Adaptive Seeding for Gaussian Mixture Models

Johannes Blömer and Kathrin Bujna[(✉)]

Paderborn University, 33098 Paderborn, Germany
{bloemer,kathrin.bujna}@uni-paderborn.de

Abstract. We present new initialization methods for the expectation-maximization algorithm for multivariate Gaussian mixture models. Our methods are adaptions of the well-known K-means++ initialization and the Gonzalez algorithm. Thereby we aim to close the gap between simple random, e.g. uniform, and complex methods, that crucially depend on the right choice of hyperparameters. Our extensive experiments indicate the usefulness of our methods compared to common techniques and methods, which e.g. apply the original K-means++ and Gonzalez directly, with respect to artificial as well as real-world data sets.

1 Introduction

Gaussian mixture modelling is an important task, e.g., in the field of cluster analysis. A common approach is the method of maximum likelihood for which the Expectation-Maximization (EM) algorithm [13] can be applied. The EM algorithm iteratively tries to improve a given initial mixture model and converges to a stationary point of the likelihood function. Unfortunately, the likelihood function is generally non-convex, possessing many stationary points [23]. The initial model determines to which of these points the EM algorithm converges [5].

1.1 Maximum Likelihood Estimation for Gaussian Mixtures

A Gaussian mixture model (K-GMM) over $\mathbb{R}^D$ can be described by a parameter $\theta = \{(w_k, \mu_k, \Sigma_k)\}_{k=1,\ldots,K}$, 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 matrix of the k-th mixture component. Its probability density function is given by $\mathcal{N}(x|\theta) = \sum_{k=1}^{K} w_k \mathcal{N}(x|\mu_k, \Sigma_k)$, where we denote the D-variate Gaussian distribution by $\mathcal{N}(\cdot|\mu, \Sigma)$. Given a data set $X \subset \mathbb{R}^D$, the Maximum Likelihood Estimation (MLE) problem is to find a K-GMM θ that maximizes the likelihood $\mathcal{L}(\theta|X) = \prod_{x \in X} \mathcal{N}(x|\theta)$. For $K = 1$, there is a closed-form solution [8]. For $K > 1$, the EM algorithm, whose outcome heavily depends on the initial model, can be applied.

1.2 Related Work

A common way to initialize the EM algorithm is to first draw means uniformly at random from the input set and then to approximate covariances and weights

[7, 21, 24, 25]. To compensate for the random choice of initial means, several candidate solutions are created and the one with the largest likelihood is chosen. Often, few steps of the EM, Classification EM, or Stochastic EM algorithm are applied to the candidates. Similarly, the K-means algorithm may be used [8, p. 427]. Due to the random choice of the initial means, all these methods are better suited for spherical and well-separated clusters. Furthermore, testing several candidates is computationally expensive.

Other popular initializations are based on hierarchical agglomerative clustering (HAC). For instance, in [21, 24, 25], HAC (with different distance measures) is used to obtain mean vectors. Since HAC is generally very slow, it is usually only executed on a random sample [24]. However, the size of any reasonable sample depends on the size of the smallest optimal component. Moreover, it is often outperformed by other methods (e.g. [24, 25]). Another approach using HAC is presented in [21]. It aims at finding the best local modes of the data set in a reduced m^*-dimensional space and applies HAC only on these modes. However, this method is time-consuming and the choice of m^* is crucial [21, p. 5, 13]. Moreover, in [22] it is outperformed by simple random methods.

[25] presents a density based approach which not only determines an initial solution but also the number of components. It initializes the means by points which have a "high concentration" of neighbors. To this end, the size m of the neighborhood of a point (i.e., the minimum number of points in a cluster) has to be fixed in advance. In our experiments, we found that the performance crucially depends on the choice of m. Hence, we ignore this method in this paper.

In [27], a greedy algorithm is presented which constructs a sequence of mixture models with 1 through K components. Given a model θ_k with k components, it constructs several new candidates with $k + 1$ components. Each candidate is constructed by adding a new component to θ_k and executing the EM algorithm. Hence, this method is only useful if several values of K need to be considered.

In [20] a modification of the Gonzalez algorithm for GMMs is presented. Furthermore, there are some practical applications using the K-means++ algorithm for the initialization of GMMs (e.g., in [19] GMMs are used for speech recognition). Additional initialization methods can be found e.g. in [6, 15, 21, 26].

1.3   Our Contribution

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

In this paper, we present new methods that are fast and do not require choosing sensitive hyperparameters. These methods can be seen as adaptions of the K-means++ algorithm [3] and the Gonzalez algorithm [17] and as an extension of the initial work in [19, 20]. We present experiments indicating the superiority of our methods compared to a large number of alternative methods.

2   Baseline Algorithms

The most widely used initializations start by choosing K data points:

Unif draws K points independently and uniformly at random from X.

HAC computes a uniform sample S of size $s \cdot |X|$ of the input set X and executes hierarchical clustering with average linkage cost on S.

G executes the algorithm given in [17], which yields a 2-approximation for the discrete radius K-clustering problem. Iteratively, it chooses the point with the largest Euclidean distance from the already chosen points.

KM++ executes the K-Means++ algorithm [3], which has been designed for the K-means problem. In each round, **KM++** samples a data point (i.e. the next mean) from the given data set X with probability proportional to its K-means cost (with respect to the points chosen so far). In expectation, the resulting K-means costs are in $\mathcal{O}(\log(K) \cdot \text{opt})$. **KM++** is particularly interesting since the K-means algorithm is a special case of the EM algorithm [8].

