Settling the Robust Learnability of Mixtures of Gaussians

Allen Liu
cli568@mit.edu
Massachusetts Institute of Technology
Cambridge, Massachusetts, USA

Ankur Moitra
moitra@mit.edu
Massachusetts Institute of Technology
Cambridge, Massachusetts, USA

ABSTRACT
This work represents a natural coalescence of two important lines of work – learning mixtures of Gaussians and algorithmic robust statistics. In particular we give the first provably robust algorithm for learning mixtures of any constant number of Gaussians. We require only mild assumptions on the mixing weights (bounded fractionality) and that the total variation distance between components is bounded away from zero. At the heart of our algorithm is a new method for proving dimension-independent polynomial identifiability through applying a carefully chosen sequence of differential operations to certain generating functions that not only encode the parameters we would like to learn but also the system of polynomial equations we would like to solve. We show how the symbolic identities we derive can be directly used to analyze a natural sum-of-squares relaxation.

CCS CONCEPTS
• Theory of computation → Unsupervised learning and clustering.

KEYWORDS
mixtures of gaussians, robustness, sum of squares

ACM Reference Format:
Allen Liu and Ankur Moitra. 2021. Settling the Robust Learnability of Mixtures of Gaussians. In Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC ’21), June 21–25, 2021, Virtual, Italy. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3406325.3451084

1 INTRODUCTION
This work represents a natural coalescence of two important lines of work – learning mixtures of Gaussians and algorithmic robust statistics – that we describe next: In 1894 Karl Pearson [46] introduced mixture models and asked:

Is there a statistically efficient method for learning the parameters of a mixture of Gaussians from samples?

Mixtures of Gaussians are natural models for when data is believed to come from two or more heterogenous sources. Since then they have found a wide variety of applications spanning statistics, biology, physics and computer science. The textbook approach for fitting the parameters is to use the maximum likelihood estimator. However it is not clear how many samples it requires to estimate the parameters up to some desired accuracy. Even worse it is hard to compute in high-dimensions [3].

In a seminal work, Sanjoy Dasgupta [18] introduced the problem to theoretical computer science and asked:

Is there an efficient algorithm for learning the parameters?

Many early works were based on clustering the samples into which component generated them [1, 3, 15, 19, 40, 51]. However when the components overlap non-trivially this is no longer possible. Nevertheless Kalai, Moitra and Valiant [36] gave an algorithm for learning the parameters of a mixture of two Gaussians that works even if the components are almost entirely overlapping. Their approach was based on reducing the high-dimensional problem to a series of one-dimensional problems and exploiting the structure of the moments. In particular they proved that every mixture of two Gaussians is uniquely determined by its first six moments. Subsequently Moitra and Valiant [44] and Belkin and Sinha [12] were able to give an algorithm for learning the parameters of a mixture of any constant number of Gaussians. These algorithms crucially made use of even higher moments along with several new ingredients like methods for separating out submixtures that are flat along some directions and difficult to directly learn. There are also approaches based on tensor decompositions [14, 25, 32] that get polynomial dependence on the number of components assuming that the parameters are non-degenerate and subject to some kind of smoothing. However all of these algorithms break down when the data does not exactly come from a mixture of Gaussians. In fact in Karl Pearson’s original application [46], and in many others, mixtures of Gaussians are only intended as an approximation to the true data generating process.

The field of robust statistics was launched by the seminal works of John Tukey [49, 50] and Peter Huber [33] and seeks to address this kind of shortcoming by designing estimators that are provably robust to some fraction of their data being adversarially corrupted. The field had many successes and explicated some of the general principles behind what makes estimators robust [26, 34]. Provably robust estimators were discovered for fundamental tasks such as estimating the mean and covariance of a distribution and for linear regression. There are a variety of types of robustness guarantees but the crucial point is that these estimators can all tolerate a constant fraction of corruptions that is independent of the dimension. However all of these estimators turn out to be hard to compute in high-dimensions [13, 27, 35].

Recently Diakonikolas et al. [21] and Lai et al. [41] designed the first provably robust and computationally efficient estimators for the mean and covariance. They operate under some kind of
assumption on the uncorrupted data – either that they come from a simple generative model like a single Gaussian or that they have bounded moments. To put this in perspective, without corruptions this is a trivial learning task because if you want to learn the mean and covariance for any distribution with bounded moments you can just use the empirical mean and empirical covariance respectively. Algorithmic robust statistics has transformed into a highly active area [6, 7, 16, 17, 22–24, 29, 38, 39, 42, 47] with many successes. Since then, a much sought-after goal has been to answer the following challenge:

Is there a provably robust and computationally efficient algorithm for learning mixtures of Gaussians? Can we robustify the existing learning results?

