Efficient greedy learning of Gaussian mixtures

Jakob Verbeek, Nikos Vlassis, Ben Krose

Efficient Greedy Learning of Gaussian Mixtures

J.J. Verbeek N. Vlassis B. Kröse

Intelligent Autonomous Systems Group, Computer Science Institute,
Faculty of Science, University of Amsterdam,
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands.

Abstract

We present a deterministic greedy method to learn a mixture of Gaussians. The key element is that we build-up the mixture component-wise: we start with one component and then add new components one at a time and update the mixtures in between the component insertions. Instead of solving directly a optimization problem involving the parameters of all components, we replace the problem by a sequence of component allocation problems involving only the parameters of the new component. Included are experimental results obtained from extensive tests on artificially generated data sets. The new learning method is compared with the standard EM with random initialization approach as well as to other existing approaches to learning Gaussian mixtures.

Keywords: unsupervised learning, finite mixtures, EM Algorithm.

1 Introduction

This paper concerns learning mixtures of Gaussian distributions [10]. Mixture models form an expressive class of models for density estimation. Applications in many 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, in situations where the data is partially supervised and in situations where some observations have 'missing values'.

The most widely used algorithm to learn mixture models is the Expectation-Maximization (EM) algorithm [5]. Given a finite data set $\mathbf{X}_n$ of n observations and a initial mixture f_0, the algorithm provides a means to generate a sequence of mixture models $\{f_i\}$ with increasing 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 method 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. The result is a non-deterministic algorithm. The non-determinism of the algorithm may be unfavorable, especially in the process of developing large systems that include the learning of a mixture model as a component. In order to effectively evaluate performance of other components it is

convenient to rule out accidental aberrant behavior of the complete system due to 'unlucky' initializations of the mixture learning component.

In this paper we present a deterministic method to learn mixtures of k Gaussians. The method is based on previous work [14]. Here, we solve the main drawbacks of the latter work: (i) the time complexity is reduced from $O(k^2n^2)$ to $O(k^2n)$ (ii) initialization of new components is done less arbitrary. This results in an algorithm that on average performs significantly better, both in terms of speed and quality of solutions. Both algorithms use roughly the same approach: the mixture is build-up by starting with a one-component mixture and adding new components one after the other. The new algorithm replaces the component insertion step with a procedure that inserts a component that is selected among a set of 'candidate' components dependent on the existing mixture. Experiments indicate the superiority of the new method.

The paper is organized as follows: In Section 2 we recapitulate the definition and EM-learning of Gaussian mixtures. Section 3 forms the core of the paper and describes our new greedy approach to Gaussian mixture learning. Then, in Section 4, we present experimental results of modeling artificially generated data drawn from several types of Gaussian mixtures. The experiments compare the results of the new algorithm with four other methods to learn mixtures of Gaussians. Section 5 ends the paper with conclusions and a discussion.

2 Gaussian Mixtures and the EM Algorithm

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

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

A mixture of k Gaussians is then defined as:

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

The π_i are called the mixing weights and $\phi(\mathbf{x};\theta_i)$ the components of the mixture.

The well known EM algorithm [5] 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 mixture of Gaussians can be accomplished by iterative application of the following equations for all components $j \in \{1,\ldots,k\}$:

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

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

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

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.[1] 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

Two recent theoretical results motivate the use of a greedy approach to mixture learning: In [8] 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[2] $D(f \parallel f_k) \leq D(f \parallel g_P) + c/k$ 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 speed c/k (where c is a constant not dependent on k but only on the component family). Furthermore, it is shown that this bound is achievable by employing a greedy approach as discussed above. This result 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 the standard (randomly initialized) EM approach 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 [3]. 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 the best model complexity can be easily identified if we use a greedy construction method.

Next, we provide the general scheme for greedy mixture learning given in [8]. 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] Some constraints have to be placed on the parameters so as to prevent degenerate solutions.

