Can clustering scale sublinearly with its clusters? A variational EM acceleration of GMMs and k-means

Dennis Forster
Machine Learning, University of Oldenburg &
FIAS, Goethe-University Frankfurt, Germany

Jörg Lücke
Machine Learning
University of Oldenburg, Germany

Abstract

One iteration of standard k-means (i.e., Lloyd's algorithm) or standard EM for Gaussian mixture models (GMMs) scales linearly with the number of clusters C, data points N, and data dimensionality D. In this study, we explore whether one iteration of k-means or EM for GMMs can scale sublinearly with C at run-time, while improving the clustering objective remains effective. The tool we apply for complexity reduction is variational EM, which is typically used to make training of generative models with exponentially many hidden states tractable. Here, we apply novel theoretical results on truncated variational EM to make tractable clustering algorithms more efficient. The basic idea is to use a partial variational E-step which reduces the linear complexity of $\mathcal{O}(NCD)$ required for a full E-step to a sublinear complexity. Our main observation is that the linear dependency on C can be reduced to a dependency on a much smaller parameter G which relates to cluster neighborhood relations. We focus on two versions of partial variational EM for clustering: variational GMM, scaling with $\mathcal{O}(NG^2D)$, and variational k-means, scaling with $\mathcal{O}(NGD)$ per iteration. Empirical results show that these algorithms still require comparable numbers of iterations to improve the clustering objective to same values as k-means. For data with many clusters, we consequently observe reductions of net computational demands between two and three orders of magnitude. More generally, our results provide substantial empirical evidence in favor of clustering to scale sublinearly with C.

Proceedings of the 21^{st} International Conference on Artificial Intelligence and Statistics (AISTATS) 2018, Lanzarote, Spain. PMLR: Volume 84.

1 Introduction

Clustering is a core Machine Learning task and one of the most widely used types of algorithms in general. k-means and Gaussian mixture models (GMMs) represent two of the most popular clustering algorithms [see, e.g., Berkhin, 2006, McLachlan and Peel, 2004]. Both require $\mathcal{O}(NCD)$ numerical operations per iteration (see abstract for definitions of N, C, and D). Active ongoing research provides ever improving bounds on convergence times in terms of iteration steps [Arthur et al., 2009, Moitra and Valiant, 2010, Xu et al., 2016], and empirical results show very fast convergence in practice [Duda et al., 2001].

Related work and own contribution. Convergence times can be strongly improved by careful seeding, and seeding methods have recently been made very efficient [Bachem et al., 2016a]. After seeding, the limiting factor for efficiency remains the complexity of one k-means or GMM iteration [e.g., Bachem et al., 2016b, for a discussion], which can be very relevant in practice [e.g., Rosenbaum and Weiss, 2015]. The reduction of complexity per iteration (e.g., of k-means) is therefore also the goal of other popular approaches. The efficiency of distance computations can, e.g., be improved by exploiting the triangle inequality [Elkan, 2003], or random projections for independence of the dimensionality D [Chan and Leung, 2017]. And by following the idea of Moore [1999] or coresets [Har-Peled and Mazumdar, 2004, Feldman et al., 2011, Lucic et al., 2017, Bachem et al., 2017] the dependency on the number of data points N can drastically be reduced. In this work our focus is on reducing the linear dependency on the number of clusters C. There are two views which may highlight the importance of this dependency: First, with very large numbers of data points N, a complexity reduction, e.g., from $\mathcal{O}(NCD)$ to $\mathcal{O}(NGD)$ with $G \ll C$ provides a very large reduction in terms of required computations. Second, if coresets are used to reduce N, and triangle inequalities or random projections are used to reduce the dependence on D, then the dependence on C remains the main bottleneck.

Our tool for complexity reduction is the application of truncated distributions to approximate exact posteriors. Truncated approximations have been applied to a number of probabilistic data models, and most frequently to multiple-causes models for which the number of latent states increases exponentially with the number of latents [Puertas et al., 2010, Dai et al., 2013, Henniges et al., 2014, Sheikh et al., 2014]. Truncated approximations do not assume a-posteriori independence like factored variational approaches [Saul et al., 1996, Jordan et al., 1999], they typically result in tight free energy bounds, and they are very efficient [Sheikh et al., 2014, Sheikh and Lücke, 2016]. For mixture models, truncated distributions represent a very natural choice. If considering the posterior of a given data point (i.e., its cluster responsibilities), then typically only few clusters contribute significantly to the over-all posterior mass. Truncated distributions approximate the full posteriors by maintaining just the C' highest posterior values while setting all other values to zero (see Fig. 1, top). Dai and Lücke [2014] used truncated distributions to make position invariant mixture models for images more efficient, Forster and Lücke [2017] applied truncated distributions to standard Poisson mixtures, and Shelton et al. [2014], Hughes and Sudderth [2016], and Lücke and Forster [2017] used truncated posteriors for standard GMMs. In all these applications, truncated distributions reduced the computational cost compared to exact EM. None of these contributions aimed to show or have shown that one complete EM iteration for clustering can scale sublinearly with C. While, e.g., Hughes and Sudderth [2016] and Forster and Lücke [2017] discuss the reduction of M-step complexity, and while k-means [see Lücke and Forster, 2017, for its variational EM formulation] has an M-step complexity of $\mathcal{O}(ND)$, all these algorithms use a full variational E-step with NC distance evaluations ($\mathcal{O}(NCD)$ computations). Other work with focus on k-means also aimed at reducing the dependency on C. Often such work remains theoretical, including runtime analysis in terms of iterations [see Kanungo et al., 2002, for a discussion of the literature], but also concrete suggestions for practically applicable versions of k-means have been made [Phillips, 2002, Shindler et al., 2011, Curtin, 2017, and others]. Shindler et al. [2011] focus on efficient memory usage. Phillips [2002] shows a reduction to $\mathcal{O}(ND\gamma + C^2D + C\log(C))$ run-time complexity per iteration but γ is large (on the order of C) initially. Curtin [2017] uses tree-based algorithms and novel pruning strategies to achieve a complexity (after initialization, per iteration, D not considered) of $\mathcal{O}(N + C\log(C))$ under mild assumptions.

To the knowledge of the authors, neither Phillips [2002], Curtin [2017] nor any other contribution shows or has considered possible a complexity reduction in C to a level similar to the one reported here. More specifically, we are not aware of work that effectively improves a k-means or GMM objective while the complexity of one iteration does not depend on C.

