

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

Ilias Diakonikolas* Daniel M. Kane† Daniel Kongsgaard‡ Jerry Li§
Kevin Tian¶

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 list-decodable mean estimation, achieving nearly-optimal statistical guarantees, with running time $O(n^{1+\epsilon_0}d)$, 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 $\alpha \rightarrow 1$ 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 [DKS18, DKK20a], providing a method to simultaneously cluster and downsample points using *one-dimensional* projections — thus, bypassing the k -PCA subroutines required by prior algorithms.

*University of Wisconsin, Madison, ilias@cs.wisc.edu.

†University of California, San Diego, dakane@cs.ucsd.edu.

‡University of California, San Diego, dkongsga@ucsd.edu.

§Microsoft Research, jerrl@microsoft.com

¶Stanford University, kjtian@stanford.edu. Part of this work was done as an intern at Microsoft Research.

Contents

1	Introduction	1
1.1	Our results	3
1.2	Technical overview	4
1.3	Related work	8
1.4	Organization	10
2	Preliminaries	10
2.1	Notation	11
2.2	Technical tools	12
2.3	Potential function approach to fast filtering	14
3	Warmup: fast Gaussian multifilter	16
3.1	Reducing GaussianPartition to GaussianSplitOrCluster	17
3.2	Implementation of GaussianSplitOrCluster	20
3.3	Full Gaussian algorithm	23
4	Fast bounded covariance multifilter	26
4.1	Reducing Partition to SplitOrCluster	27
4.2	Reducing SplitOrCluster to SplitOrTailBound and Fixing	30
4.3	Implementation of SplitOrTailBound	33
4.4	Fixing a cluster via fast filtering	36
4.5	Runtime analysis	41
4.6	Full bounded covariance algorithm	42
4.7	Cleaning up the list	45
4.8	(Slightly) improving the error rate	47
5	Clustering mixture models	48
5.1	Clustering uniform (sub-)Gaussian mixture models	48
5.2	Robustly clustering (sub-)Gaussian mixture models	50
5.3	Mixture models with bounded fourth moments	52
5.4	Bounded-covariance mixture models	54

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 [Pea94], and more recently, in theoretical computer science [Das99, AK05, VW04, AM05, KSV08, BV08, AS12, RV17a].

A canonical learning task for mixture models is the *clustering problem*. Namely, given independent samples drawn from \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 a separation as possible.

The prototypical example is the case of uniform mixtures of bounded-covariance Gaussians, i.e. mixtures 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 $\|\Sigma_i\|_{\text{op}} \leq \sigma^2$. Prior to the current work, the fastest known algorithm for this learning problem was due to [AM05], building on [VW04]. Notably, [AM05] gave a polynomial-time clustering algorithm when $\Delta = \Omega(\sigma\sqrt{k})$. Interestingly, the algorithmic approach of [VW04, AM05] 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 $\tilde{\Omega}(ndk)$ – the cost of (approximate) k -PCA.¹ 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. [Mur12].

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 roughly k . In many real-world settings, this factor of k is quite significant. For instance, modern image datasets such as ImageNet [DDS⁺09] often have hundreds or thousands of different classes and subclasses [STM20]. 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,² no faster algorithm has been developed for the clustering task. In particular, the runtime of k -PCA has remained a bottleneck in this setting.

¹Throughout the paper, when convenient, we hide polylogarithmic factors in the sample size and algorithm failure probabilities with the \tilde{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.

²We note that a recent line of work has developed sophisticated polynomial-time clustering algorithms under smaller separation assumptions, see e.g. [DKS18, HL18, KSS18]. These algorithms leverage higher moments of the distribution and consequently require significantly higher sample and computational complexity.

The preceding discussion motivates the following natural question.

Question 1. *Can we cluster mixtures of k “well-separated” structured distributions without the use of k -PCA? More ambitiously, is there a clustering 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 $\text{poly}(k)$. 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. [Pat11, WK18], 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 [BNJT10, BNL12, SKL17, DKK⁺19b] and exploratory data analysis e.g. in biology [RPW⁺02, LAT⁺08, PLJD10], 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 [Ans60, Tuk60, Hub64, Tuk75, HRRS86, Hub04] developed robust estimators for many basic tasks, although with computational costs scaling exponentially in the problem dimension. More recently, a line of work in computer science, starting with [DKK⁺19a, LRV16], developed the first computationally-efficient learning algorithms (attaining near-optimal error) for various estimation problems. Subsequently, there has been significant progress in algorithmic robust statistics in a variety of settings (see [DK19] 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* [BBV08, CSV17].

Definition 1 (List-decodable learning). *Given a parameter $0 < \alpha < \frac{1}{2}$ 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 [SVC16, MV18] or semi-random community detection in stochastic block models [CSV17]. Moreover, it is particularly useful in the context of semi-verified learning [CSV17, MV18], where a learner can audit a small amount of trusted data. *An important remark is that the parameter $\alpha \in (0, \frac{1}{2})$ 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 [MV18], where the parameter α is tiny, depending inversely-polynomially on other 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 α set to $\frac{1}{k}$, 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 distribution \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 the presence of a small fraction of corrupted samples) — a reduction we formalize in this work.

The first polynomial-time algorithm for list-decodable mean estimation, when \mathcal{F} is the family of bounded-covariance distributions, was by [CSV17]. The [CSV17] algorithm was based on black-box calls to semidefinite program solvers and had a large polynomial runtime. Since then, a sequence of works [DKK20a, CMY20, DKK⁺20b] have obtained substantially improved runtimes for this problem, while retaining the (near-optimal) statistical guarantees of [CSV17]. The algorithm by [DKK⁺20b] runs in time $\tilde{O}(\frac{nd}{\alpha})$ and achieves near-optimal error (within a polylogarithmic factor).

Interestingly, as in the case of clustering mixture models, the $\tilde{\Omega}(\frac{nd}{\alpha})$ runtime dependence of the [DKK⁺20b] 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 [DKK⁺20b] 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 $\tilde{\Omega}(\frac{nd}{\alpha})$ 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 perform list-decodable mean estimation with near-optimal statistical guarantees 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 longstanding 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 (informal, see Theorem 6). *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 $\|\Sigma\|_{\text{op}} \leq \sigma^2$, and let $\alpha \in (0, 1)$. 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*

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

Notably, 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 [DKS18]. Hence, up to a $\log(\alpha^{-1})$ factor in the error, Theorem 3 achieves optimal statistical guarantees for this problem 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 (informal, see Corollaries 6, 8, 9). *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 $\|\Sigma_i\|_{\text{op}} \leq \sigma^2$, and $\min_{i, i' \in [k], i \neq i'} \|\mu_i - \mu_{i'}\|_2 = \tilde{\Omega}(\sqrt{k})\sigma$, 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 (see Corollary 9 for a precise statement).

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 $k \leq d$. 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 (Prior work). The prior works [AM05, AS12] 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 $\Omega(k\sigma)$ [Awa21]. On the other hand, the clustering methods obtained in [CSV17] (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

Here, 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. Our full proofs are quite technically challenging, and involve several additional steps which we omit here for clarity of exposition. Throughout this section, we assume that the “scale” of the problem is $\sigma = 1$ for simplicity (e.g. distribution covariances are bounded by \mathbf{I}).

1.2.1 Prior approaches and their limitations

In this section, we briefly describe two recent fast algorithms for list-decodable mean estimation, developed by [DKK20a] and [DKK+20b],³ 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 [DKK+19a, DKK+17, Li18, Ste18, DK19]. 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, [DKS18] introduced the “multifilter” in the context of Gaussian mean estimation, which was extended to bounded covariance distributions in [DKK20a]. 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 [DHL19] and majority-outlier [DKK+20b] 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 [DHL19] 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 [WK06, AK07]. By using MMW regret bounds, [DHL19] 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, [DKK+20b] 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

³We focus on [DKK+20b] instead of [CMY20] in this technical exposition, as they both apply Ky Fan semidefinite programming machinery to obtain fast runtimes, but the [DKK+20b] approach is more relevant to this paper.

⁴In [DKK20a], these subsets were replaced by weight functions, but the intuition is very similar in both cases.

runtime bottleneck as prior algorithms for clustering mixture models [VW04, AM05].

Challenges in combining techniques. As mentioned, the approach of [DKK⁺20b] 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 [DHL19] 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) [DISZ18]. 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.

1.2.2 Our techniques

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), essentially matching the guarantees of [DHL19] 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 $\text{Tr}(\mathbf{Y}_t^2)$, 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

$$\langle \mathbf{Y}_t^2, \mathbf{M}_{t+1} \rangle \leq O(1) \text{Tr}(\mathbf{Y}_t^2). \quad (1)$$

Combining (1) with a fact from [JLT20] 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 or cluster step is used). In particular, applying (1) repeatedly for any path in the multifilter tree implies that the depth is $\text{polylog}(d)$. 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, we first show how to achieve this in the case where the ground-truth, \mathcal{D} , is a bounded-covariance Gaussian (see Section 3), 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 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, 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 $\|\hat{\mu}_i - \mu_i\|_2 = \tilde{O}(\sqrt{\alpha^{-1}})$ 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 [DKS18] and improve it to run in nearly-linear time via randomized distance comparisons. The main idea of the [DKS18] 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 $\tilde{O}(\sqrt{\alpha^{-1}})$ from μ_i . By rounding every sample to its nearest hypothesis, and assuming separation $\tilde{\Omega}(\sqrt{\alpha^{-1}})$ 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 poorly-behaved points or 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

Here we survey the most relevant prior work on learning mixture models and robust statistics.

Mixture models. The closest line of work to our results studies efficiently clustering mixture models under mean-separation conditions, and in particular Gaussian mixtures [Das99, VW04, AM05, DS07, AK05, KK10, AS12, RV17b, DKS18, HL18, KSS18, MVW17]. As mentioned previously, within the class of algorithms with runtime $\tilde{O}(\frac{nd}{\alpha})$, [AM05] 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 [AM05], [VW04] gave a similar algorithm attaining the same runtime of $\tilde{O}(\frac{nd}{\alpha})$, under a weaker separation condition (scaling as roughly $\alpha^{-\frac{1}{4}}$). We are not aware of algorithms with this runtime and separation condition for clustering Gaussian mixtures when the covariances are only *spectrally bounded by* the identity.

Subsequent work generalized the statistical setting studied in [AM05, AK05, KK10, AS12], by improving on the separation condition using more sophisticated algorithmic tools, see, e.g. [DKS18, HL18, KSS18, DK20]. 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 [BK20b, DHKK20, BDH⁺20]. Beyond clustering, stronger notions of learning have been studied in this setting, including parameter estimation [MV10, BS15, HP15], proper learning [FSO06, DK14, AJOS14, LS17, ABDH⁺18], and their robust analogues [Kan21, LM20, BDJ⁺20]. 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 [HK13, ABG⁺14, BCMV14, GHK15] and density estimation [DL12, CDSS14, ADLS17]. These latter results are orthogonal to the results of the current paper.

Robust statistics and list-decodable learning. Since the pioneering work from the statistics community in the 1960s and 1970s [Ans60, Tuk60, Hub64, Tuk75], there has been a tremendous amount of work on designing robust estimators, e.g. [HRRS86, Hub04]. 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 [DKK⁺19a, LRV16]. 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. [DKK⁺17, BDLS17, CDKS18, KKM18, DKS19, PSBR18, DKK⁺19b, DKK⁺19c, TLM18]. For a more complete account, the reader is referred to [Li18, Ste18, DK19].

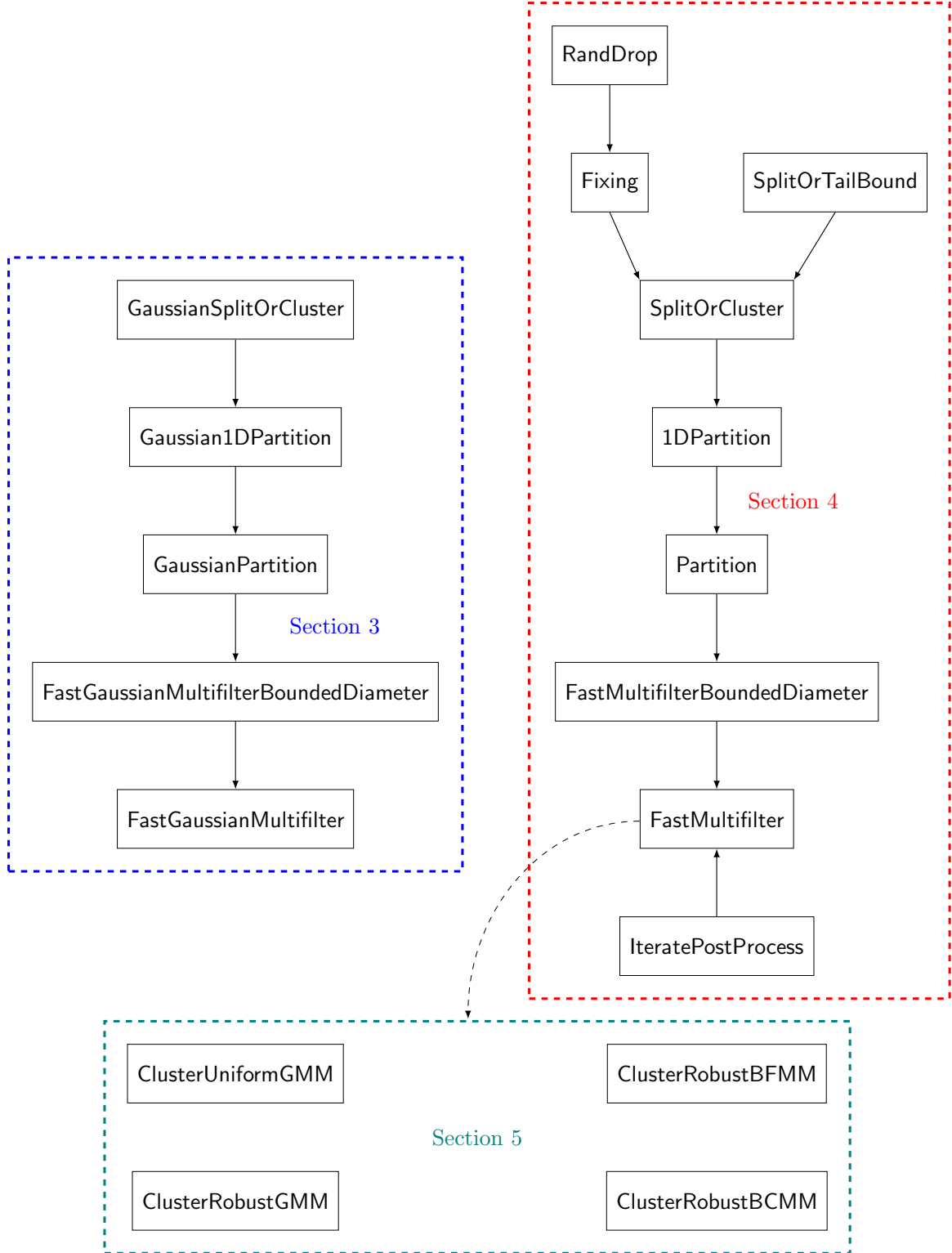


Figure 1: An illustration of the dependencies of the different algorithms of this paper together with which section the algorithms are described in. The dashed arrow from `FastMultifilter` to Section 5 is meant to indicate that the algorithms of this section utilize `FastMultifilter` indirectly.

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 [CDG19, CDGW19, DHL19, LY20, JLT20]. Importantly, these algorithms only work in the regime where the fraction of corrupted samples is small, i.e. when $\alpha \rightarrow 1$.

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 [BBV08, CSV17]. In particular, [CSV17] gave the first polynomial-time algorithm with near-optimal statistical guarantees for the problem of list-decodable mean estimation under bounded covariance assumptions. Shortly thereafter, efficient list-decodable mean estimators with near-optimal error guarantees were developed under stronger distributional assumptions [DKS18, KSS18]. More recently, a line of work developed list-decodable learners for more challenging tasks, including linear regression [RY20, KKK19] and subspace recovery [RY20, BK20a]. 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 [CMY20, DKK20a, DKK⁺20b]. We note that the algorithms in [CMY20, DKK⁺20b] 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 [DKK20a] only uses one-dimensional projections. However, their algorithm and its analysis does not have a direct interpretation as a continuous optimization method, using more problem-specific potentials, which guarantee 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 [CMY20, DKK⁺20b] to find better univariate projections for the multifilter of [DKK20a], and then designing a new potential function inspired by regret analyses of matrix multiplicative weights to demonstrate rapid termination of our fast multifilter.

1.4 Organization

In Section 2, we define notation and recall tools from prior work. We also give a technical overview of our one-shot potential approach to fast robust mean estimation. In Section 3, we give a fast multifilter implementation in the Gaussian setting, as a simplified introduction to our techniques. In Section 4, we give our full fast multifilter for bounded-covariance distributions, proving Theorem 3. In Section 5, we give our applications to clustering mixture models, proving Theorem 4. See Figure 1 for a pictorial depiction of the organization of the remainder of the paper.

2 Preliminaries

In Section 2.1, we define the notation used throughout this paper. Next, we recall some technical tools (primarily from prior work) which we draw upon in Section 2.2. We conclude with a sketch of our potential function approach to filtering in Section 2.3 by demonstrating how it works for robust mean estimation in the minority-outlier regime, giving an alternative approach to obtaining the runtimes of [DHL19]. This new approach bypasses an explicit matrix multiplicative weights argument in favor of a one-step potential. We ultimately generalize this technique to the list-decodable setting by interlacing it with clustering steps, inspired by the multifilter algorithm of [DKS18, DKK20a].

2.1 Notation

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 $[d] := \{j \mid j \in \mathbb{N}, 1 \leq j \leq d\}$. 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 $\Delta^n := \{x \in \mathbb{R}_{\geq 0}^n, \|x\|_1 \leq 1\}$. 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 $\mathbb{S}_{\geq 0}^d$. We use the Loewner partial ordering \preceq on $\mathbb{S}_{\geq 0}^d$. The largest eigenvalue, smallest eigenvalue, and trace of a matrix are given by $\lambda_{\max}(\cdot)$, $\lambda_{\min}(\cdot)$, $\text{Tr}(\cdot)$ respectively. We use $\|\cdot\|_{\text{op}}$ to mean the $(\ell_2\text{-}\ell_2)$ operator norm, which is the largest eigenvalue for arguments in $\mathbb{S}_{\geq 0}^d$. The inner product on $\mathbf{A}, \mathbf{B} \in \mathbb{S}^d$ is given by $\langle \mathbf{A}, \mathbf{B} \rangle := \text{Tr}(\mathbf{A}\mathbf{B})$.

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

$$\mu_w(T') := \sum_{i \in T'} \frac{w_i}{\|w_{T'}\|_1} X_i, \quad \text{Cov}_w(T') := \sum_{i \in T'} \frac{w_i}{\|w_{T'}\|_1} (X_i - \mu_w(T')) (X_i - \mu_w(T'))^\top.$$

For convenience, we also define the unnormalized covariance matrix by

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

We say distribution \mathcal{D} with $\mathbb{E}_{X \sim \mathcal{D}}[X] = \mu^*$ has sub-Gaussian parameter σ in all directions if $\mathbb{E}_{X \sim \mathcal{D}}[\exp(s \langle X - \mu^*, v \rangle)] \leq \exp(\frac{\sigma^2 s^2}{2})$ for all unit vectors v . In Section 5 we use concentration properties of sub-Gaussian random variables, which are well-known and can be found e.g. in the reference [RH17].

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 [DKK⁺20b]. 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 [CSV17] guarantees Assumption 1 holds with inverse-exponential failure probability. We also note that Proposition 5.4(ii) of [DKS18] 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

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

This setup 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 . Because of this, we will set $\sigma = 1$ throughout for simplicity.

Finally, throughout Sections 3 and 4 we will make the explicit assumption that $d \geq \alpha^{-1}$; for the regime where this is not the case, Algorithm 14 of [DKK⁺20b] obtains optimal error rates in time $\tilde{O}(\alpha^{-2})$ (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 $\tilde{O}(\alpha^{-1})$).

2.2 Technical tools

We will use a number of technical tools throughout this work which we list here for convenience. The first few are standard facts about covariance matrices which follow directly from computation.

Fact 1 (Convexity of covariance). *For any $w \in \Delta^n$ associated with $T \subset \mathbb{R}^d$,*

$$\mu_w(T) \mu_w(T)^\top \preceq \sum_{i \in T} \frac{w_i}{\|w\|_1} X_i X_i^\top.$$

This implies that 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 (Effect of mean shift). *For any $w \in \Delta^n$ associated with $T \subset \mathbb{R}^d$, and any $v \in \mathbb{R}^d$,*

$$\sum_{i \in T} \frac{w_i}{\|w\|_1} (X_i - v)(X_i - v)^\top = \text{Cov}_w(T) + (\mu_w(T) - v)(\mu_w(T) - v)^\top \succeq \text{Cov}_w(T).$$

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

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

Next, we need the notion of safe weight removal in the list-decodable setting, adapted from [DKK⁺20b]. 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. We define our notions of safety and saturation, and state a key technical lemma which lets us reason about when saturation is preserved. In the following discussion assume we are in the list-decodable mean estimation setting we defined in Section 2.1.

Definition 2 (γ -saturated weights). *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 (γ -safe scores). We call scores $\{s_i\}_{i \in T} \in \mathbb{R}_{\geq 0}^n$ 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 [DKK⁺20b] 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 $i \in T$:

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

Then, $w^{(N)}$ is also γ -saturated.

Proof. It suffices to prove this in the case $N = 1$ and then use induction. Define

$$\delta_S := \sum_{i \in S} \frac{w_i^{(0)} - w_i^{(1)}}{\|w_S^{(0)}\|_1}, \quad \delta_T := \sum_{i \in T} \frac{w_i^{(0)} - w_i^{(1)}}{\|w^{(0)}\|_1}.$$

By using the assumption that $s^{(0)}$ is γ -safe, we conclude

$$\delta_S = \frac{1}{s_{\max}^{(0)}} \sum_{i \in S} \frac{w_i^{(0)}}{\|w_S^{(0)}\|_1} s_i^{(0)} \leq \frac{1}{\gamma} \cdot \frac{1}{s_{\max}^{(0)}} \sum_{i \in T} \frac{w_i^{(0)}}{\|w^{(0)}\|_1} s_i^{(0)} = \frac{1}{\gamma} \delta_T.$$

Now, using γ -saturation of $w^{(0)}$ and $1 - \gamma^{-1} \delta \geq (1 - \delta)^{\gamma^{-1}}$ for all $\delta \in [0, 1]$ and $\gamma > 1$,

$$\|w_S^{(1)}\|_1 = (1 - \delta_S) \|w_S^{(0)}\|_1 \geq (1 - \delta_T)^{\gamma^{-1}} \|w_S^{(0)}\|_1 \geq \alpha \left((1 - \delta_T) \|w^{(0)}\|_1 \right)^{\gamma^{-1}} = \alpha \|w^{(1)}\|_1^{\gamma^{-1}}.$$

□

Finally, we include a technical lemma proved in [CMY20, DKK⁺20b].

Lemma 2 (Lemma 2, [DKK⁺20b]). Let $w \in \Delta^n$ have $w \leq \frac{1}{n} \mathbf{1}$ entrywise, and $\|w\|_1 \geq \alpha^2$. Then,

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

In light of the lower bound of [DKS18], Lemma 2 shows to learn the mean near-optimally in the list-decodable setting, it suffices to ensure $\|w_S\|_1 = \Omega(\alpha)$ (i.e. we retain a constant fraction of the “good” weight) and $\|\text{Cov}_w(T)\|_{\text{op}} = \tilde{O}(1)$ (i.e. the weighted covariance of the dataset is bounded).

