Accelerated greedy mixture learning

Jan Nunnink, Jakob Verbeek, Nikos Vlassis

Accelerated Greedy Mixture Learning

Jan R.J. Nunnink Jakob J. Verbeek Nikos Vlassis
Informatics Institute, Faculty of Science, University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
{*jnunnink,jverbeek,vlassis*}*@science.uva.nl*

Abstract

Mixture probability densities are popular models that are used in several data mining and machine learning applications, e.g., clustering. A standard algorithm for learning such models from data is the Expectation-Maximization (EM) algorithm. However, EM can be slow with large datasets, and therefore approximation techniques are needed. In this paper we propose a variational approximation to the greedy EM algorithm which offers speedups that are at least linear in the number of data points. Moreover, by strictly increasing a lower bound on the data log-likelihood in every learning step, our algorithm guarantees convergence. We demonstrate the proposed algorithm on a synthetic experiment where satisfactory results are obtained.

1 Introduction

An important task in data analysis is to search and identify clusters of data items. These clusters can be used for a number of purposes, such as classification of new data or for predicting missing data. Data clustering methods can also be used as learning algorithms when applied to partially labelled data, where each found cluster corresponds to a class.

The task of clustering algorithms is to fit a model to the data. A useful and popular class of models are Mixture models which are convex combinations of basic model components (McLachlan and Peel, 2000). Typically these components are Gaussian density functions, because often data is locally Gaussian distributed, in which case the data set can be assumed to have been generated by a hypothetical Gaussian mixture. The task is then to find the parameters of that generating mixture.

Because of their probabilistic nature, Gaussian mixtures are in principle preferred over models that partition a data set in discrete parts. In most applications where a new data item needs to be classified, it is more desirable to calculate the probability that this item belongs to certain clusters than to assign it to strictly one cluster.

The most popular algorithm for training a Gaussian mixture is the Expectation-Maximization (EM) algorithm (Dempster et al., 1977). Among its advantages are its easy implementation, no need to set extra user-defined parameters, and guaranteed monotone increase in model quality. There are also some downsides, however, the most important of which are its high dependency on initialization, and its computational complexity which is linear with respect to the size of the data set.

Over the past years several improvements of the EM algorithm were proposed to counteract those problems. One of them is a generalization of the EM algorithm, analogous to the (negative) variational free energy in statistical physics, that justifies several useful variants (Neal and Hinton, 1998). Also, some improvements aimed at lowering the computational cost by working with groups of data items instead of the items themselves (Moore, 1999) (In (Alsabti et al., 1998; Kanungo et al., 2002) this method is applied to the k-means algorithm). Finally, a greedy version of EM was proposed, which deals with the initialization problems of EM, thus resulting in higher quality models (Verbeek et al., 2003; Vlassis and Likas, 2002). This greedy method, however, is computationally more expensive.

In this paper we propose a variational approximation to the greedy EM algorithm for Gaussian mixtures proposed in (Verbeek et al., 2003) that offers a speedup without compromising stability. As in (Moore, 1999), we first partition the data and cache some statistics in each partition cell. A variational bound (Neal and Hinton, 1998) allows us to compute for each cell optimal responsibilities over mixing components (E-step), which are further used to update the mixture parameters (M-step). Both steps have cost that is independent of the size of the data set, and is linear in the number of partition cells. We demonstrate the proposed algorithm on a synthetic experiment where very satisfactory results are obtained. A more detailed treatment can be found in (Nunnink, 2003).

2 Gaussian mixtures and the EM algorithm

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

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

of d-dimensional Gaussian densities

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

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 $\{x_1, \ldots, x_n\}$ 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

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

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 (Lindsay, 1983).

3 The variational EM algorithm

Maximization of the data log-likelihood $\mathcal{L}(\theta)$ can be efficiently carried out by the EM algorithm (Dempster et al., 1977). In this work we consider a variational generalization of EM (Neal and Hinton, 1998) 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 θ 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) decompositions

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

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

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 Bayes' rule. The dependence of p on θ 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 $\mathcal{L}$ (Neal and Hinton, 1998). In the original derivation of EM (Dempster et al., 1977), each E step of the algorithm sets $q_i(s) = p(s|x_i)$ in which case, and for the current value θ^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:

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

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

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

from which we directly see the linear complexity of EM in the number n of datapoints.

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 algorithm is to assign equal responsibilities to chunks of data points that are nearby in the input space.

