

Simple Methods for Initializing the EM Algorithm for Gaussian Mixture Models

Johannes Blömer

Department of Computer Science
University of Paderborn
33098 Paderborn, Germany
Email: bloemer@uni-paderborn.de

Kathrin Bujna

Department of Computer Science
University of Paderborn
33098 Paderborn, Germany
Email: kathrin.bujna@uni-paderborn.de

Abstract—In this paper, we consider simple and fast approaches to initialize the Expectation-Maximization algorithm (EM) for multivariate Gaussian mixture models. We present new initialization methods based on the well-known K -means++ algorithm and the Gonzalez algorithm. These methods close the gap between simple uniform initialization techniques and complex methods, that have been specifically designed for Gaussian mixture models and depend on the right choice of hyperparameters. In our evaluation we compare our methods with a commonly used random initialization method, an approach based on agglomerative hierarchical clustering, and a known, plain adaption of the Gonzalez algorithm. Our results indicate that algorithms based on K -means++ outperform the other methods.

I. INTRODUCTION

Gaussian mixture modelling is an important task in the field of clustering and pattern recognition. There are various approaches to estimate the parameters of a mixture such as graphical methods, moment matching, Bayesian approaches, and the method of maximum likelihood estimation (MLE). A widely used approach for the latter problem is the Expectation-Maximization (EM) algorithm [1]. Starting with some initial mixture model, the EM algorithm alternates between computing a lower bound of the log-likelihood and improving the current model with respect to this lower bound. The algorithm converges to a certain stationary point of the likelihood function. Unfortunately, the likelihood function is generally nonconvex, possessing many stationary points, including small local maxima, and even worse, local minima and saddle points (cf. [2], [3]). Moreover, the convergence of the EM algorithm to either type of point highly depends on the chosen initial model.

A. K -MLE for Gaussian Mixture Models

A Gaussian mixture model (GMM) is a parametric probability density function represented as a weighted sum of Gaussian densities. We denote the parameter vector defining a mixture of K Gaussians over \mathbb{R}^D by $\theta = \{(w_k, \mu_k, \Sigma_k)\}_{k=1, \dots, K}$, where $w_k \in \mathbb{R}$ is the mixing weight ($\sum_{k=1}^K w_k = 1$), $\mu_k \in \mathbb{R}^D$ is the mean and $\Sigma_k \in \mathbb{R}^{D \times D}$ is the covariance of the k -th mixture component. The probability density function (pdf) of a mixture given by θ is defined by $\mathcal{N}(x|\theta) = \sum_{k=1}^K w_k \mathcal{N}(x|\mu_k, \Sigma_k)$

, where $\mathcal{N}(\cdot|\mu_k, \Sigma_k)$ represents the D -variate Gaussian pdf with mean μ_k and covariance Σ_k . Given a data set $X = \{x_1, \dots, x_N\} \subset \mathbb{R}^D$, the K -Maximum-Likelihood-Estimation (K -MLE) problem is to find a mixture of K Gaussians θ such that the log-likelihood $\mathcal{L}(X, \theta) = \sum_{n=1}^N \log(\mathcal{N}(x_n|\theta))$ is maximized. For $K = 1$ this problem has a closed-form solution (cf. [4]). For $K > 1$ there exist iterative algorithms to approximate the K -MLE. These algorithms cannot create a solution, but rather maintain a solution and improve it in each round. Thus, they need to be fed with some initial model. The most well-known of these algorithms is the EM algorithm, on which we will focus on this paper. We expect that the results that we present in this paper also apply to adaptations like the SEM algorithm.

B. Related Work

The simplest and most widely used random initialization schemes are the so-called `rndEM` and `emEM` method (cf. [5]–[8]), which we will also refer to as “random initializations”. These methods generate several candidates by drawing means uniformly at random from the input set and then using some (data dependent) approximation of the covariances and weights. In case of `emEM`, the candidates are improved by few iterations of the EM algorithm. Finally, these methods choose the candidate with the largest likelihood. Other popular approaches are based on hierarchical agglomerative clustering (HAC). For instance, in [6]–[8] HAC (with Ward’s criterion, average linkage, and a model-based distance measure, respectively) is used to obtain the means of the initial model. Since HAC is generally very slow, it is usually executed on a random sample of the input. The size of this sample has to be chosen appropriately and obviously depends on the size of the smallest cluster. Except for [7], [6] and [8] report that this method is generally outperformed by random initializations. Another approach that makes use of HAC is the multistage approach proposed in [7]. It aims at finding the best local modes of the data set in a reduced m^* dimensional space and apply HAC only on these modes. However, the algorithm is rather time-consuming and the choice of m^* is crucial. In [6] a density based approach is proposed. It initializes the means by choosing points which have a higher concentration of neighbors and then estimates the covariances by making use

of a truncated normal distribution. In [9] a greedy algorithm is presented which does not need an initialization but instead constructs a whole sequence of mixture models with $k = 1, \dots, K$ components. Given a model θ_k with k components, the algorithm constructs various new candidates with $k + 1$ components and chooses the one with the highest likelihood. Each candidate is constructed by adding a new component to θ_k and improving it by executing the EM algorithm. Due to its run time, this algorithm is only feasible when the k -MLE problem has to be approximated for several values of k . There are further initialization methods which can be found, e.g., in [7], [10]–[12].

C. Our Contribution

Clearly, there is no way to determine “the” best initialization algorithm that outperforms all algorithms on all instances. The performance of an initialization will depend on the data and the allowed computational cost. Nonetheless, the initializations presented so far (except the simple random initializations) face mainly two problems: First, they are rather complex and time consuming. Second, the choice of hyperparameters is crucial for the outcome of the respective algorithm.

