Clustering Mixture Models in Almost-Linear Time via List-Decodable Mean Estimation

Ilias Diakonikolas
University of Wisconsin, Madison
USA
ilias@cs.wisc.edu

Daniel M. Kane
University of California, San Diego
USA
dakane@cs.ucsd.edu

Daniel Kongsgaard
University of California, San Diego
USA
dkongsga@ucsd.edu

Jerry Li
Microsoft Research
USA
jerrl@microsoft.com

Kevin Tian
Stanford University
USA
kjtian@stanford.edu

ABSTRACT

We study the problem of list-decodable mean estimation, where an adversary can corrupt a majority of the dataset. Specifically, we are given a set T of n points in $\mathbb{R}^d$ and a parameter $0 < \alpha < \frac{1}{2}$ such that an α-fraction of the points in T are i.i.d. samples from a well-behaved distribution $\mathcal{D}$ and the remaining $(1-\alpha)$-fraction are arbitrary. The goal is to output a small list of vectors, at least one of which is close to the mean of $\mathcal{D}$. We develop new algorithms for this problem achieving nearly-optimal statistical guarantees, with runtime $O(n^{1+\epsilon_0}d)$, for any fixed $\epsilon_0 > 0$. All prior algorithms for this problem had additional polynomial factors in $\frac{1}{\alpha}$. We leverage this result, together with additional techniques, to obtain the first *almost-linear time* algorithms for clustering mixtures of k separated well-behaved distributions, nearly-matching the statistical guarantees of spectral methods. Prior clustering algorithms inherently relied on an application of k-PCA, thereby incurring runtimes of $\Omega(ndk)$. This marks the first runtime improvement for this basic statistical problem in nearly two decades.

The starting point of our approach is a novel and simpler near-linear time robust mean estimation algorithm in the $\alpha \to 1$ regime, based on a one-shot matrix multiplicative weights-inspired potential decrease. We crucially leverage this new algorithmic framework in the context of the iterative multi-filtering technique of [41, 45], providing a method to simultaneously cluster and downsample points using *one-dimensional* projections — thus, bypassing the k-PCA subroutines required by prior algorithms.

CCS CONCEPTS

• **Theory of computation → Machine learning theory**.

KEYWORDS

robust statistics; clustering; mixture models; list-decodable learning

STOC '22, June 20–24, 2022, Rome, Italy

ACM ISBN 978-1-4503-9264-8/22/06…$15.00
https://doi.org/10.1145/3519935.3520014

ACM Reference Format:
Ilias Diakonikolas, Daniel M. Kane, Daniel Kongsgaard, Jerry Li, and Kevin Tian. 2022. Clustering Mixture Models in Almost-Linear Time via List-Decodable Mean Estimation. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing (STOC '22), June 20–24, 2022, Rome, Italy.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3519935.3520014

1 INTRODUCTION

We develop novel algorithms achieving almost-optimal runtimes for two closely related fundamental problems in high-dimensional statistical estimation: clustering well-separated mixture models and mean estimation in the list-decodable learning ("majority-outlier") regime. Before we formally state our contributions, we provide the necessary background and motivation for this work.

Clustering Well-Separated Mixture Models. Mixture models are a well-studied class of generative models used widely in practice. Given a family of distributions $\mathcal{F}$, a mixture model $\mathcal{M}$ with k components is specified by k distributions $\mathcal{D}_1, \dots, \mathcal{D}_k \in \mathcal{F}$ and nonnegative mixing weights $\alpha_1, \dots, \alpha_k$ summing to one, and its law is given by $\sum_{i\in[k]} \alpha_i \mathcal{D}_i$. That is, to draw a sample from $\mathcal{M}$, we first choose $i \in [k]$ with probability α_i, and then draw a sample from $\mathcal{D}_i$. When the weights are all equal to $\frac{1}{k}$, we call the mixture *uniform*. Mixture models, especially Gaussian mixture models, have been widely studied in statistics since pioneering work of Pearson in 1894 [76], and more recently, in theoretical computer science [4, 8, 11, 23, 30, 58, 79, 89].

A canonical learning task for mixture models is the *clustering problem*. Given independent samples $\sim \mathcal{M}$, the goal is to approximately recover which samples came from which component. To ensure that this inference task is information-theoretically possible, a common assumption is that $\mathcal{M}$ is "well-separated" and "well-behaved": for example, we may assume each component $\mathcal{D}_i$ is sufficiently concentrated (with sub-Gaussian tails or bounded moments), and that component means have pairwise distance at least Δ, for sufficiently large Δ. The goal is then to efficiently and accurately cluster samples from $\mathcal{M}$ with as small separation as possible.

The prototypical example is the case of uniform mixtures of bounded-covariance Gaussians, of the form $\mathcal{M} = \sum_{i\in[k]} \frac{1}{k}\mathcal{N}(\mu_i, \Sigma_i)$, where each Σ_i is unknown and satisfies $\|\Sigma_i\|_{\mathrm{op}} \le \sigma^2$. Prior to the current work, the fastest known algorithm for this learning problem was due to [4], building on [89]. Notably, [4] gave a polynomial-time

clustering algorithm when $\Delta = \Omega(\sigma\sqrt{k})$.[1] Interestingly, the algorithmic approach of [4, 89] is surprisingly simple and elegant: first, run k-PCA on the set of n samples in $\mathbb{R}^d$ to find a k-dimensional subspace (which can be shown to approximately capture the span of the component means), and then perform a distance-based clustering algorithm in this subspace. The runtime of this algorithm is dominated by $\widetilde{\Omega}(ndk)$ – the cost of (approximate) k-PCA.[2] The idea of using k-PCA as a subroutine to solve the clustering problem is very natural and has also been useful in practice. Indeed, using PCA as a preprocessing step before applying further learning algorithms (such as clustering) is so ubiquitous that it is commonly suggested by introductory textbooks on machine learning, see e.g. [72].

However, in our setting, since the size of the description this problem is $O(nd)$, the runtime of k-PCA is off from linear time by a factor of $\approx k$. In particular, a runtime bottleneck for all (even approximate) k-PCA algorithms in the literature we are aware of for our setting, such as variants of simultaneous power iteration (see [73] and references therein), is a multiplication between an $n \times d$ matrix by a $d \times k$ matrix. In many real-world settings, this factor of k is quite significant. For instance, modern image datasets such as ImageNet [34] often have hundreds or thousands of different classes and subclasses [82]. As a result, many clustering tasks on these datasets often have k of the same order. The resulting overhead would cause many tasks to be infeasible at scale on these datasets. Yet, despite considerable attention over the last two decades,[3] no faster algorithm has been developed for the clustering task. In particular, the runtime of k-PCA has remained a bottleneck in this setting. The preceding discussion motivates the following question.

Question 1. *Can we cluster mixtures of k "well-separated" structured distributions without k-PCA? More ambitiously, is there an algorithm that runs in almost-*linear time*?*

Prior to the current work, this question remained open even for uniform k-mixtures of identity covariance Gaussians with pairwise mean separation as large as $\mathrm{poly}(k)$. In addition to its fundamental interest, a runtime improvement of this sort may have significant practical implications for clustering at scale in real-world applications, see e.g. [75, 91], where spectral methods are commonly used. As our main contribution, *we resolve Question 1 for the general class of mixtures of bounded-covariance distributions under information-theoretically near-optimal separation.*

List-Decodable Mean Estimation. In many statistical settings, including machine learning security [18, 21, 38, 84] and exploratory data analysis e.g. in biology [67, 74, 81], datasets contain arbitrary — and even adversarially chosen — outliers. The central question of the field of robust statistics is to design estimators tolerant to a small amount of unconstrained contamination. Classical work in this field [6, 50, 54, 55, 87, 88] developed robust estimators for many basic tasks, although with computational costs scaling exponentially in the problem dimension. More recently, a line of work starting with [37, 63], developed the first computationally-efficient learning algorithms (attaining near-optimal error) for various estimation problems. Subsequently, there has been significant progress in a variety of settings (see [43] for a survey).

In many of these works, it is typically assumed that the fraction of corrupted points is less than $\frac{1}{2}$. Indeed, when more than half the points are corrupted, the problem is ill-posed: there is not necessarily a uniquely-defined notion of "uncorrupted samples." While outputting a *single* accurate hypothesis in this regime is information-theoretically impossible, one may be able to compute a *small list* of hypotheses with the guarantee that *at least one of them* is accurate. This relaxed notion of estimation is known as *list-decodable learning* [17, 25].

Definition 1 (List-decodable learning)**.** *Given $0 < \alpha < \frac{1}{2}$ and a distribution family $\mathcal{F}$ on $\mathbb{R}^d$, a* list-decodable learning algorithm *takes as input α and a multiset T of n points such that an unknown α fraction of T are independent samples from an unknown distribution $\mathcal{D} \in \mathcal{F}$, and no assumptions are made on the remaining samples. Given T and α, the goal is to output a "small" list of hypotheses at least one of which is close to the target parameter of $\mathcal{D}$.*

Arguably the most fundamental problem in the list-decodable learning setting is mean estimation, wherein the goal is to output a small list of hypotheses, one of which is close to the true mean. A natural problem in its own right, list-decodable mean estimation generalizes the problem of learning well-separated mixture models (as explained below) and can model important applications such as crowdsourcing [69, 85] or semi-random community detection in stochastic block models [25]. Moreover, it is particularly useful in the context of semi-verified learning [25, 69], where a learner can audit a small amount of trusted data. *An important remark is that the parameter α can be quite small in some of these applications and should not necessarily be thought of as a constant.* In addition to applications in clustering mixture models, a concrete example is the crowdsourcing setting with many unreliable responders studied in [69], where the parameter α is tiny, depending inversely-polynomially on problem parameters such as the dimension.

The parameter α in the list-decodable mean estimation setting plays a very similar role to the parameter $\frac{1}{k}$ in learning (uniform) mixture models. This is no coincidence: list-decodable mean estimation can be thought of as a natural robust generalization of clustering well-separated mixtures. Indeed, if we run a list-decodable mean estimation algorithm on a dataset drawn from a uniform mixture of k sufficiently nice and well-separated distributions with $\alpha \leftarrow \frac{1}{k}$, the output list *must* contain a candidate mean which is close to the mean of each component. This is because from the perspective of the list-learning algorithm, each component could be the "true" unknown $\mathcal{D}$, and thus the list must contain a hypothesis close to the mean of this "true" distribution. This small list of hypotheses can then typically be used to cluster the original dataset. One conceptually important implication of this observation is that list-decodable mean estimation algorithms also naturally lead to algorithms for clustering well-separated mixture models (even in

[1] For spherical Gaussians, these works achieve separation $\propto k^{1/4}$, but no extension is known for non-spherical Gaussians or components admitting only moment bounds (the setting considered in this paper).

[2] Throughout the paper, when convenient, we hide polylogarithmic factors in the sample size and algorithm failure probabilities with the $\widetilde{O}$ notation. We reserve the terminology "almost-linear" to mean linear up to subpolynomial factors, and the terminology "nearly-linear" to mean linear up to polylogarithmic factors.

[3] We note that a recent line of work has developed sophisticated polynomial-time clustering algorithms under smaller separation assumptions, see e.g. [45, 52, 61]. These algorithms leverage higher moments of the distribution and consequently require significantly higher sample and computational complexity.

the presence of a small fraction of corrupted samples) — a reduction we formalize.

The first polynomial-time algorithm for list-decodable mean estimation, when $\mathcal{F}$ is the family of bounded-covariance distributions, was by [25]. The [25] algorithm was based on black-box calls to semidefinite program solvers and had a large polynomial runtime. Since then, a sequence of works [29, 41, 42] have obtained substantially improved runtimes for this problem, while retaining the (near-optimal) statistical guarantees of [25]. The algorithm by [42] runs in time $\widetilde{O}(\frac{nd}{\alpha})$ and achieves near-optimal error (within a polylogarithmic factor).

Interestingly, as in the case of clustering mixture models, the $\widetilde{\Omega}(\frac{nd}{\alpha})$ runtime dependence of the [42] algorithm is *also* due to running a k-PCA subroutine — for $k = \Omega(\frac{1}{\alpha})$ — to reduce the problem to a k-dimensional subspace. In more detail, the algorithm of [42] can be viewed as a reduction from list-decodable mean estimation to polylogarithmically many calls to k-PCA (for carefully chosen matrices). Thus, the cost of k-PCA appears as a runtime barrier in state-of-the-art algorithms for list-decodable mean estimation as well. In regimes where α is small, the $\widetilde{\Omega}(\frac{nd}{\alpha})$ runtime is significantly sub-optimal in the input size. This leaves open whether the extraneous linear dependence on α^{-1} is improvable, and brings us to our second main question.