2 Truncated EM for GMMs

Given a set of N data points, $(\vec{y}^{\,(1)}, \ldots, \vec{y}^{\,(N)})$, our goal is to find parameters Θ that maximize the data log-likelihood, where $p(\vec{y}\,|\,\Theta)$ is given by a GMM with isotropic, equally weighted Gaussians:

$$\mathcal{L}(\Theta) = \textstyle\sum_n \log\big(p(\vec{y}^{\,(n)}|\,\Theta)\big), \text{ where} \quad (1)$$

$$p(\vec{y}\,|\,\Theta) = \tfrac{1}{C}(2\pi\sigma^2)^{-\frac{D}{2}} \textstyle\sum_c \exp\big(-\frac{1}{2\sigma^2}\|\vec{y} - \vec{\mu}_c\|^2\big). \quad (2)$$

The presumably most popular method to optimize a GMM is EM, which consists for Eq. (2) of the updates:

$$d_c^{(n)} = \|\vec{y}^{\,(n)} - \vec{\mu}_c\|, \;\; s_c^{(n)} = \frac{\exp\big(-\frac{1}{2}(d_c^{(n)}/\sigma)^2\big)}{\sum_{c'}\exp\big(-\frac{1}{2}(d_{c'}^{(n)}/\sigma)^2\big)}, \quad (3)$$

$$\vec{\mu}_c^{\,\text{new}} = \frac{\sum_n s_c^{(n)}\vec{y}^{\,(n)}}{\sum_n s_c^{(n)}}, \sigma^2_{\text{new}} = \frac{1}{DN}\sum_{n,c} s_c^{(n)}\|\vec{y}^{\,(n)} - \vec{\mu}_c^{\,\text{new}}\|^2. \quad (4)$$

Standard EM for the model (2) iterates E-step (3) and M-step (4), and each iteration changes the parameters $\Theta = (\vec{\mu}_{1:C}, \sigma^2)$ such that the likelihood is monotonously increased (until convergence to potentially local maxima). We will refer to this algorithm as standard GMM. In order to reduce the complexity of standard GMM, we here apply variational EM. Instead of seeking to maximize the likelihood directly, the basic idea of variational EM is to maximize a lower-bound of the likelihood, the free energy. The free energy depends on variational distributions which are chosen to (A) approximate exact posterior distributions $p(c\,|\,\vec{y}, \Theta)$ as closely as possible, and to (B) result in a less complex optimization objective. For our purposes, we use variational distributions $q^{(n)}(c; \mathcal{K}, \hat{\Theta})$ which depend on two types of variational parameters, sets of states $\mathcal{K} = (\mathcal{K}^{(1)}, \ldots, \mathcal{K}^{(N)})$ and $\hat{\Theta}$:

$$s_c^{(n)} \approx q^{(n)}(c; \mathcal{K}, \hat{\Theta}) = \frac{p(c, \vec{y}^{\,(n)}|\hat{\Theta})}{\sum\limits_{c'\in\mathcal{K}^{(n)}} p(c', \vec{y}^{\,(n)}|\hat{\Theta})}\,\delta(c \in \mathcal{K}^{(n)}), \quad (5)$$

where $\delta(c \in \mathcal{K}^{(n)}) = 1$ if $c \in \mathcal{K}^{(n)}$ and zero otherwise. The used variational distributions Eq. (5) are truncated posteriors, i.e., they are proportional to the exact posterior for all $c \in \mathcal{K}^{(n)}$ while they are exactly zero for all other $c \notin \mathcal{K}^{(n)}$. Fig. 1 (top: $s_c^{(m)}$ and $q_c^{(m)}$) illustrates the approximation. If we choose $\mathcal{K}^{(n)}$ for the GMM to contain all c, i.e. $\mathcal{K}^{(n)} = \{1, \ldots, C\}$, we recover standard GMM. If we choose $\mathcal{K}^{(n)}$ to contain just one single element, we can recover standard k-means [Lücke and Forster, 2017], without having to take the limit to zero variances σ^2. Given the variational distributions

Eq. (5), the corresponding free energy is:

$$\mathcal{F}(\mathcal{K},\hat{\Theta},\Theta) := \sum_n \Big(\sum_c q^{(n)}(c;\mathcal{K},\hat{\Theta}) \log\big(p(c,\vec{y}^{\,(n)}|\,\Theta)\big)\Big) + \sum_n H\big(q^{(n)}(c;\mathcal{K},\hat{\Theta})\big)\,, \tag{6}$$

where $H(p(c))$ is the entropy of a distribution $p(c)$. Maximization of the free energy Eq. (6) involves a three-stages optimization w.r.t. the parameters $\mathcal{K}$, $\hat{\Theta}$ and Θ. Because of the standard functional form of the free energy Eq. (6), the update equations for Θ (the M-step) remain identical to Eqs. (4) but use the approximation Eq. (5) instead of the exact $s_c^{(n)}$ in Eqs. (3). The complexity of one M-step is given by the number of non-zero $s_c^{(n)}$ values and by D, i.e., for $|\mathcal{K}^{(n)}| = C'$ one update of Θ has a complexity of $\mathcal{O}(NC'D)$.

For the variational E-step it was shown [Lücke, 2016] that it is sufficient to set $\hat{\Theta} = \Theta$ and to maximize a simplified expression of $\mathcal{F}(\mathcal{K},\Theta,\Theta) =: \mathcal{F}(\mathcal{K},\Theta)$:

$$\mathcal{K}^* = \operatorname*{argmax}_{\mathcal{K}} \big\{\mathcal{F}(\mathcal{K},\Theta)\big\} \tag{7}$$

Here, we now replace the maximization of the free energy by an increase of the free energy (partial E-step). For the GMM of Eq. (2) it can be shown that an increase of $\mathcal{F}(\mathcal{K},\Theta)$ has a direct geometrical interpretation (App. A provides the proof following Lücke and Forster [2017] and Lücke [2016]):

Proposition 1. If we replace for an arbitrary n a cluster $c \in \mathcal{K}^{(n)}$ by a cluster $\tilde{c} \notin \mathcal{K}^{(n)}$ such that $\|\vec{y}^{\,(n)} - \vec{\mu}_{\tilde{c}}\| < \|\vec{y}^{\,(n)} - \vec{\mu}_c\|$, then $\mathcal{F}(\mathcal{K},\Theta)$ increases.

3 Efficient Partial EM

Our algorithms iteratively increase the free energy using Prop. 1 and M-step Eqs. (4) until convergence. By relating partial EM to the geometric interpretation of cluster distances (Prop. 1), our approach maintains the guarantee to monotonously increase the free energy whenever clusters are found that are closer to $\vec{y}^{\,(n)}$ than those previously considered. Finding closer clusters can be realized much more efficiently than finding the closest clusters (which is of order $\mathcal{O}(NCD)$). To reduce run-time complexity, our goal is to as efficiently as possible find clusters c for each $\mathcal{K}^{(n)}$ such that the free energy is increased as effectively as possible. The free energy will always increase if we use Prop. 1 to update $\mathcal{K}^{(n)}$. However, blindly (e.g. uniform randomly) searching for closer clusters can be expected to be very inefficient. Instead, we here follow the strategy of defining a search space for each data point based on nearest cluster neighborhoods of each cluster in $\mathcal{K}^{(n)}$.

