Clustering Mixtures with Almost Optimal Separation in Polynomial Time

Allen Liu
cliu568@mit.edu
Massachusetts Institute of Technology
Cambridge, Massachusetts, USA

Jerry Li
jerrl@microsoft.com
Microsoft Research
Redmond, Washington, USA

ABSTRACT

We consider the problem of clustering mixtures of mean-separated Gaussians in high dimensions. We are given samples from a mixture of k identity covariance Gaussians, so that the minimum pairwise distance between any two pairs of means is at least Δ, for some parameter $\Delta > 0$, and the goal is to recover the ground truth clustering of these samples. It is folklore that separation $\Delta = \Theta(\sqrt{\log k})$ is both necessary and sufficient to recover a good clustering (say with constant or $1/\mathrm{poly}(k)$ error), at least information theoretically. However, the estimators which achieve this guarantee are inefficient. We give the first algorithm which runs in polynomial time, and which almost matches this guarantee. More precisely, we give an algorithm which takes polynomially many samples and time, and which can successfully recover a good clustering, so long as the separation is $\Delta = \Omega(\log^{1/2+c} k)$, for any $c > 0$. Previously, polynomial time algorithms were only known for this problem when the separation was polynomial in k, and all algorithms which could tolerate poly $\log k$ separation required quasipolynomial time. We also extend our result to mixtures of translations of a distribution which satisfies the Poincaré inequality, under additional mild assumptions. Our main technical tool, which we believe is of independent interest, is a novel way to implicitly represent and estimate high degree moments of a distribution, which allows us to extract important information about high-degree moments without ever writing down the full moment tensors explicitly.

CCS CONCEPTS

• **Theory of computation → Unsupervised learning and clustering**.

KEYWORDS

mixtures of gaussians, clustering, mixture models, Poincaré distribution, method of moments

ACM Reference Format:
Allen Liu and Jerry Li. 2022. Clustering Mixtures with Almost Optimal Separation in Polynomial Time. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing (STOC '22), June 20–24, 2022, Rome, Italy.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3519935.3520012

1 INTRODUCTION

Gaussian mixture models are some of the most popular generative models in both theory and practice. A k-Gaussian mixture model $\mathcal{M}$ (henceforth k-GMM) is a distribution specified by k non-negative mixing weights $w_1, \ldots, w_k$ which sum to 1, and k component Gaussians $N(\mu_1, \Sigma_1), \ldots, N(\mu_k, \Sigma_k)$, and is given by the probability density function

$$\mathcal{M} = \sum_{i=1}^{k} w_i N(\mu_i, \Sigma_i) \ .$$

In other words, to draw a sample from $\mathcal{M}$, we select the i-th component with probability w_i, and then draw an independent sample from $N(\mu_i, \Sigma_i)$. An important special case of this model is the *isotropic* (also sometimes referred to as *spherical*) case, where $\Sigma_i = I$ for all $i = 1, \ldots, k$.

One of the most important and well-studied problems for GMMs and isotropic GMMs in particular is *clustering*. Given n independent samples $X_1, \ldots, X_n$ drawn from an unknown k-GMM, the goal of clustering is to recover a partition of the data points into k parts $S_1, \ldots, S_k$ so that (a) almost all the points in every cluster are drawn from the same component Gaussian, and (b) for every component, there is some cluster which contains almost all of the samples drawn from it. Of course, not every isotropic GMM is clusterable: for instance, if two components are identical, then it is information-theoretically impossible to detect which component a sample came from, and so recovering a good clustering is also impossible. In the isotropic setting, a necessary and sufficient condition for the GMM to be clusterable is some amount of mean separation, namely, that $\|\mu_i - \mu_j\|_2 \geq \Delta$ for all $i \neq j$, for some parameter Δ.

The question then becomes: how small can we take Δ so that we can still cluster? For simplicity, for the remainder of this section, let us assume that $w_i \geq 1/\mathrm{poly}(k)$ for all $i = 1, \ldots, k$, that is, all the components have a nontrivial amount of weight. Information-theoretically, it is well-known (see e.g. [40]) that $\Delta = \Theta(\sqrt{\log k})$ separation is necessary and sufficient to achieve a clustering which correctly clusters more than an $1 - \epsilon$ fraction of the points with high probability, for all $\epsilon > 0$. Unfortunately, all known efficient algorithms for clustering at this separation rely on brute force methods, and run in exponential time.

If we restrict our attention to efficient estimators, the state of affairs is a bit more complicated. In fact, for over 20 years, the smallest separation that polynomial time algorithms could tolerate was $\Delta = \Omega(k^{1/4})$ [44]. It was not until recently that three concurrent papers [21, 26, 31] gave algorithms which could handle separation $\Delta = \Omega(k^{\gamma})$ with runtime and sample complexity $O(d^{\mathrm{poly}(1/\gamma)})$. In particular, whenever the separation is some fixed polynomial in k, these algorithms run in polynomial time. Unfortunately, if we wish to achieve the optimal separation of $\Omega(\sqrt{\log k})$—or indeed, any polylogarithmic amount of separation—these algorithms would require quasipolynomial time and sample complexity.

The barrier at quasipolynomial time. These algorithms all get stuck at quasipolynomial time when $\Delta = \Theta(\operatorname{poly}\log k)$ for the same reason. Fundamentally, all current algorithmic approaches to this problem rely on the following geometric identifiability fact:

> *Given enough samples from an isotropic k-GMM with separation $\Delta = \Omega(k^\gamma)$, then any sufficiently large subset of samples whose empirical $t = \Theta(1/\gamma)$ moments approximately match those of a Gaussian along all projections must essentially recover one of the true clusters.*

Algorithmically, this amounts to finding a subset of points whose empirical t-th moment tensor is close to that of a Gaussian in the appropriate norm. Since this results in a hard optimization problem whenever $t > 2$, these algorithms often solve some suitable relaxation of this using something like the Sum of Squares hierarchy. But, as the separation decreases, the algorithms must match more and more moments. In particular, to achieve $\Delta = \Theta(\operatorname{poly}\log k)$, one must set $t = \operatorname{poly}\log k$, that is, one must match polylogarithmically many moments. However, even writing down the degree $t = \operatorname{poly}\log k$ moment tensor requires quasipolynomial time, and guaranteeing that the empirical moment tensor concentrates requires quasipolynomially many samples. As a result, the aforementioned algorithms all require quasipolynomial time and sample complexity, as they need to not only write down the moment tensor, but perform some fairly complex optimization tasks on top of it.

On the flip side, there is no concrete reason for pessimism either. While there are lower bounds against large classes of efficient algorithms for clustering mixtures of arbitrary Gaussians, see e.g. [10, 20], none of these apply when the components are isotropic. In particular, this leaves open the appealing possibility that one could even cluster down to separation $\Delta = \Theta(\sqrt{\log k})$ in polynomial time. Stated another way, the question becomes:

> *Can we cluster any clusterable mixture of isotropic Gaussians in polynomial time?*

1.1 Our Results

In this paper, we (almost) resolve this question in the affirmative. Namely, for all constants $c > 0$, we give an algorithm which takes polynomially many samples and time, and which can cluster with high probability, so long as the separation satisfies $\Delta = \Omega(\log^{1/2+c} k)$. In other words, our algorithm can almost match the information theoretically optimal separation, up to sub-polylogarithmic factors. Our main theorem is:

Theorem 1.1 (informal, see Theorem 2.5, Corollary 2.6)**.** *Let $c > 0$ be fixed but arbitrary. Let $\mathcal{M}$ be a mixture of k isotropic Gaussians with minimum mixing weight lower bounded by $1/\operatorname{poly}(k)$, and minimum mean separation at least $\Delta = \Omega(\log^{1/2+c} k)$. Then, there is an algorithm which, given samples $X_1, \ldots, X_n \sim \mathcal{M}$ for $n = \operatorname{poly}(k, d)$, outputs a clustering which is correct for all of the points, with high probability. Moreover, this algorithm runs in time $\operatorname{poly}(d, k)$.*

We briefly remark that we can handle arbitrary mixing weights as well, but for simplicity we only state the theorem here in the more natural regime where all the mixing weights are not too small. We also remark that a simple corollary of this is that we can also estimate the parameters of $\mathcal{M}$ to good accuracy in polynomial time. In fact, by using our algorithm as a warm start for the method proposed in [40], we can achieve arbitrarily good accuracy for parameter estimation, in polynomial time. It is known that $\Delta = \Omega(\sqrt{\log k})$ is also necessary to achieve nontrivial parameter estimation with polynomially many samples [40], so our results for parameter estimation are also almost-optimal, in terms of the separation that they handle. We note that our formal theorems are actually stated for parameter estimation, rather than clustering, however, in light of [40], these two problems are equivalent in the regime we consider.

Our main technique (as we will discuss in more detail later) extends to beyond Gaussians, and in fact also allows us to cluster any mixture of translations of a distribution $\mathcal{D}$, so long as this distribution satisfies the *Poincaré inequality*, under a mild technical condition. Recall that a distribution $\mathcal{D}$ over $\mathbb{R}^d$ is said to be σ-Poincaré if for all differentiable functions $f : \mathbb{R}^d \to \mathbb{R}$, we have

$$\operatorname*{Var}_{X\sim\mathcal{D}}[f(X)] \leq \sigma^2 \cdot \mathop{\mathbb{E}}_{X\sim\mathcal{D}}\left[\|\nabla f(X)\|_2^2\right] .$$

This condition is widely studied in probability theory, and is satisfied by many natural distribution classes. For instance, isotropic Gaussians are 1-Poincaré, and any isotropic logconcave distribution is ψ-Poincaré, where ψ is the value of the KLS constant (see e.g. [43] for an overview of the KLS conjecture).

In fact, the family of Poincaré distributions is the most general family of distributions for which the previously mentioned Sum of Squares-based methods for clustering are known to work. For any well-separated mixture of 1-Poincaré distributions, [31] shows that one can recover the same guarantees as mentioned above: if the minimum mean separation is $\Delta = \Omega(k^\gamma)$, then their algorithm successfully clusters the points in time $O(d^{\operatorname{poly}(1/\gamma)})$. As before, when the separation is polylogarithmic, the runtime and sample complexity of their method is once again quasipolynomial.

We show that one can improve the runtime and sample complexity to polynomial time, under two additional assumptions: first, the mixture must consist of translated versions of the same Poincaré distribution which we can get samples from, and second, the maximum and minimum separations between any two components must be polynomially related. More concretely, we show:

Theorem 1.2 (informal, see Theorem 2.3, Corollary 2.4)**.** *Let $c > 0$ be fixed but arbitrary. Let $\mathcal{D}$ be a fixed distribution with mean zero over $\mathbb{R}^d$ which is 1-Poincaré. Let $\mathcal{M}$ be a mixture of k distributions where each component is of the form $\mu_i + \mathcal{D}$. Assume that the minimum mixing weight in this distribution is at least $1/\operatorname{poly}(k)$, and moreover, assume that*

$$\min_{i\neq j} \left\|\mu_i - \mu_j\right\|_2 \geq \Omega(\log^{1+c} k)$$

$$\max_{i\neq j} \left\|\mu_i - \mu_j\right\|_2 \leq \operatorname{poly}\left(\min_{i\neq j} \left\|\mu_i - \mu_j\right\|_2\right) .$$

Then, there is an algorithm which, given samples $X_1, \ldots, X_n \sim \mathcal{M}$ and samples $z_1, \ldots, z_n \sim \mathcal{D}$ for $n = \operatorname{poly}(k, d)$, outputs a clustering of $X_1, \ldots, X_n$ which is correct for all of the points, with high probability. Moreover, this algorithm runs in time $\operatorname{poly}(d, k)$.

