arXiv:1703.08110v1 [stat.ML] 23 Mar 2017

Training Mixture Models at Scale via Coresets

Mario Lucic LUCIC@INF.ETHZ.CH
Department of Computer Science
ETH Zurich
Universitätstrasse 6, 8092 Zürich, Switzerland

Matthew Faulkner MNFAULK@GMAIL.COM
Department of Electrical Engineering and Computer Sciences
Caltech
1200 E California Blvd, Pasadena, California 91125

Andreas Krause KRAUSEA@ETHZ.CH
Department of Computer Science
ETH Zurich
Universitätstrasse 6, 8092 Zürich, Switzerland

Dan Feldman DANNYF.POST@GMAIL.COM
Department of Computer Science
University of Haifa
199 Aba Khoushy Ave. Mount Carmel, Haifa, Israel

Abstract

How can we train a statistical mixture model on a massive data set? In this paper, we show how to construct *coresets* for mixtures of Gaussians and natural generalizations. A coreset is a weighted subset of the data, which guarantees that models fitting the coreset also provide a good fit for the original data set. We show that, perhaps surprisingly, Gaussian mixtures admit coresets of size polynomial in dimension and the number of mixture components, while being *independent* of the data set size. Hence, one can compute a $(1+\varepsilon)$-approximation for the optimal model on a significantly smaller data set. More importantly, such coresets can be efficiently constructed both in distributed and streaming settings. Our results rely on a novel reduction of statistical estimation to problems in computational geometry and new complexity results for mixtures of Gaussians. As a by-product of our analysis, we prove that the pseudo-dimension of arbitrary mixtures of Gaussians is polynomial in the ambient dimension. Empirical evaluation on several real-world datasets suggest that our coreset-based approach enables significant reduction in training-time with negligible approximation error.

Keywords: Gaussian mixture models, coresets, streaming, distributed, pseudo-dimension

1. Introduction

We consider the problem of training statistical mixture models, in particular mixtures of Gaussians and some natural generalizations, on massive data sets. In contrast to parameter estimation for models with compact sufficient statistics, mixture models generally require inference over latent variables, which in turn depends on the full data set. Such data sets are often distributed across a cluster of machines, or arrive in a data stream, and have to

be processed with limited memory. In this paper, we show that Gaussian mixture models (GMMs), and some generalizations, admit small *coresets*: A weighted subset of the data which guarantees that models fitting the coreset will also provide a good fit for the original data set. Perhaps surprisingly, we show that Gaussian mixtures admit coresets of size *independent* of the size of the data set.

Hence, solving the estimation problem on the coreset $\mathcal{C}$ (e.g., using weighted variants of the EM algorithm, see Section 5) is as good as solving the estimation problem on the large data set $\mathcal{X}$. Our algorithm for constructing $\mathcal{C}$ is based on importance sampling and is simple to implement. We focus on training mixtures of λ-semi-spherical Gaussians, where the covariance matrix Σ_i of each component $1 \leq i \leq k$ has eigenvalues bounded in $[\lambda, 1/\lambda]$, but some of our results generalize even to the semi-definite case. In particular, we show that given a data set $\mathcal{X}$ of n points in $\mathbb{R}^d$, $\varepsilon > 0$ and $k \in \mathbb{N}$, one can efficiently construct a weighted set $\mathcal{C}$ of $\Theta\big(d^4 k^6 \varepsilon^{-2} \lambda^{-8}\big)$ points, such that for any mixture of k λ-semi-spherical Gaussians $\theta = [(w_1, \mu_1, \Sigma_1), \ldots, (w_k, \mu_k, \Sigma_k)]$ it holds that the log-likelihood $\ln P(\mathcal{X} \mid \theta)$ of $\mathcal{X}$ under θ is approximated by the (properly weighted) log-likelihood $\ln P(\mathcal{C} \mid \theta)$ of $\mathcal{C}$ under θ to arbitrary accuracy as $\varepsilon \to 0$. Moreover, the coresets can be efficiently constructed in parallel (using a map-reduce style computation), as well as in the streaming setting using space and update time per point polynomial in d, k, λ^{-1}, ε^{-1}, $\log n$ and $\log(1/\delta)$.

2. Related Work

In this section we review the results on learning mixtures of Gaussians and the relevant coreset literature. We conclude this section by contrasting prior art to this work.

2.1 Learning Mixtures of Gaussians

The fundamental problem of learning Gaussian mixture models has received a great deal of interest. The most commonly used technique is maximum likelihood estimation whereby the parameters of the model are computed via expectation-maximization (Dempster et al., 1977). As the objective is non-convex, the algorithm is guaranteed to converge only to a local optima. The work of Dasgupta (1999) was the first to show that parameters θ of an unknown GMM can be estimated in polynomial time, with arbitrary accuracy ε, given i.i.d. samples from θ. However, the proposed algorithm assumes a common covariance, bounded eccentricity, a (known) bound on the smallest component weight, as well as a separation (distance of the means) that scales with $\Omega(\sqrt{d})$. Subsequent works relax the assumption on separation to $d^{1/4}$ (Dasgupta and Schulman, 2000) and $k^{1/4}$ (Vempala and Wang, 2004). Feldman et al. (2006b) provide the first result that does not require any separation, but assumes that the Gaussians are axis-aligned. Moitra and Valiant (2010) and Belkin and Sinha (2010) prove that arbitrary GMMs with fixed number of components can be learned in polynomial time and sample complexity (but exponential dependence on k). Anandkumar et al. (2012, 2014) demonstrate that a spectral decomposition technique yields consistent parameter estimates from low-order observable moments, without additional separation assumptions. These results hinge on non-degeneracy, that is, that the component means indeed span a k-dimensional subspace and the vector w has strictly positive entries. We stress that all of the above results crucially rely on the fact that the data set $\mathcal{X}$ is actually generated by a mixture of Gaussians. The problem of fitting a mixture model with

near-optimal log-likelihood for arbitrary data is studied by Arora and Kannan (2005). They provide a polynomial-time approximation scheme, provided that the Gaussians are identical spheres. We note that in this case the maximum likelihood estimation problem reduces to k-means clustering.

In contrast, as detailed in Section 4, our results make only mild assumptions about the Gaussian components and allow one to explicitly trade-off the coreset size and the assumption strength. Critically, none of the algorithms described above applies to the streaming or parallel setting.

2.2 Approximation Algorithms via Coresets

Approximation algorithms in computational geometry often make use of random sampling, feature extraction, and ε-samples (Haussler, 1992). Coresets can be viewed as a general concept that includes all of the above, and more.

Existence and construction of coresets have been investigated for a number of problems in computational geometry (Agarwal et al., 2005; Czumaj and Sohler, 2007) and have been used to great effect for a host of geometric and graph problems, including k-median (Har-Peled and Mazumdar, 2004), k-means (Feldman et al., 2007), k-center (Har-Peled and Varadarajan, 2004), k-line median (Feldman et al., 2006a), subspace approximation (Feldman et al., 2006a; Mahoney and Drineas, 2009), (k, m)-segment mean (Feldman et al., 2012), PCA and projective clustering (Feldman et al., 2013b), distributed k-means and k-median (Balcan et al., 2013), dictionary learning (Feldman et al., 2013a), k-segmentation of streaming data (Rosman et al., 2014), non-parametric estimation (Bachem et al., 2015), clustering with Bregman divergences (Lucic et al., 2016b), etc. A framework that generalizes and improves several of these results has appeared in Feldman and Langberg (2011). Notably, coresets also imply streaming algorithms for many of these problems (Har-Peled and Mazumdar, 2004; Agarwal et al., 2005; Frahling and Sohler, 2005; Feldman et al., 2007). Recently, coresets were leveraged to establish a space-time-data-risk tradeoff in the context of unsupervised learning (Lucic et al., 2015). Promising results in the context of empirical risk minimization have been demonstrated by Reddi et al. (2015). For a survey of the recent results we refer the reader to Bachem et al. (2017b).

Apart from fast approximation, streaming and parallel computations, coresets have many other applications. For example, since our coreset approximates *any* mixture of λ-semi-spherical Gaussians, it can be used to solve optimization problems with constraints on the Gaussian components (such as forbidden areas for their centers).

2.3 Our Contributions

In this work we demonstrate how these techniques from computational geometry can be lifted to the realm of statistical estimation. As a by-product of our analysis, we also answer an open question on the pseudo-dimension (generalization of the VC dimension) of arbitrary mixtures of Gaussians. This paper is an extended version of Feldman et al. (2011) and provides several improvements over the original work.

The theoretical analysis is now simplified and executed directly on the negative log-likelihood implied by a fixed mixture of Gaussians. Based on this simplification, we provide a blueprint for constructing coresets for other statistical models. We present a new proof for

the complexity (pseudo-dimension) of a mixture of Gaussians and natural generalizations by forging a link to VC analysis of neural networks. We now directly model the impact of the assumption of λ-semi-spherical Gaussian components on the coreset size. Critically, we prove that one can use *any* bicriteria approximation for the k-means clustering problem as a basis for the importance sampling scheme. Specifically, we show that the well-known `k-means++` algorithm is a good choice, both in theory and practice. To this end, we decoupled the bicriteria approximation algorithm from the sampling scheme. Overall, we are able to construct larger coresets in less time (approximately two orders of magnitude) and significantly improve the experimental results.

The empirical evaluation is performed on six real-world data sets ranging up to 45,000,000 points (two orders of magnitude larger than prior work), up to 90 dimensions, and we fit mixture models of up to 150 components. We present both absolute as well as relative log-likelihood of models trained on coresets and the models trained on the full data set. We observe computational time reduction of *two orders of magnitude* while achieving a hold-out set likelihood competitive with the models trained on the full data set.

3. Background and Problem Statement

We briefly review the approach of fitting mixture models by maximum likelihood estimation and discuss the critical aspect of approximating the log-likelihood by a weighted subset of the data set. We conclude this section by defining coresets for Gaussian mixture models.

3.1 Fitting Mixture Models by Maximum Likelihood Estimation

Suppose we are given a data set $\mathcal{X} = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$. We consider fitting a mixture of Gaussians $\theta = [(w_1, \mu_1, \Sigma_1), \ldots, (w_k, \mu_k, \Sigma_k)]$, that is, the distribution

$$P(x \mid \theta) = \sum_{i=1}^{k} w_i \mathcal{N}(x; \mu_i, \Sigma_i)$$

where $w_1, \ldots, w_k \geq 0$ are the mixture weights and $\sum_i w_i = 1$. Mean $\mu_i \in \mathbb{R}^d$ and covariance $\Sigma_i \in \mathbb{R}^{d \times d}$ parametrize the i-th mixture component, which is modeled as a multivariate normal distribution

$$\mathcal{N}(x; \mu_i, \Sigma_i) = \frac{1}{\sqrt{|2\pi\Sigma_i|}} \exp\left(-\frac{1}{2}(x - \mu_i)^T \Sigma_i^{-1}(x - \mu_i)\right).$$

We discuss extensions to more general mixture models in Section 7. Assuming the data was generated i.i.d., the negative log-likelihood of the data is

$$\mathcal{L}(\mathcal{X} \mid \theta) = -\sum_j \ln P(x_j \mid \theta),$$

and we wish to obtain the maximum likelihood estimate (MLE) of the parameters

$$\theta^* = \operatorname*{argmin}_{\theta \in \mathfrak{C}} \mathcal{L}(\mathcal{X} \mid \theta),$$

where $\mathfrak{C}$ is a set of constraints ensuring that degenerate solutions are avoided.[1] Hereby, for a symmetric matrix $\mathbf{A}$, let spec $\mathbf{A}$ be the set of all eigenvalues of $\mathbf{A}$. We define $\mathfrak{C} = \mathfrak{C}_\lambda = \{\theta = [(w_1, \mu_1, \Sigma_1), \ldots, (w_k, \mu_k, \Sigma_k)] \mid \forall_i : \text{spec}(\Sigma_i) \subseteq [\lambda, 1/\lambda]\}$ to be the set of all mixtures of k Gaussians θ, such that all the eigenvalues of the covariance matrices of θ are bounded between λ and $1/\lambda$ for $\lambda \in (0, 1)$.

3.2 Approximating the Log-likelihood

Ideally, we would like to obtain $(1+\varepsilon)$-multiplicative approximation for the likelihood

$$\prod_{x \in \mathcal{X}} P(x \mid \theta).$$

This implies an additive ε error for the log-likelihood, which is the sum of log-likelihoods. What kind of approximation accuracy may we hope to expect? Notice that there is a non-trivial issue of scale: Suppose we have a MLE θ^* for $\mathcal{X}$, and let $\alpha > 0$. Then straightforward linear algebra shows that we can obtain a MLE θ^*_α for a scaled data set $\alpha D = \{\alpha x : x \in \mathcal{X}\}$ by simply scaling all means by α, and covariance matrices by α^2. For the log-likelihood, however, it holds that $\mathcal{L}(\alpha D \mid \theta^*_\alpha) = d \ln \alpha + \mathcal{L}(\mathcal{X} \mid \theta^*)$. Therefore, optimal solutions on one scale can be efficiently transformed to optimal solutions on a different scale, while maintaining the same *additive error*. Thus, we cannot expect to obtain a $(1+\varepsilon)$-multiplicative approximation to the likelihood since any algorithm which achieves absolute error ε at any scale could be used to achieve parameter estimates (for means, covariances) with arbitrarily small error, simply by applying the algorithm to a scaled data set and transforming back the obtained solution. An alternative, scale-invariant approach may be to strive towards a *multiplicative error* $(1+\varepsilon)$ for the sum of log-likelihoods. Unfortunately, this goal is also hard to achieve: Choosing a scaling parameter α such that $d \ln \alpha + \mathcal{L}(\mathcal{X} \mid \theta^*) = 0$ would require any algorithm that achieves any bounded multiplicative error to essentially incur *no error at all* when evaluating $\mathcal{L}(\alpha \mathcal{X} \mid \theta^*)$. The above observations hold even for the case $k = 1$ and $\Sigma = I$, where the mixture θ consists of a single Gaussian, and the log-likelihood is the sum of squared distances to a point μ and an additive term.