Consider a partitioning $\mathcal{P}$ of the data space into a collection of non-overlapping cells $\{A_1, \ldots, A_m\}$, such that each point in the data set belongs to a single cell. To all points in a cell $A \in \mathcal{P}$ 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 $\mathcal{F} = \sum_{A\in\mathcal{P}} \mathcal{F}_A$, 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

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

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 $\mathcal{F}_A$:

$$q_A(s) \propto p(s)\exp\langle\log p(x|s)\rangle_A \tag{10}$$

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[1]:

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

¿From this we see that the mean $\langle x\rangle_A$ and covariance $\langle xx^\top\rangle_A$ 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 θ. If we set the derivatives of (9) w.r.t. θ to zero we find the update equations:

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

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

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

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

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.

5 Greedy mixture learning

A recent approach to mixture learning involves building a mixture in a 'greedy' manner (Verbeek et al., 2003; Vlassis and Likas, 2002; Li and Barron, 2000). The idea is to start with a single component (which is trivial to find), and then alternate between adding a new component to the mixture and updating the complete mixture.

In particular, given a k-component Gaussian mixture $p_k(x)$ that has converged, the greedy method seeks a new component $\phi(x)$ with mean m_ϕ and covariance C_ϕ, and a mixing weight $a \in (0,1)$ that maximize the log-likelihood $\mathcal{L}_{k+1}$ of the *two*-component mixture

$$p_{k+1}(x) = (1-a)p_k(x) + a\phi(x; m_\phi, C_\phi), \quad (15)$$

where $p_k(x)$ is kept fixed. The advantages of greedy mixture learning are: (1) initializing the mixture is trivial, (2) local maxima of $\mathcal{L}$ are easier to escape, and (3) model selection becomes more manageable.

In (Verbeek et al., 2003), the search for a good component to add to $p_k(x)$ involves first splitting the data according to their 'nearest' (with highest posterior) component, then randomly generating a number of candidate components from the points in each subset, and finally maximizing $\mathcal{L}_{k+1}$ using only the data from the corresponding subset.

The same principle can also be applied in the case of pre-partitioned data sets. In particular, in component allocation we divide all cells $A \in \mathcal{P}$ into k disjoint subsets $\mathcal{P}_s$ according to their 'nearest' (with highest responsibility) component:

$$\mathcal{P}_s = \{A \in \mathcal{P} : s = \arg\max_{s'} q_A(s')\}. \quad (16)$$

Then we randomly split $\mathcal{P}_s$ in two sets of adjacent cells and for each such set $\mathcal{S}$ we generate a component $\phi(x; m_\phi, C_\phi)$ from the data contained in $\mathcal{S}$:

$$m_\phi = \frac{1}{n_\mathcal{S}} \sum_{A \in \mathcal{S}} n_A \langle x \rangle_A, \quad (17)$$

$$C_\phi = \frac{1}{n_\mathcal{S}} \sum_{A \in \mathcal{S}} n_A \langle x x^\top \rangle_A - m m^\top, \quad (18)$$

where $n_\mathcal{S}$ is the total number of points in $\mathcal{S}$. Note that both m_ϕ and C_ϕ can be calculated without requiring the data points themselves. Subsequently we update (a, m_ϕ, C_ϕ) in (15) by maximizing a lower bound of $\mathcal{L}_{k+1}$ using only the cells in $\mathcal{P}_s$. (Cells outside $\mathcal{P}_s$ will not contribute significantly to the bound.) Let q_A be the responsibility of the new component $\phi(x; m_\phi, C_\phi)$ for any cell $A \in \mathcal{P}_s$, and $1 - q_A$ the responsibility of the old mixture p_k. The free energy (9) for cell A under the two-component mixture (15) then reads

$$\mathcal{F}_A^{k+1} = n_A q_A \Big[\log \frac{a}{q_A} + \langle \log \phi(x) \rangle_A \Big] + n_A(1-q_A) \Big[\log \frac{1-a}{1-q_A} + \langle \log p_k(x) \rangle_A \Big]. \quad (19)$$

Since $\mathcal{F}_A^k$ is a lower bound on $\sum_{x \in A} \log p_k(x)$ which we have already computed from (9), we can replace the latter in (19) to get the bound

$$\mathcal{F}_A^{k+1} \geq n_A q_A \Big[\log \frac{a}{q_A} + \langle \log \phi(x) \rangle_A \Big] + n_A(1-q_A) \Big[\log \frac{1-a}{1-q_A} + \frac{\mathcal{F}_A^k}{n_A} \Big]. \quad (20)$$