2.3 Potential function approach to fast filtering

In this section, we 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 [DHL19, LY20, DKK⁺20b].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 [DHL19] for this problem, and explain how it can be replaced with our new potential function framework. Throughout this section, fix some $0 < \epsilon < \frac{1}{2}$ and assume that amongst the dataset $T \subset \mathbb{R}^d$ of n points, there is a majority subset $S \subset T$ of size $|S| = (1 - \epsilon)n$ with bounded empirical covariance: $\text{Cov}_{\frac{1}{n}\mathbf{1}}(S) \preceq \mathbf{I}$.

The main algorithmic step in [DHL19] 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 [DHL19]). Thus, the key technical challenge is to provide a nearly-linear time implementation of this subroutine. This was accomplished in [DHL19] 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 mean 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. It is natural to ask instead whether an alternative one-shot potential decrease guarantee exists; we now describe such a 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

$$\mathbf{M}_t := \widetilde{\text{Cov}}_{w^{(t)}}(T) = \sum_{i \in T} w_i^{(t)} (X_i - \mu_{w^{(t)}}(T))(X_i - \mu_{w^{(t)}}(T))^\top, \quad \mathbf{Y}_t := \mathbf{M}_t^{\log d}.$$

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

Fact 4 (Lemma 7, [JLT20]). *Let $\mathbf{A} \succeq \mathbf{B}$ be matrices in $\mathbb{S}_{\geq 0}^d$, and let $p \in \mathbb{N}$. Then*

$$\text{Tr}(\mathbf{A}^{p-1}\mathbf{B}) \geq \text{Tr}(\mathbf{B}^p).$$

Fact 5. *For any $\gamma \geq 0$ and $\mathbf{A} \in \mathbb{S}_{\geq 0}^d$,*

$$\gamma \text{Tr}(\mathbf{A}^{2 \log d}) \leq \text{Tr}(\mathbf{A}^{2 \log d+1}) + d\gamma^{2 \log d+1}.$$

Proof. This is immediate since each of the d eigenvalues of $\mathbf{A}^{2 \log d}$ is either at least $\gamma^{2 \log d}$ or not, and both of these cases are accounted for on the right hand side of the conclusion. \square

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

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

⁵MMW guarantees are also implicitly used in approaches based on packing SDPs, see e.g. [CDG19, CDGW19, CMY20]. However, [DHL19, LY20, DKK⁺20b] use MMW guarantees in a non-black-box way to design filters.

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

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

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 $\Phi_t = \text{Tr}(\mathbf{M}_t^{2\log d})$ implies that $\|\mathbf{M}_t\|_{\text{op}}$ 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

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

We remark that (randomized) constant-factor approximations can be computed to all $s_i^{(t)}$ via Johnson-Lindenstrauss projections in $\tilde{O}(nd)$ time, but for this discussion we assume we exactly know all scores. Then, by linearity of trace the condition (3) is implied by

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

since Fact 2 implies that $\langle \mathbf{Y}_t^2, \mathbf{M}_{t+1} \rangle \leq \sum_{i \in T} w_i^{(t+1)} s_i^{(t)}$. Finally, we note that 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 the criterion (5). For any smaller K , it can be shown that downweighting “one more time” preserves the invariant that more outlier mass is removed, precisely because (5) has not been met. Finally, binary searching to find the smallest value of K meeting (3) yields a complete algorithm running in $\tilde{O}(nd)$ time (for further details on the implementation of this binary search on K , see Theorem 2.4 of [DHL19]).

Generalizing to the majority-outlier regime. Our algorithms for the list-decodable setting marry this potential function argument with a multifilter, which produces multiple candidate filtered weight vectors on an input weight vector. We will instead show that for the *tree* of weight vectors, where a node has children given by the candidates produced from the multifilter, at least one child both halves the potential and performs only γ -safe weight removal, for some γ . After polylogarithmic layers, we will return all empirical means of leaf nodes as our list of estimates. The

child on the “safe branch” will then have a bounded potential and a γ -saturated weight vector, which suffices to guarantee an accurate mean estimate.

There are a number of additional complications which arise in this extension, which we briefly mention here as a preface to the following Sections 3 and 4. In order to process every layer of the multifilter tree in almost-linear time, we need to ensure that the number of datapoints across all the nodes, including repetitions, has not grown by more than a constant factor. The multifilter of [DKK20a] gives a variant of this guarantee by tracking the sums of the squared ℓ_1 norms of weights associated with different nodes as a nonincreasing potential, i.e.

$$\sum_{i \in \mathcal{S}} \|w^{(i)}\|_1^2,$$

where \mathcal{S} is the set of nodes and each $w^{(i)}$ is a current candidate weight function in \mathcal{S} . This is not sufficiently strong of a guarantee in our setting, since even points with very small weights need to be factored into calculations and thus affect runtime. We modify this approach in two ways. First, we replace the downweighting step with a randomly subsampled filter, which we show preserves various safety conditions such as those in Lemma 1 with high probability. Next, we replace the squared ℓ_1 potential with one involving $1 + \beta$ powers, for some $\beta \in (0, 1)$, which we prove is compatible with the multifilter. Overall, our filter tree contains polylogarithmically many layers, each of which accounts for sets with total cardinality $O(n^{1+O(\beta)})$, giving us our final runtime.

3 Warmup: fast Gaussian multifilter

As a warmup to our later (stronger) developments in Section 4, 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} . Conceptually, the types of statements Gaussian concentration (rather than heavy-tailed concentration) allow us to make let us simplify several of the technical difficulties alluded to at the end of Section 2.3, in particular the following.

1. Instead of a “covariance bound” statement such as (3) to use in our potential proof, we will simply guarantee that the multifilter returns sets of points which lie in short intervals along a number of random directions given by a Johnson-Lindenstrauss sketch.
2. Instead of a randomly subsampled filtering step to remove outliers without soft downweighting (to preserve truly small subsets), it will be enough to deterministically set thresholds along 1-dimensional projections to safely remove the outliers.
3. The definition of the Gaussian multifilter (see Section 3.2) will be substantially simpler, since we have more explicit tail bounds to check for outliers.

The strength of the error guarantees of the simpler algorithm in this section are somewhat weaker than those of Section 4 even when specialized to the Gaussian case, but we include this section as an introductory exposition of our techniques. We will use the stronger Gaussian concentration assumption in this section, a tightening of Assumption 1.

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 [DKK⁺20b] runs in nearly-linear time. On the other hand, randomly sampling the dataset solves the list-decodable mean estimation problem near-optimally in time $\tilde{O}(\alpha^{-1})$ (see Appendix A of [DKK⁺20b] 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 $\alpha n = \Theta(d \cdot \text{polylog}(d))$, and a vector $\mu^* \in \mathbb{R}^d$, such that for all unit vectors $v \in \mathbb{R}^d$ and thresholds $t \in \mathbb{R}_{\geq 0}$,*

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

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

This assumption is standard in the literature, and follows when the true distribution which S is sampled from is Gaussian with identity-bounded covariance (see e.g. Definition A.4, Lemma A.5 [DKK⁺17]). We remark that the sample complexity of Assumption 2 is worse than that of Assumption 1 by a polylogarithmic factor. This lossiness is just to simplify exposition in this warmup section, and indeed in the following Section 4 we give an algorithm which recovers stronger guarantees than Theorem 5, this section’s main export, under only Assumption 1.

In Section 3.1, we first give our main subroutine, **GaussianPartition**, which takes a candidate set and produces a number of children candidate sets which each satisfy a progress guarantee similar to (3). The main difficulty will be in guaranteeing that the children sets are sufficiently small, and that if the parent set was “good” (had large overlap with S), then at least one child set will as well. We reduce **GaussianPartition** to a number of one-dimensional clustering steps, which we implement as **GaussianSplitOrCluster** in Section 3.2. Finally, we use the guarantees of **GaussianPartition** within our potential-based framework outlined in Section 2.3, giving our final algorithm **FastGaussianMultifilter** in Section 3.3. Throughout, sets S and T are fixed and satisfy Assumptions 1 and 2.

3.1 Reducing GaussianPartition to GaussianSplitOrCluster

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 **GaussianPartition**, and develop it in this section. There are three key properties of **GaussianPartition** 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 $\tilde{O}(nd)$. We formalize this with a parameter $\beta \in (0, 1]$ throughout the rest of this section, and will guarantee that every time **GaussianPartition** is called on a parent node p ,

$$\sum_{\ell \in [k]} |T_{c_\ell}|^{1+\beta} \leq |T_p|^{1+\beta}. \quad (6)$$

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

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

every \mathbf{M}_{c_ℓ} satisfies the bound

$$\langle \mathbf{Y}_p^2, \mathbf{M}_{c_\ell} \rangle \leq R^2 \text{Tr}(\mathbf{Y}_p^2), \quad (8)$$

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 `GaussianPartition`.

Algorithm 1 `GaussianPartition`(T_p, α, β, 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)

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

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

$$|T_{c_\ell} \cap S| \geq |T_p \cap S| - \frac{1}{C}|S|. \quad (9)$$

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 $v_j \leftarrow \mathbf{Y}_p u_j$ for all $j \in [N_{\text{dir}}]$.

4: $\mathcal{S}_0 \leftarrow T_p$

5: **for** $j \in [N_{\text{dir}}]$ **do**

6: $\mathcal{S}_j \leftarrow \emptyset$

7: **for** $T' \in \mathcal{S}_{j-1}$ **do**

8: $\mathcal{T} \leftarrow \text{Gaussian1DPartition}(T', \alpha, v_j, \beta, C N_{\text{dir}}, R)$

9: $\mathcal{S}_j \leftarrow \mathcal{S}_j \cup \mathcal{T}$

10: **end for**

11: **end for**

12: **return** $\mathcal{S}_{N_{\text{dir}}}$

It heavily relies on a subroutine, `Gaussian1DPartition`(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 along the direction v , each child subset is contained in a relatively short interval.

Once again, `Gaussian1DPartition` heavily relies on a subroutine, `GaussianSplitOrCluster`, which we implement in Section 3.2. It takes as input a set T'' and either produces one or two subsets of T'' as output. If it outputs one set, that set has length at most $R\|v\|_2$ in the direction v ; otherwise, `Gaussian1DPartition` simply recurses on the additional two sets. Crucially, `GaussianSplitOrCluster` guarantees that if T'' has substantial overlap with S , then so does at least one child; moreover, when `GaussianSplitOrCluster` returns two sets, they satisfy a size potential such as (10). We now demonstrate correctness of `GaussianPartition`, assuming that `Gaussian1DPartition` is correct.

Lemma 3. *The output of `GaussianPartition` satisfies the guarantees given in Line 2 of Algorithm 1, assuming correctness of `Gaussian1DPartition`.*

Proof. First, to demonstrate that the subsets satisfy (6), we observe that we can view `GaussianPartition` 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 `Gaussian1DPartition` 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.

Algorithm 2 Gaussian1DPartition($T', \alpha, v, \beta, C, R$)

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)