In this paper, we present simple and fast initialization methods inspired by methods for hard clustering problems. This closes the gap between the simple uniform methods and those methods that have been specifically designed for Gaussian mixture models. The initialization methods presented in this paper can be divided into two groups. First, we consider methods that directly apply known initialization methods for different K -clustering problems. Namely, we use the K -means++ algorithm [13], the Gonzalez algorithm, and a simple uniform initialization. There are already some practical applications of the K -means++ algorithm for the initialization of GMMs (e.g., in [14] these GMMs are used for speech recognition). Second, we discuss and define generalizations of these algorithms for Gaussian mixture models. Some initial research in this area has been done in [15], where a modification of the Gonzalez algorithm is considered. We extend this work by considering a further modification of the Gonzalez algorithm. Furthermore, we present a new algorithm based on the K -means++ algorithm. In our evaluation, we focus on the EM algorithm as the main learning algorithm and execute the initializations on a large number of classes of generated instances that share certain characteristics as well as several real world data sets. We compare the aforementioned algorithms with the algorithm presented in [15] and an algorithm based on HAC. Our results indicate that our adaption of the K -means++ algorithm and a method that makes use of the plain K -means++ algorithm perform best.

II. INITIALIZATION ALGORITHMS

A. Finding Initial Means

We consider the following three known initialization methods for K -clustering algorithms. The first method is the so-called K -means++ initialization [13], which is intended to be used for the K -means clustering algorithm. In each round,

$\text{Means2GMM}(X, \{\mu_1, \dots, \mu_k\})$	<i>Algorithm 1</i>
1: Derive partition $\{C_1, \dots, C_K\}$ of X by assigning each point $x_i \in X$ to its closest mean.	
2: Set $\mu_k = 1/ C_k \sum_{x \in C_k} x$.	
3: Set $\Sigma_k = 1/ C_k \sum_{x \in C_k} (x - \mu_k)(x - \mu_k)^T$.	
4: Set the weights to the fraction of points assigned to the respective cluster, i.e. $w_k = C_k / X $.	
5: If Σ_k is not positive definite, use a spherical covariance matrix, i.e., let $\Sigma_k = 1/D \cdot C_k \sum_{n=1}^N \ x_n - \mu_k\ ^2 \cdot I_D$, where I_D denotes the D -dimensional identity matrix.	
6: If Σ_k is still not positive definite, let $\Sigma_k = I_D$.	

Fig. 1. Construction of a Gaussian mixture model from K means.

K -means++ samples a new mean from the given data set with probability proportional to its K -means cost (with respect to the means chosen so far). In expectation, the costs are in $\mathcal{O}(\log(K) \cdot \text{opt})$, where opt denotes the optimal K -means costs. This initialization is particularly interesting since the K -means algorithm can be seen as a special case of the EM (cf. [4, p.443]). The second method is the Gonzalez algorithm [16], which is a 2-approximation algorithm for the discrete radius K -clustering problem. Similar to K -means++, it determines the means iteratively. As the next mean it always chooses the data point with the largest euclidean distance from the already chosen means, i.e., with the largest K -means cost. Hence, K -means++ can be seen as a probabilistic variant of the Gonzalez algorithm. The third initialization method is to sample K means uniformly at random from the given data set (cf. `rndEM` in Sec. I-B).

B. Directly Using Initial Means

The following three methods directly use the aforementioned initializations. Gonzalez executes the Gonzalez algorithm, `Kmeans++` executes the K -means++ algorithm [13], and `Uniform` draws K means uniformly at random from the given data set. Then, they all modify and extend the resulting set of K means to a complete Gaussian mixture model by a method that we refer to as `Means2GMM` (cf. Fig. 1). Given K means, `Means2GMM` reduces the MLE problem to K independent 1-MLE problems. That is, it partitions the data according to the K -means clustering induced by the means and then solves the 1-MLE problems with respect to each of the clusters (cf. [4]). However, it might happen that a cluster is too small. In such cases, `Means2GMM` determines an appropriate approximation (i.e., a spherical covariance matrix).

C. Transferring the Ideas to Gaussian Mixture Models

Implementing the ideas behind the Gonzalez and K -means++ algorithm for Gaussian mixture models raises two problems. First, we have to choose an appropriate density function according to which a point from the data set will be chosen (either by drawing according to the density or by determining the point with highest density). Second, we also have to determine how the current Gaussian mixture model is updated.

Two natural candidates for the density function are the inverse pdf of the Gaussian mixture and the negative log-likelihood. Unfortunately, the former is unsuited due to the exponential behavior of Gaussians, while the latter may take negative values. Thus, we use an approximation of the negative log-likelihood. To define this, consider the minimum negative log-likelihood of a point x taken over the individual components

$$\min_k \left\{ -\log \left((2\pi)^{D/2} |\Sigma_k|^{1/2} \right) + \frac{1}{2} (x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k) \right\}. \quad (1)$$

Note that the logarithm of the normalization term of the Gaussian density may take negative values, while the Mahalanobis density $(x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k)$ is always non-negative. Assuming that the logarithm of the normalization term is positive, it would practically function as a shift towards a uniform distribution. The effect of this shift is that the probability for choosing outliers is reduced. Given the Gaussian mixture model $\theta = \{(w_k, \mu_k, \Sigma_k)\}_{k=1, \dots, K}$, we denote the minimum Mahalanobis density of point $x \in \mathbb{R}^D$ by

$$m_1(x|\theta) = \min_{k=1, \dots, K} (x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k).$$

To keep the effect incurred by the (unfeasible) normalization term, we replace it by a constant. That is, for some fixed $S \subseteq X$, we define for all $x \in S$ and $\alpha \in [0, 1]$

$$m_{\alpha, S}(x|\theta) = \alpha \frac{m_1(x|\theta)}{\sum_{y \in S} m_1(y|\theta)} + (1 - \alpha) \frac{1}{|S|}.$$

This density corresponds to an experiment where x is drawn uniformly at random from $S \subseteq X$ with probability $1 - \alpha$. Our adaptations draw a points according to density or choose the point with maximum density, respectively. Note that $\arg \max_{x \in S} m_1(x|\theta) = \arg \max_{x \in S} m_{\alpha, S}(x|\theta)$. Furthermore, points with maximum density are more likely to be outliers. To avoid the problem of only choosing such outliers, our adaption of Gonzalez first chooses a uniform subsample S of the input data set X and then only samples points from S . Given a sample point $x \in X$, we have to determine how the complete Gaussian mixture model is updated¹. To this end, our adaptations of the K -means++ and the Gonzalez algorithm simply apply Means2GMM to the set of points including the means of the current model θ and the newly drawn point $x \in X$.