Question 2. *Can we solve list-decodable mean estimation with near-optimal statistical rates in almost-linear time?*

In this paper, *we similarly resolve Question 2 for the class of bounded-covariance distributions.*

1.1 Our Results

We answer both Question 1 and Question 2 in the affirmative, up to subpolynomial factors. Perhaps surprisingly, to resolve the long-standing open problem of clustering mixture models in almost-linear time, we develop an almost-linear time algorithm for the (much more general) problem of list-decodable mean estimation. To then solve the clustering problem, we develop a fast post-processing technique that efficiently reduces the clustering task to list-decodable mean estimation. In light of this development, we begin by presenting our list-decodable estimation result.

Theorem 3. *For any fixed constant $\epsilon_0 > 0$, there is an algorithm* FastMultifilter *with the following guarantee. Let $\mathcal{D}$ be a distribution over $\mathbb{R}^d$ with unknown mean μ^* and unknown covariance Σ with $\|\Sigma\|_{\mathrm{op}} \le \sigma^2$, and let $\alpha \in (0, \frac{1}{2})$. Given α and a multiset of $n = \Omega(\frac{d}{\alpha})$ points on $\mathbb{R}^d$ such that an α-fraction are i.i.d. draws from $\mathcal{D}$,* FastMultifilter *runs in time $O(n^{1+\epsilon_0} d)$ and outputs a list L of $O(\alpha^{-1})$ hypotheses so that with high probability we have*

$$\min_{\hat{\mu} \in L} \left\| \hat{\mu} - \mu^* \right\|_2 = O\left(\frac{\sigma \log \alpha^{-1}}{\sqrt{\alpha}} \right).$$

In the setting of Theorem 3, a sample complexity of $\Omega(\frac{d}{\alpha})$, error of $\Omega(\sigma \alpha^{-\frac{1}{2}})$, and list size $\Omega(\alpha^{-1})$ are all information-theoretically necessary [45]. Hence, up to a $\log(\alpha^{-1})$ factor in the error, Theorem 3 achieves optimal statistical guarantees in almost-linear time.

Leveraging Theorem 3, and combining it with a new almost-linear time post-processing procedure of the resulting list, we achieve our almost-linear runtime for clustering well-separated mixtures under only a second moment bound assumption — even in the presence of a small fraction of outliers. In more detail, our algorithm can tolerate a fraction of outliers proportional to the relative size of the smallest true cluster. For brevity, in this introduction, we will state the natural special case of our clustering result for uniform bounded-covariance mixtures without outliers. We also achieve similar (indeed, slightly stronger) guarantees when the mixture components are sub-Gaussian or have bounded fourth moments.

Theorem 4. *For any fixed constant $\epsilon_0 > 0$, there is an algorithm with the following guarantee. Given a multiset of $n = \Omega(dk)$ i.i.d. samples from a uniform mixture model $\mathcal{M} = \sum_{i \in [k]} \frac{1}{k} \mathcal{D}_i$, where each component $\mathcal{D}_i$ has unknown mean μ_i, unknown covariance matrix Σ_i with $\|\Sigma_i\|_{\mathrm{op}} \le \sigma^2$, and $\min_{i,i' \in [k], i \neq i'} \|\mu_i - \mu_{i'}\|_2 = \widetilde{\Omega}(\sqrt{k})\,\sigma$, the algorithm runs in time $O(n^{1+\epsilon_0} \max(k, d))$, and with high probability correctly clusters 99% of the points.*

Some remarks are in order. First, we note that pairwise mean separation of $\Omega(\sqrt{k}\,\sigma)$ is information-theoretically necessary for accurate clustering to be possible for bounded covariance components. The algorithm establishing Theorem 4 nearly achieves the optimal separation. Secondly, and crucially, our clustering algorithm runs in almost-linear time. Finally, as previously alluded to, our clustering method is robust to outliers, and can handle mixtures with arbitrary weights, with guarantees depending on the smallest weight α (e.g. achieving separation level $\propto \alpha^{-\frac{1}{2}}$).

It is worth commenting on the $\max(k, d)$ term appearing in the running time of Theorem 4. Our algorithm runs in almost-linear time as long as $k \le d$. For the extreme regime where $k \gg d$, our algorithm has running time $O(n^{1+\epsilon_0} k)$. In this parameter regime, it is plausible that $\Omega(nk)$ is a runtime bottleneck for the following reason: even if we are given (exactly) the centers μ_i, $i \in [k]$ for free, $\Omega(nk)$ time seems to be required to simply assign each of the n points to its closest center.

Remark 1. The prior works [4, 11] obtained polynomial-time clustering algorithms with similar statistical guarantees as Theorem 4, under the (much stronger) assumption that each component distribution $\mathcal{D}_i$ has sub-Gaussian tails. For bounded covariance distributions, these algorithms require the stronger mean separation of $\Omega(k\sigma)$ [10]. On the other hand, the clustering methods obtained in [25] (as an application of their list-decodable mean estimator) (i) require sub-Gaussian components, and (ii) partition the dataset into $C \cdot k$ for some constant $C > 2$ — as opposed to k — clusters. In summary, prior work has not explicitly obtained *even a polynomial-time* clustering algorithm in the bounded covariance setting with separation $o(k)\sigma$.

1.2 Technical Overview

We describe the techniques developed in this paper at a high level, and how they circumvent several conceptual runtime barriers encountered by prior approaches to list-decodable mean estimation and clustering mixture models. We assume that the problem "scale" is $\sigma = 1$ for simplicity (e.g. distribution covariances are bounded by $\mathbf{I}$). We first briefly describe two recent fast algorithms for list-decodable mean estimation, developed by [41] and [42], focusing

on tools used in their analyses and bottlenecks in extending their techniques to obtain (almost)-linear runtimes.

Multifiltering. Filtering is one of the most popular techniques for robust estimation [37, 39, 43, 66, 83]. In the minority-outlier setting, filtering is based on the idea of designing certificates of corruption, which either ensure that a current estimate suffices, or can be used to identify a set of points to filter on containing more outliers (corrupted points) than inliers (clean points). Iterating this process terminates in polynomial time, because (roughly speaking) it eventually removes all outliers.

In the context of list-decodable mean estimation, standard filtering guarantees are insufficient, because we cannot afford to remove as many inliers as outliers. To overcome this difficulty, [45] introduced the "multifilter" in the context of Gaussian mean estimation, which was extended to bounded covariance distributions in [41]. At a high level, a multifilter iterates through a tree of candidate subsets, and looks for ways to either "cluster" a subset or "split" it into multiple (overlapping) subsets.[4] To ensure an efficient runtime, a multifilter maintains a potential guaranteeing that the tree size does not blow up (i.e. there are never too many candidate subsets), and carefully chooses to split or cluster based on subset sample statistics, thus ensuring that some tree node always retains a large fraction of inliers. Previous multifilters chose to split or cluster subsets based on *one-dimensional* projections along top eigenvectors of sample covariances, which can be dominated by a single outlier. In the worst case, this leads to an iteration count scaling polynomially with the dimension.

Filtering via Matrix Multiplicative Weights. The approach taken by the fastest algorithms for mean estimation in both majority-inlier [47] and majority-outlier [42] settings is heavily motivated by filtering. In the majority-inlier case, every iteration of the filter is nearly-linear time, so the only bottleneck to an overall fast runtime is the number of iterations. However, simple hard instances show that only projecting onto the worst directions of empirical covariances may lead to an $\Omega(d)$ runtime overhead. The main idea of [47] was to choose scores capturing *multiple* bad directions at a time, preventing this worst-case behavior. These scores were based on quadratic forms with certain trace-one matrices derived from the *matrix multiplicative weights* (MMW) regret minimization framework from semidefinite programming [7, 90]. By using MMW regret bounds, [47] designs a filter that efficiently decreases the empirical covariance operator norm, which is used as a potential to yield convergence in *polylogarithmically* many iterations.

In the majority-outlier setting, the story is somewhat murkier. To overcome complications of prior list-decodable mean estimation algorithms (e.g. the multifilter), which interleaved "filtering" and "clustering" steps, [42] designed a "k-dimensional filter", for $k = \Theta(\frac{1}{\alpha})$, that they called SIFT, decoupling the two goals. Specifically, SIFT uses scores based on k-dimensional projections to hone in on a subspace outside of which the empirical mean is accurate. It then efficiently clusters in just this subspace; combined with appropriate Ky Fan norm generalizations of MMW, the number of iterations is then improved to polylogarithmic. However, this approach of decoupling filtering and clustering appears to inherently use k-dimensional PCA as a subroutine, for $k = \Theta(\frac{1}{\alpha})$, even just to learn an "important" subspace a single time. Hence, this approach encounters a similar runtime bottleneck as prior algorithms for clustering mixture models [4, 89].

Challenges in Combining Techniques. As mentioned, the approach of [42] seems to inherently run into a runtime barrier at $\Omega(\frac{nd}{\alpha})$ due to its reliance on k-PCA. This suggests that to overcome this barrier, we need to develop a new algorithm which both (1) does not disentangle filtering and clustering steps, and (2) relies on univariate projections. It is natural to then try to merge the multifilter with a MMW-based potential to ensure rapid convergence.

Unfortunately, there are several obstacles towards combining these frameworks. A primary complication is that the regret minimization approach of [47] requires multiple consecutive rounds before it can ensure an appropriate potential decreases. This is because of its reliance on MMW, a "mirror descent" algorithm which typically does not provide monotone guarantees on iterates (and hence requires multiple iterations to bound regret) [32]. It is unclear how to make these arguments work within the multifilter framework, which interleaves two types of steps (splitting and clustering) that may have incompatible guarantees across iterations.

Finally, even if it were possible to combine the multifilter with a MMW-based potential analysis, there are still various difficulties towards obtaining an almost-linear runtime coming from the size of our hypothesis tree. For example, making the decision to split or cluster at a node typically requires $\Omega(nd)$ time (e.g. to compute scores), which we cannot afford to perform more than subpolynomially often. This is problematic because our multifilter tree certainly contains $\Omega(\frac{1}{\alpha})$ nodes: in the uniform mixture model case, our tree must contain hypotheses corresponding to every true cluster.

We now overview our techniques in overcoming these obstacles.

One-Shot Potential Framework. In order to deal with the first of the two obstacles discussed (the non-monotonicity of MMW regret guarantees), our starting point is a framework for fast robust mean estimation (cf. Section 2.3 of the full version of this paper), essentially matching the guarantees of [47] with a more transparent analysis. Crucially, our new framework comes with a "one-shot" potential function that shows monotone progress *at every iteration*, making it more amenable to combination with a multifilter (which needs to argue how potentials evolve between different types of steps).

In more detail, our new fast algorithm in the majority-inlier setting guarantees monotone progress on the "Schatten-norm" potential $\mathrm{Tr}(\mathbf{Y}_t^2)$, where $\mathbf{Y}_t := \mathbf{M}_t^{\log(d)}$ and $\mathbf{M}_t = \sum_{i \in T} [w_t]_i (X_i - \mu_t)(X_i - \mu_t)^\top$ is the weighted empirical covariance with respect to the current weight vector w_t. We then use $\mathbf{Y}_t$ to sample carefully chosen Gaussian random vectors to locate outliers in multiple univariate directions. By using the guarantees of Johnson-Lindenstrauss projections, we can use these univariate filters to ensure the next (weighted) empirical covariance matrix satisfies

$$\left\langle \mathbf{Y}_t^2, \mathbf{M}_{t+1} \right\rangle \le O(1)\mathrm{Tr}(\mathbf{Y}_t^2) \, . \tag{1}$$

Combining (1) with a fact from [56] shows that our potential decays geometrically, resulting in rapid convergence. Fortunately, we can use the same potential in the multifilter context, as long as we guarantee that (1) holds for *every* child of a node (whether a split

[4] In [41], these subsets were replaced by weight functions, but the intuition is very similar in both cases.

or cluster step is used). In particular, applying (1) repeatedly for any path in the multifilter tree implies that the depth is polylog(d). It remains to bound the *width* of the tree (the computational cost per layer), while maintaining the invariant that at least one node on every level preserves enough inliers.

Warmup: Fast Gaussian Multifilter via Indicator Weights. Recall that our other obstacle towards an almost-linear runtime is that each of the $\Omega(\alpha^{-1})$ nodes of our multifilter tree requires $\Omega(nd)$ time to decide on a multifiltering step. Our strategy is to reduce the *total number of nodes* across each layer of the tree, so that the total cost of multifiltering on all of them is roughly nd. We achieve this goal by ensuring that our multifilter always maintains nodes which specify subsets of our original data (i.e. 0-1 weights rather than soft weights $\in [0,1]$). Hence, each layer of our new multifilter trades off the *number of subsets* with the *cost of multifiltering* on each subset. Considering the two extreme layers is illustrative of this tradeoff: at the root, our algorithm performs a single one-dimensional projection on the entire dataset; at the leaves, it performs $O(\alpha^{-1})$ one-dimensional projections, each on a subset consisting of roughly an α-fraction of the original dataset.