There has been steady progress on this problem. Diakonikolas et al. [21] gave a robust algorithm for learning mixtures of spherical Gaussians. In recent breakthroughs Bakshi and Kothari [5] and Diakonikolas et al. [20] gave a robust algorithm for learning clusterable mixtures of Gaussians and building on this, Kane [37] gave a robust algorithm for learning mixtures of two Gaussians. We note that these works do place some mild restrictions on the mixing weights and the variances. In particular they need the mixing weights to have bounded fractionality and the variances of all components to be nonzero in all directions.

The algorithms of Bakshi and Kothari [5] and Diakonikolas et al. [20] rely on the powerful sum-of-squares hierarchy [45]. One view is that it finds an operator, called the pseudo-expectation, that maps low degree polynomials to real values. Moreover a large number of consistency constraints are imposed that force it to in some vague sense behave like taking the expectation over a distribution on assignments to the variables. It gives a natural way to incorporate systems of polynomial constraints into a relaxation which can model complex primitives like selecting a large subset of the samples and enforcing that they have approximately the same types of moment bounds that hold for a single Gaussian. Of course the real challenge is that you need some way to reason about the pseudo-expectation operator that only uses certain types of allowable steps that can be derived through the constraints that you enforced in the relaxation.

1.1 Key Challenges

It is believed that the sum-of-squares hierarchy might actually be the key to solving the more general problem of robustly learning a mixture of any constant number of Gaussians. However there are some obstacles that need to be overcome:

Polynomial Identifiability. Behind every polynomial time algorithm for learning the parameters of a mixture model is an argument for why there cannot be two mixtures with noticeably different parameters that produce almost the same distribution. In fact we need quantitative bounds that say any two mixtures that are $\epsilon$-different must be at least poly($\epsilon$, 1/$d$) different according to some natural family of test functions, usually the set of low degree polynomials. Here $d$ is the dimension. This is called polynomial identifiability [43, 48]. Because we allow a polynomial dependence on 1/$d$ it often does not matter too much how we measure the differences between two mixtures, either in terms of some natural parameter distance between their components or in terms of the total variation distance, again between their components.

However we need much stronger bounds when it comes to robust learning problems where we want to be able to tolerate a constant fraction of corruptions that is dimension independent. In particular we need a richer family of test functions with the property that whenever we have two mixtures whose components are $\epsilon$-different in total variation distance there is some function in the family that has at least poly($\epsilon$) discrepancy when we take the difference between its expectations with respect to the two distributions (and its variance must also be bounded). We will call this dimension independent polynomial identifiability. Recall that the non-robust learning algorithms for mixtures of Gaussians reduce to a series of one-dimensional problems. Unfortunately this strategy inherently introduces polynomial factors in $d$ and it cannot give what we are after. For the special case of clusterable mixtures of Gaussians, Bakshi and Kothari [5] and Diakonikolas et al. [20] proved dimension independent polynomial identifiability and their approach was based on classifying the ways in which two single Gaussians can have total variation distance close to one. When it comes to the more general problem of handling mixtures where the components can overlap non-trivially it seems difficult to follow the same route because we can no longer match components from the two mixtures to each other and almost cancel them both out.

1.2 Our Techniques and Main Result

Actually we overcome both obstacles using the same approach. First we store the relevant moments and formal variables we would like to solve for in certain generating functions. Let us describe the setup. For simplicity, assume that the mixture is in isotropic position. First we have the unknown parameters of the mixture. Let $\mu_i$ and $I + \Sigma_i$ denote the mean and covariance of the $i$th component. Let $w_i$ be its mixing weight. Second we have the indeterminates we would like to solve for. Let $u_1, u_2, \cdots, u_k$ be a collection of $d$-dimensional vectors and let $v_1, v_2, \cdots, v_k$ be a collection of $D$-dimensional vectors where $D = \binom{d}{2} + d$. The intention is for the $u_i$’s to be an orthonormal basis for the span of the $\mu_i$’s and for the $v_i$’s to be an orthonormal basis for the span of the $\Sigma_i$’s. We will also guess the mixing weights $\tilde{w}_i$ and guess the coefficients in the linear combinations that express each $\mu_i$ in terms of the $u_j$’s and similarly for $v_i$ and the $\Sigma_j$’s. Let $\tilde{\mu}_i$ and $\tilde{\Sigma}_i$ be these expressions. Third we have a $d$-dimensional vector $X$ of formal variables and one auxiliary formal variable $y$. 

