An Accelerated Algorithm for Density Estimation in Large Databases Using Gaussian Mixtures

Alvaro Soto[A1], Felipe Zavala[A1], and Anita Araneda[A2]

e-mail: [fzavala,asoto]@ing.puc.cl aaraneda@mat.puc.cl

[A1]Department of Computer Science

[A2]Department of Statistics

Pontificia Universidad Católica de Chile

Casilla 306 - Santiago 22 - Chile

Abstract

Today, with the advances of computer storage and technology, there are huge datasets available, offering an opportunity to extract valuable information. Probabilistic approaches are specially suited to learn from data by representing knowledge as density functions. In this paper, we choose Gaussian Mixture Models (GMMs) to represent densities, as they possess great flexibility to adequate to a wide class of problems. The classical estimation approach for GMMs corresponds to the iterative algorithm of Expectation Maximization. This approach, however, does not scale properly to meet the high demanding processing requirements of large databases. In this paper we introduce an EM-based algorithm, that solves the scalability problem. Our approach is based on the concept of data condensation which, in addition to substantially diminishing the computational load, provides sound starting values that allow the algorithm to reach convergence faster. We also focus on the model selection problem. We test our algorithm using synthetic and real databases, and find several advantages, when compared to other standard existing procedures.

1 Introduction

Today, when computers have the power to store and search huges amounts of data, the process of learning from data is receiving considerable attention (**?**). This is specially relevant, as learning can be applied to diverse areas, such as business, science, and engineering. In particular, our research is inserted in the context of a larger problem consisting of the detection of rare objects in huge astronomical databases. The work we present here focuses on the development of an efficient algorithm to estimate joint probability distributions, that will be subsequently used to estimate a Bayes Network representing the structure of the data.

There exists in the literature a wide variety of deterministic and probabilistic approaches to learning. Probabilistic approaches possess the advantage over deterministic ones, in that they provide a well known measure of uncertainty. This is attained by representing knowledge by density functions that can be learned from data. Problems benefiting from density estimation are, among others, clustering and estimation of Bayes Networks.

In our research with astronomical databases, the problem is understood as detecting points located in areas of low density of the underlying distribution.

We model the data through Bayes Networks, and we find that the process of density estimation in each node accounts for almost 90% of the total computational load. Most traditional approaches to solve the outlier detection problem, lack of good scalability properties, therefore, they are not applicable to our problem.

The problem of density estimation corresponds to learning the probabilistic structure of the process that generates the data. Gaussian Mixture Models (GMMs) (**?**) appear as an appealing family of models, as their form is flexible enough to accommodate to a large variety of distributions. GMMs also posses the advantage of providing insight about the mechanism that generates the data, as well as a sound interpretation of the parameters involved. Today, GMMs are widely used in applications such as target tracking, pattern recognition, and classification, among others.

In particular, the density function of a given data point in a dataset, $x \in \Re^d$, corresponds to a GMM with K components if it can be expressed as

$$p(x) = \sum_{h=1}^{K} w_h \; f(x|\mu_h, \Sigma_h),$$

where $f(\cdot|\mu_h, \Sigma_h)$ corresponds to a d-dimensional Gaussian distribution with parameters μ_h and Σ_h. Each term w_h, commonly called the "weight of the h-th distribution" corresponds to the relevance of the h-th Gaussian within the model, and it is subject to $\sum_h w_h = 1$.

The interpretation of GMMs data generation is as follows. We choose one of K Gaussians, with probabilities $w_h, h = 1, \ldots, K$. We generate, afterwards, an observation from the Gaussian distribution selected in the first step. In other words, the "world" is composed by K Gaussian distributions, whose sizes are proportional to their weights in the mixture.

The Expectation Maximization (EM) algorithm (**?**) is the common tool to estimate the parameters in a GMM, $\theta = (\mu_h, \Sigma_h, w_h, h = 1, \ldots, K)$. The EM algorithm aims to the maximization of the density function of the data as a function of the θ, which is called the likelihood function.