Remark. *Note that we only need the samples from $\mathcal{D}$. We do not need to actually know the distribution or access the p.d.f.*

We note that separation $\Omega(\log k)$ is optimal for general Poincaré distributions, as Poincaré distributions include some distributions with exponential tails, for which $\Omega(\log k)$ separation is necessary to cluster. Thus, the separation that we require is almost optimal for general Poincaré distributions. We also note that one immediate consequence of this theorem, alongside Chen's recent breakthrough result [13] for KLS that $\psi \leq \exp(C\sqrt{\log d \log\log d})$ for some universal constant C, and a simple application of PCA, is that we can cluster a mixture of translates of an isotropic logconcave distribution in polynomial time, as long as the separation is $\Omega(\exp(C\sqrt{\log k \log\log k}))$.

1.2 Our Techniques

In this section, we describe how our techniques work at a high level. Our goal will be to devise a method which can, given samples $X, X' \sim \mathcal{M}$, detect whether or not X and X' are from the same components or from different ones.

We first make the following reduction. Notice that if $\mathcal{M}$ is a mixture with separation Δ, then $(X - X')/\sqrt{2}$ can be thought of as a sample from the *difference mixture* $\mathcal{M}'$. This is a mixture with $\binom{k}{2} + 1$ components, each with covariance I. It has one component with mean zero, and the means of the remaining components all have norm at least $\Delta/\sqrt{2}$. Moreover, given $X, X' \sim \mathcal{M}$, we have that $X - X'$ is drawn from the mean zero component of the difference mixture if and only if X, X' were drawn from the same component in the original mixture. Thus, to check if two samples from the original mixture X, X' are from the same component, it suffices to be able to detect, given a sample from the difference mixture, whether or not this sample comes from the mean zero component or not.

In the remainder of this section, in a slight abuse of notation, we will let $\mathcal{M}$ denote the difference mixture, we will assume it has k components with nonzero mean, and we will assume that all nonzero means of the difference mixture have norm at least Δ. We will henceforth refer to the components with nonzero mean as the nonzero components of the mixture. This reparameterization of the problem only changes things by polynomial factors, which do not impact our qualitative results.

1.2.1 Rough clustering via implicit moment tensors. The main conceptual contribution of our paper is a novel way to implicitly access degree $t = O(\log k/\log\log k)$ moment information with polynomially many samples and time. We do so by carefully constructing an implicitly maintained projection map from $\mathbb{R}^{d^t}$ down to a subspace of dimension k, which still preserves meaningful information about the nonzero components. For now, let us first focus on the case where the maximum norm of any mean in the difference mixture is upper bounded by $\text{poly}(\Delta)$, or equivalently, the maximum separation between any two components in the original mixture is at most polynomially larger than the minimum separation.

Low rank estimators for Hermite polynomials. Central to our methods is the *t-th Hermite polynomial tensor*, a classical object in probability theory, which we denote $h_t : \mathbb{R}^d \to \mathbb{R}^{d^t}$. These are explicit polynomials, and are the natural analog of the univariate Hermite polynomials to high dimensions. A well-known fact about the Hermite polynomial tensor is that

$$\mathop{\mathbb{E}}_{X \sim N(\mu, I)}[h_t(X)] = \mu^{\otimes t} \ . \tag{1}$$

Throughout the introduction, we will treat all tensors as flattened into vectors in $\mathbb{R}^{d^t}$ (in a canonical way) e.g. we view the RHS of the above as a vector in $\mathbb{R}^{d^t}$. One simple but important implication of (1) is that

$$\mathop{\mathbb{E}}_{X \sim \mathcal{M}}[h_t(X)] = \sum_{i=1}^{k} w_i \mu_i^{\otimes t} \ . \tag{2}$$

This fact will be crucial for us going forward.

However, a major bottleneck for algorithmically using the Hermite polynomial tensors is that we cannot write down $h_t(X)$ in polynomial time when t gets large, e.g. when $t = \Omega(\log k/\log\log k)$, which is the regime we will require.

To get around this, we will use a modification of the Hermite polynomial tensors that can still be used to estimate the RHS of (2) but can also be easily manipulated implicitly. In particular, we will construct a *random* polynomial $R_t : \mathbb{R}^d \to \mathbb{R}^{d^t}$ i.e. we can imagine R_t is actually a polynomial in x whose coefficients are randomly chosen. The polynomial R_t satisfies two key properties.

(1) $R_t(x)$ is an unbiased estimator for $h_t(x)$ with bounded variance i.e. for a fixed x, $\mathbb{E}[R_t(x)] = h_t(x)$ where the expectation is over the random coefficients of R_t.
(2) For any choice of the randomness in the coefficients, the polynomial $R_t(x)$ can be written as a *sum of polynomially many rank-1 tensors* i.e. tensors of the form

$$v = v_1 \otimes \ldots \otimes v_t \ , \text{ where } v_i \in \mathbb{R}^d \text{ for all } i = 1, \ldots, t \ .$$

Note that the first property implies that

$$\mathop{\mathbb{E}}_{R_t, X \sim N(\mu, I)}[R_t(X)] = \mu^{\otimes t} \ , \text{ and } \mathop{\mathbb{E}}_{R_t, X \sim \mathcal{M}}[R_t(X)] = \sum_{i=1}^{k} w_i \mu_i^{\otimes t} \ . \tag{3}$$

The second property is the main motivation behind the definition of R_t, as it implies that we can have efficient, but restricted access to it. The key point is that if our algorithm can be implemented with techniques that only require accesses to rank-1 tensors, we can implicitly access R_t in polynomial time.

Implicitly finding the span of the tensorized means. Motivated by the above discussion, our goal will be to find a projection matrix $\Pi : \mathbb{R}^{d^t} \to \mathbb{R}^k$ with the following properties:

(1) **Efficient application** If $v \in \mathbb{R}^{d^t}$ is a rank-1 tensor, then Πv can be evaluated in polynomial time.
(2) **Zero component is small** If $X \sim N(0, I)$, then $\|\Pi R_t(X)\|_2 < k^{50}$ with high probability.
(3) **Nonzero components are large** If $X \sim \mathcal{M}$ is a sample from a nonzero component of the difference mixture, then $\|\Pi R_t(X)\|_2 \geq k^{100}$ with high probability.

Given such a projection map, our clustering procedure is straightforward: given two samples X, X' from the original mixture, we apply the projection map to many copies of $R_t((X - X')/\sqrt{2})$, and cluster them in the same component if and only if their projected norm is small on average. We show that the above properties, as well as some facts about the concentration of R_t, imply that this clustering algorithm succeeds with high probability, assuming we have access to Π.

It thus remains how to construct Π. In fact, there is a natural candidate for such a map. Let $\mu_1, \ldots, \mu_k$ denote the means of the nonzero components, and let

$$S_t = \mathrm{span}\left(\{\mu_i^{\otimes t}\}_{i=1}^k\right) .$$

If we could find the projection $\Gamma_t : \mathbb{R}^{d^t} \to \mathbb{R}^k$ onto the subspace S_t, it can be verified that this projection map would satisfy Conditions (2) and (3).[1]

Moreover, there is a natural estimator for this subspace. In particular (3) implies that $\mathbb{E}[R_{2t}(X)]$ rearranged as a $d^t \times d^t$ matrix in a canonical way is exactly

$$\sum_{i=1}^k w_i \left(\mu_i^{\otimes t}\right) \left(\mu_i^{\otimes t}\right)^\top .$$

Notice that this matrix is rank k, and moreover, the span of its nonzero eigenvectors is exactly S_t. Consequently, if we could estimate this quantity, then find the projection onto the span of the top k eigenvectors, we would be done.

As alluded to earlier, the difficulty is that doing this naively would not be efficient; writing down any of these objects would take quasipolynomial time. Instead, we seek to find an implicit representation of Γ_t with the key property that it can be applied to rank-1 tensors in polynomial time.

We will do so by iteratively building an approximation to this subspace. Namely, we give a method which, given a good approximation to $\Gamma_{s-1} : \mathbb{R}^{d^{s-1}} \to \mathbb{R}^k$ which can be efficiently applied to flattenings of rank-1 tensors, constructs a good approximation to Γ_s with the same property. We do so by obtaining a good approximation to the $dk \times dk$ sized matrix

$$M_s = \sum_{i=1}^k w_i \left(\mu_i \otimes \Gamma_{s-1}\mu_i^{\otimes(s-1)}\right)\left(\mu_i \otimes \Gamma_{s-1}\mu_i^{\otimes(s-1)}\right)^\top . \quad (4)$$

Notice that M_s has rank k, and moreover, the span of its k largest eigenvectors is equal to the span of $\left\{\mu_i \otimes \Gamma_{s-1}\mu_i^{\otimes(s-1)}\right\}_{i=1}^k$. Therefore, if we let $\Pi_s : \mathbb{R}^{dk} \to \mathbb{R}^k$ denote the projection onto the span of the k largest eigenvectors of M_s, then one can easily verify that

$$\Gamma_s = \Pi_s \left(I \otimes \Gamma_{s-1}\right) . \quad (5)$$

Moreover, if Γ_{s-1} can be efficiently applied to flattenings of rank-1 tensors in $\mathbb{R}^{d^{s-1}}$, then the form of (5) immediately implies that Γ_s can also be applied efficiently to flattenings of rank-1 tensors in $\mathbb{R}^{d^s}$.

It remains to demonstrate how to efficiently approximate M_s, given Γ_{s-1}. There is again a fairly natural estimator for this matrix. Namely, each component of the sum in (4) can be formed by rearranging the length-$(dk)^2$ vector

$$\left(I \otimes \Gamma_{s-1}\right)^{\otimes 2} \mu_i^{\otimes 2s} = \mathop{\mathbb{E}}_{R_{2s}, X \sim N(\mu_i, I)} \left[\left(I \otimes \Gamma_{s-1}\right)^{\otimes 2} R_{2s}(X)\right] .$$

into a $dk \times dk$ sized matrix in the canonical way. In particular, this implies that the overall matrix is the rearrangement of the length-$(dk)^2$ vector

$$\sum_{i=1}^k w_i \left(I \otimes \Gamma_{s-1}\right)^{\otimes 2} \mu_i^{\otimes 2s} = \mathop{\mathbb{E}}_{R_{2s}, X \sim \mathcal{M}} \left[\left(I \otimes \Gamma_{s-1}\right)^{\otimes 2} R_{2s}(X)\right] \quad (6)$$

into a $dk \times dk$ sized matrix in the canonical way. Since R_{2s} is a sum of polynomially many rank-1 tensors, and by our inductive hypothesis, Γ_{s-1} can be efficiently applied to rank-1 tensors, we can efficiently estimate the right hand side of (6), given samples from $\mathcal{M}$.

Putting it all together, this allows us to approximate M_s efficiently given Γ_{s-1}, which, by (5), gives us the desired expression for Γ_s. Iterating this procedure gives us a way to estimate Γ_t as a sequence of nested projection maps, i.e.

$$\Gamma_t \approx \Pi_t \left(I \otimes \Pi_{t-1} \left(I \otimes \ldots\right)\right) ,$$

where we can compute $\Pi_1, \ldots, \Pi_t$ efficiently, given samples from $\mathcal{M}$. This form allows us to evaluate Γ_t efficiently on any flattening of a rank-1 tensor, thus satisfying Condition (1), and we previously argued that Γ_t satisfies Conditions (2) and (3). Combining all of these ingredients gives us our clustering algorithm, when the minimum and maximum separations are at most polynomially separated.

Implicit moment tensors for Poincaré distributions. So far, we have only discussed how to do this implicit moment estimation for isotropic Gaussians. It turns out that all of the quantities discussed above have very natural analogues for *any* Poincaré distribution. For instance, given any Poincaré distribution $\mathcal{D}$ with zero mean, there is an explicit polynomial tensor we call the *$\mathcal{D}$-adjusted polynomial $P_{t,\mathcal{D}}$* (see Section 4) that essentially satisfies all the same properties that we needed above. If we use these polynomials instead of the Hermite polynomial tensor, it turns out that all of these proofs directly lift to any Poincaré distribution. In fact, in the actual technical sections, we directly work with arbitrary Poincaré distributions, as everything is stated very naturally there. The resulting clustering algorithm immediately gives us Theorem 2.3.

