A sampling-based approach for efficient clustering in large datasets

Georgios Exarchakis*
IHU Strasbourg, France
ICube, University of Strasbourg, CNRS, France
Sorbonne Université, INSERM, CNRS, France
Institut de la Vision, 17 rue Moreau, F-75012 Paris, France
`georgios.exarchakis@ihu-strasbourg.eu`

Omar Oubari
Sorbonne Université, INSERM, CNRS,
Institut de la Vision, 17 rue Moreau, F-75012 Paris, France

Gregor Lenz
Sorbonne Université, INSERM, CNRS,
Institut de la Vision, 17 rue Moreau, F-75012 Paris, France

Abstract

We propose a simple and efficient clustering method for high-dimensional data with a large number of clusters. Our algorithm achieves high-performance by evaluating distances of datapoints with a subset of the cluster centres. Our contribution is substantially more efficient than k-means as it does not require an all to all comparison of data points and clusters. We show that the optimal solutions of our approximation are the same as in the exact solution. However, our approach is considerably more efficient at extracting these clusters compared to the state-of-the-art. We compare our approximation with the exact k-means and alternative approximation approaches on a series of standardised clustering tasks. For the evaluation, we consider the algorithmic complexity, including number of operations to convergence, and the stability of the results. An efficient implementation of the algorithm is available online.

1. Introduction

Data clustering is an ubiquitous problem in Machine Learning literature. One of the most popular approaches for clustering is the k-means algorithm. Due to the simplicity of the algorithm and relative efficiency of identifying clusters the algorithm has found use in a wide range of fields, e.g. medicine, physics and computer vision among many others. With the increasingly large supply of data, computational

efficiency improvements become all the more significant. Recent advances in approximate inference methods have allowed for training algorithms with convergence performance that is sub-linear to the number of clusters [16]. This is typically achieved by avoiding the comparison between datapoints and clusters that lie far away in feature space.

Probabilistic data models that relate to the k-means algorithm can be found in Gaussian Mixture Models (GMM). In particular, Lücke and Forster [31] detail the relationship between k-means and a variational approximation of the Expectation Maximisation (EM) algorithm for isotropic GMMs. In most cases, the GMM-based formulation of the clustering problem provides higher likelihoods without introducing impractical constraints [31]. Theoretical developments in convergence analysis of Gaussian Mixture Models (GMM) [9, 32, 35] concerning global and optimal convergence have sparked renewed interest in the field. Novel training algorithms that aim for stability [22, 24] and increased efficiency [16, 21] are actively being developed. Markov Chain Monte Carlo (MCMC) methods have also been employed for fitting GMMs to account for input variation [13, 27, 36]. Tree-based methods have also been proposed for efficient inference in k-means [19, 34]. However, tree-based methods are known to exhibit instabilities when faced with small perturbations in the data, due to their recurrent structure, and therefore are likely to produce substantially different clusters for small changes in the input.

In this work, we propose a method for efficient EM-based learning that uses a truncated approximation [29, 30] of the posterior in the E-step. To identify the truncated space, we

---

*Corresponding Author.

draw samples from a proposal distribution that is based on the truncated subspace of the previous iteration and favours clusters near the optimal cluster of the previous truncated posterior. Our algorithm integrates recent developments in initialisation methods [1, 2] and can be applied on coresets [3, 20] to maintain comparable performance to state-of-the-art approaches.

Truncated approximations have been used in the past for multiple-cause models [8, 11, 12, 15] to achieve efficient training in discrete latent variable models. Stochastic approximations on a truncated space [18, 29] focusing on deep learning models have also been proposed. We expect a stochastic approach to avoid well-known local optima issues [23] related to EM-based learning for GMMs.

Truncated approximations on clustering algorithms have only been attempted with deterministic approximation techniques [16, 21]. In fact, our literature research shows that the vc-GMM [21] sets the state-of-the-art in terms of computational efficiency for a GMM with similar constraints to the ones we study. vc-GMM relies on a set of indices of datapoints that get assigned to the same cluster in order to identify similar clusters. The most similar cluster is identified at each step and immediately assigned to the truncated space of a datapoint to efficiently navigate to the optimal clustering solution. The approach, however, is deterministic and such methods typically exhibit results unstable to initialisation and frequently converge to local optima. It would be therefore prudent to explore stochastic analogues to these approximations. The approach we take in this work largely resembles vc-GMM, however, we utilise a similarity matrix over the clusters in order to identify clusters with a higher probability of being near a datapoint without having to evaluate its distance to all clusters. Estimating the similarity matrix relies on posterior approximations from earlier iterations and does not require excessive computation. We show that our stochastic approach improves over the performance of vc-GMM and we consider the definition and implementation of our algorithm to be significantly simpler.

In the numerical experiments section, we evaluate the performance of the algorithm and compare with relevant literature. In the artificial data section, we evaluate our algorithm in terms of extracting the ground truth and we compare it to k-means to observe an improved performance. In the real data clustering section, we apply the algorithm on four different datasets, namely, KDD, a protein homology dataset [7], CIFAR-10, an image dataset [25], SONG, a dataset of music metadata [5] and SUSY, a high energy physics dataset [4]. We compare our algorithm to the state-of-the-art in terms of efficiency, stability and accurate cluster recovery.

Results show that our algorithm sets the state-of-the-art in terms of efficiency without compromising, and probably improving, stability. Our method can be applied on a wide variety of tasks while maintaining a competitive clustering performance.

2. EM with sparsely sampled clusters for GMMs

To introduce the k-means algorithm in terms of an optimisation framework, we consider a Gaussian Mixture Model fitted with the Expectation Maximisation algorithm. The relationship between k-means and variational approximations to a GMM are detailed by Lücke and Forster [31]. Using the GMM-based formalisation, we can derive a novel clustering algorithm with same global optima as the original algorithm, and substantial computational efficiency benefits.

