

# A Variational EM Algorithm for Large-Scale Mixture Modeling

J.J. Verbeek      N. Vlassis      J.R.J. Nunnink

Informatics Institute, Faculty of Science, University of Amsterdam  
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands  
{jverbeek, vlassis, jnunnink}@science.uva.nl

**Keywords:** Gaussian mixture, EM algorithm, variational approximation, clustering, very large database.

## Abstract

Mixture densities constitute a rich family of models that can be used in several data mining and machine learning applications, for instance, clustering. Although practical algorithms exist for learning such models from data, these algorithms typically do not scale very well with large datasets. Our approach, which builds on previous work by other authors, offers an acceleration of the EM algorithm for Gaussian mixtures by precomputing and storing sufficient statistics of the data in the nodes of a kd-tree. Contrary to other works, we obtain algorithms that strictly increase a lower bound on the data log-likelihood in every learning step. Experimental results illustrate the validity of our approach.

## 1 Introduction

Mixture modeling and clustering are common tasks in data analysis and occur in applications across many fields. Clustering is employed to find prototypical data items in a large data base or to determine whether new data matches previously seen data. Clustering and mixture modeling are also used in supervised classifier systems, where new data is compared with each cluster and each cluster defines a mapping to the class labels. Mixture modeling is used for density estimation in general with many different applications.

The Expectation-Maximization (EM) algorithm and the  $k$ -means algorithm are among the most popular algorithms for data clustering and mixture modeling [1]. This is mainly due to the ease of implementation of these algorithms and the fact that they are guaranteed to improve the associated objective functions in every step of the algorithms without the need to set any parameters. However, since for both algorithms the run-time per iteration is linear in both the number of data items  $n$  and the number of clusters  $k$ ,

their applicability is limited in large scale applications with many data and many clusters.

Recently several authors [2, 3, 4, 5] proposed speedups of these algorithms based on analyzing large chunks of data at once to save distance computations. The idea is to use geometrical reasoning to determine that for chunks of data a particular prototype is the closest ( $k$ -means) or the posterior on mixture components hardly varies in the chunk of data (EM for Mixture of Gaussians). Cached sufficient statistics can then be used to perform the update step of the algorithms.

In this paper we employ a variational EM approach [6] and show how such speedups for the EM algorithm can be guaranteed to increase a lower bound on the data log-likelihood in each step. We also derive closed form and efficiently computable optimal assignments of chunks of data to mixture components (E-step). Furthermore, the variational EM framework allows for arbitrary partitionings, where existing techniques were forced to use relatively fine partitionings of the data. Thus the proposed framework allows more freedom in designing new speedup algorithms.

The rest of this paper is organized as follows: in the next section we discuss Gaussian mixture modeling and in Section 3 we derive the corresponding EM algorithm. Section 4 shows how we can use locally shared responsibilities to speed-up the EM algorithm. Then, in Section 5 we compare our work with related work. In Section 6 we describe our experiments on speeding up Gaussian mixture learning and end with conclusions in Section 7.

## 2 Clustering as Gaussian mixture learning

The problem of data clustering can be viewed as a density estimation problem if we treat the data points as samples from a mixture density of  $k$  components

(clusters). A typical case is to assume Gaussian components.

A  $k$ -component Gaussian mixture for a random vector  $x$  in  $\mathbb{R}^d$  is defined as the convex combination

$$p(x) = \sum_{s=1}^k p(x|s) p(s) \quad (1)$$

of  $d$ -dimensional Gaussian densities

$$p(x|s) = (2\pi)^{-d/2} |C_s|^{-1/2} \exp \left[ -\frac{1}{2} (x - m_s)^\top C_s^{-1} (x - m_s) \right] \quad (2)$$

each parameterized by its mean  $m_s$  and covariance matrix  $C_s$ . The components of the mixture are indexed by the random variable  $s$  that takes values from 1 to  $k$ , and  $p(s)$  defines a discrete ‘prior’ distribution over the components.