Motivated by the scaling issues discussed above, our goal is to approximate the data set $\mathcal{X}$ by a weighted set $C = \{(\gamma_1, \mathbf{x}'_1), \ldots, (\gamma_m, \mathbf{x}'_m)\} \subseteq \mathbb{R}_+ \times \mathbb{R}^d$ such that $\mathcal{L}(\mathcal{X} \mid \theta) \approx \mathcal{L}(\mathcal{C} \mid \theta)$ for all θ, where we define

$$\mathcal{L}(\mathcal{C} \mid \theta) = -\sum_i \gamma_i \ln P(\mathbf{x}'_i \mid \theta).$$

We make use of the following decomposition that was suggested in Arora and Kannan (2005) where all Gaussians are identical spheres was studied. We decompose the negative log-likelihood $\mathcal{L}(\mathcal{X} \mid \theta)$ of a data set $\mathcal{X}$ as

$$\begin{aligned}\mathcal{L}(\mathcal{X} \mid \theta) &= -\sum_{j=1}^{n} \ln \sum_{i=1}^{k} \frac{w_i}{\sqrt{|2\pi\Sigma_i|}} \exp\left(-\frac{1}{2}(x_j - \mu_i)^T \Sigma_i^{-1} (x_j - \mu_i)\right) \\ &= -n \ln Z(\theta) + \phi(\mathcal{X} \mid \theta)\end{aligned}$$

1. Equivalently, $\mathfrak{C}$ can be interpreted as prior thresholding.

where $Z(\theta) = \sum_i \frac{w_i}{\sqrt{|2\pi\Sigma_i|}}$ is a normalizer, and the function ϕ is defined as

$$\phi(\mathcal{X} \mid \theta) = -\sum_{j=1}^{n} \ln \sum_{i=1}^{k} \frac{w_i}{Z(\theta)\sqrt{|2\pi\Sigma_i|}} \exp\left(-\frac{1}{2}(x_j - \mu_i)^T \Sigma_i^{-1}(x_j - \mu_i)\right).$$

Hereby, $Z(\theta)$ plays the role of a normalizer, which can be computed *exactly*, independently of the set $\mathcal{X}$. Function $\phi(\mathcal{X} \mid \theta)$ captures all dependencies of $\mathcal{L}(\mathcal{X} \mid \theta)$ on $\mathcal{X}$. Furthermore, $\phi(\mathcal{X} \mid \theta)$ is always nonnegative by Jensen's inequality. We can now use $\phi(\mathcal{X} \mid \theta)$ as a reference for our error bounds.

3.3 Coresets for Gaussian Mixtures

Definition 1 *We call a weighted data set $\mathcal{C}$ a (k, ε)-*coreset *for another (possibly weighted) set $\mathcal{X} \subset \mathbb{R}^d$, if for all mixtures $\theta \in \mathfrak{C}$ of k Gaussians it holds that*

$$(1 - \varepsilon)\phi(\mathcal{X} \mid \theta) \leq \phi(\mathcal{C} \mid \theta) \leq (1 + \varepsilon)\phi(\mathcal{X} \mid \theta).$$

Hereby $\phi(\mathcal{C} \mid \theta)$ is generalized to weighted data sets $\mathcal{C}$ in the natural way (weighing the contribution of each summand $\mathbf{x}'_j \in \mathcal{C}$ by its weight γ_j). Thus, as $\varepsilon \to 0$, for a sequence of (k, ε)-coresets $\mathcal{C}_\varepsilon$ we have that

$$\begin{aligned}\sup_{\theta \in \mathfrak{C}} |\mathcal{L}(\mathcal{C}_\varepsilon \mid \theta) - \mathcal{L}(\mathcal{X} \mid \theta)| &= \sup_{\theta \in \mathfrak{C}} |-n \log Z(\theta) + \phi(\mathcal{C}_\varepsilon \mid \theta) + n \log Z(\theta) - \phi(\mathcal{X} \mid \theta)| \\ &= \sup_{\theta \in \mathfrak{C}} |\phi(\mathcal{C}_\varepsilon \mid \theta) - \phi(\mathcal{X} \mid \theta)| \to 0.\end{aligned}$$

which implies that $\mathcal{L}(\mathcal{C}_\varepsilon \mid \theta)$ uniformly approximates $\mathcal{L}(\mathcal{X} \mid \theta)$ (over $\theta \in \mathfrak{C}$).

The main idea behind constructing a (k, ε)-coreset $\mathcal{C}$ is to reduce the problem of fitting a mixture model on $\mathcal{X}$ to one of fitting a model on $\mathcal{C}$, since the optimal solution $\theta_\mathcal{C}$ is a good approximation (in terms of log-likelihood) of θ^*. While finding the optimal $\theta_\mathcal{C}$ is a difficult problem, one can use a (weighted) variant of the EM algorithm to find a good solution. Moreover, if $|\mathcal{C}| \ll |\mathcal{X}|$, running EM on $\mathcal{C}$ is orders of magnitude faster than solving it on $\mathcal{X}$. We provide more details on solving the density estimation problem in Section 5.

The key question is whether we can efficiently construct a *small* (k, ε)-coreset. In what follows, we show that, perhaps surprisingly, one can efficiently construct coresets of size *independent* of the cardinality of $\mathcal{X}$ with only polynomial dependence on d, k, λ^{-1} and ε^{-1}.

4. Efficient Coreset Construction

We start by contrasting the coreset approach with the naive approach of fitting the model to a uniform sample and show that the uniform sampling can perform arbitrarily badly, while explicit worst-case guarantees can be made for the coreset based approach. We then present a simple algorithm for coreset construction. We conclude with a bound on the coreset size for mixtures of Gaussians.

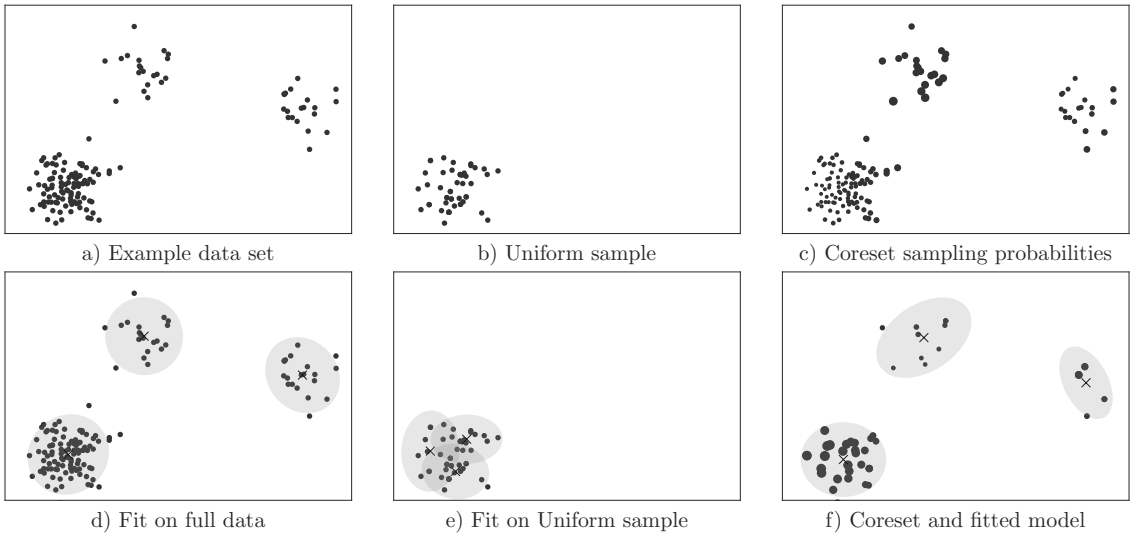

Figure 1: Example of fitting a mixture model on the full data set (a, d), uniform sample (b, e) and the coreset (c, f). With unbalanced data the uniform sample is likely to miss small clusters and can be arbitrarily penalized in terms of the log-likelihood (e). Note that the coreset sampling probabilities are inversely proportional to the size of the cluster which results in more representative samples (f).

4.1 Naive Approach via Uniform Sampling

A naive approach towards approximating $\mathcal{X}$ would be to just pick a subset $\mathcal{C}$ uniformly at random. In particular, suppose the data set is generated from a mixture of two spherical Gaussians ($\Sigma_i = \mathbf{I}$) with weights $w_1 = \frac{1}{\sqrt{n}}$ and $w_2 = 1 - \frac{1}{\sqrt{n}}$. Unless $m = \Omega(\sqrt{n})$ points are sampled, with constant probability no data point generated from the second Gaussian is sampled. By moving the means of the Gaussians apart, $\mathcal{L}(\mathcal{X} \mid \theta_{\mathcal{C}})$ can be made arbitrarily worse than $\mathcal{L}(\mathcal{X} \mid \theta_{\mathcal{X}})$, where $\theta_{\mathcal{C}}$ and $\theta_{\mathcal{X}}$ are MLEs on $\mathcal{C}$ and $\mathcal{X}$ respectively. Thus, even for two well-separated Gaussians, uniform sampling can perform arbitrarily poorly which is illustrated in Figure 1. This example already suggests that we must devise a sampling scheme that adaptively selects representative points from all "clusters" present in the data set. However, this suggests that obtaining a coreset requires solving a chicken-and-egg problem, where we need to understand the density of the data to obtain the coreset, but simultaneously would like to use the coreset for density estimation.

4.2 Better Approximation via Adaptive Sampling

The key idea behind the coreset construction is that we can break the chicken-and-egg problem by first obtaining a rough approximation $\mathcal{A}$ of the problem on $\mathcal{X}$ and using it to construct a non-uniform sampling scheme. This non-uniform random sampling can be understood as an importance-weighted estimate of the log-likelihood $\mathcal{L}(\mathcal{X} \mid \theta)$, where the weights are optimized in order to reduce the variance. Feldman and Langberg (2011) successfully used the same idea to construct coresets for geometric clustering problems

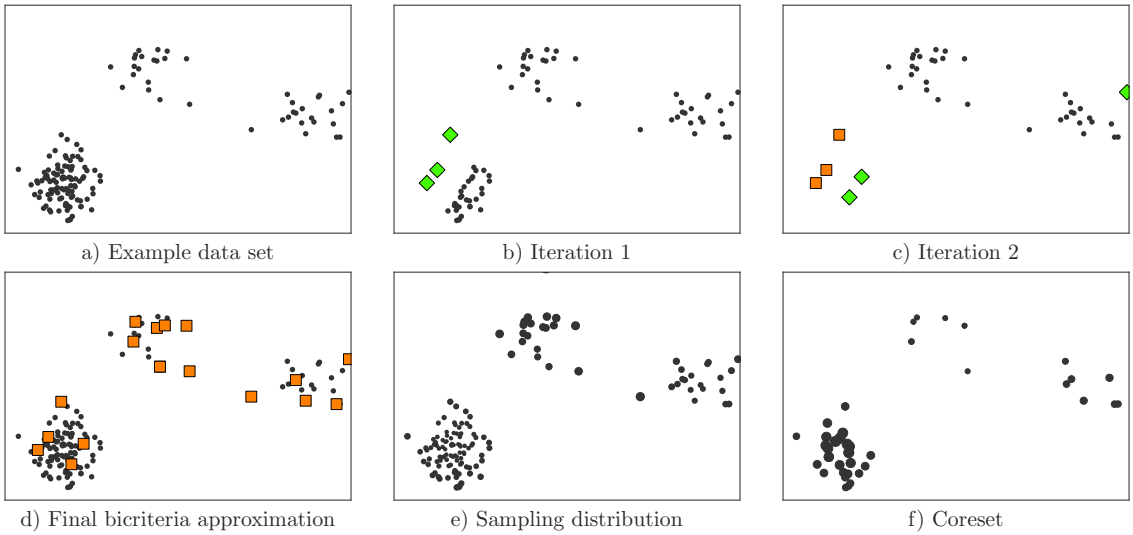

Figure 2: Illustration of the coreset construction on a synthetic data set. (b, c) Two iterations of constructing the bicriteria approximation $\mathcal{A}$ via Algorithm 3. Green diamonds are points sampled uniformly from remaining points, orange squares are bicriteria points selected in previous iterations. In each iteration a fraction of points closest to $\mathcal{A}$ are discarded. (d) Final approximate clustering $\mathcal{A}$. (e) Induced non-uniform sampling distribution: radius of points is proportional to the sampling probability. The points in dense clusters are sampled with a smaller probability. (f) Coreset points sampled from distribution in (e). The weight of the sampled points is inversely proportional to the sampling probability.

such as k-means and k-median. The rough approximation can either be a *constant-factor approximation* or a *bicriteria approximation* (using more than k components). Below we show that, even though we seek to fit Gaussian mixture models, it suffices to use a bicriteria approximation with respect to the optimal k-means clustering as the initial approximation (see Appendix B.1). Formally, denote the distance between a point $x \in \mathbb{R}^d$ and a set $\mathcal{A} \subset \mathbb{R}^d$ by $\mathrm{d}(x, \mathcal{A}) := \min_{a \in \mathcal{A}} \|x - a\|_2$. We define the bicriteria approximation for the k-means problem as follows.