For a dataset of N data points, $\mathcal{Y} = \left\{ \mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(N)} \right\}$ we wish to identify, M, cluster centres $\mu_c, \forall c \in \{1, \ldots, M\}$. To that end, each datapoint $\mathbf{y}^{(n)}$ is treated as an instance of a random variable Y that follows one of M possible Gaussian distributions $p\left(Y | C = c; \theta\right) = \mathcal{N}\left(Y; \mu_c, \sigma \mathbb{1}\right)$ with a prior probability distribution $p\left(C\right) = \frac{1}{M}$, where C takes values in $\{1 \ldots, M\}$ and $\theta = \{\mu_{1:M}, \sigma\}$ denotes the set of model parameters. We can learn the optimal parameters, $\theta = \{\mu_{1,\ldots,M}, \sigma\}$, by maximising the data log-likelihood $\mathcal{L}\left(\theta\right) \triangleq \log p\left(Y = \mathcal{Y} | \theta\right)$ using the EM algorithm. The EM algorithm optimises the variational lower bound to the log-likelihood:

$$L\left(\mathcal{Y}, \theta\right) \triangleq \sum_n \sum_c p_c^{(n)} \log p\left(C = c, Y = \mathbf{y}^{(n)} | \theta\right)$$
$$+ \sum_n \mathcal{H}\left(p_c^{(n)}\right) \quad (1)$$
$$= \sum_n \sum_c p_c^{(n)} \log \frac{p\left(C = c | Y = \mathbf{y}^{(n)}, \theta\right)}{p_c^{(n)}}$$
$$+ \sum_n \log p\left(Y = \mathbf{y}^{(n)} | \theta\right) \quad (2)$$

where $\mathcal{H}\left(p_c^{(n)}\right)$ denotes the Shannon entropy of the distribution $p_c^{(n)}$. The distribution $p_c^{(n)}$ is typically set to be the posterior distribution $p\left(C = c | Y = \mathbf{y}^{(n)}; \hat{\theta}\right)$, as it sets the first term of Eq. 2 to 0 and the variational lower bound to be equal to the log-likelihood[1]. Under the assumed model constraints each Gaussian distribution is defined as $p\left(Y = \mathbf{y}^{(n)} | C = c; \theta\right) = \left(2\pi\sigma^2\right)^{-\frac{D}{2}} e^{-\frac{d_c^{(n)}}{2\sigma^2}}$, where $d_c^{(n)} = \left\| \mathbf{y}^{(n)} - \mu_c \right\|^2$ is the squared euclidean distance between the datapoint $\mathbf{y}^{(n)}$ and the mean of Gaussian indexed by c, and D is the number of observed variables.

Exact EM is an iterative algorithm that optimises the likelihood by alternating between two steps. The first step,

---

[1]The first term of Eq. 2 is the negative KL-divergence between the distribution $p_c^{(n)}$ and the exact posterior, $p\left(C = c | Y = \mathbf{y}^{(n)}, \theta\right)$.

E-step, is to identify the distribution $p_c^{(n)}$ that sets the lower bound in Eq. 2 to be equal to the log-likelihood. That is $p_c^{(n)}$ has to be equal to the posterior in order to set the KL-divergence in Eq. 2 to be equal to 0. For the Gaussian Mixture Model in this work that would be:

$$p_c^{(n)} = \exp\left(-d_c^{(n)}/2\sigma^2\right) / \sum_{c'=1}^{M} \exp\left(-d_{c'}^{(n)}/2\sigma^2\right) \quad (3)$$

Here we notice that Eq. 3 is a softmax function which produces mostly 0 values. In fact, for $\sigma^2 \to 0$ it is exactly equal to the maximum indicator function for the (negative) distances, i.e. it returns the value 1 for the smallest distance and 0 for all others, and is often considered as the probabilistic analogue of k-means [6, 26, 31]. The second step, M-step, amounts to maximising Eq. 1 with respect to θ using a gradient update as:

$$\mu_c = \frac{\sum_{n=1}^{N} p_c^{(n)} \mathbf{y}^{(n)}}{\sum_{n=1}^{N} p_c^{(n)}} \quad (4)$$

$$\sigma^2 = \frac{1}{DN} \sum_{n=1}^{N} \sum_{c=1}^{M} p_c^{(n)} \left\|\mathbf{y}^{(n)} - \mu_c\right\|^2 \quad (5)$$

The EM algorithm iterates between the E-step and M-step until θ converges. Updating σ^2, as opposed to the hard-assignment produced by $\sigma^2 \to 0$ in k-means, increases the variational lower bound [31] and offers a better and more efficient approximation of the log-likelihood.

The E-step requires estimating the differences between all clusters and all the datapoints. Thus, the complexity of the E-step is $\mathcal{O}(DNM)$ making it a very efficient algorithm. Here, we focus on a method to avoid estimating the softmax over all dimensions since it leads to redundant computation. In order to avoid the dependency of the complexity on M, we use an approximation $q_c^{(n)}$ of the posterior, $p_c^{(n)}$, over a subset $\mathcal{K}^{(n)} \subset \{1, \ldots, M\}$, with $|\mathcal{K}^{(n)}| = H$ as:

$$q_c^{(n)} = \frac{\exp\left(-d_c^{(n)}/2\sigma^2\right)}{\sum_{c' \in \mathcal{K}^{(n)}} \exp\left(-d_{c'}^{(n)}/2\sigma^2\right)} \delta\left(c \in \mathcal{K}^{(n)}\right) \quad (6)$$

where $\delta\left(c \in \mathcal{K}^{(n)}\right)$ is the Kronecker delta. In other words, we assume that clusters outside $\mathcal{K}^{(n)}$ have a probability of 0 for datapoint $\mathbf{y}^{(n)}$, and therefore are not estimated. Using $q_c^{(n)}$ instead of $p_c^{(n)}$, modifies the exact EM algorithm by not setting the KL-divergence to 0 at the E-step. However, we can derive an algorithm an algorithm that monotonically increases the variational lower bound by identifying a $q_c^{(n)}$ that decreases the KL-divergence at each E-step.