The EM algorithm is an iterative procedure. At each iteration, it uses its current estimates of the parameters to compute an indicator of how each Gaussian explains each observation (E-Step). Afterwards, it uses these indicators to find updated estimations of the parameters in θ (M-Step).

Despite its popularity to estimate GMMs, EM presents a number of deficiencies when applied to large databases. EM is guaranteed to converge to a maximum of the likelihood function, however, depending on the starting point, this maximum may correspond to a local one instead of the global maximum. To overcome this problem, the algorithm is usually run a number of times, using different starting points. This solution is, however, unfeasible when dealing with large high dimensional databases, due to the considerable computational load of the process. In effect, the computational complexity of each iteration of EM is $O(N\ K\ d^2)$, where N is the number of points in the database, K the number of Gaussians, and d the dimensionality of each data point.

An additional problem of the EM algorithm when applied to large databases arises when we intend to perform model selection. The EM algorithm assumes that the number of Gaussian distributions involved in the mixture is fixed, and thus this number is not included in the set of parameters to be updated during iterations. Traditional approaches estimate the number of Gaussians by running the algorithm a number of times, using different a priori defined number of Gaussians. The fitted models are compared afterwards using some relevant goodness of fit indicator. These procedure is again unfeasible when dealing with large databases.

In this work we propose a new algorithm aiming to overcome the deficiencies above mentioned. Our approach in based on using a pre-clustering technique that creates a condensed data tree. A first advantage of using this approach is that the condensed data provides a sound starting value to initialize the EM algorithm. This decreases the chances of reaching a local maximum instead of the global one, while also needing fewer iterations to converge to the maximum. A second advantage corresponds to the summarization of the data and thus, EM does not need to sweep over all the observations during iterations, but just over these summaries.

In terms of model selection, we propose to use an accepting-rejection criterion to visit models with different number of Gaussian components. This algorithm does not need to restart the estimation procedure every time it moves to a different number of components model. Instead, it provides a way to decide when to accept or reject the proposed model after the execution of a few iterations of the EM algorithm. We obtain a further acceleration of the algorithm by introducing a new incremental convergence property.

This paper is organized as follows. In Section 2 we discuss in more detail the main problems of density estimation using EM, and how previews works have dealt with these problems. In Section 3 we present our algorithm to accelerate EM and to solve the model selection problem. In Section 4 we present the results of applying our approach to synthetic and real databases. Finally, in Section 5 we present the main conclusions of this work.

2 Previous Work

Previous works have tried to improve the starting value and scalability problems of EM using special data structures (?), making just one revision of the dataset (?), or limiting the size of the data to be considered at each iteration (?). However, in the last case, Hulten (?) shows cases where sampling is more expensive than using the complete database.

There are a set of approaches exploiting the non-uniform spread of data over space (?), (?), (?). In effect, data usually present high density regions, allowing a simplification of the problem by exploiting the data configuration through data condensation. In other words, similar data points are represented by a single structure that summarizes the relevant qualities of the group. An excellent way to condensate data is using sufficient statistics, which in the case of GMMs correspond to

$$\sum x_i, \qquad \sum x_i \, x_i^T, \tag{1}$$

as well as the number of observations in each group, and where summations include all data points in a given group.

Omohundro (?) (?) unifies different types of tree structures under the concept of Bumptree, based on the idea that each node carries information about its children. Omohundro develops efficient algorithms to construct these trees, showing that their use improves the speed of converge of the EM algorithm.

Moore (?) creates a fast algorithm to build balltrees exploiting the capabilities of KD-trees (?), which contain all the data points and the sufficient statistics in each node. Although KD-trees prove to scale well in the number of observations, this is not true when increasing the dimensionality of the data.