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

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

$$\sum_{\ell \in [k]} |T''_\ell|^{1+\beta} \leq |T'|^{1+\beta}. \quad (10)$$

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

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

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:    $T'' \leftarrow$  the first element of  $\mathcal{S}_{\text{in}}$ 
6:    $\mathcal{S}_{\text{in}} \leftarrow \mathcal{S}_{\text{in}} \setminus T''$ 
7:   if GaussianSplitOrCluster( $T'', \alpha, \beta, R, \frac{1}{Cn}$ ) returns one set  $T_{\text{out}}^{(0)}$  then
8:      $\mathcal{S}_{\text{out}} \leftarrow \mathcal{S}_{\text{out}} \cup \{T_{\text{out}}^{(0)}\}$ 
9:   else
10:     $T_{\text{out}}^{(1)}, T_{\text{out}}^{(2)} \leftarrow$  GaussianSplitOrCluster( $T'', \alpha, \beta, R, \frac{1}{Cn}$ )
11:     $\mathcal{S}_{\text{in}} \leftarrow \mathcal{S}_{\text{in}} \cup \{T_{\text{out}}^{(1)}, T_{\text{out}}^{(2)}\}$ 
12:   end if
13: end while

```

Next, by recursively using the guarantee of Gaussian1DPartition, every $T_{c_l} \in \mathcal{S}_{N_{\text{dir}}}$ will satisfy

all values $\{\langle v_j, X_i \rangle \mid i \in T_{c_l}\}$ are contained in an interval of length $R\|v_j\|_2$, for all $j \in [N_{\text{dir}}]$.

In other words, this set is short along all the directions $\{\mathbf{Y}_p u_j = v_j\}_{j \in [N_{\text{dir}}]}$. This lets us conclude

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

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 [Ach03] which says that for any vector v , $\frac{1}{N_{\text{dir}}} \sum_{j \in [N_{\text{dir}}]} \langle u_j, v \rangle^2 \in [0.6, 1.4] \|v\|_2^2$ 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 Gaussian1DPartition, and the fifth used $|T_{c_l}| \leq n$ and the Johnson-Lindenstrauss lemma guarantee that $\frac{1}{N_{\text{dir}}} \sum_{j \in [N_{\text{dir}}]} \|\mathbf{Y}_p u_j\|_2^2 \in [0.6, 1.4] \text{Tr}(\mathbf{Y}_p^2)$ with probability at least $1 - \frac{1}{2d^3}$, which can be deduced by the guarantee of [Ach03] applied to the rows of \mathbf{Y}_p . Union bounding over the two applications of [Ach03] 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 Gaussian1DPartition inductively, every \mathcal{S}_j will have at least one element T' satisfying $|T' \cap S| \geq \frac{1}{2}|S|$. Every call to Gaussian1DPartition only removes $\frac{1}{C N_{\text{dir}}}|S|$ points in S , so overall only $\frac{1}{C}|S|$ points are removed. \square

3.2 Implementation of GaussianSplitOrCluster

In this section, we first state GaussianSplitOrCluster and analyze its correctness. We conclude with a full runtime analysis of Gaussian1DPartition, using our GaussianSplitOrCluster implementation.

To analyze Algorithm 3 we first demonstrate that it always returns in at least one case. In particular, we demonstrate that whenever the set $T_{\text{out}}^{(0)}$ 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 $T_{\text{out}}^{(0)}$ 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 $T_{\text{out}}^{(0)}$.

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 $\geq \tau$.

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

- 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 $T_{\text{out}}^{(0)} \subset T_{\text{in}}$, or two subsets $T_{\text{out}}^{(1)}, T_{\text{out}}^{(2)} \subset T_{\text{in}}$. In the one subset case, $T_{\text{out}}^{(0)}$ has $\left\{ \langle v, X_i \rangle \mid i \in T_{\text{out}}^{(0)} \right\}$ 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 $r := \frac{R}{4k_{\max}}$, $k_{\max} = \Theta\left(\frac{\log \log(\frac{1}{\alpha\Delta})}{\beta}\right)$

$$T_{\text{out}}^{(1)} := \{X_i \mid \langle v, X_i \rangle \leq \tau + r \|v\|_2\}, \quad T_{\text{out}}^{(2)} := \{X_i \mid \langle v, X_i \rangle \geq \tau - r \|v\|_2\}, \quad (11)$$
 - and satisfy
$$\left|T_{\text{out}}^{(1)}\right|^{1+\beta} + \left|T_{\text{out}}^{(2)}\right|^{1+\beta} < |T_{\text{in}}|^{1+\beta}. \quad (12)$$
 - 3: $Y_i \leftarrow \langle v, X_i \rangle$ for all $i \in T_{\text{in}}$
 - 4: $T_{\text{out}}^{(0)} \leftarrow$ indices in the middle $1 - \alpha\Delta$ quantiles of $\{Y_i\}_{i \in T_{\text{in}}}$
 - 5: **if** $\left\{Y_i \mid i \in T_{\text{out}}^{(0)}\right\}$ is contained in an interval of length $R \|v\|_2$ **then**
 - 6: **return** $T_{\text{out}}^{(0)}$
 - 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** $T_{\text{out}}^{(1)}, T_{\text{out}}^{(2)}$ defined in (11) for any threshold τ_k inducing sets satisfying (12)
 - 11: **end if**
-

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

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

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 recursion:

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

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 $\tau_{\text{med}} + (2k_{\text{max}} - 1)r \|v\|_2 \leq \tau_{\text{med}} + \frac{1}{2}R \|v\|_2$. By repeating this argument in the range $-k_{\text{max}} \leq k \leq -1$, we obtain a contradiction (as Algorithm 3 should have returned $T_{\text{out}}^{(0)}$). \square

We next prove that if the input T' to **Gaussian1DPartition** 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. This proof uses the implementation of **GaussianSplitOrCluster** in a white-box way, as well as Assumption 2.

Lemma 5. *Whenever **Gaussian1DPartition** is called on T' with $|T' \cap S| \geq (\frac{1}{2} + \frac{1}{C})|S|$ with parameters R, C satisfying (for sufficiently large constants)*

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

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

Proof. We first discuss the structure of **Gaussian1DPartition**. We say a call to **GaussianSplitOrCluster** is a “split step” if it produces two sets, and otherwise we call it a “cluster step.” Every output child of **Gaussian1DPartition** 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 **GaussianSplitOrCluster** in **Gaussian1DPartition** 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 **Gaussian1DPartition**, *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. \square

Finally, we conclude with a runtime analysis of **Gaussian1DPartition**.

Lemma 6. *Let $n' := |T'|$ for some $T' \subseteq T$. `Gaussian1DPartition` 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 `Gaussian1DPartition` means that every set in \mathcal{S}_{in} is a subinterval of T' , since this is inductively preserved by calls to `GaussianSplitOrCluster`; hence, we can represent every set implicitly by its endpoints. Moreover, given access to the initial quantile information we can implement every call to `GaussianSplitOrCluster` 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 candidate 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 `GaussianSplitOrCluster` throughout the execution of `Gaussian1DPartition`. To this end, we bound the number of times `GaussianSplitOrCluster` can return one set, and the number of times it can return two sets. Every time `GaussianSplitOrCluster` returns one set, it adds it to \mathcal{S}_{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 `GaussianSplitOrCluster` is bounded by $O((n')^{1+\beta})$, as desired. \square

As an immediate corollary, we obtain a runtime bound on `GaussianPartition`.

Corollary 1. *Let $n_p := |T_p|$ for some $T_p \subseteq T$. `GaussianPartition` 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 `Gaussian1DPartition`. 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). \square

3.3 Full Gaussian algorithm

Finally, we are ready to give our full algorithm for list-decodable mean estimation under Assumptions 1 and 2. We begin by reducing the original problem to a number of subproblems of bounded diameter (following [DKK⁺20b]), and then showing that for each of these subproblems, polylogarithmic calls to `GaussianPartition` yield subsets of bounded covariance operator norm. We conclude by recalling that a covariance operator norm bound suffices to yield guarantees on mean estimation.

We begin by stating the guarantees of `NaiveCluster`, used in Line 3 of `FastGaussianMultifilter`.

Algorithm 4 FastGaussianMultifilter(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

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

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

Algorithm 5 FastGaussianMultifilterBoundedDiameter(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_{out} with $|L_{\text{out}}| = O(\frac{1}{\alpha})$ such that some $\hat{\mu} \in L_{\text{out}}$ satisfies

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

- 3: $L^{(0)} \leftarrow \{T\}$, $L_{\text{out}} \leftarrow \emptyset$
- 4: For sufficiently large constants,

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

- 5: **for** $\ell \in [D]$ **do**
 - 6: $L^{(\ell)} \leftarrow \emptyset$
 - 7: **for** $T' \in L^{(\ell-1)}$ **do**
 - 8: Append all elements of $\text{GaussianPartition}(T', \alpha, \frac{1}{\log d}, C, R)$ to $L^{(\ell)}$ with size at least $\frac{\alpha n}{2}$
 - 9: **end for**
 - 10: **end for**
 - 11: **return** List of empirical means of all sets in $L^{(D)}$
-

Lemma 7 (Lemma 12, [DKK⁺20b]). *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 $O(d^{1.5})$. The runtime of NaiveCluster is $O(nd + n \log n)$.*

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 $\|w\|_1 \text{Cov}_w(T) = \widetilde{\text{Cov}}_{\frac{1}{n}\mathbb{1}}(T')$, and the assumed bound. \square

We now give a full analysis of FastGaussianMultifilterBoundedDiameter.

Proposition 1. *FastGaussianMultifilterBoundedDiameter 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. We can view FastGaussianMultifilterBoundedDiameter as producing 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 GaussianPartition 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, \dots, T_D , where T_D is the leaf node and $T_0 = T$ is the original set. Define the potential function at each layer $0 \leq \ell \leq D$,

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

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))

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

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 at least 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 $\widetilde{\text{Cov}}_{\frac{1}{n}\mathbb{1}}(T')$ 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 GaussianPartition 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 **GaussianPartition** shows the failure probability is at most $\frac{1}{d}$.

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 runtime guarantee. \square

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

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

Proof. We apply Proposition 1 to the relevant call of **FastGaussianMultifilterBoundedDiameter**. 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 guarantee. The runtime follows from $\sum_{i \in [k]} |T'_i| = |T|$. \square

4 Fast bounded covariance multifilter

In this section, we give our algorithm for list-decodable mean estimation under only Assumption 1. As before, we can assume without loss of generality that $\alpha \in [1/d, 1/\log^C d]$, for some constant $C > 0$. We begin by giving our main subroutine, **Partition**, in Section 4.1. The goal of **Partition** will be to produce child subsets $\{c_\ell\}_{\ell \in [k]}$ of a given input set p , which each satisfy the potential criterion in (8), reproduced here:

$$\langle \mathbf{Y}_p^2, \mathbf{M}_{c_\ell} \rangle \leq R^2 \text{Tr}(\mathbf{Y}_p^2) . \quad (15)$$

Recall that in Section 3, the way we produced children satisfying condition (15) was by ensuring that along logarithmically many random directions, each child c_ℓ lied entirely in short intervals. We will satisfy this guarantee in this section by more directly working with the definition of (15), which requires each child to have small variance along the random directions, a looser condition.

To bound the variance of the child subsets, in Section 4.2 we develop an algorithm, **SplitOrCluster**, which is patterned off our earlier **GaussianSplitOrCluster**. It either certifies that the input set is already “close” to having bounded variance in an input direction, or identifies a split point which produces two subsets which are closer to having this property, while maintaining at least one subset retains most points in S . In the first case (the “cluster” case), we develop a postprocessing procedure **Fixing** in Section 4.4 which randomly filters points according to safe outlier scores (see Definition 3) to make the remaining cluster have truly bounded variance. In the second case (the “split” case), we develop a fast threshold checking procedure **SplitOrTailBound** in Section 4.3 which identifies a valid split in polylogarithmic time, whenever one exists; here, we note the key difficulty is that we can no longer use a fixed radius for splits, because Gaussian concentration does not hold.

We discuss runtimes of all of these algorithms in Section 4.5, and in particular give a runtime bound on **Partition**. Finally, we use **Partition** to develop our full algorithm, **FastMultifilter**, which we analyze in Section 4.6 through a potential argument similar to our analysis of **FastGaussianMultifilter**. A post-processing step used in **FastMultifilter** is analyzed in Section 4.7.

4.1 Reducing Partition to SplitOrCluster

The goal of this section is to develop **Partition**, the main subroutine of **FastMultifilter**. **Partition** has very similar guarantees to the algorithm **GaussianPartition** developed in Section 3.1. It takes as input a “parent set” $T_p \subseteq T$ and produces a number of “children subsets” $\{T_{c_\ell}\}_{\ell \in [k]}$ such that every child satisfies $\langle \mathbf{Y}_p^2, \mathbf{M}_{c_\ell} \rangle \leq R^2 \text{Tr}(\mathbf{Y}_p^2)$, where we follow the definitions (7), reproduced here:

$$\begin{aligned} \mathbf{M}_p &:= \widetilde{\text{Cov}}_{\frac{1}{n}\mathbb{1}}(T_p), \quad \mathbf{Y}_p := \mathbf{M}_p^{\log d}, \\ \mathbf{M}_{c_\ell} &:= \widetilde{\text{Cov}}_{\frac{1}{n}\mathbb{1}}(T_{c_\ell}), \quad \mathbf{Y}_{c_\ell} := \mathbf{M}_{c_\ell}^{\log d} \text{ for all } \ell \in [k], \end{aligned} \tag{16}$$

This will allow us to conduct a potential analysis to bound the depth of the multifilter tree in Section 4.6. Moreover, we require two additional guarantees of **Partition**.

1. The first is the same as (6); namely, for some parameter $\beta \in (0, 1]$, we have

$$\sum_{\ell \in [k]} |T_{c_\ell}|^{1+\beta} \leq |T_p|^{1+\beta}. \tag{17}$$

This will help us bound the total work done in each layer of the multifilter tree.

2. The second is ensures at least one child preserves most points in S , assuming that the parent T_p has this property. To this end, the tools of Section 2.2 will vastly simplify the language of this section. In particular, we will ensure that for $\gamma = O(\log(\frac{1}{\alpha}))$, every filter step in this entire section will be with respect to γ -safe weights in at least one branch. We then apply Lemma 1 to conclude that some node at every level of the multifilter tree is γ -saturated.

For the remainder of Section 4, we will define

$$\gamma := 8 \log \left(\frac{1}{\alpha} \right).$$

We demonstrate one important consequence of a set being γ -saturated.

Lemma 9. *Suppose for a set $T' \subset T$, the weights $w := \frac{1}{n}\mathbb{1}_{T'} \in \Delta^n$ which place $\frac{1}{n}$ on coordinates in T' and 0 otherwise are γ -saturated (cf. Definition 2). Then,*

$$|T' \cap S| \geq \frac{\alpha n}{2}.$$

Proof. Recall that Definition 2 gives

$$\|w\|_1 \geq \|w_S\|_1 \geq \alpha \|w\|_1^{\frac{1}{\gamma}} \implies \|w\|_1^{1-\frac{1}{\gamma}} \geq \alpha \implies \|w\|_1 \geq \frac{3\alpha}{4}.$$

The first implication was by rearrangement, and the second used $\alpha^{\frac{1}{1+\gamma-1}} \geq \alpha^{1+\frac{2}{\gamma}} \geq \frac{3\alpha}{4}$. Next,

$$\|w_S\|_1 \geq \alpha \|w\|_1^{\frac{1}{\gamma}} \geq \alpha \left(\frac{3\alpha}{4} \right)^{\frac{1}{\gamma}} \geq \frac{\alpha}{2}.$$

The conclusion follows since $\|w_S\|_1$ counts the elements of $T' \cap S$, normalized by $\frac{1}{n}$. \square

Lemmas 1 and 9 imply that as long as we can guarantee that at every filtering step, at least one

child was produced with respect to γ -safe scores, that child retains half the elements of S . We are now ready to state the algorithm **Partition**, which heavily relies on a subroutine **1DPartition**.

Algorithm 6 **Partition**($T_p, \alpha, \delta, \beta, R$)

- 1: **Input:** $T_p \subset T$, $\alpha \in (0, \frac{1}{2})$, $\delta \in (0, 1)$, $\beta \in (0, 1]$, $R \in \mathbb{R}_{\geq 0}$ satisfying (for a sufficiently large constant)

$$R = \Omega \left(\max \left(\frac{1}{\beta} \cdot \sqrt{\gamma \log \left(\frac{1}{\alpha\beta} \right)}, \sqrt{\gamma \log \left(\frac{\log d}{\delta} \right)} \right) \right).$$

- 2: **Output:** With failure probability $\leq \delta$: subsets $\{T_{c_\ell}\}_{\ell \in [k]}$ of T_p , satisfying (17). Every child satisfies (15) (using notation (16)). If $w := \frac{1}{n} \mathbb{1}_{T_p}$ is γ -saturated, then $w' := \frac{1}{n} \mathbb{1}_{T_{c_\ell}}$ is γ -saturated for at least one child T_{c_ℓ} .
- 3: Sample $N_{\text{dir}} = \Theta(\log \frac{d}{\delta})$ vectors $\{u_j\}_{j \in [N_{\text{dir}}]} \in \mathbb{R}^d$ each with independent entries ± 1 . Following notation (16), let $v_j \leftarrow \mathbf{Y}_p u_j$ for all $j \in [N_{\text{dir}}]$.
- 4: $\mathcal{S}_0 \leftarrow T_p$
- 5: **for** $j \in [N_{\text{dir}}]$ **do**
- 6: $\mathcal{S}_j \leftarrow \emptyset$
- 7: **for** $T' \in \mathcal{S}_{j-1}$ **do**
- 8: $\mathcal{T} \leftarrow \text{1DPartition}(T', \alpha, v_j, \frac{\delta}{2N_{\text{dir}}}, \beta, R)$
- 9: $\mathcal{S}_j \leftarrow \mathcal{S}_j \cup \mathcal{T}$
- 10: **end for**
- 11: **end for**
- 12: **return** $\mathcal{S}_{N_{\text{dir}}}$
-

As in the Gaussian case, we develop an algorithm **1DPartition** which in turn is based on a subroutine **SplitOrCluster**, which we implement in Section 4.2. The algorithm **1DPartition** takes an input direction v and guarantees that along this direction, every child subset produced has small variance (scaled by the length of v). **1DPartition** is implemented by recursively calling **SplitOrCluster**, which takes as input a set T'' and produces either one or two subsets, analogously to **GaussianSplitOrCluster**.

Lemma 10. *The output of **Partition** satisfies the guarantees given in Line 2 of Algorithm 6, assuming correctness of **1DPartition**.*

Proof. We will follow the proof of Lemma 3. First, to demonstrate that the subsets satisfy (17), inducting on the guarantee (18) of **1DPartition** suffices. Similarly, γ -saturation of some child follows from inducting on the corresponding guarantee of **1DPartition**.

Finally, using the guarantee (19) of **1DPartition**, every $T_{c_\ell} \in \mathcal{S}_{N_{\text{dir}}}$ satisfies

$$\left\langle v_j v_j^\top, \widetilde{\text{Cov}}_{\frac{1}{n} \mathbb{1}}(T_{c_\ell}) \right\rangle \leq \frac{1}{2} R^2 \|v_j\|_2^2, \text{ for all } j \in [N_{\text{dir}}].$$

Algorithm 7 1DPartition($T', \alpha, v, \delta, \beta, R$)

- 1: **Input:** $T' \subset T$, $\alpha \in (0, \frac{1}{2})$, $v \in \mathbb{R}^d$, $\delta \in (0, 1)$, $\beta \in (0, 1]$, $R \in \mathbb{R}_{\geq 0}$ satisfying (for a sufficiently large constant)

$$R = \Omega \left(\max \left(\frac{1}{\beta} \cdot \sqrt{\gamma \log \left(\frac{1}{\alpha\beta} \right)}, \sqrt{\gamma \log \left(\frac{\log d}{\delta} \right)} \right) \right).$$

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

$$\sum_{\ell \in [k]} |T''_\ell|^{1+\beta} \leq |T'|^{1+\beta}. \quad (18)$$

Every child T''_ℓ for $\ell \in [k]$ has

$$\left\langle \widetilde{\text{Cov}}_{\frac{1}{n}\mathbb{1}}(T''_\ell), vv^\top \right\rangle \leq \frac{1}{2} R^2 \|v\|_2^2. \quad (19)$$

If $w = \frac{1}{n}\mathbb{1}_{T'}$ is γ -saturated, then $w' = \frac{1}{n}\mathbb{1}_{T''_\ell}$ is γ -saturated for at least one child T''_ℓ , with failure probability $\leq \delta$.

- 3: $\mathcal{S}_{\text{in}} \leftarrow T'$, $\mathcal{S}_{\text{out}} \leftarrow \emptyset$
4: **while** $\mathcal{S}_{\text{in}} \neq \emptyset$ **do**
5: $T'' \leftarrow$ the first element of \mathcal{S}_{in}
6: $\mathcal{S}_{\text{in}} \leftarrow \mathcal{S}_{\text{in}} \setminus \{T''\}$
7: **if** SplitOrCluster($T'', \alpha, v, \delta, \beta, R$) returns one set $T_{\text{out}}^{(0)}$ **then**
8: $\mathcal{S}_{\text{out}} \leftarrow \mathcal{S}_{\text{out}} \cup \{T_{\text{out}}^{(0)}\}$
9: **else**
10: $T_{\text{out}}^{(1)}, T_{\text{out}}^{(2)} \leftarrow$ SplitOrCluster($T'', \alpha, v, \delta, \beta, R$)
11: $\mathcal{S}_{\text{in}} \leftarrow \mathcal{S}_{\text{in}} \cup \{T_{\text{out}}^{(1)}, T_{\text{out}}^{(2)}\}$
12: **end if**
13: **end while**
14: **return** \mathcal{S}_{out}
-

In other words, the variance is small along all directions $\{\mathbf{Y}_p u_j = v_j\}_{j \in [N_{\text{dir}}]}$. We conclude

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

with probability at least $1 - \delta$; the first two lines and the last line follow the proof of Lemma 3, and we used the variance guarantee of 1DPartition in the third line. We remark that we make sure

to take the number of directions N_{dir} to depend logarithmically on δ (as opposed to just d), so we can apply the guarantees of [Ach03] with probability $1 - \frac{\delta}{2}$ on the first and last lines. \square