Proposition 1. *Let $\mathcal{K}^{(n)}$ be a set of cluster indices, and $\mathcal{K}'^{(n)} = \mathcal{K}^{(n)} \setminus \{i\} \cup \{j\}$, where $i \in \mathcal{K}^{(n)}$, $j \notin \mathcal{K}^{(n)}$. Then $KL[q^{(n)} \| p^{(n)}] < KL[q'^{(n)} \| p^{(n)}]$ if and only if $d_i^{(n)} < d_j^{(n)}$.*

Proof. Since all the Gaussians are equiprobable $d_i^{(n)} < d_j^{(n)} \Rightarrow p_i^{(n)} > p_j^{(n)}$. Note that $\lim_{x \to 0} x \log x = 0$. It follows that:

$$KL[q^{(n)} \| p^{(n)}] < KL[q'^{(n)} \| p^{(n)}] \Leftrightarrow$$

$$\sum_{c \in \mathcal{K}^{(n)}} q_c^{(n)} \log \frac{q_c^{(n)}}{p_c^{(n)}} < \sum_{c \in \mathcal{K}'^{(n)}} q'_c^{(n)} \log \frac{q'_c^{(n)}}{p_c^{(n)}} \Leftrightarrow$$

$$\sum_{c \in \mathcal{K}^{(n)}} q_c^{(n)} \log \frac{p_c^{(n)} / \sum_{c' \in \mathcal{K}^{(n)}} p_{c'}^{(n)}}{p_c^{(n)}} <$$

$$\sum_{c \in \mathcal{K}'^{(n)}} q'_c^{(n)} \log \frac{p_c^{(n)} / \sum_{c' \in \mathcal{K}'^{(n)}} p_{c'}^{(n)}}{p_c^{(n)}} \Leftrightarrow$$

$$\sum_{c \in \mathcal{K}^{(n)}} q_c^{(n)} \log \sum_{c' \in \mathcal{K}^{(n)}} p_{c'}^{(n)} >$$

$$\sum_{c \in \mathcal{K}'^{(n)}} q'_c^{(n)} \log \sum_{c' \in \mathcal{K}'^{(n)}} p_{c'}^{(n)} \Leftrightarrow$$

$$\log \sum_{c' \in \mathcal{K}^{(n)}} p_{c'}^{(n)} > \log \sum_{c' \in \mathcal{K}'^{(n)}} p_{c'}^{(n)} \Leftrightarrow p_i^{(n)} > p_j^{(n)}$$

$\square$

Proposition 1 shows that in order to decrease the KL-divergence at each E-step we only need to iteratively update the set $\mathcal{K}^{(n)}$ with clusters that are closer to the data-point, $\mathbf{y}^{(n)}$. The M-step can be modified to utilise $q_c^{(n)}$, instead of $p_c^{(n)}$, and maintain monotonic convergence [33].

To identify the clusters in $\mathcal{K}^{(n)}$, we start by selecting H clusters uniformly at random. We iteratively update $\mathcal{K}^{(n)}$ by using R randomly sampled clusters in the vicinity of the one that is nearest to the datapoint $\mathbf{y}^{(n)}$. To efficiently identify the clusters centred near a datapoint, we define a distribution $p(C_t|C_{t-1} = \bar{c}_n; S)$, where $\bar{c}_n = \arg\min_c \left\{d_c^{(n)} | c \in \mathcal{K}^{(n)}\right\}$. The parameter $S \in \mathbb{R}^{M \times M}$ denotes a similarity matrix among the clusters that assigns higher values, $S_{i,j}$, to cluster pairs, $\{i, j\}$, that are likely to be close to the same datapoints, as in Eq. 12. The iterative update of $\mathcal{K}^{(n)}$ is defined as:

$$\bar{\mathcal{K}}_t^{(n)} = \mathcal{K}_{t-1}^{(n)} \cup \left\{c_{1:R} | c_i \sim p(C_t|C_{t-1} = \bar{c}_n) \land c_i \notin \mathcal{K}_{t-1}^{(n)}\right\} \quad (7)$$

$$\mathcal{K}_t^{(n)} = \left\{c | c \in \bar{\mathcal{K}}_t^{(n)} \text{ with the } H \text{ smallest } d_c^{(n)}\right\} \quad (8)$$

where t denotes the EM iteration. $p(C_t|C_{t-1} = \bar{c}_n; S)$ is the distribution that is given by the normalised row of a cluster similarity matrix S after setting the probabilities corresponding to $\mathcal{K}^{(n)}$ to 0:

$$p(C_t = c|C_{t-1} = \bar{c}_n; S) = \frac{S_{\bar{c}_n, c}}{\sum_{c' \in \mathcal{K}^{(n)}} S_{\bar{c}_n, c'}} \delta\left(c \notin \mathcal{K}^{(n)}\right) \quad (9)$$

i.e. the distribution at time t is given by the row defined by the cluster $\bar{c}_n$ that had the minimal distance with the datapoint $\mathbf{y}^{(n)}$ at time $t - 1$.

The parameter updates, from Eq. 4 and 5, are adapted to the approximate posterior.

$$\mu_c = \sum_{n=1}^{N} q_c^{(n)} \mathbf{y}^{(n)} / \sum_{n=1}^{N} q_c^{(n)} \tag{10}$$

$$\sigma^2 = \frac{1}{DN} \sum_{n=1}^{N} \sum_{c \in \mathcal{K}^{(n)}} q_c^{(n)} \left\| \mathbf{y}^{(n)} - \mu_c \right\|^2 \tag{11}$$

The similarity matrix is defined based on the distances $d_c^{(n)}$ under the assumption that nearby clusters have small distances to similar datapoints

$$S_{i,j} = \frac{1}{N} \sum_{n=1}^{N} e^{-\left(d_i^{(n)} + d_j^{(n)}\right)} \delta\left(\{i,j\} \subset \mathcal{K}^{(n)}\right) \tag{12}$$

Algorithm 1 Data Similarity Gaussian Mixture Model (D-GMM)

Require: Dataset $\mathcal{X}$, # of centres M
1: initialise $\mu_{1:M}$, σ, $\mathcal{K}^{(n)}$ and $S = 0$ for all n;
2: **repeat**
3: $\quad \mu_{1:M}^{new} = 0$, $\sigma^{new} = 0$, and $S^{new} = 0$
4: $\quad \mathcal{J} = \{1, \ldots, N\}$
5: $\quad$ **for** $n \in \mathcal{J}$ **do**
6: $\quad\quad \bar{c}_n = \arg \min_c \left\{ \|\mathbf{y}^{(n)} - \mu_c\|^2 | c \in \mathcal{K}^{(n)} \right\}$
7: $\quad\quad p\left(C_t = c | C_{t-1} = \bar{c}_n; S\right) := \frac{S_{\bar{c}_n, c}}{\sum_{c'} S_{\bar{c}_n, c'}}$
8: $\quad\quad \bar{\mathcal{K}}^{(n)} = \left\{ c_{1:R} | c_i \sim p\left(C_t | C_{t-1} = \bar{c}_n; S\right) \wedge c_i \notin \mathcal{K}^{(n)} \right\}$