Bradley (?) condensates the data points in a incremental way which is highly efficient, but once a decision has been made, there is no way to go back, so when all the data has been processed not further improvements can be done. Zhang (?) creates a CF-tree which corresponds to a condensed data tree. This approach provides a method to condensate data in a incremental way without the need of keeping all the data points. A drawback of this data structure is that all leaves in the tree are hyperspheres of the same radius, which makes them inefficient as some of them may include regions with sparse or no data.

A common approach to the model selection problem is to compare models with different number of Gaussians through the value of their fitted likelihoods. These quantities may, however, be artificially increased by means of introducing additional number of Gaussians to the model. One way to overcome this problem is to consider, instead, the Bayesian Information Criterium (BIC). BIC evaluates the fit of a model through the likelihood function penalizing for additional model complexity, i.e., the excessive number of Gaussians. The BIC statistic is computed as

$$BIC(\theta) = \log(\Lambda(X;\theta)) - \frac{\log(N)}{2}\, p,$$

where $\Lambda(X;\cdot)$ corresponds to the likelihood function, and p corresponds to the number of parameters in the model. Other criteria for model comparison have been developed, as the *Maximum a Posteriori Probability* (MAP) (?), the *Minimum Description Length* (MDL) (?), and the *Bayesian Ying-Yang* (BYY) (?). Most previous works, however, suggest better behavior of the BIC statistic over the others.

Using the BIC model selection criterion, it is possible to compare multiple fitted models in order to choose the best one. This is not a trivial task, however, when considering large datasets. The X-means algorithm (?) incrementally modifies a model guided by BIC. In order to do so, it uses K-means (?) as a fast clustering technique. The main restriction of the X-means algorithm is that it needs the specification of a range for the number of components to perform the search. Hamerly (?) shows a similar algorithm that uses a statistical test based on a projection of all dimensions of the data over a single dimension. His results seem to be better than the results obtained by X-means, however, the projection of all dimensions onto a single one may be questioned. A further step is taken by Sand (?) who proposes KD-Clust, an algorithm based on the idea of accepting or rejecting potential models obtained by splitting or deleting existing Gaussians during EM iterations. The results obtained by Sand are promising but a main restriction corresponds to the number of EM iterations needed to complete the task.

3 Our Approach

Our approach is based on the idea of summarizing data which is usually called data condensation. The process of creating condensed data, however, is still expensive an thus, we must ensure that every calculation results in benefits for the estimating algorithm. With this goal in mind, we evaluate several data structures to condensate data, such as KD-trees, Anchor Hierarchy (?), and Zhang version of CF-trees. Comparing these approaches, we choose to build our solution upon the idea of CF-trees, as they offer a good trade off between quality of the results and computational load.

3.1 Data Structure

To improve the performance of our CF-tree like algorithm, we introduce some features that take advantage of the tree structure. In particular, we select the most relevant part of the tree for each Gaussian. In effect, our tree structure reduces the number of nodes to be used for each Gaussian by identifying the specific ones concerned to its domains. To do so, each Gaussian crosses the tree in a Depth First Search way (**?**) and prunes the tree when it is no longer useful to go deeper. When the likelihood of a node under a given Gaussian is above a certain threshold, the algorithm goes down further in the tree. Otherwise, the algorithm stops. The last node is kept to represent their children.

As we mentioned earlier, all the leaves in a CF-tree are hyper-spheres with constant radius. In our implementation, each leaf is initialized with a radius that is proportional to the variance in each dimension, avoiding volumes containing little data in them. In order to do so, we compute the range of each attribute and we define the radius of each leaf as:

$$r \propto \ (Max - Min),$$

where *Max* and *Min* are the vectors containing maximum and minimum values in each dimension.

Our data structure provides two desirable features to our algorithm that enables it to scale to large databases. The first feature corresponds to the direct consequence of condensing data. An adequate summary of the relevant information allows the algorithm to run iterations with lower computational load.