Our adaptations, Adaptive and GonzalezForGMM, are shown in Fig. 2 and Fig. 3, respectively. In our experiments, we compare them with another adaption of the Gonzalez algorithm presented in [15], which we refer to as KwedlosGonzalez (cf. Fig. 4 and Fig. 5). In contrast to our adaption, it chooses the covariances randomly (in relation to the covariance of the overall data set X) and draws the weights uniformly at random.

¹ Our first attempts consisted in just adding a new component to the old mixture model, for instance, by choosing the covariances as in [17] (i.e., a spherical covariance depending on the minimum distances between means). However, this approach did not work out at all.

Adaptive($X \subset \mathbb{R}^D, K \in \mathbb{N}, \alpha \in [0, 1]$)

- 1: compute the optimal 1-solution θ_1
- 2: **for** For $i = 2, \dots, K$ **do**
- 3: Draw p from X with probability $m_{\alpha, S}(x|\theta_{k-1})$
- 4: Obtain θ_k by calling Means2GMM with all means from θ_{k-1} and the point p as input.
- 5: **end for**

Fig. 2. Our adaption of the K -means++ algorithm.

GonzalezForGMM($X \subset \mathbb{R}^D, K \in \mathbb{N}, s \in (0, 1]$)

- 1: Let S be a uniform sample of X of size $s \cdot |X|$.
- 2: Compute the optimal 1-MLE θ_1 .
- 3: **for** For $i = 2, \dots, K$ **do**
- 4: Choose $p = \arg \min_{x \in S} m_1(x|\theta_{k-1})$.
- 5: Obtain θ_k by calling Means2GMM with all means from θ_{k-1} and the point p as input.
- 6: **end for**

Fig. 3. Our adaption of the Gonzalez algorithm.

KwedlosGonzalez($X \subset \mathbb{R}^D, K \in \mathbb{N}, s \in (0, 1]$)

- 1: Let S be a uniform sample of X of size $s \cdot |X|$.
- 2: Let $\theta_1 = (\mu_1, \Sigma_1)$, where μ_1 is chosen uniformly at random from S and $\Sigma_1 = \text{RandomCovar}()$.
- 3: **for** For $i = 2, \dots, K$ **do**
- 4: Choose $\mu_k = \arg \min_{x \in S} m_1(x|\theta_{k-1})$.
- 5: Let $\Sigma_k = \text{RandomCovar}(X)$.
- 6: **end for**
- 7: Draw $w_k, k = 1, \dots, K$ uniformly at random from $[0, 1]$ and normalize them by $\sum_{k=1}^K w_k$.

Fig. 4. Adaption of the Gonzalez algorithm according to [15].

RandCovar($X \subset \mathbb{R}^D$)

- 1: Let $\Sigma_X = 1/|X| \cdot \sum_{x \in X} (x - \mu_X)(x - \mu_X)^T$,
- 2: where $\mu_X = 1/|X| \cdot \sum_{x \in X} x$.
- 3: Choose eigenvalues $\lambda_1, \dots, \lambda_D \in \mathbb{R}$ such that
- 4: $\sum_{d=1}^D \lambda_k = \text{trace}(\Sigma_X)/(10 \cdot D \cdot K)$ and
- 5: $\max_{d=1, \dots, D} (\lambda_d) / \min_{d=1, \dots, D} (\lambda_d) \leq 10$.
- 6: Generate a random orthonormal matrix $Q \in \mathbb{R}^{D \times D}$.
- 7: Return $Q^T \text{diag}(\lambda_1, \dots, \lambda_D) Q$.

Fig. 5. Generation of a random covariance matrix according to [15].

III. EXPERIMENTS

Besides the presented algorithms, we evaluated a method using HAC with average linkage cost. That is, a method that executes HAC on a uniform sample of size $s \cdot |X|$ of the input set X and then used Means2GMM. For this approach and for the algorithms presented in Sec. II-C, we only tested a few, reasonable values for hyperparameters, i.e., $s = 0.1$ and $\alpha \in \{1, 0.5\}$.

To get comparable results, all methods were implemented in C++ (available at [18]). Since all presented methods are randomized, we executed each initialization with 30 different

seeds² to obtain reasonable results. As a stopping criterion for the EM algorithm we simply used a fixed number of rounds. Usually, a stopping criterion based on the relative change of the likelihood is used. However, this requires choosing a reasonable threshold, which is crucial not only for the resulting likelihood but also for the run time. After some initial experimental evaluations, we chose a run time of 50 rounds.

A. Data

1) *Real World Data Sets*: We use four publicly available data sets. The first data set is provided by the *ELKI* project [19] and based on the *Amsterdam Library of Object Images (ALOI)* [20], which consists of 110 250 pictures of 1 000 small objects (taken under various conditions). We use a 27-dimensional feature vector set that is based on color histograms in HSV color space (cf. [21]). We created the second data set, denoted as *Cities* data set, from the data provided by the GeoNames geographical database (<http://www.geonames.org/>), which contains several features of 135 082 cities around the world with a population of at least 1000. Our data set is a projection of the geographic coordinates of the cities onto a plane. The third one is the *Forest Covertype* data set which contains 581 012 data points and is available from the *UCI Machine Learning Library* [22]. To get reasonable data, we ignored the class labels and 44 binary attributes and only used the remaining 10 coordinates. The fourth data set is the *Spambase* data set which is also provided by the *UCI Machine Learning Library* [22]. It contains 57 features of 4601 e-mails and a binary classification whether the e-mail was considered spam or not. Again, we ignored the latter.