9: $\quad\quad \bar{\mathcal{K}}^{(n)} = \bar{\mathcal{K}}^{(n)} \cup \mathcal{K}^{(n)}$
10: $\quad\quad$ **for** $c \in \bar{\mathcal{K}}^{(n)}$ **do**
11: $\quad\quad\quad d_c^{(n)} = \left\| \mathbf{y}^{(n)} - \mu_c \right\|^2$
12: $\quad\quad$ **end for**
13: $\quad\quad \mathcal{K}^{(n)} = \left\{ c | c \in \bar{\mathcal{K}}^{(n)} \text{ with the } H \text{ smallest } d_c^{(n)} \right\}$
14: $\quad$ **end for**
15: $\quad$ Calculate $\mu_{1:M}$, σ^2, and S using Eqs. 10–12
16: **until** $\mu_{1:M}$ and σ^2 have converged

Eq. 12 produces a symmetric positive definite matrix, that is used to sample datapoints near the optimal at each step of the process, with a simple reduction operation over pre-computed values. Iterating between Eq. 6 and the parameter updates, Eqs. 10-12, details an algorithm that we call Data Similarity GMM (D-GMM), Alg. 1, due to the similarity matrix being based on a data "voting" process. The complexity of an E-step of the *D-GMM* algorithm reduces compared to an E-step of the exact EM algorithm for GMMs from $\mathcal{O}(NMD)$ to $\mathcal{O}(N(R+H)D)$, where

typically $R + H << M$. For the M-step, the complexity becomes $\mathcal{O}\left(NHD + NH^2\right)$ from $\mathcal{O}(NMD)$, however, as we will show in the experiments' section, $H^2 << M$ to be sufficient for most applications.

Initialisation. During the first epoch of the proposed algorithm the sets $\mathcal{K}^{(n)}$ are initialised using prior samples. The centres of the gaussians, $\mu_{1:C}$, are initialised using the AFK-MC^2 [1] initialisation method. After an epoch has passed, the $\mathcal{K}^{(n)}$ is updated as in algorithm 1. The AFK-MC^2 algorithm samples an initial centre $\mu_1 \in \mathcal{Y}$ uniformly at random and then uses it to derive the proposal distribution $g(\mathbf{y}|\mu_1)$. A markov chain of length m is used to sample sufficiently distinct new centres, $\mu_{2:M}$, iteratively. The complexity of AFK-MC^2 is $\mathcal{O}(ND)$ to define the proposal distribution $g(\mathbf{y}|\mu_1)$. The centres are sampled from the data using Markov chains of length m with a complexity of $\mathcal{O}\left(m(M-1)^2 D\right)$.

Lightweight Coresets (lwcs). To further improve computational efficiency we can optionally use coresets of the dataset [3, 14, 28]. Coresets are smaller, $N' << N$, representative subsets, $\mathcal{Y}' = \{(\mathbf{y}_1, w_1), \ldots, (\mathbf{y}_{N'}, w_{N'})\}$, of a full dataset, $\mathcal{Y}$, in which each datapoint is individually weighted by a weight $w_{1:N'}$ depending on its significance in describing the original data. The objective on a coreset is adjusted to account for the weights on each data point:

$$L\left(\mathcal{Y}', \theta\right) \triangleq \sum_n w_n \sum_{c \in \mathcal{K}^{(n)}} q_c^{(n)} \log p\left(C = c, Y = \mathbf{y}^{(n)} | \theta\right)$$
$$+ \sum_n w_n \mathcal{H}\left(q_c^{(n)}\right) \tag{13}$$

Since the parameter updates are gradient-based updates of Eq. 13, the weights $w_{1:N'}$ are a multiplicative constant on the parameters and therefore the parameter updates become:

$$\mu_c = \sum_{n=1}^{N'} w_n q_c^{(n)} \mathbf{y}^{(n)} / \sum_{n=1}^{N'} w_n q_c^{(n)} \tag{14}$$

$$\sigma^2 = \frac{1}{DN'} \sum_{n=1}^{N'} \sum_c w_n q_c^{(n)} \left\| \mathbf{y}^{(n)} - \mu_c \right\|^2 \tag{15}$$

$$S_{i,j} = \frac{1}{N'} \sum_{n=1}^{N'} w_n e^{-\left(d_i^{(n)} + d_j^{(n)}\right)} \delta\left(\{i,j\} \subset \mathcal{K}^{(n)}\right) \tag{16}$$

These updates can replace Eq. 10 to 12 in algorithm 1 to allow applications on a coreset $\mathcal{Y}'$. Working on coresets introduces an error in the approximation that has been analysed rigorously in earlier work [3]. Constructing the coreset requires two iterations of complexity $\mathcal{O}(ND)$ over the data. Working on coreset reduces the complexity of D-GMM to
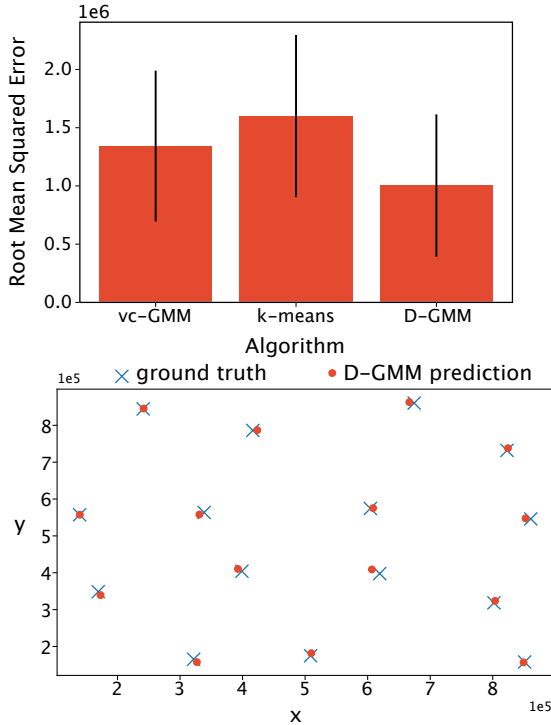
Figure 1. **Validation on a two-dimensional synthetic dataset:** D-GMM outperforms other methods across 500 trials (top) and fully recovers all centres (bottom)
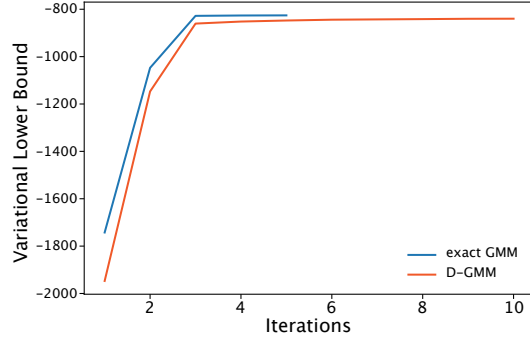

Figure 2. **Variational lower bound values of D-GMM through EM iterations compared to the log-likelihood of the exact algorithm:** We can see that our approximation is slower to converge, however, each iteration is considerably more efficient.