Definition 2 (Bicriteria approximation) *Let $\mathcal{X} \subset \mathbb{R}^d$, $\alpha \geq 1$, and $\beta \in \mathbb{N}$. Set $\mathcal{A} \subset \mathbb{R}^d$ of β points is an (α, β)-bicriteria approximation of the optimal k-means clustering if*

$$\sum_{x \in \mathcal{X}} \mathrm{d}^2(x, \mathcal{A}) \leq \alpha \min_{\substack{C \subset \mathbb{R}^d \\ |C| = k}} \sum_{x \in \mathcal{X}} \mathrm{d}^2(x, C).$$

One choice for the bicriteria approximation for k-means clustering is to use `k-means++` given in Algorithm 2. The algorithm produces a $(c \log_2 k, k)$-bicriteria approximation to the k-means objective, where c is a sufficiently large constant (Arthur and Vassilvitskii, 2007). Interestingly, for $\beta \geq \lceil 16(k + \sqrt{k}) \rceil$ the algorithm produces a constant factor approximation ($\alpha = 20$) with probability at least 0.03 (Aggarwal et al., 2009). This probability can be boosted to $1 - \delta$ by repeating the algorithm $\ln 1/\delta$ times and selecting the solution with the smallest quantization error. In the case of GMMs (and natural extensions) the approach

Algorithm 1 Coreset($\mathcal{X}, \mathcal{A}, \alpha, s$)

for each j in $\mathcal{A}$
  $\mathcal{X}_j \leftarrow$ Set of points from $\mathcal{X}$ closest to point j. Ties can be broken arbitrarily.
for each $j \in \mathcal{A}$, **for each** $x \in \mathcal{X}_j$
  $s(x) \leftarrow \alpha \mathrm{d}^2(x, \mathcal{A}) + \alpha \sum_{x' \in \mathcal{X}_j} \mathrm{d}^2(x', \mathcal{A})/|\mathcal{X}_j| + \sum_{x' \in \mathcal{X}} \mathrm{d}^2(x', \mathcal{A})/|\mathcal{X}_j|$
for each $x \in \mathcal{X}$
  $p(x) \leftarrow \frac{s(x)}{\sum_{x' \in \mathcal{X}} s(x')}$
$\mathcal{C} \leftarrow$ Sample s weighted points from $\mathcal{X}$, where each point x is sampled with probability
  $p(x)$ and has weight $\frac{1}{s \cdot p(x)}$.
return $\mathcal{C}$

based on this bicriteria approximation leads to coresets of size *independent* of the data set size (Corollary 3). Other choices for the bicriteria approximation will lead to different tradeoffs in terms of the approximation factor and coreset size. For example, Feldman and Langberg (2011) apply Algorithm 3 which, with probability at least $1 - \delta$, provides a constant-factor approximation of size $\mathcal{O}(dk \log 1/\delta \log n)$. The algorithm illustrated in Figure 2 iteratively samples a small number c (sublinear in n) of points, and removes half of the data set closest to the sampled points in each iteration. For other bicriteria approximation algorithms for k-means clustering offering different tradeoffs in terms of computational complexity and the approximation guarantee we refer the reader to Bachem et al. (2016a,b) and Makarychev et al. (2016).

The pseudocode for the coreset construction is given in Algorithm 1. The sampling strategy is biased towards sampling points from small clusters, as well as points which are further away from cluster centers in the bicriteria approximation. We note that this sampling strategy was not selected arbitrarily, but rather follows from a framework introduced by Langberg and Schulman (2010). It can be shown that sampling proportionally to the *sensitivity* — the worst case impact of each point – guarantees that the cost estimated on the coreset $\phi(\mathcal{C} \mid \theta)$ is an unbiased estimator of $\phi(\mathcal{X} \mid \theta)$ with bounded variance (Langberg and Schulman, 2010). For a detailed discussion on why sensitivity is *the* critical quantity we refer the reader to the tutorial in Bachem et al. (2017b). The following corollary bounds the coreset size and is a direct consequence of our main technical result, Theorem 7 (Section 7).

Corollary 3 *Let $\mathcal{X} \subset \mathbb{R}^d$, $\delta \in (0, 1)$, $\varepsilon \in (0, 1/2)$, $k \geq 1$, $\lambda \in (0, 1)$. Let $\mathfrak{C}$ be the collection of all mixtures of k components $\theta = [(w_1, \mu_1, \Sigma_1), \ldots, (w_k, \mu_k, \Sigma_k)]$ such that Σ_i is a $d \times d$ covariance matrix whose singular values are between λ and $1/\lambda$ for every $1 \leq i \leq k$. Let $\mathcal{A}$ be the output with smallest quantization error of $\Theta(\ln 1/\delta)$ runs of Algorithm 2 with $\beta = k$. Let $\mathcal{C}$ be the output of Algorithm 1 with $\alpha = 16(\log_2 k + 2)$, bicriteria approximation $\mathcal{A}$ and coreset size $s = \Theta\big(d^4 k^6 \lambda^{-8} \varepsilon^{-2}\big)$. Then, with probability at least $1 - \delta$, it holds that*

$$(1 - \varepsilon)\phi(\mathcal{X} \mid \theta) \leq \phi(\mathcal{C} \mid \theta) \leq (1 + \varepsilon)\phi(\mathcal{X} \mid \theta).$$

for every mixture $\theta \in \mathfrak{C}$. The total computational complexity is $\mathcal{O}(nkd \ln 1/\delta)$.

Algorithm 2 K-Means++$(\mathcal{X}, k)$

$\mathcal{A} \leftarrow \{$Sample $x \in \mathcal{X}$ uniformly at random$\}$
for $j \leftarrow 2$ **to** k
  for x **in** $\mathcal{X}$
    $p(x) \leftarrow \frac{\mathrm{d}^2(x,\mathcal{A})}{\sum_{x' \in \mathcal{X}} \mathrm{d}^2(x',\mathcal{A})}$
  Sample $x \in \mathcal{X}$ with probability $p(x)$ and add it to $\mathcal{A}$.
return $\mathcal{A}$

Algorithm 3 Adaptive sampling$(\mathcal{X}, k, \delta)$

$R \leftarrow D, \mathcal{A} \leftarrow \emptyset, c \leftarrow \lceil 10dk \ln(1/\delta) \rceil$
while $|R| > c$
  $S \leftarrow$ Sample c points uniformly from R
  $P \leftarrow \lceil |R|/2 \rceil$ points from R closest to S
  $R \leftarrow R \setminus P$
  $\mathcal{A} \leftarrow \mathcal{A} \cup S$
$\mathcal{A} \leftarrow \mathcal{A} \cup R$
return $\mathcal{A}$

5. Fitting a GMM on the Coreset using Weighted EM

Once the coreset $\mathcal{C}$ is constructed, we need to fit a mixture model that takes into account the point weights. Since the coreset size is independent of the cardinality of $\mathcal{X}$ we can (at least from the perspective of developing a polynomial time algorithm) afford to use an "expensive" method. In geometric clustering problems such as k-means or k-median, where data points are hard-assigned to the closest cluster (point, subspace, etc.), it is possible to find the *optimal* clustering via exhaustive search, by simply considering all possible partitions of the coreset, and picking the best one. This procedure – constructing a coreset of size independent of n, and then using exhaustive search on the coreset – yields a (randomized) polynomial time approximation scheme (PTAS): It is guaranteed to achieve multiplicative error $1 + \varepsilon$, in time which is polynomial in n, but exponential in all other quantities (in particular $1/\varepsilon$). For mixture models, this exhaustive search algorithm is not feasible, since points are not hard-assigned to a cluster, but "soft-assigned" (according to the cluster membership probabilities). One approach, which we employ in our experiments, is to use a natural generalization of the EM algorithm, which takes the coreset weights into account. We provide the algorithm for the case of GMMs. For other mixture distributions, the *expectation* and *maximization* steps need to be modified appropriately. Since the EM algorithm is applied on a significantly smaller data set, it can be initialized using multiple random restarts. In our experiments, we show that running weighted EM on the coreset typically leads to comparable performance (in terms of log-likelihood) as running EM on the full data set. The derivation of the EM update equations is presented in Appendix C.

Algorithm 4 Fitting Gaussian mixture models with weighted data sets

procedure Initialize$(\mathcal{X}, \boldsymbol{\gamma}, k, \lambda)$
  $\boldsymbol{\eta} \leftarrow$ Weighted K-Means$(\mathcal{X}, \boldsymbol{\gamma}, k)$ ▷ $\eta_{ij} = 1$ iff $\boldsymbol{x_i}$ is assigned to the j-th center
  return Maximization$(\boldsymbol{\eta}, \lambda)$
procedure Fit$(\mathcal{X}, \boldsymbol{\gamma}, k, \lambda)$
  $\boldsymbol{w}, \boldsymbol{\mu}, \boldsymbol{\Sigma} \leftarrow$ Initialize$(\mathcal{X}, \boldsymbol{\gamma}, k)$
  while not converged
    $\boldsymbol{\eta} \leftarrow$ Expectation$(\mathcal{X}, \boldsymbol{\gamma}, \boldsymbol{w}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$
    $\boldsymbol{w}, \boldsymbol{\mu}, \boldsymbol{\Sigma} \leftarrow$ Maximization$(\boldsymbol{\eta}, \lambda)$
  return $\boldsymbol{w}, \boldsymbol{\mu}, \boldsymbol{\Sigma}$

Algorithm 5 Expectation($\mathcal{X}, \boldsymbol{\gamma}, \boldsymbol{w}, \boldsymbol{\mu}, \boldsymbol{\Sigma}$)

Require: $\boldsymbol{w} \in \mathbb{R}^k, \boldsymbol{\mu} \in \mathbb{R}^{k \times d}, \boldsymbol{\Sigma} \in \mathbb{R}^{k \times d \times d}$

$\boldsymbol{z} \leftarrow \mathbf{0}_n$
for i $\leftarrow 1$ **to** n
  for j $\leftarrow 1$ **to** k
    $\eta_{ij} \leftarrow w_j \cdot \mathcal{N}(\boldsymbol{x_i} \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$
    $z_i \leftarrow z_i + \eta_{ij}$
for i $\leftarrow 1$ **to** n
  for j $\leftarrow 1$ **to** k
    $\eta_{ij} \leftarrow \gamma_i \cdot \eta_{ij} \,/\, z_i$
return $\boldsymbol{\eta}$

Algorithm 6 Maximization($\boldsymbol{\eta}$, λ)

Require: $\boldsymbol{\eta} \in \mathbb{R}^{n \times k}$, $\lambda > 0$

$\boldsymbol{z} \leftarrow \mathbf{0}_k, \{\boldsymbol{\mu}\}_{j=1}^k \leftarrow \mathbf{0}_d, \{\boldsymbol{\Sigma}\}_{j=1}^k \leftarrow \mathbf{0}_{d \times d}$
for j $\leftarrow 1$ **to** k
  for i $\leftarrow 1$ **to** n
    $z_j \leftarrow z_j + \eta_{ij}$
for j $\leftarrow 1$ **to** k
  for i $\leftarrow 1$ **to** n
    $\boldsymbol{\mu}_j \leftarrow \boldsymbol{\mu}_j + \eta_{ij} \boldsymbol{x}_j \,/\, z_j$
    $\boldsymbol{\Sigma}_j \leftarrow \boldsymbol{\Sigma}_j + \eta_{ij} (\boldsymbol{x}_i - \boldsymbol{\mu}_j)(\boldsymbol{x}_i - \boldsymbol{\mu}_j)^T$
for j $\leftarrow 1$ **to** k
  $\boldsymbol{\mu}_j \leftarrow \boldsymbol{\mu}_j \,/\, z_j, \boldsymbol{\Sigma}_j \leftarrow \boldsymbol{\Sigma}_j \,/\, z_j + \boldsymbol{I}_\lambda$
return $(z \,/\, ||\boldsymbol{z}||_1, \boldsymbol{\mu}, \boldsymbol{\Sigma})$

6. Streaming and Parallel Computation

One advantage of coresets is that they can be constructed in parallel, as well as in a streaming setting where data points arrive one by one, and it is impossible to remember the entire data set due to memory constraints. The key insight is that coresets satisfy certain composition properties, which have previously been used by Har-Peled and Mazumdar (2004) for streaming and parallel construction of coresets for geometric clustering problems such as k-median and k-means.

1. Suppose $\mathcal{C}_1$ is a (k, ε)-coreset for $\mathcal{X}_1$, and $\mathcal{C}_2$ is a (k, ε)-coreset for $\mathcal{X}_2$. Then $\mathcal{C}_1 \cup \mathcal{C}_2$ is a (k, ε)-coreset for $\mathcal{X}_1 \cup \mathcal{X}_2$.

2. Suppose $\mathcal{C}$ is a (k, ε)-coreset for $\mathcal{X}$, and $\mathcal{C}'$ is a (k, δ)-coreset for $\mathcal{C}$. Then $\mathcal{C}'$ is a $(k, (1+\varepsilon)(1+\delta) - 1)$-coreset for $\mathcal{X}$.

We now discuss how to exploit these properties for parallel and streaming computation.

6.1 Streaming Computation

In the streaming setting, we assume that points arrive one-by-one, but we do not have enough memory to remember the entire data set. Thus, we wish to maintain a coreset over time, while keeping only a small subset of $\mathcal{O}(\log n)$ coresets in memory, where n is the number of points seen so far. There is a general reduction that shows that a small coreset scheme to a given problem suffices to solve the corresponding problem on a streaming input (Bentley and Saxe, 1980; Har-Peled and Mazumdar, 2004). The idea is to construct and store in memory a coreset for every block of $\text{poly}(d, k, \lambda^{-1}, \varepsilon^{-1})$ consecutive points arriving in a stream. When we have two coresets in memory we first merge them which results in a (k, ε)-coreset via property (1). Then we can compress them by computing a single coreset from the merged coresets via property (2) to avoid the increase in the coreset size.