[2] $D(f \parallel g) = \int_\Omega f(x) \log(f(x)/g(x)) \mathrm{d}x$ where Ω is the domain of the densities f and g, see [4] for details.

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

2. Perform a search to find the optimal new component $\phi(\mathbf{x};\theta^*)$ and corresponding mixing weight α^* while keeping f_k fixed, where

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

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.

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

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. Note that step 4 may also be implemented by other algorithms than EM. In the rest of this section we are concerned with the search in step 2.

Let f_{k+1} be as in step 3, then it is easily shown that if we fix f_k and ϕ then $\mathcal{L}_{k+1}$ is concave as function of α only, allowing efficient optimization.[3] 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. In our new algorithm, the global search for the optimal new component is achieved by starting $6k$ 'partial' EM searches. By a 'partial' EM search we mean that we fix f_k and optimize over ϕ and α only. 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 (i) if we would update all parameters then the number of computations needed in *each* search would be $O(nk)$ where it is $O(n)$ if we perform partial EM updates. (ii) the partial searches consitute a more robust approximation for step 2.

Each partial search starts with a different initial configuration. After these multiple partial searches we end up with $6k$ 'candidate' new components. 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 (1). Then, in step 3 of the general algorithm, instead of inserting the global maximum $\phi(\mathbf{x};\theta^*)$ with a factor α^* we insert $\phi(\mathbf{x};\hat{\theta})$ with a factor $\hat{\alpha}$. The rest of this section discusses how we construct the $6k$ initial configurations. First we introduce kd-trees, which are used in the construction.

Originally kd-trees [1] were designed to speed-up the execution of nearest neighbor queries and related problems like computing all points in an ϵ neighborhood of a given query point. A kd-tree defines a recursive binary partitioning of a k-dimensional data set, where the root node contains all data. Sproull [11] proposed to partition the data at each node by cutting with a hyper-plane perpendicular to

[3]The concavity follows if we note that the second derivative is everywhere non-positive [2].

the direction with greatest variance of the data present in that node, i.e. perpendicular to the Principal Component [7]. One may view the resulting procedure as a 'nested' Principal Component Analysis. If we fully expand the tree, the leafs of the tree are given by the individual data points. We can regard each node in the tree as a bucket containing a portion of the data. Every layer of the tree gives a partitioning of the data. It turns out that these partitions provide quite reasonable *clusterings* of the data in terms of mean squared distance from the data to their closest cluster center (the cluster centers are given by the bucket means), see [9].

Two observations motivate our new search method: (i) The size (i.e. the determinant of the covariance matrix) of the *inserted* component (*after* the partial EM steps) of the above mentioned method is generally smaller than the size of the existing components. (ii) It seems (we do not have any proof) that the smaller (the determinant of the covariance of) the new component gets, the larger risk we run to end up in a local maximum when applying EM to find the globally optimal new component. In our search strategy we account for both observations by (i) setting the size of the initial candidate configurations 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 candidate insertion configurations linearly in k.

For each insertion problem, our method constructs one kd-tree for each existing mixture component, which is expanded up to two layers. Both the means *and the covariance matrices* of the candidate components are then initialized according to the 'buckets' of the kd-tree. Below we discuss the procedure in more detail. Based on the posterior distributions, we partition the data set $\mathbf{X}_n$ in k disjoint subsets $A_i = \{\mathbf{x} \in \mathbf{X}_n : P(i \mid \mathbf{x}) = \max_j\{P(j \mid \mathbf{x})\}\}$ with $1 \leq i \leq k$. The posteriors $P(i \mid \mathbf{x})$ are available directly since we already computed them for the EM updates of the k-component mixture. For each set A_i we construct the first 6 nodes, residing in the first two layers, of the kd-tree T_i based on the data $\mathbf{x} \in A_i$ only. Then, for each of the $6k$ nodes we initialize a new component with the mean and covariance of the data present in that node. The initial mixing weights for candidates generated from A_i are set to $\pi_i/2$. The reader may have noticed that using only two layers (and hence six candidate components) per the 'local' kd-tree is somewhat arbitrary. However, experiments indicated that using more layers does not improve results significantly while slowing down the algorithm.