- N' denotes the coreset size
- H and C' denote the size of the truncated subspace for D-GMM and vc-GMM respectively.
- R and G are the search space hyperparameters for D-GMM and vc-GMM respectively.

When choosing the truncation hyperparameters H (C'), we consider that the probability values of the exact posterior decays exponentially and accordingly set $H = 5$ ($C' = 5$) under the assumption that lower probability values will be negligible. We follow the same rationale for the truncation updates R (G). We use various configurations for M and N' so we can compare with the state-of-the-art.

3.1. Artificial Data

In this section, we present a convergence analysis on artificial data [17] with $N = 5000$ data points and 15 Gaussian centres. Fig. 1 on the left shows the root mean squared error between the learned centres of the algorithms and the ground truth centres. We compare our algorithm, D-GMM, against vc-GMM, and standard k-means, setting the hyperparameters to $M = 15$, $N' = 1000$, $H = 3$ and $R = 5$. The vc-GMM is parametrised with $C' = 3$ and $G = 5$. The results suggest that both truncated algorithms are able to recover the centres as well as the exact algorithm. The slight improvement (below a standard deviation) might be attributed to the fact that a truncated approximation will "hard–code" very low probabilities to 0 which may enhance numerical stability. With the D-GMM algorithm, the stochastic behaviour might also have an effect on avoiding locally optimal solutions. In Fig. 1 on the right, we present an example of a run where the centres were successfully recovered.

3.2. Clustering Analysis

For a more detailed comparison with the state-of-the-art, we consider a series of well-known clustering datasets. Tab.

$\mathcal{O}\left(N'\left(R+H\right)D\right)$ for the E-step and $\mathcal{O}\left(N'HD + N'H^2\right)$ for the M-step. The complexity of AFK-MC^2 is also reduced since the proposal distribution is defined on the coreset with complexity $\mathcal{O}\left(N'D\right)$.

3. Experiments and results

We evaluate the performance of the algorithm experimentally on three classes of tasks. The software used for the experiments is a vectorised C++ implementation provided in the supplementary material. First, we examine convergence on artificial data where the ground truth is known. We proceed with a comparison against the state-of-the-art algorithm for training GMMs with similar constraints vc-GMM [21] on popular clustering datasets.

For all tasks, the Gaussian centres are initialised using the AFK-MC^2 algorithm with $m = 5$. Furthermore, we follow the same **convergence protocol** as in [21] and terminate the algorithm when the variational lower bound increment following Eq. 13 is less than $\epsilon = 10^{-3}$. Unless stated otherwise, we evaluate the stability of the results on 10 repetitions for all experiments.

For clarity, below is a reminder for the hyperparameter notations:

- M denotes the number of centres

Table 1. **Relative quantisation error and distance evaluation speedup**

Dataset	Algorithm				Relative Error η	Distance Eval. Speedup	Iters.
	name	N'	$H(C')$	$R(G)$			
KDD $N = 145,751$ $D = 74$ $M = 500$	k-means	-	-	-	$0.0 \pm 0.7\%$	$\times 1.0 \pm 0.0$	5.0 ± 0.0
	k-means + lwcs	2^{12}	-	-	$14.0 \pm 0.5\%$	$\times 35.1 \pm 0.0$	5.0 ± 0.0
	vc-GMM	2^{12}	5	5	$12.0 \pm 0.4\%$	$\times 533.1 \pm 36.5$	11.7 ± 1.0
	D-GMM	$\mathbf{2^{12}}$	**5**	**5**	$\mathbf{12.0 \pm 1.0\%}$	$\mathbf{\times 622.1 \pm 28.0}$	$\mathbf{17.0 \pm 0.7}$
CIFAR-10 $N_{\text{train}} = 50,000$ $N_{\text{test}} = 10,000$ $D = 3,072$ $M = 500$	k-means	-	-	-	$0.0 \pm 0.0\%$	$\times 1.0 \pm 0.0$	7.6 ± 0.4
	k-means + lwcs	2^{12}	-	-	$7.0 \pm 0.1\%$	$\times 48.9 \pm 3.5$	5.4 ± 0.4
	vc-GMM	2^{12}	5	5	$7.0 \pm 0.0\%$	$\times 674.7 \pm 45.8$	11.8 ± 0.8
	D-GMM	$\mathbf{2^{12}}$	**5**	**5**	$\mathbf{8.0 \pm 0.0\%}$	$\mathbf{\times 731.5 \pm 41.9}$	$\mathbf{21.4 \pm 1.2}$
SONG $N = 515,345$ $D = 90$ $M = 4000$	k-means	-	-	-	$0.0 \pm 0.0\%$	$\times 1.0 \pm 0.0$	5.0 ± 0.0
	k-means + lwcs	2^{16}	-	-	$8.0 \pm 0.0\%$	$\times 7.8 \pm 0.0$	5.0 ± 0.0
	vc-GMM	2^{16}	5	5	$8.0 \pm 0.1\%$	$\times 698.2 \pm 0.7$	12.0 ± 0.0
	D-GMM	$\mathbf{2^{16}}$	**5**	**5**	$\mathbf{8.0 \pm 0.2\%}$	$\mathbf{\times 862.1 \pm 18.3}$	$\mathbf{21.7 \pm 0.4}$
SUSY $N = 5,000,000$ $D = 18$ $M = 2000$	k-means	-	-	-	$0.0 \pm 0.0\%$	$\times 1.0 \pm 0.0$	14.7 ± 0.4
	k-means + lwcs	2^{16}	-	-	$6.0 \pm 0.1\%$	$\times 11.1 \pm 0.4$	14.1 ± 0.5
	vc-GMM	2^{16}	5	5	$6.0 \pm 0.1\%$	$\times 663.1 \pm 17.1$	25.4 ± 0.6
	D-GMM	$\mathbf{2^{16}}$	**5**	**5**	$\mathbf{5.0 \pm 0.1\%}$	$\mathbf{\times 605.7 \pm 11.1}$	$\mathbf{55.6 \pm 1.0}$