As a warmup, in this version of the paper, we show how to achieve this in the case where the ground-truth, $\mathcal{D}$, is a bounded-covariance Gaussian, so we can exploit strong concentration bounds. In particular, we know that in any linear projection almost all of the inliers will lie in an interval of logarithmic length. If almost all of our sample points in a subset are clustered within such an interval, we can explicitly remove all samples outside of it. On the other hand, if our samples are spread out, we can split them into two (unweighted) subsets with sufficient overlap to ensure that at least one of the children subsets will contain almost all the inliers, as long as the parent did. We can in fact apply such a partitioning strategy iteratively along each univariate projection, until each remaining subset is contained in a short interval; this suffices to imply (1).

From Gaussians to Bounded-Covariance Distributions. Substantially more technical care is required in the bounded-covariance setting to achieve an almost-linear runtime without sacrificing the error rate. Notably, we will no longer be able to guarantee that the subsets lie in short intervals, due to weaker concentration properties. This also means that we cannot deterministically remove points, making it more challenging to ensure the weight functions we keep are indicators.

We overcome these challenges in Section 4 of the full version of this paper through several new technical developments. We first weaken the outcome guarantee of our recursive partitioning strategy, from ensuring each cluster lies in a short interval, to requiring bounded variance, which we show suffices to advance on the potential (1). Furthermore, we use a randomized dropout strategy in place of the "clustering" step of the multifilter, and design fast quantile checks to ensure the "split" step can be conducted in nearly-constant time. By carefully combining these subroutines, we can indeed ensure every child of nodes in a layer satisfies (1), and that the total computational cost of splitting or clustering on the entire layer is almost-linear. With our earlier depth bound, this yields our full runtime guarantee.

Reducing Clustering to List-Decodable Learning. In Section 5 of the full version of this paper we demonstrate that several mixture model clustering tasks enjoy benefits from the speedups afforded by our list-decodable learning methods. In the following, assume we have a list L of size $O(\alpha^{-1})$ and $L \supseteq \{\hat{\mu}_i\}_{i\in[k]}$ with $\|\hat{\mu}_i - \mu_i\|_2 = \widetilde{O}(\sqrt{\alpha^{-1}})$ for all $i \in [k]$, where μ_i is the mean of the mixture component $\mathcal{D}_i$.

For sub-Gaussian components, we build on a clustering algorithm of [45] and improve it to run in nearly-linear time via randomized distance comparisons. The main idea of the [45] algorithm is to exploit concentration, which implies that with high probability, all points drawn from $\mathcal{D}_i$ have a closest hypothesis in L at distance $\widetilde{O}(\sqrt{\alpha^{-1}})$ from μ_i. By rounding every sample to its nearest hypothesis, and assuming separation $\widetilde{\Omega}(\sqrt{\alpha^{-1}})$ between component means, we can perform an efficient equivalence class partitioning which clusters the data. We observe that this framework is tolerant to a small amount of outliers and generalizes to cluster components with bounded fourth moments.

For our most general application of clustering mixtures under only bounded component covariances, as stated in Theorem 4, the same framework does not apply as a constant fraction of all points may be misbehaved due to weak concentration. To address this, we develop a new postprocessing technique, relying on the following observation: letting $\mathbf{P}$ be the projection onto the $O(\alpha^{-1})$-dimensional subspace spanned by L, any sample hit by $\mathbf{P}$ will lie within distance $O(\sqrt{\alpha^{-1}})$ of its corresponding cluster mean in the low-dimensional subspace with constant probability. We use this observation to drop hypotheses which are too far away from the true means, and then an appropriate equivalence relation suffices for clustering. The runtime bottleneck of this strategy is the computation and application of $\mathbf{P}$ to our dataset, which can be quite expensive. We show that by instead measuring distances in a $O(\log d)$-dimensional subspace formed by random projections within $\mathbf{P}$, and clustering based on these estimates, we obtain similar clustering performance by exploiting guarantees of Johnson-Lindenstrauss transforms.

1.3 Related Work

Mixture Models. The closest line of work to our results studies efficiently clustering mixture models under mean-separation conditions, and in particular Gaussian mixtures [4, 8, 11, 22, 30, 31, 45, 52, 61, 62, 70, 80, 89]. As mentioned previously, within the class of algorithms with runtime $\widetilde{O}(\frac{nd}{\alpha})$, [4] achieves the best known mean separation condition (scaling as $\Theta(\alpha^{-\frac{1}{2}})$, where α is the minimum component weight) for clustering mixtures of Gaussians with bounded covariance. This separation condition is nearly-matched (within a logarithmic factor) by our almost-linear time algorithm (Theorem 4), which in addition is robust to outliers and generalizes to broader distribution families. We note that for the special case where the covariances are all known to be *exactly* the identity, prior to [4], [89] gave a similar algorithm attaining the same runtime of $\widetilde{O}(\frac{nd}{\alpha})$, under a weaker separation condition (scaling as roughly $\alpha^{-\frac{1}{4}}$). We are not aware of algorithms with this runtime and separation for clustering Gaussian mixtures when covariances are only *spectrally bounded by* the identity.

Subsequent work generalized the statistical setting studied in [4, 8, 11, 62], by improving on the separation condition using more

sophisticated algorithmic tools, see, e.g. [44, 45, 52, 61]. More recent work developed efficient algorithms for clustering mixtures of Gaussians, in the presence of a small constant fraction of outliers, under even weaker (algebraic) separation conditions [12, 15, 36]. Beyond clustering, stronger notions of learning have been studied in this setting, including parameter estimation [19, 51, 71], proper learning [2, 9, 33, 48, 64], and their robust analogues [13, 57, 68]. All of the aforementioned algorithms are statistically and computationally intensive, in particular have sample complexities and runtimes scaling super-polynomially with the number of components. Finally, we acknowledge a related line of work studying learning in smoothed settings [5, 20, 49, 53] and density estimation [1, 24, 35], orthogonal to the results of the current paper.

Robust Statistics and List-Decodable Learning. Since pioneering work in the 1960s and 1970s [6, 54, 87, 88], there has been a tremendous amount of work on designing robust estimators, e.g. [50, 55]. However, as discussed earlier, the estimators proposed in the statistics community are intractable to compute in high dimensions. The first algorithmic progress on robust statistics in high dimensions came in two independent works from the theoretical computer science community [37, 63]. Since then, there has been an explosion of work in this area, resulting in computationally efficient estimators for a range of increasingly complex tasks, including the aforementioned work on robust clustering, amongst many others, e.g. [16, 28, 38–40, 46, 60, 77, 86]. For a more complete account, the reader is referred to [43, 66, 83].

We also highlight a line of work, relevant to our main result, which combines tools from robust statistics with ideas from continuous optimization to achieve near-linear runtimes for high-dimensional robust estimation tasks [26, 27, 47, 56, 65]. Importantly, these algorithms only work in the regime where the fraction of corrupted samples is small, i.e. when $\alpha \to 1$.

The list-decodable learning setting we consider (i.e. when the trusted proportion of the data is $\alpha < \frac{1}{2}$) was first considered in [17, 25]. In particular, [25] gave the first polynomial-time algorithm with near-optimal statistical guarantees for the problem of list-decodable mean estimation under bounded covariance assumptions. Thereafter, efficient list-decodable mean estimators with near-optimal error guarantees were developed under stronger distributional assumptions [45, 61]. More recently, a line of work developed list-decodable learners for more challenging tasks, including linear regression [59, 78] and subspace recovery [14, 78]. Similar techniques were also crucial in the recent development of robust clustering algorithms we previously described.

The most directly related prior research to the current paper is the sequence of recent papers developing faster algorithms for list-decodable mean estimation [29, 41, 42]. We note that the algorithms in [29, 42] critically use projection of the data onto a $O(\frac{1}{\alpha})$-dimensional subspace, and therefore are bottlenecked by the cost of this projection, yielding $\Omega(\frac{nd}{\alpha})$ runtimes. In the regime that α is a fixed constant, these works achieve runtimes which are linear in n and d, by reinterpreting the problem of list-decodable mean estimation in a way which is amenable to speedups via tools from continuous optimization (specifically, regret guarantees over the "k-Fantopes", which capture Ky Fan norms in hindsight). On the other hand, the multifilter approach of [41] only uses one-dimensional projections. However, their algorithm and its analysis does not have a direct interpretation as a continuous optimization method, using problem-specific potentials guaranteeing termination after n iterations.

In many ways, the algorithm we develop in this paper can be viewed as the synthesis of these two approaches, by incorporating the ideas of [29, 42] to find better univariate projections for the multifilter of [41], and then designing a new potential function inspired by regret analyses of matrix multiplicative weights to demonstrate rapid termination of our fast multifilter.

2 PRELIMINARIES

General Notation. For mean $\mu \in \mathbb{R}^d$ and positive semidefinite covariance matrix $\Sigma \in \mathbb{R}^{d\times d}$, we let $\mathcal{N}(\mu, \Sigma)$ be the standard multivariate Gaussian. For $d \in \mathbb{N}$ we let $[d] := \{j \mid j \in \mathbb{N}, 1 \le j \le d\}$. We refer to the ℓ_p norm of a vector argument by $\|\cdot\|_p$, and overload this to mean the Schatten-p norm of a symmetric matrix argument. The all-ones vector (when the dimension is clear from context) is denoted $\mathbf{1}$. The (solid) probability simplex is denoted $\Delta^n := \{x \in \mathbb{R}^n_{\ge 0}, \|x\|_1 \le 1\}$. We refer to the i^{th} coordinate of a vector v by $[v]_i$.

Matrices. Matrices will be in boldface throughout, and when the dimension is clear from context we let $\mathbf{0}$ and $\mathbf{I}$ be the zero and identity matrices. The set of $d \times d$ symmetric matrices is $\mathbb{S}^d$ and the $d \times d$ positive semidefinite cone is $\mathbb{S}^d_{\ge 0}$. We use the Loewner partial ordering $\preceq$ on $\mathbb{S}^d_{\ge 0}$. The largest eigenvalue, smallest eigenvalue, and trace of a matrix are given by $\lambda_{\max}(\cdot), \lambda_{\min}(\cdot), \mathrm{Tr}(\cdot)$ respectively. We use $\|\cdot\|_{\mathrm{op}}$ to mean the (ℓ_2-ℓ_2) operator norm, which is the largest eigenvalue for arguments in $\mathbb{S}^d_{\ge 0}$. The inner product on $\mathbf{A}, \mathbf{B} \in \mathbb{S}^d$ is given by $\langle \mathbf{A}, \mathbf{B}\rangle := \mathrm{Tr}(\mathbf{AB})$.

Distributions. We often associate a weight vector $w \in \Delta^n$ with a set of points $T \subset \mathbb{R}^d$ with $|T| = n$. Typically we denote this set by $\{X_i\}_{i\in T}$, where we overload T to mean the indices as well as the points. For any $T' \subseteq T$ we let $w_{T'} \in \Delta^n$ be the vector which agrees with w on T' and is 0 elsewhere. The empirical mean and covariance matrix on any subset are denoted

$$\mu_w(T') := \sum_{i\in T'} \frac{w_i}{\|w'_T\|_1} X_i,$$
$$\mathrm{Cov}_w(T') := \sum_{i\in T'} \frac{w_i}{\|w'_T\|_1} \left(X_i - \mu_w(T')\right)\left(X_i - \mu_w(T')\right)^\top.$$

We also define the unnormalized covariance by

$$\widetilde{\mathrm{Cov}}_w(T') := \sum_{i\in T'} w_i \left(X_i - \mu_w(T')\right)\left(X_i - \mu_w(T')\right)^\top.$$

List-Decodable Mean Estimation. We state the model of list-decodable mean estimation we use throughout the paper; the setting we consider is standard from the literature, and this description is repurposed from [42]. Fix a parameter $0 < \alpha < \frac{1}{2}$. Then a set $T := \{X_i\}_{i\in T} \subset \mathbb{R}^d$ of size $|T| = n = \Theta(d\alpha^{-1})$ is given, containing a subset $S \subset T$ such that the following assumption holds.

Assumption 1. *There is a subset $S \subseteq T \subset \mathbb{R}^d$ of size $\alpha n = \Theta(d)$, and a vector $\mu^* \in \mathbb{R}^d$, such that*

$$\frac{1}{|S|}\sum_{i\in S}(X_i - \mu^*)(X_i - \mu^*)^\top \preceq \mathbf{I}.$$

We remark that this assumption is motivated by the statistical model where there is an underlying distribution $\mathcal{D}$ supported on $\mathbb{R}^d$ with mean μ^* and covariance bounded by $\mathbf{I}$, and the dataset T is formed by αn independent draws from $\mathcal{D}$ combined with $(1-\alpha)n$ arbitrary points. Up to constants in the "good" fraction α and the covariance bound, Proposition B.1 of [25] guarantees Assumption 1 holds with inverse-exponential failure probability. We also note that Proposition 5.4(ii) of [45] shows that the information-theoretic optimal guarantee for list-decodable estimation in the setting of Assumption 1 is to return a list L of candidate means, where $|L| = \Theta(\alpha^{-1})$, and

$$\min_{\mu \in L} \left\|\mu - \mu^*\right\|_2 = \Theta\left(\frac{1}{\sqrt{\alpha}}\right) \tag{2}$$

This handles the more general case where the upper bound matrix in Assumption 1 is $\sigma^2\mathbf{I}$ for some positive parameter σ by rescaling the space appropriately, and the error guarantee (2) becomes worse by a factor of σ.