Although more efficient in terms of computational complexity, partial E-steps require us to memorize the variational parameters $\mathcal{K}^{(n)}$ across iterations. With one set $\mathcal{K}^{(n)}$ for each data point, we have with $|\mathcal{K}^{(n)}| = C'$ an additional memory requirement of $\mathcal{O}(NC')$ for all our variational algorithms.

3.1 Exhaustive Cluster Neighborhoods

To illustrate our approach, first consider Fig. 1. It visualizes a set $\mathcal{K}^{(n)}$ and its distances to the data point $\vec{y}^{\,(n)}$ and its closest neighboring clusters. We denote the set of clusters consisting of c and its $G-1$ nearest neighboring clusters by $\mathcal{G}_c$ and the union of sets $\mathcal{G}_c$ belonging to clusters $c \in \mathcal{K}^{(n)}$ as $\mathcal{G}^{(n)}$. To update $\mathcal{K}^{(n)}$ such that it only includes clusters with equal or smaller distances to $\vec{y}^{\,(n)}$ as before, we now won't evaluate all data-to-cluster distances. Instead, we only calculate the distances of $\vec{y}^{\,(n)}$ to the clusters in $\mathcal{G}^{(n)}$ and choose the C' closest clusters from these as new $\mathcal{K}^{(n)}$. Optimization of $\mathcal{K}^{(n)}$ then involves at most $C'G < C$ distance evaluations, and it can be considered much more likely than random search that clusters c which improve the free energy are found. However, we require additional computations to determine the nearest neighbors for each cluster. These computations are, however, independent of N.

Alg. 1 shows the complete algorithm, which consists of

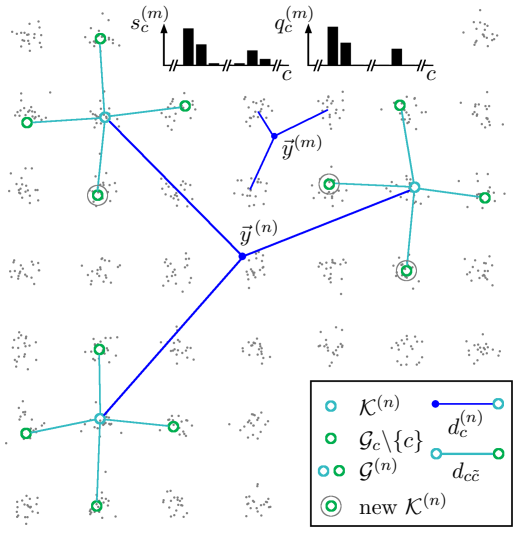

Figure 1: **Top:** Exact responsibilities $s_c^{(m)}$ and truncated responsibilities $q_c^{(m)}$ for a data point $\vec{y}^{(m)}$ with $\mathcal{K}^{(m)}$ containing the closest $C' = 3$ clusters. **Remainder:** Illustration of the search space $\mathcal{G}^{(n)}$ to find clusters increasingly close to $\vec{y}^{\,(n)}$. The search space consists of the clusters in $\mathcal{K}^{(n)}$ and the nearest neighbors of these clusters ($\mathcal{G}_c$ with $c \in \mathcal{K}^{(n)}$). For the illustration we used well separated clusters, $|\mathcal{K}^{(n)}| = C' = 3$ and $|\mathcal{G}_c| = G = 5$. Cluster centers $\vec{\mu}_c$ were assumed here to already represent the clusters well. See App. B for more information and a complete illustration across iterations.

Algorithm 1: GMM with partial truncated E-step and exhaustive cluster neighborhood evaluation.

init $\vec{\mu}_{1:C}$, σ and $\mathcal{K}^{(n)}$ for all n;
repeat
  for $c = 1 : C$ **do**
    for $\tilde{c} = 1 : C$ **do**
      $d_{c\tilde{c}} = \|\vec{\mu}_{\tilde{c}} - \vec{\mu}_c\|$;
    $\mathcal{G}_c = \{\tilde{c} \,|\, d_{c\tilde{c}}$ is among the G smallest distances $d_{c:}\}$;
  for $n = 1 : N$ **do**
    $\mathcal{G}^{(n)} = \bigcup_{c \in \mathcal{K}^{(n)}} \mathcal{G}_c$;
    for $c \in \mathcal{G}^{(n)}$ **do**
      $d_c^{(n)} = \|\vec{y}^{\,(n)} - \vec{\mu}_c\|$;
    $\mathcal{K}^{(n)} = \{c \,|\, d_c^{(n)}$ is among the C' smallest distances$\}$;
  for $n = 1 : N$ **do**
    for $c \in \mathcal{K}^{(n)}$ **do**
      $s_c^{(n)} = \frac{\exp\left(-\frac{1}{2}(d_c^{(n)}/\sigma)^2\right)}{\sum_{c' \in \mathcal{K}^{(n)}} \exp\left(-\frac{1}{2}(d_{c'}^{(n)}/\sigma)^2\right)}$;
  update $\vec{\mu}_{1:C}$ and σ^2 using Eqs. (4) with Eq. (5);
until $\vec{\mu}_{1:C}$ *and* σ^2 *have converged*;

three blocks of computation: (1st block) computation of cluster-to-cluster distances and definition of sets $\mathcal{G}_c$; (2nd block) computation of data-to-cluster distances and update of $\mathcal{K}^{(n)}$; and (3rd block) update of model parameters Θ. The complexities are given as follows:

The first block computes the nearest neighbor sets $\mathcal{G}_c$ (where we do not exploit symmetry of $d_{c\tilde{c}}$ for simplicity). The complexity of the required distance evaluations is $\mathcal{O}(C^2D)$. The following problem of finding the G smallest elements in an array of C elements is an unordered partial sorting problem. This is provably solvable with complexity $\mathcal{O}(C)$, e.g., by applying the median-of-medians algorithm [Blum et al., 1973] or introselect [Musser, 1997].

