been used to learn Gaussian mixtures. For example in (Richardson and Green, 1997), a reversible jump Markov Chain Monte Carlo (MCMC) method is proposed. There, the MCMC allows jumps between parameter spaces of different dimensionality (i.e. parameter spaces for mixtures consisting of differing number of components). However, the experimental results reported in (Richardson and Green, 1997) indicate that such sampling methods are rather slow as compared to constructive maximum likelihood algorithms. It is reported that about 160 'sweeps' per second are performed on a SUN Sparc 4 workstation. The experiments involve 200.000 sweeps, resulting in about 20 minutes run time. Although it is remarked that the 200.000 sweeps are not needed for reliable results, this contrasts sharply with the 2.8 seconds and 5.7 seconds run time (allowing respectively about 480 and 960 sweeps) of the standard EM and our greedy EM in a similar experimental setting executed also on a SUN Sparc 4 workstation.

Conclusions We proposed an efficient greedy method to learn mixtures of Gaussian densities that has run time $O(k^2n + kmn)$ for n data points, a final number of k mixture components and m candidates per component.

The experiments we conducted have shown that our greedy algorithm generally outperforms the k-means initialized EM. In experiments on artificial data we observed that SMEM performs in general comparable to the new method in terms of quality of the solutions found. The results on natural data were comparable to those obtained with SMEM, if SMEM was initialized with the k-means algorithm. An important benefit of the greedy method, both compared to SMEM and 'normal' EM, is the production of a sequence of mixtures, which obviates the need for initialization and facilitates model selection. As compared to VL we note: (i) The $O(n^2k^2)$ time complexity has been reduced by a factor n. (ii) The somewhat arbitrary choice of spherical candidate components with fixed variance has been replaced by a search for candidate components that depends on the data and the current mixture. (iii) Experiments suggest that if the methods yield different performance, then the new method generally outperforms the old one.

Software implementing the algorithm in MATLAB is available from the first author via e-mail.

Acknowledgments We would like to thank Aristidis Likas for useful discussions and a referee for pointing out the confidence bounds used in (Smola and Schölkopf, 2000). This research is supported by the Technology Foundation STW (project nr. AIF4997) applied science division of NWO and the technology programme of the Dutch Ministry of Economic Affairs.

References

Böhning, D. (1995). A review of reliable maximum likelihood algorithms for semiparametric mixture models. *J. Statist. Plann. Inference*, 47:5–28.

Cadez, I. V. and Smyth, P. (2000). On model selection and concavity for finite mixture models. In *Proc. of Int. Symp. on Information Theory (ISIT)*, available at `http://www.ics.uci.edu/~icadez/publications.html`.

Cover, T. and Thomas, J. (1991). *Elements of Information Theory*. Wiley.

Dasgupta, S. (1999). Learning mixtures of Gaussians. In *Proc. IEEE Symp. on Foundations of Computer Science*, New York.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38.

DerSimonian, R. (1986). Maximum likelihood estimation of a mixing distribution. *J. Roy. Statist. Soc. C*, 35:302–309.

Figueiredo, M. A. T. and Jain, A. K. (2002). Unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):381–396.

Gersho, A. and Gray, R. M. (1992). *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, Boston.

Li, J. Q. (1999). *Estimation of Mixture Models*. PhD thesis, Department of Statistics, Yale University.

Li, J. Q. and Barron, A. R. (2000). Mixture density estimation. In *Advances in Neural Information Processing Systems 12*. The MIT Press.

Likas, A., Vlassis, N., and Verbeek, J. J. (2002). The global k-means clustering algorithm. *Pattern Recognition*. to appear.

Lindsay, B. G. (1983). The geometry of mixture likelihoods: a general theory. *Ann. Statist.*, 11(1):86–94.

McLachlan, G. J. and Peel, D. (2000). *Finite Mixture Models*. Wiley, New York.

Neal, R. M. and Hinton, G. E. (1998). A view of the EM algorithm that justifies incremental, sparse, and other variants. In Jordan, M., editor, *Learning in Graphical Models*, pages 355–368. Kluwer Academic Publishers, Dordrecht, The Netherlands.