Then, given K data points, Algorithm 1 is used to create a GMM, which is then the initial solution that is fed to the EM algorithm.

Algorithm 1. Means2GMM$(X \subset \mathbb{R}^D, \mathcal{C} \subset \mathbb{R}^D, |\mathcal{C}| = k)$

1: Derive a partition $\{C_1, \ldots, C_k\}$ by assigning each $x \in X$ to a closest point in $\mathcal{C}$.
2: **for** $l = 1, \ldots, k$ **do**
3:     Set $\mu_l := 1/|C_l| \sum_{x \in C_l} x$, $w_l := |C_l|/|X|$, $\Sigma_l := 1/|C_l| \sum_{x \in C_l} (x - \mu_l)(x - \mu_l)^T$.
4:     If Σ_l is not positive definite, set $\Sigma_l := 1/(D \cdot |C_l|) \sum_{x \in C_l} \|x - \mu_l\|^2 \cdot I_D$
5:     If Σ_l is still not positive definite, set $\Sigma_l := I_D$
6: **return** $\theta = \{(w_l, \mu_l, \Sigma_l)\}_{l=1,\ldots,k}$.

A popular alternative is to apply the K-means algorithm with the chosen data points before executing Algorithm 1. The main idea behind this is that starting the EM algorithm with a coarse initial solution (where e.g. not all clusters are covered well) might impose a high risk of getting stuck at a poor local minimum. To avoid this problem, one first runs a different algorithm that optimizes a function similar to the likelihood, i.e. the K-means costs (cf. [8, p. 427, 443]). We refer to the K-means algorithm as an *intermediate* algorithm and indicate its use by the postfix "$_{km}$".

3   Adaptive Seeding for GMMs

Our new adaptive methods construct a sequence of models with $k = 1$ through $k = K$ components adaptively. Given a $(k-1)$-GMM θ_{k-1}, our methods try to choose a point from the data set that is not described well by the given θ_{k-1} and which is hopefully a good representative of a component of an optimal k-GMM. The idea behind is that this point can lead us to a significant refinement of θ_{k-1}.

Choosing a Point. The negative log-likelihood of a point $x \in \mathbb{R}^D$, given the GMM θ_{k-1}, measures how well x is described by θ_{k-1}.[1] Unfortunately, it may take negative values and does not scale with the data set and the GMM.[2] This also applies to the minimum component-wise negative log-likelihood

$$\min \left\{ - \log \left((2\pi)^{D/2} |\Sigma_l|^{1/2} \right) + \tfrac{1}{2}(x - \mu_l)^T \Sigma_l^{-1}(x - \mu_l) \mid (w_l, \mu_l, \Sigma_l) \in \theta_{k-1} \right\},$$

due to the first summand. Hence, we use the minimum Mahalanobis distance

$$m(x|\theta_{k-1}) := \min \left\{ (x - \mu_l)^T \Sigma_l^{-1}(x - \mu_l) \mid (w_l, \mu_l, \Sigma_l) \in \theta_{k-1} \right\}.$$

Our first method chooses the point $x \in X$ maximizing $m(x|\theta_{k-1})$. Since these points are more likely to be outliers, we also consider choosing a point only from a uniform sample of X, which is chosen in advance (cf. Algorithm 2).

Our second method chooses point $x \in X$ with probability $\propto m(x|\theta_{k-1})$ (cf. Algorithm 3). In order to reduce the probability to choose an outlier, we also consider adding an α portion of uniform distribution, i.e. drawing x with probability

$$m_\alpha(x|\theta_{k-1}) := \alpha \cdot m(x|\theta_{k-1}) / \textstyle\sum_{y \in X} m(y|\theta_{k-1}) + (1 - \alpha) \cdot 1/|X|.$$

Constructing a GMM. Then, given a point $x \in X$ and the means of θ_{k-1}, we construct a k-GMM. In our first experiments, we used Algorithm 1 to construct a k-GMM. However, it turned out that estimating only spherical covariance matrices (with variable variances) yields a better performance than estimating full covariance matrices. We assume that this is due to the fact that θ_{k-1} is only a very coarse estimate of $(k-1)$-components of an optimal k-GMM. Formally, we replace the covariance update in Line 3 of Algorithm 1 by $\Sigma_l = 1/(D \cdot |C_l|) \sum_{x \in C_l} \|x - \mu_l\|^2 \cdot I_D$. We denote this version of Algorithm 1 as `Means2SphGMM`. Given the resulting k-GMM, our methods then again choose a new point from X as already described above.

Intermediate Algorithm. Recall that some baselines use the K-means algorithm as an intermediate algorithm (cf. Sect. 2). Since we do not only construct means but GMMs, we apply a hard-clustering variant of the EM algorithm, i.e. the Classification EM algorithm (CEM) [10], and let it only estimate spherical covariances. We indicate its use by the postfix "$_{cem}$".

Algorithms 2 and 3 summarize our methods. Note that we do not optimize the hyperparameters α and s in our experiments.

Comparison to Baselines. Our adaptive initializations can be seen as adaptions of the `Gonzalez` and `Kmeans++` algorithm. Simply speaking, these methods assume that each component is represented by a Gaussian with *the same*

---

[1] The (inverse) pdf is unsuited due to the exponential behavior (over-/underflows).

[2] Even wrt. a single Gaussian, i.e. $\log \mathcal{N}(c \cdot x | c \cdot \mu, c^2 \cdot \Sigma) = \log \mathcal{N}(x|\mu, \Sigma) - D \ln(c)$.

fixed spherical covariance matrix and fixed uniform weights. In contrast, our goal is to estimate also the covariance matrices *adaptively*. Furthermore, in [20] another adaption of the Gonzalez algorithm is presented, which we denote by `KwedlosGonzalez (KG)`. Unlike our method, it chooses weights and covariance matrices randomly and independently of the means (and of each other).