The second block computes for each data point the distances to all clusters in $\mathcal{G}^{(n)}$ consisting of all G neighbors $\mathcal{G}_c$ (including c) of all C' clusters $c \in \mathcal{K}^{(n)}$. We thus have to compute $\mathcal{O}(C'G)$ distances for each n, which amounts to $\mathcal{O}(NC'GD)$. If the clusters in $\mathcal{K}^{(n)}$ are close together, they might share some nearest neighbors, which reduces the actual number of distance computations. Also note that the $\mathcal{O}(C'G)$ can never exceed the total number of clusters C. To update one set of $\mathcal{K}^{(n)}$, we again have to solve an unordered partial sorting problem. For a given n, we have $C'G$ distances, such that finding the C' smallest elements is of complexity $\mathcal{O}(C'G)$, with $\mathcal{O}(NC'G)$ total evaluations. Hence, the evaluations of distances $d_c^{(n)}$ dominate the second block, which has an overall complexity of $\mathcal{O}(NC'GD)$.

Finally, the third block computes the responsibilities $s_c^{(n)}$ and updates the model parameters $\vec{\mu}_{1:C}$ and σ^2. Both have a complexity of $\mathcal{O}(NC'D)$.

The overall complexity of one EM iterations of Alg. 1 is consequently given by $\mathcal{O}(NC'GD + C^2D)$. Storage of sets $\mathcal{G}_c$ requires $\mathcal{O}(CG)$ additional memory, which results in a total of $\mathcal{O}(CD+NC'+CG)$ memory demand. We provide an illustration of the algorithm in App. B and a line-by-line complexity analysis in App. D.

3.2 Estimated Cluster Neighborhoods

For Alg. 1, we introduced new sets $\mathcal{G}_c$ that require additional evaluations of cluster-to-cluster distances. This can allow for a complexity reduction in the updates of our variational parameters $\mathcal{K}^{(n)}$, but adds a computational cost of $\mathcal{O}(C^2D)$. With Alg. 2 we show a new way to further reduce this algorithmic complexity.

In Alg. 2, the data-to-cluster distance computations and updates of the sets $\mathcal{K}^{(n)}$ remain exactly as in Alg. 1. However, to further improve efficiency, we now *estimate* the cluster-to-cluster distances $d_{c\tilde{c}}$ using the data-to-cluster distances $d_c^{(n)}$ which we anyway have to compute. To do this, the ordering of the first and second blocks of Alg. 1 now changes – Alg. 2 first computes the data-to-cluster distances $d_c^{(n)}$ before the sets $\mathcal{G}_c$ are updated – and we replace the exhaustive $\mathcal{G}_c$ evaluation of Alg. 1 by estimated cluster neighborhoods.

To illustrate the estimation approach, first consider two non-overlapping clusters c and $\tilde{c}$ with cluster centers $\vec{\mu}_c$ and $\vec{\mu}_{\tilde{c}}$ that already represent the two clusters well. If we denote by $\mathcal{I}_c$ the set of all points $\vec{y}^{\,(n)}$ that have c as closest cluster, $\mathcal{I}_c = \{n \,|\, c = \operatorname{argmin}_{\tilde{c}=1:C} \|\vec{y}^{\,(n)} - \vec{\mu}_{\tilde{c}}\|\}$, we may estimate:

$$d_{c\tilde{c}} \approx \frac{1}{|\mathcal{I}_c^*|} \sum_{\substack{n \in \mathcal{I}_c \\ \text{if } \tilde{c} \in \mathcal{G}^{(n)}}} \|\vec{y}^{\,(n)} - \vec{\mu}_{\tilde{c}}\| = \frac{1}{|\mathcal{I}_c^*|} \sum_{\substack{n \in \mathcal{I}_c \\ \text{if } \tilde{c} \in \mathcal{G}^{(n)}}} d_{\tilde{c}}^{(n)} \,. \quad (8)$$

If we sum over all $n \in \mathcal{I}_c$ (by ignoring the additional conditions for the sums and assuming $\mathcal{I}_c = \mathcal{I}_c^*$ for now), then Eq. (8) becomes exact for well separated clusters and close to optimal convergence points. However, in general not all the $d_{\tilde{c}}^{(n)}$ that we require for Eq. (8) have been computed before. We therefore introduce the condition $\tilde{c} \in \mathcal{G}^{(n)}$, which ensures that only data points n with known $d_{\tilde{c}}^{(n)}$ are summed over (see App. B for an example). $|\mathcal{I}_c^*|$ is then simply the number of summands. For already well-learned $\mathcal{K}^{(n)}$ and cluster centers $\vec{\mu}_c$, Eq. (8) estimates the $d_{c\tilde{c}}$ of close-by clusters well (and we are only interested in those close-by distances). The summation in Eq. (8) for distances to far-away clusters may not contain any data points and are ignored in practice. Away from convergence of Θ and $\mathcal{K}^{(n)}$, the estimation of $d_{c\tilde{c}}$ may be very coarse, but because of

Prop. 1 the updates of $\mathcal{K}^{(n)}$ still warrant a monotonous increase of the free energy.

Regarding the computational complexity of Alg. 2, the first and last block are identical to the second and third block of Alg. 1. Thus, we now only have to evaluate the complexity of the new middle part for the $\mathcal{G}_c$ estimation, replacing what was the first block of Alg. 1.

This new middle part of Alg. 2 consists of two steps: definition of auxiliary sets $\mathcal{I}_c$ and subsequent estimation of cluster neighborhoods $\mathcal{G}_c$. The definition of all sets $\mathcal{I}_c$ here requires a complexity of $\mathcal{O}(NC'G)$. In the next computational block, the sets $\mathcal{I}_c$ are used to estimate the distances $d_{c\tilde{c}}$ according to Eq. (8). There we loop over all clusters c and $\tilde{c}$ and then over all $n \in \mathcal{I}_c$ to estimate the distances. But there we only consider already computed $d^{(n)}_{\tilde{c}}$ (using the condition of Eq. 8). The loops over $c = 1 : C$ and $\tilde{c} = 1 : C$ of the third block may at first seem to suggest that their complexity depends on C. We can however rewrite the loops to show that this is actually not the case. If we formalize them as sums for computing functions $F^{(n)}_{c\tilde{c}}$, we obtain:

$$\sum_{c=1}^{C}\sum_{\tilde{c}=1}^{C}\sum_{n\in\mathcal{I}_c}\delta(\tilde{c}\in\mathcal{G}^{(n)})\,F^{(n)}_{c\tilde{c}} = \sum_{c=1}^{C}\sum_{n\in\mathcal{I}_c}\sum_{\tilde{c}\in\mathcal{G}^{(n)}}F^{(n)}_{c\tilde{c}}, \quad (9)$$

where $\delta(\tilde{c}\in\mathcal{G}^{(n)})$ is an indicator function representing the 'if'-condition. Further note that the sets $\mathcal{I}_c$ contain (N/C) data points on average and that $\sum_{c=1}^{C}\sum_{n\in\mathcal{I}_c}$ by construction of $\mathcal{I}_c$ then goes over exactly N values. As a set $\mathcal{G}^{(n)}$ contains at most $C'G$ elements, the total summation, i.e., the total cost of $d_{c\tilde{c}}$ estimations in Alg. 2, has a complexity of $\mathcal{O}(NC'G)$. For each c our procedure estimates on average $(N/C)C'G$ distances $d_{c\tilde{c}}$. For each $\mathcal{G}_c$, finding the G smallest elements (unordered partial sorting) then requires a computational cost of on average $\mathcal{O}((N/C)C'G)$, with a total cost for defining all sets $\mathcal{G}_c$ of $\mathcal{O}(NC'G)$. Storage of all $d^{(n)}_c$ distances within one iteration results (without additional measures) in an extra $\mathcal{O}(NC'G)$ memory demand.