1 details a comparison between k-means, vc-GMM [16, 21], and D-GMM. We use the k-means algorithm on the full dataset to define a baseline for the centres. The accuracy of the rest of the algorithms is measured using the relative error $\eta = (\mathcal{Q}_{\text{algorithm}} - \mathcal{Q}_{\text{k-means}})/\mathcal{Q}_{\text{k-means}}$, where $\mathcal{Q}$ stands for an algorithm's quantization error. Since D-GMM and vc-GMM are going through fewer clusters per datapoint in each iteration, convergence is slower for these two algorithms (see Fig. 2). However, the efficiency of the algorithm is determined by the overall number of distance evaluations. In the last two columns of Tab. 1, we present the average number of iterations from initialisation to convergence as well as the average speedup in terms of distance evaluations, $d_c^{(n)}$, relative to k-means. The results show a clear speedup for D-GMM in most cases and comparable relative error.

Fig. 3 presents a comparison between the efficient clustering algorithms with increasing coreset size. K-means on the full dataset is presented as baseline. The size of each marker in Fig. 3 represents the size of the coreset. We find that in most cases D-GMM clusters data with a low relative error to the baseline for the least amount of distance evaluations.

Complexity The approximation method we use is focused on avoiding distance evaluations, $d_c^{(n)}$ with all available clusters. Therefore, it is very efficient in problems where a high number of clusters is expected to be present in the dataset. Fig. 4 (left) shows the scaling behaviour of our algorithm with an increasing number of clusters M on the CIFAR-10 dataset, with M ranging from 100 to 1500 cluster centres. The distance evaluations for each algorithm are normalised by the minimum value across all M and presented in a log-log plot which indicates the power of the relationship between operation complexity and number of clusters. We normalised both axes for an easier visualisation of the complexity. As expected, the scaling behaviour of k-means is linear to the number of clusters while the approximations are sub-linear. D-GMM is the most efficient algorithm in terms of distance evaluations as the number of cluster centres increases.

Stability We test the ability of the algorithm to recover the same clusters using different initialisation. We run the clustering algorithm on CIFAR-10 100 times with hyperparameters $M = 500, H = 5, R = 5$, and compare the recovered centres of every distinct pair of runs using the l_2-norm between the centres after reshuffling them. The average and standard deviation between all errors are plotted in Fig. 4 (right).

Hyperparameter Search In Fig. 5, we see the effect that the hyperparameters H and R have to the optimisation speedup. At the top plot, we fix $H = 5$ and view the effect R has to the algorithm's number of operations and runtime. We see that reducing the values of R progressively decreases the amount of required operations. In terms of runtime, values lower than $R = 40$ introduce a lower speedup. This is due to the fact that we need to perform more iterations and therefore spend more time instantiating samplers than
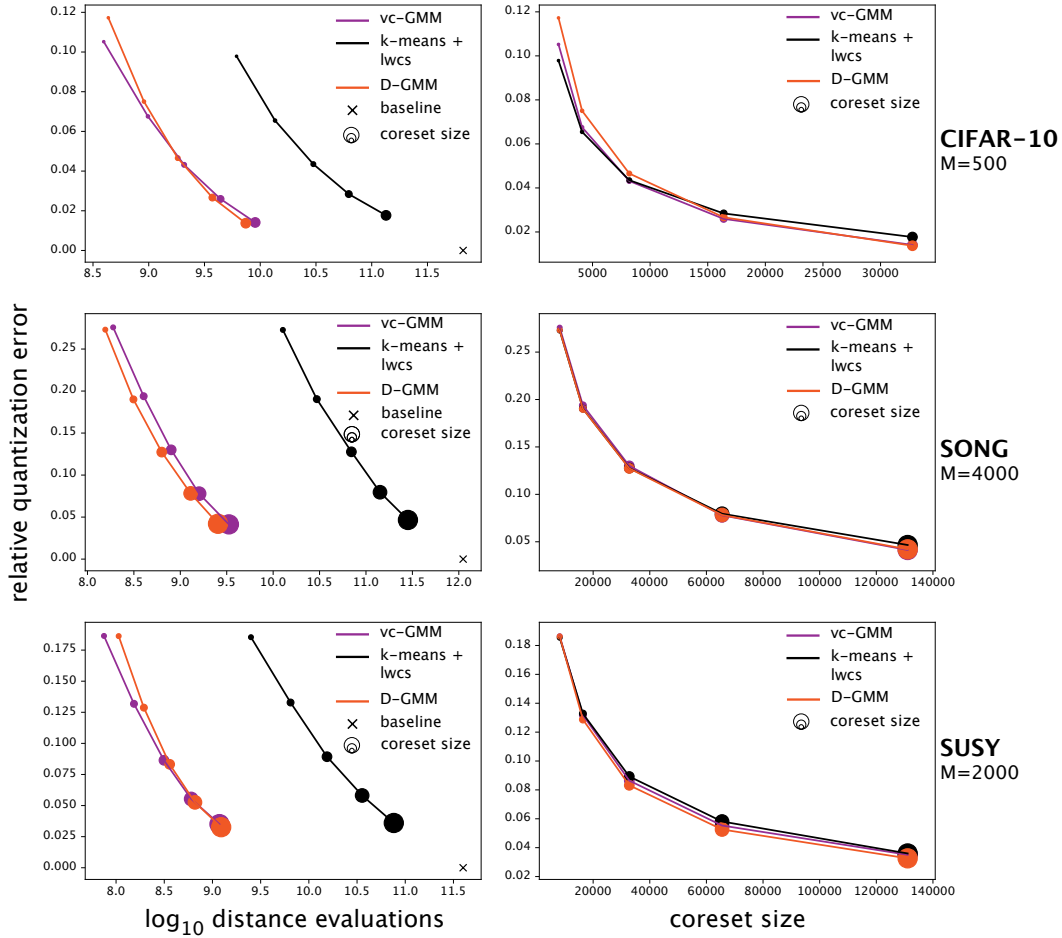
Figure 3. **Distance evaluations vs relative quantisation error on increasingly large datasets**
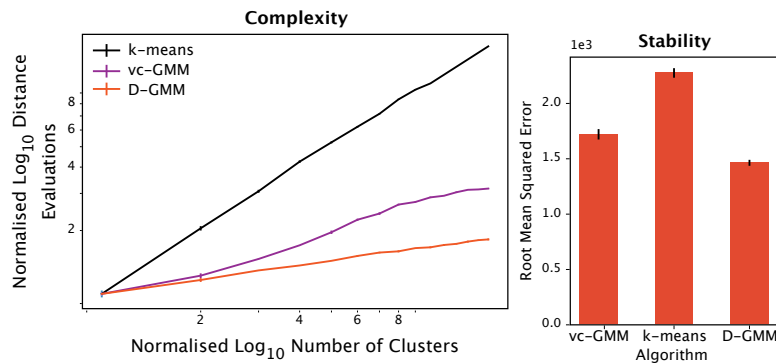