Richardson, S. and Green, P. J. (1997). On Bayesian analysis of mixtures with an unknown number of components. *J. Roy. Statist. Soc. B*, 59(4):731–792.

Smola, A. J. and Schölkopf, B. (2000). Sparse greedy matrix approximation for machine learning. In *Proc. of Int. Conf. on Machine Learning*, pages 911–918, San Francisco, CA. Morgan-Kaufmann.

Ueda, N., Nakano, R., Ghahramani, Z., and Hinton, G. E. (2000). SMEM algorithm for mixture models. *Neural Computation*, 12:2109–2128.

Vlassis, N. and Likas, A. (1999). A kurtosis-based dynamic approach to Gaussian mixture modeling. *IEEE Trans. Systems, Man, and Cybernetics, Part A*, 29:393–399.

Vlassis, N. and Likas, A. (2002). A greedy EM algorithm for Gaussian mixture learning. *Neural Processing Letters*, 15(1):77–87.

Wu, C. F. (1978). Some algorithmic aspects of the theory of optimal designs. *Annals of Statistics*, 6:1286–1301.

A A Lower-bound for fast partial EM

In the insertion step we have a two component mixture problem. The first component is the mixture f_k which we keep fixed in the partial EM steps. The second component is the new component ϕ which has a mixing weight α. The initial $k+1$ component mixture is given by $f_{k+1} = \alpha\phi + (1-\alpha)f_k$.

Using the maximization-maximization view on EM (Neal and Hinton, 1998), we can define a lower-bound F on the actual log-likelihood:

$$\sum_i \log f_{k+1}(\mathbf{x}_i) \geq F(Q,\theta) = \sum_i \log f_{k+1}(\mathbf{x}_i) - \mathcal{D}_{KL}(Q_i \parallel p(\cdot \mid \mathbf{x}_i)), \tag{14}$$

where $\mathcal{D}_{KL}$ denotes Kullback-Leibler divergence and $p(\cdot|\mathbf{x}_i)$ denotes the posterior probability on the mixture components. The distributions Q_i are the 'responsibilities' used in EM. We write q_i for the responsibility of the new component and $(1-q_i)$ for the responsibility of f_k for data point $\mathbf{x}_i$. The above lower bound can be made to match the log-likelihood by setting the responsibilities equal to the posteriors $p(s \mid \mathbf{x}_i)$.

However, in our case, when considering a candidate based on component j, we fix the Q_i for all points outside A_j such that $\forall \mathbf{x}_i \notin A_j : q_i = 0$. It is easy to see from (14), that the Q_i for which $\mathbf{x}_i \in A_j$ should match the posterior probability on the new component in order to maximize F. Note that the more the assumption that data points outside A_j have zero posterior on the new component is true, the tighter the lower bound becomes. If the data outside A_j indeed has zero posterior on the new component the lower-bound equals the log-likelihood after the partial E-step.

For the M-step we use the alternative decomposition of F:

$$F(Q,\theta) = \sum_i \mathrm{E}_{Q_i} \log p(\mathbf{x}_i, s) + \mathcal{H}(Q_i), \tag{15}$$

where we used the variable s to range over the two mixture components: f_k and ϕ. Note that the entropies $\mathcal{H}(Q_i)$ remain fixed in the M-step since they do not depend on θ. The first term can be written as:

$$\sum_i q_i[\log \phi(\mathbf{x}_i) + \log \alpha] \quad + \quad (1-q_i)[\log f_k(\mathbf{x}_i) + \log(1-\alpha)] \tag{16}$$

$$= \sum_{i \in A_j} q_i[\log \phi(\mathbf{x}_i) + \log \alpha] \quad + \quad \sum_i (1-q_i)[\log f_k(\mathbf{x}_i) + \log(1-\alpha)]. \tag{17}$$

Note that in (17) the new mixture component ϕ only occurs in terms for data in A_j. Setting the derivative of (17) to zero w.r.t. the parameters of the new component (mixing weight α, mean $\mathbf{m}$ and covariance matrix $\mathbf{C}$) gives the update equations in Section 4.3. Note that from (17) F can also be evaluated using only the points in A_j if we stored the complete data log-likelihood under f_k in advance.