Everything taken together, Alg. 2 has a run-time complexity of $\mathcal{O}(NC'GD)$ and a memory demand of $\mathcal{O}(CD + NC'G + CG)$ for the storage of all model parameters $\vec{\mu}_c$ and σ, variational parameters $\mathcal{K}^{(n)}$ and nearest neighbors $\mathcal{G}_c$, and computed distances $d^{(n)}_c$. A line-by-line complexity analysis of a more detailed version of Alg. 2 can be found in App. D.

Considering Alg. 2, e.g., the estimation of cluster-to-cluster distances may be deemed to cause problems. For the sake of complexity reduction, we accepted very coarse estimates. However, those estimates were far from random and can in principle and finally be good estimates for close-by clusters. Because of the way we defined the $\mathcal{K}^{(n)}$ updates in Alg. 2, a data-driven and finally relatively precise cluster-to-cluster distance estimation is all we require. The reason is that Prop. 1 warrants that the $\mathcal{K}^{(n)}$ updates will always monotonously increase the free energy, and the M-step with these updated $\mathcal{K}^{(n)}$ will in turn monotonously increase the objective. Such a provably monotonous increase ensures that learning proceeds in the right direction, and we know that truncated free-energies can result in very tight lower likelihood bounds (see also App. C). If Alg. 2 really does result in efficient optimization of the clustering objective, remains to be verified and investigated empirically in the next section.

Algorithm 2: GMM with partial truncated E-step and estimated cluster neighborhood evaluation.

init $\vec{\mu}_{1:C}$ and σ^2, and init $\mathcal{G}^{(n)}$ for all n;
repeat
  for $n = 1 : N$ **do**
    $\mathcal{G}^{(n)} = \bigcup_{c\in\mathcal{K}^{(n)}}\mathcal{G}_c$;
    for $c \in \mathcal{G}^{(n)}$ **do**
      $d^{(n)}_c = \|\vec{y}^{\,(n)} - \vec{\mu}_c\|$;
    $\mathcal{K}^{(n)} = \{c \,|\, d^{(n)}_c$ is among the smallest distances$\}$;
  for $n = 1 : N$ **do**
    $c^{(n)}_o = \underset{c\in\mathcal{G}^{(n)}}{\operatorname{argmin}}\{d^{(n)}_c\}$;
    $\mathcal{I}_{c^{(n)}_o} = \mathcal{I}_{c^{(n)}_o} \cup \{n\}$;
  for $c = 1 : C$ **do**
    for $\tilde{c} = 1 : C$ **do**
      for $n \in \mathcal{I}_c$ **do**
        if $\tilde{c} \in \mathcal{G}^{(n)}$ **then**
          $d_{c\tilde{c}} = d_{c\tilde{c}} + d^{(n)}_{\tilde{c}}$;
      $d_{c\tilde{c}} = d_{c\tilde{c}} \,/\, |\mathcal{I}^*_c|$;
    $d_{cc} = 0$;
    $\mathcal{G}_c = \{\tilde{c} \,|\, d_{c\tilde{c}}$ is among the G smallest distances $d_{c:}\}$;
  for $n = 1 : N$ **do**
    for $c \in \mathcal{K}^{(n)}$ **do**
      $s^{(n)}_c = \frac{\exp\left(-\frac{1}{2}(d^{(n)}_c/\sigma)^2\right)}{\sum_{c'\in\mathcal{K}^{(n)}}\exp\left(-\frac{1}{2}(d^{(n)}_{c'}/\sigma)^2\right)}$;
  update $\vec{\mu}_{1:C}$ and σ^2 using Eqs. (4) with Eq. (5);
until $\vec{\mu}_{1:C}$ *and* σ^2 *have converged*;

4 Numerical Experiments

Theoretical investigations of convergence are typically very intricate already for standard k-means [e.g., Har-Peled and Sadri, 2005] or standard GMMs [e.g., Xu et al., 2016]. We therefore verify sufficiently efficient optimization of the clustering objectives numerically.

Algs. 1 and 2 both allow for different choices of G and C', which optimized on a variety of data sets

would quickly result in a large combinatorics of different algorithms and data sets. As our primary aim here is to show that a clustering which scales sublinearly with C is possible, we focus on two algorithms: variational GMM and variational k-means (see below). We use a BIRCH artificial data set with 5×5 clusters to show the viability of the exhaustive and estimated algorithms (Algs. 1 and 2) compared to their non-variational counterparts. We then use the more efficient estimated algorithm (Alg. 2) on artificial BIRCH data sets of up to $C = 4096$ clusters to show that the gained computational benefits are not traded off by a higher number of necessary training iterations. Further experiments on KDD2004 ($C = 200$) and SONG ($C = 2000$) demonstrate large-scale applicability of the algorithms to natural data.

Variational GMM. For simplicity we choose $C' = G$ such that we remain with one parameter G that relates to cluster neighborhood relations. The resulting algorithm is similar to EM for GMM but optimizes a variational free energy instead of the likelihood directly. We therefore refer to Alg. 1 with $C' = G$ as variational-GMM-exhaustive (var-GMM-X), and to Alg. 2 with $C' = G$ as variational-GMM-estimated (var-GMM-S).

Variational k-means. It was recently shown [Lücke and Forster, 2017] that k-means is equivalent to truncated variational EM with a full E-step if we choose $|\mathcal{K}^{(n)}| = C' = 1$. If we maintain the choice of $C' = 1$ but replace the full E-step by a partial E-step with $G < C$ following Alg. 1 or Alg. 2, we obtain algorithms for which each iteration is more efficient in terms of required distance evaluations than k-means. Following the naming above, Alg. 1 with $C' = 1 < G < C$ will be referred to as var-k-means-X, and Alg. 2 with $C' = 1 < G < C$ we will refer to as var-k-means-S.