Sample complexity. Previous approaches always needed to estimate high degree moment tensors, and as a result, their sample complexity was quasipolynomial. While we may also need to estimate fairly high degree polynomials, notice that all quantities that we will deal with will be polynomially bounded. This is because we can terminate our procedure at any t which satisfies Conditions (2) and (3). Therefore, all the quantities that we need are polynomially large. As a result, one can verify that the polynomials we construct will also only ever have polynomially large range, with high probability. Therefore, all quantities we need to estimate can be estimated using polynomially many samples. We defer the detailed proofs outlined in this discussion to Sections 4, 5, 6 and 7. Note that in those sections, we work with a general Poincaré distribution but the outline follows the description here.

1.2.2 *Fine-grained clustering for Gaussians.*

We now discuss how to handle general mixtures of isotropic Gaussians, without any assumption on the maximum separation. The problem with applying the implicit moment estimation method outlined above to this general setting is that the signal from the components in the

[1]Here and throughout the introduction, we will assume for simplicity of exposition that the vectors $\mu_1^{\otimes s}, \ldots, \mu_k^{\otimes s}$ are linearly independent, for all $s = 1, \ldots, t$, so that S_s is always a k-dimensional subspace, for all $s = 1, \ldots, t$. In general, our algorithms work even if they are not linearly independent, and will always find a subspace which contains S_t, which will suffice for our purposes.

difference mixture with relatively small mean will be drowned out by the signal from the components with much larger norm. Consequently, we can reliably cluster points from the components with large mean, but we could obtain an imperfect clustering for some components with somewhat smaller mean, and we will be unable to detect components with very small mean.

To overcome this, we devise a recursive clustering strategy. One somewhat simple approach is as follows. We first use our rough clustering algorithm described above to find a "signal direction" $v \in \mathbb{R}^d$. This direction will have the property that there is a pair of well-separated means along this direction. Thus, if we project the data points on this direction, and take only points which lie within a randomly chosen small interval on this interval, we can guarantee that with reasonable probability, we only accept points from at most half of the components of the mixture. Of course, after restricting to this interval, the resulting distribution is no longer a mixture of Gaussians. However, if we consider the projection of these accepted points to the subspace orthogonal to v, the resulting distribution will again be a mixture of fewer isotropic Gaussians. We can then recurse on this mixture with fewer components. Here, we are crucially using the fact that isotropic Gaussians remain isotropic Gaussians after slicing and projecting orthogonally.

While this strategy described above, when implemented carefully, would work down to $\Delta = \mathrm{poly}(\log k)$, it would not be able to achieve the nearly optimal separation in Theorem 2.5. To achieve the nearly optimal separation, there are several more technical details that need to be dealt with and thus there will be several additional steps in the algorithm. Due to space constraints, the fine-grained clustering steps are omitted here and deferred to the full version.

1.3 Related Work

The literature on mixture models—and Gaussian mixture models in particular—dates back to seminal work of Pearson [38] and is incredibly vast. For conciseness, we will focus only on the most related papers here. Our results are most closely related to the aforementioned line of work on studying efficient algorithms for clustering and parameter estimation under mean-separation conditions [5, 14, 15, 19, 21, 26, 31, 40, 44]. However, none of these algorithms can handle polylogarithmic separation in polynomial time.

We also note that a number of papers also generalize from mixtures of Gaussians to mixtures of more general classes of distributions [3, 26, 31, 32, 36]. These algorithms fall into two classes: either they require separation which is at least $\Omega(k^{1/2})$ or even larger, but they can handle general subgaussian distributions, or they require more structure on the higher moments of the distribution, but they can tolerate much less separation. The most general condition under which the latter is known to work is the condition commonly referred to as *certifiable hypercontractivity*, which roughly states that the Sum-of-Squares hierarchy can certify that the distribution has bounded tails. While there is no complete characterization of what distributions satisfy this condition, the most general class of distributions for which it is known to hold is the class of Poincaré distributions [31], which is also the class of distributions we consider here.

Another line of work focuses on parameter estimation for mixtures models without separation conditions [7, 19, 25, 30, 37]. These papers typically make much weaker assumptions on the components, namely, that they are statistically distinguishable, and the goal is to recover the parameters of the mixture, even in settings where clustering is impossible. However, typically, these algorithms incur a sample complexity and runtime which is exponential in the number of components; indeed, [25] demonstrate that this is tight, even in one dimension, when the Gaussians can have different variances.

To circumvent this, researchers have also considered the easier notion of *proper* or *semi-proper* learning, where the goal is to output a mixture of Gaussians which is close to the unknown mixture in statistical distance. A learning algorithm is said to be proper if its output is a mixture of k Gaussians, where k is the number of components in the true mixture, and semi-proper if it outputs a mixture of $k' \geq k$ Gaussians. While the sample complexity of proper learning is polynomial in all parameters, all known algorithms still incur a runtime which is exponential in k, even in the univariate setting [2, 6, 16, 22, 34]. When the hypothesis is only constrained to be semi-proper, polynomial time algorithms are known in the univariate setting [9, 18, 33], but these do not extend to high dimensional settings. In the more challenging high dimensional regime, a remarkable recent result of [19] demonstrate that for a mixture of isotropic Gaussians, one can achieve semi-proper learning with quasipolynomial sample complexity and runtime. They do this by explicitly constructing a small cover for the candidate means, by techniques inspired by algebraic geometry.

An even weaker goal that has been commonly considered is that of *density estimation*, where the objective is to output any hypothesis which is close to the unknown mixture in statistical distance. Efficient (in fact, nearly optimal) algorithms are again known for this problem in low dimensions, see e.g. [1, 11, 12, 18], but as was the case with proper learning, these techniques do not extend nicely to high dimensional settings.

An alternate assumption that has been considered in the literature is that the means satisfy some algebraic non-degeneracy assumptions. For instance, such assumptions are satisfied in smoothed analysis settings [4, 8, 23, 29]. Often in these settings, access to constant order moments suffice (e.g. 3rd or 4th moments), although we note that [8] is a notable exception. In contrast, we do not make any non-degeneracy assumptions, and our methods need to access much higher moments.

We also mention a line of work studying the theoretical behavior of the popular *expectation-maximization* (EM) algorithm for learning mixtures of Gaussians [15, 17, 46]. However, while the above works can prove that the dynamics of EM converge in limited settings, it is known that EM fails to converge in general, even for mixtures of 3 Gaussians [17, 45].

Finally, we note that our work bears some vague resemblance to the general line of work that uses spectral-based methods to speed up Sum-of-Squares (SoS) algorithms. Spectral techniques have been used to demonstrate to speed up SoS-based algorithms in various settings such as tensor decomposition [27, 28, 35, 41] and refuting random CSPs [39]. Similarly, it has been observed that in some settings, SoS-based algorithms can be sped up, when the SoS proofs are much smaller than the overall size of the program [24, 42]. Our algorithm shares some qualitative similarities with some of these approaches—for instance, it is based on a (fairly complicated) spectral algorithm. However, we do not know of a concrete connection between our algorithm and this line of work. It is possible, for instance, that our algorithm could be interpreted as extracting a specific randomized polynomially-sized SoS proof of identifiability, but we leave further investigations of this to future work.

2 FORMAL PROBLEM SETUP AND OUR RESULTS

In this section, we formally define the problems we consider, and state our formal results. For the remainder of this paper, we will always let $\|\cdot\|$ denote the ℓ_2 norm.

2.1 Clustering Mixtures of Poincare Distributions

The general problem that we study involves clustering mixtures of Poincare distributions. We begin with a few definitions.

Definition 2.1 (Poincare Distribution). *For a parameter σ, we say a distribution $\mathcal{D}$ on $\mathbb{R}^d$ is σ-Poincare if for all differentiable functions $f : \mathbb{R}^d \to \mathbb{R}$,*

$$\underset{z\sim\mathcal{D}}{Var}[f(z)] \leq \sigma^2 \mathop{\mathbb{E}}_{z\sim\mathcal{D}}[\|\nabla f(z)\|^2] .$$

Definition 2.2. *Let $\mathcal{D}$ be a distribution on $\mathbb{R}^d$. We use $\mathcal{D}(\mu_1)$ for $\mu_1 \in \mathbb{R}^d$ to denote the distribution obtained by shifting $\mathcal{D}$ by the vector μ_1.*

We assume that there is some σ-Poincare distribution $\mathcal{D}$ on $\mathbb{R}^d$ that we have sample access to. Since everything will be scale invariant, it will suffice to focus on the case $\sigma = 1$. For simplicity we assume that $\mathcal{D}$ has mean 0 (it is easy to reduce to this case since we can simply estimate the mean of $\mathcal{D}$ and subtract it out). We also assume that we have sample access to a mixture

$$\mathcal{M} = w_1\mathcal{D}(\mu_1) + \cdots + w_k\mathcal{D}(\mu_k)$$

where the mixing weights $w_1, \dots, w_n$ and means $\mu_1, \dots, \mu_k$ are unknown. We will assume that we are given a lower bound on the mixing weights $w_{\min}$. We consider the setting where there is some minimum separation between all pairs of means μ_i, μ_j so that the mixture is clusterable. In the proceeding sections, when we say an event happens with high probability, we mean that the failure probability is smaller than any inverse polynomial in $k, 1/w_{\min}$. Our main theorem is stated below.

Theorem 2.3. *Let $\mathcal{D}$ be a 1-Poincare distribution on $\mathbb{R}^d$. Let*

$$\mathcal{M} = w_1\mathcal{D}(\mu_1) + \cdots + w_k\mathcal{D}(\mu_k)$$

be a mixture of translated copies of $\mathcal{D}$. Let $w_{\min}, s$ be parameters such that $w_i \geq w_{\min}$ for all i and $\|\mu_i - \mu_j\| \geq s$ for all $i \neq j$. Let $\alpha = (w_{\min}/k)^{O(1)}$ be some desired accuracy (that is inverse polynomial) [2]. Assume that

$$s \geq (\log(k/w_{\min}))^{1+c}$$

for some $0 < c < 1$. Also assume that

$$\max \|\mu_i - \mu_j\| \leq s^C$$

for some C. There is an algorithm that takes

$$n = \text{poly}((kd/(w_{\min}\alpha))^{C/c})$$

samples from $\mathcal{M}$ and $\mathcal{D}$ and runs in $\text{poly}(n)$ time and with high probability, outputs estimates

$$\widetilde{w_1}, \dots, \widetilde{w_k}, \widetilde{\mu_1}, \dots, \widetilde{\mu_k}$$

such that for some permutation π on $[k]$,

$$|w_i - \widetilde{w_{\pi(i)}}| \leq \alpha, \|\mu_i - \widetilde{\mu_{\pi(i)}}\| \leq \alpha$$

for all i.

Remark. *If we could remove the assumption $\|\mu_i - \mu_j\| \leq s^C$, then we would get a complete polynomial time learning result. Still, our learning algorithm works in polynomial time for mixtures where the maximum separation is polynomially bounded in terms of the minimum separation.*

An immediate consequence of Theorem 2.3 is that we can cluster samples from the mixture with accuracy better than any inverse polynomial.

Corollary 2.4. *Under the same assumptions as Theorem 2.3, we can recover the ground-truth clustering of the samples with high probability i.e. we output k clusters $S_1, \dots S_k$ such that for some permutation π on $[k]$, the set $S_{\pi(i)}$ consists precisely of the samples from the component $\mathcal{D}(\mu_i)$ for all i.*

2.2 Clustering Mixtures of Gaussians

In the case where the distribution $\mathcal{D}$ in the setup in Section 2.1 is a Gaussian, we can obtain a stronger result that works in full generality, without any assumption about the maximum separation. The result for Gaussians also works with a smaller separation of $(\log(k/w_{\min}))^{1/2+c}$ which, as mentioned before, is essentially optimal for Gaussians.