2) *Generated Data Sets*: We created different test sets which contain data sets with certain characteristics. That is, for each test set we created Gaussian mixture models that share certain predefined properties. Then, to obtain a data set, we drew data points according to a mixture and added a certain amount of noise points.

Our generation of Gaussian mixture models focused on the following four properties. First, the components of a Gaussian mixture model can either be spherical or elliptical. We describe the eccentricity of covariance matrix Σ_k by $e_k = \frac{\max_d \lambda_{kd}}{\min_d \lambda_{kd}}$, where λ_{kd}^2 denotes the d -th eigenvalue of Σ_k . Second, components can have different sizes, in terms of the smallest eigenvalue of the corresponding covariance matrices. Third, the Gaussian components can overlap more or less. Following [17], we define the separation c_θ of a Gaussian mixture model θ by

$$c_\theta = \min_{c \in \mathbb{R}; l, k=1, \dots, K} \frac{\|\mu_l - \mu_k\|}{\sqrt{\max\{\text{trace}(\Sigma_l), \text{trace}(\Sigma_k)\}}}.$$

² Furthermore, since our EM implementation includes an randomized error handling, we executed the EM with 3 different seeds. The EM encounters a problem when the expected number of points generated by a component is too small or the computed covariances are not positive semi-definite. In the first case, we sample a new mean and covariance. In the latter case, we first try to mix the underdetermined covariance with the previous covariance, then simply keep the old covariance if necessary.

TABLE I
OVERVIEW OF TEST SETS

notation	separation	weights	size	eccentricity e_k
<i>Spherical</i>	0.5, 1, 2	uniform	const.	1
	0.5, 1, 2	different	const.	1
	0.5, 1, 2	uniform	different	1
	0.5, 1, 2	different	different	1
<i>Elliptical</i>	0.5, 1, 2	uniform	const.	2, 5, 10
<i>Elliptical-Difficult</i>	0.5, 1, 2	uniform	different	5
	0.5, 1, 2	different	const.	5
	0.5, 1, 2	uniform	different	[1, 10]

Finally, when drawing points according to a mixture, its weights determine the expected number of points are drawn according to a component. To control these properties, we generated random Gaussian mixture models as follows. Initially, the means are drawn uniformly at random from a cube with a fixed side length. The values of the weights are determined deterministically by normalizing the K values in $\{2^{c_w \cdot i}\}_{i=1, \dots, K}$ by their sum. By choosing c_w appropriately, we obtain a certain sequence of weights. To obtain a random covariance matrix whose eigenvalues satisfy predetermined properties, we first determine eigenvalues $\lambda_{k1}^2, \dots, \lambda_{kD}^2$ that match our restrictions. Then, we determine a random orthonormal matrix $Q \in \mathbb{R}^{D \times D}$ (by QR -decomposition of a matrix with independently and uniformly drawn values) and set $Q^T \text{diag}(\lambda_{k1}^2, \dots, \lambda_{kD}^2) Q$ to be the k -th covariance matrix. To ensure different sized components, we initialize the minimum eigenvalues of each component appropriately. To control the eccentricity, we set the maximum eigenvalues according to the minimum eigenvalues and the chosen eccentricity and fill the remaining eigenvalues uniformly at random. Finally, the means are scaled as to fit the predefined separation³.

Table I gives an overview of the test sets that we considered. Each combination of values in a row describes a single test set, i.e., there are 12 *Spherical*, 9 *Elliptical*, and 9 *Elliptical-Difficult* test sets. For each test set, we generated 30 data sets by choosing 30 Gaussian mixture models with $K = 10$ components randomly (as described before) and drawing 10 000 points according to each model. To obtain noisy data sets, we only drew 9 000 points according to the mixture model and added 1 000 uniform noise points by constructing a bounding box of all drawn points, elongating its side lengths by a factor 1.2 and then drawing points uniformly at random from the resized box. Our implementation available at [18] offers the opportunity to generate the data.

B. Results

In our evaluation we focus on the quality of the initial and the final solution. The complete results can also be found at [18].

³ Thus, when keeping the remaining parameters fixed, we obtain basically (except the scaled means) the same mixture models and can evaluate the impact of different degrees of separation.

1) *Generated Data Sets*: Remember, we have generated a huge amount of data. For every data set, we calculate the average value and the variance of the negative log-likelihood of the solutions obtained by an initialization method. In order to summarize this information with respect to a fixed test set, we proceed in the following two steps⁴: First, we create rankings of the initialization methods. That is, for each data set we obtain four different rankings based on initial and final solution and measured by average and variance of the negative log-likelihood. Second, we count the respective times a certain rank has been achieved. Since we can not depict all the $(12 + 9 + 9) \cdot 2 \cdot 8 = 480$ summarized rankings in this paper, we made them available at [18] along with some further summaries. To give an overall impression of the results, we summed up the rankings obtained for data sets contained in the *Spherical*, *Elliptical*, and *Elliptical-Difficult* test sets, respectively. Furthermore, we summed up the rankings obtained for data sets with separation 0.5, 1, and 2, respectively. The results are depicted in Tab. II, III, IV, and V.