In the E-step we compute the optimal q_A for each cell $A \in \mathcal{P}_s$ by setting the derivative of (20) w.r.t. q_A to zero. This gives:

$$q_A = \frac{a \exp \langle \log \phi(x) \rangle_A}{(1-a)\exp(\mathcal{F}_A^k / n_A) + a \exp \langle \log \phi(x) \rangle_A}, \quad (21)$$

where $\langle \log \phi(x) \rangle_A$ can be computed fast using (11). Similarly, in the M-step we maximize $\mathcal{F}_\mathcal{P}^{k+1} = \sum_{A \in \mathcal{P}} \mathcal{F}_A^{k+1}$ using the q_A found in (21).

As in (Verbeek et al., 2003), we set the responsibility of the new component for all cells outside $\mathcal{P}_s$ to zero, in which case it is not difficult to see that we get the following update equations:

$$a = \frac{\sum_{A \in \mathcal{P}_s} n_A q_A}{n}, \tag{22}$$

$$m_\phi = \frac{\sum_{A \in \mathcal{P}_s} n_A q_A \langle x \rangle_A}{na}, \tag{23}$$

$$C_\phi = \frac{\sum_{A \in \mathcal{P}_s} n_A q_A \langle x x^\top \rangle_A}{na} - m_\phi m_\phi^\top. \tag{24}$$

Note that the sums run over cells in $\mathcal{P}_s \subset \mathcal{P}$. We refer to (Nunnink, 2003) for more details.

6 Choosing a partition

The analysis presented in the previous sections applies to any partition, as long as sufficient statistics of the data have been stored in the corresponding cells. As we showed above, for *any* partition we obtain a convergent algorithm that strictly increases in each step a lower bound on the data log-likelihood. Moreover, by refining a given partition $\mathcal{F}$ cannot decrease, and in the limit holds $\mathcal{F} = \mathcal{L}$ making the approximation bound tight. Clearly, various trade-offs can be made between the computational cost and the approximation quality.

A convenient structure for storing statistics in a way that permits the use of different partitions in the course of the algorithm is a kd-tree (Bentley, 1975; Moore, 1999). This is a binary tree in which the root contains all data points, and each node is recursively split by a hyperplane that cuts through the data points contained in the node. Typically, axis-aligned hyperplanes are used for splitting nodes. In our experiments we used hyperplanes that cut along the bisector of the first principal component of the points in the node, leading to irregularly shaped cells (Sproull, 1991). As in (Moore, 1999), we store in each node of the kd-tree the sufficient statistics of all data points under this node. Building the kd-tree and storing statistics in its nodes has cost $O(n \log n)$, but this needs to be done only once at the beginning of the algorithm.

The outer nodes of a given expansion of the kd-tree form a partition $\mathcal{P}$ of the data set. Further expanding the tree means refining a current partition. 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 (Moore, 1999), a bound on the size of the partition (number of outer nodes at any step), or sampled approximations of the difference between log-likelihood and $\mathcal{F}$. Another possibility, which we adopted in our experiments, is to control the tree expansion based on the performance of the algorithm, that is, we refine a partition only if this (significantly) improves the value of $\mathcal{F}$.

7 Related work

The idea of using a kd-tree for accelerating the EM algorithm in large-scale mixture modeling was first proposed in (Moore, 1999). 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., $q_A = p(s|\langle x \rangle_A)$. (Compare this with (10).) If there is little variation in the posteriors within a node, which is in turn achieved by having relatively fine partitions, the approximation $q_A = p(s|\langle x \rangle_A)$ will only slightly affect the update in the M-step and therefore this will probably increase the data log-likelihood. However this is not guaranteed. Also, in (Moore, 1999) 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 require solving a quadratic programming problem).

The main advantage of our method compared to (Moore, 1999) is that our algorithm strictly increases in each 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 use even rough partitions. As mentioned above and as demonstrated in the experiments below, by gradually refining the partition while running the algorithm we can get close to the optima of the log-likelihood in relatively few steps.

8 Demonstration

In a synthetic experiment we compared the greedy EM algorithm described in (Verbeek et al., 2003) with the proposed accelerated greedy algorithm, for learning a k-component Gaussian mixture. We used artificially generated data sampled from a randomly initialized k-component Gaussian mixture in d dimensions with a component separation of c (Dasgupta, 1999). For each data set we built a kd-tree and stored in its nodes the data statistics as explained above.