Given a set  $\{x_1, \dots, x_n\}$  of points, assumed to be independent and identically distributed, the learning task is to estimate the parameter vector  $\theta = \{p(s), m_s, C_s\}_{s=1}^k$  of the  $k$  components that maximizes the log-likelihood function

$$\mathcal{L}(\theta) = \sum_{i=1}^n \log p(x_i; \theta) = \sum_{i=1}^n \log \sum_{s=1}^k p(x_i|s) p(s). \quad (3)$$

Throughout we assume that the likelihood function is bounded from above (e.g., by placing lower bounds on the eigenvalues of the components covariance matrices) in which case the maximum likelihood estimate is known to exist [7].

### 3 The variational EM algorithm

Maximization of the data log-likelihood  $\mathcal{L}(\theta)$  can be efficiently carried out by the EM algorithm [8]. In this work we consider a variational generalization of EM [6] which allows for useful extensions.

The variational EM algorithm performs iterative maximization of a lower bound of the data log-likelihood. In our case, this bound  $\mathcal{F}(\theta, Q)$  is a function of the current mixture parameters  $\theta$  and a factorized distribution  $Q = \prod_{i=1}^n q_i(s)$ , where each  $q_i(s)$  corresponds to a data point  $x_i$  and defines an arbitrary discrete distribution over  $s$ . For a particular realization of  $s$  we will refer to  $q_i(s)$  as the ‘responsibility’ of component  $s$  for the point  $x_i$ .

This lower bound, analogous to the (negative) variational free energy in statistical physics, can be expressed by the following two (equivalent) decompo-

sitions

$$\mathcal{F}(\theta, Q) = \sum_{i=1}^n [\log p(x_i; \theta) - D(q_i(s) \parallel p(s|x_i; \theta))] \quad (4)$$

$$= \sum_{i=1}^n \sum_{s=1}^k q_i(s) [\log p(x_i, s; \theta) - \log q_i(s)] \quad (5)$$

where  $D(\cdot \parallel \cdot)$  denotes the Kullback-Leibler divergence between two distributions, and  $p(s|x_i)$  is the posterior distribution over components of a data point  $x_i$  computed from (1) by applying the Bayes’ rule. The dependence of  $p$  on  $\theta$  is throughout assumed, although not always written explicitly.

Since the Kullback-Leibler divergence between two distributions is non-negative, the decomposition (4) defines indeed a lower bound on the log-likelihood. Moreover, the closer the responsibilities  $q_i(s)$  are to the posteriors  $p(s|x_i)$ , the tighter the bound. In particular, maxima of  $\mathcal{F}$  are also maxima of  $\mathcal{L}$  [6]. In the original derivation of EM [8], each E step of the algorithm sets  $q_i(s) = p(s|x_i)$  in which case, and for the current value  $\theta^t$  of the parameter vector, holds  $\mathcal{F}(\theta^t, Q) = \mathcal{L}(\theta^t)$ . However, other (suboptimal) assignments to the individual  $q_i(s)$  are also allowed provided that  $\mathcal{F}$  increases in each step.

For particular values of the responsibilities  $q_i(s)$ , we can solve for the unknown parameters of the mixture by using the second decomposition (5) of  $\mathcal{F}$ . It is easy to see that maximizing  $\mathcal{F}$  for the unknown parameters of a component  $s$  yields the following solutions:

$$p(s) = \frac{1}{n} \sum_{i=1}^n q_i(s), \quad (6)$$

$$m_s = \frac{1}{np(s)} \sum_{i=1}^n q_i(s) x_i, \quad (7)$$

$$C_s = \frac{1}{np(s)} \sum_{i=1}^n q_i(s) x_i x_i^\top - m_s m_s^\top. \quad (8)$$

### 4 Locally shared responsibilities

As mentioned above, in each step of the variational EM we are allowed to assign any responsibilities  $q_i(s)$  to the data as long as this increases  $\mathcal{F}$ . The idea behind our Chunky EM algorithm is to assign equal responsibilities to chunks of data points that are nearby in the input space.

Consider a partitioning  $\mathcal{A}$  of the data space into a collection of non-overlapping cells  $\{A_1, \dots, A_m\}$ , such that each point in the data set belongs to a single cell. To all points in a cell  $A \in \mathcal{A}$  we assign the same responsibility distribution  $q_A(s)$  which we can compute in an optimal way as we show next.