Finally, throughout we will make the explicit assumption that $d \geq \alpha^{-1}$; for the regime where this is not the case, Algorithm 14 of [42] obtains optimal error rates in time $\widetilde{O}(\alpha^{-2})$ (and in fact, if we tolerate a list size of $O(\alpha^{-1}\log\frac{1}{\delta})$ where $\delta \in (0,1)$ is the failure probability of the algorithm, obtains optimal error in time $\widetilde{O}(\alpha^{-1})$).

Fact 1. *For any $w \in \Delta^n$ associated with $T \subset \mathbb{R}^d$, $\mu_w(T)\mu_w(T)^\top \preceq \sum_{i\in T}\frac{w_i}{\|w\|_1}X_iX_i^\top$. So for any $v \in \mathbb{R}^d$, $(\mu_w(T)-v)(\mu_w(T)-v)^\top \preceq \sum_{i\in T}\frac{w_i}{\|w\|_1}(X_i-v)(X_i-v)^\top$.*

Fact 2. *For any $w \in \Delta^n$ associated with $T \subset \mathbb{R}^d$, and any $v \in \mathbb{R}^d$, $\sum_{i\in T}\frac{w_i}{\|w\|_1}(X_i-v)(X_i-v)^\top = \mathrm{Cov}_w(T)+(\mu_w(T)-v)(\mu_w(T)-v)^\top$.*

Fact 3 (Alternate covariance characterization)**.** *For any $w \in \Delta^n$ associated with $T \subset \mathbb{R}^d$,*

$$\frac{1}{2\|w\|_1^2}\left(\sum_{i,j\in T} w_iw_j(X_i-X_j)(X_i-X_j)^\top\right) = \mathrm{Cov}_w(T).$$

Next, we need the notion of safe weight removal in the list-decodable setting, adapted from [42]. The idea behind safe weight removal is that repeatedly performing a downweighting operation with respect to scores satisfying a certain condition results in weights which preserve some invariant, which we call saturation.

Definition 2. *We call weights $w \in \Delta^n$ γ-saturated, for some $\gamma > 1$, if $w \leq \frac{1}{n}\mathbf{1}$ entrywise, and*

$$\|w_S\|_1 \geq \alpha\|w\|_1^{\frac{1}{\gamma}}.$$

Definition 3. *We call scores $\{s_i\}_{i\in T} \in \mathbb{R}^n_{\geq 0}$ γ-safe with respect to w if $w \in \Delta^n$ and*

$$\sum_{i\in S}\frac{w_i}{\|w_S\|_1}s_i \leq \frac{1}{\gamma}\sum_{i\in T}\frac{w_i}{\|w_T\|_1}s_i.$$

Roughly speaking, we require this alternative notion of safe scores in the majority-outlier regime because there are less good points we can afford to throw away; see [42] for additional exposition. The key property connecting these two definitions is the following.

Lemma 1. *Let $w^{(0)} \in \Delta^n$ be γ-saturated, and consider any algorithm of the form:*

(1) For $0 \leq t < N$:
  (a) Let $\{s_i^{(t)}\}_{i\in T}$ be γ-safe with respect to $w^{(t)}$.
  (b) Update for all $i \in T$:

$$w_i^{(t+1)} \leftarrow \left(1 - \frac{s_i^{(t)}}{s_{\max}^{(t)}}\right)w_i^{(t)}, \text{ where } s_{\max}^{(t)} := \max_{i\in T \mid w_i^{(t)}\neq 0} s_i^{(t)}.$$

Then, $w^{(N)}$ is also γ-saturated.

Finally, we include a technical lemma proved in [29, 42].

Lemma 2 (Lemma 2, [42])**.** *Let $w \in \Delta^n$ have $w \leq \frac{1}{n}\mathbf{1}$ entrywise, and $\|w\|_1 \geq \alpha^2$. Then,*

$$\left\|\mu_w(T) - \mu^*\right\|_2 \leq \sqrt{2\left\|\mathrm{Cov}_w(T)\right\|_{\mathrm{op}}\frac{\|w\|_1}{\|w_S\|_1} + \frac{2}{\alpha}}.$$

In light of the lower bound of [45], Lemma 2 shows to learn the mean near-optimally, it suffices to ensure $\|w_S\|_1 = \Omega(\alpha)$ and $\|\mathrm{Cov}_w(T)\|_{\mathrm{op}} = \widetilde{O}(1)$ (i.e. the weighted covariance is bounded).

We next outline an example of a potential function approach to fast filtering, an alternative to filtering based on matrix multiplicative weights (MMW) used in recent literature [42, 47, 65].[5] This replacement is very useful in the list-decodable setting, as it greatly simplifies the requirements of our fast multifilter which interlaces clustering and filtering steps.

The example problem we consider in this expository section is the minority-outlier regime for robust mean estimation, when the "ground truth" distribution has covariance norm bounded by $\mathbf{I}$. We briefly describe the approach of [47] for this problem, and explain how it can be replaced with our new potential function framework. Throughout fix some $0 < \epsilon < \frac{1}{2}$ and assume that amongst the dataset $T \subset \mathbb{R}^d$ of n points, there is $S \subset T$ of size $|S| = (1-\epsilon)n$ with bounded empirical covariance: $\mathrm{Cov}_{\frac{1}{n}\mathbf{1}}(S) \preceq \mathbf{I}$.

The main algorithmic step in [47] is an efficient subroutine for halving the operator norm of the empirical covariance while filtering more weight from $T \setminus S$ than from S. It is well-known in the literature that whenever the operator norm is $O(1)$, the empirical mean is within $O(\sqrt{\epsilon})$ in ℓ_2 norm from the ground truth mean (for an example of this derivation, see Lemma 3.2 of [47]). This was accomplished in [47] using MMW-based regret guarantees, with "gain matrices" given by covariances and iterative filtering based on MMW responses. The result was a procedure which either terminates with a good estimate, or halves the covariance operator norm after $O(\log d)$ rounds of filtering. The latter is an artifact of many regret minimization techniques, which only guarantee progress after multiple rounds. We now describe an alternative one-shot potential decrease guarantee.

One-Shot Potential Decrease. Our algorithm will proceed in a number of iterations, where we modify a weight vector in Δ^n associated with T. We initialize $w^{(0)} \leftarrow \frac{1}{n}\mathbf{1}$. In iteration t, we will downweight $w^{(t)} \in \Delta^n$ to obtain a new vector $w^{(t+1)}$ as follows. Define the matrices

$$\mathbf{M}_t := \widetilde{\mathrm{Cov}}_{w^{(t)}}(T), \ \mathbf{Y}_t := \mathbf{M}_t^{\log d}.$$

[5] MMW guarantees are also implicitly used in approaches based on packing SDPs, see e.g. [26, 27, 29]. However, [42, 47, 65] use MMW guarantees in a non-black-box way to design filters.

The potential we will track is $\Phi_t := \mathrm{Tr}(\mathbf{Y}_t^2)$. In order to analyze Φ_t, we require two helper facts.

Fact 4 (Lemma 7, [56]). *Let $\mathbf{A} \succeq \mathbf{B}$ be matrices in $\mathbb{S}^d_{\succeq 0}$, and let $p \in \mathbb{N}$. Then $\mathrm{Tr}(\mathbf{A}^{p-1}\mathbf{B}) \geq \mathrm{Tr}(\mathbf{B}^p)$.*

Fact 5. *For any $\gamma \geq 0$ and $\mathbf{A} \in \mathbb{S}^d_{\succeq 0}$, $\gamma \mathrm{Tr}(\mathbf{A}^{2\log d}) \leq \mathrm{Tr}(\mathbf{A}^{2\log d+1}) + d\gamma^{2\log d+1}$.*

We now give the potential analysis. Our main goal will be ensuring

$$\left\langle \mathbf{Y}_t^2, \mathbf{M}_{t+1} \right\rangle \leq 20\mathrm{Tr}(\mathbf{Y}_t^2). \tag{3}$$

The specific constant in the above equation is not particularly important, but is used for illustration. We now show how (3) implies a rapid potential decrease. Observe that

$$\begin{aligned} \Phi_{t+1} &= \mathrm{Tr}\left(\mathbf{M}_{t+1}^{2\log d}\right) \leq \frac{1}{40}\mathrm{Tr}\left(\mathbf{M}_{t+1}^{2\log d+1}\right) + d(40)^{2\log d} \\ &\leq \frac{1}{40}\mathrm{Tr}\left(\mathbf{M}_t^{2\log d}\mathbf{M}_{t+1}\right) + d(40)^{2\log d} \\ &\leq \frac{1}{2}\mathrm{Tr}(\mathbf{Y}_t^2) + d(40)^{2\log d} = \frac{1}{2}\Phi_t + d(40)^{2\log d}. \end{aligned} \tag{4}$$

The first line used Fact 5 with $\gamma = 40$, the second used Fact 4 with $\mathbf{A} = \mathbf{M}_t$ and $\mathbf{B} = \mathbf{M}_{t+1}$ (noting that if $w^{(t+1)} \leq w^{(t)}$ entrywise, the unnormalized covariance matrices respect the Loewner order by Fact 2), and the third line used the assumption (3). This implies that we decrease Φ_t by a constant factor in every iteration, until it is roughly $d(40)^{2\log d}$, at which point the definition $\Phi_t = \mathrm{Tr}(\mathbf{M}_t^{2\log d})$ implies that $\|\mathbf{M}_t\|_{\mathrm{op}}$ is bounded by a constant. By using a naïve filtering preprocessing step, we can guarantee that $\Phi_0 = d^{O(\log d)}$, and hence the process will terminate in $O(\log^2 d)$ rounds.

Meeting the Filter Criterion (3). To complete the outline of this algorithm, we need to explain how to satisfy (3) via downweighting, while ensuring that we remove more weight from $T \setminus S$ than S. To do so, we define scores

$$s_i^{(t)} := (X_i - \mu_{w^{(t)}}(T))^\top \mathbf{M}_t^{2\log d}(X_i - \mu_{w^{(t)}}(T)) \text{ for all } i \in T.$$

Then, by linearity of trace the condition (3) is implied by

$$\sum_{i \in T} w_i^{(t+1)} s_i^{(t)} \leq 20\mathrm{Tr}(\mathbf{Y}_t^2), \tag{5}$$

since Fact 2 implies that $\left\langle \mathbf{Y}_t^2, \mathbf{M}_{t+1} \right\rangle \leq \sum_{i \in T} w_i^{(t+1)} s_i^{(t)}$. Further, whenever (5) does not hold, it must be primarily due to the effect of the outliers $T \setminus S$, because the covariance of S is bounded. Hence, we can simply set $w_i^{(t+1)} = \left(1 - \frac{s_i^{(t)}}{s_{\max}^{(t)}}\right)^K w_i^{(t)}$, where K is the smallest natural number which passes (5). Finally, binary searching to find the smallest value of K meeting (3) yields a complete algorithm running in $\widetilde{O}(nd)$ time (for further details on this binary search, see Theorem 2.4 of [47]).

3 WARMUP: FAST GAUSSIAN MULTIFILTER

As a warmup to our later (stronger) developments in the full version of this paper, we give a complete algorithm for list-decodable mean estimation in the Gaussian case, i.e. where the "true" distribution $\mathcal{D}$ is drawn from a Gaussian with covariance bounded by $\mathbf{I}$. The strength of the error guarantees of the simpler algorithm in this section are somewhat weaker than those of the algorithm in the full version of this paper, but we include this section as an introductory exposition of our techniques.

Throughout this section, we will assume that $\alpha \in [1/d, 1/\log^C d]$, for some constant $C > 0$. We claim this is without loss of generality. Specifically, for α^{-1} sub-logarithmic in the dimension d, the algorithm in the prior work by [42] runs in nearly-linear time. On the other hand, randomly sampling the dataset solves the list-decodable mean estimation problem near-optimally in time $\widetilde{O}(\alpha^{-1})$ (see Appendix A of [42] for a proof). We now formally define the regularity condition which we will use throughout this section.

Assumption 2. *There is a subset $S \subseteq T \subset \mathbb{R}^d$ of size $\alpha n = \Theta(d \cdot \mathrm{polylog}(d))$, and a vector $\mu^* \in \mathbb{R}^d$, such that for all unit vectors $v \in \mathbb{R}^d$ and thresholds $t \in \mathbb{R}_{\geq 0}$,*

$$\Pr_{i \sim_{\mathrm{unif}} S}[\left\langle X_i - \mu^*, v \right\rangle > t] \leq \exp(-\Omega(t^2)) + \frac{1}{\Omega\left(\log^3 d\right)}.$$

Here, the notation $i \sim_{\mathrm{unif}} S$ means that i is a uniformly random sampled index from S.