Reasoning About the Sum-of-Squares Relaxation. The sum-of-squares hierarchy is a general framework for coming up with large and powerful semidefinite programming relaxations that can be applied to many sorts of problems [9–11, 31]. However it is often quite challenging to understand whether or not it works and/or to identify, out of all the constraints that are enforced on the pseudo-expectation, which ones are actually useful in the analysis [28, 30]. What makes matters especially challenging in our setting is that it is clear the structure of the higher moments of a mixture of Gaussians must play a major role. But how exactly do we leverage them in our analysis?
We will work with the following generating functions. First let

\[ F(y) = \sum_{i=1}^{k} w_i e^{\mu_i(X)y + \frac{1}{2} \Sigma_i(X)y^2} \]

Here we have used the notation that \( \mu_i(X) \) denotes the inner product of \( \mu_i \) with the \( d \)-dimensional vector \( X \) and that \( \Sigma_i(X) \) denotes the quadratic form of \( X \) on \( \Sigma_i \). Second let

\[ \tilde{F}(y) = \sum_{i=1}^{k} \tilde{w}_i e^{\mu_i(X)y + \frac{1}{2} \Sigma_i(X)y^2} \]

As is familiar from elementary combinatorics we can tease out important properties of the generating function by applying carefully chosen operators that involve differentiation. This requires a lot more bookkeeping than usual because there are unknown parameters of the mixture, indeterminates and formal variables. But it turns out that there are simple differential operators we can apply which can isolate components. To gain some intuition, consider the operator

\[ \mathcal{D}_i = \frac{\partial}{\partial y} - (\mu_i(X) + \Sigma_i(X)y). \]

Note that

\[ \mathcal{D}_i \left( e^{\mu_i(X)y + \frac{1}{2} \Sigma_i(X)y^2} \right) = 0, \]

in other words, this operator annihilates the \( i \)-th component. Thus, by composing such operators we can annihilate all but one of the components in \( F \). On the other hand, note that applying differential operators is just a rearrangement of the polynomials that show up in the infinite sum representation of the generating function but using differential operators and generating functions in exponential form gives a particularly convenient way to derive useful expressions that would otherwise be extremely complex to write down.

Ultimately we derive a symbolic identity

\[ \tilde{w}_k \prod_{i=1}^{k} \left( \tilde{\Sigma}_k(X) - \Sigma_i(X) \right)^{2i-1} \prod_{i=1}^{k-1} \left( \tilde{\Sigma}_k(X) - \Sigma_i(X) \right)^{2k+1} \]

\[ = \sum_{i=1}^{m} P_i(X)(\tilde{h}_i(X) - h_i(X)) \]  

(1)

where \( m \) is a function of \( k \). In the above, the \( h_i(X) \)'s are the expectations of the multivariate Hermite polynomials for the true mixture we would like to learn and the \( \tilde{h}_i \)'s are the expectations of the multivariate Hermite polynomials for the hypothetical mixture, i.e. the mixture with means \( \tilde{\mu}_i \) and covariances \( I + \tilde{\Sigma}_i \). A more detailed explanation of Hermite polynomials is given in Section 3 but for now we may simply think of them as modified moments. The reason that we use Hermite polynomials instead of standard polynomials in the indeterminates that we can write down. Also the \( P_i \)'s are polynomials in everything: the unknown parameters, the indeterminates and the formal variables (except for \( y \)). To set up an SOS system, we obtain robust estimates \( \tilde{h}_i \) for the expectations of the Hermite polynomials \( h_i(X) \) for the true mixture that we can compute from existing techniques in the literature. We then enforce that the expectations of the Hermite polynomials for the hypothetical mixture \( \tilde{h}_i(X) \) are close to these robust estimates where closeness is defined in terms of the distance between their coefficient vectors.