Theorem 2.5. *Let $\mathcal{M} = w_1N(\mu_1, I) + \cdots + w_kN(\mu_k, I)$ be an unknown mixture of Gaussians in $\mathbb{R}^d$ such that $w_i \geq w_{\min}$ for all i and $\|\mu_i - \mu_j\| \geq (\log(k/w_{\min}))^{1/2+c}$ for some constant $c > 0$. Then for any desired (inverse polynomial) accuracy $\alpha \geq (w_{\min}/k)^{O(1)}$ given $n = \text{poly}((dk/(w_{\min}\alpha))^{1/c})$ samples and $\text{poly}(n)$ runtime, there is an algorithm that with high probability outputs estimates $\{\widetilde{\mu_1}, \dots, \widetilde{\mu_k}\}$ and $\{\widetilde{w_1}, \dots, \widetilde{w_k}\}$ such that for some permutation π on $[k]$, we have*

$$|w_i - \widetilde{w_{\pi(i)}}|, \|\mu_i - \widetilde{\mu_{\pi(i)}}\| \leq \alpha$$

for all $i \in [k]$.

Again, once we have estimated the parameters of $\mathcal{M}$, it is easy to cluster samples from $\mathcal{M}$ into each of the components with accuracy better than any inverse polynomial.

Corollary 2.6. *Under the same assumptions as Theorem 2.5, with high probability, we can recover the ground-truth clustering of the samples i.e. we output k clusters $S_1, \dots S_k$ such that for some permutation π on $[k]$, the set $S_{\pi(i)}$ consists precisely of the samples from the component $N(\mu_i, I)$ for all i.*

[2] If d is much larger than k and we wanted inverse polynomial accuracy like $1/d$ then we can simply decrease the parameter $w_{\min}$ (and then we would need separation $\log(dk)$ instead of $\log k$)

Note that throughout this paper, we do not actually need to know the true number of components k. All of the algorithms that we write will work if instead of the number of components being k, the number of components is upper bounded by k i.e. we only need to be told an upper bound on the number of components. In fact, we can simply use $1/w_{\min}$ as the upper bound on the number of components.

2.3 Organization

In Section 3, we introduce basic notation and prove a few basic facts that will be used later on.

Clustering Test: In Sections 4 - 7, we develop our key clustering test i.e. we show how to test if two samples are from the same component or not. Note that throughout these sections, when we work with a mixture

$$\mathcal{M} = w_1\mathcal{D}(\mu_1) + \cdots + w_k\mathcal{D}(\mu_k)\,,$$

this will correspond to the "difference mixture" of the mixture that we are trying to learn and thus our goal will be to test whether a sample came from a component with $\mu_i = 0$ or μ_i far from 0.

In Section 4, we discuss how to construct estimators for the moments of a Poincare distribution that can be manipulated implicitly. In the end, we construct a random polynomial R_t such that

$$\mathop{\mathbb{E}}_{x\sim\mathcal{D}(\mu)}[R_t(x)] = \mu^{\otimes t}$$

and such that R_t can be written as the sum of polynomially many rank-1 tensors (see Corollary 4.9). In Section 5, we describe our iterative projection technique and explain how it can be used to implicitly store and apply a projection map $\Gamma : \mathbb{R}^{d^t} \to \mathbb{R}^k$ in polynomial time to rank-1 tensors. In Section 6, we combine the techniques in Section 4 and Section 5 to achieve the following: given samples from

$$\mathcal{M} = w_1\mathcal{D}(\mu_1) + \cdots + w_k\mathcal{D}(\mu_k)$$

we can find a $k \times d^t$ projection matrix Γ_t (where $t \sim \log k/\log\log k$) whose row span essentially contains all of $\mu_1^{\otimes t}, \dots \mu_k^{\otimes t}$ (see Lemma 6.5). Finally, in Section 7, we use the projection map computed in the previous step and argue that if $x \sim \mathcal{D}(0)$ then $\|\Gamma_t R_t(x)\|$ is small and if $x \sim \mathcal{D}(\mu_i)$ for μ_i far from 0, then $\|\Gamma_t R_t(x)\|$ is large with high probability. Thus, to test a sample x, it suffices to measure the length of $\|\Gamma_t R_t(x)\|$.

Main Result for Mixtures of Poincare Distributions: In Section 8, we prove Theorem 2.3, our main result for mixtures of Poincare distributions. It will follow fairly easily from the guarantees of the clustering test in Section 7.

Main Result for Mixtures of Gaussians: Proving our main result for mixtures of Gaussians requires some additional work although all of the machinery from Sections 4 - 7 can still be used. Roughly, the two additional ingredients are

(1) Quantitatively stronger versions of the bounds in Sections 4 - 7 that exploit special properties of Gaussians in order to get down from $\log^{1+c} k$ to $\log^{1/2+c} k$ separation
(2) A recursive clustering procedure that allows us to eliminate the assumption about the maximum separation.

Due to space constraints these parts are omitted here and deferred to the full version. While the additional pieces require working through many technical details, the core of our learning algorithm is still built from the machinery in Sections 4 - 7.

3 PRELIMINARIES

We now introduce some notation that will be used throughout the paper. We use I_n to denote the $n \times n$ identity matrix. For matrices A, B we define $A \otimes_{\mathsf{kr}} B$ to be their Kronecker product. This is to avoid confusion with our notation for tensor products. For a tensor T, we use $\mathsf{flatten}(T)$ to denote the flattening of T into a vector. We assume that this is done in a canonical way throughout this paper.

3.1 Manipulating Tensors

We will need to do many manipulations with tensors later on so we first introduce some notation for working with tensors.

Definition 3.1 (Tensor Notation). *For an order-t tensor, we index its dimensions $\{1, 2, \dots, t\}$. For a partition of $[t]$ into subsets $S_1, \dots, S_a$ and tensors $T_1, \dots, T_a$ of orders $|S_1|, \dots, |S_a|$ respectively we write*

$$T_1^{(S_1)} \otimes \cdots \otimes T_a^{(S_a)}$$

to denote the tensor obtained by taking the tensor product of T_1 in the dimensions indexed by S_1, T_2 in the dimensions indexed by S_2, and so on for $T_3, \dots, T_a$.

Definition 3.2. *For a vector x (viewed as an order-1 tensor), we will use the shorthand $x^{\otimes S}$ to denote $(x^{\otimes |S|})^{(S)}$ (i.e. the product of copies x in dimensions indexed by elements of S). For example,*

$$x^{\otimes\{1,3\}} \otimes y^{\otimes\{2,4\}} = x \otimes y \otimes x \otimes y\,.$$

Definition 3.3 (Tensor Slicing). *For an order-t tensor T, we can imagine indexing its entries with indices $(\eta_1, \dots, \eta_t) \in [d_1] \times \cdots \times [d_t]$ where $d_1, \dots, d_t$ are the dimensions of T. We use the notation*

$$T_{\eta_{a_1}=b_1,\dots,\eta_{a_j}=b_j}$$

to denote the slice of T of entries whose indices satisfy the constraints $\eta_{a_1} = b_1, \dots, \eta_{a_j} = b_j$.

Definition 3.4 (Unordered Partitions). *We use $Z_t(S)$ to denote all partitions of S into t unordered, possibly empty, parts.*

Remark. *Note the partitions are not ordered so $\{\{1\},\{2\}\}$ is the same as $\{\{2\},\{1\}\}$.*

Definition 3.5. *For a collection of sets $S_1, \dots, S_t$, we define*

$$C(\{S_1, \dots, S_t\})$$

to be the number of sets among $S_1, \dots, S_t$ that are nonempty.

Definition 3.6 (Symmetrization). *Let $A_1, \dots, A_n$ be tensors such that A_i is an order a_i tensor having dimensions $\underbrace{d \times \cdots \times d}_{a_i}$ for some integers $a_1, \dots, a_n$. We define*

$$\sum_{sym}(A_1 \otimes \cdots \otimes A_n) = \sum_{\substack{S_1 \cup \cdots \cup S_n = [a_1 + \cdots + a_n] \\ S_i \cap S_j = \emptyset \\ |S_i| = a_i}} A_1^{(S_1)} \otimes \cdots \otimes A_n^{(S_n)}\,.$$

In other words, we sum over all ways to tensor $A_1, \dots, A_n$ together to form a tensor of order $a_1 + \cdots + a_n$.

3.2 Properties of Poincare Distributions

Here we state a few standard facts about Poincare distributions that will be used later on.

Fact 3.7. *Poincare distributions satisfy the following properties:*

- ***Direct Products:*** *If $\mathcal{D}_1$ and $\mathcal{D}_2$ are σ-Poincare distributions then their product $\mathcal{D}_1 \times \mathcal{D}_2$ is σ-Poincare*
- ***Linear Mappings:*** *If $\mathcal{D}$ is σ-Poincare and A is a linear mapping then the distribution Ax for $x \sim \mathcal{D}$ is $\sigma \|A\|_{op}$-Poincare*
- ***Concentration:*** *If $\mathcal{D}$ is σ-Poincare then for any L-Lipchitz function f and any parameter $t \geq 0$, we have*

$$\Pr_{z \sim \mathcal{D}}[|f(z) - \mathbb{E}[f(z)]| \geq t] \leq 6e^{-t/(\sigma L)} .$$

The following concentration inequality for samples from a Poincare distribution is also standard.

Claim 3.8. *Let $\mathcal{D}$ be a distribution in $\mathbb{R}^d$ that is 1-Poincare. Let $0 < \epsilon < 0.1$ be some parameter. Given $n \geq (d/\epsilon)^8$ independent samples $z_1, \dots, z_n \sim \mathcal{D}$, with probability at least $1 - 2^{-d/\epsilon}$, we have*

$$\left\| \frac{z_1 + \cdots + z_n}{n} - \mathbb{E}_{z \sim \mathcal{D}}[z] \right\| \leq \epsilon .$$

3.3 Basic Observations

Before we begin with the main proofs of Theorem 2.3 and Theorem 2.5, it will be useful to make a few simple reductions that allow us to make the following simplifying assumptions:

- **Means Polynomially Bounded:** For all i, we have $\|\mu_i - \mu_j\| \leq O((k/w_{\min})^2)$, and
- **Dimension Not Too High:** We have $d \leq k$.

Reducing to the case when the above assumptions hold is straightforward and omitted due to space constraints. The reductions work in both settings (general Poincare distributions and Gaussians) so in future sections, we will be able to work assuming that the above properties hold.

4 MOMENT ESTIMATION

We will now work towards proving our result for Poincare distributions. A key ingredient in our algorithm will be estimating the moment tensor of a mixture $\mathcal{M}$ i.e. for an unknown mixture

$$\mathcal{M} = w_1 \mathcal{D}(\mu_1) + \cdots + w_k \mathcal{D}(\mu_k)$$

we would like to estimate the tensor

$$w_1 \mu_1^{\otimes t} + \cdots + w_k \mu_k^{\otimes t}$$

for various values of t using samples from $\mathcal{M}$. Naturally, it suffices to consider the case where we are given samples from $\mathcal{D}(\mu)$ for some unknown μ and our goal is to estimate the tensor $\mu^{\otimes t}$. For our full algorithm, we will need to go up to $t \sim \log k / \log \log k$ but of course for such t, our estimate has to be implicit because we cannot write down the full tensor in polynomial time. In this section, we address this task by constructing an unbiased estimator with bounded variance that can be easily manipulated implicitly.

We make the following definition to simplify notation later on.

Definition 4.1. *For integers t and a distribution $\mathcal{D}$, we define the tensor*

$$D_{t,\mathcal{D}} = \mathbb{E}_{z \sim \mathcal{D}}[z^{\otimes t}] .$$

We will drop the subscript $\mathcal{D}$ when it is clear from context.