Algorithm 2. `SphericalGonzalez (SG)`

Require: $X \subset \mathbb{R}^D, K \in \mathbb{N}, s \in (0,1]$
1: $\theta_1 :=$ optimal 1-MLE wrt. X
2: If $s < 1$, let S be a uniform sample of X of size $\lceil s \cdot |X| \rceil$. Otherwise, set $S = X$.
3: **for** $k = 2, \dots, K$ **do**
4:     $p := \arg\max_{x \in S} m(x|\theta_{k-1})$
5:     $M_k := \{\mu | (\cdot, \mu, \cdot) \in \theta_{k-1}\} \cup \{p\}$
6:     $\theta_k :=$ `Means2SphGMM`(X, M_k)
7: (optional) Run CEM algorithm
8: **return** θ_K

Algorithm 3. `Adaptive (Ad)`

Require: $X \subset \mathbb{R}^D, K \in \mathbb{N}, \alpha \in [0,1]$
1: $\theta_1 :=$ optimal 1-MLE wrt. X
2: **for** $k = 2, \dots, K$ **do**
3:     Draw p from X with probability $m_\alpha(p|\theta_{k-1})$.
4:     $M_k := \{\mu | (\cdot, \mu, \cdot) \in \theta_{k-1}\} \cup \{p\}$
5:     $\theta_k :=$ `Means2SphGMM`(X, M_k)
6: (optional) Run CEM algorithm
7: **return** θ_K

---

4   Experiments

We evaluated all presented methods with respect to artificial as well as real world data sets. Our implementation as well as the complete results are available at [9]. Due to space limitations, we omit the results of those algorithms that are consistently outperformed by others. These results are available at [9] as well.

Quality Measure. Recall that the goal of our paper (and the EM algorithm) is to find a maximum likelihood estimate (MLE). Thus, the *likelihood* is not only the common but also the appropriate way of evaluating our methods.

Other measures need to be treated with caution: Some authors consider their methods only with respect to *some specific tasks* where fitting a GMM to some data is part of some framework. Hence, any observed effects might be due to several reasons (i.e. correlations). In particular, GMMs are often compared with respect to certain *classifications*. As already pointed out by [14], the class labels of real world data sets do not necessarily correspond to the structure of an MLE. The same holds for data sets and classifications generated according to some GMM. Moreover, a *cross-validation*, that examines whether methods overfit models to training data, is not reasonable, since our methods do *not* aim at finding a model that does not fit too well to the given data set. Finally, one should not generate data sets according to some *"ground truth" GMM* θ_{gt} and compare GMMs with θ_{gt} because in many cases (e.g. small $|X|$) one cannot expect θ_{gt} to be a good surrogate of the MLE.

Setup. Recall that in Algorithms 2 and 3 hyperparameters α and s are used. We do not optimize them, but test reasonable values, i.e. $\alpha \in \{0.5, 1\}$ and

$s \in \{0.1, 1\}$. We execute each method with 30 different seeds. On the basis of some initial experiments, we decided to execute the intermediate algorithms for 25 rounds and the EM algorithm for 50 rounds. If only the EM algorithm is applied, then we execute it for 75 rounds.

4.1 Artificial Data Sets

Data Generation. We create 192 test sets, each containing 30 data sets that share certain characteristics [9]. For each data set, we first create a GMM[3] at random but control the following properties: First, the components of a GMM can either be spherical or elliptical. We describe the eccentricity of 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, components have different weights.

Fourth, components can overlap more or less. Following [12], we define the separation parameter $c_\theta = \min_{l \neq k} \|\mu_l - \mu_k\| / \sqrt{\max\{\text{trace}(\Sigma_l), \text{trace}(\Sigma_k)\}}$. In high dimension $D \gg 1$, $c_\theta = 2$ corresponds to almost completely separated clusters (i.e. points generated by the same component), while $c_\theta \in \{0.5, 1\}$ indicates a slight but still negligible overlap [11]. However, in small dimension, $c_\theta \in \{0.5, 1\}$ corresponds to significant overlaps between clusters, while $c_\theta = 2$ implies rather separated clusters (cf. Fig. 1).

(a) $c_\theta = 0.5$ (b) $c_\theta = 1$ (c) $c_\theta = 2$

(d) $c_\theta = 0.5$ (e) $c_\theta = 1$ (f) $c_\theta = 2$

Fig. 1. Examples for different separation parameters. Figures show orthogonal projections to random plane. Data sets in (a)–(c) have $D = 3$. (d)–(f) have $D = 10$.

We generate random GMMs as follows. Initially, we draw means uniformly at random from a cube with a fixed side length. For the weights, we fix some $c_w \geq 0$, construct a set of weights $\{2^{c_w \cdot i} / \sum_{j=1}^{K} 2^{c_w \cdot j}\}_{i=1,\ldots,K}$ and assign these weights randomly. To control the sizes and the eccentricity, we fix the minimum and maximum eigenvalue and draw the remaining values uniformly at random from the interval. Then, we set $\Sigma_k = Q^T \text{diag}(\lambda_{k1}^2, \ldots, \lambda_{kD}^2) Q$ for a random $Q \in \text{SO}(D)$. Finally, the means are scaled as to fit the predefined separation parameter. Given the resulting GMM θ, we first draw some points according to θ. Then, we construct a bounding box, elongate its side lengths by a factor 1.2, and draw noise points uniformly at random from the resized box.