4 Experimental Demonstration

In the experiment we generated data sets of 400 points in a $\mathbb{R}^d$ space with $d \in \{2, 3, 4, 5\}$. 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: $\forall i \neq j : \| \mu_i - \mu_j \|^2 \geq c \max_{i,j}\{\text{trace}(\mathbf{C}_i), \text{trace}(\mathbf{C}_j)\}$. For each mixture configuration we generated 50 data sets (resulting in 3200 data sets in total). 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 1000 points not presented to the mixture learning algorithms. In the comparison below, we compare the

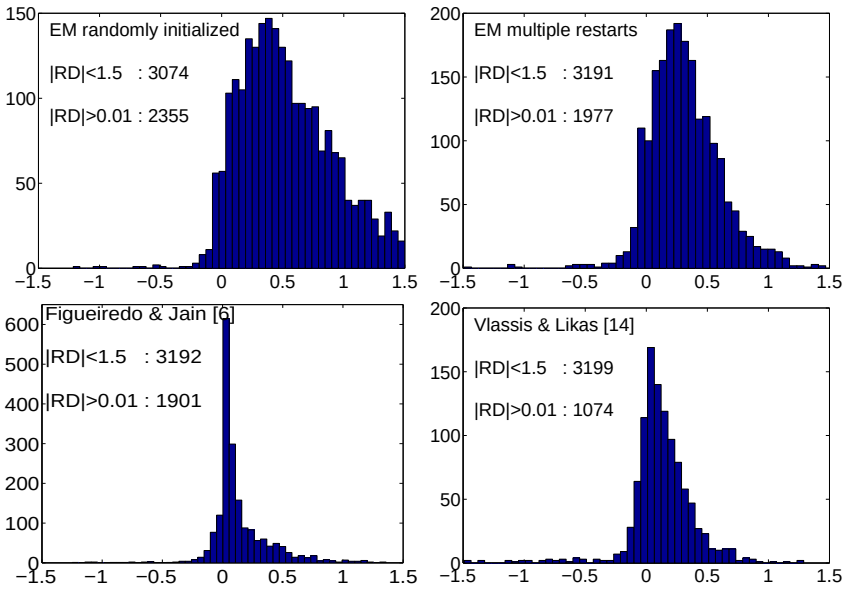

Figure 1: Histograms comparing the new method with several existing methods in terms of $RD = D_{\text{other method}} - D_{\text{our method}}$.

log-likelihood of the test sets under the mixtures $\hat{f}$ provided by several different learning methods with the log-likelihood under the generating mixture f. Let $D = \mathcal{L}(\mathbf{X}_n, f) - \mathcal{L}(\mathbf{X}_n, \hat{f})$. This difference D provides an empirical estimate of the KL divergence between $\hat{f}$ and the generating mixture, where the integral is replaced by a sum over the observed data.

In the histograms of Figure 1, we compare our new learning method with four other methods by looking at the difference RD in (estimated) KL divergence. The four other methods are: randomly initialized EM, best of multiple randomly initialized EM runs, and the methods described in [6] and [14]. For reasons of clear exposition we plotted in the histograms only results for which $0.01 <\mid RD \mid< 1.5$. In each plot it is indicated how many of the 3200 experiments satisfied these constraints. For each comparison our new method out performed the other method on average. In figure 2 we provide tables summarizing average $RD = D_{EM} - D_{\text{our method}}$ obtained for the different types of generating mixtures. We conclude that the new algorithm gives greater improvement over randomly initialized EM as the separation and the number of components in the generating mixture increase. Space limits prevent further discussion of the experimental results here.