Complexities. The complexities of the four algorithms above directly result from the complexity considerations of Algs. 1 and 2. We here summarize the run-time and memory demand of the algorithms. The main approximation parameter is G, while keeping in mind that for var-GMM $C' = G$ and for var-k-means $C' = 1$. By inserting into the previously derived complexity formulas for Algs. 1 and 2, we obtain Tab. 1:

Table 1: Computational complexities, see also App. D

	Run-Time	Memory
var-GMM-X	$\mathcal{O}(NG^2D + C^2D)$	$\mathcal{O}(CD + NG + CG)$
var-GMM-S	$\mathcal{O}(NG^2D)$	$\mathcal{O}(CD + NG^2 + CG)$
var-k-means-X	$\mathcal{O}(NGD + C^2D)$	$\mathcal{O}(CD + N + CG)$
var-k-means-S	$\mathcal{O}(NGD)$	$\mathcal{O}(CD + NG + CG)$

Data sets. We apply the var-GMM and var-k-means algorithms to three data sets: (I) A 'BIRCH' data set [based on Zhang et al., 1997], which is an artificial $D = 2$-dimensional data set of C isotropic Gaussians, arranged in equal distances on a $\sqrt{C}\times\sqrt{C}$ grid. We investigate settings of $C = 25$ to 4096 Gaussians with $\sigma^2 = 1$ in nearest neighbor distances of $4\sqrt{2}$ and 100 samples per cluster, i.e., $N = 2500$ to 409 600 samples in total. (II) The KDD-Cup 2004 Protein Homology (KDD2004) data set, which was originally designed as a supervised classification set for the KDD competition 2004 but is also frequently used as a clustering benchmark. It consists of $N = 145\,751$ data points with $D = 74$ numerical features each. (III) The Year Prediction Million Song Dataset (SONG) [Bertin-Mahieux et al., 2011], which contains $N = 515\,345$ data points with $D = 90$ audio features (timbre averages and covariances). This represents the largest data set in our experiments.

4.1 Empirical Results

For Algs. 1 and 2, we monitor the standard quantization error during training: $\phi = \sum_n \min_{c\in C} \|\vec{y}^{\,(n)} - \vec{\mu}_c\|^2$. On the BIRCH and KDD data sets we allow for a maximum of 200 and on SONG of 500 training iterations for all algorithms. In all cases, we use AFK-MC^2 [Bachem et al., 2016a] for initialization of the means.

Validity. We first consider a small scale 5×5 BIRCH data set as described above to compare the partial variational algorithms with their respective non-variational counterparts. Considering Fig. 2, we note that initially var-GMM and var-k-means require more EM iterations than standard GMM or k-means to obtain comparable quantization errors. The primary reason for this is the random $\mathcal{K}^{(n)}$ and $\mathcal{G}^{(n)}$ initialization, which requires a couple of iterations until these reflect the true neighborhood of a data point. The number of additional EM iterations is however relatively small, and only more significant for very low values of G (small cluster neighborhoods). Differences between exhaustive cluster neighborhoods (var-GMM-X and var-k-means-X) and estimated neighborhoods (var-GMM-S and var-k-means-S) are only observable for very low G, which verifies that the estimation of cluster-to-cluster distances does not negatively affect performance at least not above such very low values. We will therefore from here on focus on the two most run-time efficient algorithms, var-GMM-S and var-k-means-S, which scale with $\mathcal{O}(NG^2D)$ and $\mathcal{O}(NGD)$, respectively.

Neighborhood initialization and exploratory neighbors. For large scale data with many clusters in low-dimensional spaces, the random initialization of cluster neighborhoods can be arbitrarily bad and can take many iterations until better neighborhoods are found. Performing M-steps during this initial period can have unwanted, destructive effects. For large-scale

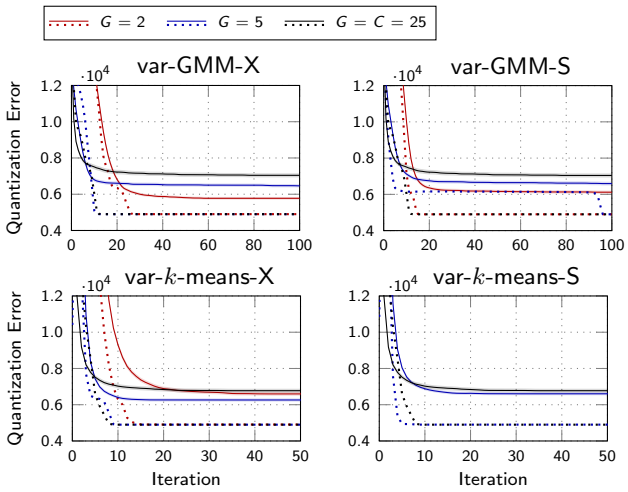

Figure 2: Results of Algs. 1 and 2 on the BIRCH 5×5 data set. The mean quantization error (solid), shaded with its SEM, as well as the single run with lowest final quantization error (dotted) over 100 independent training runs is given. Only in the extreme case of var-k-means-S with $C' = 1$ and $G = 2$ the distance estimation fails to recover viable results.

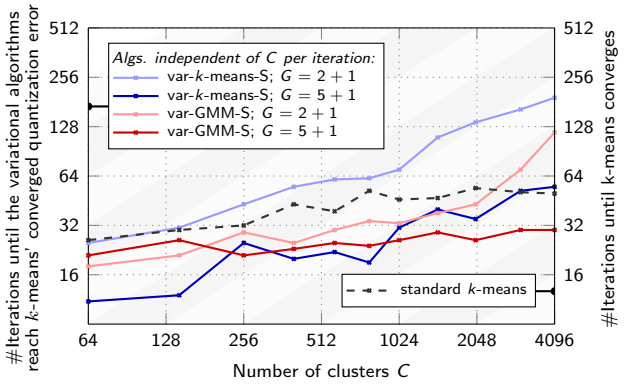

Figure 3: Total number of iterations (for initialization and training) until the variational algorithms or surpass the converged quantization error of standard k-means on BIRCH data sets of different sizes, given by the number of clusters C. The number of iterations for k-means until convergence is given as dashed line. A slope of 1 is indicated by the background for reference.

experiments we therefore allow for a couple of initial E-steps (optimized on the training quantization error) to find better initial neighborhood relations $\mathcal{K}^{(n)}$, $\mathcal{G}_c$. Furthermore, allowing for a single extra randomly picked exploratory cluster per data point within $\mathcal{G}^{(n)}$ helps to overcome large gaps between areas of dense clusters.