We created a test set (i.e. 30 data sets) for each combination of the following parameters: $K = 20$, $|X| \in \{1000, 5000\}$, $D \in \{3, 10\}$, $c_\theta \in \{0.5, 1, 2\}$, $c_w \in \{0.1, 1\}$, different combinations of size and eccentricity (i.e., equal size and $e_k = 10$,
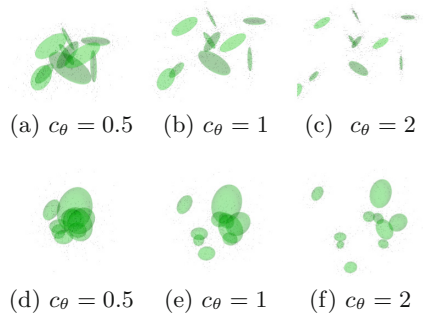
---

[3] As explained before, our goal is *not* to identify these GMMs.

equal size and $e_k \in [1, 10]$, different size and $e_k = 1$, different size and $e_k \in [1, 10]$), and without or with 10 % noise points.

Evaluation Method. We consider the *initial solutions* produced by the initialization (possibly followed by an intermediate algorithm) and the *final solutions* obtained by running the EM algorithm afterwards. For each data set, we compute the average log-likelihood of the initial and final solution, respectively. Based on these averages, we create rankings of the algorithms[4]. Then, we compute the average rank (and standard deviation of the rank) of each algorithm over all datasets matching certain properties.

Results. In general, we observe that one should use an intermediate algorithm before applying the EM algorithm. Thus, we omit the results of some methods [9].

Data without Noise. For these rather simple data sets, there is no method that constantly outperforms *all others*. Nonetheless, it is always one of our adaptive or the $\mathtt{G}_{km}$ initialization that performs best.

Table 1. Average ranks ($\pm$ std.dev.) for generated data with $K = 20$, $|X| = 1000$, $D = 10$, **different weights**, and **without noise.**

	separation $c_\theta = 0.5$		separation $c_\theta = 1$		separation $c_\theta = 2$	
	initial	final	initial	final	initial	final
$\mathtt{SG}\left(s = \frac{1}{10}\right)$	7.53±1.08	3.58±1.93	7.29±0.86	5.08±2.95	7.14±0.57	7.28±2.31
$\mathtt{SG}(s = 1)$	8.00±1.52	3.26±2.27	8.77±0.68	5.53±3.42	8.72±0.53	7.44±2.60
$\mathtt{KG}(s = 1)$	10.00±0.00	9.75±0.72	10.00±0.00	7.68±2.52	10.00±0.00	2.38±1.34
$\mathtt{Unif}_{km}$	1.56±0.74	8.39±1.15	2.19±0.61	7.46±1.88	2.98±0.13	7.08±2.14
$\mathtt{G}_{km}$	3.34±1.33	6.03±2.32	2.38±0.86	5.16±2.79	1.23±0.46	1.99±1.31
$\mathtt{KM++}_{km}$	1.85±0.64	7.87±1.14	1.43±0.64	6.22±2.60	1.78±0.41	3.75±2.25
$\mathtt{SG}\left(s = \frac{1}{10}\right)_{cem}$	6.15±0.60	3.95±1.24	6.30±0.68	4.71±2.54	6.10±0.40	6.65±2.13
$\mathtt{SG}(s = 1)_{cem}$	6.47±1.51	3.20±2.48	7.32±1.26	5.12±3.31	8.03±0.61	7.80±2.61
$\mathtt{Ad}(\alpha = 1)_{cem}$	5.16±1.84	4.31±1.77	4.59±0.64	3.88±1.75	4.47±0.50	5.06±1.40
$\mathtt{Ad}\left(\alpha = \frac{1}{2}\right)_{cem}$	4.93±2.11	4.67±1.93	4.72±0.80	4.15±1.77	4.53±0.50	5.58±1.71

The results depicted in Tables 1 and 2 suggest that, regardless of the weights, the performance is determined by the separation. Furthermore, a good initial solution does not imply a good final solution. Given overlap ($c_\theta = 0.5$) or moderate separation ($c_\theta = 1$), $\mathtt{SG}_{cem}(s = 1)$ and $\mathtt{Ad}_{cem}(\alpha = 1)$ work best, even though their initial solutions have low average ranks compared to $\mathtt{KM++}_{km}$. Given higher separation ($c_\theta = 2$), we expect it to be easier to identify clusters and that skewed covariance matrices do not matter much if means are assigned properly in the first place. Indeed, the simple $\mathtt{G}_{km}$ and $\mathtt{KG}$ do the trick.

---

[4] Averaging the (average) log-likelihood values over different data sets is not meaningful since the optimal log-likelihoods may deviate significantly.