A second feature arises as a by-product. As we mentioned in Section 1, avoidance of convergence to local maximums is usually obtained by running the algorithm a number of times, using different starting values for the parameters. Since this is unfeasible when dealing with large databases, we need the availability of a sound starting point. The tree structure obtained through condensation provides information in order to do so. If K components are required, we randomly choose K non-repetitive *end nodes* from the tree, where an *end node* corresponds to the deepest internal node of the tree. This means that each *end node* represents a set of leaves. The parameters of the Gaussians considered are obtained through the sufficient statistics presented in Eq. (1).

3.2 Convergence of Gaussian Components

As an additional strategy to diminish the computational load of the algorithm, we track the change of each Gaussian component. If a Gaussian component does not change substantially between successive iterations, we stop updating it. This brings a reduction in the number of components to be considered in subsequent iterations, therefore providing a faster algorithm.

Changes in the components are evaluated through the *Kullback-Leibler* (KL) divergence between two density functions (**?**), which in the case of Gaussian distributions corresponds to

$$\frac{1}{2}(log\frac{|\Sigma_2|}{|\Sigma_1|} + Tr(\Sigma_2^{-1}\Sigma_1) + (\mu_1 - \mu_2)^T\Sigma_2^{-1}(\mu_1 - \mu_2) - d),$$

where μ_i and Σ_i correspond to the parameters of the Gaussians to be compared, $i = 1, 2$, and Tr corresponds to the trace function.

3.3 Merging, Splitting and Deleting Gaussians

In order to perform model selection during EM iterations, we create an algorithm based on KD-Clust (**?**). The main idea of KD-Clust is the proposal of splitting or deleting existing Gaussian components. One of these jumps, split or delete, is proposed in a given iteration of the algorithm, and evaluated in a couple of subsequent iterations. The algorithm decides where to stay with the previous model or to adopt the proposed one based on the results of those iterations.

We improve the KD-Clust algorithm by adding a merging action. This action consists of merging the two most similar Gaussians into one that represents both of them. Again, we use the KL divergence as a measure of similarity.

In particular, our overall split-delete-merge algorithm corresponds to:

1. With probability p_{jump} an action is taken. If so, go to 2. Else go to 4.
2. Save state of the GMM. Choose to merge, split or delete with probabilities p_{merge}, p_{split} and p_{delete}, respectively. Go to 3.
3. Run EM $few-iterations$ and compare the new fit with the previous one using BIC criterion. Choose the best model and go to 1 for a new possible jump.
4. Run EM $few-iterations$ and return to 1 for a new possible jump.

In our study we run the algorithm proposing jumps every 5 and 10 iterations, obtaining similar results under both schemes.

4 Applications of the Algorithm

In this section we compare the results obtained by our algorithm, that we call an accelerated EM algorithm to estimate GMMs, AGMM-EM. We compare AGMM-EM to the results of popular existing algorithms, as are regular EM, KD-Clust (?), and FASTEM (?).

We first present the results obtained by the algorithms when applied to synthetic datasets, generated to evaluate different aspects of the algorithms. To evaluate the scalability of our algorithm with respect to data size and dimensionality, we generate datasets with the same number of Gaussian components, $K = 10$, but increasing the number of observations and dimensions, respectively.

As for model selection, we compare the algorithms when applied to datasets with $K = 10$ Gaussian components. Considering that the ability of an algorithm to detect the right number of Gaussians may depend on how far apart these Gaussians are, we worked with three kind of databases: little, medium, and considerable overlap between Gaussians.

We also show the results obtained by our algorithm when applied to two real databases. The first of them contains several records related to metallic parts, where the main goal corresponds to the detection of faulty pieces. The second database contains records related to astronomical objects, where the main goal corresponds to the detection of anomalies or rare objects.

4.1 Scalability of the Algorithm

Our first experience is designed to evaluate the scalability of our algorithm when compared to a regular implementation of EM. Since regular EM does not include a model selection criteria, in this experience we assume that the number of Gaussian components is known, $K = 10$. We use databases with little overlap.