An important subtlety arises: While merging two coresets via property (1) does not increase the approximation error, compressing a coreset via property (2) *does* increase

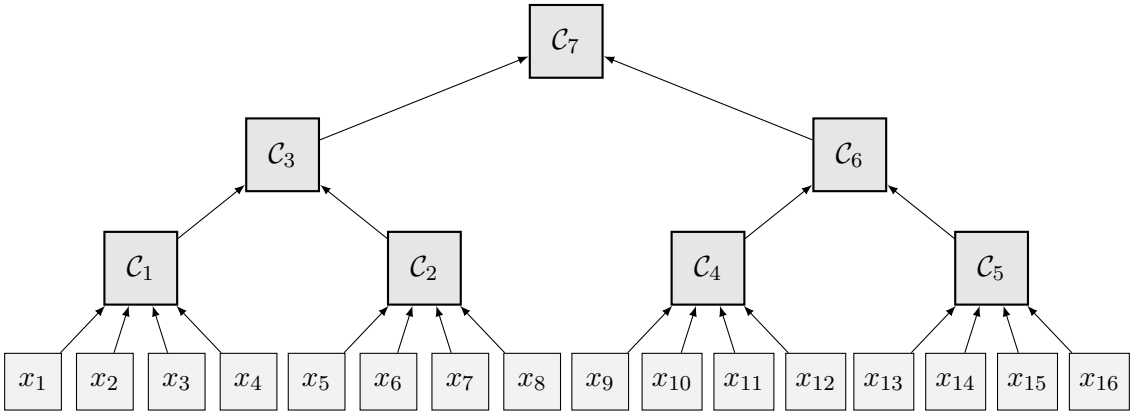

Figure 3: Tree construction for generating coresets in parallel or from data streams. Black arrows indicate "merge-and-compress" operations. The (intermediate) coresets $\mathcal{C}_1, \ldots, \mathcal{C}_7$ are enumerated in the order in which they would be generated in the streaming case. In the parallel case, $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_4$ and $\mathcal{C}_5$ would be constructed in parallel, followed by parallel construction of $\mathcal{C}_3$ and $\mathcal{C}_6$, finally resulting in $\mathcal{C}_7$.

the error. A naive approach that merges and compresses immediately as soon as two coresets have been constructed can incur an exponential increase in approximation error. Fortunately, it is possible to organize the merge-and-compress operations in a binary tree of height $\mathcal{O}(\log n)$, where we need to store in memory a single coreset for each level on the tree (thus requiring only $\text{poly}(d, k, \lambda^{-1}, \varepsilon^{-1}, \log n)$ memory). The following analysis is based on Feldman et al. (2013b, Theorem 10.1.).

Consider Figure 3 which illustrates the tree computation for an example data set. In the following, ε-coreset denotes a (ε, k)-coreset (k is fixed). In the first step, we construct a ε-coreset for $x_1, \ldots, x_4$. We then construct a ε-coreset for $x_5, \ldots, x_8$. At this point we have two ε-coresets and their union is, by property (1), a ε-coreset for $x_1, \ldots, x_8$. By property (2) a coreset $\mathcal{C}_3$ of the union of those two coresets is a 4ε-coreset for $x_1, \ldots, x_8$ since

$$(1+\varepsilon)(1+\varepsilon) = 1 + 2\varepsilon + \varepsilon^2 \leq 1 + 4\varepsilon$$

for $0 \leq \varepsilon \leq 1$. Hence, we discard coresets $\mathcal{C}_1$ and $\mathcal{C}_2$ and keep only $\mathcal{C}_3$ in memory. We apply the same approach for $x_9, \ldots, x_{16}$ and obtain $\mathcal{C}_6$, a 4ε-coreset for $x_9, \ldots, x_{16}$. Once again, we obtained two coresets at the same level of the tree ($\mathcal{C}_3$ and $\mathcal{C}_6$) and merging them we obtain a 4ε-coreset for $x_1, \ldots, x_{16}$. By property (2), the coreset of the union of $\mathcal{C}_3$ and $\mathcal{C}_6$ is a 13ε-coreset for the whole data set since

$$(1+4\varepsilon)(1+\varepsilon) = 1 + 5\varepsilon + 4\varepsilon^2 \leq 1 + 13\varepsilon.$$

We now consider the general setting whereby we need to return a coreset after reading n points. To ensure final accuracy of ε, it suffices that the intermediate coreset has accuracy $\varepsilon' = \frac{\varepsilon}{6 \log n}$. Indeed, since the height of the tree is at most $\lceil \log n \rceil$, the approximation error of the union of all coresets is at most

$$(1+\varepsilon')^{\lceil \log n \rceil} = (1 + \frac{\varepsilon}{6 \log n})^{\lceil \log n \rceil} \leq e^{\frac{\varepsilon}{6}} \leq 1 + \frac{\varepsilon}{3}.$$

Thus, by property (2), a $(\varepsilon/3)$-coreset of the union of all coresets in memory has the approximation error bounded by $(1+\varepsilon/3)^2 \leq 1+\varepsilon$. We conclude that for a desired approximation error of ε it suffices to set each coreset error to $\varepsilon' = \frac{\varepsilon}{6\log n}$. A subtle issue arises in the streaming setting — we do not know n a priori. Intuitively, this issue can be resolved by computing coresets for data batches of exponentially increasing size – the total time and space requirements are dominated by the last batch whose size is upper bounded by n (Feldman et al., 2013b). We summarize this result in the following theorem.

Theorem 4 *A (k,ε)-coreset for a stream of n points in $\mathbb{R}^d$ can be computed for the λ-semi-spherical GMM using update time per point and memory* $\mathrm{poly}(d,k,\lambda^{-1},\varepsilon^{-1},\log n,\log(1/\delta))$ *with probability at least $1-\delta$.*

In order to construct a coreset for the union of two (weighted) coresets, we use weighted versions of Algorithms 1 and 2, where we consider a weighted point as duplicate copies of a non-weighted point (possibly with fractional weight).

6.2 Distributed Computation

Using the same ideas from the streaming model, a (non-parallel) coreset construction can be transformed into a parallel one. We partition the data and compute a coreset for each partition independently. We then in parallel merge via property (1) two coresets, and compute a single coreset for every pair of such coresets exploiting the property (2). Continuing in this manner yields a process that takes $\mathcal{O}(\log n)$ iterations of parallel computation. This computation is naturally suited for map-reduce (Dean and Ghemawat, 2004) style computations, where the map tasks compute coresets for disjoint parts of $\mathcal{X}$, and the reduce tasks perform the merge-and-compress operations. Figure 3 illustrates this parallel construction. We summarize our distributed computation result in the following theorem.

Theorem 5 *A (k,ε)-coreset for a set of n points in $\mathbb{R}^d$ can be computed for the λ-semi-spherical GMM using m machines in time* $(n/m)\cdot\mathrm{poly}(d,k,\lambda^{-1},\varepsilon^{-1},\log(1/\delta),\log n)$ *with probability at least $1-\delta$.*

Furthermore, if we have enough memory on one of the machines we can apply a simpler algorithm. First, partition the data to m machines and compute a $(\varepsilon/3)$-coreset on each, producing coresets $\mathcal{C}_1,\mathcal{C}_2,\ldots,\mathcal{C}_m$. Then the union $C=\cup_{i=1}^m\mathcal{C}_i$ is a $(\varepsilon/3)$-coreset for the whole data set and its size is bounded by $m\cdot\max_i|\mathcal{C}_i|$. To obtain a coreset of size independent of the number of machines m, construct a $(\varepsilon/3)$-coreset of $\mathcal{C}$. The last coreset is in fact a ε-coreset since $(1+\varepsilon/3)^2\leq 1+\varepsilon$ for $\varepsilon\in(0,1)$

7. Extensions and Generalizations

We now show how the connection between estimating the parameters for mixture models and problems in computational geometry can be leveraged further. Our analysis is based on combinatorial considerations (such as the complexity of sub-levelsets of GMMs) and probabilistic methods (importance sampling). Therefore, generalizations to other non-Euclidean distance functions, or error functions such as (non-squared) distances is straightforward. The critical property that we need is a generalization of the triangle inequality, as

shown in Lemma 6. The double triangle inequality $\|a-c\|_2^2 \leq 2(\|a-b\|_2^2 + \|b-c\|_2^2)$ that we used in this paper can be replaced by the generalized triangle inequality, $\|a-c\|_2^z \leq 2^{\mathcal{O}(z)}(\|a-b\|_2^z + \|b-c\|_2^z)$ which allows us to develop the following lemma.

Lemma 6 *Let $x \in \mathbb{R}^d, z \in \mathbb{N}$, $\theta \in \mathfrak{C}$ and define*

$$\phi_z(x \mid \theta) = -\ln\left(\sum_{i=1}^{k} w_i \exp\left(-\frac{1}{2}\left\|\Sigma_i^{-1/2}(x-\mu_i)\right\|_2^z\right)\right).$$

Then, for every $x, a \in \mathbb{R}^d$

$$\phi_z(x \mid \theta) \leq \frac{2^{z-2}}{\lambda^z} \cdot \|x-a\|_2^z + 2^{z-1}\phi_z(a \mid \theta).$$

The proof is provided in the Appendix B.1. We apply Lemma 6 to bound the necessary coreset size in case where the distances are not necessarily Euclidean as formalized by the following theorem.

Theorem 7 *Let $\mathcal{X} \subset \mathbb{R}^d$, $|\mathcal{X}| = n$, $\delta \in (0,1)$, $\varepsilon \in (0, 1/2)$, $k \geq 1$, $\lambda \in (0,1)$ and $z \in \mathbb{N}$, Let $\mathfrak{C}$ be the collection of all mixtures of k components $\theta = [(w_1, \mu_1, \Sigma_1), \ldots, (w_k, \mu_k, \Sigma_k)]$ such that Σ_i is a $d \times d$ covariance matrix whose singular values are between λ and $1/\lambda$ for every $1 \leq i \leq k$. Consider Algorithm 1 and Algorithm 2 where $\mathrm{d}^2(\cdot,\cdot)$ is replaced by $\mathrm{d}^z(\cdot,\cdot)$. Let $\mathcal{A}$ be the output with smallest quantization error of $\Theta(\ln 1/\delta)$ runs of Algorithm 2 with $\beta = k$. Let $\mathcal{C}$ be the output of Algorithm 1 with $\alpha = 2^{z+2}(\log_2 k + 2)$, bicriteria approximation $\mathcal{A}$ and coreset size*

$$s = \Theta\left(\frac{d^4 k^6}{\lambda^{4z}\varepsilon^2}\right).$$

Then, with probability at least $1-\delta$, for all $\theta \in \mathfrak{C}$,

$$(1-\varepsilon)\phi_z(\mathcal{X} \mid \theta) \leq \phi_z(\mathcal{C} \mid \theta) \leq (1+\varepsilon)\phi_z(\mathcal{X} \mid \theta),$$

where $Z(\theta) = \sum_i \frac{w_i}{g(\theta_i)}$ and

$$\phi_z(\mathcal{X} \mid \theta) = -\sum_{x \in \mathcal{X}} \ln \sum_{i=1}^{k} \frac{w_i}{Z(\theta) g(\theta_i)} \exp\left(-\frac{1}{2}\left\|\Sigma_i^{-1/2}(x-\mu_i)\right\|_2^z\right)$$

using the normalizer

$$g(\theta_i) = \int \exp\left(-\frac{1}{2}\left\|\Sigma_i^{-1/2}(x-\mu_i)\right\|_2^z\right) dx.$$

The detailed analysis is provided in Appendix B.

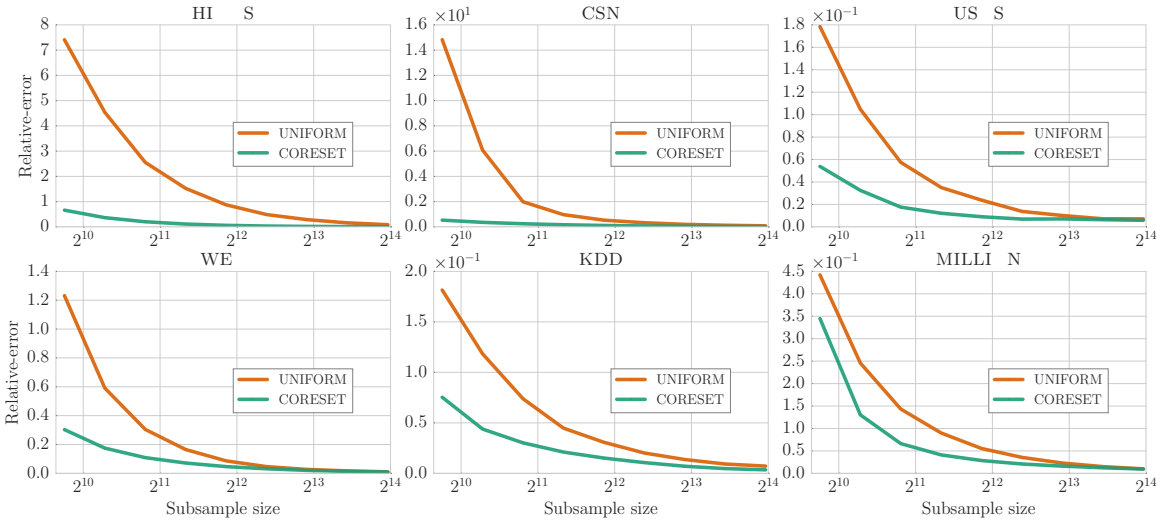

Figure 4: Models trained on coresets drastically outperform those trained on uniform samples of the same size. Small coresets are enough to train a competitive model.