3.1 Reducing GPartition to GSplitOrCluster

Our final algorithm creates a tree of candidate sets. Every node p in the tree is associated with a subset T_p. In order to progress down the tree, at a given node p we form children $\{c_\ell\}_{\ell \in [k]}$ with associated sets $\{T_{c_\ell}\}_{\ell \in [k]}$; we call the procedure which produces the children node GPartition, and develop it in this section. There are three key properties of GPartition which we need.

(1) The sum of the cardinalities of $\{T_{c_\ell}\}_{\ell \in [k]}$ is not too large compared to $|T_p|$. This is to guarantee that at each layer of the tree, we perform about the same amount of work, namely $\widetilde{O}(nd)$. We formalize this with a parameter $\beta \in (0, 1]$ throughout the rest of this section, and will guarantee that every time GPartition is called on a parent node p,

$$\sum_{\ell \in [k]} \left|T_{c_\ell}\right|^{1+\beta} \leq |T_p|^{1+\beta}. \tag{6}$$

(2) If the parent vertex p has substantial overlap with S (at least $\frac{1}{2}|S|$ points), then at least one of the produced children continues to retain all but a small fraction of points in S.

(3) Defining the matrices

$$\begin{aligned} \mathbf{M}_p &:= \widetilde{\mathrm{Cov}}_{\frac{1}{n}\mathbf{1}}(T_p), \ \mathbf{Y}_p := \mathbf{M}_p^{\log d}, \\ \mathbf{M}_{c_\ell} &:= \widetilde{\mathrm{Cov}}_{\frac{1}{n}\mathbf{1}}(T_{c_\ell}), \ \mathbf{Y}_{c_\ell} := \mathbf{M}_{c_\ell}^{\log d} \text{ for all } \ell \in [k], \end{aligned} \tag{7}$$

every $\mathbf{M}_{c_\ell}$ satisfies the bound

$$\left\langle \mathbf{Y}_p^2, \mathbf{M}_{c_\ell} \right\rangle \leq R^2 \mathrm{Tr}\left(\mathbf{Y}_p^2\right), \tag{8}$$

for some (polylogarithmic) value R we will specify. Note the similarity between this and (3); this will be used in a potential analysis to bound progress on covariance operator norms.

We are now ready to state the algorithm GPartition.

It heavily relies on a subroutine, G1DPartition(T', v) which takes a subset T' and a vector $v \in \mathbb{R}^d$, and produces children subsets of T' satisfying the first two conditions above, and also guarantees that

Algorithm 1 GPartition$(T_p, \alpha, \beta, C, R)$

1: **Input:** $T_p \subseteq T$, $\alpha \in (0, \frac{1}{2})$, $\beta \in (0,1]$, $C, R \in \mathbb{R}_{\geq 0}$ satisfying (for sufficiently large constants)

$$R = \Omega\left(\sqrt{\log(C)} \cdot \frac{\log\log\left(C\alpha^{-1}\right)}{\beta}\right),\ C = \Omega\left(\log^2 d\right).$$

2: **Output:** With failure probability $\leq \frac{1}{d^3}$: subsets $\{T_{c_\ell}\}_{\ell\in[k]}$ of T_p, satisfying (6). Every child satisfies (8) (using notation (7)). If $|T_p \cap S| \geq (\frac{1}{2} + \frac{1}{C})|S|$, at least one child T_{c_ℓ} satisfies

$$\left|T_{c_\ell} \cap S\right| \geq \left|T_p \cap S\right| - \frac{1}{C}|S|. \tag{9}$$

3: Sample $N_{\text{dir}} = \Theta(\log d)$ vectors $\{u_j\}_{j\in[N_{\text{dir}}]} \in \mathbb{R}^d$ each with independent entries ± 1. Following notation (7), let $v_j \leftarrow \mathrm{Y}_p u_j$ for all $j \in [N_{\text{dir}}]$.
4: $\mathcal{S}_0 \leftarrow T_p$
5: **for** $j \in [N_{\text{dir}}]$ **do**
6: $\quad \mathcal{S}_j \leftarrow \emptyset$
7: $\quad$ **for** $T' \in \mathcal{S}_{j-1}$ **do**
8: $\quad\quad \mathcal{T} \leftarrow$ G1DPartition$(T', \alpha, v_j, \beta, CN_{\text{dir}}, R)$
9: $\quad\quad \mathcal{S}_j \leftarrow \mathcal{S}_j \cup \mathcal{T}$
10: $\quad$ **end for**
11: **end for**
12: **return** $\mathcal{S}_{N_{\text{dir}}}$

along the direction v, each child subset is contained in a relatively short interval.

Once again, G1DPartition heavily relies on a subroutine we develop, GSplitOrCluster, which is implemented in Section 3.2. It takes as input a set T'' and either produces one or two subsets of T''. If it outputs one set, that set has length at most $R\|v\|_2$ in the direction v; otherwise, G1DPartition simply recurses on the additional two sets. Crucially, GSplitOrCluster guarantees that if T'' has substantial overlap with S, then so does at least one child; moreover, when GSplitOrCluster returns two sets, they satisfy a size potential such as (10). We now demonstrate correctness of GPartition, assuming that G1DPartition is correct.

Lemma 3. GPartition *satisfies the guarantees in Line 2 of Algorithm 1, assuming correctness of* G1DPartition.

Proof. First, to demonstrate that the subsets satisfy (6), we observe that we can view GPartition as always maintaining a set of subsets, $\mathcal{S}_j$ (in the beginning, $\mathcal{S}_0 = T_p$). The set $\mathcal{S}_j$ is formed by calling G1DPartition on elements of $\mathcal{S}_{j-1}$, each of which satisfy (10), so inductively $\mathcal{S}_{N_{\text{dir}}} = \{T_{c_l}\}_{l\in[k]}$ will satisfy (6) with respect to $\mathcal{S}_0 = T_p$ as desired.

Next, by recursively using the guarantee of G1DPartition, every $T_{c_l} \in \mathcal{S}_{N_{\text{dir}}}$ will satisfy that all values $\left\{\langle v_j, X_i\rangle \mid i \in T_{c_l}\right\}$ are contained in an interval of length $R\left\|v_j\right\|_2$, for all $j \in [N_{\text{dir}}]$. In other words, this set is short along all the directions $\{\mathrm{Y}_p u_j = v_j\}_{j\in[N_{\text{dir}}]}$.

Algorithm 2 G1DPartition$(T', \alpha, v, \beta, C, R)$

1: **Input:** $T' \subseteq T$, $\alpha \in (0, \frac{1}{2})$, $v \in \mathbb{R}^d$, $\beta \in (0,1]$, $C, R \in \mathbb{R}_{\geq 0}$ satisfying (for sufficiently large constants)

$$R = \Omega\left(\sqrt{\log(C)} \cdot \frac{\log\log\left(C\alpha^{-1}\right)}{\beta}\right),\ C = \Omega\left(\log^3 d\right).$$

2: **Output:** Subsets $\{T''_\ell\}_{\ell\in[k]} \subseteq T'$, such that

$$\sum_{\ell\in[k]} \left|T''_\ell\right|^{1+\beta} \leq |T'|^{1+\beta}. \tag{10}$$

If $|T' \cap S| \geq (\frac{1}{2} + \frac{1}{C})|S|$, at least one child T''_ℓ satisfies

$$\left|T''_\ell \cap S\right| \geq \left|T' \cap S\right| - \frac{1}{C}|S|.$$

Every child has all values $\{\langle v, X_i\rangle \mid i \in T''_\ell\}$ contained in an interval of length $R\|v\|_2$.
3: $\mathcal{S}_{\text{in}} \leftarrow \{T'\}$, $\mathcal{S}_{\text{out}} \leftarrow \emptyset$
4: **while** $\mathcal{S}_{\text{in}} \neq \emptyset$ **do**
5: $\quad T'' \leftarrow$ the first element of $\mathcal{S}_{\text{in}}$
6: $\quad \mathcal{S}_{\text{in}} \leftarrow \mathcal{S}_{\text{in}} \setminus T''$
7: $\quad$ **if** GSplitOrCluster$(T'', \alpha, \beta, R, \frac{1}{Cn})$ returns one set $T^{(0)}_{\text{out}}$ **then**
8: $\quad\quad \mathcal{S}_{\text{out}} \leftarrow \mathcal{S}_{\text{out}} \cup \left\{T^{(0)}_{\text{out}}\right\}$
9: $\quad$ **else**
10: $\quad\quad T^{(1)}_{\text{out}}, T^{(2)}_{\text{out}} \leftarrow$ GSplitOrCluster$(T'', \alpha, \beta, R, \frac{1}{Cn})$
11: $\quad\quad \mathcal{S}_{\text{in}} \leftarrow \mathcal{S}_{\text{in}} \cup \left\{T^{(1)}_{\text{out}}, T^{(2)}_{\text{out}}\right\}$
12: $\quad$ **end if**
13: **end while**

This lets us conclude

$$\begin{aligned}
\left\langle \mathrm{Y}_p^2, \mathbf{M}_{c_l}\right\rangle &= \frac{1}{2n\left|T_{c_l}\right|}\left\langle \mathrm{Y}_p^2, \sum_{i,i'\in T_{c_l}} (X_i - X_{i'})(X_i - X_{i'})^\top\right\rangle \\
&= \frac{1}{2n\left|T_{c_l}\right|}\sum_{i,i'\in T_{c_l}} \left\|\mathrm{Y}_p(X_i - X_{i'})\right\|_2^2 \\
&\leq \frac{1.4}{2n\left|T_{c_l}\right|N_{\text{dir}}}\sum_{i,i'\in T_{c_l}}\sum_{j\in[N_{\text{dir}}]} \left\langle \mathrm{Y}_p u_j, X_i - X_{i'}\right\rangle^2 \\
&\leq \frac{1.4}{2n\left|T_{c_l}\right|N_{\text{dir}}}\sum_{i,i'\in T_{c_l}}\sum_{j\in[N_{\text{dir}}]} R^2\left\|\mathrm{Y}_p u_j\right\|_2^2 \\
&\leq \frac{1.4}{2N_{\text{dir}}}\sum_{j\in[N_{\text{dir}}]} R^2\left\|\mathrm{Y}_p u_j\right\|_2^2 \leq R^2\mathrm{Tr}\left(\mathrm{Y}_p^2\right),
\end{aligned}$$

with probability at least $1 - \frac{1}{2d^3}$. Here, we used Fact 3 in the first line and linearity of trace in the second line. The third line used the Johnson-Lindenstrauss lemma of [3] which says that for any vector v, $\frac{1}{N_{\text{dir}}}\sum_{j\in[N_{\text{dir}}]}\left\langle u_j, v\right\rangle^2 \in [0.6, 1.4]\|v\|_2^2$ for a sufficiently large $N_{\text{dir}} = \Theta(\log(d))$ with probability at least $1 - \frac{1}{2d^6}$, which we union bound over all $|T_{c_l}|^2 \leq n^2 \leq d^4$ pairs of points. The fourth line used the radius guarantee of G1DPartition, and the fifth used $|T_{c_l}| \leq n$ and the Johnson-Lindenstrauss lemma guarantee that $\frac{1}{N_{\text{dir}}}\sum_{j\in[N_{\text{dir}}]}\left\|\mathrm{Y}_p u_j\right\|_2^2 \in [0.6, 1.4]\mathrm{Tr}(\mathrm{Y}_p^2)$ with probability at least

$1 - \frac{1}{2d^3}$, which can be deduced by the guarantee of [3] applied to the rows of $\mathbf{Y}_p$. Union bounding over the two applications of [3] yields the claim.

Finally, to demonstrate that at least one child satisfies (9), suppose p satisfies $|T_p \cap S| \geq (\frac{1}{2} + \frac{1}{C})|S|$ (i.e. it has substantial overlap with S). Then by applying the guarantee of G1DPartition inductively, every $\mathcal{S}_j$ will have at least one element T' satisfying $|T' \cap S| \geq \frac{1}{2}|S|$. Every call to G1DPartition only removes $\frac{1}{CN_{\text{dir}}}|S|$ points in S, so overall only $\frac{1}{C}|S|$ points are removed. □

3.2 Implementation of GSplitOrCluster

In this section, we first state GSplitOrCluster and prove correctness, and then analyze the runtime of G1DPartition, using our GSplitOrCluster implementation.