Next, we state a correctness guarantee for **1DPartition**, assuming correctness of its main subroutine, **SplitOrCluster**. The guarantees of **SplitOrCluster** are summarized in Line 2 of Algorithm 8. The salient features are that it takes a set T_{in} and either produces one set satisfying (19) deterministically, or two sets which each are strict subsets of T_{in} (and hence remove at least one point) deterministically, and (18) is always maintained. In the two set case, if $\frac{1}{n}\mathbb{1}_{T_{\text{in}}}$ is γ -saturated then so is at least one output deterministically; in the one set case, the output is saturated with probability at least $1 - \delta$. We will use only these features to analyze **1DPartition** in Lemma 11.

Lemma 11. *The output of **1DPartition** satisfies the guarantees given in Line 2 of Algorithm 7, assuming correctness of **SplitOrCluster**.*

Proof. Each run of Lines 4-13 results in either one set (which we call a “cluster step”) or two sets (which we call a “split step”). We can view this process as a tree, where a leaf node corresponds to the result of a cluster step, and every node on the leaf-to-node path corresponds to a split step. Every time a split step occurs, it increases $|\mathcal{S}_{\text{in}}| + |\mathcal{S}_{\text{out}}|$ by one, so there are at most $(n')^{1+\beta}$ calls to **SplitOrCluster** where $|T'| = n'$, and thus the algorithm terminates in finite time.

Next, if T' is not saturated, then there is no failure probability, since the conditions (18) and (19) deterministically succeed. Otherwise, from the root of this partition tree, consider the root-to-leaf path which at each node corresponding to a split step takes any child which corresponds to a saturated child (one always exists because split steps deterministically succeed). The only failure probability comes from the success of the leaf-parent to leaf cluster step, which fails with probability δ . We can ignore all other bad events, because we only need to ensure one child is saturated. \square

4.2 Reducing SplitOrCluster to SplitOrTailBound and Fixing

In this section, we state and analyze **SplitOrCluster**, the main subroutine of **1DPartition**.

SplitOrCluster uses two subroutines, **Fixing** and **SplitOrTailBound**, which are respectively used to handle the one child and two children cases. Roughly speaking, **Fixing** takes as input a set T_{in} which “almost” has bounded variance in the direction v , and slightly filters extreme outliers in a way so that the result has truly bounded variance. On the other hand, **SplitOrTailBound** is used at a candidate threshold τ to either check that it induces sets $T_{\text{out}}^{(1)}, T_{\text{out}}^{(2)}$ satisfying (21) or satisfies a certain tail bound. By stitching together tail bounds at a small number of quantiles, **SplitOrCluster** guarantees that at least one of these quantiles was a valid threshold, else we would attain a contradiction as Line 6 of **SplitOrCluster** would have passed. We state guarantees of **SplitOrTailBound** and **Fixing** as Lemmas 12 and 13, and prove them in Sections 4.3 and 4.4 respectively. We then use Lemmas 12 and 13 to prove Lemma 14, which demonstrates correctness of **SplitOrCluster**.

Lemma 12. *There is an algorithm, **SplitOrTailBound** (Algorithm 9), which takes as input $T_{\text{in}} \subseteq T$, $v \in \mathbb{R}^d$, $\beta \in (0, 1]$, $R \in \mathbb{R}_{\geq 0}$, and $\tau_0 \in \mathbb{R}$, and returns in one of two cases we call the “split” case and the “tail bound” case. In the split case, it returns (τ, r) such that the induced sets (20) satisfy (21), and if $\frac{1}{n}\mathbb{1}_{T_{\text{in}}}$ is γ -saturated then one of the induced sets T_{out} has $\frac{1}{n}\mathbb{1}_{T_{\text{out}}}$ is γ -saturated. Otherwise, for all $t \in \mathbb{R}$ define the upper and lower tail probabilities*

$$\rho^+(t) := \Pr_{i \sim \text{unif } T_{\text{in}}} [Y_i \geq t], \quad \rho^-(t) := \Pr_{i \sim \text{unif } T_{\text{in}}} [Y_i \leq t]. \quad (22)$$

Algorithm 8 SplitOrCluster($T_{\text{in}}, \alpha, v, \delta, \beta, R$)

- 1: **Input:** $T_{\text{in}} \subseteq T$, $\alpha \in (0, \frac{1}{2})$, $v \in \mathbb{R}^d$, $\delta \in (0, 1)$, $\beta \in (0, 1]$, $R \in \mathbb{R}_{\geq 0}$ satisfying (for a sufficiently large constant)

$$R = \Omega \left(\max \left(\frac{1}{\beta} \cdot \sqrt{\gamma \log \left(\frac{1}{\alpha\beta} \right)}, \sqrt{\gamma \log \left(\frac{\log d}{\delta} \right)} \right) \right).$$

- 2: **Output:** Either one subset $T_{\text{out}}^{(0)} \subset T_{\text{in}}$, or two subsets $T_{\text{out}}^{(1)}, T_{\text{out}}^{(2)} \subset T_{\text{in}}$. In the one subset case, $T_{\text{out}}^{(0)}$ has $\left\langle \widetilde{\text{Cov}}_{\frac{1}{n}\mathbb{1}} \left(T_{\text{out}}^{(0)} \right), vv^\top \right\rangle \leq \frac{1}{2}R^2 \|v\|_2^2$. In the two subsets case, they take the form, for some threshold value $\tau \in \mathbb{R}$ and $r \in \mathbb{R}_{\geq 0}$

$$T_{\text{out}}^{(1)} := \{X_i \mid \langle v, X_i \rangle \leq \tau + r \|v\|_2\}, \quad T_{\text{out}}^{(2)} := \{X_i \mid \langle v, X_i \rangle \geq \tau - r \|v\|_2\}, \quad (20)$$

and satisfy

$$\begin{aligned} & \left| T_{\text{out}}^{(1)} \right|^{1+\beta} + \left| T_{\text{out}}^{(2)} \right|^{1+\beta} < |T_{\text{in}}|^{1+\beta}, \\ & \min \left(1 - \frac{|T_{\text{out}}^{(1)}|}{|T_{\text{in}}|}, 1 - \frac{|T_{\text{out}}^{(2)}|}{|T_{\text{in}}|} \right) \geq \frac{2\gamma}{r^2}. \end{aligned} \quad (21)$$

In either case if $\frac{1}{n}\mathbb{1}_{T_{\text{in}}}$ is γ -saturated then $\frac{1}{n}\mathbb{1}_{T_{\text{out}}}$ is γ -saturated for at least one child T_{out} , with failure probability $\leq \delta$ only in the case one set is returned (deterministically otherwise).

- 3: $Y_i \leftarrow \langle v, X_i \rangle$ for all $i \in T_{\text{in}}$
4: $\tau_{\text{med}} \leftarrow \text{med}(\{Y_i \mid i \in T_{\text{in}}\})$, where med returns the median
5: $I \leftarrow [\tau_{\text{med}} - c, \tau_{\text{med}} + c]$ is the smallest interval containing the $1 - \frac{\alpha}{4}$ quantiles of $\{Y_i \mid i \in T_{\text{in}}\}$ for $c \in \mathbb{R}_{\geq 0}$ and $2I \leftarrow [\tau_{\text{med}} - 2c, \tau_{\text{med}} + 2c]$
6: **if** $\left\langle \widetilde{\text{Cov}}_{\frac{1}{n}\mathbb{1}}(T_{\text{mid}}), vv^\top \right\rangle \leq \frac{1}{8}R^2 \|v\|_2^2$ where $T_{\text{mid}} := \{X_i \in T_{\text{in}} \mid Y_i \in 2I\}$ **then**
7: **return** Fixing($T_{\text{in}}, \alpha, v, \delta, R$)
8: **else**
9: Run both SplitOrTailBound($T_{\text{in}}, v, \beta, \tau_{\text{med}} \pm \frac{1}{2^k} \cdot \sqrt{\frac{2048c}{\beta^2\alpha}} \|v\|_2$) for integers k with

$$0 \leq k \leq \log_2 \left(\frac{2048}{\beta^2\alpha} \right)$$

until one returns $(T_{\text{out}}^{(1)}, T_{\text{out}}^{(2)})$ satisfying (20), (21)

- 10: **return** $T_{\text{out}}^{(1)}, T_{\text{out}}^{(2)}$
11: **end if**
-

Then, in the tail bound case SplitOrTailBound certifies

$$\rho^+(\tau_0) \leq \frac{128\gamma}{\beta^2 |\tau_0 - \tau_{\text{med}}|^2} \|v\|_2^2 \text{ if } \tau_0 \geq \tau_{\text{med}}, \text{ and } \rho^-(\tau_0) \leq \frac{128\gamma}{\beta^2 |\tau_0 - \tau_{\text{med}}|^2} \|v\|_2^2 \text{ if } \tau_0 \leq \tau_{\text{med}}. \quad (23)$$

Lemma 13. *There is an algorithm, Fixing (Algorithm 11), which takes as input $T_{\text{in}} \subseteq T$, $v \in \mathbb{R}^d$, and $R \in \mathbb{R}_{\geq 0}$ and produces T_{out} with the following guarantee with probability at least $1 - \delta$. Define T_{mid} as in Line 5 of Algorithm 8. Then if $\frac{1}{n}\mathbb{1}_{T_{\text{in}}}$ is γ -saturated, so is $\frac{1}{n}\mathbb{1}_{T_{\text{out}}}$, and if $\left\langle \widetilde{\text{Cov}}_{\frac{1}{n}\mathbb{1}}(T_{\text{mid}}), vv^\top \right\rangle \leq \frac{1}{8}R^2 \|v\|_2^2$, then $\left\langle \widetilde{\text{Cov}}_{\frac{1}{n}\mathbb{1}}(T_{\text{out}}), vv^\top \right\rangle \leq \frac{1}{2}R^2 \|v\|_2^2$.*

Finally, we are ready to prove Lemma 14, the main export of this section.

Lemma 14. *The output of SplitOrCluster satisfies the guarantees given in Line 2 of Algorithm 8.*

Proof. If the check in Line 6 passes (the one subset case), correctness of SplitOrCluster follows immediately from the guarantees of Fixing in Lemma 13. We now focus on the two subset case.

Assume throughout this proof that the $\{Y_i\}_{i \in T_{\text{mid}}}$ are ordered by distance to τ_{med} , so $|Y_1 - \tau_{\text{med}}| \leq \dots \leq |Y_m - \tau_{\text{med}}|$, where $m := |T_{\text{mid}}|$; we order the remaining elements $i \in T_{\text{in}} \setminus T_{\text{mid}}$ arbitrarily. We also define τ_{med} , T_{mid} , c , I , and $2I$ as in Lines 4-6 of SplitOrCluster. If Line 9 outputs a split for any k , then by Lemma 12, (20), (21) are satisfied, and the saturation condition is met for one of the children. It remains to show that some value of k will result in the split case of SplitOrTailBound. We show this by contradiction; if all k resulted in SplitOrTailBound returning with a tail bound guarantee, we prove T_{mid} would have passed Line 6. Assume for the remainder of the proof that SplitOrTailBound failed to find a split for all k .

First, we bound the length of the interval $2I$. Because SplitOrCluster failed to return a split for $k = 1$, by combining the corresponding tail bounds (23),

$$\Pr_{i \sim \text{unif } T_{\text{in}}} \left[|Y_i - \tau_{\text{med}}| > \sqrt{\frac{512}{\beta^2 \alpha}} \|v\|_2 \right] \leq \frac{\alpha}{4}.$$

Thus, we conclude

$$2I \subset [\tau_{\text{med}} - C \|v\|_2, \tau_{\text{med}} + C \|v\|_2], \text{ for } C := \sqrt{\frac{2048}{\beta^2 \alpha}}.$$

We proceed to bound $\langle \widetilde{\text{Cov}}_{\frac{1}{n}\mathbb{1}}(T_{\text{mid}}), vv^\top \rangle$ to obtain our desired contradiction. Observe that

$$\begin{aligned} \widetilde{\text{Cov}}_{\frac{1}{n}\mathbb{1}}(T_{\text{mid}}) &\preceq \text{Cov}_{\frac{1}{n}\mathbb{1}}(T_{\text{mid}}) \\ &= \frac{1}{|T_{\text{mid}}|} \sum_{i \in T_{\text{mid}}} \left(X_i - \mu_{\frac{1}{n}\mathbb{1}}(T_{\text{mid}}) \right) \left(X_i - \mu_{\frac{1}{n}\mathbb{1}}(T_{\text{mid}}) \right)^\top \\ &\preceq \frac{1}{|T_{\text{mid}}|} \sum_{i \in T_{\text{mid}}} (X_i - \bar{X}) (X_i - \bar{X})^\top \text{ where } \langle v, \bar{X} \rangle = \tau_{\text{med}} \quad (24) \\ \implies \langle \widetilde{\text{Cov}}_{\frac{1}{n}\mathbb{1}}(T_{\text{mid}}), vv^\top \rangle &\leq \frac{1}{|T_{\text{mid}}|} \sum_{i \in T_{\text{mid}}} (Y_i - \tau_{\text{med}})^2. \end{aligned}$$

Here, we used the definitions of $\widetilde{\text{Cov}}$, Cov in the first two lines, and Fact 2 in the third. The last follows by definition of $\{Y_i\}_{i \in T_{\text{in}}}$ and τ_{med} . Define the random variable Z to be the realization of $|Y_i - \tau_{\text{med}}|$ for i a uniform draw from T_{mid} , let $Z_i := |Y_i - \tau_{\text{med}}|$, and let $G(t) = \Pr[Z \geq t]$ be the inverse cumulative density function of Z . Notice that directly expanding implies that (where we let $m := |T_{\text{mid}}|$, $Z_0 := 0$, and recall we argued $Z_m \leq C \|v\|_2$ earlier)

$$\mathbb{E}[Z^2] = \sum_{i \in [m]} (Z_i^2 - Z_{i-1}^2) G(Y_i) = \int_0^{Z_m} 2tG(t)dt \leq \int_0^{C\|v\|_2} 2tG(t)dt.$$

Define now $K(t)$ for each $\|v\|_2 \leq t \leq C\|v\|_2$ to be the smallest k such that $t^{(k)} := \frac{C}{2^k} \|v\|_2 \leq t$, so

$K(C\|v\|_2) = 0$, $K(t) = 1$ for $t \in [\frac{C}{2}\|v\|_2, C\|v\|_2)$, and so on. By construction, for all relevant t ,

$$t^{(K(t))} \leq t < 2t^{(K(t))}.$$

Since G is decreasing in its argument, we can write

$$\int_0^{C\|v\|_2} 2tG(t)dt \leq \int_0^{\|v\|_2} 2tdt + \int_{\|v\|_2}^{C\|v\|_2} 2tG\left(t^{(K(t))}\right)dt \leq \|v\|_2^2 + \int_{\|v\|_2}^{C\|v\|_2} 2tG\left(t^{(K(t))}\right)dt.$$

Now, for all $0 \leq k \leq \log_2(\frac{80}{\beta^2\alpha})$, we recall that we assumed both calls to `SplitOrTailBound` with thresholds $\tau_{\text{med}} \pm \frac{C}{2^k}\|v\|_2$ failed to produce a split, and hence certify a tail bound (23). Thus,

$$G\left(t^{(k)}\right) \leq \frac{512\gamma}{\beta^2(t^{(k)})^2}\|v\|_2^2 \leq \frac{2048\gamma}{\beta^2 t^2}\|v\|_2^2, \text{ for any } t \text{ with } K(t) = k.$$

Here, the first inequality used both tail bounds in (23) and accounted for the fact that the quantizations ρ^+ , ρ^- are defined over T_{in} , and G is defined over T_{mid} with $|T_{\text{mid}}| \geq \frac{1}{2}|T_{\text{in}}|$; the second inequality used that for any such t , $t < 2t^{(k)}$. Putting all these pieces together,

$$\mathbb{E}[Z^2] \leq \|v\|_2^2 + \int_{\|v\|_2}^{C\|v\|_2} \frac{4096\gamma}{\beta^2 t^2} \|v\|_2^2 dt \leq \|v\|_2^2 + \frac{4096\gamma}{\beta^2} \log(C) \|v\|_2^2 = O\left(\frac{\gamma}{\beta^2} \cdot \log\left(\frac{1}{\alpha\beta}\right)\right) \|v\|_2^2.$$

Finally, recall (24) shows that $\left\langle \widetilde{\text{Cov}}_{\frac{1}{n}\mathbb{1}}(T_{\text{mid}}), vv^\top \right\rangle \leq \mathbb{E}[Z^2]$. Thus, the set T_{mid} should have passed the check in Line 6 under the assumed lower bound on R , yielding the desired contradiction. \square

4.3 Implementation of SplitOrTailBound

In this section, we prove Lemma 12 by providing `SplitOrTailBound` and giving its analysis.

Lemma 12. *There is an algorithm, `SplitOrTailBound` (Algorithm 9), which takes as input $T_{\text{in}} \subseteq T$, $v \in \mathbb{R}^d$, $\beta \in (0, 1]$, $R \in \mathbb{R}_{\geq 0}$, and $\tau_0 \in \mathbb{R}$, and returns in one of two cases we call the “split” case and the “tail bound” case. In the split case, it returns (τ, r) such that the induced sets (20) satisfy (21), and if $\frac{1}{n}\mathbb{1}_{T_{\text{in}}}$ is γ -saturated then one of the induced sets T_{out} has $\frac{1}{n}\mathbb{1}_{T_{\text{out}}}$ is γ -saturated. Otherwise, for all $t \in \mathbb{R}$ define the upper and lower tail probabilities*

$$\rho^+(t) := \Pr_{i \sim \text{unif } T_{\text{in}}} [Y_i \geq t], \quad \rho^-(t) := \Pr_{i \sim \text{unif } T_{\text{in}}} [Y_i \leq t]. \quad (22)$$

Then, in the tail bound case `SplitOrTailBound` certifies

$$\rho^+(\tau_0) \leq \frac{128\gamma}{\beta^2 |\tau_0 - \tau_{\text{med}}|^2} \|v\|_2^2 \text{ if } \tau_0 \geq \tau_{\text{med}}, \text{ and } \rho^-(\tau_0) \leq \frac{128\gamma}{\beta^2 |\tau_0 - \tau_{\text{med}}|^2} \|v\|_2^2 \text{ if } \tau_0 \leq \tau_{\text{med}}. \quad (23)$$

Proof. This proof proceeds in two parts which we show separately. First, we demonstrate that if $\tau_0 \geq \tau_{\text{med}}$ and none of the runs of Lines 9-17 in Algorithm 9 return with a valid split, then indeed we can certify the tail bound (23) holds (and a similar guarantee holds for $\tau_0 < \tau_{\text{med}}$). Second, we show whenever a split is returned and T_{in} is γ -saturated, then one of the output sets will be as well.

Correctness of tail bound. We consider the case $\tau_0 \geq \tau_{\text{med}}$ here, as the other case follows symmetrically. Let $K+1$ be the first index such that $\tau_{K+1} < \tau_{\text{med}}$, so the algorithm checks all pairs $(\tau_j - r_j\|v\|_2, r_j)$ for $0 \leq j \leq K$. Suppose that all such induced splits fail to satisfy (21). For all

Algorithm 9 SplitOrTailBound($T_{\text{in}}, v, \beta, \tau_0$)

- 1: **Input:** $T_{\text{in}} \subseteq T$, $v \in \mathbb{R}^d$, $\beta \in (0, 1]$, $\tau_0 \in \mathbb{R}$
2: **Output:** Either outputs (τ, r) such that $T_{\text{out}}^{(1)} := \{X_i \mid \langle v, X_i \rangle \leq \tau + r \|v\|_2\}$, $T_{\text{out}}^{(2)} := \{X_i \mid \langle v, X_i \rangle \geq \tau - r \|v\|_2\}$ satisfy

$$\left|T_{\text{out}}^{(1)}\right|^{1+\beta} + \left|T_{\text{out}}^{(2)}\right|^{1+\beta} < |T_{\text{in}}|^{1+\beta}, \quad \min \left(1 - \frac{|T_{\text{out}}^{(1)}|}{|T_{\text{in}}|}, 1 - \frac{|T_{\text{out}}^{(2)}|}{|T_{\text{in}}|} \right) \geq \frac{2\gamma}{r^2},$$

or returns “Tail bound” guaranteeing that for $\tau_{\text{med}} := \text{med}(\{\langle v, X_i \rangle \mid i \in T_{\text{in}}\})$ (following (23))

$$\rho^+(t) \leq \frac{128\gamma}{\beta^2 |\tau_0 - \tau_{\text{med}}|^2} \|v\|_2^2 \text{ if } \tau_0 \geq \tau_{\text{med}}, \text{ and } \rho^-(t) \leq \frac{128\gamma}{\beta^2 |\tau_0 - \tau_{\text{med}}|^2} \|v\|_2^2 \text{ if } \tau_0 \leq \tau_{\text{med}}.$$

- 3: $Y_i \leftarrow \langle v, X_i \rangle$ for all $i \in T_{\text{in}}$, $\tau_{\text{med}} \leftarrow \text{med}(\{Y_i \mid i \in T_{\text{in}}\})$
4: $j \leftarrow 0$
5: **if** $\tau_0 > \max_{i \in T_{\text{in}}} Y_i$ or $\tau_0 < \min_{i \in T_{\text{in}}} Y_i$ **then**
6: **return** “Tail bound”
7: **end if**
8: **if** $\tau_0 \geq \tau_{\text{med}}$ **then**
9: **while** $\tau_j \geq \tau_{\text{med}}$ **do**
10: $g_j \leftarrow \rho^+(\tau_j)$ and $r_j \leftarrow \sqrt{\frac{2\gamma}{g_j}}$
11: **if** $T_{\text{out}}^{(1)}, T_{\text{out}}^{(2)}$ induced by $(\tau_j - r_j \|v\|_2, r_j)$ satisfy (21) **then**
12: **return** $(\tau_j - r_j \|v\|_2, r_j)$
13: **else**
14: $\tau_{j+1} \leftarrow \tau_j - 2r_j$
15: **end if**
16: $j \leftarrow j + 1$
17: **end while**
18: **else**
19: **while** $\tau_j \leq \tau_{\text{med}}$ **do**
20: $\ell_j \leftarrow \rho^-(\tau_j)$ and $r_j \leftarrow \sqrt{\frac{2\gamma}{\ell_j}}$
21: **if** $T_{\text{out}}^{(1)}, T_{\text{out}}^{(2)}$ induced by $(\tau_j + r_j \|v\|_2, r_j)$ satisfy (21) **then**
22: **return** $(\tau_j + r_j \|v\|_2, r_j)$
23: **else**
24: $\tau_{j+1} \leftarrow \tau_j + 2r_j$
25: **end if**
26: $j \leftarrow j + 1$
27: **end while**
28: **end if**
29: **return** “Tail bound”
-

$0 \leq j \leq K$, let $T_j^{(1)}, T_j^{(2)}$ be the induced sets by the pair $(\tau_j - r_j \|v\|_2, r_j)$. Then by construction

$$\frac{|T_j^{(1)}|}{|T_{\text{in}}|} = 1 - g_j, \quad \frac{|T_j^{(2)}|}{|T_{\text{in}}|} = g_{j+1}.$$

For all $0 \leq j \leq K-1$, recall $(\tau_j - r_j \|v\|_2, r_j)$ did not pass the check (21), but the second expression reads $g_j \geq \frac{2\gamma}{r_j^2}$ (since $g_j \leq \frac{1}{2} \leq 1 - g_{j+1}$), which is true by construction. Thus the first check did not pass and we conclude

$$g_{j+1}^{1+\beta} + (1 - g_j)^{1+\beta} > 1 \implies g_{j+1}^{1+\beta} > g_j. \quad (25)$$

By applying this inequality inductively with $j = K-1$, and recalling $g_K \leq \frac{1}{2}$, we have

$$\left(\frac{1}{2}\right)^{(1+\beta)^K} \geq g_K^{(1+\beta)^K} > g_0 \implies 2^{(1+\beta)^K} < \frac{1}{g_0} \implies K = O\left(\frac{1}{\beta} \log \log \left(\frac{1}{g_0}\right)\right). \quad (26)$$

Next, since $\tau_{K+1} = \tau_0 - 2 \sum_{0 \leq j \leq K} r_j \|v\|_2 < \tau_{\text{med}}$, we have

$$\tau_0 - \tau_{\text{med}} < 2 \sum_{j=0}^K r_j \|v\|_2 = \sqrt{8\gamma} \|v\|_2 \sum_{j=0}^K \sqrt{\frac{1}{g_j}} < \sqrt{8\gamma} \|v\|_2 \sum_{j=0}^K A^{\frac{1}{(1+\beta)^j}}, \text{ for } A := \sqrt{\frac{1}{g_0}}.$$

where we used our earlier guarantee (25) inductively. By Lemma 15, we have the desired tail bound:

$$\tau_0 - \tau_{\text{med}} < \sqrt{8\gamma} \|v\|_2 \cdot \frac{4A}{\beta} \leq \frac{\sqrt{128}}{\beta} \sqrt{\frac{\gamma}{g_0}} \implies \rho^+(\tau_0) = g_0 < \frac{128\gamma}{\beta^2 |\tau_0 - \tau_{\text{med}}|^2} \|v\|_2^2.$$

Correctness of split. Suppose that we find (τ, r) such that for

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

we have

$$\min \left(1 - \frac{|T_{\text{out}}^{(1)}|}{|T_{\text{in}}|}, 1 - \frac{|T_{\text{out}}^{(2)}|}{|T_{\text{in}}|} \right) \geq \frac{2\gamma}{r^2}. \quad (27)$$

We show that the downweighting $\frac{1}{n} \mathbb{1}_{T_{\text{in}}} \rightarrow \frac{1}{n} \mathbb{1}_{T_{\text{out}}^{(i)}}$ is a weight removal with respect to γ -safe scores for one of $i = 1, 2$. Let $\tau^* := \langle v, \mu^* \rangle$ where μ^* is the “true mean vector” in Assumption 1. Clearly, either $\tau^* \geq \tau$ or $\tau^* \leq \tau$; suppose without loss of generality $\tau^* \geq \tau$ as the other case follows symmetrically. Define the scores $\{s_i\}_{i \in T_{\text{in}}}$ to be 1 if $i \in T_{\text{in}} \setminus T_{\text{out}}^{(2)}$ and 0 otherwise; then the downweighting $\frac{1}{n} \mathbb{1}_{T_{\text{in}}} \rightarrow \frac{1}{n} \mathbb{1}_{T_{\text{out}}^{(2)}}$ is of the form in Lemma 1, with respect to these scores. Note that

$$\frac{1}{\gamma} \sum_{i \in T_{\text{in}}} \frac{1}{|T_{\text{in}}|} s_i = \frac{1}{\gamma} \left(1 - \frac{|T_{\text{out}}^{(2)}|}{|T_{\text{in}}|} \right) \geq \frac{2}{r^2}$$

by the assumption (27). To apply Lemma 1, it remains to show that

$$\sum_{i \in S \cap T_{\text{in}}} \frac{1}{|S \cap T_{\text{in}}|} s_i \leq \frac{2}{r^2}. \quad (28)$$

However, we can extend the definition of the scores $\{s_i\}_{i \in S}$ to include points in $S \setminus T_{\text{in}}$, so that s_i is the indicator function of $\langle v, X_i \rangle < \tau - r \|v\|_2$ for all $i \in S$, which is consistent with our definitions

$\{s_i\}_{i \in T_{\text{in}}}$. Then by Chebyshev's inequality and Assumption 1, using $\langle v, \mu^* \rangle \geq \tau$,

$$\sum_{i \in S} \frac{1}{|S|} s_i \leq \Pr_{i \sim \text{unif} S} \left[\langle v, X_i - \mu^* \rangle^2 > r^2 \|v\|_2^2 \right] \leq \frac{1}{r^2}. \quad (29)$$

By Lemma 9, if T_{in} is γ -saturated then $|S \cap T_{\text{in}}| \geq \frac{1}{2}|S|$; combining with (29) yields (28). \square

Lemma 15. *Let $A > \sqrt{2}$, $\beta \in (0, 1]$, and let K be such that $A^{\frac{1}{(1+\beta)^K}} > \sqrt{2}$. Then, we have*

$$\sum_{j=0}^K A^{\frac{1}{(1+\beta)^j}} \leq \frac{4A}{\beta}.$$

Proof. Define $f(x) = A^{\frac{1}{(1+\beta)^x}}$ for any $0 \leq x \leq K$, and note this is a decreasing function in x . Thus, since by direct computation the antiderivative of A^{B^x} is $\frac{1}{\log B} \text{Ei}(B^x \log(A))$ where Ei is the exponential integral,

$$\begin{aligned} \sum_{j=0}^K A^{\frac{1}{(1+\beta)^j}} &\leq A + \int_0^K f(x) dx = A + \frac{1}{\log(1+\beta)} \left(\text{Ei}(\log A) - \text{Ei}\left(\frac{\log A}{(1+\beta)^K}\right) \right) \\ &\leq A + \frac{2}{\beta} (\text{Ei}(\log A) + 1). \end{aligned}$$