8. Experimental Results

In this section we demonstrate the effectiveness of using coresets for training mixture models. We propose that instead of fitting the model on the full data set it is enough to consider relatively small coresets. We compare our coreset based approach to the "naive" approach of uniformly subsampling the data using models trained on the full data set as a baseline. In particular, we compare both absolute log-likelihood on *hold-out data* and well as the log-likelihood relative to the models trained on the full data set.

8.1 Protocol

In the experiments we use coresets and uniform samples of various sizes selected from a logarithmic grid in range $[864, 16000]$. For uniform subsampling, we subsample the data set

DATA SET	SIZE	TRAIN	FULL	TEST	D	K	λ
HIGGS	11×10^6	10×10^6	2×10^6	1×10^6	2	150	10^{-3}
CSN	120×10^3	100×10^3	100×10^3	20×10^3	17	30	10^{-3}
USGS	59×10^3	50×10^3	60×10^3	9×10^3	3	100	10^{-3}
KDD	145×10^3	100×10^3	100×10^3	45×10^3	74	10	1
WEBSCOPE	45×10^6	44×10^6	2×10^6	1×10^6	5	150	10^{-6}
MILLION	515×10^3	400×10^3	400×10^3	115×10^3	90	50	10^{-3}

Table 1: Data set size, number of training instances for the subsampling methods, number of training instances for the full model, hold-out data set size, ambient dimension, the number of mixture components, and the covariance threshold.

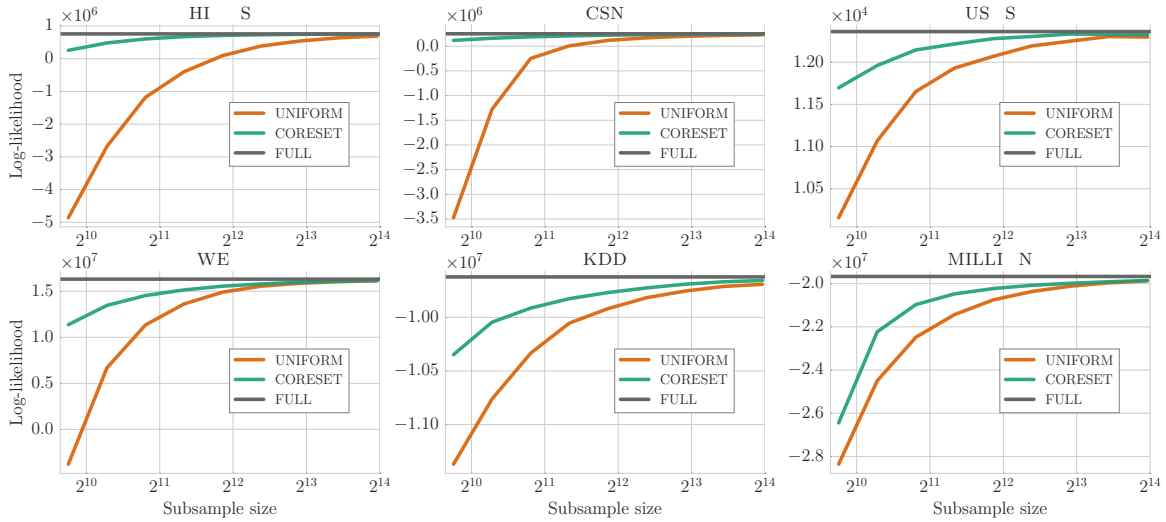

Figure 5: Average log-likelihood of the models trained on uniform samples and models trained on coresets with respect to the models trained on the full data set. Coresets dominate the uniform samples and, in some cases, outperform the models trained on the full data set.

and fit the model using Algorithm 4 with uniform weights for each point. For the coreset based approach with k mixture components we first construct a bicriteria approximation using Algorithm 2 and pass it to Algorithm 1 with $\alpha = 16(\log_2 k + 2)$ as suggested by Corollary 3. Finally, we fit a GMM on the weighed coreset using Algorithm 4 with λ as specified in Table 1. For each run we store the sampling time, solving time and the log-likelihood on a hold-out set. We compare the log-likelihood on hold-out data of models trained on the uniform subsample, the coreset and the full data set. To ensure robustness against bad initializations we compare the median results from 200 runs for the subsampling methods and 100 runs on the full data set.

8.2 Data Sets and Results

1. Higgs high-energy physics contains 11,000,000 instances describing signal processes which produce Higgs bosons and background processes which do not (Baldi et al., 2014). We consider the two principal components components and fit GMMs with 150 components. We use 9,000,000 instances for training and the rest for testing. As we can't fit the GMM on the full data set in reasonable time, we consider a uniform subsample of 2,000,000 points. Models trained on coresets achieve a comparable log-likelihood to the full solution 111 times faster (2 minutes vs 3.7 hours).

2. Community Seismic Network (CSN) uses smart phones with accelerometers as inexpensive seismometers for earthquake detection. In Faulkner et al. (2011), 7 GB of acceleration data was recorded from volunteers carrying and operating their phone in normal conditions (walking, talking, on desk, etc.). From this data, 17-dimensional feature vectors were computed (containing frequency information, moments, etc.). Out of 120,000 points we use 100,000 for training and the rest for testing and fit

GMMs with 30 components. Models trained on coresets achieve a comparable log-likelihood to the full solution 8 times faster.

3. U.S. geological survey contains coordinates of all earthquakes since 1972 as reported by USGS. It contains 59,000 instances with 8 features which we mapped to 3D space using WGS 84. We use 50,000 for training and the rest for testing and fit GMMs with 100 components. Since this is a relatively small data set and the data is balanced we don't expect large speedups. Models trained on coresets achieve a comparable log-likelihood 6 times faster.

4. Yahoo! Webscope R6A contains a fraction of user click log for news articles displayed in the Featured Tab of the Today Module on Yahoo! Front Page during the first ten days in May 2009.[2] The data set contains 45,000,000 user visits to the Today Module. We use 44,000,000 for training and the rest for testing. Figure 5) and Figure 4e) show that coresets consistently obtain higher log-likelihood for uniform subsample GMMs of the same size. Models trained on coresets achieve a comparable log-likelihood 80 times faster (5 minutes vs 7.2 hours).

5. KDD Cup 2004 The data set contains 145,000 samples with 74 features measuring the match between a protein and a native sequence.[3] We use 100,000 instances for training and the rest for testing and fit GMMs with 10 components. Models trained on coresets achieve a comparable log-likelihood 10 times faster.

6. Million song data set contains 515.345 instances with 90 features each. The features represent timbre average and timbre covariance for mostly western, commercial tracks ranging from 1922–2011. We normalize each column to unit variance. We use 400,000 for training and the rest for testing and fit GMMs with 50 components. Models trained on coresets achieve a comparable log-likelihood 40 times faster.

2. Available at `http://research.yahoo.com`.
3. Available at `http://osmot.cs.cornell.edu/kddcup/datasets.html`.

Table 2: Relative quality and speedup for various coreset sizes. The coreset size required in practice is drastically smaller than worst-case size predicted by the analysis.

	relative error			speedup		
	m = 2581	m = 5355	m = 11109	m = 2581	m = 5355	m = 11109
higgs	11.15%	3.39%	1.10%	212.84×	181.78×	111.86×
csn	16.95%	7.54%	3.48%	66.28×	25.75×	9.59×
usgs	1.21%	0.69%	0.64%	25.44×	14.71×	6.66×
web	7.17%	3.17%	1.39%	106.54×	102.67×	87.55×
kdd	2.11%	1.07%	0.47%	66.79×	12.08×	5.91×
million	4.11%	2.09%	1.24%	930.61×	163.90×	47.70×

8.3 Discussion

The experimental results are summarized in Table 2. The empirical results suggest that the coreset sizes required to obtain a competitive log-likelihood to the models trained on full data set are drastically smaller than the ones suggested by the analysis. This discrepancy is due the fact that coresets uniformly approximate the log-likelihood. As such, larger samples are required to allow for degenerate queries which rarely happen in practice. Both Figure 4 and Figure 5 demonstrate that the models trained on coresets generalize better than the ones trained on uniform samples. Furthermore, even models trained on small coresets provide competitive performance with respect to the ones trained on the full data set, while enabling a significant saving in training time. The reported time includes both sampling and solving times.[4]

9. Conclusion and Future Work

We have shown how to construct coresets for estimating parameters of Gaussian mixture models and natural generalizations by exploiting a connection between statistical estimation and clustering problems in computational geometry. We show theoretical existence of coresets of size *independent* of the original data set size. To our knowledge, our results provide the first rigorous guarantees for obtaining compressed ε-approximations of the log-likelihood of mixture models for large data sets. Our coreset construction is based on a two-step intuitive sampling scheme, and can be easily (and efficiently) implemented. Furthermore, we show that any future improvement in approximation algorithms for the k-means clustering problem can readily be used to improve the speed and/or quality of our methods. We further show how the method can be extended to other mixture models

We demonstrate that, by exploiting certain closure properties of coresets, it is possible to construct them in parallel, or in a single pass through a stream of data, using only $\mathrm{poly}(d, k, \lambda^{-1}, \varepsilon^{-1}, \log n, \log(1/\delta))$ space and update time. Unlike most of the related work, our coresets provide guarantees for any given (possibly unstructured) data, without assumptions on the distribution or model that generated it. While proving the correctness of our construction we also upper bound the pseudo-dimension of the function family of mixtures of Gaussians, for which only a lower bound was known (Akama and Irie, 2011). We empirically demonstrate the practicality of our approach on six real-world data sets. Critically, we show that coresets consistently outperform uniform sampling and enjoy fast convergence (in terms of relative log-likelihood approximation error) to the models trained on the full data set. Empirical evaluation on several real-world datasets suggest that our coreset-based approach enables significant reduction in training-time with negligible approximation error.

Acknowledgments

We thank Olivier Bachem for invaluable discussions, suggestions and comments. This research was partially supported by ONR grant N00014-09-1-1044, NSF grants CNS-0932392, IIS-0953413, DARPA MSEE grant FA8650-11-1-7156, and the Zurich Information Security Center.

4. The algorithms are implemented in Python 2.7 using NumPy and SciPy libraries. The experiments were ran on Intel Xeon 3.3GHz machine with 32 cores and 256GB RAM.

Appendix A. Sampling Complexity

Our goal is to bound the coreset size required to guarantee the coreset property. It is clear that the original data set is a coreset with uniform weights. The question is whether we can obtain arbitrarily good approximation with coresets which are sublinear (or even *independent* of the data set size).

A.1 Sensitivity

For a fixed $z \in \mathbb{N}$ we define $f : \mathcal{X} \times \mathfrak{C} \to \mathbb{R}_+$ as

$$f_\theta(x) = \phi_z(x \mid \theta) = -\ln\left(\sum_{i=1}^{k} w_i \exp\left(-\frac{1}{2}\left\|\Sigma_i^{-1/2}(x-\mu_i)\right\|_2^z\right)\right).$$

As such, $f_\theta(x)$ measures the contribution of the point x to the log-likelihood for a given a parametrization of the mixture model θ – a *query*. In the case of a mixture of Gaussians, the space of possible queries $\mathfrak{C}$ is the space of all possible GMMs with k components in d dimensions. The role of $z \in \mathbb{N}$ is to generalize the model to powers of the Euclidean distance. Note that vector $\theta \in \mathfrak{C}$ represents one GMM. Langberg and Schulman (2010) show that the following quantity plays a critical role in bounding the coreset size.

Definition 8 (Sensitivity) *Let $\mathfrak{C}$ be the set of all queries and $\mathcal{F} = \{f_\theta(\cdot) \mid \theta \in \mathfrak{C}\}$. The sensitivity of $x \in \mathcal{X}$ with respect to $\mathcal{F}$ is defined as*

$$\sigma_{\mathfrak{C}}(x) = \max_{\theta \in \mathfrak{C}} \frac{f_\theta(x)}{\sum_{x \in \mathcal{X}} f_\theta(x)}.$$

The total sensitivity is

$$\mathfrak{S} = \frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} \sigma_{\mathfrak{C}}(x).$$

The exact sensitivity is usually hard to compute, and we will instead find some upper bound $s_{\mathfrak{C}}(x)$ of $\sigma_{\mathfrak{C}}(x), \forall x \in \mathcal{X}$ and use it to upper bound the total sensitivity. By definition, points with large sensitivity contribute more to the log-likelihood for the worst-case queries — not having them in the sample will introduce larger penalties. Hence, the sensitivity of the points may be used to construct an importance sampling strategy that provably reduces the variance of the likelihood estimate.

A.2 Combinatorial Complexity

The other key factor in bounding the coreset size is the combinatorial complexity of the function family $\mathcal{F}$ induced by the maximum likelihood estimation of the Gaussian mixture. The higher the complexity, more samples are necessary to obtain uniform convergence over all queries $\mathfrak{C}$.

Definition 9 (VC dimension) *Let $\mathcal{X}$ be a ground set and $\mathcal{F}$ be a set of functions from $\mathcal{X}$ to $\{0,1\}$. Fix a set $S = \{x_1, \ldots, x_n\} \subset \mathcal{X}$ and a function $f \in \mathcal{F}$. We call $S_f = \{x_i \in S \mid f(x_i) = 1\}$ the induced subset of S by f. A subset $S = \{x_1, \ldots, x_n\}$ of $\mathcal{X}$ is shattered by $\mathcal{F}$ if $|\{S_f \mid f \in \mathcal{F}\}| = 2^n$. VC dimension of $\mathcal{F}$ is the size of the largest subset of $\mathcal{X}$ shattered by $\mathcal{F}$. If $\mathcal{F}$ can shatter sets of arbitrary size VC dimension of $\mathcal{F}$ is ∞.*

These notions naturally extend to functions mapping to $\mathbb{R}$ (or a subset thereof).