4.1 Adjusted Polynomials

First, we just construct an unbiased estimator for $\mu^{\otimes t}$ (without worrying about making it implicit). This estimator is given in the definition below.

Definition 4.2. *Let $\mathcal{D}$ be a distribution on $\mathbb{R}^d$. For $x \in \mathbb{R}^d$, define the polynomials $P_{t,\mathcal{D}}(x)$ for positive integers t as follows. $P_{0,\mathcal{D}}(x) = 1$ and for $t \geq 1$,*

$$\begin{aligned} P_{t,\mathcal{D}}(x) = x^{\otimes t} - \sum_{sym} D_{1,\mathcal{D}} \otimes P_{t-1,\mathcal{D}}(x) \\ - \sum_{sym} D_{2,\mathcal{D}} \otimes P_{t-2,\mathcal{D}}(x) - \cdots - D_{t,\mathcal{D}} . \end{aligned} \tag{7}$$

We call $P_{t,\mathcal{D}}$ the $\mathcal{D}$-adjusted polynomials and will sometimes drop the subscript $\mathcal{D}$ when it is clear from context.

Through direct computation, we can verify that the $\mathcal{D}$-adjusted polynomials indeed give an unbiased estimator for $\mu^{\otimes t}$ when given samples from $\mathcal{D}(\mu)$.

Claim 4.3. *For any $\mu \in \mathbb{R}^d$,*

$$\mathbb{E}_{z \sim \mathcal{D}(\mu)}[P_{t,\mathcal{D}}(z)] = \mu^{\otimes t} .$$

4.2 Variance Bounds for Poincare Distributions

In the previous section, we showed that the $\mathcal{D}$-adjusted polynomials give an unbiased estimator for $\mu^{\otimes t}$. We now show that they also have bounded variance when $\mathcal{D}$ is Poincare. This will rely on the following claim which shows that the $\mathcal{D}$-adjusted polynomials recurse under differentiation.

Claim 4.4. *Let $\mathcal{D}$ be a distribution on $\mathbb{R}^d$. Then*

$$\frac{\partial P_{t,\mathcal{D}}(x)}{\partial x_i} = \sum_{sym} e_i \otimes P_{t-1,\mathcal{D}}(x)$$

where we imagine $x = (x_1, \dots, x_d)$ so x_i is the i^{th} coordinate of x and

$$e_i = (\underbrace{0, \dots, 1}_{i}, \dots, 0)$$

denotes the i^{th} coordinate basis vector.

Now, by using the Poincare property, we can prove a bound on the variance of the estimator $P_{t,\mathcal{D}}(x)$.

Claim 4.5. *Let $\mathcal{D}$ be a distribution on $\mathbb{R}^d$ that is 1-Poincare. Let $v \in \mathbb{R}^{d^t}$ be a vector. Then*

$$\mathbb{E}_{z \sim \mathcal{D}(\mu)}[(v \cdot \mathit{flatten}(P_{t,\mathcal{D}}(z)))^2] \leq (\|\mu\|^2 + t^2)^t \|v\|^2 .$$

4.3 Efficient Implicit Representation

In the previous section, we showed that for $x \sim \mathcal{D}(\mu)$ for unknown μ, $P_{t,\mathcal{D}}(x)$ gives us an unbiased estimator of $\mu^{\otimes t}$ with bounded variance. Still, it is not feasible to actually compute $P_{t,\mathcal{D}}(x)$ in polynomial time because we cannot write down all of its entries and there is no nice way to implicitly work with terms such as $D_{t,\mathcal{D}}$ that appear in $P_{t,\mathcal{D}}(x)$. In this section, we construct a modified estimator that is closely related to $P_{t,\mathcal{D}}(x)$ but is also easy to work with implicitly because all of the terms will be rank-1 i.e. of the form $v_1 \otimes \dots \otimes v_t$ for some vectors $v_1, \dots, v_t \in \mathbb{R}^d$. Throughout this section, we will assume that the distribution $\mathcal{D}$ that we are working with is fixed and we will drop it from all subscripts as there will be no ambiguity.

Roughly, the way that we construct this modified estimator is that we take multiple variables $x_1, \dots, x_t \in \mathbb{R}^d$. We start with $P_t(x_1)$. We then add various products

$$P_{a_1}(x_1) \otimes \dots \otimes P_{a_t}(x_t)$$

to it in a way that when expanded as monomials, only the leading terms, which are rank-1 since they are a direct product of the form $x_1^{\otimes a_1} \otimes \dots \otimes x_t^{\otimes a_t}$, remain. If we then take $x_1 \sim \mathcal{D}(\mu)$ and $x_2, \dots, x_t \sim \mathcal{D}$, then Claim 4.3 will immediately imply that the expectation is $\mu^{\otimes t}$. The key properties are stated formally in Corollary 4.9, Corollary 4.10 and Corollary 4.11.

The following polynomial and its properties will be the key in our construction.

Definition 4.6. *For $x_1, \dots, x_t \in \mathbb{R}^d$, define the polynomial*

$$\begin{aligned} &Q_t(x_1, \dots, x_t) \\ &= \sum_{\substack{S_1 \cup \dots \cup S_t = [t] \\ |S_i \cap S_j| = 0}} \frac{(-1)^{C\{S_1,\dots,S_t\}}}{\binom{t-1}{C\{S_1,\dots,S_t\}-1}} \left(P_{|S_1|}(x_1)\right)^{(S_1)} \otimes \dots \\ &\quad \otimes \left(P_{|S_t|}(x_t)\right)^{(S_t)} . \end{aligned} \tag{8}$$

Lemma 4.7. *All nonzero monomials in Q_t either have total degree t in the variables $x_1, \dots, x_t$ or are constant.*

Now, we can easily eliminate the constant term by subtracting off $Q(x_{t+1}, \dots, x_{2t})$ for some additional variables $x_{t+1}, \dots, x_{2t}$ and we will be left with only degree-t terms. It will be immediate that the degree-t terms are all rank-1 and this will give us an estimator that can be efficiently manipulated implicitly.

Definition 4.8. *For $x_1, \dots, x_{2t} \in \mathbb{R}^d$, define the polynomial*

$$R_t(x_1, \dots, x_{2t}) = -Q_t(x_1, \dots, x_t) + Q_t(x_{t+1}, \dots, x_{2t}) .$$

Corollary 4.9. *We have the identity*

$$\begin{aligned} R_t(x_1, \dots, x_{2t}) = \sum_{\substack{S_1 \cup \dots \cup S_t = [t] \\ |S_i \cap S_j| = 0}} &\frac{(-1)^{C\{S_1,\dots,S_t\}-1}}{\binom{t-1}{C\{S_1,\dots,S_t\}-1}} \cdot \\ &\left(x_1^{\otimes S_1} \otimes \dots \otimes x_t^{\otimes S_t} - x_{t+1}^{\otimes S_1} \otimes \dots \otimes x_{2t}^{\otimes S_t}\right) . \end{aligned}$$

Corollary 4.9 gives us a convenient representation for working implicitly with $R_t(x_1, \dots, x_{2t})$. We now show why this polynomial is actually useful. In particular, we show that for $x_1 \sim \mathcal{D}(\mu)$ and $x_2, \dots, x_{2t} \sim \mathcal{D}$, $R_t(x_1, \dots, x_{2t})$ is an unbiased estimator of $\mu^{\otimes t}$ and furthermore that its variance is bounded. These properties will follow directly from the definitions of R_t, Q_t combined with Claim 4.3 and Claim 4.5.

Corollary 4.10. *We have*

$$\mathbb{E}_{z_1 \sim \mathcal{D}(\mu), z_2, \dots, z_{2t} \sim \mathcal{D}} [R_t(z_1, \dots, z_{2t})] = \mu^{\otimes t}$$

and for fixed z_1, we have

$$\mathbb{E}_{z_2, \dots, z_{2t} \sim \mathcal{D}} [R_t(z_1, \dots, z_{2t})] = P_t(z_1) .$$

Corollary 4.11. *Let $\mathcal{D}$ be a distribution that is 1-Poincare. We have*

$$\mathbb{E}_{\substack{z_1 \sim \mathcal{D}(\mu) \\ z_2, \dots, z_{2t} \sim \mathcal{D}}} \left[\textit{flatten}(R_t(z_1, \dots, z_{2t}))^{\otimes 2} \right] \leq (20t)^{2t} (\|\mu\|^{2t} + 1) I_{d^t} .$$

where recall I_{d^t} denotes the d^t-dimensional identity matrix.

5 ITERATIVE PROJECTION

In this section, we explain our technique for implicitly working with tensors that have too many entries to write down. Recall that we would like to estimate the moment tensor

$$w_1 \mu_1^{\otimes t} + \dots + w_k \mu_k^{\otimes t}$$

for

$$t \sim \frac{\log(k/w_{\min})}{\log\log(k/w_{\min})} .$$

However doing this directly requires quasipolynomial time (because there are quasipolynomially many entries). Roughly, the way we get around this issue is by, iteratively for each t, computing a k-dimensional subspace that contains the span of $\mu_1^{\otimes t}, \dots, \mu_k^{\otimes t}$. We then only need to compute the projection of $w_1 \mu_1^{\otimes t} + \dots + w_k \mu_k^{\otimes t}$ onto this subspace. Of course, the subspace and projection need to be computed implicitly because we cannot explicitly write out these expressions in polynomial time.

5.1 Nested Projection Maps

At a high level, to implicitly estimate the span of $\mu_1^{\otimes t}, \dots, \mu_k^{\otimes t}$, we will first estimate the span of $\mu_1^{\otimes t-1}, \dots, \mu_k^{\otimes t-1}$ and then bootstrap this estimate to estimate the span of $\mu_1^{\otimes t}, \dots, \mu_k^{\otimes t}$. Since we cannot actually write down the span even though it is k-dimensional (because the vectors have super-polynomial length), we will store the span implicitly through a sequence of projections. We explain the details below.

Definition 5.1 (Nested Projection). *Let $c_0 = 1$ and $c_1, \dots, c_t$ be positive integers. Let $\Pi_1 \in \mathbb{R}^{c_1 \times dc_0}, \Pi_2 \in \mathbb{R}^{c_2 \times dc_1}, \dots, \Pi_t \in \mathbb{R}^{c_t \times dc_{t-1}}$ be matrices. Define the $c_t \times d^t$ nested projection matrix*

$$\Gamma_{\Pi_t, \dots, \Pi_1} = \Pi_t (I_d \otimes_{kr} (\Pi_{t-1} (I_d \otimes_{kr} \cdots))) .$$

It is not hard to verify (see below) that when $\Pi_1, \dots, \Pi_t$ are projection matrices then $\Gamma_{\Pi_t, \dots, \Pi_1}$ is as well.

Claim 5.2. *Let $c_0 = 1$ and $c_1, \dots, c_t$ be positive integers. Let $\Pi_1 \in \mathbb{R}^{c_1 \times dc_0}, \Pi_2 \in \mathbb{R}^{c_2 \times dc_1}, \dots, \Pi_t \in \mathbb{R}^{c_t \times dc_{t-1}}$ be matrices whose rows are orthonormal. Then $\Gamma_{\Pi_t, \dots, \Pi_1}$ has orthonormal rows.*

Note that in our paper, $\Pi_1, \dots, \Pi_t$ will always have orthonormal rows so $\Gamma_{\Pi_t,\dots,\Pi_1}$ always does as well. This fact will often be used without explicitly stating it. The key point about the construction of $\Gamma_{\Pi_t,\dots,\Pi_1}$ is that instead of storing a full $c_t \times d^t$-sized matrix, it suffices to store the individual matrices $\Pi_1, \dots, \Pi_t$ which are all polynomially sized. The next important observation is that for certain vectors $v \in \mathbb{R}^{d^t}$ that are "rank-1" i.e. those that can be written in the form

$$v = \mathsf{flatten}(v_t \otimes \cdots \otimes v_1)\,,$$

the expression $\Gamma_{\Pi_t,\dots,\Pi_1} v$ can be computed efficiently. This is shown in the following claim.