Absolute vs. relative complexity. Per iteration, the var-k-means and var-GMM algorithms save a significant amount of required distance evaluations. However, if this would result in an equally higher amount of required training iterations, the absolute gain of this relative reduction would vanish again. Already in Fig. 2 we saw that training times will not necessarily increase (equally) with $G < C$. We now further investigate this behavior on larger data sets by using systematically increasing sizes of BIRCH. As measure for training time we count the initial E-step iterations (see previous paragraph) and following EM iterations until the mean quantization error of var-k-means-S and var-GMM-S surpasses the mean converged quantization error of standard k-means. For both algorithms, we use $G = 2$ as well as $G = 5$, and for each G an additional randomly picked cluster in $\mathcal{G}^{(n)}$ (as described before). The compared means were taken over 5 independent training runs for each setting. As seen in Fig. 3, the reduction of distance evaluations per iteration in the variational algorithms does here not result in an equally increasing number of necessary iterations with respect to k-means. In most cases even less iterations were necessary than for k-means while the same quantization error was reached (all plots below the black dashed line). Importantly, the number of iterations increases less than linearly with C (which would in the log-log plot for small offsets result in linear graphs with slopes of 1, parallel to the lines of the background shading).

In all here investigated cases, the quantization error of var-k-means-S and var-GMM-S improved on standard k-means. Similar observations of better avoidance of local optima through truncated variational approaches were reported e.g. by Forster and Lücke [2017] for Poisson Mixtures and by Lücke and Forster [2017] for GMMs (however on a much smaller scale). The optimal degree of truncation is thereby primarily dependent on the geometric structure in the data. However, using smaller than optimal neighborhoods can still lead to viable results with potentially huge computational savings for large scale data.

Performance on large scale and natural data. The BIRCH data set is well suited to controllably increase the number of clusters within the data without changing their overall geometric structure. However, the low dimensionality, high regularity and good cluster separation within this data set does not necessarily reflect natural data very well. Real world practical implications are therefore demonstrated on two additional, large scale, high-dimensional, natural data sets: KDD and SONG. As for the BIRCH data sets we again also for SONG use initial E-steps and one single additional randomly selected cluster in $\mathcal{G}^{(n)}$ (denoted by the '+1' for G), and we now report mean quantization errors over 5 independent runs. Tab. 2 shows the trade-off between computational gain and quantization error for these two natural as well as high scale BIRCH data sets. Especially where the number of clusters is very large, var-k-means shows significant speedups of two to three orders of magnitude, reaching comparable

Table 2: Relative speedup in terms of saved distance evaluations per iteration with respect to the respective non-variational variants and relative quantization error with respect to standard k-means for both algorithms (as evaluation with full GMMs was computationally not feasible for these large data sets). For var-GMM-S both the theoretical minimal (first value) as well as the experimentally measured (second value) speedup is given, as in praxis overlaps of cluster neighborhoods $\mathcal{G}_c$ can reduce the number of distance evaluations significantly.

				var-k-means-S		var-GMM-S	
data set	D	C	G	speedup	q. error	speedup	q. error
KDD	74	200	2	100×	+101.8%	50×/55×	+4.2%
($N = 145\,751$)			5	40×	+5.8%	8×/15×	−0.0%
			20	10×	+0.3%	1×/3.3×	−0.0%
SONG	90	2000	$2+1$	667×	+1.0%	400×/412×	+1.8%
($N = 515\,345$)			$5+1$	333×	+0.5%	77×/105×	−0.2%
			$10+1$	182×	+0.2%	20×/40×	−0.3%
			$20+1$	95×	+0.1%	5×/15×	−0.3%
			$50+1$	39×	+0.0%		
BIRCH	2	2025	$2+1$	675×	−2.8%	405×/458×	−4.6%
$N = (202\,500)$			$5+1$	338×	−4.3%	78×/143×	−9.1%
BIRCH	2	4096	$2+1$	1365×	−3.7%	819×/927×	−4.4%
($N = 409\,600$)			$5+1$	683×	−4.0%	158×/287×	−11.7%

quantization errors on the natural data sets and even improving on k-means on the BIRCH data sets. On the other hand, var-GMM-S consistently improves on the k-means baseline for $G \geq 5$ on the natural data sets, and in all settings on BIRCH. Furthermore, while results on KDD for the full GMM were still obtainable with high computational effort (see App. C for results), evaluation of GMMs on SONG was only possible due to the significant computational savings of the var-GMM-S algorithm of up to two orders of magnitude.

5 Discussion

Using partial variational EM, we derived efficient clustering algorithms which require less than $\mathcal{O}(NCD)$ numerical operations per iteration. Most notably, algorithm var-GMM-S reduces the number of required distance evaluations per iteration from $\mathcal{O}(NC)$ to $\mathcal{O}(NG^2)$ where G is a small constant which relates to the cluster neighborhood relationship. Fig. 3 shows that the clustering objective is effectively and efficiently increased within the same number of iterations as k-means even if G is kept at a constant value while C is increased. The strong reduction of complexity per iteration with a sublinear scaling of required iterations with C (e.g. Fig. 3) is evidence for clustering being scalable sublinearly with C. For real data with many clusters, the complexity reduction amounted to a reduction of distance evaluations by two to almost three orders of magnitude (Tab. 2) compared to k-means, while comparable (or better) clustering results were obtained. And distance evaluations dominate the computational demand (see App. D). But in what sense can clustering feature such a sublinear scaling? In one sense, all clustering algorithms scale with C as implicitly we demand $N > C$, and only a value of N several times larger than C is sensible [but see Klami and Jitta, 2016]. A sublinear dependency on C may also be perceived as counter-intuitive because we want to update each of the C clusters in one iteration. Sublinear scaling with C, therefore, has to be interpreted as a sublinear scaling with NC. In this case, our reduction to $\mathcal{O}(NG^2)$ (or $\mathcal{O}(NG)$) distance evaluations per iteration still provides sufficient information (in the form of non-zero $s_c^{(n)}$) to update all clusters because NG or NG^2 is greater than C (typically several times). The sublinearity suggested by our empirical results also relies on specific properties of the data, especially certain neighborhood relationships among clusters. Data sets and initial conditions *can* be constructed that will result in a poor performance. These situations can typically be obtained using well separated areas with many close-by clusters, and by deliberately not assigning any cluster center to one area. With careful seeding (we used Bachem et al. [2016a]) and for typical data sets, such situations are unlikely, however. In practice, seeding (which may itself depend on C but not necessarily on NC, Bachem et al. [2016a]) also limits the number of EM iterations required for convergence although it remains very difficult to obtain theoretical results for convergence times. Finally, our results clearly show that memory requirement does linearly scale with C (but not with NC), and increases for algorithms with decreasing run-time complexity (Tab. 1). For memory reduction and more generally, combinations with coresets may represent promising future research. Coresets are complementary as they focus on the dependence on N, and as their weighted likelihood [compare, e.g., Lucic et al., 2017] can be treated by our approach very similarly to (1). Coresets are also available for general GMMs, and such generalizations likewise represents future research in the context of this work.