The starting values for the regular EM algorithm are chosen randomly, while our AGMM-EM algorithm is initialized using the starting points suggested by our tree structure. The time to build the tree is added to the first iteration of the algorithm. Both algorithms are run during 100 iterations.

The quality of the resulting estimated models is shown in Table 1 through their log-likelihood values. The table shows that our algorithm obtains larger likelihoods than regular EM, ensuring that our fitted models are as good or better than the models obtained by EM. We infer that this improvement in quality of the estimated model is due to a better selection of the starting values.

The scalabilities of EM and AGMM-EM algorithms are compared in Figure 1. The figure shows that the additional time needed by EM, compared to that of AGMM-EM, increases as data size does. Specifically, in the case of datasize of 400K, EM takes 64 times the time needed by AGMM-EM to perform 100 iterations.

As for scalability in the number of dimensions, Figure 2 compares the speed of the EM algorithm when using a data structure that considers all leaves, to AGMM-EM using selected leaves only. The figure shows a faster behavior of AGMM-EM. Speed differences between the two algorithms increase as the dimensionality of the data does.

4.2 Model Selection of the Algorithm

Our model selection approach is built upon KD-Clust algorithm and thus, we define KD-Clust as the reference algorithm. Performance is evaluated as the quality of the estimated models in terms of their BIC statistic and their estimated number of Gaussian components.

We test these algorithms with the three types of datasets, having little, medium, and large overlap. We generated bidimensional data with $K = 10$ Gaussian components. Both algorithms are run during 100 iterations.

Considering that both algorithms present random steps when choosing changes between the number of components, we run each algorithm several times using the same databases, and we report here an average of the results.

The results of this experience are shown in Table 2. The table shows that although our AGMM-EM algorithm presents slightly lower BICs, it provides better estimation of the true number of Gaussian components, $K = 10$.

4.3 Speed to Reach an Adequate Model

This experiment is designed to compare our AGMM-EM algorithm against a well known accelerated algorithm that performs model selection at the same time. For this purposes we chose FASTEM (**?**) as the reference algorithm. As in previous experiences, we work with databases with $K = 10$ Gaussian components. All datasets are created with intermediate overlap. Initial values for FASTEM are chosen according to the original implementation of the algorithm. Initial values for AGMM-EM are chosen based on our tree structure.

We start by comparing the scalability of the two algorithms. Both algorithms were run during 100 iterations and the amount of time required by each of them is shown in Figure 3. The figure shows that AGMM-EM performs faster than FASTEM and that the difference between both algorithms increases with the number of dimensions.

As for the quality of the results, Table 3 shows the BIC statistics and the number of Gaussian found by the algorithms. The table shows both, better BICs and better estimates of the number of components, for AGMM-EM over FASTEM.

A last experience was performed to evaluate the speed of the algorithms to perform good model selection. In order to do so, we compare the qualities of the estimated models at each iteration, using the BIC statistic. Figure 4 shows that AGMM-EM finds good models during the first iterations, although the figure suggests that both algorithms perform similarly when increasing the number of iterations. As we showed before, the advantage of AGMM-EM consists of its higher speed to perform the iterations.

4.4 Performance with two real databases

We test our algorithm in the estimation of the structure and probability densities of a Bayes Networks, where the joint distributions of the nodes of the network are assumed to follow GMMs. We estimate the structure of the network using an scoring based approach. As described earlier, we work with two databases. One of them contains records about metallic parts. The other database contains records about astronomical objects. In both experiences, the estimation of GMMs consumed almost 90% of the total time needed to build the Bayes network, showing the importance of an efficient procedure to estimate GMMs.

In our test, the estimation of the Bayes Networks is performed using both, GMM trained with regular EM and with the AGMM-EM algorithm. The times needed by both algorithms to reach convergence are shown in Table 4. Considerable savings of time are obtained when using AGMM-EM.