Definition 10 (Pseudo-dimension) *Let $\mathcal{X}$ be a ground set and $\mathcal{F}$ be a set of functions from $\mathcal{X}$ to the interval $[0,1]$. Fix a set $S = \{x_1, \ldots, x_n\} \subset \mathcal{X}$, a set of reals $R = \{r_1, \ldots, r_n\}, r_i \in [0,1]$ and a function $f \in \mathcal{F}$. We call $S_f = \{x_i \in S \mid f(x_i) \geq r_i\}$ the induced subset of S formed by f and R. Subset S with associated values R is shattered by $\mathcal{F}$ if $|\{S_f \mid f \in \mathcal{F}\}| = 2^n$. Pseudo-dimension of $\mathcal{F}$ is the cardinality of the largest shattered subset of $\mathcal{X}$. If $\mathcal{F}$ can shatter sets of arbitrary size pseudo-dimension of $\mathcal{F}$ is ∞.*

Clearly, for every space of a given pseudo-dimension we can construct a space with the same VC dimension as formalized by the following lemma.

Lemma 11 (Anthony and Bartlett (2009)) *For any $f \in \mathcal{F}$ let B_f be the indicator function of the region below or on the graph of f, i.e.*

$$B_f(x, y) = \operatorname{sgn}(f(x) - y)$$

The pseudo-dimension of $\mathcal{F}$ is precisely the VC-dimension of the subgraph class

$$B_{\mathcal{F}} = \{B_f \mid f \in \mathcal{F}\}.$$

We refer the reader to Anthony and Bartlett (2009) for an extensive discussion and results on the topic.

A.3 Bounding the Coreset Size

Given the pseudo-dimension $\dim \mathcal{F}$ and some upper bound on the total sensitivity $\mathfrak{S}$ we may bound the coreset size by using the theorem from Bachem et al. (2017b). A bound on the coreset size can also be obtained by applying the theorem of Feldman and Langberg (2011), where one needs to bound the primal shattering dimension instead of the pseudo-dimension. For a detailed discussion of the effects introduced by this difference we refer the reader to Bachem et al. (2017a).

Theorem 12 *Let $\varepsilon > 0$ and $\delta \in (0,1)$. Let $\mathcal{X}$ be a weighted data set, $\mathcal{Q}$ the set of all possible queries and $f_Q(x) : \mathcal{X} \times \mathfrak{C} \to \mathbb{R}_{\geq 0}$ a cost function. Let $s(x) : \mathcal{X} \to \mathbb{R}_{\geq 0}$ denote any upper bound on the sensitivity $\sigma(x)$ and define $S = \frac{1}{n}\sum_{i=1}^{n} s(x)$. Let $\mathcal{C}$ be a sample of m points from $\mathcal{X}$ with replacement where each point $x \in \mathcal{X}$ is sampled with probability $q(x) = \frac{s(x)}{S}$ and each point $x \in \mathcal{C}$ is assigned the weight $\mu_{\mathcal{C}}(x) = \frac{\mu_{\mathcal{X}}(x)}{mq(x)}$. Let $\mathcal{F} = \left\{ \frac{\mu_{\mathcal{X}}(\cdot) f_Q(\cdot)}{\operatorname{cost}(\mathcal{X}, Q) S q(\cdot)} \mid Q \in \mathcal{Q} \right\}$ and $d' = \dim \mathcal{F}$. Then, the set $\mathcal{C}$ is an ε-coreset of $\mathcal{X}$ with probability at least $1 - \delta$ for*

$$m \in \Omega\left(\frac{S^2}{\varepsilon^2}\left(d' + \log\frac{1}{\delta}\right)\right).$$

We now focus on upper-bounding the total sensitivity and pseudo-dimension with the goal of applying Theorem 12.

Appendix B. Reduction to Euclidean Space

In the following lemma we prove a variant of the triangle inequality that we will use to bound the individual contribution of each point to the log-likelihood.

Lemma 6 *For every $a, x \in \mathbb{R}^d, z \in \mathbb{N}$,*

$$\phi_z(x \mid \theta) \leq \frac{2^{z-2}}{\lambda^z} \|x-a\|_2^z + 2^{z-1}\phi_z(a \mid \theta).$$

Proof Let $a, x \in \mathbb{R}^d$. By definition,

$$\phi_z(x \mid \theta) = -\ln\left(\sum_{i=1}^{k} w_i \exp\left(-\frac{1}{2}\left\|\Sigma_i^{-1/2}(x-\mu_i)\right\|_2^z\right)\right). \tag{1}$$

Put $i \in [k]$. By the weak triangle inequality

$$\left\|\Sigma_i^{-1/2}(x-\mu_i)\right\|_2^z \leq 2^{z-1}\left(\left\|\Sigma_i^{-1/2}(x-a)\right\|_2^z + \left\|\Sigma_i^{-1/2}(a-\mu_i)\right\|_2^z\right). \tag{2}$$

Let UDU^T denote the SVD of Σ_i. Since U is a rotation it preserves the L_2 norm. As such,

$$\begin{aligned}\left\|\Sigma_i^{-1/2}(x-a)\right\|_2 &= \left\|UD^{-1/2}U^T(x-a)\right\|_2 = \left\|D^{-1/2}U^T(x-a)\right\|_2 \\ &\leq \frac{\left\|U^T(x-a)\right\|_2}{\lambda} \leq \frac{\|x-a\|_2}{\lambda},\end{aligned}$$

which combined with Equation 2 yields

$$\left\|\Sigma_i^{-1/2}(x-\mu_i)\right\|_2^z \leq \frac{2^{z-1}}{\lambda^z}\cdot\|x-a\|_2^z + 2^{z-1}\left\|\Sigma_i^{-1/2}(a-\mu_i)\right\|_2^z. \tag{3}$$

Substituting Equation 3 in Equation 1 yields

$$\begin{aligned}\phi_z(x \mid \theta) &\leq -\ln\left(\sum_{i=1}^{k} w_i \exp\left(-\frac{2^{z-2}}{\lambda^z}\cdot\|x-a\|_2^z - 2^{z-2}\left\|\Sigma_i^{-1/2}(a-\mu_i)\right\|_2^z\right)\right) \\ &= \frac{2^{z-2}}{\lambda^z}\cdot\|x-a\|_2^z - \ln\left(\sum_{i=1}^{k} w_i \exp\left(-\frac{1}{2}\left\|\Sigma_i^{-1/2}(a-\mu_i)\right\|_2^z\right)^{2^{z-1}}\right).\end{aligned} \tag{4}$$

Since $2^{z-1} \geq 1$, we have that $y^{(2^{z-1})}$ is convex over $y \in [0,1]$, and thus

$$\begin{aligned}&-\ln\left(\sum_{i=1}^{k} w_i \exp\left(-\frac{1}{2}\left\|\Sigma_i^{-1/2}(a-\mu_i)\right\|_2^z\right)^{2^{z-1}}\right) \\ &\leq -\ln\left(\sum_{i=1}^{k} w_i \exp\left(-\frac{1}{2}\left\|\Sigma_i^{-1/2}(a-\mu_i)\right\|_2^z\right)\right)^{2^{z-1}} \\ &= -2^{z-1}\ln\left(\sum_{i=1}^{k} w_i \exp\left(-\frac{1}{2}\left\|\Sigma_i^{-1/2}(a-\mu_i)\right\|_2^z\right)\right) = 2^{z-1}\phi_z(a \mid \theta).\end{aligned}$$

Finally, by applying the last inequality to Equation 4 we complete the proof. ■

B.1 Upper-bound on the Total Sensitivity

We will now introduce a rough approximation to the clustering problem which will be used to bound the sensitivity of each $x \in \mathcal{X}$. Intuitively, the contribution of each point can be bounded by as a function of the contribution of the closest point in the rough approximation. We will prove that it suffices to compute a *bicriteria approximation* for the k-means clustering problem (or a variant of k-means where the Euclidean distance is raised to power $z \in \mathbb{N}$)

Lemma 13 *Let* $\mathrm{cost}_{\mathrm{H}}(\mathcal{X}, C) = \sum_{x \in \mathcal{X}} \min_{c \in C} ||x-c||_2^z$ *and* $OPT = \mathrm{argmin}_{C \in \mathbb{R}^{d \times k}} \mathrm{cost}_{\mathrm{H}}(\mathcal{X}, C)$. *Let* $\mathcal{A}$ *be a* (α, β)*-bicriteria approximation such that* $|\mathcal{A}| = \beta$ *and*

$$\mathrm{cost}_{\mathrm{H}}(\mathcal{X}, \mathcal{A}) \leq \alpha \, \mathrm{cost}_{\mathrm{H}}(\mathcal{X}, OPT).$$

The sensitivity $\sigma_{\mathfrak{C}}(x)$ *of* $x \in \mathcal{X}$ *is bounded by*

$$\sigma_{\mathfrak{C}}(x) \leq s(x) = n \frac{2^{z-1}}{\lambda^{2z}} \left(\frac{\alpha \mathrm{d}^z(x, \mathcal{A})}{\sum_{x' \in \mathcal{X}} \mathrm{d}^z(x', \mathcal{A})} + \frac{\alpha}{|\mathcal{X}_j|} \frac{\sum_{x' \in \mathcal{X}_j} \mathrm{d}^z(x', \mathcal{A})}{\sum_{x' \in \mathcal{X}} \mathrm{d}^z(x', \mathcal{A})} + \frac{1}{|\mathcal{X}_j|} \right).$$

The total sensitivity is bounded by

$$\mathfrak{S} \leq \frac{1}{n} \sum_{f_x \in \mathcal{F}} s(x) = \frac{2^{z-1}}{\lambda^{2z}} (2\alpha + \beta).$$

Proof Fix $i \in [k]$, $x \in \mathcal{X}_j$, $\theta \in \mathfrak{C}$ and let $a \in \mathcal{A}$ such that $\mathrm{d}(x, \mathcal{A}) = \|x - a\|_2$. By Lemma 6,

$$\frac{\phi_z(x \mid \theta)}{\sum_{x' \in \mathcal{X}} \phi_z(x' \mid \theta)} \leq \underbrace{\frac{2^{z-2} \mathrm{d}^z(x, \mathcal{A})}{\sum_{x' \in \mathcal{X}} \phi_z(x' \mid \theta) \lambda^z}}_{\mathfrak{u}} + \underbrace{\frac{2^{z-1} \phi_z(a \mid \theta)}{\sum_{x' \in \mathcal{X}} \phi_z(x' \mid \theta)}}_{\mathfrak{v}}. \tag{5}$$

To bound $\mathfrak{u}$, let $UDU^T = \Sigma_i$ denote the SVD of Σ_i. It follows that

$$\begin{aligned} \left\| \Sigma_i^{-1/2} (x - \mu_i) \right\|_2 &= \left\| U D^{-1/2} U^T (x - \mu_i) \right\|_2 = \left\| D^{-1/2} U^T (x - \mu_i) \right\|_2 \\ &\geq \lambda \left\| U^T (x - \mu_i) \right\|_2 = \lambda \left\| x - \mu_i \right\|_2 \geq \lambda \mathrm{d}(x, \mu). \end{aligned}$$

Hence,

$$\begin{aligned} \phi_z(x \mid \theta) &= -\ln \left(\sum_{i=1}^k w_i \exp \left(-\frac{1}{2} \left\| \Sigma_i^{-\frac{1}{2}} (x - \mu_i) \right\|_2^z \right) \right) \\ &\geq -\ln \left(\sum_{i=1}^k w_i \exp \left(-\frac{\lambda^z}{2} \mathrm{d}^z(x, \mu) \right) \right) \\ &= \frac{\lambda^z}{2} \mathrm{d}^z(x, \mu) \end{aligned}$$

Summing over every $x' \in \mathcal{X}$ yields

$$\sum_{x' \in \mathcal{X}} \phi_z(x' \mid \theta) \geq \frac{\lambda^z}{2} \sum_{x' \in \mathcal{X}} \mathrm{d}^z(x, \mu) \geq \frac{\lambda^z}{2} \min_{C \subset \mathbb{R}^{d \times k}} \sum_{x' \in \mathcal{X}} \mathrm{d}^z(x', C) \geq \frac{\lambda^z}{2\alpha} \sum_{x' \in \mathcal{X}} \mathrm{d}^z(x', \mathcal{A}),$$

where the last inequality follows by definition of $\mathcal{A}$. As such, we can bound $\mathtt{u}$ as

$$\frac{2^{z-2}\mathrm{d}^z(x,\mathcal{A})}{\sum_{x'\in\mathcal{X}}\phi_z(x'\mid\theta)\lambda^z} \leq \frac{\alpha 2^{z-1}\mathrm{d}^z(x,\mathcal{A})}{\lambda^{2z}\sum_{x'\in\mathcal{X}}\mathrm{d}^z(x',\mathcal{A})}. \tag{6}$$

Next we will upper-bound $\mathtt{v}$. By swapping x and b in Lemma 6 we have

$$\phi_z(a\mid\theta) \leq \frac{2^{z-2}}{\lambda^z}\cdot\mathrm{d}^z(x,\mathcal{A}) + 2^{z-1}\phi_z(x\mid\theta).$$

Let $j\in[|\mathcal{A}|]$ such that $x\in\mathcal{X}_j$. Summing the last inequality over every $x'\in\mathcal{X}_j$ yields

$$\begin{aligned}|\mathcal{X}_j|\phi_z(a\mid\theta) &\leq \frac{2^{z-2}}{\lambda^z}\sum_{x'\in\mathcal{X}_j}\mathrm{d}^z(x',\mathcal{A}) + 2^{z-1}\sum_{x'\in\mathcal{X}_j}\phi_z(x'\mid\theta)\\ &\leq \frac{2^{z-2}}{\lambda^z}\sum_{x'\in\mathcal{X}_j}\mathrm{d}^z(x',\mathcal{A}) + 2^{z-1}\sum_{x'\in\mathcal{X}}\phi_z(x'\mid\theta).\end{aligned}$$