From our experiments, the following comments can be made: We constructed spherical as well as (difficult) elliptical data sets. Some authors pointed out that methods based on *K*-means may encounter problems if the given clusters are skewed (e.g. [6]). However, this is not confirmed by our experiments since the overall impression is the same: The methods *Kmeans++* and *Adaptive* perform best. Note that we also experimented with different choices of the parameter α for *Adaptive* and found that it is rather insensitive, given $\alpha \in [0.5, 0.9]$. Moreover, we investigated the impact of noise. As to be expected, given noisy data, the different versions of the Gonzalez algorithm lag far behind the other methods. This is most likely due to the fact that outliers are chosen as means. Furthermore, we tested different degrees of separation. For separation $c_\theta \geq 1$ the methods *Kmeans++* and *Adaptive* clearly outperform the other methods. Even for $c_\theta = 0.5$, where the situation is not quite as clear, these methods perform well. Generally, the method that uses HAC and Kwedlo's Gonzalez version, which uses basically random covariances, are outperformed by the other methods. In addition, from the rankings according to the variances we see that their results also vary the most. Last but not least, notice that an initial solution with high likelihood does not necessarily result in a final solution with high likelihood (compared to other initialization techniques).

2) *Real World Data Sets*: The results for the real world data sets are depicted in Fig. 6. They confirm our impression regarding the adaptive initialization methods (i.e., *Kmeans++* and *Adaptive*). Regarding the *Aloi*, *Cities*, and *Spambase*, these methods are among the top performers. A slight exception is their performance with respect to the *Covertypes* data set, where Gonzalez performs surprisingly well.

⁴ Note that simply averaging over the evaluations of different data sets is not meaningful since their (optimal) likelihoods may deviate significantly.

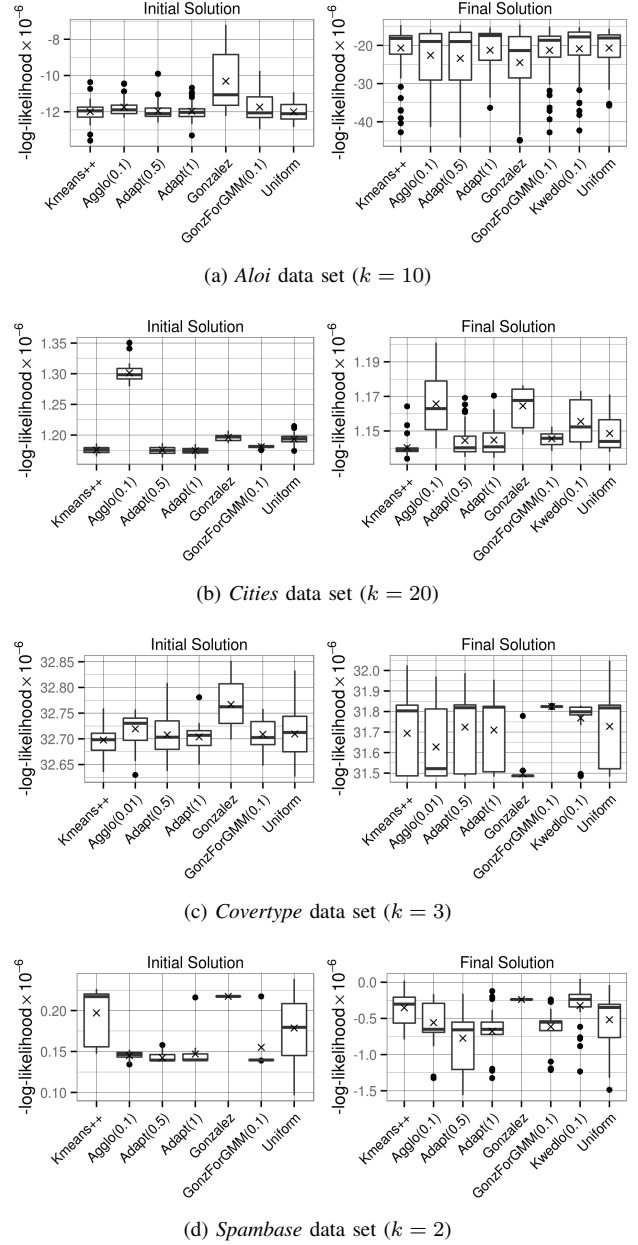


Fig. 6. Results for the real world data sets depicted as boxplots. The bottom and top of the box are the first and third quartiles. The band inside the box is the median, while the cross indicates the mean. The ends of the whiskers represent the lowest/highest datum still within 1.5 interquartile range (IQR) of the lower/upper quartile. We omit Kwedlo's Gonzalez in the "initial" plots due to its high costs.

C. Conclusion

If you need a fast and simple initialization method that does not need any training of hyperparameters then we suggest to use one of the following. Given a data set that presumably contains noise, the *Adaptive* initialization method with $\alpha = 0.5$ is a good choice. If there is no noise, then simply use the plain *K*-means++ initialization followed by *Means2GMM*.

TABLE II
SUMMARIZED RANKINGS WRT. THE NEGATIVE -LOG-LIKELIHOOD FOR GENERATED DATA SETS – WITHOUT NOISE.