We started the accelerated greedy EM algorithm with a partition of size four and expanded the tree one node at a time, best first, as in the first experiment. We used a default data set of 10,000 points and a test set of 500 points drawn from a 5-component 2-separated mixture in two dimensions. In Fig. 1 we show the results averaged over 20 trials. The accelerated greedy algorithm is always faster, with a speedup that is linear in the size of the data set. Moreover, this speedup comes almost 'for free': the log-likelihoods of both algorithms are practically equal to that of the generating mixture.

9 Conclusions

We have presented a variant of the Expectation-Maximization algorithm specifically designed for dealing with large-scale data sets, without paying much in terms of quality. The experiment described above clearly shows that the proposed algorithm is faster than previous versions and that this speedup is linear with respect to the size of the data set. The experiment also shows that there is barely any loss in quality even when working with large or complex data sets.

The theory shows that the effect of the EM algorithm, namely the guaranteed monotone increase in model quality with respect to the free energy, still holds independent of the size of the partition. Therefore one can use the size of the partition as an easy to implement trade-off between speed and quality. This is useful when there are, for example, only limited resources available.

Acknowledgements

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

References

K. Alsabti, S. Ranka, and V. Singh. 1998. An efficient k-means clustering algorithm. In *Proc. 1st Workshop on High Performance Data Mining*.

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

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

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

T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu. 2002. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(7):881–892.

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

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

G. J. McLachlan and D. Peel. 2000. *Finite Mixture Models*. John Wiley & Sons.

A. W. Moore. 1999. Very fast EM-based mixture model clustering using multiresolution kd-trees. In *Advances in Neural Information Processing Systems 11*. The MIT Press.

R. M. Neal and G. E. Hinton. 1998. 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 Academic Publishers, Dordrecht, The Netherlands.

J. R. J. Nunnink. 2003. Large scale Gaussian mixture modelling using a greedy Expectation-Maximisation algorithm. Mas-

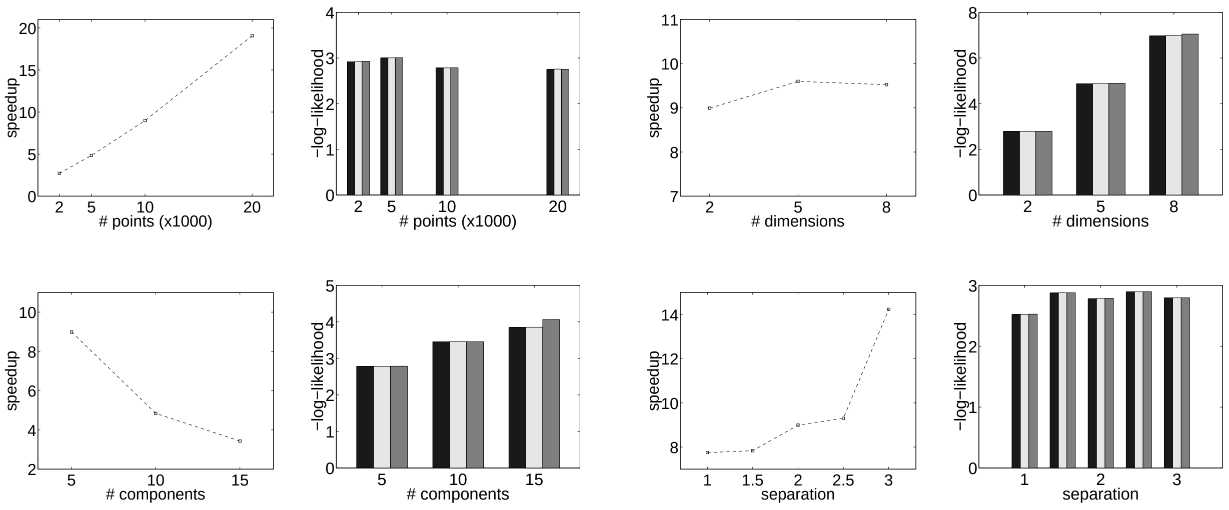

Figure 1: Greedy mixture learning with the algorithm in (Verbeek et al., 2003) vs. the accelerated greedy EM. The graphs show the speedup factor and the bar charts show the negative log-likelihood: generating mixture (black), greedy EM (light), accelerated greedy EM (dark).

ter's thesis, Informatics Institute, University of Amsterdam, May.

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

J. J. Verbeek, N. Vlassis, and B. Kröse. 2003. Efficient greedy learning of Gaussian mixture models. *Neural Computation*, 15(2):469–485, February.

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