In the last line, we used $\log(1+\beta) \geq \frac{\beta}{2}$ for $\beta \in (0, 1]$, $\frac{\log A}{(1+\beta)^K} > \log(\sqrt{2})$ by assumption, and Ei is increasing with $\text{Ei}(\log(\sqrt{2})) > -1$. The conclusion follows from $\text{Ei}(\log A) + 1 \leq \frac{3}{2}A$ for $A > \sqrt{2}$. \square

4.4 Fixing a cluster via fast filtering

In this section, we prove Lemma 13 by providing Fixing and giving its analysis. Before stating the algorithm, we provide a helper result which analyzes the effect of a “randomized dropout scheme” with respect to safe scores, and shows with high probability it is still safe.

Algorithm 10 RandDrop($T', \delta_{\text{rd}}, s$)

1: **Input:** $T' \subseteq T$, $\delta_{\text{rd}} \in (0, 1)$, 4γ -safe scores $\{s_i\}_{i \in T'}$ with respect to $w := \frac{1}{n} \mathbb{1}_{T'}$ such that $s_{\max} := \max_{i \in T'} s_i \leq 24|T' \cap S|$, and

$$\sum_{i \in T'} \frac{1}{|T'|} s_i \geq 288\gamma \log\left(\frac{2}{\delta_{\text{rd}}}\right). \quad (30)$$

2: **Output:** With failure probability $\leq \delta_{\text{rd}}$, outputs $T'' \subseteq T'$ such that if w is γ -saturated, then $\frac{1}{n} \mathbb{1}_{T''}$ is γ -saturated.

3: $T'' \leftarrow \emptyset$

4: **for** $i \in T'$ **do**

5: $T'' \leftarrow T'' \cup \{X_i\}$ with probability $1 - \frac{s_i}{s_{\max}}$

6: **end for**

7: **return** T''

In other words, RandDrop removes points from T' with probability proportional to their score.

Lemma 16. *The output of RandDrop satisfies the guarantees given in Line 2 of Algorithm 10.*

Proof. For all $i \in T'$, let Z_i be the random variable defined as

$$Z_i = \begin{cases} 1 & \text{with probability } \frac{s_i}{s_{\max}} \\ 0 & \text{with probability } 1 - \frac{s_i}{s_{\max}} \end{cases}.$$

Note that the number of points removed from T' and $T' \cap S$ are respectively $\sum_{i \in T'} Z_i$ and $\sum_{i \in T' \cap S} Z_i$. We now obtain high-probability bounds on both of these totals.

First, we lower bound $\sum_{i \in T'} Z_i$. Observe that $\mathbb{E} [\sum_{i \in T'} Z_i] = \sum_{i \in T'} \frac{s_i}{s_{\max}}$, and each Z_i is Bernoulli. Thus we can apply a Chernoff bound to obtain

$$\Pr \left[\sum_{i \in T'} Z_i < \frac{1}{2} \sum_{i \in T'} \frac{s_i}{s_{\max}} \right] \leq \exp \left(-\frac{1}{8} \sum_{i \in T'} \frac{s_i}{s_{\max}} \right) \leq \frac{\delta_{\text{rd}}}{2},$$

where we used $s_{\max} \leq 24|T'|$ and the assumed lower bound (30) to conclude

$$\sum_{i \in T'} \frac{s_i}{s_{\max}} \geq \sum_{i \in T'} \frac{s_i}{24|T'|} \geq 8 \log \left(\frac{2}{\delta_{\text{rd}}} \right).$$

Next, we upper bound $\sum_{i \in T' \cap S} Z_i$. We claim with failure probability at most $\frac{\delta_{\text{rd}}}{2}$,

$$\sum_{i \in T' \cap S} Z_i \leq \frac{1}{2\gamma} \frac{|T' \cap S|}{|T'|} \sum_{i \in T'} \frac{s_i}{s_{\max}}. \quad (31)$$

Define $\mu := \sum_{i \in T' \cap S} \frac{s_i}{s_{\max}}$ to be the expectation of the left hand side of (31), and set

$$\Delta := \frac{1}{\mu} \left(\frac{1}{2\gamma} \frac{|T' \cap S|}{|T'|} \sum_{i \in T'} \frac{s_i}{s_{\max}} \right) - 1$$

so that $(1 + \Delta)\mu$ is the right hand side of (31). Recall that we assumed that $\{s_i\}_{i \in T}$ were 4γ -safe; rearranging this definition (cf. Definition 3) yields

$$\mu \leq \frac{1}{4\gamma} \cdot \frac{|T' \cap S|}{|T'|} \sum_{i \in T'} \frac{s_i}{s_{\max}} \implies \Delta\mu = \frac{1}{2\gamma} \frac{|T' \cap S|}{|T'|} \sum_{i \in T'} \frac{s_i}{s_{\max}} - \mu \geq \frac{1}{4\gamma} \cdot \frac{|T' \cap S|}{|T'|} \sum_{i \in T'} \frac{s_i}{s_{\max}}.$$

However, since $\frac{|T' \cap S|}{s_{\max}} \geq \frac{1}{24}$ by assumption, we use (30) and the above equation to conclude

$$\Delta\mu \geq 3 \log \left(\frac{2}{\delta_{\text{rd}}} \right).$$

Finally, a Chernoff bound shows the failure probability of (31) is at most $\exp \left(-\frac{\Delta\mu}{3} \right) \leq \frac{\delta_{\text{rd}}}{2}$, as desired. Thus with probability at least $1 - \delta_{\text{rd}}$,

$$\sum_{i \in T' \cap S} \frac{1}{|T' \cap S|} Z_i \leq \frac{1}{\gamma} \sum_{i \in T'} \frac{1}{|T'|} Z_i.$$

Now observe that the $\{Z_i\}_{i \in T'}$ meet the definition of γ -safe scores (Definition 3). Thus, Lemma 1 applies with weights $\frac{1}{n}\mathbb{1}_{T'}$ and $\frac{1}{n}\mathbb{1}_{T''}$ and we obtain the conclusion. \square

We are now ready to state the algorithm **Fixing** and prove its guarantees in Lemma 13.

Algorithm 11 **Fixing**($T_{\text{in}}, \alpha, v, \delta, R$)

- 1: **Input:** $T_{\text{in}} \subseteq T$, $\alpha \in (0, \frac{1}{2})$, $v \in \mathbb{R}^d$, $\delta \in (0, 1)$, $R \in \mathbb{R}_{\geq 0}$ satisfying (for a sufficiently large constant)

$$R = \Omega \left(\sqrt{\gamma \log \left(\frac{\log d}{\delta} \right)} \right),$$

such that for T_{mid} defined in Algorithm 8, $\langle \widetilde{\text{Cov}}_{\frac{1}{n}\mathbb{1}}(T_{\text{mid}}), vv^\top \rangle \leq \frac{1}{8}R^2 \|v\|_2^2$

- 2: **Output:** Outputs $T_{\text{out}} \subset T_{\text{in}}$ with

$$\langle \widetilde{\text{Cov}}_{\frac{1}{n}\mathbb{1}}(T_{\text{out}}), vv^\top \rangle \leq \frac{1}{2}R^2 \|v\|_2^2.$$

If $\frac{1}{n}\mathbb{1}_{T_{\text{in}}}$ is γ -saturated, so is $\frac{1}{n}\mathbb{1}_{T_{\text{out}}}$, with failure probability $\leq \delta$.

- 3: **if** $\langle \widetilde{\text{Cov}}_{\frac{1}{n}\mathbb{1}}(T_{\text{in}}), vv^\top \rangle \leq \frac{1}{2}R^2 \|v\|_2^2$ **then**
4: **return** T_{in}
5: **end if**
6: $Y_i \leftarrow \langle v, X_i \rangle$ for all $i \in T_{\text{in}}$, $\tau_{\text{med}} \leftarrow \text{med}(\{Y_i \mid i \in T_{\text{in}}\})$
7: $I \leftarrow [\tau_{\text{med}} - c, \tau_{\text{med}} + c]$ is the smallest interval containing the $1 - \frac{\alpha}{4}$ quantiles of $\{Y_i \mid i \in T_{\text{in}}\}$ for $c \in \mathbb{R}_{\geq 0}$ and $2I \leftarrow [\tau_{\text{med}} - 2c, \tau_{\text{med}} + 2c]$
8: Define scores $\{s_i\}_{i \in T_{\text{in}}}$ by

$$s_i \leftarrow \begin{cases} 0 & Y_i \in I \\ (Y_i - (\tau_{\text{med}} - c))^2 & Y_i \leq \tau_{\text{med}} - c \\ (Y_i - (\tau_{\text{med}} + c))^2 & Y_i \geq \tau_{\text{med}} + c \end{cases}$$

- 9: $\delta_{\text{rd}} \leftarrow \frac{\delta}{\Omega(\log d \cdot \log \frac{d}{\delta})}$ for a sufficiently large constant
10: $T_{\text{out}} \leftarrow T_{\text{in}} \setminus \{X_i \mid s_i \geq 12 \|v\|_2^2 |S|\}$
11: **while** $\langle \widetilde{\text{Cov}}_{\frac{1}{n}\mathbb{1}}(T_{\text{out}}), vv^\top \rangle > \frac{1}{2}R^2 \|v\|_2^2$ **do**
12: $T_{\text{out}} \leftarrow \text{RandDrop}(T_{\text{out}}, \delta_{\text{rd}}, \frac{s}{\|v\|_2^2})$
13: **end while**
14: **return** T_{out}
-

Lemma 13. *There is an algorithm, **Fixing** (Algorithm 11), which takes as input $T_{\text{in}} \subseteq T$, $v \in \mathbb{R}^d$, and $R \in \mathbb{R}_{\geq 0}$ and produces T_{out} with the following guarantee with probability at least $1 - \delta$. Define T_{mid} as in Line 5 of Algorithm 8. Then if $\frac{1}{n}\mathbb{1}_{T_{\text{in}}}$ is γ -saturated, so is $\frac{1}{n}\mathbb{1}_{T_{\text{out}}}$, and if $\langle \widetilde{\text{Cov}}_{\frac{1}{n}\mathbb{1}}(T_{\text{mid}}), vv^\top \rangle \leq \frac{1}{8}R^2 \|v\|_2^2$, then $\langle \widetilde{\text{Cov}}_{\frac{1}{n}\mathbb{1}}(T_{\text{out}}), vv^\top \rangle \leq \frac{1}{2}R^2 \|v\|_2^2$.*

Proof. This proof proceeds in three parts. First, we show that whenever the average score is small:

$$\frac{1}{|T_{\text{out}}|} \sum_{i \in T_{\text{out}}} s_i \leq 288\gamma \log\left(\frac{2}{\delta_{\text{rd}}}\right) \|v\|_2^2,$$

then the check in Line 11 will fail and the algorithm will terminate. Next, we show calls to **RandDrop** meet its input criteria so its conclusion holds inductively (correctness of Line 10 is also handled here). Finally, we show that **Fixing** fails with probability at most δ . Assume throughout that $\frac{1}{n}\mathbb{1}_{T_{\text{in}}}$ is γ -saturated; else there is nothing to prove. We also use the following notation throughout:

$$\text{Var}_v(T') := \frac{1}{|T'|} \sum_{i \in T'} (Y_i - \mu_v(T'))^2, \text{ where } \mu_v(T') := \left\langle v, \mu_{\frac{1}{n}\mathbb{1}}(T') \right\rangle, \text{ for all } T' \subseteq T. \quad (32)$$

Small average score implies termination. We show that whenever the average score is small: $\frac{1}{|T_{\text{out}}|} \sum_{i \in T_{\text{out}}} s_i \leq 288\gamma \log\left(\frac{2}{\delta_{\text{rd}}}\right) \|v\|_2^2$, we terminate since this implies

$$\left\langle \widetilde{\text{Cov}}_{\frac{1}{n}\mathbb{1}}(T_{\text{out}}), vv^\top \right\rangle \leq \frac{1}{2} R^2 \|v\|_2^2.$$

To show this, we first prove

$$\begin{aligned} s_i &> \frac{1}{16} (Y_i - \mu_{2I})^2 \text{ for all } i \in T_{\text{out}}, Y_i \notin 2I, \\ \text{where } \mu_{2I} &:= \frac{1}{|T_{\text{in}} \cap \{i \mid Y_i \in 2I\}|} \sum_{i \in T_{\text{out}} \cap \{i \mid Y_i \in 2I\}} Y_i. \end{aligned} \quad (33)$$

In other words, μ_{2I} is the mean of points in $2I$. To see this, if $Y_i = \tau_{\text{med}} - 2c - \Delta$ for $\Delta > 0$,

$$s_i = (c + \Delta)^2, (Y_i - \mu_{2I})^2 \leq (4c + \Delta)^2 < 16s_i.$$

The case when $Y_i = \tau_{\text{med}} + 2c + \Delta$ is handled similarly, which covers all $Y_i \notin 2I$. Then, following notation (32),

$$\begin{aligned} \text{Var}_v(T_{\text{out}}) &= \sum_{i \in T_{\text{out}}} \frac{1}{|T_{\text{out}}|} (Y_i - \mu_v(T_{\text{out}}))^2 \leq \sum_{i \in T_{\text{out}}} \frac{1}{|T_{\text{out}}|} (Y_i - \mu_{2I})^2 \\ &= \sum_{i \in T_{\text{out}} \cap \{i \mid Y_i \in 2I\}} \frac{1}{|T_{\text{out}}|} (Y_i - \mu_{2I})^2 + \sum_{i \in T_{\text{out}} \cap \{i \mid Y_i \notin 2I\}} \frac{1}{|T_{\text{out}}|} (Y_i - \mu_{2I})^2 \\ &\leq \sum_{i \in T_{\text{out}} \cap \{i \mid Y_i \in 2I\}} \frac{1}{|T_{\text{out}}|} (Y_i - \mu_{2I})^2 + 16 \sum_{i \in T_{\text{out}}} \frac{1}{|T_{\text{out}}|} s_i \\ &\leq \sum_{i \in T_{\text{out}} \cap \{i \mid Y_i \in 2I\}} \frac{1}{|T_{\text{out}}|} (Y_i - \mu_{2I})^2 + O\left(\gamma \log\left(\frac{1}{\delta_{\text{rd}}}\right)\right) \|v\|_2^2. \end{aligned}$$

Here, the first line used Fact 2, the third used (33) and that all scores are nonnegative, and the

last used our assumption on the average score in T_{out} . Thus,

$$\begin{aligned}
\left\langle \widetilde{\text{Cov}}_{\frac{1}{n}\mathbb{1}}(T_{\text{out}}), vv^\top \right\rangle &= \sum_{i \in T_{\text{out}}} \frac{1}{n} (Y_i - \mu_v(T_{\text{out}}))^2 \\
&\leq \sum_{i \in T_{\text{out}} \cap \{i \mid Y_i \in 2I\}} \frac{1}{n} (Y_i - \mu_{2I})^2 + O\left(\gamma \log\left(\frac{1}{\delta_{\text{rd}}}\right)\right) \|v\|_2^2 \\
&= \left\langle \widetilde{\text{Cov}}_{\frac{1}{n}\mathbb{1}}(T_{\text{out}} \cap \{i \mid Y_i \in 2I\}), vv^\top \right\rangle + O\left(\gamma \log\left(\frac{1}{\delta_{\text{rd}}}\right)\right) \|v\|_2^2 \\
&\leq \frac{1}{8} R^2 \|v\|_2^2 + O\left(\gamma \log\left(\frac{1}{\delta_{\text{rd}}}\right)\right) \|v\|_2^2 \leq \frac{1}{2} R^2 \|v\|_2^2.
\end{aligned}$$

In the second line we used $n \geq |T_{\text{out}}|$ to handle the second term, the third line used the definition of $\widetilde{\text{Cov}}$, and the fourth line used the assumed bound on $\left\langle \widetilde{\text{Cov}}_{\frac{1}{n}\mathbb{1}}(T_{\text{mid}}), vv^\top \right\rangle$, and

$$\widetilde{\text{Cov}}_{\frac{1}{n}\mathbb{1}}(T_{\text{out}} \cap \{i \mid Y_i \in 2I\}) \preceq \widetilde{\text{Cov}}_{\frac{1}{n}\mathbb{1}}(T_{\text{mid}})$$

since $T_{\text{out}} \cap \{i \mid Y_i \in 2I\} \subseteq T_{\text{mid}}$ and Fact 2 implies that dropping terms from the covariance formula and shifting to the mean only decreases Loewner order. The last line used the lower bound on R .

Correctness of calls to RandDrop. We first bound the average score in $T_{\text{in}} \cap S$ at the beginning of the algorithm. Let $\text{Var}_v(T_{\text{in}} \cap S)$ denote the variance of $T_{\text{in}} \cap S$ in the direction v following (32). We claim that the mean of $T_{\text{in}} \cap S$ lies close to I : in particular,

$$\mu_{\frac{1}{n}\mathbb{1}}(T_{\text{in}} \cap S) \in \left[\tau_{\text{med}} - c - \sqrt{2\text{Var}_v(T_{\text{in}} \cap S)}, \tau_{\text{med}} + c + \sqrt{2\text{Var}_v(T_{\text{in}} \cap S)} \right]. \quad (34)$$

If this were not the case, we would have a contradiction:

$$\begin{aligned}
\text{Var}_v(T_{\text{in}} \cap S) &\geq \frac{1}{|T_{\text{in}} \cap S|} \sum_{i \in T_{\text{in}} \cap S \mid Y_i \in I} (Y_i - \mu_v(T_{\text{in}} \cap S))^2 \\
&> \frac{|T_{\text{in}} \cap S \cap \{i \mid Y_i \in I\}|}{|T_{\text{in}} \cap S|} (2\text{Var}_v(T_{\text{in}} \cap S)) \geq \text{Var}_v(T_{\text{in}} \cap S).
\end{aligned}$$

The second inequality used that every summand is at least $2\text{Var}_v(T_{\text{in}} \cap S)$ if (34) does not hold, and the last used that I contains a $1 - \frac{\alpha}{4}$ proportion of the points in T_{in} , and by Lemma 9 $T_{\text{in}} \cap S$ contains at least $\frac{\alpha}{2}$ of the points in T_{in} . Now using (34) and the definition of the scores,

$$\begin{aligned}
\sum_{i \in T_{\text{in}} \cap S} \frac{1}{|T_{\text{in}} \cap S|} s_i &\leq \sum_{i \in T_{\text{in}} \cap S} \frac{1}{|T_{\text{in}} \cap S|} \left(|Y_i - \mu_v(T_{\text{in}} \cap S)| + \sqrt{2\text{Var}_v(T_{\text{in}} \cap S)} \right)^2 \\
&\leq 6\text{Var}_v(T_{\text{in}} \cap S) \leq 12\|v\|_2^2.
\end{aligned} \quad (35)$$

In the last line, we used that $T_{\text{in}} \cap S$ contains at least half the points in S by Lemma 9, so Assumption 1 applies with a normalizing factor at most twice as large. This shows that Line 10 of Algorithm 11 preserves saturation, since it can only remove points in $T_{\text{in}} \setminus S$ (if any point in $T_{\text{in}} \cup S$ had a score larger than $12|S|\|v\|_2^2$, it would violate (35)). This also shows that if at any point in

running Algorithm 11 we have a γ -saturated subset $T_{\text{out}} \subset T_{\text{in}}$, then

$$\sum_{i \in T_{\text{out}} \cap S} \frac{1}{|T_{\text{out}} \cap S|} s_i \leq 24 \|v\|_2^2. \quad (36)$$

This is because compared to (35), we can at most double the normalizing factor by Lemma 9, and all scores are nonnegative. By combining with the first part of this proof, whenever Line 11 passes,

$$\frac{1}{|T_{\text{out}}|} \sum_{i \in T_{\text{out}}} s_i > 288\gamma \log\left(\frac{2}{\delta_{\text{rd}}}\right) \|v\|_2^2, \quad (37)$$

and hence the scores are 4γ -safe as required by **RandDrop**. The second requirement of **RandDrop** is that $s_{\text{max}} \leq 24|T_{\text{out}} \cap S| \|v\|_2^2$, which is taken care of by Line 10 as $|S| \leq 2|T_{\text{out}} \cap S|$ by Lemma 9. Finally, Line 11 implies (37) by the first part of this proof, which is the third condition of **RandDrop**.

Bounding failure probability. We bound the failure probability in two steps. First, we show with probability at least $1 - \frac{\delta}{2}$, there are at most (for a suitable constant)

$$N := O\left(\log d \cdot \log \frac{d}{\delta}\right)$$

calls to **RandDrop**. Then, we union bound to show that all these calls to **RandDrop** pass with probability at least $1 - \frac{\delta}{2}$. Combining gives the overall failure probability to **Fixing**.

To see the bound on N , observe that after Line 10, the largest score is at most $O(\|v\|_2^2 d)$, and the algorithm ends when the largest score is at most a constant (since then (37) clearly fails, at which point we terminate on Line 11 by the first part of this proof). Thus, the largest score can only halve at most $O(\log d)$ times. However, observing the implementation of **RandDrop**, any point with score at least half the largest is dropped with probability at least $\frac{1}{2}$, and hence after $O(\log \frac{d}{\delta})$ rounds, the largest score will halve with probability at least $1 - \frac{\delta}{\Omega(\log d)}$. Union bounding over all the phases of halving the max score implies after N loops the algorithm terminates with probability $1 - \frac{\delta}{2}$.

Since there are at most N calls to **RandDrop**, it suffices to set $\delta_{\text{rd}} = \frac{\delta}{2N}$ to check that all calls to **RandDrop** pass with probability $1 - \frac{\delta}{2}$. If all calls pass, we have the desired conclusion. \square

4.5 Runtime analysis

We now give a runtime bound for **1DPartition**, and use it to obtain a similar bound on **Partition**.

Lemma 17. *Let $n' := |T'|$, where T' is the input to **1DPartition**. Then **1DPartition** can be implemented to run in time*

$$O\left(n'd + (n')^{1+\beta} \left(\frac{1}{\beta} \log \log d \cdot \log\left(\frac{1}{\alpha\beta}\right) + \log d \log \frac{d}{\delta}\right)\right).$$

Proof. We begin by computing all the points $Y_i := \langle v, X_i \rangle$ for $i \in T'$ and sorting them, and store all quantiles (i.e. the number of points less than any given Y_i), which takes time $O(n'd + n' \log n')$.

Next, we bound the cost of running **Fixing** on an input T_{in} of size n_{in} . Given access to quantile information, and since **Fixing** is only ever called on a set which is formed after applying some number of splits to the original dataset T' , it is straightforward to implement Lines 3-10 in time $O(n_{\text{in}})$. Moreover, each loop in Lines 11-13 costs $O(n_{\text{in}})$ time, and by the proof of Lemma 13, there are at

most $O(\log d \log \frac{d}{\delta})$ loops. Thus overall the runtime of **Fixing** is

$$O\left(n_{\text{in}} \log d \log \frac{d}{\delta}\right).$$

We now consider the cost of running **SplitOrTailBound** with a given threshold τ_0 . If τ_0 does not lie in the interval of $\{Y_i\}_{i \in T_{\text{in}}}$, then the runtime is $O(1)$. Otherwise, consider the case $\tau_0 \geq \tau_{\text{med}}$ (note τ_{med} can be computed in constant time given quantile information). Since at least one point is larger than τ_0 , $g_0 \geq \frac{1}{n}$, and hence (26) shows the number of threshold checks is bounded by $O(\frac{1}{\beta} \log \log d)$. Each threshold check takes constant time (we just need to compute the cardinalities of the induced $T_{\text{out}}^{(1)}, T_{\text{out}}^{(2)}$) and computing the next g_j and r_j takes constant time given quantile information, so the cost of **SplitOrTailBound** is

$$O\left(\frac{1}{\beta} \log \log d\right).$$

Correspondingly, the cost of each run of Lines 8-11 of **SplitOrCluster** is bounded by

$$O\left(\frac{1}{\beta} \log \log d \cdot \log\left(\frac{1}{\alpha\beta}\right)\right).$$

Now, consider the structure of **1DPartition**. Lemma 11 shows that there are at most $(n')^{1+\beta}$ split steps total, so the total cost of all split steps (which run Lines 8-11 of **SplitOrCluster**) is

$$O\left(\frac{(n')^{1+\beta}}{\beta} \log \log d \cdot \log\left(\frac{1}{\alpha\beta}\right)\right).$$

Finally, consider all nodes in the **1DPartition** which are parents of leaves. The sums of cardinalities of all such nodes is bounded by $(n')^{1+\beta}$, so the cost of running **Fixing** on all these nodes is

$$O\left((n')^{1+\beta} \log d \log \frac{d}{\delta}\right).$$

□

As an immediate corollary, we obtain a runtime bound on **Partition**.

Corollary 2. *Let $n_p := |T_p|$ for some $T_p \subseteq T$. **Partition** called on input T_p with parameter C can be implemented to run in time*

$$O\left(n_p^{1+\beta} d \log d \log \frac{d}{\delta} + n_p^{1+\beta} \left(\frac{1}{\beta} \log \log d \cdot \log\left(\frac{1}{\alpha\beta}\right) \log \frac{d}{\delta} + \log d \log^2 \frac{d}{\delta}\right)\right).$$

Proof. The proof is identical to Corollary 1, where we use Lemma 17 to bound the cost over all elements of each \mathcal{S}_j , and there are $N_{\text{dir}} = \Theta(\log \frac{d}{\delta})$ calls to **1DPartition**. □

4.6 Full bounded covariance algorithm

Finally, we give our full algorithm for list-decodable mean estimation under Assumption 1. As in Section 3.3, we will reduce to the bounded diameter case via the algorithm **NaiveCluster** (cf. Lemma 7); we reproduce its guarantees for arbitrary failure probabilities as **NaiveClusterPlus**.

Lemma 18 (Lemma 12, [DKK⁺20b]). *There is a randomized algorithm, `NaiveClusterPlus`(T, δ), 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 $1 - \delta$, all of S is contained in the same subset, and every subset has diameter bounded by $O(\frac{d^8}{\delta^2})$. The runtime of `NaiveClusterPlus` is $O(nd + n \log n)$.*

We also require a post-processing procedure to reduce the list size, which we call `IteratePostProcess`. We state its guarantees in Lemma 19, and defer the description and analysis to Section 4.7.

Lemma 19. *There is an algorithm, `IteratePostProcess` (Algorithm 14), which takes as input T satisfying Assumption 1 and a list $L \subset \mathbb{R}^d$ of length $m \leq n$ such that*

$$\min_{\hat{\mu} \in L} \|\hat{\mu} - \mu^*\|_2 \leq \Delta, \Delta = \Omega\left(\frac{1}{\sqrt{\alpha}}\right)$$

and returns with probability at least $1 - \delta$ a subset $L' \subset L$ of size $O(\frac{1}{\alpha})$ such that $\min_{\hat{\mu} \in L'} \|\hat{\mu} - \mu^\|_2 = O(\Delta)$, within runtime*

$$O\left((m + n)d + \alpha m^2 n \log \frac{d}{\delta}\right).$$

Algorithm 12 `FastMultifilter`(T, α, δ, β)

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

$$\|\hat{\mu} - \mu^*\|_2 = O\left(\sqrt{\frac{\log(\frac{1}{\alpha})}{\alpha}} \cdot \max\left(\frac{1}{\beta} \sqrt{\log\left(\frac{1}{\alpha\beta}\right)}, \sqrt{\log \log d}\right)\right). \quad (38)$$

- 3: $\delta_{\text{outer}} \leftarrow \frac{1}{2}$
 - 4: $N_{\text{runs}} \leftarrow \lceil 2 \log \frac{2}{\delta} \rceil$
 - 5: $L \leftarrow \emptyset$
 - 6: **for** $j \in [N_{\text{runs}}]$ **do**
 - 7: $\{T'_i\}_{i \in [k]} \leftarrow \text{NaiveClusterPlus}(T, \frac{\delta_{\text{outer}}}{3})$
 - 8: $\alpha_i \leftarrow \frac{|T|}{|T'_i|} \alpha$ for all $i \in [k]$