Claim 5.3. *Let $c_0 = 1$ and $c_1, \dots, c_t$ be positive integers. Let $\Pi_1 \in \mathbb{R}^{c_1 \times dc_0}, \Pi_2 \in \mathbb{R}^{c_2 \times dc_1}, \dots, \Pi_t \in \mathbb{R}^{c_t \times dc_{t-1}}$ be matrices. Let $v \in \mathbb{R}^{d^t}$ satisfy $v = \mathit{flatten}(v_1 \otimes \cdots \otimes v_t)$ for some $v_1, \dots, v_t \in \mathbb{R}^d$. Then in* $\mathrm{poly}(d, t, \max(c_i))$ *time, we can compute $\Gamma_{\Pi_t,\dots,\Pi_1} v$.*

Proof. We will prove the claim by induction on t. For each $t' = 1, 2, \dots, t$, we compute

$$\Gamma_{\Pi_{t'},\dots,\Pi_1} \mathsf{flatten}(v_{t'} \otimes \cdots \otimes v_1)\,.$$

The base case of the induction is clear. To do the induction step, note that

$$\begin{aligned}&\Gamma_{\Pi_{t'+1},\dots,\Pi_1} \mathsf{flatten}(v_{t'+1} \otimes \cdots \otimes v_1)\\ &= \Pi_{t'+1} \mathsf{flatten}\left(v_{t'+1} \otimes \left(\Gamma_{\Pi_{t'},\dots,\Pi_1} \mathsf{flat}(v_{t'} \otimes \cdots \otimes v_1)\right)\right)\,.\end{aligned}$$

It is clear that this computation can be done in $\mathrm{poly}(d, t, \max(c_i))$ time so iterating this operation completes the proof. ■

Throughout our paper, we will only compute nested projections of the above form so it will be easy to verify that all steps can be implemented in polynomial time.

6 IMPLICITLY ESTIMATING THE MOMENT TENSOR

In this section, we combine the iterative projection techniques from Section 5 with the estimators from Section 4 to show how to implicitly estimate the moment tensor given sample access to the distribution $\mathcal{D}$ and the mixture

$$\mathcal{M} = w_1 \mathcal{D}(\mu_1) + \cdots + w_k \mathcal{D}(\mu_k)\,.$$

By implicitly estimate, we mean that we will compute projection matrices $\Pi_t, \dots, \Pi_1$ such that the row-span of $\Gamma_{\Pi_t,\dots,\Pi_1}$ essentially contains all of the flattenings of $\mu_1^{\otimes t}, \dots, \mu_k^{\otimes t}$.

It will be convenient to make the following definitions.

Definition 6.1. *For a mixture $\mathcal{M} = w_1\mathcal{D}(\mu_1) + \cdots + w_k\mathcal{D}(\mu_k)$, we use $T_{t,\mathcal{M}}$ to denote the tensor $w_1\mu_1^{\otimes t} + \cdots + w_k\mu_k^{\otimes t}$. We may drop the subscript $\mathcal{M}$ and just write T_t when it is clear from context.*

Definition 6.2. *For a mixture $\mathcal{M} = w_1\mathcal{D}(\mu_1) + \cdots + w_k\mathcal{D}(\mu_k)$, we define $M_{2s,\mathcal{M}}$ to be the tensor $T_{2s,\mathcal{M}}$ rearranged (in a canonical way) as a $d^s \times d^s$ square matrix. Again, we may drop the subscript $\mathcal{M}$ when it is clear from context.*

We define $\mu_{\max} = \max(1, \|\mu_1\|, \dots, \|\mu_k\|)$. We do not assume that we know $\mu_{\max}$ in advance. However, the reduction in Section 3.3 means that it suffices to consider when $\mu_{\max}$ is polynomially bounded. Also we can assume $d = k$ i.e. the dimension of the underlying space is equal to the number of components. This is because we can use the reduction in Section 3.3 and if $d < k$, then we can simply add independent standard Gaussian entries in the remaining $k - d$ dimensions.

We now describe our algorithm for implicitly estimating the moment tensor. For the remainder of this section, we will only work with a fixed mixture $\mathcal{M}$ so we will drop it from all subscripts e.g. in Definitions 6.1 and 6.2. At a high level, we will recursively compute a sequence of projection matrices $\Pi_1 \in \mathbb{R}^{k \times d}, \Pi_2, \dots, \Pi_s \in \mathbb{R}^{k \times dk}$. Our goal will be to ensure that $\Gamma_{\Pi_s,\dots,\Pi_1}$ (which is a $k \times d^s$ matrix) essentially contains the flattenings of $\mu_1^{\otimes s}, \dots, \mu_k^{\otimes s}$ in its row span.

To see how to do this, assume that we have computed $\Pi_{s-1}, \dots, \Pi_1$ so far. By the inductive hypothesis, $\Gamma_{\Pi_{s-1},\dots,\Pi_1}$ tells us a k-dimensional subspace that essentially contains the flattenings of $\mu_1^{\otimes s-1}, \dots, \mu_k^{\otimes s-1}$. Thus, we trivially have a dk dimensional subspace, given by the rows of $\left(I_d \otimes_{\mathrm{kr}} \Gamma_{\Pi_{s-1},\dots,\Pi_1}\right)$ that must essentially contain all of the flattenings of $\mu_1^{\otimes s}, \dots, \mu_k^{\otimes s}$. It remains to reduce from this dk dimensional space back to a k-dimensional space. However, we can now write everything out in this dk-dimensional space and simply run PCA and take the top-k singular subspace. Our algorithm is described in full below.

Algorithm 1 Iterative Projection Step

```
Input: Samples $z_1, \dots, z_n$ from unknown mixture
        $\mathcal{M} = w_1\mathcal{D}(\mu_1) + \cdots + w_k\mathcal{D}(\mu_k)$
Input: integer $t > 0$
Split samples into $t$ sets $S_1, \dots, S_t$ of equal size
Let $\Pi_1 = I_d$ (recall $k = d$)
for s = 2, ..., t do
    Run Estimate Moment Tensor using samples $S_s$ to get approximation $\widetilde{A_{2s}} \in \mathbb{R}^{dk \times dk}$ to
        $A_{2s} = \left(I_d \otimes_{\mathrm{kr}} \Gamma_{\Pi_{s-1},\dots,\Pi_1}\right) M_{2s} \left(I_d \otimes_{\mathrm{kr}} \Gamma_{\Pi_{s-1},\dots,\Pi_1}\right)^T$
    Let $\Pi_s \in \mathbb{R}^{k \times dk}$ have rows forming an orthonormal basis of the top $k$ singular subspace of $\widetilde{A_{2s}}$
Output: $(\Pi_t, \dots, \Pi_1)$
```

Algorithm 2 Estimate Moment Tensor

```
Input: Samples $z_1, \dots, z_n$ from unknown mixture
        $\mathcal{M} = w_1\mathcal{D}(\mu_1) + \cdots + w_k\mathcal{D}(\mu_k)$
Input: Integer $s > 0$
Input: Matrices $\Pi_{s-1} \in \mathbb{R}^{k \times dk}, \dots, \Pi_2 \in \mathbb{R}^{k \times dk}, \Pi_1 \in \mathbb{R}^{k \times d}$
for i = 1,2, ..., n do
    Independently draw samples $x_1, \dots, x_{4s-1}$ from $\mathcal{D}$
    Compute the $(kd)^2$-dimensional vector (recall Definition 4.8)
        $X_i = ((I_d \otimes_{\mathrm{kr}} \Gamma_{\Pi_{s-1},\dots,\Pi_1}) \otimes_{\mathrm{kr}} (I_d \otimes_{\mathrm{kr}} \Gamma_{\Pi_{s-1},\dots,\Pi_1}))$
            $\mathsf{flatten}\left(R_{2s}(z_i, x_1, \dots, x_{4s-1})\right)\,.$
    Let $K_i$ be the rearrangement of $X_i$ into a square $dk \times dk$-dimensional matrix
Output: $A = (K_1 + \cdots + K_n)/n$
```

The main algorithm, Algorithm 1, computes the projection matrices $\Pi_1, \dots \Pi_s$ following the outline above. As a subroutine, it needs to estimate the matrix A_{2s}. This is done in Algorithm 2 which relies on the results in Section 4, namely Corollary 4.10 and Corollary 4.11.

6.1 Efficient Implementation

A naive implementation of Algorithm 1 requires d^t time, which is too large. However, in this section, we show that we can implement all of the steps more efficiently using only $\mathrm{poly}(ndk, t^t)$ time.

Remark. *We will later show that it suffices to consider*

$$t \sim O\left(\frac{\log(k/w_{\min})}{\log\log(k/w_{\min})}\right)$$

so this runtime is actually polynomial in all parameters that we need.

Claim 6.3. *Algorithm 1 can be implemented to run in* $\mathrm{poly}(n, d, k, t^t)$ *time.*

Proof. Follows easily from Claim 5.3. ■

6.2 Accuracy Analysis

Now, we analyze the correctness of Algorithm 1 namely that the span of the rows of the matrix $\Gamma_{\Pi_t,\dots,\Pi_1}$ indeed essentially contain all of $\mu_1^{\otimes t}, \dots, \mu_k^{\otimes t}$. To simplify notation, we make the following definition.

Definition 6.4. *For all s, we define the matrix*

$$A_{2s} = \left(I_d \otimes_{kr} \Gamma_{\Pi_{s-1},\dots,\Pi_1}\right) M_{2s} \left(I_d \otimes_{kr} \Gamma_{\Pi_{s-1},\dots,\Pi_1}\right)^T .$$

Remark. *Note that in the execution of Algorithm 1, $\widetilde{A_{2s}}$ is intended to be an estimate of A_{2s}.*

The main result that we will prove is stated below. Note that this lemma does not require any assumptions about minimum mixing weights or means in the mixture. Instead, it simply says that the subspace spanned by the rows of $\Gamma_{\Pi_s,\dots,\Pi_1}$ essentially contains $\mathsf{flatten}(\mu_i^{\otimes s})$ for all components $\mathcal{D}(\mu_i)$ with mean and mixing weight bounded away from 0.

Lemma 6.5. *Let $\mathcal{D}$ be a distribution on $\mathbb{R}^d$ that is 1-Poincare and let $\mathcal{M} = w_1\mathcal{D}(\mu_1) + \cdots + w_k\mathcal{D}(\mu_k)$ be a mixture of translations of $\mathcal{D}$. Let $w^*, \epsilon > 0$ be parameters. Assume that the number of samples satisfies*

$$n \geq \left(\frac{t^t \mu_{\max}^t kd}{w^*\epsilon}\right)^C$$

for some sufficiently large universal constant C. Then with probability at least $1 - n^{-0.2}$, in the execution of Algorithm 1, the following condition holds: for all $i \in [k]$ such that $\|\mu_i\| \geq 1$ and $w_i \geq w^*$, we have

$$\left\|\Gamma_{\Pi_s,\dots,\Pi_1} \mathit{flatten}(\mu_i^{\otimes s})\right\| \geq (1 - s\epsilon)\, \|\mu_i\|^s$$

for all $s = 1, 2, \dots, t$.

Remark. *The parameter ϵ represents the desired accuracy and the parameter w^* is a weight cutoff threshold where we guarantee to recover "significant" components whose mixing weight is at least w^*.*

Roughly, the proof of Lemma 6.5 will involve following the outline at the beginning of this section but quantitatively tracking the errors more precisely. Before we prove Lemma 6.5, we first prove a preliminary claim that our estimation error $\left\|A_{2s} - \widetilde{A_{2s}}\right\|_F$ is small.