Summing over all 360 rankings obtained from data sets contained in the 12 <i>Spherical</i> test sets wrt. the average negative log-likelihood.																	
initial solution								final solution									
	#1	#2	#3	#4	#5	#6	#7	#8		#1	#2	#3	#4	#5	#6	#7	#8
Uniform			1	13	158	188			Uniform	4	8	10	34	37	68	124	75
Kmeans++	70	86	62	128	14				Kmeans++	84	36	149	41	25	18	7	
Adaptive(1)	57	115	98	63	27				Adaptive(1)	24	34	51	76	86	60	24	5
Adaptive(0.5)	50	85	124	67	29	5			Adaptive(0.5)	29	40	27	78	84	64	33	5
Agglomerative(0.1)					1	29	330		Agglomerative(0.1)	12	8	12	40	45	53	74	116
Gonzalez	78	37	13	38	69	107	18		Gonzalez	117	63	55	21	28	16	24	36
Gonz.ForGMM(0.1)	105	37	62	51	62	31	12		Gonz.ForGMM(0.1)	26	25	30	45	32	54	43	105
KwedlosGonz.(0.1)								360	KwedlosGonz.(0.1)	64	146	26	25	23	27	31	18
Summing over all 270 rankings obtained from data sets contained in the 9 <i>Elliptical</i> test sets wrt. the average negative log-likelihood.																	
initial solution								final solution									
	#1	#2	#3	#4	#5	#6	#7	#8		#1	#2	#3	#4	#5	#6	#7	#8
Uniform					155	115			Uniform	2	2	10	16	29	69	124	18
Kmeans++	68	52	65	82	3				Kmeans++	39	32	143	20	18	13	5	
Adaptive(1)	82	75	73	40					Adaptive(1)	20	30	19	72	73	41	12	3
Adaptive(0.5)	68	97	68	35	2				Adaptive(0.5)	22	19	20	69	87	31	20	2
Agglomerative(0.1)						10	260		Agglomerative(0.1)			4	32	15	45	57	117
Gonzalez	31	38	4	33	59	105			Gonzalez	108	59	31	18	15	10	18	11
Gonz.ForGMM(0.1)	21	8	60	80	51	40	10		Gonz.ForGMM(0.1)	7	10	11	23	25	43	32	119
KwedlosGonz.(0.1)								270	KwedlosGonz.(0.1)	72	118	32	20	8	18	2	
Summing over all 270 rankings obtained from data sets contained in the 9 <i>Elliptical-Difficult</i> test sets wrt. the average negative log-likelihood.																	
initial solution								final solution									
	#1	#2	#3	#4	#5	#6	#7	#8		#1	#2	#3	#4	#5	#6	#7	#8
Uniform				20	154	96			Uniform	5	6	22	31	25	58	76	47
Kmeans++	115	30	63	61	1				Kmeans++	102	39	74	39	13	2	1	
Adaptive(1)	47	104	81	28	10				Adaptive(1)	15	32	25	62	63	47	22	4
Adaptive(0.5)	48	71	85	52	14				Adaptive(0.5)	19	26	30	40	66	53	27	9
Agglomerative(0.1)						24	246		Agglomerative(0.1)	2	10	8	33	54	34	64	65
Gonzalez	16	47	9	37	49	108	4		Gonzalez	71	41	43	15	12	15	25	48
Gonz.ForGMM(0.1)	44	18	32	72	42	42	20		Gonz.ForGMM(0.1)	12	9	30	33	25	31	34	96
KwedlosGonz.(0.1)								270	KwedlosGonz.(0.1)	44	107	38	17	12	30	21	1

TABLE III
SUMMARIZED RANKINGS WRT. THE NEGATIVE -LOG-LIKELIHOOD FOR GENERATED DATA SETS – WITH UNIFORM NOISE.

Summing over all 360 rankings obtained from data sets from the noisy <i>Spherical</i> test sets wrt. the average negative log-likelihood																	
initial solution									final solution								
	#1	#2	#3	#4	#5	#6	#7	#8		#1	#2	#3	#4	#5	#6	#7	#8
Uniform				7	293	60			Uniform	30	28	55	125	79	27	13	3
Kmeans++	5	14	97	240	4				Kmeans++	122	78	67	43	48	2		
Adaptive(1)	105	199	33	22	1				Adaptive(1)	67	96	124	66	7			
Adaptive(0.5)	55	85	159	26	35				Adaptive(0.5)	93	134	81	42	8	2		
Agglomerative(0.1)						86	274		Agglomerative(0.1)	3	11	17	46	118	89	23	53
Gonzalez	14	22	1	1	23	213	86		Gonzalez		4	5	3	17	32	69	230
Gonz.ForGMM(0.1)	181	40	70	64	4	1			Gonz.ForGMM(0.1)	45	8	11	29	52	150	41	24
KwedlosGonz.(0.1)								360	KwedlosGonz.(0.1)		1		6	31	58	214	50
Summing over all 270 rankings obtained from data sets from the noisy <i>Elliptical</i> test sets wrt. the average negative log-likelihood																	
initial solution									final solution								
	#1	#2	#3	#4	#5	#6	#7	#8		#1	#2	#3	#4	#5	#6	#7	#8
Uniform					214	56			Uniform	21	60	70	96	23			
Kmeans++	4	20	71	171	4				Kmeans++	74	39	38	40	79			
Adaptive(1)	104	142	16	8					Adaptive(1)	37	56	101	74	2			
Adaptive(0.5)	15	60	139	32	24				Adaptive(0.5)	116	102	38	14				
Agglomerative(0.1)						39	231		Agglomerative(0.1)	21	12	21	41	117	43	7	8
Gonzalez	3	13	8	4	28	175	39		Gonzalez					2	25	41	202
Gonz.ForGMM(0.1)	144	35	36	55					Gonz.ForGMM(0.1)	1	1	2	4	31	133	59	39
KwedlosGonz.(0.1)								270	KwedlosGonz.(0.1)				1	16	69	163	21
Summing over all 270 rankings obtained from data sets from the noisy <i>Elliptical-Difficult</i> test sets wrt. the average negative log-likelihood																	
initial solution									final solution								
	#1	#2	#3	#4	#5	#6	#7	#8		#1	#2	#3	#4	#5	#6	#7	#8
Uniform				6	235	28	1		Uniform	41	58	52	83	21	9	4	2
Kmeans++	5	9	95	158	3				Kmeans++	86	37	35	37	73	1	1	
Adaptive(1)	126	130	14						Adaptive(1)	26	55	90	88	11			
Adaptive(0.5)	39	100	114	15	2				Adaptive(0.5)	97	95	61	14	3			
Agglomerative(0.1)					1	92	177		Agglomerative(0.1)	19	22	27	32	102	27	17	24
Gonzalez		2	1	2	24	149	92		Gonzalez	1		1	2	15	16	40	195
Gonz.ForGMM(0.1)	100	29	46	89	5	1			Gonz.ForGMM(0.1)		1	4	7	25	133	67	33
KwedlosGonz.(0.1)								270	KwedlosGonz.(0.1)		2		7	20	84	141	16

TABLE IV
SUMMARIZED RANKINGS WRT. THE NEGATIVE -LOG-LIKELIHOOD FOR GENERATED DATA SETS – WITHOUT UNIFORM NOISE.