 - 9: $L \leftarrow L \cup \text{IteratePostProcess}\left(T, \bigcup_{i \in [k]} \text{FastMultifilterBoundedDiameter}(T'_i, \alpha_i, \frac{\delta_{\text{outer}}}{3}, \beta), \frac{\delta_{\text{outer}}}{3}\right)$
 - 10: **end for**
 - 11: **return** `IteratePostProcess` $(T, L, \frac{\delta}{2})$
-

Proposition 2. `FastMultifilterBoundedDiameter` meets its output specifications with probability at least $1 - \delta$, within runtime

$$O\left(n^{1+\beta} d \log^2 d \log^2 \frac{d}{\delta} + n^{1+\beta} \left(\frac{1}{\beta} \log \log d \cdot \log\left(\frac{1}{\alpha\beta}\right) \log d \log^2 \frac{d}{\delta} + \log^2 d \log^3 \frac{d}{\delta}\right)\right).$$

Proof. The proof of the error rate is identical to that in Proposition 1, where the initial potential Φ_0 is bounded by $(\frac{d}{\delta})^{O(\log d)}$ via Lemma 18, which implies the operator norm of $\widetilde{\text{Cov}}_{\frac{1}{n}\mathbb{1}}(T')$ for every node T' on layer D is $O(R^2)$, and inductively at least one such node has $|T' \cap S| \geq \frac{1}{2}|S|$ by virtue

Algorithm 13 FastMultifilterBoundedDiameter(T, α, δ, β)

1: **Input:** $T \subset \mathbb{R}^d$, $|T| = n$ satisfying Assumption 1 with parameter $\alpha \in (0, \frac{1}{2})$, $\delta \in (0, 1)$, $\beta \in (0, 1]$

2: **Output:** With failure probability $\leq \delta$: L_{out} with $|L_{\text{out}}| = O(\frac{n^\beta}{\alpha})$ such that some $\hat{\mu} \in L_{\text{out}}$ satisfies

$$\|\hat{\mu} - \mu^*\|_2 = O\left(\sqrt{\frac{\log(\frac{1}{\alpha})}{\alpha}} \cdot \max\left(\frac{1}{\beta} \sqrt{\log\left(\frac{1}{\alpha\beta}\right)}, \sqrt{\log\left(\frac{\log d}{\delta}\right)}\right)\right).$$

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

4: For sufficiently large constants,

$$R \leftarrow \Theta\left(\max\left(\frac{1}{\beta} \cdot \sqrt{\log\left(\frac{1}{\alpha}\right) \log\left(\frac{1}{\alpha\beta}\right)}, \sqrt{\log\left(\frac{1}{\alpha}\right) \log\left(\frac{d}{\delta}\right)}\right)\right), D \leftarrow \Theta\left(\log d \log \frac{d}{\delta}\right)$$

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

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

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

8: Append all elements of $\text{Partition}(T', \alpha, \frac{\delta}{n^{1+\beta}D}, \beta, R)$ to $L^{(\ell)}$

9: **end for**

10: **end for**

11: **return** List of empirical means of all sets in $L^{(D)}$ with size at least $\frac{\alpha n}{2}$

of being γ -saturated and applying Lemma 9. The failure probability follows since Partition is called at most $n^{1+\beta}D$ times, as there are at most $n^{1+\beta}$ elements of each $L^{(\ell)}$. Finally, the list size follows since Lemma 9 implies every leaf node contains $\frac{\alpha n}{2}$, but the total size across leaves is at most $n^{1+\beta}$.

Finally, to obtain the runtime bound we can sum the guarantee of Corollary 2 across each of the D layers, and use the potential to bound the sum of all $n_p^{1+\beta}$ across the layer. \square

We are now ready to state our main claim on list-decodable mean estimation. For simplicity, we state the result for $\beta \geq \frac{1}{\log d}$, as otherwise there are no runtime or statistical gains asymptotically.

Theorem 6. For $\frac{1}{\log d} \leq \beta \leq 1$, and $\log^{\Omega(1)}(d) \leq \alpha^{-1} \leq d$, FastMultifilter returns a list of size $O(\frac{1}{\alpha})$ such that

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

with probability at least $1 - \delta$, within runtime

$$O\left(n^{1+2\beta} d \log^4 d \log \frac{1}{\delta} + n d \log^2 \frac{1}{\delta} \log \frac{d}{\delta}\right).$$

Proof. We first analyze Lines 6-10 of FastMultifilter. We claim that each of the N_{runs} times these lines run, there is a $\geq \frac{1}{2}$ probability that some $\hat{\mu}$ will be added to L satisfying (38), within runtime

$$O\left(n^{1+2\beta} d \log^4 d + n^{1+\beta} \left(\frac{1}{\beta} \log \log d \cdot \log\left(\frac{1}{\alpha\beta}\right) \log^3 d + \log^5 d\right)\right) = O\left(n^{1+2\beta} d \log^4 d\right).$$

To see this, we apply Proposition 2 to the relevant call of `FastMultifilterBoundedDiameter`. The correctness follows identically to the proof of Theorem 5, except that the size of the list of candidate means $\bigcup_{i \in [k]} \text{FastMultifilterBoundedDiameter}(T'_i, \alpha_i, \frac{\delta_{\text{outer}}}{3}, \beta)$ is $m = O(\frac{n^\beta}{\alpha})$. By Lemma 19, after applying `IteratePostProcess` the error rate is not affected by more than a constant, and the list size is $O(\frac{1}{\alpha})$. The runtime of this last step is dominated by $O(\alpha m^2 n \log d) = O(n^{1+2\beta} d \log d)$.

Next, this implies that after all runs of Lines 6-10 have finished running (with independent internal randomness), there is a $\geq 1 - \frac{\delta}{2}$ probability that L contains an element $\hat{\mu}$ satisfying (38). At this point, the size of the list is $m = O(\frac{\log \delta^{-1}}{\alpha})$, so Line 11 takes time $O(nd \log^2 \frac{1}{\delta} \log \frac{d}{\delta})$ by Lemma 19. \square

This theorem, combined with the previously discussed fact that we can assume that $\alpha \in [1/d, 1/\log^{\Omega(1)} d]$, gives our desired conclusion.

4.7 Cleaning up the list

In this section, we provide the subroutine `IteratePostProcess` used in `FastMultifilter`, and prove Lemma 19, which shows correctness of this subroutine. At a high level, `IteratePostProcess` first finds a greedy cover of the input list L at distance $O(\Delta)$. Then, while the greedy cover has size at least $4k$ for $k := \lceil \frac{1}{\alpha} \rceil$, it iteratively prunes away $2k$ out of $4k$ hypotheses by testing that there are enough datapoints closest to retained hypotheses; otherwise, it returns the greedy cover.

Algorithm 14 `IteratePostProcess`($T, \alpha, L, \delta, \Delta$)

1: **Input:** $T \subset \mathbb{R}^d$, $|T| = n$ satisfying Assumption 1 with parameter $\alpha \in (0, \frac{1}{2})$, $\delta \in (0, 1)$, L with $|L| = m \leq n$ such that

$$\min_{\hat{\mu} \in L} \|\hat{\mu} - \mu^*\|_2 \leq \Delta, \Delta = \Omega\left(\frac{1}{\sqrt{\alpha}}\right).$$

2: **Output:** With failure probability $\leq \delta$: $L' \subset L$ with $|L'| = O(\frac{1}{\alpha})$ such that

$$\min_{\hat{\mu} \in L'} \|\hat{\mu} - \mu^*\|_2 = O(\Delta).$$

3: $\mathbf{G} \in \mathbb{R}^{d \times c} \leftarrow$ entrywise $\pm \frac{1}{\sqrt{c}}$ uniformly at random, for $c = \Theta(\log \frac{d}{\delta})$ (Johnson-Lindenstrauss matrix [Ach03])

4: $k \leftarrow \lceil \frac{1}{\alpha} \rceil$

5: $L' \leftarrow$ maximal subset of L such that $\forall \hat{\mu} \neq \hat{\mu}' \in L', \|\mathbf{G}^\top(\hat{\mu} - \hat{\mu}')\|_2 \geq 5\Delta$

6: **while** $|L'| \geq 4k$ **do**

7: $L_{\text{head}} \leftarrow$ first $4k$ elements of L'

8: $L_{\text{prune}} \leftarrow$ elements of L_{head} which are nearest neighbors of $< \frac{\alpha n}{2}$ elements of T , where $\hat{\mu} \in L_{\text{head}}$ is the nearest neighbor of $X_i \in T$ if $\|\mathbf{G}^\top(\hat{\mu} - X_i)\|_2$ is minimal amongst L_{head}

9: $L \leftarrow L \setminus L_{\text{prune}}$

10: $L' \leftarrow$ maximal subset of L such that $\forall \hat{\mu} \neq \hat{\mu}' \in L', \|\mathbf{G}^\top(\hat{\mu} - \hat{\mu}')\|_2 \geq 5\Delta$

11: **end while**

12: **return** L'

Lemma 19. *There is an algorithm, `IteratePostProcess` (Algorithm 14), which takes as input T*

satisfying Assumption 1 and a list $L \subset \mathbb{R}^d$ of length $m \leq n$ such that

$$\min_{\hat{\mu} \in L} \|\hat{\mu} - \mu^*\|_2 \leq \Delta, \quad \Delta = \Omega\left(\frac{1}{\sqrt{\alpha}}\right)$$

and returns with probability at least $1 - \delta$ a subset $L' \subset L$ of size $O(\frac{1}{\alpha})$ such that $\min_{\hat{\mu} \in L'} \|\hat{\mu} - \mu^*\|_2 = O(\Delta)$, within runtime

$$O\left(\left((m+n)d + \alpha m^2 n\right) \log \frac{d}{\delta}\right).$$

Proof. We first prove correctness, and then prove the runtime bound.

Correctness. By the Johnson-Lindenstrauss lemma as analyzed in [Ach03], with probability at least $1 - \delta$ every pair of points in $L \cup T \cup \{\mu^*\}$ has their distance preserved to a 1.1 multiplicative factor under multiplication by \mathbf{G}^\top . Condition on this event for the remainder of the proof.

Let $\bar{\mu}$ be the element of the input L which is guaranteed to be within distance Δ of μ^* . We will first show that $\bar{\mu}$ is never removed from L by the loop in Lines 6-11. If $\bar{\mu}$ is not a part of L_{head} in a given loop, clearly this is true, so suppose $\bar{\mu} \in L_{\text{head}}$, and let $\hat{\mu}$ be some other element in L_{head} with $\|\mathbf{G}^\top(\bar{\mu} - \hat{\mu})\|_2 \geq 5\Delta$; by definition of L_{head} as a subset of L' , all such $\hat{\mu}$ satisfy this. Our goal will be to show that at least half of the points in S have nearest neighbor $\bar{\mu}$; to do so, it suffices to show that $\|\mathbf{G}^\top(X_i - \hat{\mu})\|_2 \geq \|\mathbf{G}^\top(X_i - \bar{\mu})\|_2$ with probability at most $\frac{1}{8k}$ over $i \sim S$ for each $\hat{\mu} \neq \bar{\mu}$, so the $< 4k$ other hypotheses in L_{head} can only remove $\frac{\alpha m}{2}$ of the points in S from having nearest neighbor $\bar{\mu}$, and hence $\bar{\mu}$ will not be pruned.

We now show the key claim: that for all $\hat{\mu} \neq \bar{\mu} \in L_{\text{head}}$,

$$\Pr_{i \sim \text{unif} S} \left[\|\mathbf{G}^\top(X_i - \hat{\mu})\|_2 \leq \|\mathbf{G}^\top(X_i - \bar{\mu})\|_2 \right] \leq \frac{1}{8k}.$$

Observe that by the triangle inequality, for any $i \in S$ satisfying the event above,

$$\begin{aligned} 2 \|\mathbf{G}^\top(X_i - \bar{\mu})\|_2 &\geq \|\mathbf{G}^\top(X_i - \bar{\mu})\|_2 + \|\mathbf{G}^\top(X_i - \hat{\mu})\|_2 \\ &\geq \|\mathbf{G}^\top(\hat{\mu} - \bar{\mu})\|_2 \geq 5\Delta \\ \implies \|X_i - \mu^*\|_2 &\geq \|X_i - \bar{\mu}\|_2 - \|\bar{\mu} - \mu^*\|_2 \geq \Delta. \end{aligned}$$

Here we used $\|X_i - \bar{\mu}\|_2 \geq \frac{1}{1.1} \|\mathbf{G}^\top(X_i - \bar{\mu})\|_2 \geq 2\Delta$, and $\|\bar{\mu} - \mu^*\|_2 \leq \Delta$ by assumption. By Chebyshev's inequality and Assumption 1, we conclude for sufficiently large $\Delta = \Omega(\frac{1}{\sqrt{\alpha}})$,

$$\Pr_{i \sim \text{unif} S} \left[\|\mathbf{G}^\top(X_i - \hat{\mu})\|_2 \leq \|\mathbf{G}^\top(X_i - \bar{\mu})\|_2 \right] \leq \Pr_{i \sim \text{unif} S} [\|\bar{\mu} - \mu^*\|_2 \geq \Delta] \leq \frac{1}{8k}.$$

Finally, we have shown that when the algorithm exits on Line 12, L' is a maximal separated subset of a pruned list L containing $\bar{\mu}$. If $\bar{\mu} \in L'$, the guarantee is immediate; otherwise, there must have been some other $\hat{\mu} \in L'$ with $\|\mathbf{G}^\top(\hat{\mu} - \bar{\mu})\|_2 \leq 5\Delta$, else $\bar{\mu}$ would have been added. For this $\hat{\mu}$,

$$\|\hat{\mu} - \mu^*\|_2 \leq 1.1 \|\mathbf{G}^\top(\hat{\mu} - \mu^*)\|_2 \leq 1.1 \|\mathbf{G}^\top(\hat{\mu} - \bar{\mu})\|_2 + 1.1 \|\mathbf{G}^\top(\bar{\mu} - \mu^*)\|_2 = O(\Delta).$$

The list size bound follows from Line 6, as the returned L' has at most $4k$ elements.

Runtime. First, the cost of computing all projections $\mathbf{G}^\top X$ for $X \in L \cup T$ is $O((m+n)d \log \frac{d}{\delta})$. Next,

Lines 6-11 can only be looped over at most $O(\alpha m)$ times, since every loop removes $2k$ elements from L which originally has size m . It remains to argue about the complexity of each loop.

The cost of computing a maximal subset in Lines 5 and 10 is $O(m^2 \log \frac{d}{\delta})$, since distance comparisons under multiplication by \mathbf{G} take $O(\log \frac{d}{\delta})$ and it suffices to greedily loop over the list. Similarly, the cost of computing nearest neighbors of all elements in T in Line 8 is $O(mn \log \frac{d}{\delta})$, which is the dominant term. Combining these components yields the claim. \square

4.8 (Slightly) improving the error rate

We give a brief discussion of how it is possible to shave a $\sqrt{\log \alpha^{-1}}$ factor from the error guarantees of Theorem 6, bringing it to within a $\sqrt{\log \alpha^{-1}}$ factor from optimal when β is a constant. At a high level, this extraneous factor is due to our insistence that all weight removals be γ -safe, for some $\gamma = \Theta(\log \alpha^{-1})$. This causes the thresholds required for termination of our subroutines (e.g. for `SplitOrCluster` to enter the `Fixing` stage) to be inflated by roughly a γ factor.

We can remove this factor by using 2-safe scores instead of $\Theta(\log \alpha^{-1})$ -safe scores, an idea introduced by [DKK⁺20b] to obtain improved estimation rates over the multifilter of [DKK20a]. The idea is to restart the algorithm in *phases*, where each phase corresponds to the total maintained weight being stable up to a factor of 2 (in our case, this means subset sizes are stable up to factors of 2).

We now summarize the changes to our algorithm. We will run the “outer loop” subroutine `FastMultifilterBoundedDiameter` (which can be viewed as constructing a multifilter tree) up until a depth of $O(\log^2 d \log \frac{1}{\alpha})$ is reached, in batches of $O(\log^2 d)$ each corresponding to a stable phase. Each batch will either meet the relevant termination condition (bounded covariance, such that e.g. (19) is trivially satisfied), or make progress by entering the next phase via safe weight removals.

Correspondingly, the condition (15) required to make improvements on the potential will be scaled differently, according to the size of the relevant set T_p at some node p . In particular, suppose we are in a phase when $\frac{1}{2}n' < |T_p| \leq n'$. Then we will aim to guarantee

$$\langle \mathbf{Y}_p^2, \mathbf{M}_{c_\ell} \rangle \leq R^2 \sqrt{\frac{n'}{n}} \text{Tr}(\mathbf{Y}_p^2),$$

where R has the same value as in `FastMultifilterBoundedDiameter` up to removing a $\sqrt{\log \alpha^{-1}}$. This allows us to terminate when the operator norm of some (unnormalized) $\widetilde{\text{Cov}}$ matrix is $O(R^2 \sqrt{\frac{n'}{n}})$, at which point Lemma 2 concludes a distance of

$$O\left(\sqrt{R^2 \sqrt{\frac{n}{n'}} \cdot \frac{n'}{|T_p \cap S|}}\right) = O\left(R \cdot \frac{\sqrt[4]{n'n}}{\sqrt{|T_p \cap S|}}\right) = O\left(\frac{R}{\sqrt{\alpha}}\right),$$

where we use that 2-saturation of T_p implies $\frac{|T_p \cap S|}{n} \geq \alpha \sqrt{\frac{n'}{2n}}$ (cf. Lemma 1, [DKK⁺20b]).

Because of the complications this type of argument introduces, e.g. every one of our subroutines needs an extra exit condition (when the maintained subset enters the next phase), we omit a formal treatment in this paper. However, we remark that to remove the entire $\log \alpha^{-1}$ factor from our error likely requires new ideas. This is because both branches of our key subroutine `SplitOrCluster`, namely `SplitOrTailBound` and `Fixing`, require this overhead. The former is because integrating variance tail bounds decaying as $t \cdot \frac{1}{t^2}$ out to $O(\alpha)$ quantiles (cf. Lemma 14) introduces a gap of $\log(\frac{1}{\alpha})$. The latter is because we employ randomize dropout to maintain subsets (rather than weights); our

dropout method requires a threshold of roughly $\log \log d$ (cf. Lemma 16) to obtain high-probability guarantees after union bounding $\text{polylog}(d)$ times. For $\alpha^{-1} = \log^{\Omega(1)} d$, this is again a $\log(\frac{1}{\alpha})$ gap.

5 Clustering mixture models

We define a *mixture model* to be a mixture $\sum_{i \in [k]} \alpha_i \mathcal{D}_i$ where $\{\alpha_i\}_{i \in [k]} \in \mathbb{R}_{\geq 0}^k$, $\sum_{i \in [k]} \alpha_i = 1$, and all \mathcal{D}_i are supported on \mathbb{R}^d . In Sections 5.1 and 5.2 we handle the case where all distributions are sub-Gaussians: \mathcal{D}_i has mean μ_i , and sub-Gaussian parameter ≤ 1 in all directions (cf. Section 2.1). We begin with the uncorrupted, uniform mixture case as a warmup in Section 5.1, and show how our method tolerates non-uniformity and adversarial outliers in Section 5.2. We then give a simple extension of our algorithm to handle mixtures where each component has bounded fourth moment in Section 5.3, and finally tackle the case of bounded-covariance mixture models in Section 5.4.

Broadly, all of our clustering algorithms follow the same design framework. We first demonstrate using concentration and existence of a good hypothesis (the list-decodable learning guarantee), that the “nearest hypothesis” to every non-adversarial point is close to the true mean. We next prune our hypotheses down by only keeping those with a substantial number of nearby points; by arguing that the number of adversarial points (or points that appear adversarial due to anti-concentration) is small, no large “coalition” of bad points can be formed, and hence all kept hypotheses are near a true mean. Finally, assuming enough separation between true means, we can define a partition of the points based on their nearest hypotheses. In Section 5.4, we will use a more direct clustering process in the subspace spanned by candidates, combined with fast projected distance approximations, in order to obtain a tighter separation guarantee.

Throughout, we will frequently use that by Chernoff, the sum of any Bernoulli random variables whose expectation is $\Omega(d)$ will deviate from its expectation by at most any multiplicative constant with probability at least $1 - \exp(-\Omega(d))$. For example, for a dataset of size $n = \Omega(dk)$ drawn from a uniform mixture $\sum_{i \in [k]} \frac{1}{k} \mathcal{D}_i$, each component \mathcal{D}_i will contribute between $0.99 \frac{n}{k}$ and $1.01 \frac{n}{k}$ points with probability at least $1 - k \exp(-\Omega(d))$, or more simply $1 - \exp(-\Omega(d))$ for $k = O(d)$.

5.1 Clustering uniform (sub-)Gaussian mixture models

We first consider the simple setting where all $\alpha_i = \frac{1}{k}$ and all \mathcal{D}_i has mean μ_i and sub-Gaussian parameter ≤ 1 in all directions. We assume access to a *list-decoding* algorithm \mathcal{A} which returns a list L of length $O(k)$, such that for each $i \in [k]$, L contains $\hat{\mu}_i$ such that $\|\hat{\mu}_i - \mu_i\|_2 \leq \Delta$, for some $\Delta = \Omega(\sqrt{k})$ (in particular, **FastMultifilter** suffices for \mathcal{A}). Finally, we assume access to a dataset $\mathbf{X} = \{X_j\}_{j \in [n]}$ of size $n = \Theta(dk)$ drawn from the mixture model independently of \mathcal{A} , where we say that each X_j is “associated with” an index $i \in [k]$ (designating the component it is drawn from). We will now demonstrate how to cluster a dataset using calls to \mathcal{A} , assuming a sufficiently large separation between the means of any two mixture components.

We begin with the following observation.

Lemma 20. *Consider some $X_j \in \mathbf{X}$ associated with $i \in [k]$. With probability at least $1 - \exp(-\Omega(\Delta^2))$, for every pair $\hat{\mu}, \hat{\mu}' \in L$, $\hat{\mu} \neq \hat{\mu}'$ letting $v_{\hat{\mu}\hat{\mu}'}$ be the unit vector in the direction $\hat{\mu} - \hat{\mu}'$,*

$$\langle v_{\hat{\mu}\hat{\mu}'}, X_j \rangle < \langle v_{\hat{\mu}\hat{\mu}'}, \mu_i \rangle + \Delta.$$

Proof. This is a standard application of sub-Gaussian concentration (on the one-dimensional distribution $\mathcal{N}(\langle v_{\hat{\mu}\hat{\mu}'}, \mu_i \rangle, \langle \Sigma_i, v_{\hat{\mu}\hat{\mu}'} v_{\hat{\mu}\hat{\mu}'}^\top \rangle)$), where we union bound across $O(k^2)$ pairs of elements in L . We simplify by using $\Delta = \Omega(\sqrt{k})$, so the exponential term dominates the k^2 union bound overhead. \square

Algorithm 15 ClusterUniformGMM($\mathbf{X}, L, \Delta, k, \delta$)

- 1: **Input:** $\mathbf{X} = \{X_j\}_{j \in [n]} \sim \sum_{i \in [k]} \frac{1}{k} \mathcal{D}_i$ where \mathcal{D}_i has mean μ_i and sub-Gaussian parameter ≤ 1 in all directions, and $n = \Theta(dk)$, L of size $O(k)$ containing (for all $i \in [k]$) $\hat{\mu}_i \in L$ with $\|\hat{\mu}_i - \mu_i\|_2 \leq \Delta$ for $\Delta = \Omega(\sqrt{k})$, $\delta \in (0, 1)$
 - 2: $\mathbf{G} \in \mathbb{R}^{d \times c} \leftarrow$ entrywise $\pm \frac{1}{\sqrt{c}}$ uniformly at random, for $c = \Theta(\log \frac{n}{\delta})$ (Johnson-Lindenstrauss matrix [Ach03])
 - 3: Let $m : [n] \rightarrow L$ map each X_j to the element $\hat{\mu} \in L$ minimizing $\|\mathbf{G}^\top(X_j - \hat{\mu})\|_2$
 - 4: Define an equivalence relation \sim on \mathbf{X} by $X_i \sim X_j$ iff $\|\mathbf{G}^\top(m(i) - m(j))\|_2 \leq 18\Delta$; if this is not an equivalence relation, then return any labeling
 - 5: **return** Labeling of \mathbf{X} associated with \sim
-

Next, we give our key structural lemma regarding the map m .

Lemma 21. *Following notation of Algorithm 15, with probability at least $1 - \delta - n \exp(-\Omega(\Delta^2))$, every X_j associated with $i \in [k]$ satisfies $\|m(j) - \mu_i\|_2 \leq 7\Delta$.*

Proof. With probability at least $1 - \delta$, all pairwise distances between $\mathbf{X} \cup L$ and itself are preserved by multiplication through \mathbf{G}^\top up to a 1 ± 0.1 factor [Ach03] (which we will call the “Johnson-Lindenstrauss guarantee” henceforth); condition on this event for the remainder of the proof. Suppose for contradiction that $\|m(j) - \mu_i\|_2 > 7\Delta$, and let $\hat{\mu}_i \in L$ denote any (fixed) hypothesis which is promised to satisfy $\|\hat{\mu}_i - \mu_i\|_2 \leq \Delta$.⁶ By the triangle inequality, $\|m(j) - \hat{\mu}_i\|_2 > 6\Delta$. Then letting v be the unit vector in the direction of $m(j) - \hat{\mu}_i$,

$$\begin{aligned} \langle v, \mu_i \rangle &\leq \langle v, \hat{\mu}_i \rangle + \Delta, \quad \langle v, m(j) \rangle > \langle v, \hat{\mu}_i \rangle + 6\Delta \\ \implies \langle v, \mu_i \rangle &< \langle v, m(j) \rangle - 5\Delta. \end{aligned}$$