Claim 6.6. *Assume that for a fixed integer s, Algorithm 2 is run with a number of samples*

$$n \geq \left(\frac{s^s \mu_{\max}^s kd}{w^*\epsilon}\right)^C$$

for some sufficiently large universal constant C. Then with probability at least $1 - n^{-0.4}$, its output $\widetilde{A_{2s}}$ satisfies

$$\left\|\widetilde{A_{2s}} - A_{2s}\right\|_F \leq 0.5 w^* \epsilon^2 .$$

Proof. Recall that we are trying to estimate A_{2s} which is a $dk \times dk$ matrix. It will suffice for us to obtain a concentration bound for our estimate of each entry and then union bound. Recall that in Algorithm 2, we estimate A_{2s} by averaging $K_1, \dots, K_n$. By Corollary 4.10, we have that

$$\mathbb{E}[K_i] = A_{2s} \tag{9}$$

so our estimator is unbiased. Next, observe that each entry of K_i, say $K_i[a,b]$ where $1 \leq a, b \leq dk$ can be written as

$$K_i[a,b] = v \cdot \mathsf{flatten}(R_{2s}(z_i, x_1, \dots, x_{4s-1}))$$

where $v \in \mathbb{R}^{d^{2s}}$ is some unit vector (this is by Claim 5.2). By Corollary 4.11, we have

$$\mathbb{E}\left[(K_i[a,b] - A_{2s}[a,b])^2\right] \leq \mathbb{E}[K_i[a,b]^2] \leq (20s)^{2s}(\mu_{\max}^{2s} + 1)$$

where the first inequality above is true by (9). Since our final estimate is obtained by averaging over n independent samples, we have

$$\mathbb{E}\left[(\widetilde{A_{2s}}[a,b] - A_{2s}[a,b])^2\right] \leq \frac{(20s)^{2s}(\mu_{\max}^{2s} + 1)}{n} .$$

Thus, with probability at least $1 - n^{-0.5}$, we must have

$$\left|\widetilde{A_{2s}}[a,b] - A_{2s}[a,b]\right| \leq \frac{(20s)^{2s}(\mu_{\max}^{2s} + 1)}{\sqrt{n}} \leq \frac{0.5 w^* \epsilon^2}{dk}$$

where the last inequality holds as long as we choose n sufficiently large. Union bounding the above over all entries (there are only $(dk)^2$ entries to union bound over) and ensuring that n is sufficiently large, we get the desired bound. ■

Now we are ready to prove Lemma 6.5.

Proof of Lemma 6.5. We will prove the claim by induction on s. The base case for $s = 0$ is clear. Now let $i \in [k]$ be such that $\|\mu_i\| \geq 1$ and $w_i \geq w^*$. Define the vector $v_{i,s} \in \mathbb{R}^{dk}$ as

$$v_{i,s} = \mathsf{flatten}\left(\mu_i \otimes \Gamma_{\Pi_{s-1},\dots,\Pi_1} \mathsf{flatten}(\mu_i^{s-1})\right) .$$

Note that this allows us to rewrite the matrix A_{2s} as

$$A_{2s} = w_1(v_{1,s} \otimes v_{1,s}) + \cdots + w_k(v_{k,s} \otimes v_{k,s}) .$$

Let $u_{i,s}$ be the projection of $v_{i,s}$ onto the orthogonal complement of Π_s. Note that

$$u_{i,s} \cdot v_{i,s} = \left\|u_{i,s}\right\|^2 .$$

Thus, we must have

$$u_{i,s}^T A_{2s} u_{i,s} \geq w_i \left\|u_{i,s}\right\|^4 .$$

On the other hand, note that A_{2s} has rank at most k. Assuming that the hypothesis of Claim 6.6 holds, the $k+1$st singular value of $\widetilde{A_{2s}}$ has size at most $w^*\epsilon^2$. Thus,

$$u_{i,s}^T \widetilde{A_{2s}} u_{i,s} \leq 0.5 w^* \epsilon^2 \left\|u_{i,s}\right\|^2 .$$

Finally, using the hypothesis of Claim 6.6 again, we must have

$$\left| u_{i,s}^T (A_{2s} - \widetilde{A_{2s}}) u_{i,s} \right| \leq 0.5 w^* \epsilon^2 \left\|u_{i,s}\right\|^2 .$$

Putting the previous three inequalities together, we deduce that we must have

$$w_i \left\|u_{i,s}\right\|^4 \leq w^* \epsilon^2 \left\|u_{i,s}\right\|^2$$

which implies $\left\|u_{i,s}\right\| \leq \epsilon$. Also, the induction hypothesis implies that

$$\left\|\Gamma_{\Pi_{s-1},\dots,\Pi_1} \mathsf{flatten}(\mu_i^{s-1})\right\| \geq (1-(s-1)\epsilon) \left\|\mu_i\right\|^{s-1}$$

and thus

$$\left\|v_{i,s}\right\| \geq (1-(s-1)\epsilon) \left\|\mu_i\right\|^s .$$

Finally, note that

$$\begin{aligned} \left\|\Gamma_{\Pi_s,\dots,\Pi_1} \mathsf{flatten}(\mu_i^s)\right\| = \left\|v_{i,s} - u_{i,s}\right\| &\geq (1-(s-1)\epsilon) \left\|\mu_i\right\|^s - \epsilon \\ &\geq (1-s\epsilon) \left\|\mu_i\right\|^s . \end{aligned}$$

This completes the inductive step. Finally, it remains to note that the overall failure probability can be bounded by union bounding over all applications of Claim 6.6 and is clearly at most $n^{-0.2}$ as long as we choose n sufficiently large. This completes the proof. ■

7 TESTING SAMPLES USING IMPLICIT MOMENTS

Now we show how to use the projection maps $\Pi_t,\dots,\Pi_1$ computed by Algorithm 1 to test whether a sample came from a component with mean close to 0 or mean far away from 0. Roughly, given a sample z, the test simply works by computing $R_t(z, z_1, \dots, z_{2t-1})$ for $z_1,\dots,z_{2t-1} \sim \mathcal{D}$ and computing

$$\left\|\Gamma_{\Pi_t,\dots,\Pi_1} \mathsf{flatten}(R_t(z, z_1, \dots, z_{2t-1}))\right\| .$$

We output Far if the above is larger than some threshold and otherwise we output Close. For technical reasons, we will actually average over multiple independent draws for $z_1,\dots,z_{2t-1} \sim \mathcal{D}$.

Roughly, the intuition for why this test works is as follows. Note that if $z \sim \mathcal{D}$ then by Corollary 4.10,

$$\mathbb{E}\left[\Gamma_{\Pi_t,\dots,\Pi_1} \mathsf{flatten}(R_t(z, z_1, \dots, z_{2t-1}))\right] = 0$$

and if we control the variance using Corollary 4.11, then we can upper bound the length with reasonable probability. On the other hand if $z \sim \mathcal{D}(\mu_i)$ for some μ_i with large norm, then

$$\mathbb{E}\left[\Gamma_{\Pi_t,\dots,\Pi_1} \mathsf{flatten}(R_t(z, z_1, \dots, z_{2t-1}))\right] = \Gamma_{\Pi_t,\dots,\Pi_1} \mathsf{flatten}(\mu_i^{\otimes t})$$

and since the algorithm in the previous section can ensure that $\mu_i^{\otimes t}$ is essentially contained in the row span of $\Gamma_{\Pi_t,\dots,\Pi_1}$, the RHS above has large norm. The details of our algorithm for testing samples are described below.

Algorithm 3 Test Samples

```
Input: Projection matrices $\Pi_t,\dots,\Pi_2 \in \mathbb{R}^{k\times dk}, \Pi_1 \in \mathbb{R}^{k\times d}$
Input: Sample $z \in \mathbb{R}^d$ to test
Input: Threshold $\tau$, desired accuracy $\delta$
Set $n = ((10^3 t)^t/\delta)^3$
for $i = 1,2,\dots,n$ do
    Draw samples $z_1,\dots,z_{2t-1} \sim \mathcal{D}$
    Let $A_i = \Gamma_{\Pi_t,\dots,\Pi_1}\mathsf{flatten}(R_t(z, z_1,\dots,z_{2t-1}))$
Set $A = (A_1 + \dots + A_n)/n$
if $\|A\| \geq \tau$ then
    Output: Far
else
    Output: Close
```

7.1 Analysis of Test Samples

Now we analyze the behavior of Algorithm 3. The key properties that the test satisfies are summarized in the following two lemmas. Lemma 7.2 say that with $1-\delta$ probability the test will successfully output Far for samples from a component with mean far from 0 and Lemma 7.1 says that with $1-\delta$ probability, the test will successfully output Close for samples from a component with mean 0.

Note that Lemma 7.2 requires that the row span of $\Gamma_{\Pi_t,\dots,\Pi_1}$ essentially contains $\mathsf{flatten}(\mu_i^{\otimes t})$ (which can be guaranteed by Algorithm 1 and Lemma 6.5). Lemma 7.1 actually does not require anything about $\Pi_t,\dots,\Pi_1$ (other than the fact that they are actually projections).

Lemma 7.1. *Let $\mathcal{D}$ be a distribution that is 1-Poincare. Let $t \in \mathbb{N}$ and $0 < \delta < 0.01$ be some parameters. Let $\Pi_t,\dots,\Pi_2 \in \mathbb{R}^{k\times dk}, \Pi_1 \in \mathbb{R}^{k\times d}$ be any matrices whose rows are orthonormal. Let τ be some parameter satisfying $\tau \geq (20t)^t k/\delta$. Let $z \sim \mathcal{D}$. Then with probability at least $1-\delta$, Algorithm 3 run with these parameters outputs* Close *where the randomness is over z and the random choices within Algorithm 3.*

Lemma 7.2. *Let $\mathcal{D}$ be a distribution that is 1-Poincare. Let $t \in \mathbb{N}$ and $0 < \delta < 0.01$ be some parameters. Let $z \sim \mathcal{D}(\mu_i)$ where $\|\mu_i\| \geq 10^4(\log 1/\delta + t)$. Let τ be some parameter satisfying $\tau \leq (0.4\|\mu_i\|)^t$. Assume that the matrices $\Pi_t,\dots,\Pi_2 \in \mathbb{R}^{k\times dk}, \Pi_1 \in \mathbb{R}^{k\times d}$ satisfy that*

$$\left\|\Gamma_{\Pi_t,\dots,\Pi_1} \mathit{flatten}(\mu_i^{\otimes t})\right\| \geq (1-t\epsilon)\left\|\mu_i\right\|^t .$$

where

$$\epsilon < \frac{\delta}{(10t\|\mu_i\|)^{4t}} .$$

Then with probability at least $1-\delta$, Algorithm 3 run with these parameters outputs Far *(where the randomness is over z and the random choices within Algorithm 3).*

Remark. *Note that we will want to run the test with some inverse polynomial failure probability i.e. $\delta = \mathrm{poly}(k/w^*)$ for some weight threshold w^*. In order to be able to combine Lemma 7.1 and Lemma 7.2 meaningfully, we need*

$$(0.4\|\mu_i\|)^t \geq (20t)^t k/\delta .$$

If $\|\mu_i\| \geq (\log(k/w^*))^{1+c}$ for some constant $c > 0$ then setting $t \sim O\left(c^{-1}\log(k/w^*)/\log\log(k/w^*)\right)$ ensures that the above inequality is true. Note that for this setting, $t^t = \mathrm{poly}(k/w^*)$ (where we treat

c as a constant) and thus we will be able to ensure that our overall runtime is polynomial.