Note from (5) that the objective function  $\mathcal{F}$  can be written as a sum of local parts  $\mathcal{F} = \sum_{A \in \mathcal{A}} \mathcal{F}_A$ , one per cell. If we impose  $q_i(s) = q_A(s)$  for all data points  $x_i \in A$ , then the part of  $\mathcal{F}$  corresponding to a cell  $A$  reads

$$\begin{aligned} \mathcal{F}_A &= \sum_{x_i \in A} \sum_{s=1}^k q_A(s) [\log p(x_i|s) + \log p(s) - \log q_A(s)] \\ &= |A| \sum_{s=1}^k q_A(s) [\log p(s) - \log q_A(s) \\ &\quad + \frac{1}{|A|} \sum_{x_i \in A} \log p(x_i|s)]. \end{aligned} \quad (9)$$

If we set the derivatives of  $\mathcal{F}_A$  w.r.t.  $q_A(s)$  to zero we find the optimal distribution  $q_A(s)$  that (globally) maximizes  $\mathcal{F}_A$ :

$$q_A(s) \propto p(s) \exp(\log p(x|s))_A \quad (10)$$

where  $\langle \cdot \rangle_A$  denotes average over all points in cell  $A$ . Such an optimal distribution can be separately computed for each cell  $A \in \mathcal{A}$ , and only requires computing the average joint log-likelihood of the points in  $A$ .

#### 4.1 Speedup using cached statistics

We now show that it is possible to efficiently compute (i) the optimal  $q_A(s)$  for each cell  $A$  in the E-step and (ii) the new values of the unknown mixture parameters in the M-step, if some statistics of the points in each cell  $A$  are cached in advance. The averaging operation in (10) can be written<sup>1</sup>:

$$\begin{aligned} \langle \log p(x|s) \rangle_A &= \frac{1}{|A|} \sum_{x_i \in A} \log p(x_i|s) \\ &= -\frac{1}{2} [\log |C_s| + \frac{1}{|A|} \sum_{x_i \in A} (x_i - m_s)^\top C_s^{-1} (x_i - m_s)] \\ &= -\frac{1}{2} [\log |C_s| + m_s^\top C_s^{-1} m_s + \langle x^\top C_s^{-1} x \rangle_A - \\ &\quad 2m_s^\top C_s^{-1} \langle x \rangle_A] \\ &= -\frac{1}{2} [\log |C_s| + m_s^\top C_s^{-1} m_s + \text{Tr}\{C_s^{-1} \langle x x^\top \rangle_A\} - \\ &\quad 2m_s^\top C_s^{-1} \langle x \rangle_A]. \end{aligned} \quad (11)$$

From this we see that the mean  $\langle x \rangle_A$  and covariance  $\langle x x^\top \rangle_A$  of the points in  $A$  are sufficient statistics for computing the optimal responsibilities  $q_A(s)$ .

The same statistics can be used for updating the mixture parameters  $\theta$ . If we set the derivatives of (9)

<sup>1</sup>We ignore the additive constant  $-\frac{d}{2} \log(2\pi)$  which translates into a multiplicative constant in (10).

w.r.t.  $\theta$  to zero we find the update equations:

$$p(s) = \frac{1}{n} \sum_{A \in \mathcal{A}} |A| q_A(s) \quad (12)$$

$$m_s = \frac{1}{np(s)} \sum_{A \in \mathcal{A}} |A| q_A(s) \langle x \rangle_A \quad (13)$$

$$C_s = \frac{1}{np(s)} \sum_{A \in \mathcal{A}} |A| q_A(s) \langle x x^\top \rangle_A - m_s m_s^\top \quad (14)$$

in direct analogy to the update equations (6)–(8), with the advantage that the linear complexity in the number of data points has been replaced by linear complexity in the number of cells of the particular partitioning.

Note that, whatever partitioning we choose, the Chunky EM algorithm presented above (which interacts with the data only through the cached statistics of chunks of data) is always a convergent algorithm that strictly increases in each step a lower bound on data log-likelihood. In the limit, if we partition all data points into separate cells, the algorithm will converge to a local maximum of the data log-likelihood.