Table 2. Average ranks ($\pm$ std.dev.) for generated data with $K = 20$, $|X| = 1000$, dimension $D = 10$, **equal weights**, and **without noise**.

	separation $c_\theta = 0.5$		separation $c_\theta = 1$		separation $c_\theta = 2$	
	initial	final	initial	final	initial	final
$\texttt{SG}\left(s = \frac{1}{10}\right)$	7.58±0.98	3.98±1.87	7.36±0.73	5.02±2.97	7.08±0.41	7.35±2.24
$\texttt{SG}(s = 1)$	8.11±1.53	3.62±2.58	8.67±0.85	5.67±3.20	8.79±0.43	7.77±2.59
$\texttt{KG}(s = 1)$	10.00±0.00	9.54±0.96	10.00±0.00	7.97±2.40	10.00±0.00	2.38±1.23
$\texttt{Unif}_{km}$	1.44±0.70	8.36±1.25	2.14±0.61	7.28±1.83	2.98±0.16	7.00±1.98
$\texttt{G}_{km}$	3.39±1.34	6.12±2.17	2.53±0.83	6.04±2.76	1.27±0.50	1.82±1.08
$\texttt{KM++}_{km}$	1.91±0.55	7.82±1.30	1.35±0.56	5.89±2.77	1.75±0.43	3.77±2.50
$\texttt{SG}\left(s = \frac{1}{10}\right)_{cem}$	6.18±0.62	3.58±1.31	6.44±0.87	4.26±2.43	6.02±0.13	6.55±1.95
$\texttt{SG}(s = 1)_{cem}$	6.49±1.44	3.17±2.73	7.35±1.13	5.30±3.23	8.12±0.45	8.04±2.37
$\texttt{Ad}(\alpha = 1)_{cem}$	5.03±1.75	4.12±1.79	4.49±0.64	3.48±1.61	4.42±0.50	4.90±1.35
$\texttt{Ad}\left(\alpha = \frac{1}{2}\right)_{cem}$	4.87±1.99	4.69±1.91	4.67±0.65	4.09±1.73	4.58±0.50	5.42±1.38

Table 3. Average ranks ($\pm$ std.dev.) for generated data with $K = 20$, $|X| = 1000$, dimension $D = 10$, and **without noise**. Only final solutions.

	equal weights			different weights		
	spherical	elliptical	both	spherical	elliptical	both
$\texttt{SG}\left(s = \frac{1}{10}\right)$	6.13±2.63	5.22±2.80	5.45±2.78	6.03±2.65	5.07±2.90	5.31±2.87
$\texttt{SG}(s = 1)$	6.64±2.99	5.37±3.30	5.69±3.27	6.04±3.22	5.20±3.28	5.41±3.28
$\texttt{KG}(s = 1)$	6.78±3.19	6.58±3.59	6.63±3.49	6.81±3.20	6.53±3.65	6.60±3.54
$\texttt{Unif}_{km}$	7.67±1.48	7.50±1.91	7.54±1.81	7.64±1.64	7.64±1.92	7.64±1.85
$\texttt{G}_{km}$	3.03±2.26	5.20±2.92	4.66±2.92	2.83±2.33	4.91±2.78	4.39±2.82
$\texttt{KM++}_{km}$	5.44±2.60	5.96±2.87	5.83±2.81	5.57±2.66	6.07±2.69	5.95±2.69
$\texttt{SG}\left(s = \frac{1}{10}\right)_{cem}$	4.77±2.61	4.81±2.23	4.80±2.33	5.68±2.19	4.91±2.35	5.10±2.33
$\texttt{SG}(s = 1)_{cem}$	6.53±3.16	5.16±3.46	5.51±3.43	6.07±3.30	5.14±3.40	5.37±3.39
$\texttt{Ad}(\alpha = 1)_{cem}$	3.62±1.61	4.34±1.69	4.16±1.69	4.02±1.82	4.55±1.66	4.42±1.71
$\texttt{Ad}\left(\alpha = \frac{1}{2}\right)_{cem}$	4.38±1.78	4.86±1.75	4.74±1.77	4.30±1.86	4.97±1.88	4.80±1.89

Table 3 shows that $\texttt{Ad}_{cem}(\alpha = 1)$ works well for elliptical data, while $\texttt{G}_{km}$ should be chosen for spherical data. Recall that there are no noise points yet. We expect that the performance of $\texttt{G}_{km}$ degenerates in the presence of noise since it is prone to choose outliers. Overall, given data sets without noise, $\texttt{Ad}_{cem}(\alpha = 1)$ performs best.

Noisy Data. When introducing noise, our adaptive methods are still among the best methods, while the performance of some others degenerates significantly. Tables 4 and 5 show that $\texttt{SG}_{cem}$ and $\texttt{Ad}_{cem}$ still work well for $c_w \leq 1$ and, in contrast to data without noise, also for separated instances ($c_w = 2$). KG and $\texttt{G}_{km}$ are now among the methods with the lowest average rank. This is not a surprise since our noise contains outliers. From the results depicted in Table 6 one can draw the same conclusion, i.e. KG and $\texttt{G}_{km}$ can not handle noisy data. For noisy data, our $\texttt{Ad}_{cem}$ methods outperform the others.

Table 4. Average ranks ($\pm$ std.dev.) for generated data with $K = 20$, $|X| = 1000$, dimension $D = 10$, **different weights**, and **10 % noise**.