Hence,

$$\begin{aligned}\frac{\phi_z(a\mid\theta)}{\sum_{x'\in\mathcal{X}}\phi_z(x'\mid\theta)} &\leq \frac{2^{z-2}}{|\mathcal{X}_j|\lambda^z}\frac{\sum_{x'\in\mathcal{X}_j}\mathrm{d}^z(x',\mathcal{A})}{\sum_{x'\in\mathcal{X}}\phi_z(x'\mid\theta)} + \frac{2^{z-1}}{|\mathcal{X}_j|}\\ &\leq \frac{\alpha 2^{z-1}}{|\mathcal{X}_j|\lambda^{2z}}\frac{\sum_{x'\in\mathcal{X}_j}\mathrm{d}^z(x',\mathcal{A})}{\sum_{x'\in\mathcal{X}}\mathrm{d}^z(x',B)} + \frac{2^{z-1}}{|\mathcal{X}_j|}.\end{aligned} \tag{7}$$

Applying (7) and (6) to (5) yields

$$\begin{aligned}\sigma_{\mathfrak{C}}(x) &\leq n\frac{2^{z-1}}{\lambda^{2z}}\left(\frac{\alpha\mathrm{d}^z(x,\mathcal{A})}{\sum_{x'\in\mathcal{X}}\mathrm{d}^z(x',\mathcal{A})} + \frac{\alpha}{|\mathcal{X}_j|}\frac{\sum_{x'\in\mathcal{X}_j}\mathrm{d}^z(x',\mathcal{A})}{\sum_{x'\in\mathcal{X}}\mathrm{d}^z(x',\mathcal{A})} + \frac{\lambda^{2z}}{|\mathcal{X}_j|}\right)\\ &\leq n\frac{2^{z-1}}{\lambda^{2z}}\left(\frac{\alpha\mathrm{d}^z(x,\mathcal{A})}{\sum_{x'\in\mathcal{X}}\mathrm{d}^z(x',\mathcal{A})} + \frac{\alpha}{|\mathcal{X}_j|}\frac{\sum_{x'\in\mathcal{X}_j}\mathrm{d}^z(x',\mathcal{A})}{\sum_{x'\in\mathcal{X}}\mathrm{d}^z(x',\mathcal{A})} + \frac{1}{|\mathcal{X}_j|}\right)\\ &= s(x),\end{aligned}$$

since the choice of $\theta\in\mathfrak{C}$ was arbitrary and $\lambda\in(0,1)$. The total sensitivity is bounded by

$$\mathfrak{S} \leq \frac{1}{n}\sum_{x\in D}s(x) = \frac{2^{z-1}}{\lambda^{2z}}(2\alpha+\beta).$$

■

B.2 Pseudo-dimension of Mixtures of Gaussians

To upper-bound the pseudo-dimension of the function class $\mathcal{F}$ implied induced by maximum likelihood estimation of a mixture of Gaussians, we will construct a feed-forward neural network which can compute each function from $\mathcal{F}$ and for which we can bound the VC dimension. The following theorem quantifies the VC dimension of feed-forward neural networks that we will employ.

Theorem 14 (Theorem 8.14, Anthony and Bartlett (2009)) *Let h be a function from $\mathbb{R}^m \times \mathbb{R}^d$ to $\{0,1\}$, determining the class*

$$\mathcal{H} = \{h_\theta(\cdot) \mid h_\theta : \mathcal{X} \to \mathbb{R}_+, \theta \in \mathbb{R}^m\}.$$

Suppose that h can be computed by an algorithm that takes as input the pair $(\theta, x) \in \mathbb{R}^m \times \mathbb{R}^d$ and returns $h_\theta(x)$ after no more than t of the following operations:

- *the exponential function $x \mapsto e^x$ on real numbers,*
- *the arithmetic operations $+, -, \times,$ and $/$ on real numbers,*
- *jumps conditioned on $>, \geq, <, \leq, =,$ and $\neq$ comparisons of real numbers, and*
- *output $0, 1$.*

If the t operations include no more than p in which the exponential function is evaluated, then the VC-dimension of $\mathcal{H}$ is $\mathcal{O}\big(m^2p^2 + mp(t + \log mp)\big)$.

Theorem 15 *Let $\theta = [(w_1, \mu_1, \Sigma_1), \ldots, (w_k, \mu_k, \Sigma_k)]$ where $w_i \in \mathbb{R}$, $\mu_i \in \mathbb{R}^d$, and $\Sigma_i \in \mathbb{R}^{d \times d}$ for $i \in [k]$. Let $m = k + kd + kd^2$, $z \leq 2^m \in \mathbb{N}$ and define $f_\theta : \mathcal{X} \to [0, \infty), \theta \in \mathfrak{C} \subset \mathbb{R}^m$ as*

$$f_\theta(x) = -\ln\left(\sum_{i=1}^{k} w_i \exp\left(-\frac{1}{2}\left\|{\Sigma_i}^{-1/2}(x - \mu_i)\right\|_2^z\right)\right).$$

The pseudo-dimension of $\mathcal{F} = \{f_\theta(x) \mid \theta \in \mathbb{R}^m\}$ is $\mathcal{O}\big(d^4k^4\big)$.

Proof For $\theta \in \mathfrak{C}$, $r \in \mathbb{R}$ denote their concatenation by $\theta_r = (\theta, r) \in \mathbb{R}^{m+1}$. Let $h : \mathbb{R}^{m+1} \times \mathbb{R}^d \to \{0,1\}$ such that $h_{\theta,r}(x) = 1$ iff $f_\theta(x) \geq r$ and

$$\mathcal{H} = \left\{h_{\theta,r}(\cdot) \mid \theta_r \in \mathbb{R}^{m+1}\right\}.$$

As shown in Figure 6 where $d_i = -\frac{1}{2}\left\|\Sigma_i^{-1/2}(x - \mu_i)\right\|_2^z$ function $h_{\theta,r}(x)$ can be evaluated using $t = \mathcal{O}(m)$ (since $\log(z) \leq m$) arithmetic operations out of which $\mathcal{O}(k)$ are evaluations of the exponential function. Furthermore, it can be evaluated without using the natural logarithm by directly comparing $e^{f_\theta(x)}$ to e^r. By Theorem 14, fthe VC-dimension of $\mathcal{H}$ is

$$\dim_{\mathrm{VC}} \mathcal{H} = \mathcal{O}\big(m^2k^2 + mk(t + \log mk)\big) = \mathcal{O}\big(k^4d^4\big).$$

By Lemma 11 $\dim \mathcal{F} = \dim_{\mathrm{VC}} \mathcal{H} = \mathcal{O}\big(k^4d^4\big)$ which concludes the proof. ■

The lower-bound of $\Omega(kd^2)$ was established by Akama and Irie (2011). To establish this bound notice that k-means is a special case of the isotropic Gaussian mixture model.

B.3 Main Theorem and Extensions

Now we are ready to present the proof of the Theorem 7 which states that, under natural assumptions, we can uniformly approximate the log-likelihood of the model trained on the coreset and the likelihood of the model trained on the full data set as $\varepsilon \to 0$. We will then show that Corollary 3 follows from Theorem 7. We re-state the theorem for completeness.

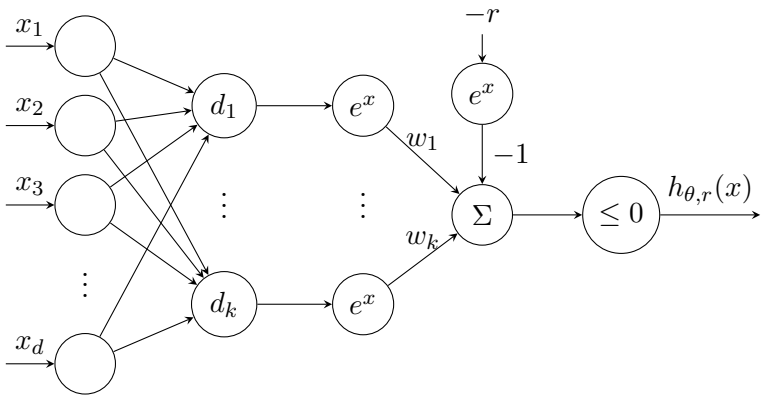

Figure 6: Feed-forward network that calculates $h_{\theta,r}(x)$. The network is parametrized by the Gaussian mixture given by θ and the threshold r. The first hidden layer computes k functions of the form $d_i = -\frac{1}{2}\left\|\Sigma_i^{-1/2}(x-\mu_i)\right\|_2^z$ for a total of $\mathcal{O}(kd^2)$ operations. The second layer evaluates k exponential functions. Finally, the sum of $k+1$ real numbers is compared with zero.

Theorem 7 *Let $\mathcal{X} \subset \mathbb{R}^d$, $|\mathcal{X}| = n$, $\delta \in (0,1)$, $\varepsilon \in (0,1/2)$, $k \geq 1$, $\lambda \in (0,1)$ and $z \in \mathbb{N}$, Let $\mathfrak{C}$ be the collection of all mixtures of k components $\theta = [(w_1,\mu_1,\Sigma_1),\ldots,(w_k,\mu_k,\Sigma_k)]$ such that Σ_i is a $d \times d$ covariance matrix whose singular values are between λ and $1/\lambda$ for every $1 \leq i \leq k$. Consider Algorithm 1 and Algorithm 2 where $\mathrm{d}^2(\cdot,\cdot)$ is replaced by $\mathrm{d}^z(\cdot,\cdot)$. Let $\mathcal{A}$ be the output with smallest quantization error of $\Theta(\ln 1/\delta)$ runs of Algorithm 2 with $\beta = k$. Let $\mathcal{C}$ be the output of Algorithm 1 with $\alpha = 2^{z+2}(\log_2 k + 2)$, bicriteria approximation $\mathcal{A}$ and coreset size*

$$s = \Theta\left(\frac{d^4k^6}{\lambda^{4z}\varepsilon^2}\right).$$

Then, with probability at least $1-\delta$, for all $\theta \in \mathfrak{C}$, it holds that

$$(1-\varepsilon)\phi_z(\mathcal{X} \mid \theta) \leq \phi_z(\mathcal{C} \mid \theta) \leq (1+\varepsilon)\phi_z(\mathcal{X} \mid \theta),$$

where $Z(\theta) = \sum_i \frac{w_i}{g(\theta_i)}$ and

$$\phi_z(\mathcal{X} \mid \theta) = -\sum_{x\in\mathcal{X}} \ln \sum_{i=1}^{k} \frac{w_i}{Z(\theta)g(\theta_i)} \exp\left(-\frac{1}{2}\left\|\Sigma_i^{-1/2}(x-\mu_i)\right\|_2^z\right)$$

using the normalizer

$$g(\theta_i) = \int \exp\left(-\frac{1}{2}\left\|\Sigma_i^{-1/2}(x-\mu_i)\right\|_2^z\right) dx.$$

Proof Fix $\theta = [(w_1,\mu_1,\Sigma_1),\ldots,(w_k,\mu_k,\Sigma_k)] \in \mathfrak{C}$ and let

$$f_\theta(x) = -\ln\left(\sum_{i=1}^{k} w_i \exp\left(-\frac{1}{2}\left\|{\Sigma_i}^{-1/2}(x-\mu_i)\right\|_2^z\right)\right).$$

Let $\mathcal{F} = \{f_\theta(\cdot) \mid \theta \in \mathfrak{C}\}$ and define

$$s(x) = n\frac{2^{z-1}}{\lambda^{2z}}\left(\frac{\alpha \mathrm{d}^z(x,\mathcal{A})}{\sum_{x'\in\mathcal{X}} \mathrm{d}^z(x',\mathcal{A})} + \frac{\alpha}{|\mathcal{X}_j|}\frac{\sum_{x'\in\mathcal{X}_j} \mathrm{d}^z(x',\mathcal{A})}{\sum_{x'\in\mathcal{X}} \mathrm{d}^z(x',\mathcal{A})} + \frac{1}{|\mathcal{X}_j|}\right).$$

Let $\mathrm{cost}_{\mathrm{H}}(\mathcal{X}, C) = \sum_{x\in\mathcal{X}} \min_{c\in C} ||x-c||_2^z$. By Theorem 5.5 of Arthur and Vassilvitskii (2007) any solution computed by Algorithm 2 satisfies $\mathrm{cost}_{\mathrm{H}}(\mathcal{X},\mathcal{A}) \leq \alpha\, \mathrm{cost}_{\mathrm{H}}(\mathcal{X}, OPT)$ in expectation with $\alpha = 2^{z+1}(\log_2 k + 2)$. By Markov's inequality $\mathrm{cost}_{\mathrm{H}}(\mathcal{X},\mathcal{A}) \leq 2\alpha\, \mathrm{cost}_{\mathrm{H}}(\mathcal{X}, OPT)$ with probability at least $1/2$. As a result, the best solution out of $\Theta(\log 1/\delta)$ runs of Algorithm 2 with $\beta = k$ is a $2^{z+2}(\log_2 k + 2)$ approximation with probability at least $1-\delta$. By Lemma 13 we have that

$$s(x) \geq \sigma_{\mathfrak{C}}(x) = \max_{\theta\in\mathfrak{C}} \frac{f_\theta(x)}{\sum_{x'\in\mathcal{X}} f_\theta(x')}.$$

and $\mathfrak{S} \leq \frac{1}{n}\sum_{x\in\mathcal{X}} s(x) = \frac{2^{z-1}}{\lambda^{2z}}(2\alpha+\beta) = \mathcal{O}\Big(k\frac{2^{z-1}}{\lambda^{2z}}\Big)$ since $\beta = k$. Applying Theorem 15 to Theorem 12 completes the proof.