#### 4.2 Using a kd-tree for partitioning

The algorithm presented in the previous section works for any partitioning, as long as sufficient statistics of the data have been stored in the corresponding cells. We note that the amount of computation needed to perform one iteration of the Chunky EM increases linearly with the number of cells. Moreover, it is easy to see that if we refine a specific partitioning and recompute the responsibilities  $q_A(s)$  then  $\mathcal{F}$  cannot decrease. In the limit holds  $\mathcal{F} = \mathcal{L}$  and the approximation bound is tight. Clearly, various trade-offs between the amount of computation and the tightness of the bound can be made, depending on the particular application.

A convenient structure for storing statistics in a way that permits the use of different partitionings in the course of the algorithm is a kd-tree [9, 2]. This is a binary tree that defines a hierarchical decomposition of the data space into cells, each cell corresponding to a node of the tree. Each node is split into two children nodes by a hyperplane that cuts through the data points in the node. Typically, a node is represented by a hyper-rectangle and an axis-aligned hyperplane is used for splitting nodes. In our case we use a kd-tree where each node is split along the bisector of the first principal component of the data in the node [10], leading to irregularly shaped cells. As in [2] we store in each node of the kd-tree the sufficient statistics of all data points under this node. Building these cached statistics can be efficiently done bottom-up by noticing that the statistics of a parent node are the sum of the statistics of its children nodes. Building the kd-tree and storing statistics in its nodes has cost  $O(n \log n)$ .

A particular expansion of the kd-tree corresponds to a specific partitioning  $\mathcal{A}$  of the data space, where each cell  $A \in \mathcal{A}$  corresponds to an outer node of the expanded tree. Further expanding the tree means refining the current partitioning, and in our implementations as heuristic to guide the tree expansion we employ a best-first search strategy in which we expand the node that leads to maximal increase in  $\mathcal{F}$ . Note that computing the change in  $\mathcal{F}$  involves only a node and its children so it can be done fast.

We also need a criterion when to stop expanding the tree, and one could use among others bounds on the variation of the data posteriors inside a node like in [2], a bound on the size of the partitioning (number of outer nodes at any step), or sampled approximations of the difference between log-likelihood and  $\mathcal{F}$ . Another possibility, explained in the next section, is to control the tree expansion based on the performance of the algorithm, i.e., whether a new partitioning improves the value of  $\mathcal{F}$  from the previous partitioning.

## 5 Related work

The idea of using a kd-tree for accelerating the EM algorithm in large-scale mixture modeling was first proposed in [2]. In that work, in each EM step every node in the kd-tree is assigned responsibility distribution equal to the Bayes posterior of the centroid of the data points stored in the node, i.e.,  $q_A = p(s|\langle x \rangle_A)$ . In comparison, we use  $q_A$  proportional to the average joint log-likelihood, c.f. (10). If there is little variation in the posteriors within a node (and this is achieved by having very fine-grained partitionings), the approximation will hardly affect the update in the M-step, and therefore the M-step will probably still increase the data log-likelihood. However this is not guaranteed. Also, in [2] a different tree expansion is computed in each EM step, while as stopping criterion for tree expansion bounds are used on the variation of the posterior probabilities of the data inside a node of the kd-tree (a nontrivial operation that in principle would involve solving a quadratic programming problem).

The main advantage of our method compared to [2] is that we *provably* increase in each EM step a lower bound of the data log-likelihood by computing the *optimal* responsibility distribution for each node. Moreover, this optimal distribution is independent of the size, shape, or other properties of the node, allowing us to run EM even with very coarse-grained partitionings. As mentioned above and as demonstrated in the experiments below, by gradually refining the partitioning while running EM we can get close to the optima of the log-likelihood in relatively few EM steps.

## 6 Experimental verification

In this section we describe our experiments and present and discuss the obtained results.

## 6.1 Implementation

In the experiment we applied our variational EM algorithm to learn a  $k$ -component mixture of Gaussians for the data. First, a kd-tree is constructed as in [10] and we cache in its nodes the data statistics bottom-up, as explained above. The EM algorithm is started with an initial expansion of the tree to depth two. We keep this partitioning fixed and run the variational EM till convergence. Convergence is measured in terms of relative increase in  $\mathcal{F}$ . We then refine the partitioning by expanding the node of the tree that leads to maximal increase in  $\mathcal{F}$ . Then we run again the variational EM till convergence, refine the partitioning by expanding best-first a single node of the tree, etc. We stop the algorithm if  $\mathcal{F}$  is hardly improved between two successive partitionings.