5 Conclusions and Discussion

We proposed a greedy method to learn mixtures of Gaussians that has run time $O(nk^2)$. As compared to the standard randomly initialized EM algorithm we

$d=2$	$c=1$	2	3	4	$d=3$	$c=1$	2	3	4
$k=4$	0.05	0.24	0.35	1.11	$k=4$	0.10	0.26	0.33	0.89
6	0.13	0.34	0.51	0.61	6	0.17	0.44	0.46	0.68
8	0.12	0.48	0.60	0.73	8	0.24	0.53	0.69	0.70
10	0.20	0.36	0.68	0.75	10	0.25	0.58	0.66	0.86
$d=4$	$c=1$	2	3	4	$d=5$	$c=1$	2	3	4
$k=4$	0.17	0.21	0.22	0.26	$k=4$	0.18	0.35	0.31	1.05
6	0.16	0.37	0.10	0.50	6	0.26	0.47	1.64	1.69
8	0.26	0.52	0.64	0.65	8	0.36	0.62	0.80	1.27
10	0.35	0.61	0.73	0.62	10	0.41	0.67	0.87	2.57

Figure 2: Averages (50 experiments each) of RD for different experimental settings.

observe: (i) The proposed algorithm is deterministic. (ii) Experiments indicate superior performance of the new method while run time is increased only by a factor linear in k. As compared to the method proposed in [14] we note: (i) The $O(n^2k^2)$ time complexity has been reduced by a factor n. (ii) The somewhat arbitrary choice for spherical candidate components with fixed variance has been replaced by a search for candidate components that depends on the current mixture. (iii) Experiments suggest that if the methods yield different performance, then the new method generally outperforms the old one.

Recently, several other new methods to learn mixtures (of Gaussians) were proposed among which we mention [6, 12]. In [12] split and merge operations are applied to locally optimal solutions found by 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 re-applying EM a better (hopefully global) optimum is found. An important benefit of our new method over [12] 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 the model selection criterion proposed in [13], which is based on kurtosis tests may be used here. In [6] it is proposed to start with a 'large' number k_{max} of mixture components and to successively annihilate components with small mixing weights. This approach can be characterized as 'pruning' a given mixture, where our approach can be characterized as 'growing' a mixture. There, also a sequence of mixtures of different number of components is generated and it is exploited by integrating the process of model selection and fitting of the mixtures. However, note that (i) In general we may not know how to set k_{max} if we do not know what the 'correct' number k of components is. (ii) Suppose that the 'correct' number of components is k. For the 'pruning' approach, the considered mixtures f_i consist of $k \leq i \leq k_{max}$ components, so at every stage of the algorithm *at least* k components are updated with EM. For our 'growing' approach, at every stage *at most* k components are updated.

Acknowledgments We would like to thank Aris Likas for many useful remarks. 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

[1] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, 1975.

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

[3] I. V. Cadez and P. Smyth. 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`, 2000.

[4] T. Cover and J. Thomas. *Elements of Information Theory*. Wiley, 1991.

[5] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Roy. Statist. Soc. B*, 39:1–38, 1977.

[6] M.A.T. Figueiredo and A.K. Jain. Unsupervised learning of finite mixture models. *to appear in IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2001.

[7] I.T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, 1986.

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

[9] A. Likas, N. Vlassis, and J.J. Verbeek. The global k-means clustering algorithm. Technical report, Computer Science Institute, University of Amsterdam, The Netherlands, February 2001. IAS-UVA-01-02.

[10] G. J. McLachlan and D. Peel. *Finite Mixture Models*. Wiley, New York, 2000.

[11] R. F. Sproull. Refinements to nearest-neighbor searching in k-dimensional trees. *Algorithmica*, 6:579–589, 1991.

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

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

[14] N. Vlassis and A. Likas. A greedy EM algorithm for Gaussian mixture learning. Technical report, Computer Science Institute, University of Amsterdam, The Netherlands, September 2000. IAS-UVA-00-08, to appear in Neural Processing Letters.