Next, $\|\mathbf{G}^\top(X_j - m(j))\|_2 \leq \|\mathbf{G}^\top(X_j - \hat{\mu}_i)\|_2$ implies $\|X_j - m(j)\|_2 \leq 2\|X_j - \hat{\mu}_i\|_2$ by the Johnson-Lindenstrauss guarantee, or $\langle v, X_j \rangle \geq \left\langle v, \frac{2\hat{\mu}_i + m(j)}{3} \right\rangle$. Combining with the above displayed equation,

$$\langle v, X_j \rangle \geq \frac{2}{3} \langle v, \hat{\mu}_i \rangle + \frac{1}{3} \langle v, m(j) \rangle > \langle v, \mu_i \rangle + \Delta.$$

Applying Lemma 20 and union bounding over all $j \in [n]$ concludes the proof. \square

This implies that with high probability, Algorithm 15 (given a list L meeting its prerequisites) succeeds in correctly labelling all data points, assuming $\Omega(\Delta)$ separation between component means.

Lemma 22. *Suppose every pair $i, i' \in [k]$, $i \neq i'$ satisfies $\|\mu_i - \mu_{i'}\|_2 > 34\Delta$. Then with probability at least $1 - \delta - n \exp(-\Omega(\Delta^2))$, Algorithm 15 (assuming its preconditions) outputs a correct clustering of all points (up to label permutation).*

Proof. Assume the result of Lemma 21 and that multiplication through \mathbf{G}^\top preserves all pairwise distances between $\mathbf{X} \cup L$ and itself up to a 1 ± 0.1 factor throughout this proof.

We first prove that for any two $X_j, X_{j'}$ associated to the same $i \in [k]$, Line 4 of Algorithm 15 sets $X_j \sim X_{j'}$. To see this, Lemma 21 and the triangle inequality give $\|m(j) - m(j')\|_2 \leq 14\Delta$, so this will pass Line 4 by the Johnson-Lindenstrauss guarantee. Next, suppose X_j is associated with

⁶In the case multiple such hypotheses exist, any satisfactory (but fixed) $\{\hat{\mu}_i\}_{i \in [k]} \subseteq L$ will do.

$i \in [k]$ and $X_{j'}$ is associated with $i' \in [k]$ with $i \neq i'$, and suppose for contradiction $X_j \sim X_{j'}$. By the Johnson-Lindenstrauss guarantee, $\|m(j) - m(j')\|_2 \leq 20\Delta$, which yields by Lemma 21 and the triangle inequality that $\|\mu_i - \mu_{i'}\|_2 \leq 34\Delta$, contradicting the separation assumption. \square

We conclude with the following guarantee on ClusterUniformGMM.

Corollary 3. *Suppose every pair $i, i' \in [k]$, $i \neq i'$ satisfies $\|\mu_i - \mu_{i'}\|_2 = \Omega(\sqrt{k} \log k)$ for an appropriate constant. There is an algorithm drawing $n = \Theta(dk)$ samples from the mixture $\sum_{i \in [k]} \frac{1}{k} \mathcal{D}_i$ where \mathcal{D}_i has mean μ_i and sub-Gaussian parameter ≤ 1 in all directions, and returns a correct clustering of all points (up to label permutation) with probability at least*

$$1 - \delta - n \exp(-\Omega(k \log^2 k)) - k \exp(-\Omega(d)).$$

The algorithm runs in time, for any fixed $\epsilon_0 > 0$,

$$O\left(n^{1+\epsilon_0} d \log^4 n \log^4 \frac{n}{\delta} + k^2 \log^4 \frac{n}{\delta} + nk \log \frac{n}{\delta}\right).$$

Proof. We begin by stating the algorithm. We take $\frac{1}{10}$ of the dataset and run FastMultifilter⁷ on it to produce L satisfying the prerequisites of Algorithm 15, and then cluster the remaining $\frac{9}{10}$ of the dataset using L via ClusterUniformGMM; then, we take a disjoint $\frac{1}{10}$ and cluster the remaining $\frac{9}{10}$ using ClusterUniformGMM. We then match labels based on which clusters overlap on at least $\frac{1}{2}$ of their points between these two runs. The runtime follows from Theorem 5, and the runtime of ClusterUniformGMM, which is clearly $O(nk \log \frac{n}{\delta})$ since Line 3 dominates, as Line 4 can be greedily implemented using distance comparisons between only L once the map m has been formed.

Next, for correctness, Theorem 6 and Proposition B.1 of [CSV17] (which says Assumption 1 is met for both FastMultifilter runs with probability $\geq 1 - \exp(-\Omega(d))$) imply both runs of FastMultifilter correctly return lists satisfying the precondition of Algorithm 15; here we note that the dataset partition ensures independence of lists used and datasets clustered. Then, Lemma 22 implies both clusterings are completely correct on $\frac{9}{10}$ of the data. The conclusion follows from standard binomial concentration, which implies that the $\frac{8}{10}$ of the data which was held-out contains at least $\frac{1}{2}$ the points associated with each $i \in [k]$ in the overall dataset, with probability at least $1 - \exp(-\Omega(d))$. \square

We remark that for n which grows super-exponentially in k (such that the failure probability guarantee of Corollary 3 becomes vacuous), it is straightforward to obtain an appropriate high-probability guarantee for clustering all points by assuming that the minimum pairwise cluster separation scales as $\sqrt{\log n}$. A similar remark also applies to Corollary 6.

5.2 Robustly clustering (sub-)Gaussian mixture models

In this section, we generalize Corollary 3 to non-uniform corrupted mixture models. In particular, we consider an adversarially corrupted mixture model

$$\mathcal{M} = (1 - \epsilon) \sum_{i \in [k]} \alpha_i \mathcal{D}_i + \epsilon \mathcal{D}_{\text{adv}}, \quad (39)$$

⁷If $\alpha^{-1} = \log^{o(1)} d$, we instead run Algorithm 8 of [DKK⁺20b] to obtain the desired error guarantee, which fits within the runtime budget by Theorem 4 of [DKK⁺20b]. Similarly, if $\alpha^{-1} = \Omega(d)$, we instead run Algorithm 14 of [DKK⁺20b] which fits within the runtime budget by Proposition 9 of that paper.

where for all $i \in [k]$, \mathcal{D}_i has mean μ_i and sub-Gaussian parameter ≤ 1 in all directions. Moreover, for some fixed known α , we assume all $\alpha_i \geq \alpha$ and $\epsilon \leq \frac{\alpha}{4}$. By definition of α , note that we must have $\alpha = O(\frac{1}{k})$. In this section, we assume our list-decoding subroutine \mathcal{A} returns a list of size $O(\alpha^{-1})$, and guarantees estimation error Δ . We now state our algorithm.

Algorithm 16 ClusterRobustGMM($\mathbf{X}, L, \Delta, k, \delta, \alpha$)

- 1: **Input:** $\mathbf{X} = \{X_j\}_{j \in [n]} \sim (1 - \epsilon) \sum_{i \in [k]} \alpha_i \mathcal{D}_i + \epsilon \mathcal{D}_{\text{adv}}$ where \mathcal{D}_i has mean μ_i and sub-Gaussian parameter ≤ 1 in all directions, all $\alpha_i \geq \alpha$, $\epsilon \leq \frac{\alpha}{4}$, and $n = \Theta(\frac{d}{\alpha})$, L of size $O(\alpha^{-1})$ containing (for all $i \in [k]$) $\hat{\mu}_i \in L$ with $\|\hat{\mu}_i - \mu_i\|_2 \leq \Delta$ for $\Delta = \Omega(\sqrt{\alpha^{-1}})$, $\delta \in (0, 1)$
 - 2: $\mathbf{G} \in \mathbb{R}^{d \times c} \leftarrow$ entrywise $\pm \frac{1}{\sqrt{c}}$ uniformly at random, for $c = \Theta(\log \frac{n}{\delta})$ (Johnson-Lindenstrauss matrix [Ach03])
 - 3: Let $m : [n] \rightarrow L$ map each X_j to the element $\hat{\mu} \in L$ minimizing $\|\mathbf{G}^\top(X_j - \hat{\mu})\|_2$
 - 4: $\mathcal{S}_{\hat{\mu}} \leftarrow \{j \in [n] \mid m(j) = \hat{\mu}\}$ for all $\hat{\mu} \in L$, $\mathcal{B}_{\hat{\mu}} \leftarrow \bigcup_{\hat{\mu}' \in L \mid \|\mathbf{G}^\top(\hat{\mu} - \hat{\mu}')\|_2 \leq 16\Delta} \mathcal{S}_{\hat{\mu}'}$
 - 5: $L' \leftarrow \{\hat{\mu} \in L \mid |\mathcal{B}_{\hat{\mu}}| \geq 0.9\alpha n\}$
 - 6: Define an equivalence relation \sim on \mathbf{X}' by $X_i \sim X_j$ iff $\|\mathbf{G}^\top(m(i) - m(j))\|_2 \leq 55\Delta$, for $\mathbf{X}' := \{X_i \in \mathbf{X} \mid m(i) \in L'\}$; if this is not an equivalence relation, then return any labeling
 - 7: **return** Labeling of \mathbf{X}' associated with \sim , along with $\mathbf{X} \setminus \mathbf{X}'$ as “unlabeled”
-

We again refer to X_j as associated with some $i \in [k]$ if it is a draw from \mathcal{D}_i , and as “adversarial” if it is drawn from \mathcal{D}_{adv} . We begin with the following consequences of Lemma 21.

Corollary 4. *With probability at least $1 - \delta - n \exp(-\Omega(\Delta^2)) - k \exp(-\Omega(d))$, both of the following events hold. For all $i \in [k]$, every $\hat{\mu} \in L$ with $\|\hat{\mu} - \mu_i\|_2 \leq 7\Delta$ is in L' . Moreover, every $\hat{\mu} \in L'$ has $\|\hat{\mu} - \mu_i\|_2 \leq 25\Delta$ for some $i \in [k]$.*

Proof. By standard binomial concentration, for every $i \in [k]$, the set of $j \in [n]$ associated with i has size at least $0.9\alpha n$ with probability $1 - k \exp(-\Omega(d))$. Condition on this event, all pairwise distances between $\mathbf{X} \cup L$ and itself being preserved up to 1 ± 0.1 by multiplication through \mathbf{G}^\top , and the conclusion of Lemma 20 for the remainder of this proof.

We begin with the first claim: let $\|\hat{\mu} - \mu_i\|_2 \leq 7\Delta$. For every X_j associated with i , Lemma 21 shows $\|m(j) - \mu_i\|_2 \leq 7\Delta$, implying by the Johnson-Lindenstrauss guarantee and triangle inequality, $\|\mathbf{G}^\top(m(j) - \hat{\mu})\|_2 \leq 16\Delta$. Hence X_j counts towards $\mathcal{B}_{\hat{\mu}}$, which then captures all points associated with i , so $\hat{\mu} \in L'$. For the second claim, suppose $\|\hat{\mu} - \mu_i\|_2 > 25\Delta$; then, no $\hat{\mu}' \in L$ with $\|\hat{\mu}' - \mu_i\|_2 \leq 7\Delta$ will count towards $\mathcal{B}_{\hat{\mu}}$, since such $\hat{\mu}'$ has $\|\mathbf{G}^\top(\hat{\mu} - \hat{\mu}')\|_2 > 16\Delta$. By Lemma 21, the only points that can contribute to $\mathcal{B}_{\hat{\mu}}$ are then the ones drawn from \mathcal{D}_{adv} , which by standard binomial concentration will be less than $0.6\alpha n$ with probability $1 - \exp(-\Omega(d))$, and hence $\hat{\mu} \notin L'$. \square

The following conclusion then follows immediately in the vein of Lemma 22.

Corollary 5. *Suppose every pair $i, i' \in [k]$, $i \neq i'$ satisfies $\|\mu_i - \mu_{i'}\|_2 > 55\Delta$. Then with probability at least $1 - \delta - n \exp(-\Omega(\Delta^2)) - k \exp(-\Omega(d))$, Algorithm 16 (assuming its preconditions) outputs a correct clustering of all points associated with some component $i \in [k]$ (up to label permutation).*

Proof. The proof is identical to Lemma 22, where we note for each $\hat{\mu} \in L'$, there is a unique $i \in [k]$ promised by Corollary 4 where $\|\hat{\mu} - \mu_i\|_2 \leq 25\Delta$ (by our separation assumption). Thus, \sim defines an equivalence relation, since every $\hat{\mu} \in L'$ is mapped to the unique equivalence class associated with $\hat{\mu}_i$. This successfully recovers the true clusters, since every point X_j associated with some $i \in [k]$ will be in \mathbf{X}' by Corollary 4 and Lemma 21, and thus it will be classified correctly by \sim . \square

Finally, we give a complete guarantee on ClusterRobustGMM.

Corollary 6. *Suppose every pair $i, i' \in [k]$, $i \neq i'$ satisfies $\|\mu_i - \mu_{i'}\|_2 = \Omega(\sqrt{\alpha^{-1}} \log \alpha^{-1})$ for an appropriate constant. There is an algorithm drawing $n = \Theta(\frac{d}{\alpha})$ samples from the mixture $(1 - \epsilon) \sum_{i \in [k]} \alpha_i \mathcal{D}_i + \epsilon \mathcal{D}_{\text{adv}}$ where \mathcal{D}_i has mean μ_i and sub-Gaussian parameter ≤ 1 in all directions, and returns a correct clustering of all points drawn from some \mathcal{D}_i (up to label permutation) with probability at least*

$$1 - \delta - n \exp(-\Omega(\alpha^{-1} \log^2 \alpha^{-1})) - k \exp(-\Omega(d)).$$

The algorithm runs in time, for any fixed $\epsilon_0 > 0$,

$$O\left(n^{1+\epsilon_0} d \log^4 n \log^4 \frac{n}{\delta} + \frac{1}{\alpha^2} \log^4 \frac{n}{\delta} + \frac{n}{\alpha} \log \frac{n}{\delta}\right).$$

Proof. The proof is the same as Corollary 3, where we note that the mislabeled points from \mathcal{D}_{adv} can only affect the overlapping labels between the two runs by at most a $1.1\epsilon \leq \frac{1.1}{4}\alpha$ fraction, a minority compared to the number of correct labels (due to the true mixture), which in both runs will contribute at least $\frac{4}{5}\alpha_i$ of the points assigned by \sim to the true clustering. Hence, between runs the clusters assigned by \sim to the points drawn from \mathcal{D}_i will overlap on at least half their points, and we can use this agreement to correctly label all points not drawn from \mathcal{D}_{adv} (where throughout, we conditioned on all high-probability events in Corollary 3's proof holding). \square

5.3 Mixture models with bounded fourth moments

In this section, we consider non-uniform corrupted mixture models under a weaker distributional assumption based on bounded fourth moments. We will no longer be able to correctly cluster all (non-adversarial) points, but will instead guarantee that amongst non-adversarial points, at least a $1 - o(\alpha)$ fraction are correctly classified. Concretely, we consider the adversarially corrupted mixture model (39), where for all $i \in [k]$, and some constant $C = O(1)$,

$$\mathbb{E}_{X \sim \mathcal{D}_i} [\langle v, X - \mu_i \rangle^4] \leq C \text{ for all } v \in \mathbb{R}^d, \|v\|_2 = 1. \quad (40)$$

We remark that this fourth-moment bound also implies the covariance is bounded by $O(1)\mathbf{I}$ for all components, by Jensen's inequality. We again assume that all $\alpha_i \geq \alpha$, and $\epsilon \leq \frac{\alpha}{4}$. Our algorithm for this setting will be exactly the same as ClusterRobustGMM, but for convenience we restate it under the new distributional assumptions. We also assume $\Delta = \omega(\sqrt{\alpha^{-1}})$ for notational simplicity.

Our analysis of ClusterRobustBFMM follows from the following key observation: define $X_j \in \mathbf{X}$ as “adversarial” if it is drawn from \mathcal{D}_{adv} , and say it is “pseudo-adversarial” if it is drawn from some component $i \in [k]$, but satisfies $\|m(j) - \mu_i\|_2 > 7\Delta$. Then, by the bounded fourth moment assumption (40), there are few pseudo-adversarial points, made rigorous as follows.

Lemma 23. *With probability at least $1 - \delta - \exp(-\Omega(d))$, the number of pseudo-adversarial points in \mathbf{X} is $o(\alpha n)$.*

Proof. We first bound the probability that some $X_j \sim \mathcal{D}_i$ will be pseudo-adversarial. By the proof of Lemma 21, if X_j were pseudo-adversarial, it must be the case that in the direction v corresponding to $m(j) - \hat{\mu}_i$, $\langle v, X_j \rangle > \langle v, \mu_i \rangle + \Delta$. However, by (40) and Markov,

$$\Pr_{X \sim \mathcal{D}_i} [\langle v, X - \mu_i \rangle^4 \geq \Delta^4] \leq \frac{C}{\Delta^4} = o(\alpha^2).$$

Algorithm 17 ClusterRobustBFMM($\mathbf{X}, L, \Delta, k, \delta, \alpha$)

- 1: **Input:** $\mathbf{X} = \{X_j\}_{j \in [n]} \sim (1 - \epsilon) \sum_{i \in [k]} \alpha_i \mathcal{D}_i + \epsilon \mathcal{D}_{\text{adv}}$ where \mathcal{D}_i has mean μ_i and satisfies (40), all $\alpha_i \geq \alpha$, $\epsilon \leq \frac{\alpha}{4}$, and $n = \Theta(\frac{d}{\alpha})$, L of size $O(\alpha^{-1})$ containing (for all $i \in [k]$) $\hat{\mu}_i \in L$ with $\|\hat{\mu}_i - \mu_i\|_2 \leq \Delta$ for $\Delta = \omega(\sqrt{\alpha^{-1}})$, $\delta \in (0, 1)$
 - 2: $\mathbf{G} \in \mathbb{R}^{d \times c} \leftarrow$ entrywise $\pm \frac{1}{\sqrt{c}}$ uniformly at random, for $c = \Theta(\log \frac{n}{\delta})$ (Johnson-Lindenstrauss matrix [Ach03])
 - 3: Let $m : [n] \rightarrow L$ map each X_j to the element $\hat{\mu} \in L$ minimizing $\|\mathbf{G}^\top (X_j - \hat{\mu})\|_2$
 - 4: $\mathcal{S}_{\hat{\mu}} \leftarrow \{j \in [n] \mid m(j) = \hat{\mu}\}$ for all $\hat{\mu} \in L$, $\mathcal{B}_{\hat{\mu}} \leftarrow \bigcup_{\hat{\mu}' \in L} \|\mathbf{G}^\top (\hat{\mu} - \hat{\mu}')\|_2 \leq 15\Delta \mathcal{S}_{\hat{\mu}'}$
 - 5: $L' \leftarrow \{\hat{\mu} \in L \mid |\mathcal{B}_{\hat{\mu}}| \geq 0.9\alpha n\}$
 - 6: Define an equivalence relation \sim on \mathbf{X}' by $X_i \sim X_j$ iff $\|\mathbf{G}^\top (m(i) - m(j))\|_2 \leq 55\Delta$, for $\mathbf{X}' := \{X_i \in \mathbf{X} \mid m(i) \in L'\}$; if this is not an equivalence relation, then return any labeling
 - 7: **return** Labeling of \mathbf{X}' associated with \sim , along with $\mathbf{X} \setminus \mathbf{X}'$ as “unlabeled”
-

Union bounding over all possible directions v (one for each of the $O(\alpha^{-1})$ elements of L other than $\hat{\mu}_i$), this implies the chance that X drawn from *any* \mathcal{D}_i is pseudo-adversarial is $o(\alpha)$. Hence, the expected number of pseudo-adversarial points in our entire dataset is $o(\alpha n)$, and the conclusion follows by applying standard binomial concentration. \square

Now, Corollary 4 holds for ClusterRobustBFMM simply by lumping together the pseudo-adversarial points and the adversarial points in its proof (in particular, no hypothesis more than 25Δ away from a true mean can capture any points in the dataset other than adversarial and pseudo-adversarial ones). We now state analogs of Corollaries 5 and 6.

Corollary 7. *Suppose every pair $i, i' \in [k]$, $i \neq i'$ satisfies $\|\mu_i - \mu_{i'}\|_2 > 55\Delta$. Then with probability at least $1 - \delta - n \exp(-\Omega(\Delta^2)) - k \exp(-\Omega(d))$, Algorithm 17 (assuming its preconditions) outputs a correct clustering of a $1 - o(\alpha)$ proportion of points associated with some component $i \in [k]$ (up to label permutation).*

Proof. The proof is identical to Corollary 5, where we only guarantee correctly labeling points which are not pseudo-adversarial; \sim will correctly cluster all such points by separation and Corollary 4. \square

Corollary 8. *Suppose every pair $i, i' \in [k]$, $i \neq i'$ satisfies $\|\mu_i - \mu_{i'}\|_2 = \Omega(\sqrt{\alpha^{-1}} \log \alpha^{-1})$ for an appropriate constant. There is an algorithm drawing $n = \Theta(\frac{d}{\alpha})$ samples from the mixture $(1 - \epsilon) \sum_{i \in [k]} \alpha_i \mathcal{D}_i + \epsilon \mathcal{D}_{\text{adv}}$ where \mathcal{D}_i has mean μ_i and satisfies the bounded fourth moment condition (40), and returns a correct clustering of a $1 - o(\alpha)$ fraction of all points drawn from some \mathcal{D}_i (up to label permutation) with probability at least*

$$1 - \delta - k \exp(-\Omega(d)).$$

The algorithm runs in time, for any fixed $\epsilon_0 > 0$,

$$O\left(n^{1+\epsilon_0} d \log^4 n \log^4 \frac{n}{\delta} + \frac{1}{\alpha^2} \log^4 \frac{n}{\delta} + \frac{n}{\alpha} \log \frac{n}{\delta}\right).$$

Proof. The proof is identical to Corollary 6, using Corollary 7 instead of Corollary 5, and using that the pseudo-adversarial points will not substantially affect cluster overlap guarantees. \square

5.4 Bounded-covariance mixture models

In this section, we finally handle the case of non-uniform corrupted mixture models under only a bounded-covariance assumption; in particular, we study (39) where every component \mathcal{D}_i for $i \in [k]$ has covariance bounded by \mathbf{I} . We again assume that all $\alpha_i \geq \alpha$, and $\epsilon \leq \frac{\alpha}{4}$.

Algorithm 18 ClusterRobustBCMM($\mathbf{X}, L, \Delta, k, \delta, \alpha$)

- 1: **Input:** $\mathbf{X} = \{X_j\}_{j \in [n]} \sim (1 - \epsilon) \sum_{i \in [k]} \alpha_i \mathcal{D}_i + \epsilon \mathcal{D}_{\text{adv}}$ where \mathcal{D}_i has mean μ_i and has covariance bounded by \mathbf{I} , all $\alpha_i \geq \alpha$, $\epsilon \leq \frac{\alpha}{4}$, and $n = \Theta(\frac{d}{\alpha})$, L of size $O(\alpha^{-1})$ containing (for all $i \in [k]$) $\hat{\mu}_i \in L$ with $\|\hat{\mu}_i - \mu_i\|_2 \leq \Delta$ for $\Delta = \Omega(\sqrt{\alpha^{-1}} \log \alpha^{-1})$, $\delta \in (0, 1)$
 - 2: $\mathbf{G} \in \mathbb{R}^{d \times c} \leftarrow$ entrywise $\pm \frac{1}{\sqrt{c}}$ uniformly at random, for $c = \Theta(\log \frac{n}{\delta})$ (Johnson-Lindenstrauss matrix [Ach03])
 - 3: $\mathbf{P} \leftarrow \mathbf{L}^\top (\mathbf{L}\mathbf{L}^\top)^{-1} \mathbf{L}$ (i.e. projection matrix onto the subspace spanned by L) where $\mathbf{L} \in \mathbb{R}^{O(\alpha^{-1}) \times d}$ is the vertical concatenation of L
 - 4: $\tilde{X}_j \leftarrow \mathbf{G}^\top \mathbf{P} X_j$ for all $j \in [n]$, $\tilde{\mu}_i \leftarrow \mathbf{G}^\top \hat{\mu}_i$ for all $\hat{\mu}_i \in L$
 - 5: $\tilde{L} \leftarrow \left\{ \hat{\mu}_i \in L \mid |\mathcal{S}(\hat{\mu}_i)| \geq \frac{\alpha n}{2} \right\}$, where $\mathcal{S}(\hat{\mu}_i) := \left\{ X_j \in \mathbf{X} \mid \|\tilde{X}_j - \tilde{\mu}_i\|_2 \leq 2.5\Delta \right\}$
 - 6: Define an equivalence relation \sim on \tilde{L} by $\hat{\mu}_i \sim \hat{\mu}_j$ iff $\|\tilde{\mu}_i - \tilde{\mu}_j\|_2 \leq 20\Delta$; if this is not an equivalence relation, then return any labeling
 - 7: **return** Labeling of \mathbf{X} where \tilde{X}_j and $\tilde{X}_{j'}$ have the same label iff $\tilde{X}_j \in \mathcal{S}(\hat{\mu}_i)$ and $\tilde{X}_{j'} \in \mathcal{S}(\hat{\mu}_{i'})$ for $\hat{\mu}_i \sim \hat{\mu}_{i'}$
-

We briefly describe Algorithm 18. Line 3 forms the projection matrix onto the span of L . Line 5 “prunes” the set L to a set \tilde{L} where we only keep candidates with enough data points close by (in the subspace spanned by L). Line 6 then appropriately partitions the candidates, and Line 7 partitions the dataset based on the candidate partition. In the following proof we use $\mathbf{P}\hat{\mu} = \hat{\mu}$ for all $\hat{\mu} \in L$, and that all μ_i have a point in the subspace projected to by \mathbf{P} at most Δ away.

Lemma 24. *With probability at least $1 - \delta - k \exp(-\Omega(d))$, if $\Delta = \Omega(\sqrt{\alpha^{-1}} \log \alpha^{-1})$, and every pair $i, i' \in [k]$, $i \neq i'$ satisfies $\|\mu_i - \mu_{i'}\|_2 > 20\Delta$, Algorithm 18 returns a correct clustering of a $1 - o(1)$ fraction of all points drawn from some \mathcal{D}_i . The algorithm runs in time*

$$O\left(n \cdot \left(d + \frac{1}{\alpha}\right) \cdot \log \frac{n}{\delta}\right).$$

Proof. Throughout this proof, condition on the event that distances between $\mathbf{P}\mathbf{X} \cup L$ are preserved up to 1 ± 0.1 by multiplication through \mathbf{G}^\top and that there are at most $\frac{\alpha n}{3}$ adversarial points; we will union bound this conditioning with an event of probability $1 - k \exp(-\Omega(d))$.