Note that a refining a particular partitioning and computing the new (shared and optimal) responsibilities, can be viewed as applying a single E step which justifies the use of the relative improvement of  $\mathcal{F}$  as a convergence measure.

In all experiments we used artificially generated data sampled from a randomly initialized  $k$ -component Gaussian mixture in  $d$  dimensions with a component separation of  $c$ . A separation of  $c$  means [11]:

$$\forall_{i \neq j} : \|m_i - m_j\|^2 \geq c^2 \cdot \max_{\{i,j\}} \{\text{trace}(C_i), \text{trace}(C_j)\}. \quad (15)$$

## 6.2 Difference with suboptimal shared responsibilities

As indicated by theory, the responsibilities computed from (10) are optimal w.r.t. maximizing  $\mathcal{F}$ . Since  $\theta$  is unaffected by the choice of responsibilities, we see from (4) that the sum of Kullback-Leibler divergences should be smaller when using (10) than when for example using  $p(s|\langle x \rangle_A)$ , as in [2]. In this experiment, we investigate the difference in summed Kullback-Leibler divergences (or equivalently the difference in  $\mathcal{F}$ ) between using (10) and the method of [2].

We generated 20 data sets of 5000 points from a random  $k = 20$  component Gaussian mixture in  $d = 10$  dimensions with a separation of  $c = 2$ . We then initialized a mixture using  $k$ -means and approximated the responsibilities for this mixture. Figure 1 shows the differences in the Kullback-Leibler divergence summed over all data points, for different partitioning sizes, averaged over the 20 data sets. Each unit on the horizontal axis corresponds to expanding the tree one level down, and hence doubling the number of nodes used in the partition.

The result confirms that our method indeed gives a better approximation, as predicted from theory. The

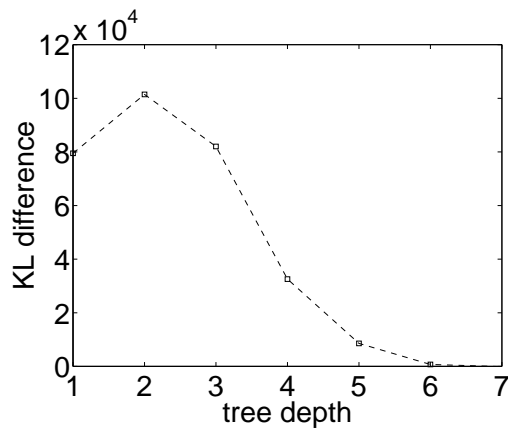


Figure 1: Differences in Kullback-Leibler divergence as a function of partition size.

difference is larger for rough partitionings, this is as expected since  $p(s|\langle x \rangle_A)$  becomes a better approximation of the individual responsibilities as the partition gets finer.

### 6.3 Gaussian mixture learning: Chunky vs. regular EM

In the second experiment we compared our method to the regular EM algorithm in terms of speed and quality of the resulting mixture. We looked at how difference in performance and speed is influenced by the number of data points, dimensions, components, and the amount of separation.

The default data set consisted of 10,000 points drawn from a 10-component 3-separated Gaussian mixture in 2 dimensions. To compare log-likelihoods we also created a test set of 1000 points from the same mixture. We applied both algorithms to a mixture initialized using  $k$ -means. Figure 2 shows the (differences in) negative log-likelihood and speed between the algorithms and the generating mixture averaged over 20 data sets. Speed was measured using the total number of basic floating point operations needed for convergence.

The results show with respect to the run-times that (i) the speedup is at least linear in the number of data points (ii) the number of dimensions and components have a negative effect on the speedup and (iii) the amount of separation has a positive effect. In general the Chunky EM requires more EM iterations to converge but it is still faster than regular EM since the iterations themselves are executed much faster. The difference in the performance of the two algorithms w.r.t. log-likelihood becomes larger (our method performs worse) as the number of components and data dimension increase. However, the difference in log-likelihood between Chunky and regular EM is still small compared to the respective differences with the

log-likelihood of the generating mixture, even using many components in high dimensional spaces.