Figure 4. **Operations complexity and stability on CIFAR-10 over 100 trials. (right)** RMSE between the learned centres ($M = 500$). **(left)** Normalised log distance evaluations for an increasing M, with 100 trials for each M. **(both)** error bars denote 1 standard deviation.

drawing samples from them

To identify the optimal hyperparameter H, we fix $R = 10$ and observe the effect varying values of H have to the runtime and number of operations of the algorithm. Both runtime and number of operations monotonically reduce

with smaller truncated space H. Suggesting that caching only a small number of centres is sufficient to efficiently cluster datapoints.

To test the scalability of D-GMM to larger datasets, we use the full ImageNet dataset downsampled to a 64x64 res-
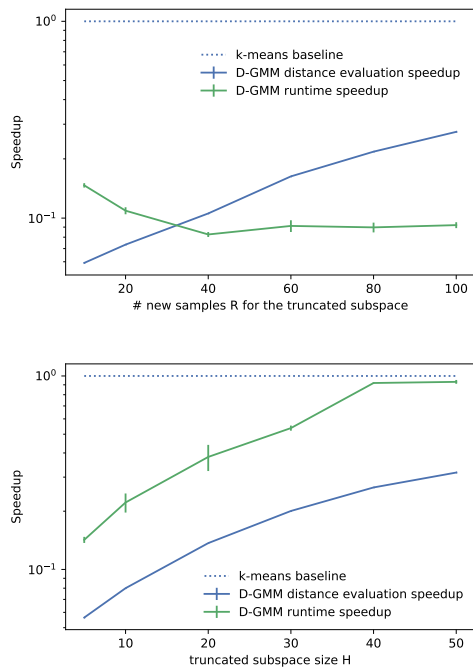
Figure 5. **Hyperparameter search for R (top) and H (bottom).** The blue line shows performance gains with respect to computation. The green line shows performance increase in terms of runtime. The results are relative to the dotted black line which is the exact k-means.

olution [10]. For such a large dataset it was impractical to use k-means as a baseline, as it would not converge in a reasonable time, so we resort to comparing the improvement in number of distance evaluations from initialisation to convergence of the two algorithms. For coreset sizes 2^{13}, 2^{14}, 2^{15}, 2^{16}, and 2^{17} we get a reduction on the number of distance evaluations of a factor of 22%, 31%, 40%, 44%, and 49% respectively when we use D-GMM compared to vc-GMM.

4. Discussion

We have presented a novel data clustering algorithm. Our algorithm considerably increases computational efficiency compared to k-means by calculating the posterior over a data-specific subset of clusters. The subset is iteratively refined by sampling in the neighbourhood of the best performing cluster at each EM iteration. To identify the neighbourhood of each cluster we propose a similarity matrix based on earlier computed distances between the clusters and datapoints, thus avoiding additional complexity. Furthermore, we implemented lightweight coresets and the AFK-MC^2 initialisation [1, 3] which are state-of-the-art methods in the literature for data pre-processing and GMM centre initialisation respectively. We compare our algorithm to vc-GMM [16,21] which

is, to our knowledge, the most efficient GMM algorithm currently available. In terms of computational complexity, our algorithm is more efficient in most cases, improving both with an increasing number of datapoints and with an increasing number of clusters compared to vc-GMM. Furthermore, the advantage in efficiency is complemented by a more stable recovery of clusters centres, as demonstrated on the CIFAR-10 database.

It is significant to emphasise that D-GMM is substantially simpler to intuit and implement compared to vc-GMM. Arguably, the use of elementary operations on matrix containers is easier to implement than task specific containers for comparison. Simplicity of an algorithm is a considerable advantage when it comes to communicating and implementing the algorithm in different contexts.

Our experiments suggest that the bottleneck for D-GMM lies with the efficiency of low-level operators like calculating exponentials and sampling from discrete distributions. Improving these operators could be an interesting future direction in this work, affecting an even larger body of literature. A key feature of D-GMM that we consider valuable for further development is that reduced computational complexity implies lower energetic demands. Therefore, when setting future software development strategies considering considering low-level operators used in clustering algorithms we can take into account the reduced number of operations of D-GMM. The constraints we introduce on the GMM data model are crucial for the D-GMM approximation, however, they have an impact on the expressive potential of our algorithm. Future work aims at developing the optimisation algorithm in a way that would allow training GMMs with fewer constraints on covariance and prior structure.

To enable further development of this work, we participate in the "variational sublinear clustering" organisation with an international team of independent researchers to jointly develop software for efficient probabilistic clustering. The D-GMM implementation, found in the supplementary material, will be contributed to this organisation, under an open source license, and further developed through a collaboration with a larger pool of researchers.

In conclusion, we find that D-GMM sets the state-of-the-art in terms of efficiency and stability for GMM-based clustering. There is room for improvement in terms of optimisation of low-level operators and loosening the GMM constraints. A long-term plan to develop and popularise efficient clustering is under way.