Algorithm 3 GSplitOrCluster$(T_{\text{in}}, \alpha, v, \beta, R, \Delta)$

1: **Input:** $T_{\text{in}} \subseteq T$, $\alpha \in (0, \frac{1}{2})$, $v \in \mathbb{R}^d$, $\beta \in (0, 1]$, $R \in \mathbb{R}_{\geq 0}$, $\Delta \in (0, 1)$

2: **Output:** Either one subset $T^{(0)}_{\text{out}} \subset T_{\text{in}}$, or two subsets $T^{(1)}_{\text{out}}, T^{(2)}_{\text{out}} \subset T_{\text{in}}$. In the one subset case, $T^{(0)}_{\text{out}}$ has $\left\{\langle v, X_i\rangle \mid i \in T^{(0)}_{\text{out}}\right\}$ contained in an interval of length $R\|v\|_2$. In the two subsets case, they take the form, for some threshold value $\tau \in \mathbb{R}$ and $r := \frac{R}{4k_{\max}}, k_{\max} = \Theta\left(\frac{\log\log(\frac{1}{\alpha\Delta})}{\beta}\right)$

$$T^{(1)}_{\text{out}} := \{X_i \mid \langle v, X_i\rangle \leq \tau + r\|v\|_2\},\ T^{(2)}_{\text{out}} := \{X_i \mid \langle v, X_i\rangle \geq \tau - r\|v\|_2\}, \tag{11}$$

and satisfy

$$\left|T^{(1)}_{\text{out}}\right|^{1+\beta} + \left|T^{(2)}_{\text{out}}\right|^{1+\beta} < |T_{\text{in}}|^{1+\beta}. \tag{12}$$

3: $Y_i \leftarrow \langle v, X_i\rangle$ for all $i \in T_{\text{in}}$

4: $T^{(0)}_{\text{out}} \leftarrow$ indices in the middle $1 - \alpha\Delta$ quantiles of $\{Y_i\}_{i \in T_{\text{in}}}$

5: **if** $\left\{Y_i \mid i \in T^{(0)}_{\text{out}}\right\}$ is contained in an interval of length $R\|v\|_2$ **then**

6: **return** $T^{(0)}_{\text{out}}$

7: **else**

8: $\tau_{\text{med}} \leftarrow \text{med}(\{Y_i \mid i \in T_{\text{in}}\})$, where med returns the median

9: $\tau_k \leftarrow \tau_{\text{med}} + 2kr\|v\|_2$ for all integers $-k_{\max} \leq k \leq k_{\max}$

10: **return** $T^{(1)}_{\text{out}}, T^{(2)}_{\text{out}}$ defined in (11) for any threshold τ_k inducing sets satisfying (12)

11: **end if**

To analyze Algorithm 3 we first demonstrate that it always returns in at least one case. In particular, we demonstrate that whenever the set $T^{(0)}_{\text{out}}$ is not sufficiently short, then there will be a threshold parameter k such that the induced sets in (11) satisfy the size bound (12).

Lemma 4. *Suppose Algorithm 3 does not return on Line 6. Then, there exists a $k \in \mathbb{Z}$ in the range $-k_{\max} \leq k \leq k_{\max}$ such that Algorithm 3 is able to return on Line 10.*

Proof. We instead prove that if there is no such k, then we will have a contradiction on the length of the set $T^{(0)}_{\text{out}}$ in the direction v. We first lower bound the length of the $[\frac{1}{2}, 1 - \frac{\alpha\Delta}{2}]$ quantiles of $\{Y_i \mid i \in T_{\text{in}}\}$ by $\frac{1}{2}R\|v\|_2$; the lower bound for the $[\frac{\alpha\Delta}{2}, \frac{1}{2}]$ quantiles will follow analogously. Combining shows that if no threshold works, then the algorithm should have returned $T^{(0)}_{\text{out}}$.

For any threshold τ, define $g(\tau) \in [0, 1]$ to be the proportion of $\{Y_i \mid i \in T_{\text{in}}\}$ which are $\geq \tau$. Moreover, define for all $1 \leq k \leq k_{\max}$,

$$\gamma_k := g(\tau_k - r\|v\|_2) = g(\tau_{\text{med}} + (2k-1)r\|v\|_2),$$

and note that $\gamma_1 \leq \frac{1}{2}$ by definition, since τ_{med} was the median. Now, for each $1 \leq k \leq k_{\max}$, since τ_k was not a valid threshold, the sets

$$T^{(1)}_k := \{X_i \mid Y_i \leq \tau_{\text{med}} + (2k+1)r\|v\|_2\},$$
$$T^{(2)}_k := \{X_i \mid Y_i \geq \tau_{\text{med}} + (2k-1)r\|v\|_2\}$$

do not satisfy the size bound (12). Normalizing both sides of (12) by $|T_{\text{in}}|^{1+\beta}$ and using the definitions of $\{\gamma_k\}$, we obtain the following:

$$(1-\gamma_{k+1})^{1+\beta} + \gamma_k^{1+\beta} = \left(\frac{\left|T^{(1)}_k\right|}{|T_{\text{in}}|}\right)^{1+\beta} + \left(\frac{\left|T^{(2)}_k\right|}{|T_{\text{in}}|}\right)^{1+\beta} \geq 1 \implies \gamma_{k+1} \leq \gamma_k^{1+\beta}. \tag{13}$$

To obtain the above implication, we used $1 - (1-x)^{1+\beta} > x^{1+\beta}$ for all $x, \beta \in [0, 1]$. By repeatedly applying the recursion (13), we have

$$\gamma_{k_{\max}} \leq \gamma_1^{(1+\beta)^{(k_{\max}-1)}} \leq \left(\frac{1}{2}\right)^{(1+\beta)^{(k_{\max}-1)}} \leq \frac{\alpha\Delta}{2},$$

where we use the definition of $k_{\max}$ and $\gamma_1 \leq \frac{1}{2}$. Thus, the $[\frac{1}{2}, 1 - \frac{\alpha\Delta}{2}]$ quantiles are contained between τ_{med} and $\tau_{\text{med}} + (2k_{\max} - 1)r\|v\|_2 \leq \tau_{\text{med}} + \frac{1}{2}R\|v\|_2$. By repeating this argument in the range $-k_{\max} \leq k \leq -1$, we obtain a contradiction (as Algorithm 3 should have returned $T^{(0)}_{\text{out}}$). □

We next prove that if the input T' to G1DPartition has large overlap with S, then the algorithm always returns some child T'' which removes at most $\frac{1}{C}|S|$ points from this overlap.

Lemma 5. *Whenever* G1DPartition *is called on T' with $|T' \cap S| \geq \left(\frac{1}{2} + \frac{1}{C}\right)|S|$ with parameters R, C satisfying (for sufficiently large constants)*

$$R = \Omega\left(\sqrt{\log(C)} \cdot \frac{\log\log(Cd)}{\beta}\right),\ C = \Omega\left(\log^3 d\right),$$

it produces some child T'' satisfying $|T'' \cap S| \geq |T' \cap S| - \frac{1}{C}|S|$.

Proof. We first discuss the structure of G1DPartition. We say a call to GSplitOrCluster is a "split step" if it produces two sets, and otherwise we call it a "cluster step." Every output child of G1DPartition is the result of a consecutive number of split steps, and then one cluster step. Also, every split step replaces an interval with its intersections with two half-lines which overlap by $2r\|v\|_2 = \Omega(\sqrt{\log(C)}\|v\|_2)$. Assume for simplicity that $\|v\|_2 = 1$ in this proof; analogous arguments hold for all v by scaling everything appropriately. Finally, we recall that all calls to GSplitOrCluster in G1DPartition are with $\Delta = \frac{1}{Cn}$.

Our key technical claim is that after any number of split steps forming a partition of the real line, there is always some interval such that $\langle v, \mu^* \rangle$ is r away from both endpoints (in this proof, we allow intervals to have endpoints at $\pm\infty$). This is clearly true at the beginning, since the only interval is $(-\infty, \infty)$. Next, we induct and assume that on the current partition, after some number of split steps, there is an interval $[a, b]$ in the partition such that $\langle v, \mu^* \rangle \in [a + r, b - r]$. Consider the intersection of this interval with any split step, parameterized by the half-lines $(-\infty, \tau + r]$ and $[\tau - r, \infty)$ for some $\tau \in \mathbb{R}$. If $\langle v, \mu^* \rangle \geq \tau$, then one of the resulting intervals is $[\max(a, \tau - r), b]$ where we note that this interval is non-degenerate by assumption; $\tau \leq \langle v, \mu^* \rangle \leq b - r \implies \tau - r \leq b$. If the result of the max is $[a, b]$, then the claim holds; otherwise, the interval is $[\tau - r, b]$ and the claim holds by induction ($\langle v, \mu^* \rangle \leq b - r$) and the assumption $\langle v, \mu^* \rangle \geq \tau$. The other case when $\langle v, \mu^* \rangle \leq \tau$ follows symmetrically by considering the interval $[a, \min(b, \tau + r)]$.

Now, consider the partition of the real line which is induced by the eventual children outputted by G1DPartition, *right before* the last cluster step is applied to them (in other words, this partition is formed only by split steps). Using the above argument, there is some element of this partition $[a, b]$ so that $\langle v, \mu^* \rangle \in [a + r, b - r]$. Applying Assumption 2 shows that if we consider the effects of truncating the set $\{Y_i \mid i \in S\}$ at the endpoints of this interval, we remove at most a $\frac{1}{2C}$ fraction of the points from S. Finally, the interval that is returned is the result of a cluster step applied to this interval. This can only remove at most an $\alpha\Delta \leq \frac{\alpha}{2C}$ fraction of the overall points, which is at most $\frac{1}{2C}|S|$. Combining these two bounds yields the claim. □

Finally, we conclude with a runtime analysis of G1DPartition.

Lemma 6. *Let $n' := |T'|$ for some $T' \subseteq T$.* G1DPartition *called on input T' with parameter C can be implemented to run in time $O\left(n'd + (n')^{1+\beta} \log n' \cdot \frac{\log\log(Cd)}{\beta}\right)$.*

Proof. We begin by forming all of the one-dimensional projections $\langle v, X_i \rangle$ for all $i \in T'$, and sorting these values. We also store the quantile of each point (i.e. the number of points larger than it). The total cost of these operations is $O(n'd + n' \log n')$.

Next, given this total ordering, observe that the structure of our algorithm G1DPartition means that every set in $\mathcal{S}_{\text{in}}$ is a subinterval of T', since this is inductively preserved by calls to GSplitOrCluster; hence, we can represent every set implicitly by its endpoints. Moreover, given access to the initial quantile information we can implement every call to GSplitOrCluster in time $O(k_{\max} \log n') = O(\log n' \cdot \frac{\log\log(Cd)}{\beta})$, since the cost of checking the length of $T_{\text{out}}^{(0)}$ is constant, and the cost of checking each candidiate τ_k is dominated by determining the thresholds of the corresponding induced sets $T_{\text{out}}^{(1)}$ and $T_{\text{out}}^{(2)}$. These can be performed via binary searches in $O(\log n')$ time.

It remains to bound the number of calls to GSplitOrCluster throughout the execution of G1DPartition. To this end, we bound the number of times GSplitOrCluster can return one set, and the number of times it can return two sets. Every time GSplitOrCluster returns one set, it adds it to $\mathcal{S}_{\text{out}}$, and by using the guarantee (12) recursively, there can only ever be $(n')^{1+\beta}$ such sets. Similarly, every time it returns two sets it increases $|\mathcal{S}_{\text{in}}| + |\mathcal{S}_{\text{out}}|$ by one, but we know at termination this is at most $(n')^{1+\beta}$, and this potential never decreases. Thus, the total number of calls to GSplitOrCluster is bounded by $O((n')^{1+\beta})$, as desired. □

We obtain a runtime bound on GPartition as a corollary.

Corollary 1. *Let $n_p := |T_p|$ for some $T_p \subseteq T$.* GPartition *called on input T_p with parameter C can be implemented to run in time $O\left(n_p^{1+\beta} d \log^2(d) + n_p^{1+\beta} \log^2(d) \cdot \frac{\log\log(Cd))}{\beta}\right)$.*

Proof. First, consider the cost of computing all vectors $\mathbf{Y}_p u_j$. It is straightforward to implement matrix-vector multiplications through $\mathbf{M}_p$ in time $O(n_p d)$, so this cost is $O(n_p d \log^2(d))$.

We next require a bound on the cost of $N_{\text{dir}} = \Theta(\log d)$ consecutive calls to G1DPartition. The cost of each is given by Lemma 6, and the result follows by summing this cost over all elements of each $\mathcal{S}_j$, which can be bounded since for all $j \in [N_{\text{dir}}]$, the cardinalities of all sets contained in $\mathcal{S}_j$ have $1 + \beta$ powers bounded by $n_p^{1+\beta}$ by repeatedly using the guarantee (10). □

3.3 Full Gaussian Algorithm

Finally, we are ready to give our full (warmup) algorithm for list-decodable mean estimation under Assumptions 1 and 2.