	separation $c_\theta = 0.5$		separation $c_\theta = 1$		separation $c_\theta = 2$	
	initial	final	initial	final	initial	final
$\text{SG}\left(s = \frac{1}{10}\right)$	8.41±0.68	3.40±1.75	8.22±0.64	4.44±2.45	8.06±0.68	5.46±2.42
$\text{SG}(s = 1)$	8.25±0.98	3.46±2.60	8.67±0.47	4.13±3.02	8.74±0.44	5.93±2.87
$\text{KG}(s = 1)$	10.00±0.00	9.95±0.22	10.00±0.00	9.72±0.76	10.00±0.00	9.02±1.49
Unif_{km}	1.98±0.89	8.65±1.03	1.05±0.25	7.89±1.71	1.19±0.49	7.34±1.87
G_{km}	4.29±1.29	5.31±1.76	4.17±0.98	6.59±1.43	3.97±0.96	7.16±1.38
KM++_{km}	3.23±0.98	6.45±1.40	2.27±0.60	6.83±1.62	2.17±0.60	6.69±1.69
$\text{SG}\left(s = \frac{1}{10}\right)_{cem}$	6.06±0.55	4.26±1.36	6.04±0.20	3.90±0.90	6.01±0.091	3.65±1.27
$\text{SG}(s = 1)_{cem}$	6.31±1.43	3.49±2.64	7.08±0.39	3.80±2.89	7.19±0.42	5.00±3.01
$\text{Ad}(\alpha = 1)_{cem}$	3.64±1.88	4.61±2.26	4.05±0.90	3.57±2.03	3.76±1.26	2.06±1.39
$\text{Ad}\left(\alpha = \frac{1}{2}\right)_{cem}$	2.83±2.26	5.42±2.70	3.46±0.89	4.12±2.38	3.92±0.78	2.69±1.45

Table 5. Average ranks ($\pm$ std.dev.) for generated data sets with $K = 20$, $|X| = 1000$, dimension $D = 10$, **equal weights**, and **10 % noise**.

	separation $c_\theta = 0.5$		separation $c_\theta = 1$		separation $c_\theta = 2$	
	initial	final	initial	final	initial	final
$\text{SG}\left(s = \frac{1}{10}\right)$	8.57±0.62	3.38±1.92	8.18±0.65	4.17±2.28	7.94±0.77	5.02±2.44
$\text{SG}(s = 1)$	8.05±1.08	3.19±2.23	8.68±0.47	3.77±2.68	8.73±0.44	5.47±2.96
$\text{KG}(s = 1)$	10.00±0.00	9.93±0.35	10.00±0.00	9.62±0.87	10.00±0.00	7.85±2.22
Unif_{km}	1.92±0.87	8.83±0.77	1.02±0.13	8.39±1.22	1.11±0.31	8.18±1.63
G_{km}	4.47±0.99	5.53±1.75	4.01±1.01	6.74±1.51	3.53±0.83	7.18±1.28
KM++_{km}	3.20±1.07	6.66±1.22	2.08±0.31	7.04±1.46	1.93±0.37	7.66±1.56
$\text{SG}\left(s = \frac{1}{10}\right)_{cem}$	6.08±0.53	4.47±1.31	6.03±0.18	3.92±0.97	6.00±0.00	3.64±1.11
$\text{SG}(s = 1)_{cem}$	6.20±1.43	3.14±2.43	7.10±0.40	3.88±2.75	7.33±0.47	5.13±2.76
$\text{Ad}(\alpha = 1)_{cem}$	3.62±1.85	4.42±2.35	4.20±0.79	3.54±2.11	4.15±0.82	2.27±1.59
$\text{Ad}\left(\alpha = \frac{1}{2}\right)_{cem}$	2.89±2.45	5.45±2.47	3.69±0.74	3.92±2.29	4.28±0.66	2.60±1.51

Table 6. Average ranks ($\pm$ std.dev.) for generated data sets with $K = 20$, $|X| = 1000$, dimension $D = 10$, and **10 % noise**. Only final solutions.

	equal weights			different weights		
	spherical	elliptical	both	spherical	elliptical	both
$\text{SG}\left(s = \frac{1}{10}\right)$	5.38±2.26	3.79±2.20	4.19±2.32	5.53±2.30	4.07±2.29	4.43±2.38
$\text{SG}(s = 1)$	4.98±2.79	3.87±2.76	4.14±2.81	5.53±3.14	4.17±2.90	4.51±3.02
$\text{KG}(s = 1)$	8.61±2.03	9.31±1.49	9.13±1.66	9.32±1.28	9.64±0.95	9.56±1.05
Unif_{km}	8.36±1.36	8.50±1.26	8.47±1.28	7.63±1.72	8.07±1.63	7.96±1.66
G_{km}	6.70±1.69	6.41±1.67	6.49±1.68	6.42±1.87	6.33±1.66	6.35±1.71
KM++_{km}	7.09±1.71	7.13±1.39	7.12±1.48	6.46±1.71	6.72±1.53	6.66±1.58
$\text{SG}\left(s = \frac{1}{10}\right)_{cem}$	4.00±1.45	4.01±1.09	4.01±1.18	4.03±1.34	3.90±1.17	3.94±1.22
$\text{SG}(s = 1)_{cem}$	5.04±2.79	3.72±2.69	4.05±2.77	4.88±2.98	3.84±2.86	4.10±2.92
$\text{Ad}(\alpha = 1)_{cem}$	2.01±1.29	3.88±2.27	3.41±2.22	2.31±1.57	3.78±2.25	3.41±2.19
$\text{Ad}\left(\alpha = \frac{1}{2}\right)_{cem}$	2.83±1.72	4.37±2.50	3.99±2.42	2.88±1.78	4.48±2.57	4.08±2.50

Table 7. Average ranks ($\pm$ std.dev.) for generated data ($K = 20$, $|X| = 1000$, $D = 3$).