Summing over all rankings obtained from data sets with separation 0.5 wrt. the average negative log-likelihood																	
initial solution								final solution									
	#1	#2	#3	#4	#5	#6	#7	#8		#1	#2	#3	#4	#5	#6	#7	#8
Uniform			1	20	214	65			Uniform	10	11	29	40	31	49	90	40
Kmeans++	20	13	90	163	14				Kmeans++	60	30	74	60	40	25	11	
Gonzalez	6	8		11	51	204	20		Gonzalez	50	42	23	25	30	25	38	67
Adaptive(1)	95	141	50	9	5				Adaptive(1)	43	63	51	56	51	25	10	1
Adaptive(0.5)	82	111	91	9	2	5			Adaptive(0.5)	51	53	46	42	48	41	12	7
Agglomerative(0.1)					1	20	279		Agglomerative(0.1)	12	8	9	20	34	23	57	137
Gonz.ForGMM(0.1)	97	27	68	88	13	6	1		Gonz.ForGMM(0.1)	33	27	37	25	40	62	43	33
KwedlosGonz.(0.1)								300	KwedlosGonz.(0.1)	41	66	31	32	26	50	39	15
Summing over all rankings obtained from data sets with separation 1 wrt. the average negative log-likelihood																	
initial solution								final solution									
	#1	#2	#3	#4	#5	#6	#7	#8		#1	#2	#3	#4	#5	#6	#7	#8
Uniform				10	132	158			Uniform	1	4	10	29	32	62	109	53
Kmeans++	78	45	79	97	1				Kmeans++	77	43	121	34	15	8	2	
Gonzalez	20	27	10	38	97	106	2		Gonzalez	112	61	38	17	13	15	21	23
Adaptive(1)	75	110	75	31	9				Adaptive(1)	15	27	34	56	72	66	23	7
Adaptive(0.5)	71	99	83	40	7				Adaptive(0.5)	19	26	23	73	84	39	32	4
Agglomerative(0.1)						4	296		Agglomerative(0.1)	1	6	11	31	45	42	55	109
Gonz.ForGMM(0.1)	56	19	53	84	54	32	2		Gonz.ForGMM(0.1)	7	13	26	35	26	48	44	101
KwedlosGonz.(0.1)								300	KwedlosGonz.(0.1)	68	120	37	25	13	20	14	3
Summing over all rankings obtained from data sets with separation 2 wrt. the average negative log-likelihood																	
initial solution								final solution									
	#1	#2	#3	#4	#5	#6	#7	#8		#1	#2	#3	#4	#5	#6	#7	#8
Uniform				3	121	176			Uniform		1	3	12	28	84	125	47
Kmeans++	155	110	21	11	3				Kmeans++	88	34	171	6	1			
Gonzalez	99	87	16	59	29	10			Gonzalez	134	60	68	12	12	1	8	5
Adaptive(1)	16	43	127	91	23				Adaptive(1)	1	6	10	98	99	57	25	4
Adaptive(0.5)	13	43	103	105	36				Adaptive(0.5)		6	8	72	105	68	36	5
Agglomerative(0.1)						39	261		Agglomerative(0.1)	1	4	4	54	35	67	83	52
Gonz.ForGMM(0.1)	17	17	33	31	88	75	39		Gonz.ForGMM(0.1)	5	4	8	41	16	18	22	186
KwedlosGonz.(0.1)								300	KwedlosGonz.(0.1)	71	185	28	5	4	5	1	1

TABLE V
SUMMARIZED RANKINGS WRT. THE NEGATIVE -LOG-LIKELIHOOD FOR GENERATED DATA SETS – WITH UNIFORM NOISE.

Summing over all rankings obtained from data sets with separation 0.5 wrt. the average negative log-likelihood																	
initial solution									final solution								
	#1	#2	#3	#4	#5	#6	#7	#8		#1	#2	#3	#4	#5	#6	#7	#8
Uniform				3	189	108			Uniform	60	66	49	62	49	10	4	
Kmeans++		2	68	219	11				Kmeans++	18	45	49	46	140	1	1	
Gonzalez	17	37	9	6	39	146	46		Gonzalez		1	4		6	5	33	251
Adaptive(1)	47	118	23	21	1				Adaptive(1)	24	44	75	62	5			
Adaptive(0.5)	14	44	153	30	59				Adaptive(0.5)	106	91	58	36	7	2		
Agglomerative(0.1)						46	254		Agglomerative(0.1)	33	31	40	45	57	76	5	13
Gonz.ForGMM(0.1)	215	35	35	14	1				Gonz.ForGMM(0.1)	39	5	3	14	29	160	44	6
KwedlosGonz.(0.1)								300	KwedlosGonz.(0.1)		1		4	6	46	213	30
Summing over all rankings obtained from data sets with separation 1 wrt. the average negative log-likelihood																	
initial solution									final solution								
	#1	#2	#3	#4	#5	#6	#7	#8		#1	#2	#3	#4	#5	#6	#7	#8
Uniform				3	280	17			Uniform	23	55	64	100	40	10	5	3
Kmeans++		6	70	224					Kmeans++	75	56	55	56	56	2		
Gonzalez			1	1	15	194	89		Gonzalez	1	1	2	1	7	22	54	212
Adaptive(1)	77	119	14						Adaptive(1)	45	51	62	49	3			
Adaptive(0.5)	36	79	164	19	2				Adaptive(0.5)	130	95	57	14	4			
Agglomerative(0.1)						89	211		Agglomerative(0.1)	7	13	20	47	136	43	14	20
Gonz.ForGMM(0.1)	156	40	48	53	3				Gonz.ForGMM(0.1)	6	4	6	9	33	155	50	37
KwedlosGonz.(0.1)								300	KwedlosGonz.(0.1)		1		5	21	68	177	28
Summing over all rankings obtained from data sets with separation 2 wrt. the average negative log-likelihood																	
initial solution									final solution								
	#1	#2	#3	#4	#5	#6	#7	#8		#1	#2	#3	#4	#5	#6	#7	#8
Uniform				7	273	19	1		Uniform	9	25	64	142	34	16	8	2
Kmeans++	14	35	125	126					Kmeans++	189	53	36	18	4			
Gonzalez					21	197	82		Gonzalez		2		4	21	46	63	164
Adaptive(1)	107	92	10	1					Adaptive(1)	24	56	77	43	10			
Adaptive(0.5)	59	122	95	24					Adaptive(0.5)	70	145	65	20				
Agglomerative(0.1)					1	82	217		Agglomerative(0.1)	3	1	5	27	144	40	28	52
Gonz.ForGMM(0.1)	54	29	69	141	5	2			Gonz.ForGMM(0.1)	1	1	8	17	46	101	73	53
KwedlosGonz.(0.1)								300	KwedlosGonz.(0.1)		1		5	40	97	128	29