## 7 Conclusions

We presented a variational EM algorithm that can be used to speed-up large-scale applications of EM to learn Mixtures of Gaussians. Our contributions over similar existing techniques are two-fold.

Firstly, we can show directly that the algorithm maximizes a lower bound on data log-likelihood and that the bound becomes tighter if we use finer partitionings. The algorithm finds mixture configurations near local optima of the log-likelihood surface which are comparable with those found by the regular EM algorithm, but considerably faster.

Secondly, because we have a convergent algorithm that maximizes a lower bound on data log-likelihood we can use any partitioning of the data. This allows us to use partitionings where the true posteriors differ heavily in each part, which might be needed when working on very large data sets and only limited computational resources are available. Another option is to start the algorithm with a rough partitioning (for which the EM iterations are performed very fast) and only refine the partitioning when needed because the algorithm converged with the rough partitioning.

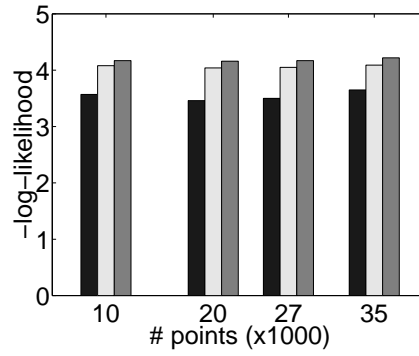
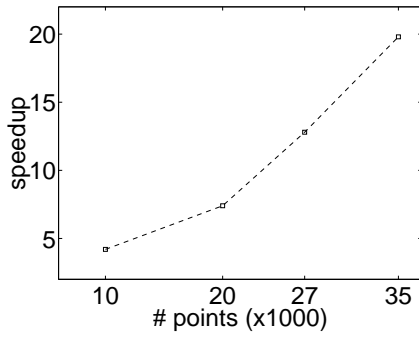
Comparing our work with [2], we conclude that with the same computational effort we can use the optimal shared responsibilities instead of the posterior at the node centroid. Furthermore, we obtained a provably convergent algorithm and have the freedom to use arbitrary partitionings of the data.

Current research involves integrating the Chunky EM with greedy Gaussian mixture learning [12, 13]. In greedy mixture learning, the mixture is build component-wise, starting from one component. After the EM for the  $k$ -component mixture converged, a new mixture component is added and EM cycles are performed on the resulting  $k + 1$  component mixture, etc. Finally, we would like to remark that the proposed technique can directly be applied to other Gaussian mixture models like e.g. the Generative Topographic Mapping [14] or the supervised mixture modeling [15].

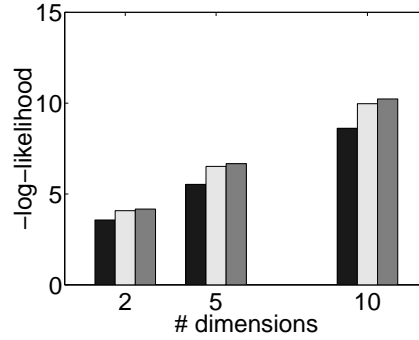
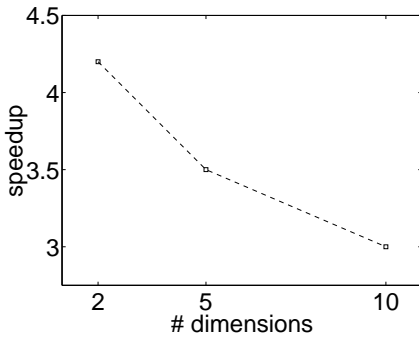
## Acknowledgements

This research is supported by the Technology Foundation STW (project nr. AIF 4997) applied science division of NWO and the technology program of the Dutch Ministry of Economic Affairs.