We first claim that any $\hat{\mu} \in L$ satisfying $\|\hat{\mu} - \mu_i\|_2 \leq \Delta$ for some $i \in [k]$ is contained in \tilde{L} , with probability at least $1 - \exp(-\Omega(d))$: call this claim \dagger . To see \dagger , with probability at least $1 - \exp(-\Omega(d))$, a $\frac{9}{10}$ fraction of $X_j \in \mathbf{X}$ sampled from \mathcal{D}_i satisfy $\|\mathbf{P}(X_j - \mu_i)\|_2 \leq \Delta$. This follows since Chebyshev’s inequality, $\Delta = \omega(\sqrt{\alpha^{-1}})$, and \mathbf{P} projecting onto a $O(\alpha^{-1})$ -dimensional subspace imply the probability $\|\mathbf{P}(X_j - \mu_i)\|_2 \leq \Delta$ is $o(1)$. By triangle inequality and $\mathbf{P} \preceq \mathbf{I}$, for such X_j ,

$$\|\mathbf{P}(X_j - \hat{\mu})\|_2 \leq \|\mathbf{P}(X_j - \mu_i)\|_2 + \|\mathbf{P}(\hat{\mu} - \mu_i)\|_2 \leq \Delta + \|\hat{\mu} - \mu_i\|_2 \leq 2\Delta.$$

Since \mathbf{G}^\top preserves distances to a 1.1 factor, this implies $\|\tilde{X}_j - \tilde{\mu}\|_2 \leq 2.5\Delta$ as desired.

We next claim that any $\hat{\mu} \in L$ satisfying $\|\hat{\mu} - \mu_i\|_2 > 7\Delta$ for all $i \in [k]$ will not be contained in \tilde{L} , with probability at least $1 - \exp(-\Omega(d))$. To see this, we claim $\mathcal{S}(\hat{\mu})$ can only contain an $o(\alpha)$ fraction of the points drawn from \mathcal{D}_i ; summing over all \mathcal{D}_i , and adding in all $\leq \frac{\alpha n}{3}$ adversarial points, shows $|\mathcal{S}(\hat{\mu})| < \frac{\alpha n}{2}$. This latter claim follows by first observing that for $X_j \in \mathcal{S}(\hat{\mu})$,

$$\begin{aligned} \|\mathbf{P}(X_j - \mu_i)\|_2 &\geq \|\mu_i - \hat{\mu}\|_2 - \|\mu_i - \mathbf{P}\mu_i\|_2 - \|\mathbf{P}(X_j - \hat{\mu})\|_2 \\ &> 7\Delta - \Delta - 3\Delta = 3\Delta \geq \|\mathbf{P}(X_j - \hat{\mu})\|_2. \end{aligned}$$

The first inequality used that $\mathbf{P}\hat{\mu} = \hat{\mu}$, and the triangle inequality twice. In the second inequality, we used $\|\mu_i - \mathbf{P}\mu_i\|_2 \leq \Delta$ since the subspace projected to by \mathbf{P} contains a point at most Δ away from μ_i , and $\|\mathbf{P}(X_j - \hat{\mu})\|_2 \leq 3\Delta$ by the assumption on \mathbf{G} and $X_j \in \mathcal{S}(\hat{\mu})$. This implies that the projection of X_j is closer to $\mathbf{P}\hat{\mu} = \hat{\mu}$ than $\mathbf{P}\mu_i$. Letting \mathcal{D}'_i be the projection of \mathcal{D}_i by \mathbf{P} , $X \sim \mathcal{D}'_i$ is closer to $\hat{\mu}$ than $\mathbf{P}\mu_i$ with probability at most $\frac{1}{\Omega(\Delta^2)} = o(\alpha)$ by arguments in Lemma 21 and Chebyshev. Hence, the probability that $X_j \in \mathcal{S}(\hat{\mu})$ is $o(\alpha)$, as desired.

Now, under the success of all above conditioning events, by the minimum separation assumption of 20Δ , each surviving $\hat{\mu} \in \tilde{L}$ has a unique μ_i such that $\|\hat{\mu} - \mu_i\|_2 \leq 7\Delta$, and each designated $\hat{\mu}_i$ with $\|\hat{\mu}_i - \mu_i\|_2 \leq \Delta$ survives. Hence, the equivalence partition succeeds and captures all $\hat{\mu} \in \tilde{L}$ at distance at most 7Δ from a μ_i . Moreover, for every pair $\hat{\mu}, \hat{\mu}' \in \tilde{L}$ such that $\|\hat{\mu} - \mu_i\|_2 \leq 7\Delta$ and $\|\hat{\mu}' - \mu_{i'}\|_2 \leq 7\Delta$ for some $i \neq i'$,

$$X \in \mathcal{S}(\hat{\mu}) \cap \mathcal{S}(\hat{\mu}') \implies \|\hat{\mu} - \hat{\mu}'\|_2 \leq 6\Delta.$$

Since the minimum separation between μ_i and $\mu_{i'}$ is 20Δ , this is a contradiction by the triangle inequality. So, no sets $\mathcal{S}(\hat{\mu})$ and $\mathcal{S}(\hat{\mu}')$ intersect, where the hypotheses are close to different means. Thus the labeling in Line 7 is well-defined. It remains to show that if $X_j \sim \mathcal{D}_i$, we will have $X_j \in \mathcal{S}(\hat{\mu}_i)$ with probability $1 - o(1)$ (and hence a $1 - o(1)$ fraction of points is labeled correctly). This was in fact shown earlier, when we proved the claim \dagger .

Finally, it is clear that all operations can be performed in the desired runtime: $\mathbf{G}^\top \mathbf{P}$ can be computed explicitly in time $O(d^2\alpha^{-1} + d^2 \log \frac{n}{\delta}) = O(nd + d^2 \log \frac{n}{\delta})$ via naïve matrix multiplication, and all projection and distance computations (multiplying all points and hypotheses by $\mathbf{G}^\top \mathbf{P}$, computing all $\mathcal{S}(\hat{\mu})$, and checking all pairwise distances in Line 6) take time $O(n \cdot (d + \frac{1}{\alpha}) \cdot \log \frac{n}{\delta})$. \square

Corollary 9. *Suppose every pair $i, i' \in [k]$, $i \neq i'$ satisfies $\|\mu_i - \mu_{i'}\|_2 = \Omega(\sqrt{\alpha^{-1}} \log \alpha^{-1})$ for an appropriate constant. There is an algorithm drawing $n = \Theta(\frac{d}{\alpha})$ samples from the mixture $(1 - \epsilon) \sum_{i \in [k]} \alpha_i \mathcal{D}_i + \epsilon \mathcal{D}_{\text{adv}}$ where \mathcal{D}_i has mean μ_i and covariance bounded by \mathbf{I} , and returns a correct clustering of a $1 - \epsilon_1$ fraction of all points drawn from some \mathcal{D}_i (up to label permutation) for any fixed $\epsilon_1 > 0$, with probability at least*

$$1 - \delta - k \exp(-\Omega(d)).$$

The algorithm runs in time, for any fixed $\epsilon_0 > 0$,

$$O\left(n^{1+\epsilon_0} d \log^4 n \log^4 \frac{n}{\delta} + \frac{1}{\alpha^2} \log^4 \frac{n}{\delta} + \frac{n}{\alpha} \log \frac{n}{\delta}\right).$$

Proof. Since the goal is to correctly label a $1 - \epsilon_1$ fraction of all points, we can use a $\frac{\epsilon_1}{2}$ fraction of our dataset as holdout for learning an independent list L . We then use this list to cluster a $1 - \frac{\epsilon_1}{2}$ fraction of the remaining points via Lemma 24. \square

Acknowledgments

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

- [ABDH⁺18] Hassan Ashtiani, Shai Ben-David, Nicholas JA Harvey, Christopher Liaw, Abbas Mehrabian, and Yaniv Plan. 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*, pages 3416–3425, 2018. 1.3
- [ABG⁺14] Joseph Anderson, Mikhail Belkin, Navin Goyal, Luis Rademacher, and James Voss. The more, the merrier: the blessing of dimensionality for learning large gaussian mixtures. In *Conference on Learning Theory*, pages 1135–1164. PMLR, 2014. 1.3
- [Ach03] Dimitris Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *J. Comput. Syst. Sci.*, 66(4):671–687, 2003. 3.1, 4.1, 3, 4.7, 2, 5.1, 2, 2, 2
- [ADLS17] Jayadev Acharya, Ilias Diakonikolas, Jerry Li, and Ludwig Schmidt. Sample-optimal density estimation in nearly-linear time. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1278–1289. SIAM, 2017. 1.3
- [AJOS14] Jayadev Acharya, Ashkan Jafarpour, Alon Orlitsky, and Ananda Theertha Suresh. Near-optimal-sample estimators for spherical gaussian mixtures. *arXiv preprint arXiv:1402.4746*, 2014. 1.3
- [AK05] Sanjeev Arora and Ravi Kannan. Learning mixtures of separated nonspherical gaussians. *The Annals of Applied Probability*, 15(1A):69–92, 2005. 1, 1.3
- [AK07] Sanjeev Arora and Satyen Kale. 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*, pages 227–236, 2007. 1.2.1
- [AM05] Dimitris Achlioptas and Frank McSherry. On spectral learning of mixtures of distributions. In *International Conference on Computational Learning Theory*, pages 458–469. Springer, 2005. 1, 1, 1.2.1, 1.3
- [Ans60] Frank J Anscombe. Rejection of outliers. *Technometrics*, 2(2):123–146, 1960. 1, 1.3
- [AS12] Pranjali Awasthi and Or Sheffet. Improved spectral-norm bounds for clustering. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 37–49. Springer, 2012. 1, 1, 1.3
- [Awa21] Pranjali Awasthi. Personal communication, September 2021. 1
- [BBV08] Maria-Florina Balcan, Avrim Blum, and Santosh Vempala. A discriminative framework for clustering via similarity functions. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 671–680, 2008. 1, 1.3
- [BCM⁺14] Aditya Bhaskara, Moses Charikar, Ankur Moitra, and Aravindan Vijayaraghavan. Smoothed analysis of tensor decompositions. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 594–603, 2014. 1.3
- [BDH⁺20] Ainesh Bakshi, Ilias Diakonikolas, Samuel B. Hopkins, Daniel Kane, Sushrut Karmalkar, and Pravesh K. Kothari. Outlier-robust clustering of gaussians and other non-spherical mixtures. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020*, pages 149–159. IEEE, 2020. 1.3

-
- [BDJ⁺20] Ainesh Bakshi, Ilias Diakonikolas, He Jia, Daniel M Kane, Pravesh K Kothari, and Santosh S Vempala. Robustly learning mixtures of k arbitrary gaussians. *arXiv preprint arXiv:2012.02119*, 2020. 1.3
 - [BDLS17] Sivaraman Balakrishnan, Simon S Du, Jerry Li, and Aarti Singh. Computationally efficient robust sparse estimation in high dimensions. In *Conference on Learning Theory*, pages 169–212, 2017. 1.3
 - [BK20a] Ainesh Bakshi and Pravesh Kothari. List-decodable subspace recovery via sum-of-squares. *arXiv preprint arXiv:2002.05139*, 2020. 1.3
 - [BK20b] Ainesh Bakshi and Pravesh Kothari. Outlier-robust clustering of non-spherical mixtures. *arXiv preprint arXiv:2005.02970*, 2020. 1.3
 - [BNJT10] Marco Barreno, Blaine Nelson, Anthony D. Joseph, and J. D. Tygar. The security of machine learning. *Mach. Learn.*, 81(2):121–148, 2010. 1
 - [BNL12] Battista Biggio, Blaine Nelson, and Pavel Laskov. 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*, 2012. 1
 - [BS15] Mikhail Belkin and Kaushik Sinha. Polynomial learning of distribution families. *SIAM Journal on Computing*, 44(4):889–911, 2015. 1.3
 - [BV08] S. Charles Brubaker and Santosh S. Vempala. 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*, pages 551–560, 2008. 1
 - [CDG19] Yu Cheng, Ilias Diakonikolas, and Rong Ge. High-dimensional robust mean estimation in nearly-linear time. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2755–2771. SIAM, 2019. 1.3, 5
 - [CDGW19] Yu Cheng, Ilias Diakonikolas, Rong Ge, and David P Woodruff. Faster algorithms for high-dimensional robust covariance estimation. In *Conference on Learning Theory*, pages 727–757, 2019. 1.3, 5
 - [CDKS18] Yu Cheng, Ilias Diakonikolas, Daniel Kane, and Alistair Stewart. Robust learning of fixed-structure bayesian networks. In *Advances in Neural Information Processing Systems*, pages 10283–10295, 2018. 1.3
 - [CDSS14] Siu-On Chan, Ilias Diakonikolas, Rocco A Servedio, and Xiaorui Sun. Efficient density estimation via piecewise polynomial approximation. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 604–613, 2014. 1.3
 - [CMY20] Yeshwanth Cherapanamjeri, Sidhanth Mohanty, and Morris Yau. List decodable mean estimation in nearly linear time. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020*, 2020. 1, 3, 1.3, 2.2, 5
 - [CSV17] Moses Charikar, Jacob Steinhardt, and Gregory Valiant. 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*, pages 47–60, 2017. 1, 1, 1, 1.3, 2.1, 5.1
 - [Das99] Sanjoy Dasgupta. Learning mixtures of gaussians. In *40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039)*, pages 634–644. IEEE, 1999. 1, 1.3
 - [DDS⁺09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 1

-
- [DHKK20] Ilias Diakonikolas, Samuel B Hopkins, Daniel Kane, and Sushrut Karmalkar. Robustly learning any clusterable mixture of gaussians. *arXiv preprint arXiv:2005.06417*, 2020. [1.3](#)
- [DHL19] Yihe Dong, Samuel B. Hopkins, and Jerry Li. 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*, pages 6065–6075, 2019. [1.2.1](#), [1.2.1](#), [1.2.2](#), [1.3](#), [2](#), [2.3](#), [5](#), [2.3](#)
- [DISZ18] Constantinos Daskalakis, Andrew Ilyas, Vasilis Syrgkanis, and Haoyang Zeng. Training gans with optimism. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018. [1.2.1](#)
- [DK14] Constantinos Daskalakis and Gautam Kamath. Faster and sample near-optimal algorithms for proper learning mixtures of gaussians. In *Conference on Learning Theory*, pages 1183–1213. PMLR, 2014. [1.3](#)
- [DK19] Ilias Diakonikolas and Daniel M Kane. Recent advances in algorithmic high-dimensional robust statistics. *arXiv preprint arXiv:1911.05911*, 2019. [1](#), [1.2.1](#), [1.3](#)
- [DK20] Ilias Diakonikolas and Daniel M. Kane. 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*, pages 184–195. IEEE, 2020. [1.3](#)
- [DKK⁺17] Ilias Diakonikolas, Gautam Kamath, Daniel M. Kane, Jerry Li, Ankur Moitra, and Alistair Stewart. 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*, pages 999–1008, 2017. [1.2.1](#), [1.3](#), [3](#)
- [DKK⁺19a] Ilias Diakonikolas, Gautam Kamath, Daniel Kane, Jerry Li, Ankur Moitra, and Alistair Stewart. Robust estimators in high-dimensions without the computational intractability. *SIAM Journal on Computing*, 48(2):742–864, 2019. [1](#), [1.2.1](#), [1.3](#)
- [DKK⁺19b] Ilias Diakonikolas, Gautam Kamath, Daniel Kane, Jerry Li, Jacob Steinhardt, and Alistair Stewart. Sever: A robust meta-algorithm for stochastic optimization. In *International Conference on Machine Learning*, pages 1596–1606, 2019. [1](#), [1.3](#)
- [DKK⁺19c] Ilias Diakonikolas, Daniel Kane, Sushrut Karmalkar, Eric Price, and Alistair Stewart. Outlier-robust high-dimensional sparse estimation via iterative filtering. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d’Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019*, pages 10688–10699, 2019. [1.3](#)
- [DKK20a] Ilias Diakonikolas, Daniel Kane, and Daniel Kongsgaard. 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*, 2020. [\(document\)](#), [1](#), [1.2.1](#), [1.2.1](#), [4](#), [1.3](#), [2](#), [2.3](#), [4.8](#)
- [DKK⁺20b] Ilias Diakonikolas, Daniel M. Kane, Daniel Kongsgaard, Jerry Li, and Kevin Tian. List-decodable mean estimation in nearly-pca time. *CoRR*, abs/2011.09973, 2020. [1](#), [1.2.1](#), [1.2.1](#), [3](#), [1.2.1](#), [1.3](#), [2.1](#), [2.1](#), [2.2](#), [2.2](#), [2.2](#), [2](#), [2.3](#), [5](#), [3](#), [3.3](#), [7](#), [18](#), [4.8](#), [7](#)

-
- [DKS18] Ilias Diakonikolas, Daniel M. Kane, and Alistair Stewart. 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*, pages 1047–1060, 2018. (document), 2, 1.1, 1.2.1, 1.2.2, 1.3, 1.3, 2, 2.1, 2.2
- [DKS19] Ilias Diakonikolas, Weihao Kong, and Alistair Stewart. Efficient algorithms and lower bounds for robust linear regression. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019*, pages 2745–2754. SIAM, 2019. 1.3
- [DL12] Luc Devroye and Gábor Lugosi. *Combinatorial methods in density estimation*. Springer Science & Business Media, 2012. 1.3
- [DS07] Sanjoy Dasgupta and Leonard Schulman. A probabilistic analysis of em for mixtures of separated, spherical gaussians. *Journal of Machine Learning Research*, 8(Feb):203–226, 2007. 1.3
- [FSO06] Jon Feldman, Rocco A Servedio, and Ryan O’Donnell. Pac learning axis-aligned mixtures of gaussians with no separation assumption. In *International Conference on Computational Learning Theory*, pages 20–34. Springer, 2006. 1.3
- [GHK15] Rong Ge, Qingqing Huang, and Sham M Kakade. Learning mixtures of gaussians in high dimensions. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 761–770, 2015. 1.3
- [HK13] Daniel Hsu and Sham M Kakade. Learning mixtures of spherical gaussians: moment methods and spectral decompositions. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 11–20, 2013. 1.3
- [HL18] Samuel B Hopkins and Jerry Li. Mixture models, robustness, and sum of squares proofs. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1021–1034, 2018. 2, 1.3
- [HP15] Moritz Hardt and Eric Price. Tight bounds for learning a mixture of two gaussians. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 753–760, 2015. 1.3
- [HRRS86] Frank R. Hampel, Elvezio M. Ronchetti, Peter J. Rousseeuw, and Werner A. Stahel. *Robust Statistics: the Approach based on Influence Functions*. Wiley Series in Probability and Mathematical Statistics, 1986. 1, 1.3
- [Hub64] Peter J Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1):73–101, 1964. 1, 1.3
- [Hub04] Peter J Huber. *Robust statistics*, volume 523. John Wiley & Sons, 2004. 1, 1.3
- [JLT20] Arun Jambulapati, Jerry Li, and Kevin Tian. 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*, 2020. 1.2.2, 1.3, 4
- [Kan21] Daniel M Kane. Robust learning of mixtures of gaussians. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1246–1258. SIAM, 2021. 1.3
- [KK10] Amit Kumar and Ravindran Kannan. Clustering with spectral norm and the k-means algorithm. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 299–308. IEEE, 2010. 1.3

-
- [KKK19] Sushrut Karmalkar, Adam Klivans, and Pravesh Kothari. List-decodable linear regression. In *Advances in Neural Information Processing Systems*, pages 7425–7434, 2019. 1.3
 - [KKM18] Adam Klivans, Pravesh K Kothari, and Raghu Meka. Efficient algorithms for outlier-robust regression. In *Conference On Learning Theory*, pages 1420–1430, 2018. 1.3
 - [KSS18] Pravesh K Kothari, Jacob Steinhardt, and David Steurer. Robust moment estimation and improved clustering via sum of squares. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1035–1046, 2018. 2, 1.3, 1.3
 - [KSV08] Ravindran Kannan, Hadi Salmasian, and Santosh S. Vempala. The spectral method for general mixture models. *SIAM J. Comput.*, 38(3):1141–1156, 2008. 1
 - [LAT⁺08] 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. Worldwide human relationships inferred from genome-wide patterns of variation. 319:1100–1104, 2008. 1
 - [Li18] Jerry Zheng Li. *Principled approaches to robust machine learning and beyond*. PhD thesis, Massachusetts Institute of Technology, 2018. 1.2.1, 1.3
 - [LM20] Allen Liu and Ankur Moitra. Settling the robust learnability of mixtures of gaussians. *arXiv preprint arXiv:2011.03622*, 2020. 1.3
 - [LRV16] Kevin A Lai, Anup B Rao, and Santosh Vempala. Agnostic estimation of mean and covariance. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 665–674. IEEE, 2016. 1, 1.3
 - [LS17] Jerry Li and Ludwig Schmidt. Robust and proper learning for mixtures of gaussians via systems of polynomial inequalities. In *Conference on Learning Theory*, pages 1302–1382. PMLR, 2017. 1.3
 - [LY20] Jerry Li and Guanghao Ye. 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*, 2020. 1.3, 2.3, 5
 - [Mur12] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012. 1
 - [MV10] Ankur Moitra and Gregory Valiant. Settling the polynomial learnability of mixtures of gaussians. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 93–102. IEEE, 2010. 1.3
 - [MV18] Michela Meister and Gregory Valiant. A data prism: Semi-verified learning in the small-alpha regime. In *Conference On Learning Theory*, pages 1530–1546. PMLR, 2018. 1
 - [MVW17] Dustin G Mixon, Soledad Villar, and Rachel Ward. Clustering subgaussian mixtures by semidefinite programming. *Information and Inference: A Journal of the IMA*, 6(4):389–415, 2017. 1.3
 - [Pat11] Seth Patinkin. Method, apparatus, and system for clustering and classification, August 30 2011. US Patent 8,010,466. 1
 - [Pea94] Karl Pearson. Contributions to the mathematical theory of evolution. *Philosophical Transactions of the Royal Society of London*, 185:71–110, 1894. 1
 - [PLJD10] Peristera Paschou, Jamey Lewis, Asif Javed, and Petros Drineas. Ancestry informative markers for fine-scale individual assignment to worldwide populations. 47(12):835–847, 2010. 1

-
- [PSBR18] Adarsh Prasad, Arun Sai Suggala, Sivaraman Balakrishnan, and Pradeep Ravikumar. Robust estimation via robust gradient estimation. *arXiv preprint arXiv:1802.06485*, 2018. 1.3
- [RH17] Philippe Rigollet and Jan-Christian Hütter. *High-Dimensional Statistics*. 2017. 2.1
- [RPW⁺02] Noah A. Rosenberg, Jonathan K. Pritchard, James L. Weber, Howard M. Cann, Kenneth K. Kidd, Lev A. Zhivotovsky, and Marcus W. Feldman. Genetic structure of human populations. *Science*, 298:2381–2385, 2002. 1
- [RV17a] Oded Regev and Aravindan Vijayaraghavan. 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*, pages 85–96, 2017. 1
- [RV17b] Oded Regev and Aravindan Vijayaraghavan. On learning mixtures of well-separated gaussians. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 85–96. IEEE, 2017. 1.3
- [RY20] Prasad Raghavendra and Morris Yau. List decodable learning via sum of squares. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 161–180. SIAM, 2020. 1.3
- [SKL17] Jacob Steinhardt, Pang Wei Koh, and Percy Liang. 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*, pages 3517–3529, 2017. 1
- [Ste18] Jacob Steinhardt. *Robust Learning: Information Theory and Algorithms*. PhD thesis, Stanford University, 2018. 1.2.1, 1.3
- [STM20] Shibani Santurkar, Dimitris Tsipras, and Aleksander Madry. Breeds: Benchmarks for subpopulation shift. *arXiv preprint arXiv:2008.04859*, 2020. 1
- [SVC16] Jacob Steinhardt, Gregory Valiant, and Moses Charikar. Avoiding imposters and delinquents: Adversarial crowdsourcing and peer prediction. In *Advances in Neural Information Processing Systems*, pages 4439–4447, 2016. 1
- [TLM18] Brandon Tran, Jerry Li, and Aleksander Madry. Spectral signatures in backdoor attacks. In *Advances in Neural Information Processing Systems*, pages 8000–8010, 2018. 1.3
- [Tuk60] John W Tukey. A survey of sampling from contaminated distributions. *Contributions to probability and statistics*, pages 448–485, 1960. 1, 1.3
- [Tuk75] John W. Tukey. Mathematics and the picturing of data. In *Proceedings of the International Congress of Mathematicians, Vancouver, 1975*, volume 2, pages 523–531, 1975. 1, 1.3
- [VW04] Santosh Vempala and Grant Wang. A spectral algorithm for learning mixture models. *Journal of Computer and System Sciences*, 68(4):841–860, 2004. 1, 1.2.1, 1.3
- [WK06] Manfred K. Warmuth and Dima Kuzmin. 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*, pages 1481–1488, 2006. 1.2.1
- [WK18] Sławomir T Wierzchoń and Mieczysław A Kłopotek. *Modern algorithms of cluster analysis*. Springer, 2018. 1