REFERENCES

- [1] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society, Series B: Statistical Methodology*, vol. 39, no. 1, pp. 1–38, 1977.
- [2] C. Wu, "On the convergence properties of the EM algorithm," *The Annals of Statistics*, vol. 11, no. 1, pp. 95–103, 1983.
- [3] G. J. McLachlan and T. Krishnan, *The EM Algorithm and Extensions (Wiley Series in Probability and Statistics)*, 2nd ed. Wiley-Interscience, Mar. 2008.
- [4] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.
- [5] C. Biernacki, G. Celeux, and G. Govaert, "Choosing starting values for the EM algorithm for getting the highest likelihood in multivariate Gaussian mixture models," *Comput. Stat. Data Anal.*, vol. 41, no. 3-4, pp. 561–575, Jan. 2003.
- [6] V. Melnykov and I. Melnykov, "Initializing the EM algorithm in Gaussian mixture models with an unknown number of components," *Computational Statistics & Data Analysis*, Nov. 2011.
- [7] R. Maitra, "Initializing Partition-Optimization Algorithms," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 6, no. 1, pp. 144–157, 2009.
- [8] M. Meilă and D. Heckerman, "An Experimental Comparison of Several Clustering and Initialization Methods," in *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann, Inc., San Francisco, CA, 1998, pp. 386–395.
- [9] J. J. Verbeek, N. Vlassis, and B. Kröse, "Efficient greedy learning of Gaussian mixture models," *Neural computation*, vol. 15, no. 2, pp. 469–485, 2003.
- [10] B. Thiesson, *Accelerated quantification of Bayesian networks with incomplete data*. University of Aalborg, Institute for Electronic Systems, Department of Mathematics and Computer Science, 1995.
- [11] U. M. Fayyad, C. Reina, and P. S. Bradley, "Initialization of Iterative Refinement Clustering Algorithms," in *KDD*, 1998, pp. 194–198.
- [12] C. Biernacki, "Initializing EM using the properties of its trajectories in Gaussian mixtures," *Statistics and Computing*, vol. 14, no. 3, pp. 267–279, 2004.
- [13] D. Arthur and S. Vassilvitskii, "k-means++: the advantages of careful seeding," in *SODA*, N. Bansal, K. Pruhs, and C. Stein, Eds. SIAM, 2007, pp. 1027–1035.
- [14] A. Krueger, V. Leutnant, R. Haeb-Umbach, M. Ackermann, and J. Bloemer, "On the initialization of dynamic models for speech features," *ITG-Fachbericht-Sprachkommunikation 2010*, 2010.
- [15] W. Kwedlo, "A New Method for Random Initialization of the EM Algorithm for Multivariate Gaussian Mixture Learning," in *Proceedings of the 8th International Conference on Computer Recognition Systems CORES 2013*. Springer International Publishing, 2013, pp. 81–90.
- [16] T. F. Gonzalez, "Clustering to minimize the maximum intercluster distance," *Theoretical Computer Science*, vol. 38, pp. 293–306, 1985.
- [17] S. Dasgupta, "Learning Mixtures of Gaussians," in *FOCS*. IEEE Computer Society, 1999, pp. 634–644.
- [18] "Supplemental material," <https://github.com/ka-bu/simple-gmm-initializations/>.
- [19] E. Achtert, S. Goldhofer, H.-P. Kriegel, E. Schubert, and A. Zimek, "Evaluation of Clusterings – Metrics and Visual Support," *Data Engineering, International Conference on*, vol. 0, pp. 1285–1288, 2012. [Online]. Available: <http://elki.dbs.ifi.lmu.de/wiki/DataSets/MultiView>
- [20] Geusebroek, J. M. and Burghouts, G. J. and Smeulders, A. W. M., "The Amsterdam Library of Object Images," *International Journal of Computer Vision*, vol. 6, no. 1, pp. 103–112, 2005.
- [21] H.-P. Kriegel, E. Schubert, and A. Zimek, "Evaluation of Multiple Clustering Solutions," in *Proc. ECML PKDD Workshop MultiClust*, 2011.
- [22] D. N. A. Asuncion, "UCI machine learning repository," 2007. [Online]. Available: <http://www.ics.uci.edu/~mllearn/MLRepository.html>