The quality of the estimated models can also be evaluated through the accomplishment of our final goal: to detect potential anomalies in the databases. Here we assume that the anomalies are given by the data point with the lowest likelihood. In particular, in the metallic parts database, we know that it contains a total of 60 anomalous data points (faulty metallic parts). A Bayes Network estimated using the regular EM ranked those failures among the first 1,500 observations with the lowest likelihoods, out of approximately 23,000 observations in the database. On the other hand, AGMM-EM ranked those failures among the first 1,000 observations with the lowest likelihoods, showing a better performance than regular EM. Furthermore, AGMM-EM was able to rank 95% of the failures among the 500 lowest likelihood observations.

5 Conclusions

In this paper we propose two features to enhance the behavior of regular EM algorithm applied to GMMs in large databases. Our first contribution consists of the use of a new improved version of CF-trees to condensate data observations. Our condensation algorithm shows to be more efficient than CF-trees. This is accomplished by both, pruning the tree and allowing leaves to have different radius. Acceleration of the overall estimation procedure is then obtained in two ways. The first is a direct consequence of designing a successful condensation algorithm, as we are able to summarize the data through sufficient statistics without losing relevant information. On the other hand, our condensation algorithm provides sound starting values for both, the parameters that are to be updated by EM, and the number of Gaussians used.

In our experiences, we find that AGMM-EM largely outperforms regular EM and FASTEM in terms of their speed to perform a given number of iterations. Also, our results indicate that the gap between the times needed in each iteration by AGMM-EM and regular EM, and AGMM-EM and FASTEM, increases when increasing the number of observations and dimensions, respectively.

Our second contribution relates to model selection. We propose a model selection strategy that is able to evaluate models with different number of Gaussian components, during iterations of a same run of the algorithm. Our strategy introduces a new merging operator to an existing algorithm, KD-Clust. Furthermore, our algorithm overcomes several deficiencies of KD-Clust. In effect, the decision between two models is more efficient as it can be achieved in less number of iterations that executes faster. Our results indicate that AGMM-EM achieves better estimates of the true number of Gaussian components than KD-Clust does.

The highlighted features of our algorithm enables it to be used in a wider range of applications, as it provides good results in less time. In particular, the application of the algorithm to two real database to estimate the structure of a Bayes Network resulted in very encouraging results.

Acknowledgments

This work was partially funded by FONDECYT grant 1030336.

Table 1: Log-likelihood achieved by EM and AGMM-EM, after 100 iterations.

Size	Loglikelihood	
	Regular EM	AGMM-EM
100K	-256.571	-234.452
200K	-547.910	-526.537
300K	-825312	-817.495

Table 2: Number of components and BIC statistics obtained by KD-Clust and AGMM-EM algorithm. $K = 10$ components, 100 iterations.

Overlap	KD-Clust		AGMM-EM	
	K	BIC	K	BIC
Minimum	11.7	-223759	10.3	-227192
Medium	14.0	-271716	10.0	-273848
Large	13.3	-292035	10.0	-298618

Table 3: Estimated number of components and BIC statistic for AGMM-EM and FASTEM. True number of components $K = 10$.

Dim	FASTEM		AGMM-EM	
	K	BIC	K	BIC
3	11	-155904	9	-155626
4	19	-173650	11	-172409
5	11	-205471	9	-203266

Table 4: Times needed by regular EM and AGMM-EM to construct a Bayes Network.

	Algorithm	
Database	Regular EM	AGMM-EM
Metallic parts	2 hrs	25 min
Astronomical objects	5 days	251 min

Figure 1: Time required by regular EM (solid line) and AGMM-EM (dotted line) to perform 100 iterations, vs. size of the database.