	without noise		noisy	
	initial	final	initial	final
$\mathtt{SG}\big(s = \frac{1}{10}\big)$	7.31±0.63	7.94±1.39	8.05±0.70	7.93±1.31
$\mathtt{SG}(s = 1)$	8.90±0.39	8.56±1.98	8.72±0.45	7.99±2.19
$\mathtt{KG}(s = 1)$	9.94±0.42	3.28±1.98	10.00±0.00	8.09±1.46
$\mathtt{Unif}_{km}$	2.82±0.58	4.63±1.40	3.38±0.94	2.76±1.38
$\mathtt{G}_{km}$	1.93±1.04	2.80±1.97	4.43±1.59	6.01±1.59
$\mathtt{KM++}_{km}$	1.51±0.60	1.99±1.21	2.96±1.12	2.35±1.57
$\mathtt{SG}\big(s = \frac{1}{10}\big)_{cem}$	6.10±0.42	7.46±1.14	5.75±0.82	6.07±1.16
$\mathtt{SG}(s = 1)_{cem}$	7.65±0.94	8.83±1.74	7.04±0.86	8.16±2.10
$\mathtt{Ad}(\alpha = 1)_{cem}$	4.35±0.83	4.66±1.48	2.54±1.43	3.02±1.13
$\mathtt{Ad}\big(\alpha = \frac{1}{2}\big)_{cem}$	4.49±0.74	4.85±1.51	2.12±1.36	2.62±1.28

Low Dimensional or High Sample Size Data. We expect that, if the dimension is low or the sample size is large enough, it is generally easier to identify clusters. Indeed the results differ significantly from our previous results. In general, the $\mathtt{KM++}_{km}$ and $\mathtt{Unif}_{km}$ perform best. For data sets with $D = 3$ and $|X| = 1000$, Table 7 shows that the $\mathtt{KM++}_{km}$ method works well even in the presence of noise. However, if we are given noise *and* small separation, the simple $\mathtt{Unif}_{km}$ does well. We also increased the sample size to $|X| = 5000$ *and* the dimension to $D = 10$, expecting that the higher sample size can make up for the higher dimension (results available in [9]). Indeed, for data sets without noise, where clusters can presumably be identified easier, $\mathtt{KM++}_{km}$ still suffices. However, given noise or too small a separation, our $\mathtt{Ad}_{cem}$ methods and the simple $\mathtt{Unif}_{km}$ work better.

4.2   Real World Data Sets

We use four publicly available data sets: *Covertype* ($|X| = 581\,012$, $D = 10$ real-valued features) [4]; two *Aloi* data sets ($|X| = 110\,250$, $D \in \{27, 64\}$) based on color histograms in HSV color space [18] from data provided by the ELKI project [2] and the Amsterdam Library of Object Images [16]; *Cities* ($|X| = 135\,082$, $D = 2$) is a projection of the coordinates of cities with a population of at least 1000 [1]; *Spambase* ($|X| = 4601$, $D = 10$ real-valued features) [4].

The results are depicted in Fig. 2: For *Aloi* ($D = 27$) and *Spambase* ($K = 3$), $\mathtt{SG}_{cem}(s = 1)$ is considerably better than the other methods. For *Cities* and *Spambase* ($K = 10$), $\mathtt{SG}(s = 1)$ does better (without running the CEM). For *Aloi* ($D = 64$) and the *Covertype*, $\mathtt{Ad}_{cem}(\alpha = 1)$ works better than the others.

4.3   Time Measurement

The run times of the compared methods match our expectation (cf. Table 8). First, (intermediate) steps of the CEM algorithm are faster than (more) steps of the EM algorithm. Second, sampling and running methods on a random subset of the data should in general reduce the run time.
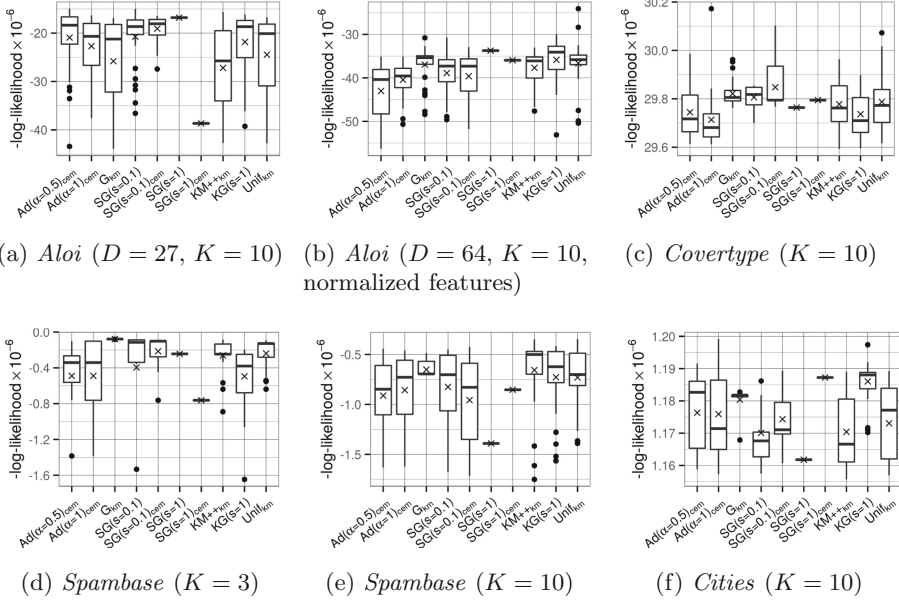
(a) *Aloi* ($D = 27$, $K = 10$)

(b) *Aloi* ($D = 64$, $K = 10$, normalized features)

(c) *Covertype* ($K = 10$)

(d) *Spambase* ($K = 3$)

(e) *Spambase* ($K = 10$)

(f) *Cities* ($K = 10$)

Fig. 2. Results for the real world data sets depicted as boxplots (final solutions only).

Table 8. Average run times (in seconds) over 12 data sets with $|X| = 10^3$ and $D = 10$ and different runs per data set using an Intel Core i7-3770 CPU (3.40 GHz, 8 GB RAM).