Acknowledgement

References

[1] Olivier Bachem, Mario Lucic, Hamed Hassani, and Andreas Krause. Fast and provably good seedings for k-means. In *Advances in neural information processing systems*, pages 55–63, 2016. 2, 4, 8

[2] Olivier Bachem, Mario Lucic, S Hamed Hassani, and Andreas Krause. Approximate k-means++ in sublinear time. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016. 2

[3] Olivier Bachem, Mario Lucic, and Andreas Krause. Scalable k-means clustering via lightweight coresets. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1119–1127, 2018. 2, 4, 8

[4] Pierre Baldi, Peter Sadowski, and Daniel Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nature communications*, 5:4308, 2014. 2

[5] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011. 2

[6] Tamara Broderick, Brian Kulis, and Michael Jordan. Mad-bayes: Map-based asymptotic derivations from bayes. In *International Conference on Machine Learning*, pages 226–234. PMLR, 2013. 3

[7] Rich Caruana, Thorsten Joachims, and Lars Backstrom. Kdd-cup 2004: results and analysis. *ACM SIGKDD Explorations Newsletter*, 6(2):95–108, 2004. 2

[8] Zhenwen Dai, Georgios Exarchakis, and Jörg Lücke. What are the invariant occlusive components of image patches? a probabilistic generative approach. In *Advances in neural information processing systems*, pages 243–251, 2013. 2

[9] Constantinos Daskalakis and Gautam Kamath. Faster and sample near-optimal algorithms for proper learning mixtures of gaussians. In *Conference on Learning Theory*, pages 1183–1213, 2014. 1

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 8

[11] Georgios Exarchakis, Marc Henniges, Julian Eggert, and Jörg Lücke. Ternary sparse coding. In *International Conference on Latent Variable Analysis and Signal Separation*, pages 204–212. Springer, 2012. 2

[12] Georgios Exarchakis and Jörg Lücke. Discrete sparse coding. *Neural computation*, 29(11):2979–3013, 2017. 2

[13] Stefano Favaro, Yee Whye Teh, et al. Mcmc for normalized random measure mixture models. *Statistical Science*, 28(3):335–359, 2013. 1

[14] Dan Feldman, Matthew Faulkner, and Andreas Krause. Scalable training of mixture models via coresets. In *Advances in neural information processing systems*, pages 2142–2150, 2011. 4

[15] Dennis Forster and Jorg Lücke. Truncated variational em for semi-supervised neural simpletrons. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 3769–3776. IEEE, 2017. 2

[16] Dennis Forster and Jörg Lücke. Can clustering scale sublinearly with its clusters? a variational em acceleration of gmms and k-means. In *International Conference on Artificial Intelligence and Statistics*, pages 124–132, 2018. 1, 2, 6, 8

[17] P. Fränti and O. Virmajoki. Iterative shrinking method for clustering problems. *Pattern Recognition*, 39(5):761–765, 2006. 5

[18] Enrico Guiraud, Jakob Drefs, and Jörg Lücke. Evolutionary expectation maximization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 442–449, 2018. 2

[19] Greg Hamerly. Making k-means even faster. In *Proceedings of the 2010 SIAM international conference on data mining*, pages 130–140. SIAM, 2010. 1

[20] Sariel Har-Peled and Soham Mazumdar. On coresets for k-means and k-median clustering. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 291–300, 2004. 2

[21] Florian Hirschberger, Dennis Forster, and Jörg Lücke. Accelerated training of large-scale gaussian mixtures by a merger of sublinear approaches. *IEEE Transaction on Pattern Analysis and Machine Intelligence.*, 1050:12, 2021. 1, 2, 5, 6, 8

[22] Reshad Hosseini and Suvrit Sra. An alternative to em for gaussian mixture models: Batch and stochastic riemannian optimization. *Mathematical Programming*, pages 1–37, 2019. 1

[23] Chi Jin, Yuchen Zhang, Sivaraman Balakrishnan, Martin J Wainwright, and Michael I Jordan. Local maxima

in the likelihood of gaussian mixture models: Structural results and algorithmic consequences. In *Advances in neural information processing systems*, pages 4116–4124, 2016. 2

[24] Soheil Kolouri, Gustavo K. Rohde, and Heiko Hoffmann. Sliced wasserstein distance for learning gaussian mixture models. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 1

[25] Alex Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 05 2012. 2

[26] Brian Kulis and Michael I Jordan. Revisiting k-means: New algorithms via bayesian nonparametrics. *arXiv preprint arXiv:1111.0352*, 2011. 3

[27] Y. Li, S. Pal, and M. J. Coates. Invertible particle-flow-based sequential mcmc with extension to gaussian mixture noise models. *IEEE Transactions on Signal Processing*, 67(9):2499–2512, 2019. 1

[28] Mario Lucic, Matthew Faulkner, Andreas Krause, and Dan Feldman. Training gaussian mixture models at scale via coresets. *The Journal of Machine Learning Research*, 18(1):5885–5909, 2017. 4

[29] Jörg Lücke, Zhenwen Dai, and Georgios Exarchakis. Truncated variational sampling for 'black box'optimization of generative models. In *International Conference on Latent Variable Analysis and Signal Separation*, pages 467–478. Springer, 2018. 1, 2

[30] Jörg Lücke and Julian Eggert. Expectation truncation and the benefits of preselection in training generative models. *Journal of Machine Learning Research*, 11(Oct):2855–2900, 2010. 1

[31] Jörg Lücke and Dennis Forster. k-means as a variational em approximation of gaussian mixture models. *Pattern Recognition Letters*, 125:349–356, 2019. 1, 2, 3

[32] Ankur Moitra and Gregory Valiant. Settling the polynomial learnability of mixtures of gaussians. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 93–102. IEEE, 2010. 1

[33] Radford M Neal and Geoffrey E Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. Springer, 1998. 3

[34] James Newling and François Fleuret. Fast k-means with accurate bounds. In *International Conference on Machine Learning*, pages 936–944. PMLR, 2016. 1

[35] Ji Xu, Daniel J Hsu, and Arian Maleki. Global analysis of expectation maximization for mixtures of two gaussians. In *Advances in Neural Information Processing Systems*, pages 2676–2684, 2016. 1

[36] Ming-Hsuan Yang and Narendra Ahuja. Gaussian mixture model for human skin color and its applications in image and video databases. In *Storage and retrieval for image and video databases VII*, volume 3656, pages 458–466. International Society for Optics and Photonics, 1998. 1