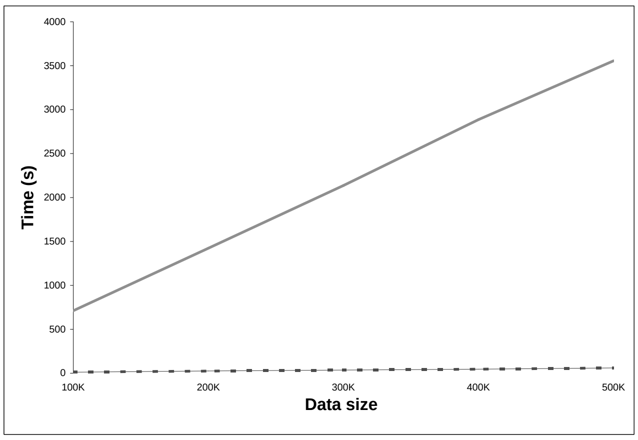


Figure 2: Time required by EM using all leaves (solid line) and AGMM-EM using selected leaves (dotted line) to perform 100 iterations, vs. dimensionality of the database.

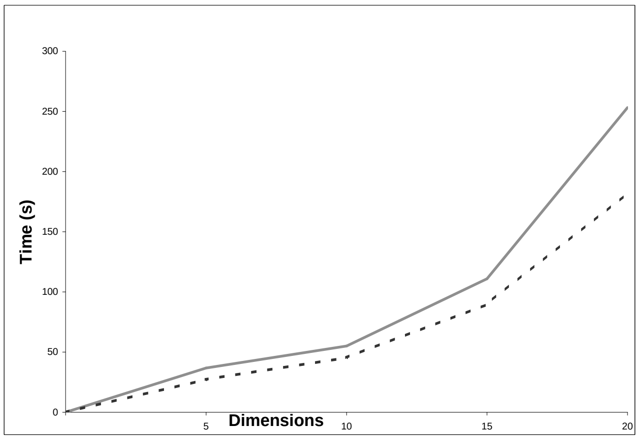


Figure 3: Time required by FASTEM (solid line) and AGMM-EM (dotted line) to perform 100 iterations, vs. dimensionality of the database.

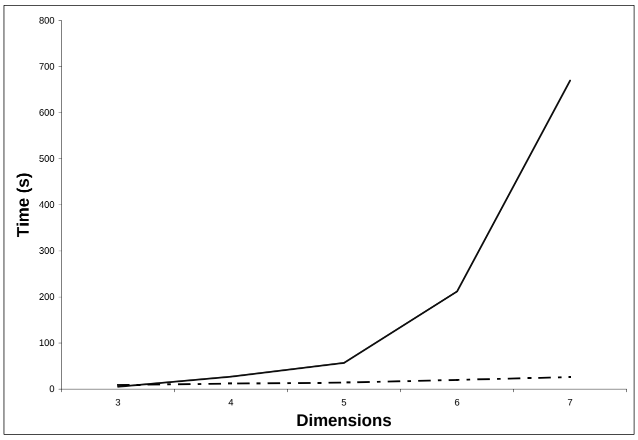
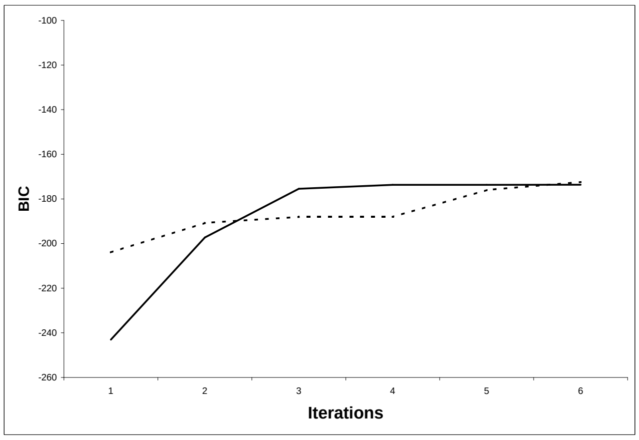


Figure 4: BIC statistic of the models obtained by FASTEM (solid line) and AGMM-EM (dotted line) vs. number of iterations.


BIC
Iterations