Algorithm 4 FastGMultifilter(T, α)

1: **Input:** $T \subset \mathbb{R}^d$, $|T| = n$ satisfying Assumptions 1 and 2 with parameter $\alpha \in (0, \frac{1}{2})$
2: **Output:** With failure probability $\leq \frac{1}{d}$: L with $|L| = O(\frac{1}{\alpha})$ such that some $\hat{\mu} \in L$ satisfies

$$\left\|\hat{\mu} - \mu^*\right\|_2 = O\left(\frac{\log(d) \log\log^{1.5}(d)}{\sqrt{\alpha}}\right). \tag{14}$$

3: $\{T_i'\}_{i \in [k]} \leftarrow$ NaiveCluster(T)
4: $\alpha_i \leftarrow \frac{|T|}{|T_i'|}\alpha$ for all $i \in [k]$
5: **return** $\bigcup_{i \in [k]}$ FastGMultifilterBoundedDiameter(T_i', α_i)

We begin by stating the guarantees of NaiveCluster.

Lemma 7 (Lemma 12, [42])**.** *There is a randomized algorithm,* NaiveCluster*, which takes as input $T \subset \mathbb{R}^d$ satisfying Assumption 1 and partitions it into disjoint subsets $\{T_i'\}_{i \in [k]}$ such that with probability at least $1 - \frac{1}{d^2}$, all of S is contained in the same subset, and every subset has diameter bounded by $O(d^{12})$, in time $O(nd + n \log n)$.*

We next demonstrate that if the operator norm of the (unnormalized) covariance matrix of a set of points T' is bounded, and T' has sufficient overlap with S, then its empirical mean is close to μ^*.

Lemma 8. *For $T' \subset T$ with empirical mean $\hat{\mu}$, if $|T' \cap S| \geq \frac{1}{2}|S|$ and $\widetilde{\text{Cov}}_{\frac{1}{n}\mathbf{1}}(T') \leq R^2$, $\|\hat{\mu} - \mu^*\|_2 = O\left((1 + R) \cdot \frac{1}{\sqrt{\alpha}}\right)$.*

Proof. Let w place weight $\frac{1}{n}$ on coordinates in T', and 0 on all other coordinates. Clearly this w satisfies the assumption of Lemma 2, since its ℓ_1 norm is simply $\frac{|T'|}{|T|} \geq \frac{1}{2}\alpha$. The conclusion follows by applying Lemma 2, where we use $\|w\|_1 \text{Cov}_w(T) = \widetilde{\text{Cov}}_{\frac{1}{n}\mathbf{1}}(T')$, and the assumed bound. □

Algorithm 5 FastGMultifilterBoundedDiameter(T, α)

Input: $T \subset \mathbb{R}^d$, $|T| = n$ satisfying Assumptions 1 and 2 with parameter $\alpha \in (0, \frac{1}{2})$

Output: With failure probability $\leq \frac{1}{d}$: L_{out} with $|L_{\text{out}}| = O(\frac{1}{\alpha})$ such that some $\hat{\mu} \in L_{\text{out}}$ satisfies

$$\left\|\hat{\mu} - \mu^*\right\|_2 = O\left(\frac{\log(d)\log\log^{1.5}(d)}{\sqrt{\alpha}}\right).$$

$L^{(0)} \leftarrow \{T\}$, $L_{\text{out}} \leftarrow \emptyset$

For sufficiently large constants,

$$R \leftarrow \Theta\left(\log(d)\log\log^{1.5}(d)\right),\ C \leftarrow \Theta(\log^2 d),\ D \leftarrow \Theta(\log^2 d)$$

for $\ell \in [D]$ **do**

  $L^{(\ell)} \leftarrow \emptyset$

  for $T' \in L^{(\ell-1)}$ **do**

    Append all elements of GPartition$(T', \alpha, \frac{1}{\log d}, C, R)$ to $L^{(\ell)}$ with size at least $\frac{\alpha n}{2}$

  end for

end for

return List of empirical means of all sets in $L^{(D)}$

We give a full analysis of FastGMultifilterBoundedDiameter.

Proposition 1. FastGMultifilterBoundedDiameter *meets its output specifications with probability at least* $1 - \frac{1}{d}$*, within runtime*

$$O\left(nd\log^4(d) + n\log^5(d)\log\log(d)\right).$$

Proof. Throughout, we denote $\beta := \frac{1}{\log d}$. There are three main guarantees of the algorithm: that the list size is $O(\frac{1}{\alpha})$, that some list element satisfies (14), and that the runtime is as claimed. We first bound the list size. FastGMultifilterBoundedDiameter produces a tree of subsets, of depth D. Each layer of the tree is composed by the sets in $L^{(\ell)}$ where $0 \leq \ell \leq D$, and $L^{(0)}$ is the root node. The children of each node are the results of calling GPartition on the associated subset. Moreover, by repeatedly using the guarantee (6) inductively, the total cardinality of all sets at layer ℓ is bounded by $n^{1+\beta} = O(n)$. Since we only return means from sets with size at least $\frac{\alpha n}{2}$ on layer D, there can only be $O(\frac{1}{\alpha})$ such sets.

Next, we bound error rate. Consider some leaf node, and its path to the root; call the sets associated with these vertices $T_0, T_1, \ldots T_D$, where T_D is the leaf node and $T_0 = T$ is the original set. Define the potential function at each layer $0 \leq \ell \leq D$,

$$\Phi_\ell := \mathrm{Tr}\left(\mathbf{M}_\ell^{2\log d}\right),\ \text{where } \mathbf{M}_\ell := \widetilde{\mathrm{Cov}}_{\frac{1}{n}\mathbf{1}}(T_\ell).$$

Note that every parent-child pair along this path satisfies the guarantee (8). We thus conclude that for each $0 \leq \ell < D$, we have the recurrence (analogously to (4))

$$\begin{aligned}\Phi_{\ell+1} = \mathrm{Tr}\left(\mathbf{M}_{\ell+1}^{2\log d}\right) &\leq \frac{1}{2R^2}\mathrm{Tr}\left(\mathbf{M}_{\ell+1}^{2\log d+1}\right) + d(2R^2)^{2\log d} \\ &\leq \frac{1}{2R^2}\mathrm{Tr}\left(\mathbf{M}_\ell^{2\log d}\mathbf{M}_{\ell+1}\right) + d(2R^2)^{2\log d} \\ &\leq \frac{1}{2}\mathrm{Tr}\left(\mathbf{M}_\ell^{2\log d}\right) + d(2R^2)^{2\log d} = \frac{1}{2}\Phi_\ell + d(2R^2)^{2\log d}.\end{aligned}$$

The first line used Fact 5 with $\gamma = 2R^2$, the second used Fact 4, and the third used the guarantee (8). Thus, as long as at a layer ℓ we have $\Phi_\ell > 4d(2R^2)^{2\log d}$, we have $\Phi_{\ell+1} \leq \frac{3}{4}\Phi_\ell$, and so the potential is decreasing by a constant factor. The potential Φ_0 is bounded by $d^{O(\log d)}$, because we assumed the input set has polynomially bounded diameter, so within $D = \Omega(\log^2 d)$ layers, every node on layer D must have $\Phi_D \leq 4d(2R^2)^{2\log d}$. This implies that the operator norm of $\widetilde{\mathrm{Cov}}_{\frac{1}{n}\mathbf{1}}(T')$ for every node T' on layer D is $O(R^2)$.

We next show at least one node T' on every layer has $|T' \cap S| \geq \frac{1}{2}|S|$. By inductively using (9) with our chosen value of C, summing over the $O(\log^2 d)$ layers guarantees that we only remove at most $\frac{1}{2}|S|$ points from the intersection throughout the root-to-leaf path, for some path. We can now apply Lemma 8 to guarantee (14). To obtain the high-probability bound, note that the number of times we call GPartition is bounded by $O(\frac{1}{\alpha}\log^2 d)$, since at each layer we prune every node with less than $\frac{\alpha n}{2}$ points; there can only be $O(\frac{1}{\alpha})$ surviving nodes per layer (since the total cardinalities of the layer is bounded by $n^{1+\beta} = O(n)$), and taking a union bound over all calls to GPartition shows the failure probability.

Finally, we discuss runtime. We simply apply Corollary 1 to each layer, which bounds the runtime of each layer by $O(nd\log(d) + n\log^3(d)\log\log(d))$, since the sets on that layer satisfy (6) inductively. Summing over all layers yields the desired guarantee. □

Theorem 5. FastGMultifilter *meets its output specifications with probability at least* $1 - \frac{1}{d}$*, within runtime*

$$O\left(nd\log^4(d) + n\log^5(d)\log\log(d)\right).$$

Proof. We apply Proposition 1 to the relevant call of the algorithm FastGMultifilterBoundedDiameter. Note that all $\alpha_i \geq \alpha$, giving the error guarantee (14), and $\sum_{i\in[k]}\frac{1}{\alpha_i} = \frac{1}{\alpha}$, giving the list size bound. The runtime follows from $\sum_{i\in[k]}|T_i'| = |T|$. □

ACKNOWLEDGEMENTS

Ilias Diakonikolas is supported by NSF Medium Award CCF-2107079, NSF Award CCF-1652862 (CAREER), a Sloan Research Fellowship, and a DARPA Learning with Less Labels (LwLL) grant. Daniel Kane is supported by NSF Medium Award CCF-2107547, NSF CAREER Award ID 1553288, and a Sloan fellowship. Kevin Tian is supported by a Google Ph.D. Fellowship, a Simons-Berkeley VMware Research Fellowship, NSF CAREER Award CCF-1844855, NSF Grant CCF-1955039, and the Alfred P. Sloan Foundation.

REFERENCES

[1] Jayadev Acharya, Ilias Diakonikolas, Jerry Li, and Ludwig Schmidt. 2017. Sample-optimal density estimation in nearly-linear time. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 1278–1289.

[2] Jayadev Acharya, Ashkan Jafarpour, Alon Orlitsky, and Ananda Theertha Suresh. 2014. Near-optimal-sample estimators for spherical gaussian mixtures. *arXiv preprint arXiv:1402.4746* (2014).

[3] Dimitris Achlioptas. 2003. Database-friendly random projections: Johnson-Lindenstrauss with binary coins. *J. Comput. Syst. Sci.* 66, 4 (2003), 671–687.

[4] Dimitris Achlioptas and Frank McSherry. 2005. On spectral learning of mixtures of distributions. In *International Conference on Computational Learning Theory*. Springer, 458–469.

[5] Joseph Anderson, Mikhail Belkin, Navin Goyal, Luis Rademacher, and James Voss. 2014. The more, the merrier: the blessing of dimensionality for learning large gaussian mixtures. In *Conference on Learning Theory*. PMLR, 1135–1164.

[6] Frank J Anscombe. 1960. Rejection of outliers. *Technometrics* 2, 2 (1960), 123–146.

[7] Sanjeev Arora and Satyen Kale. 2007. A combinatorial, primal-dual approach to semidefinite programs. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*. 227–236.

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

[9] Hassan Ashtiani, Shai Ben-David, Nicholas JA Harvey, Christopher Liaw, Abbas Mehrabian, and Yaniv Plan. 2018. Nearly tight sample complexity bounds for learning mixtures of Gaussians via sample compression schemes. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. 3416–3425.

[10] Pranjal Awasthi. 2021. (September 2021). Personal communication.

[11] Pranjal Awasthi and Or Sheffet. 2012. Improved spectral-norm bounds for clustering. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Springer, 37–49.

[12] Ainesh Bakshi, Ilias Diakonikolas, Samuel B. Hopkins, Daniel Kane, Sushrut Karmalkar, and Pravesh K. Kothari. 2020. Outlier-Robust Clustering of Gaussians and Other Non-Spherical Mixtures. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020*. IEEE, 149–159.

[13] Ainesh Bakshi, Ilias Diakonikolas, He Jia, Daniel M Kane, Pravesh K Kothari, and Santosh S Vempala. 2020. Robustly Learning Mixtures of k Arbitrary Gaussians. *arXiv preprint arXiv:2012.02119* (2020).

[14] Ainesh Bakshi and Pravesh Kothari. 2020. List-Decodable Subspace Recovery via Sum-of-Squares. *arXiv preprint arXiv:2002.05139* (2020).

[15] Ainesh Bakshi and Pravesh Kothari. 2020. Outlier-robust clustering of non-spherical mixtures. *arXiv preprint arXiv:2005.02970* (2020).

[16] Sivaraman Balakrishnan, Simon S Du, Jerry Li, and Aarti Singh. 2017. Computationally efficient robust sparse estimation in high dimensions. In *Conference on Learning Theory*. 169–212.

[17] Maria-Florina Balcan, Avrim Blum, and Santosh Vempala. 2008. A discriminative framework for clustering via similarity functions. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*. 671–680.

[18] Marco Barreno, Blaine Nelson, Anthony D. Joseph, and J. D. Tygar. 2010. The security of machine learning. *Mach. Learn.* 81, 2 (2010), 121–148.

[19] Mikhail Belkin and Kaushik Sinha. 2015. Polynomial learning of distribution families. *SIAM J. Comput.* 44, 4 (2015), 889–911.