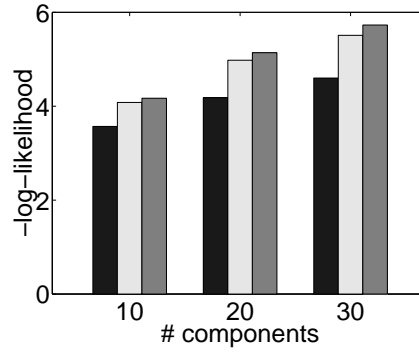
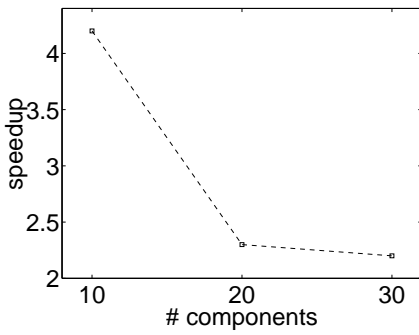
Number of datapoints (x1000):



Number of dimensions:



Number of components:



Amount of separation:

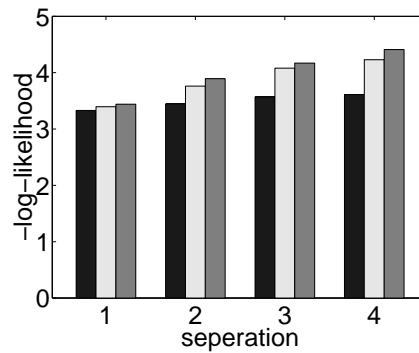
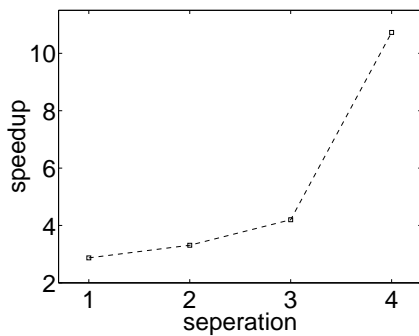


Figure 2: Experimental results on Gaussian mixture learning. (Left) Speedup factor. (Right) Negative log-likelihoods at convergence: generating mixture (black), regular EM (light), Chunky EM (dark).

## References

- [1] G. J. McLachlan and D. Peel. *Finite Mixture Models*. Wiley, New York, 2000.
- [2] A. Moore. Very fast EM-based mixture model clustering using multiresolution kd-trees. In M. Kearns and D. Cohn, editors, *Advances in Neural Information Processing Systems*, pages 543–549. Morgan Kaufman, 1999.
- [3] A. Moore and D. Pelleg. Accelerating exact k-means algorithms with geometric reasoning. In *Proc. of 5th Int. Conf. on Knowledge Discovery and Data Mining*, pages 277–281, 1999.
- [4] T. Kanungo, D. M. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Transactions PAMI*, 24:881–892, 2002.
- [5] K. Alsabti, S. Ranka, and V. Singh. An efficient  $k$ -means clustering algorithm. In *Proc. First Workshop High Performance Data Mining*, 1998.
- [6] R. M. Neal and G. E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M.I. Jordan, editor, *Learning in Graphical Models*, pages 355–368. Kluwer, 1998.
- [7] B. G. Lindsay. The geometry of mixture likelihoods: a general theory. *Ann. Statist.*, 11(1):86–94, 1983.
- [8] 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.
- [9] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, 1975.
- [10] R. F. Sproull. Refinements to nearest-neighbor searching in  $k$ -dimensional trees. *Algorithmica*, 6:579–589, 1991.
- [11] S. Dasgupta. Learning mixtures of Gaussians. In *Proc. IEEE Symp. on Foundations of Computer Science*, New York, October 1999.
- [12] N. Vlassis and A. Likas. A greedy EM algorithm for Gaussian mixture learning. *Neural Processing Letters*, 15(1):77–87, February 2002.
- [13] J. J. Verbeek, N. Vlassis, and B. J. A. Kröse. Efficient greedy learning of Gaussian mixture models. *Neural Computation*, 15(2):469–485, 2003.
- [14] C. M. Bishop, M. Svensén, and C. K. I Williams. GTM: The generative topographic mapping. *Neural Computation*, 10:215–234, 1998.
- [15] M. Titsias and A. Likas. Shared kernel models for class conditional density estimation. *IEEE Trans. on Neural Networks*, 12(5):987–997, 2001.