To summarize, we can (under the conditions discussed above) state that our empirical results suggest that a run-time scaling of clustering sublinear with its clusters is possible. In combination with recent advances in complementary lines of research, especially seeding [Bachem et al., 2016a], efficient distance evaluations [Elkan, 2003], or coresets [Har-Peled and Mazumdar, 2004, Feldman et al., 2011, Bachem et al., 2017], the significance of such a scaling being in principle and in practice possible may be very substantial.

Acknowledgments

We acknowledge funding by DFG grants LU 1196/5-1 (DF) and EXC 1077/1 (JL), and thank Florian Hirschberger for very helpful feedback.

References

D. Arthur, B. Manthey, and H. Röglin. k-Means has polynomial smoothed complexity. In *IEEE Foundations of Computer Science (FOCS)*, pages 405–414, 2009.

O. Bachem, M. Lucic, H. Hassani, and A. Krause. Fast and provably good seedings for k-means. In *Proceedings NIPS*, pages 55–63, 2016a.

O. Bachem, M. Lucic, S. H. Hassani, and A. Krause. Approximate k-means++ in sublinear time. In *Proceedings AAAI*, pages 1459–1467, 2016b.

O. Bachem, M. Lucic, and A. Krause. Scalable and distributed clustering via lightweight coresets. *arXiv preprint arXiv:1702.08248*, 2017.

P. Berkhin. A survey of clustering data mining techniques. In *Grouping multidimensional data*, pages 25–71. Springer, 2006.

T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere. The Million Song Dataset. In *Ismir*, volume 2, page 10, 2011.

M. Blum, R. W. Floyd, V. Pratt, R. L. Rivest, and R. E. Tarjan. Time bounds for selection. *Journal of computer and system sciences*, 7(4):448–461, 1973.

J. Y. K. Chan and A. P. Leung. Efficient k-means++ with random projection. In *Proceedings IJCNN*, 2017.

R. R. Curtin. A dual-tree algorithm for fast k-means clustering with large k. In *Proceedings SIAM International Conference on Data Mining*, pages 300–308, 2017.

Z. Dai and J. Lücke. Autonomous document cleaning. *Proceedings TPAMI*, 36(10):1950–1962, 2014.

Z. Dai, G. Exarchakis, and J. Lücke. What are the invariant occlusive components of image patches? A probabilistic generative approach. In *Proceedings NIPS*, pages 243–251, 2013.

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience (2nd Edition), Nov. 2001. ISBN 0471056693.

C. Elkan. Using the triangle inequality to accelerate k-means. In *Proceedings ICML*, volume 3, pages 147–153, 2003.

D. Feldman, M. Faulkner, and A. Krause. Scalable training of mixture models via coresets. In *Proceedings NIPS*, pages 2142–2150, 2011.

D. Forster and J. Lücke. Truncated variational EM for semi-supervised Neural Simpletrons. In *Proceedings IJCNN*, pages 3769–3776. IEEE, 2017.

S. Har-Peled and S. Mazumdar. On coresets for k-means and k-median clustering. In *ACM symposium on Theory of computing*, pages 291–300. ACM, 2004.

S. Har-Peled and B. Sadri. How fast is the k-means method? *Algorithmica*, 41(3):185–202, 2005.

M. Henniges, R. E. Turner, M. Sahani, J. Eggert, and J. Lücke. Efficient occlusive components analysis. *Journal of Machine Learning Research (JMLR)*, 15:2689–2722, 2014.

M. C. Hughes and E. B. Sudderth. Fast learning of clusters and topics via sparse posteriors. *arXiv preprint arXiv:1609.07521*, 2016.

M. Jordan, Z. Ghahramani, T. Jaakkola, and L. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37:183–233, 1999.

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

A. Klami and A. Jitta. Probabilistic size-constrained microclustering. In *Proceedings UAI*, 2016.

M. Lucic, M. Faulkner, A. Krause, and D. Feldman. Training mixture models at scale via coresets. *arXiv preprint arXiv:1703.08110*, 2017.

J. Lücke. Truncated variational Expectation Maximization. *arXiv preprint, arXiv:1610.03113*, 2016.

J. Lücke and D. Forster. k-means is a variational EM approximation of Gaussian mixture models. *arXiv preprint arXiv:1704.04812*, 2017.

G. McLachlan and D. Peel. *Finite mixture models*. John Wiley & Sons, 2004.

A. Moitra and G. Valiant. Settling the polynomial learnability of mixtures of Gaussians. In *IEEE Foundations of Computer Science (FOCS)*, pages 93–102, 2010.

A. W. Moore. Very fast EM-based mixture model clustering using multiresolution kd-trees. *Proceedings NIPS*, pages 543–549, 1999.

D. R. Musser. Introspective sorting and selection algorithms. *Software: Practice and Experience (SPE)*, 27(8):983–993, 1997.

R. Neal and G. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*. Kluwer, 1998.

S. J. Phillips. Acceleration of k-means and related clustering algorithms. In *Algorithm Engineering and Experimentation*, pages 166–177. Springer, 2002.

G. Puertas, J. Bornschein, and J. Lücke. The maximal causes of natural scenes are edge filters. In *Proceedings NIPS*, volume 23, pages 1939–1947. 2010.

D. Rosenbaum and Y. Weiss. The return of the gating network: combining generative models and discriminative training in natural image priors. In *Proceedings NIPS*, pages 2683–2691, 2015.

L. K. Saul, T. Jaakkola, and M. I. Jordan. Mean field theory for sigmoid belief networks. *Journal of Machine Learning Research (JMLR)*, 4(1):61–76, 1996.

A.-S. Sheikh and J. Lücke. Select-and-sample for spike-and-slab sparse coding. In *Proceedings NIPS*, volume 29, pages 3927–3935, 2016.

A.-S. Sheikh, J. A. Shelton, and J. Lücke. A truncated EM approach for spike-and-slab sparse coding. *Journal of Machine Learning Research (JMLR)*, 15:2653–2687, 2014.

J. A. Shelton, J. Gasthaus, Z. Dai, J. Lücke, and A. Gretton. GP-select: Accelerating EM using adaptive subspace preselection. *arXiv:1412.3411*, 2014. Now published by Neural Computation 29(8):2177–2202, 2017.

M. Shindler, A. Wong, and A. W. Meyerson. Fast and accurate k-means for large datasets. In *Proceedings NIPS*, pages 2375–2383, 2011.

J. Xu, D. J. Hsu, and A. Maleki. Global analysis of Expectation Maximization for mixtures of two Gaussians. In *Proceedings NIPS*, pages 2676–2684, 2016.

T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery*, 1(2):141–182, 1997.