[20] Aditya Bhaskara, Moses Charikar, Ankur Moitra, and Aravindan Vijayaraghavan. 2014. Smoothed analysis of tensor decompositions. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*. 594–603.

[21] Battista Biggio, Blaine Nelson, and Pavel Laskov. 2012. Poisoning Attacks against Support Vector Machines. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*.

[22] S. Charles Brubaker. 2009. Robust PCA and clustering in noisy mixtures. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009, New York, NY, USA, January 4-6, 2009*, Claire Mathieu (Ed.). SIAM, 1078–1087.

[23] S. Charles Brubaker and Santosh S. Vempala. 2008. Isotropic PCA and Affine-Invariant Clustering. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*. 551–560.

[24] Siu-On Chan, Ilias Diakonikolas, Rocco A Servedio, and Xiaorui Sun. 2014. Efficient density estimation via piecewise polynomial approximation. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*. 604–613.

[25] Moses Charikar, Jacob Steinhardt, and Gregory Valiant. 2017. Learning from untrusted data. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*. 47–60.

[26] Yu Cheng, Ilias Diakonikolas, and Rong Ge. 2019. High-dimensional robust mean estimation in nearly-linear time. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2755–2771.

[27] Yu Cheng, Ilias Diakonikolas, Rong Ge, and David P Woodruff. 2019. Faster Algorithms for High-Dimensional Robust Covariance Estimation. In *Conference on Learning Theory*. 727–757.

[28] Yu Cheng, Ilias Diakonikolas, Daniel Kane, and Alistair Stewart. 2018. Robust learning of fixed-structure bayesian networks. In *Advances in Neural Information Processing Systems*. 10283–10295.

[29] Yeshwanth Cherapanamjeri, Sidhanth Mohanty, and Morris Yau. 2020. List Decodable Mean Estimation in Nearly Linear Time. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020*.

[30] Sanjoy Dasgupta. 1999. Learning mixtures of Gaussians. In *40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039)*. IEEE, 634–644.

[31] Sanjoy Dasgupta and Leonard Schulman. 2007. A probabilistic analysis of EM for mixtures of separated, spherical Gaussians. *Journal of Machine Learning Research* 8, Feb (2007), 203–226.

[32] Constantinos Daskalakis, Andrew Ilyas, Vasilis Syrgkanis, and Haoyang Zeng. 2018. Training GANs with Optimism. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

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

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

[35] Luc Devroye and Gábor Lugosi. 2012. *Combinatorial methods in density estimation*. Springer Science & Business Media.

[36] Ilias Diakonikolas, Samuel B Hopkins, Daniel Kane, and Sushrut Karmalkar. 2020. Robustly learning any clusterable mixture of gaussians. *arXiv preprint arXiv:2005.06417* (2020).

[37] Ilias Diakonikolas, Gautam Kamath, Daniel Kane, Jerry Li, Ankur Moitra, and Alistair Stewart. 2019. Robust estimators in high-dimensions without the computational intractability. *SIAM J. Comput.* 48, 2 (2019), 742–864.

[38] Ilias Diakonikolas, Gautam Kamath, Daniel Kane, Jerry Li, Jacob Steinhardt, and Alistair Stewart. 2019. Sever: A robust meta-algorithm for stochastic optimization. In *International Conference on Machine Learning*. 1596–1606.

[39] Ilias Diakonikolas, Gautam Kamath, Daniel M. Kane, Jerry Li, Ankur Moitra, and Alistair Stewart. 2017. Being Robust (in High Dimensions) Can Be Practical. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*. 999–1008.

[40] Ilias Diakonikolas, Daniel Kane, Sushrut Karmalkar, Eric Price, and Alistair Stewart. 2019. Outlier-Robust High-Dimensional Sparse Estimation via Iterative Filtering. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019*, Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (Eds.). 10688–10699. http://papers.nips.cc/paper/9253-outlier-robust-high-dimensional-sparse-estimation-via-iterative-filtering

[41] Ilias Diakonikolas, Daniel Kane, and Daniel Kongsgaard. 2020. List-Decodable Mean Estimation via Iterative Multi-Filtering. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

[42] Ilias Diakonikolas, Daniel Kane, Daniel Kongsgaard, Jerry Li, and Kevin Tian. 2021. List-Decodable Mean Estimation in Nearly-PCA Time. In *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (Eds.), Vol. 34. Curran Associates, Inc., 10195–10208. https://proceedings.neurips.cc/paper/2021/file/547b85f3fafdf30856386753dc21c4e1-Paper.pdf

[43] Ilias Diakonikolas and Daniel M Kane. 2019. Recent advances in algorithmic high-dimensional robust statistics. *arXiv preprint arXiv:1911.05911* (2019).

[44] Ilias Diakonikolas and Daniel M. Kane. 2020. Small Covers for Near-Zero Sets of Polynomials and Learning Latent Variable Models. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020*. IEEE, 184–195.

[45] Ilias Diakonikolas, Daniel M. Kane, and Alistair Stewart. 2018. List-decodable robust mean estimation and learning mixtures of spherical gaussians. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*. 1047–1060.

[46] Ilias Diakonikolas, Weihao Kong, and Alistair Stewart. 2019. Efficient Algorithms and Lower Bounds for Robust Linear Regression. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019*, Timothy M. Chan (Ed.). SIAM, 2745–2754.

[47] Yihe Dong, Samuel B. Hopkins, and Jerry Li. 2019. Quantum Entropy Scoring for Fast Robust Mean Estimation and Improved Outlier Detection. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*. 6065–6075.

[48] Jon Feldman, Rocco A Servedio, and Ryan O'Donnell. 2006. PAC learning axis-aligned mixtures of Gaussians with no separation assumption. In *International Conference on Computational Learning Theory*. Springer, 20–34.

[49] Rong Ge, Qingqing Huang, and Sham M Kakade. 2015. Learning mixtures of gaussians in high dimensions. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*. 761–770.

[50] Frank R. Hampel, Elvezio M. Ronchetti, Peter J. Rousseeuw, and Werner A. Stahel. 1986. *Robust Statistics: the Approach based on Influence Functions*. Wiley Series in Probability and Mathematical Statistics.

[51] Moritz Hardt and Eric Price. 2015. Tight bounds for learning a mixture of two gaussians. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*. 753–760.

[52] Samuel B Hopkins and Jerry Li. 2018. Mixture models, robustness, and sum of squares proofs. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*. 1021–1034.

[53] Daniel Hsu and Sham M Kakade. 2013. Learning mixtures of spherical gaussians: moment methods and spectral decompositions. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*. 11–20.

[54] Peter J Huber. 1964. Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics* 35, 1 (1964), 73–101.

[55] Peter J Huber. 2004. *Robust statistics*. Vol. 523. John Wiley & Sons.

[56] Arun Jambulapati, Jerry Li, and Kevin Tian. 2020. Robust Sub-Gaussian Principal Component Analysis and Width-Independent Schatten Packing. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

[57] Daniel M Kane. 2021. Robust learning of mixtures of gaussians. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 1246–1258.

[58] Ravindran Kannan, Hadi Salmasian, and Santosh S. Vempala. 2008. The Spectral Method for General Mixture Models. *SIAM J. Comput.* 38, 3 (2008), 1141–1156.

[59] Sushrut Karmalkar, Adam Klivans, and Pravesh Kothari. 2019. List-decodable linear regression. In *Advances in Neural Information Processing Systems*. 7425–7434.

[60] Adam Klivans, Pravesh K Kothari, and Raghu Meka. 2018. Efficient Algorithms for Outlier-Robust Regression. In *Conference On Learning Theory*. 1420–1430.

[61] Pravesh K Kothari, Jacob Steinhardt, and David Steurer. 2018. Robust moment estimation and improved clustering via sum of squares. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*. 1035–1046.

[62] Amit Kumar and Ravindran Kannan. 2010. Clustering with spectral norm and the k-means algorithm. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*. IEEE, 299–308.

[63] Kevin A Lai, Anup B Rao, and Santosh Vempala. 2016. Agnostic estimation of mean and covariance. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 665–674.

[64] Jerry Li and Ludwig Schmidt. 2017. Robust and proper learning for mixtures of gaussians via systems of polynomial inequalities. In *Conference on Learning Theory*. PMLR, 1302–1382.

[65] Jerry Li and Guanghao Ye. 2020. Robust Gaussian Covariance Estimation in Nearly-Matrix Multiplication Time. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

[66] Jerry Zheng Li. 2018. *Principled approaches to robust machine learning and beyond*. Ph. D. Dissertation. Massachusetts Institute of Technology.

[67] Jun Z. Li, Devin M. Absher, Hua Tang, Audrey M. Southwick, Amanda M. Casto, Sohini Ramachandran, Howard M. Cann, Gregory S. Barsh, Marcus Feldman, Luigi L. Cavalli-Sforza, and Richard M. Myers. 2008. Worldwide Human Relationships Inferred from Genome-Wide Patterns of Variation. 319 (2008), 1100–1104.

[68] Allen Liu and Ankur Moitra. 2020. Settling the Robust Learnability of Mixtures of Gaussians. *arXiv preprint arXiv:2011.03622* (2020).

[69] Michela Meister and Gregory Valiant. 2018. A data prism: Semi-verified learning in the small-alpha regime. In *Conference On Learning Theory*. PMLR, 1530–1546.

[70] Dustin G Mixon, Soledad Villar, and Rachel Ward. 2017. Clustering subgaussian mixtures by semidefinite programming. *Information and Inference: A Journal of the IMA* 6, 4 (2017), 389–415.

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

[72] Kevin P Murphy. 2012. *Machine learning: a probabilistic perspective*. MIT press.

[73] Cameron Musco and Christopher Musco. 2015. Randomized Block Krylov Methods for Stronger and Faster Approximate Singular Value Decomposition. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*. 1396–1404.

[74] Peristera Paschou, Jamey Lewis, Asif Javed, and Petros Drineas. 2010. Ancestry informative markers for fine-scale individual assignment to worldwide populations. 47, 12 (2010), 835–847.

[75] Seth Patinkin. 2011. Method, apparatus, and system for clustering and classification. US Patent 8,010,466.

[76] Karl Pearson. 1894. Contributions to the mathematical theory of evolution. *Philosophical Transactions of the Royal Society of London* 185 (1894), 71–110.

[77] Adarsh Prasad, Arun Sai Suggala, Sivaraman Balakrishnan, and Pradeep Ravikumar. 2018. Robust estimation via robust gradient estimation. *arXiv preprint arXiv:1802.06485* (2018).

[78] Prasad Raghavendra and Morris Yau. 2020. List decodable learning via sum of squares. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 161–180.

[79] Oded Regev and Aravindan Vijayaraghavan. 2017. On Learning Mixtures of Well-Separated Gaussians. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*. 85–96.

[80] Oded Regev and Aravindan Vijayaraghavan. 2017. On learning mixtures of well-separated gaussians. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 85–96.

[81] Noah A. Rosenberg, Jonathan K. Pritchard, James L. Weber, Howard M. Cann, Kenneth K. Kidd, Lev A. Zhivotovsky, and Marcus W. Feldman. 2002. Genetic structure of human populations. *Science* 298 (2002), 2381–2385.

[82] Shibani Santurkar, Dimitris Tsipras, and Aleksander Madry. 2020. Breeds: Benchmarks for subpopulation shift. *arXiv preprint arXiv:2008.04859* (2020).

[83] Jacob Steinhardt. 2018. *Robust Learning: Information Theory and Algorithms*. Ph. D. Dissertation. Stanford University.

[84] Jacob Steinhardt, Pang Wei Koh, and Percy Liang. 2017. Certified Defenses for Data Poisoning Attacks. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. 3517–3529.

[85] Jacob Steinhardt, Gregory Valiant, and Moses Charikar. 2016. Avoiding imposters and delinquents: Adversarial crowdsourcing and peer prediction. In *Advances in Neural Information Processing Systems*. 4439–4447.

[86] Brandon Tran, Jerry Li, and Aleksander Madry. 2018. Spectral signatures in backdoor attacks. In *Advances in Neural Information Processing Systems*. 8000–8010.

[87] John W Tukey. 1960. A survey of sampling from contaminated distributions. *Contributions to probability and statistics* (1960), 448–485.

[88] John W. Tukey. 1975. Mathematics and the picturing of data. In *Proceedings of the International Congress of Mathematicians, Vancouver, 1975*, Vol. 2. 523–531.

[89] Santosh Vempala and Grant Wang. 2004. A spectral algorithm for learning mixture models. *J. Comput. System Sci.* 68, 4 (2004), 841–860.

[90] Manfred K. Warmuth and Dima Kuzmin. 2006. Randomized PCA Algorithms with Regret Bounds that are Logarithmic in the Dimension. In *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006*. 1481–1488.

[91] Sławomir T Wierzchoń and Mieczysław A Kłopotek. 2018. *Modern algorithms of cluster analysis*. Springer.