8 LEARNING MIXTURES OF POINCARE DISTRIBUTIONS

We are now ready to prove Theorem 2.3, our full result for mixtures of Poincare distributions. Let $\mathcal{D}' = (\mathcal{D} - \mathcal{D})/\sqrt{2}$ i.e. $\mathcal{D}'$ is the distribution of the difference between two independent samples from $\mathcal{D}$ scaled down by $\sqrt{2}$. The $\sqrt{2}$ scaling ensures that $\mathcal{D}'$ is 1-Poincare by Fact 3.7. Note that we can take pairwise differences between samples to simulate access to the mixture

$$\mathcal{M}' = (w_1^2 + \dots + w_k^2)\mathcal{D}' + \sum_{i \neq j} w_i w_j \mathcal{D}'((\mu_i - \mu_j)/\sqrt{2}) .$$

We will run Algorithm 1 and Algorithm 3 for the distribution $\mathcal{D}'$ and mixture $\mathcal{M}'$. Note that the test in Algorithm 3 will test for a pair of samples say z, z', from components $\mathcal{D}(\mu_i), \mathcal{D}(\mu_j)$ of the original mixture, whether $\|\mu_i - \mu_j\|$ is large or zero. Running this test between all pairs of samples will let us form clusters of samples that correspond to each of the components. We begin with a lemma that more precisely specifies the guarantees of this test and then Theorem 2.3 will follow easily from it.

Lemma 8.1. *Let $\mathcal{D}$ be a 1-Poincare distribution on $\mathbb{R}^d$. Let*

$$\mathcal{M} = w_1\mathcal{D}(\mu_1) + \dots + w_k\mathcal{D}(\mu_k)$$

be a mixture of translated copies of $\mathcal{D}$. Let w^*, s, δ be parameters and assume that $s \geq (\log k/(w^*\delta))^{1+c}$ for some $0 < c < 1$. Also assume that

$$\max \|\mu_i - \mu_j\| \leq \min((k/(w^*\delta))^2, s^C)$$

for some C. There is an algorithm that takes $n = \mathrm{poly}((kd/(w^*\delta))^{C/c})$ samples from $\mathcal{M}$ and $\mathcal{D}$ and runs in $\mathrm{poly}(n)$ time and achieves the following testing guarantees: for a pair of (independent) samples $z \sim \mathcal{D}(\mu_i), z' \sim \mathcal{D}(\mu_j)$,

- *If $i = j$ and $w_i \geq w^*$ then with probability $1 - \delta$ the output is accept*
- *If $w_i, w_j \geq w^*$ and $\|\mu_i - \mu_j\| \geq s$ then with probability $1 - \delta$ the output is reject*

where the randomness is over the samples z, z' and the random choices in the algorithm.

Using Lemma 8.1, it is not difficult to prove Theorem 2.3.

Proof of Theorem 2.3. Recall by the reduction in Section 3.3 that we may assume $d \leq k$ and that $\|\mu_i - \mu_j\| \leq O((k/w_{\min})^2)$ for all i, j. Now we will do the following process to estimate the means of the components.

(1) Draw a sample $z \sim \mathcal{M}$
(2) Take $m = (k/(w_{\min}\alpha))^{10^2}$ samples $z_1, \dots, z_m \sim \mathcal{M}$
(3) Use Lemma 8.1 with parameters

$$w^* = w_{\min}, \delta = (w_{\min}\alpha/k)^{10^4}$$

to test the pair of samples z, z_i for all for all $i \in [m]$
(4) Let $S \subset [m]$ be the set of all i that are Accepted and compute $\mu = \frac{1}{|S|}\sum_{i \in S} z_i$

We first argue that if z is a sample from $\mathcal{D}(\mu_i)$ for some i, then with $1 - (w_{\min}\alpha/k)^{10^2}$ probability, the procedure returns μ such that $\|\mu - \mu_i\| \leq 0.1\alpha$. This is because by the guarantees of Lemma 8.1, with probability at least $1 - (w_{\min}\alpha/k)^{10^3}$ the test will accept all samples from among $z_1, \dots, z_m$ that are from the component $\mathcal{D}(\mu_i)$ and reject all of the others. Also, with high probability, there will be at least $(k/(w_{\min}\alpha))^{99}$ samples from the component $\mathcal{D}(\mu_i)$ so by Claim 3.8 and union bounding all of the failure probabilities, we have that if z is a sample from $\mathcal{D}(\mu_i)$ then $\|\mu - \mu_i\| \leq 0.1\alpha$ with probability at least $1 - (w_{\min}\alpha/k)^{10^2}$.

Now it suffices to repeat the procedure in steps $1 - 4$ for $l = (k/(w_{\min}\alpha))^{10^2}$ independent samples $z \sim \mathcal{M}$. This gives us a list of means say $S = \{\widetilde{\mu_1}, \dots, \widetilde{\mu_l}\}$. The previous argument implies that most of these estimates will be close to one of the true means and that all of the true means will be represented. To ensure that with high probability we output exactly one estimate corresponding to each true mean and no extraneous estimates, we do a sort of majority voting.

We will inspect the estimates in S one at a time and decide whether to output them or not. Let T be the set of estimates that we will output. Note that T is initially empty. Now for each i, let S_i be the subset of $\{\widetilde{\mu_1}, \dots, \widetilde{\mu_l}\}$ consisting of all means with $\|\widetilde{\mu_j} - \widetilde{\mu_i}\| \leq 0.2\alpha$. If $|S_i| \geq 0.9w_{\min}l$ and $\widetilde{\mu_i}$ is not within α of any element of T then add $\widetilde{\mu_i}$ to T. Otherwise do nothing.

We claim that with high probability, this procedure returns one estimate corresponding to each true means and nothing extraneous. First, with high probability there will be no extraneous outputs because if say $\widetilde{\mu_i}$ is at least 0.5α away from all of the true means, then with high probability we will have $|S_i| < 0.9w_{\min}l$. Now it is also clear that with high probability we output exactly one estimate corresponding to each true mean (since l is sufficiently large that with high probability we get enough samples from each component). Thus, with high probability, the final output will be a set of means that are within α of the true means up to some permutation. Once we have learned the means, we can learn the mixing weights by simply taking fresh samples from $\mathcal{M}$ and clustering since with high probability, we can uniquely identify which component a sample came from. This completes the proof. ■

The clustering guarantee in Corollary 2.4 follows as an immediate consequence of Theorem 2.3.

Proof of Corollary 2.4. Let the estimated means computed by Theorem 2.3 for $\alpha = (w_{\min}/k)^{10}$ be $\widetilde{\mu_1}, \dots, \widetilde{\mu_k}$. Now for all $j_1, j_2 \in [k]$ with $j_1 \neq j_2$, let

$$v_{j_1 j_2} = \frac{\widetilde{\mu_{j_1}} - \widetilde{\mu_{j_2}}}{\|\widetilde{\mu_{j_1}} - \widetilde{\mu_{j_2}}\|} .$$

Now given a sample z from $\mathcal{M}$, we compute the index j such that for all j_1, j_2, we have

$$|v_{j_1 j_2} \cdot (\widetilde{\mu_j} - z)| \leq (\log(k/w_{\min}))^{1+0.5c} .$$

Note that by the guarantees of Theorem 2.3, there is a permutation π such that $\|\widetilde{\mu_{\pi(i)}} - \mu_i\| \leq \alpha$ for all i. If z is a sample from $\mathcal{D}(\mu_i)$, then by the tail bound in Fact 3.7, with high probability the unique index j that satisfies the above is exactly $j = \pi(i)$ and thus, we recover the ground truth clustering with high probability. ■

9 LEARNING MIXTURES OF GAUSSIANS

See the full version.

ACKNOWLEDGMENTS

This work was partially done during an internship at Microsoft Research. This work was supported in part by an NSF Graduate Research Fellowship and a Fannie and John Hertz Foundation Fellowship.

REFERENCES

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

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

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

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

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

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

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

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

[9] Aditya Bhaskara, Ananda Suresh, and Morteza Zadimoghaddam. 2015. Sparse solutions to nonnegative linear systems and applications. In *Artificial Intelligence and Statistics*. PMLR, 83–92.

[10] Matthew Brennan, Guy Bresler, Samuel B Hopkins, Jerry Li, and Tselil Schramm. 2020. Statistical query algorithms and low-degree tests are almost equivalent. *arXiv preprint arXiv:2009.06107* (2020).

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

[12] Siu-On Chan, Ilias Diakonikolas, Rocco A Servedio, and Xiaorui Sun. 2014. Near-optimal density estimation in near-linear time using variable-width histograms. *arXiv preprint arXiv:1411.0169* (2014).

[13] Yuansi Chen. 2021. An almost constant lower bound of the isoperimetric coefficient in the KLS conjecture. *Geometric and Functional Analysis* 31, 1 (2021), 34–61.

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

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

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

[17] Constantinos Daskalakis, Christos Tzamos, and Manolis Zampetakis. 2017. Ten steps of EM suffice for mixtures of two Gaussians. In *Conference on Learning Theory*. PMLR, 704–710.

[18] Luc Devroye and Gábor Lugosi. 2001. *Combinatorial methods in density estimation*. Springer Science & Business Media.

[19] Ilias Diakonikolas and Daniel M Kane. 2020. Small covers for near-zero sets of polynomials and learning latent variable models. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 184–195.

[20] Ilias Diakonikolas, Daniel M Kane, and Alistair Stewart. 2017. Statistical query lower bounds for robust estimation of high-dimensional gaussians and gaussian mixtures. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 73–84.

[21] Ilias Diakonikolas, Daniel M Kane, and Alistair Stewart. 2018. List-decodable robust mean estimation and learning mixtures of spherical Gaussians. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*. 1047–1060.

[22] Jon Feldman, Ryan O'Donnell, and Rocco A Servedio. 2008. Learning mixtures of product distributions over discrete domains. *SIAM J. Comput.* 37, 5 (2008), 1536–1564.

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

[24] Venkatesan Guruswami and Ali Kemal Sinop. 2012. Faster SDP hierarchy solvers for local rounding algorithms. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*. IEEE, 197–206.

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

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

[27] Samuel B Hopkins, Tselil Schramm, and Jonathan Shi. 2019. A robust spectral algorithm for overcomplete tensor decomposition. In *Conference on Learning Theory*. PMLR, 1683–1722.

[28] Samuel B Hopkins, Tselil Schramm, Jonathan Shi, and David Steurer. 2016. Fast spectral algorithms from sum-of-squares proofs: tensor decomposition and planted sparse vectors. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*. 178–191.

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

[30] Adam Tauman Kalai, Ankur Moitra, and Gregory Valiant. 2010. Efficiently learning mixtures of two Gaussians. In *Proceedings of the forty-second ACM symposium on Theory of computing*. 553–562.

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

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

[33] Jerry Li, Allen Liu, and Ankur Moitra. 2021. Sparsification for Sums of Exponentials and its Algorithmic Applications. *arXiv preprint arXiv:2106.02774* (2021).

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

[35] Tengyu Ma, Jonathan Shi, and David Steurer. 2016. Polynomial-time tensor decompositions with sum-of-squares. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 438–446.

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

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

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

[39] Prasad Raghavendra, Satish Rao, and Tselil Schramm. 2017. Strongly refuting random csps below the spectral threshold. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*. 121–131.

[40] Oded Regev and Aravindan Vijayaraghavan. 2017. On Learning Mixtures of Well-Separated Gaussians. arXiv:1710.11592 [cs.DS]

[41] Tselil Schramm and David Steurer. 2017. Fast and robust tensor decomposition with applications to dictionary learning. In *Conference on Learning Theory*. PMLR, 1760–1793.

[42] David Steurer and Stefan Tiegel. 2021. SoS degree reduction with applications to clustering and robust moment estimation. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 374–393.

[43] Yin Tat Lee and Santosh S Vempala. 2018. The Kannan-Lovász-Simonovits Conjecture. *arXiv e-prints* (2018), arXiv–1807.

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

[45] CF Jeff Wu. 1983. On the convergence properties of the EM algorithm. *The Annals of statistics* (1983), 95–103.

[46] Ji Xu, Daniel Hsu, and Arian Maleki. 2016. Global analysis of expectation maximization for mixtures of two gaussians. *arXiv preprint arXiv:1608.07630* (2016).