$SG(s = 1)$	0.314	$SG(s = 0.1)$	0.307	$Ad_{cem}(\alpha = 1)$	0.226	$Unif_{km}$	0.206
$SG_{cem}(s = 1)$	0.225	$SG_{cem}(s = 0.1)$	0.197	$Ad_{cem}(\alpha = 0.5)$	0.253	$HAC(s = 1)$	1.588
G_{km}	0.205	$KG(s = 1)$	0.315				

5 Conclusion and Future Work

If you need a fast and simple method, then we suggest to use one of the following methods: Given a data set with a large number of points per cluster or low dimension, the K-means++ initialization followed by the K-means algorithm and `Means2GMM` should do well. Otherwise, we recommend our new methods `Ad` and `SG` followed by the spherical CEM algorithm, especially if your data is presumably noisy. Last but not least, whatever you prefer, we suggest trying intermediate steps of the spherical CEM or K-means algorithm.

For the K-means++ algorithm and the Gonzalez algorithm there are provable guarantees. We hope that our results are a good starting point for a theoretical analysis that will transfer these results to the MLE problem for GMMs.

References

1. GeoNames geographical database. http://www.geonames.org/
2. Achtert, E., Goldhofer, S., Kriegel, H.-P., Schubert, E., Zimek, A.: Evaluation of Clusterings - Metrics and Visual Support. http://elki.dbs.ifi.lmu.de/wiki/DataSets/MultiView
3. Arthur, V.: k-means++: The advantages of careful seeding. In: SODA 2007 (2007)
4. Asuncion: UCI machine learning repository (2007). http://www.ics.uci.edu/mlearn/MLRepository.html
5. Baudry, J.-P., Celeux, G.: EM for mixtures. Stat. Comput. **25**(4), 713–726 (2015)
6. Biernacki, C.: Initializing EM using the properties of its trajectories in Gaussian mixtures. Stat. Comput. **14**(3), 267–279 (2004)
7. Biernacki, C., Celeux, G., Govaert, G.: Choosing starting values for the EM algorithm for getting the highest likelihood in multivariate Gaussian mixture models. Comput. Stat. Data Anal. **41**(3–4), 561–575 (2003)
8. Bishop, C.: Pattern Recognition and Machine Learning. Information Science and Statistics. Springer, Secaucus (2006)
9. Bujna, K., Kuntze, D.: Supplemental Material. http://www-old.cs.upb.de/fachgebiete/ag-bloemer/forschung/clusteranalyse/adaptive_seeding_for_gmms.html
10. Celeux, G., Govaert, G.: A classification EM algorithm for clustering and two stochastic versions. Comput. Stat. Data Anal. **14**(3), 315–332 (1992)
11. Dasgupta, S.: Experiments with random projection. In: UAI 2000 (2000)
12. Dasgupta, S.: Learning mixtures of gaussians. In: FOCS 1999 (1999)
13. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. J. R. Stat. Soc. Ser. B Stat. Methodol. **39**(1), 1–38 (1977)
14. Färber, I., Günnemann, S., Kriegel, H., Kröger, P., Müller, E., Schubert, E., Seidl, T., Zimek, A.: On using class-labels in evaluation of clusterings. In: MultiClust 2010 (2010)
15. Fayyad, U., Reina, C., Bradley, P.S.: Initialization of iterative refinement clustering algorithms. In: KDD 1998 (1998)
16. Geusebroek, J.M., Burghouts, G.J., Smeulders, A.W.M.: The Amsterdam library of object images. Int. J. Comput. Vis. **6**(1), 103–112 (2005)
17. Gonzalez, T.F.: Clustering to minimize the maximum intercluster distance. Theor. Comput. Sci. **38**, 293–306 (1985)
18. Kriegel, H.-P., Schubert, E., Zimek, A.: Evaluation of multiple clustering solutions. In: MultiClust 2010 (2010)
19. Krüger, A., Leutnant, V., Haeb-Umbach, R., Ackermann, M., Blömer, J.: On the initialization of dynamic models for speech features. In: Sprachkommunikation 2010 (2010)
20. Kwedlo, W.: A new random approach for initialization of the multiple restart EM algorithm for Gaussian model-based clustering. Pattern Anal. Appl. **18**(4), 757–770 (2015)
21. Maitra, R.: Initializing partition-optimization algorithms. IEEE/ACM Trans. Comput. Biol. Bioinform. **6**(1), 144–157 (2009)
22. Maitra, R., Melnykov, V.: Simulating data to study performance of finite mixture modeling and clustering algorithms. J. Comput. Graph. Stat. **19**(2), 354–376 (2010)

23. McLachlan, G.J., Krishnan, T.: The EM Algorithm and Extensions. Wiley Series in Probability and Statistics, 2nd edn. Wiley-Interscience, New York (2008)
24. Meilă, M., Heckerman, D.: An experimental comparison of several clustering and initialization methods. In: UAI 1998. Morgan Kaufmann Inc., San Francisco (1998)
25. Melnykov, V., Melnykov, I.: Initializing the EM algorithm in Gaussian mixture models with an unknown number of components. Comput. Stat. Data Anal. **56**, 1381–1395 (2011)
26. Thiesson, B.: Accelerated quantification of Bayesian networks with incomplete data. University of Aalborg (1995)
27. Verbeek, J.J., Vlassis, N., Kröse, B.: Efficient greedy learning of Gaussian mixture models. Neural Comput. **15**(2), 469–485 (2003)