■

Corollary 3 is now straightforward application of the theorem with $z = 2$. As shown in Lucic et al. (2016a), the bound on the total sensitivity is is tight (up to a constant) as there exists a data set $\mathcal{X}$ for which $\mathfrak{S} \in \Theta(k)$.

B.4 Directly Approximating the Log-Likelihood

Under additional assumptions on the singular values we can derive a stronger result directly relating the approximated log-likelihood with the true likelihood.

Theorem 16 *Let the conditions of Theorem 7 hold. If* $\prod_{\lambda_j\in\mathrm{spec}(\Sigma_i)} \lambda_j \geq \frac{1}{(2\pi)^d}$ *we have*

$$|\mathcal{L}(\mathcal{X} \mid \theta) - \mathcal{L}(\mathcal{C} \mid \theta)| \leq \varepsilon \mathcal{L}(\mathcal{X} \mid \theta).$$

Proof By Theorem 7 with $z = 2$, $\alpha = 16(\log_2 k + 2)$, $\beta = k$, $s = \Theta\big(d^4k^6\lambda^{-8}\varepsilon^{-2}\big)$, with probability at least $1-\delta$, it holds that

$$(1-\varepsilon)\phi(\mathcal{X} \mid \theta) \leq \phi(\mathcal{C} \mid \theta) \leq (1+\varepsilon)\phi(\mathcal{X} \mid \theta).$$

By assumption that all eigenvalues are sufficiently large, namely $\prod_{\lambda_j\in\mathrm{spec}(\Sigma_i)} \lambda_j \geq \frac{1}{(2\pi)^d}$, for all components i, the log-normalizer $\ln Z(\theta)$ is negative since

$$Z(\theta) = \sum_i \frac{w_i}{\sqrt{|2\pi\Sigma_i|}} \leq \max_i \frac{1}{\sqrt{|2\pi\Sigma_i|}} = \max_i \frac{1}{\sqrt{(2\pi)^d \prod_{\lambda_j\in\mathrm{spec}(\Sigma_i)} \lambda_j}} \leq 1.$$

Hence,

$$\begin{aligned}\mathcal{L}(\mathcal{C} \mid \theta) &= -n\ln Z(\theta) + \phi(\mathcal{C} \mid \theta) \leq -n\ln Z(\theta) + (1+\varepsilon)\phi(\mathcal{X} \mid \theta) \\ &\leq (1+\varepsilon)\big(-n\ln Z(\theta) + \phi(\mathcal{X} \mid \theta)\big) = (1+\varepsilon)\mathcal{L}(\mathcal{X} \mid \theta),\end{aligned}$$

and similarly,

$$\begin{aligned}\mathcal{L}(\mathcal{C} \mid \theta) &= -n \ln Z(\theta) + \phi(\mathcal{C} \mid \theta) \geq -n \ln Z(\theta) + (1-\varepsilon)\phi(\mathcal{X} \mid \theta) \\ &\geq (1-\varepsilon)\big(-n \ln Z(\theta) + \phi(\mathcal{X} \mid \theta)\big) = (1-\varepsilon)\mathcal{L}(\mathcal{X} \mid \theta).\end{aligned}$$

■

Appendix C. Convergence of Weighted EM for Gaussian Mixtures

Here we present the EM update equations for fitting a weighted set of points using Algorithm 4. Since we are interested in an MLE we begin by stating the necessary conditions for a stationary point of

$$\mathcal{L}(\mathcal{C} \mid \theta) = -n \ln Z(\theta) + \phi(\mathcal{C} \mid \theta) = -\sum_i \gamma_i \ln P(\mathbf{x}'_i \mid \theta).$$

We assume that all covariance matrices are non-singular. Taking the derivative of $\mathcal{L}(\mathcal{C} \mid \theta)$ with respect to μ_i and Σ_i and setting it equal to zero yields

$$\mu_i = \frac{1}{N_i}\sum_{j=1}^{n} \eta_{i,j} x_j \quad \text{and} \quad \Sigma_i = \frac{1}{N_i}\sum_{j=1}^{n} \eta_{i,j}(x_j - \mu_i)(x_j - \mu_i)^T$$

where

$$N_i = \sum_{j=1}^{n} \eta_{i,j}\gamma_j \quad \text{and} \quad \eta_{i,j} = \gamma_j \frac{w_i \mathcal{N}(\mathbf{x}'_j; \mu_i, \Sigma_i)}{\sum_\ell w_\ell \mathcal{N}(\mathbf{x}'_j; \mu_\ell, \Sigma_\ell)}.$$

To find the mixing weights we minimize the negative log-likelihood under the constraint

$$\sum_{i=1}^{k} w_i = 1, w_i \geq 0, i = 1, \ldots, k.$$

To this end we introduce a Lagrange multiplier λ and minimize $\mathcal{L}(\mathcal{C} \mid \theta) + \lambda(\sum_{i=1}^{k} w_i - 1)$. Setting the derivative with respect to w_i to zero yields

$$w_i = \frac{N_i}{\sum_{j=1}^{n} \eta_{i,j}}$$

As expected, the only difference to the non-weighted version of the EM algorithm is that the updates are now scaled proportionally to the weight of each point. As shown in Dempster et al. (1977) this algorithm will converge to a stationary point. A practical concern is that a Gaussian mixture model without a prior can result in a singularity. For example, when only one sample point is assigned to a cluster, the cluster variance is zero, the corresponding density infinite, which in turn implies infinite log-likelihood. The generally accepted remedy is to lower-bound the variances with some apriori chosen value $\lambda > 0$, which we employ in the experiments. In the Bayesian framework, the likelihood maximization problem is replaced by a penalized likelihood function incorporating a prior density on the mixture parameter wherby small values of variances are heavily penalized.

References

Pankaj K Agarwal, Sariel Har-Peled, and Kasturi R Varadarajan. Geometric approximation via coresets. *Combinatorial and Computational Geometry*, 52:1–30, 2005.

Ankit Aggarwal, Amit Deshpande, and Ravi Kannan. Adaptive sampling for k-means clustering. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 15–28. Springer, 2009.

Yohji Akama and Kei Irie. VC dimension of ellipsoids. *arXiv preprint arXiv:1109.4347*, 2011.

Animashree Anandkumar, Daniel J Hsu, and Sham M Kakade. A method of moments for mixture models and hidden markov models. In *Conference On Learning Theory (COLT)*, 2012.

Animashree Anandkumar, Rong Ge, Daniel Hsu, Sham M Kakade, and Matus Telgarsky. Tensor decompositions for learning latent variable models. *Journal of Machine Learning Research (JMLR)*, 15(1):2773–2832, 2014.

Martin Anthony and Peter L Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 2009.

Sanjeev Arora and Ravi Kannan. Learning mixtures of separated nonspherical Gaussians. *Annals of Applied Probability*, 15(1A):69–92, 2005.

David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Symposium on Discrete Algorithms (SODA)*, pages 1027–1035. SIAM, 2007.

Olivier Bachem, Mario Lucic, and Andreas Krause. Coresets for nonparametric estimation - the case of DP-means. In *International Conference on Machine Learning (ICML)*, 2015.

Olivier Bachem, Mario Lucic, S. Hamed Hassani, and Andreas Krause. Approximate k-means++ in sublinear time. In *Proc. Conference on Artificial Intelligence (AAAI)*, 2016a.

Olivier Bachem, Mario Lucic, S. Hamed Hassani, and Andreas Krause. Fast and provably good seedings for k-means. In *Neural Information Processing Systems (NIPS)*, 2016b.

Olivier Bachem, Mario Lucic, and Andreas Krause. Scalable and distributed clustering via lightweight coresets. *arXiv preprint arXiv:1702.08248*, 2017a.

Olivier Bachem, Mario Lucic, and Andreas Krause. Practical coreset constructions for machine learning. *arXiv preprint*, 2017b.

Maria-Florina Balcan, Steven Ehrlich, and Yingyu Liang. Distributed k-means and k-median clustering on general topologies. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1995–2003, 2013.

Pierre Baldi, Peter Sadowski, and Daniel Whiteson. Searching for exotic particles in high-energy physics with Deep learning. *Nature Communications*, 5, 2014.

Mikhail Belkin and Kaushik Sinha. Polynomial learning of distribution families. In *Foundations of Computer Science (FOCS)*, pages 103–112. IEEE, 2010.

Jon Louis Bentley and James B Saxe. Decomposable searching problems I. Static-to-dynamic transformation. *Journal of Algorithms*, 1(4):301–358, 1980.

Artur Czumaj and Christian Sohler. Sublinear-time approximation algorithms for clustering via random sampling. *Random Structures & Algorithms*, 30(1-2):226–256, 2007.

Sanjoy Dasgupta. Learning mixtures of Gaussians. In *Foundations of Computer Science (FOCS)*, pages 634–644. IEEE, 1999.

Sanjoy Dasgupta and Leonard J Schulman. A two-round variant of EM for Gaussian mixtures. In *Uncertainty in Artificial Intelligence (UAI)*, pages 152–159, 2000.

Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. In *Symposium on Operating System Design and Implementation (OSDI)*, 2004.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977.

Matthew Faulkner, Michael Olson, Rishi Chandy, Jonathan Krause, K. Mani Chandy, and Andreas Krause. The next big one: Detecting earthquakes and other rare events from community-based sensors. In *ACM/IEEE International Conference on Information Processing in Sensor Networks*, 2011.

Dan Feldman and Michael Langberg. A unified framework for approximating and clustering data. In *Symposium on Theory of Computing (STOC)*, pages 569–578. ACM, 2011.

Dan Feldman, Amos Fiat, and Micha Sharir. Coresets for weighted facilities and their applications. In *Foundations of Computer Science (FOCS)*, pages 315–324. IEEE, 2006a.

Dan Feldman, Morteza Monemizadeh, and Christian Sohler. A PTAS for k-means clustering based on weak coresets. In *Symposium on Computational Geometry (SoCG)*, pages 11–18. ACM, 2007.

Dan Feldman, Matthew Faulkner, and Andreas Krause. Scalable training of mixture models via coresets. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2142–2150, 2011.

Dan Feldman, Cynthia Sung, and Daniela Rus. The single pixel GPS: learning big data signals from tiny coresets. In *Advances in Geographic Information Systems*, pages 23–32. ACM, 2012.

Dan Feldman, Micha Feigin, and Nir Sochen. Learning big (image) data via coresets for dictionaries. *Journal of Mathematical Imaging and Vision*, 46(3):276–291, 2013a.

Dan Feldman, Melanie Schmidt, and Christian Sohler. Turning big data into tiny data: Constant-size coresets for k-means, PCA and projective clustering. In *Symposium on Discrete Algorithms (SODA)*, pages 1434–1453. SIAM, 2013b.

Jon Feldman, Rocco A Servedio, and Ryan O'Donnell. PAC learning axis-aligned mixtures of Gaussians with no separation assumption. In *Learning Theory*, pages 20–34. Springer, 2006b.

Gereon Frahling and Christian Sohler. Coresets in dynamic geometric data streams. In *Symposium on Theory of Computing (STOC)*, pages 209–217. ACM, 2005.

Sariel Har-Peled and Soham Mazumdar. On coresets for k-means and k-median clustering. In *Symposium on Theory of Computing (STOC)*, pages 291–300. ACM, 2004.

Sariel Har-Peled and Kasturi R Varadarajan. High-dimensional shape fitting in linear time. *Discrete & Computational Geometry*, 32(2):269–288, 2004.

David Haussler. Decision theoretic generalizations of the PAC model for neural net and other learning applications. *Information and Computation*, 100(1):78–150, 1992.

Michael Langberg and Leonard J Schulman. Universal ε-approximators for integrals. In *Symposium on Discrete Algorithms (SODA)*, pages 598–607. SIAM, 2010.

Mario Lucic, Mesrob I. Ohannessian, Amin Karbasi, and Andreas Krause. Tradeoffs for space, time, data and risk in unsupervised learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 663–671, 2015.

Mario Lucic, Olivier Bachem, and Andreas Krause. Linear-time outlier detection via sensitivity. In *Proc. International Joint Conference on Artificial Intelligence (IJCAI)*, 2016a.

Mario Lucic, Olivier Bachem, and Andreas Krause. Strong coresets for hard and soft Bregman clustering with applications to exponential family mixtures. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1–9, 2016b.

Michael W Mahoney and Petros Drineas. CUR matrix decompositions for improved data analysis. *Proceedings of the National Academy of Sciences (PNAS)*, 106(3):697–702, 2009.

Konstantin Makarychev, Yury Makarychev, Maxim Sviridenko, and Justin Ward. A Bi-Criteria Approximation Algorithm for k-Means. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2016)*, volume 60, Dagstuhl, Germany, 2016.

Ankur Moitra and Gregory Valiant. Settling the polynomial learnability of mixtures of Gaussians. In *Foundations of Computer Science (FOCS)*, 2010.

Sashank J Reddi, Barnabás Póczos, and Alex Smola. Communication efficient coresets for empirical loss minimization. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2015.

Guy Rosman, Mikhail Volkov, Dan Feldman, John W Fisher III, and Daniela Rus. Coresets for k-segmentation of streaming data. In *Advances in Neural Information Processing Systems (NIPS)*, pages 559–567, 2014.

Santosh Vempala and Grant Wang. A spectral algorithm for learning mixture models. *Journal of Computer and System Sciences*, 68(4):841–860, 2004.