It is not immediately clear why the expression in (1) ought to be useful for solving the SOS system that we set up. After all, we cannot explicitly compute it because it depends on things we do not have, like the true parameters, the \( \mu_i \)'s and \( \Sigma_i \)'s. However the sum-of-squares relaxation enforces that the pseudo-expectation operator assigns values to polynomials in the indeterminates in a way that behaves like an actual distribution on solutions when we are evaluating certain types of low degree expressions that contain the one above. So even though we do not know the actual polynomials in the identity, they exist and the fact that they are enforced is enough to ensure that we can estimate the covariance of the \( k \)-th component. We stress that this is just the high-level picture and many more details are needed to fill it in.

Using these techniques, we come to the main result of our paper, which is a polynomial-time algorithm for robustly learning the parameters of a high-dimensional mixture of a constant number of Gaussians. Our main theorem is (informally) stated below. A formal statement can be found in Theorem 8.2.

**Theorem 1.1.** Let \( k \) be a constant. Let \( M = w_1 G_1 + \cdots + w_k G_k \) be a mixture of Gaussians in \( \mathbb{R}^d \) whose components are non-degenerate and such that the mixing weights have bounded fractionality and 7V distances between different components are lower bounded. (Both of these bounds can be any function of \( k \).) Given \( \text{poly}(d,\epsilon) \) samples from \( M \) that are \( \epsilon \)-corrupted, there is an algorithm that runs in time \( \text{poly}(n) \) and with high probability outputs a set of mixing weights \( \tilde{w}_1, \ldots, \tilde{w}_k \) and components \( \tilde{G}_1, \ldots, \tilde{G}_k \) that are \( \text{poly}(\epsilon) \)-close to the true components (up to some permutation).

**Remark.** The assumption of bounded fractionality stems from an issue in previous work \cite{20} about learning clusterable mixtures of Gaussians i.e. when the components have essentially no overlap. We use some of their subroutines in our algorithm for clustering the mixture into submixtures where the components are not too far apart from each other (see Section 6). While \cite{20} claims to handle general mixing weights, the analysis of the algorithm in \cite{20} is only done in detail for the case of uniform mixing weights and the argument in Appendix C for reducing from general mixing weights to uniform mixing weights does not work. We do believe that the analysis in \cite{20} can be extended to handle general mixing weights and this would immediately generalize to our result as well. Our other subroutines do not require bounded fractionality.

### 1.3 Proof Overview

The proof of our main theorem can be broken down into several steps. We first present our main contribution, an algorithm for learning mixtures of Gaussians when no pair of components are too far apart. We introduce the necessary generating function machinery in Section 3 and then present our algorithm in Sections 4 and 5. Specifically, in Section 4 we show how to learn the parameters once we have estimates for the Hermite polynomials of the true mixture.
And in Section 5, we show how to robustly estimate the Hermite polynomials, using similar techniques to [37].

To complete our full algorithm for learning general mixtures of Gaussians, we combine our aforementioned results with a clustering algorithm similar to [20]. Combining these algorithms, we prove that our algorithm outputs a mixture that is close to the true mixture in TV distance. This is done in Sections 6 and 7. We then prove identifiability in Section 8, implying that our algorithm actually learns the true parameters.

1.4 Concurrent Work

In independent and concurrent work, Bakshi et. al. [4] obtain a related result using rather different techniques. There is a few key differences. We learn the mixture to accuracy $\epsilon^{Ω(k)}$ while their result only achieves accuracy $(1/\log(1/\epsilon))^{Ω(k)}$, an exponentially worse guarantee. Also, our result solves parameter learning — i.e., we estimate the parameters of the true mixture — while their algorithm solves proper density estimation — i.e., it outputs a mixture of $k$ Gaussians that is close to the true density. Their algorithm does not need any lower bound on the mixing weights or on the pairwise separation of the components. In fact lower bounds on these quantities are necessary for parameter learning. However, our algorithm needs to make an even stronger assumption about the bounded fractionality of the mixing weights.

2 PRELIMINARIES

2.1 Problem Setup

We use $N(\mu, \Sigma)$ to denote a Gaussian with mean $\mu$ and covariance $\Sigma$. We use $d_{\text{TV}}(D, D')$ to denote the total variation distance between two distributions $D, D'$. We begin by formally defining the problem that we will study. First we define the contamination model. This is a standard definition from robust learning (see e.g. [20]).

Definition 2.1 (Strong Contamination Model). We say that a set of vectors $Y_1, \ldots, Y_n$ is an $\epsilon$-corrupted sample from a distribution $D$ over $\mathbb{R}^d$ if it is generated as follows. First $X_1, \ldots, X_n$ are sampled i.i.d. from $D$. Then a (malicious, computationally unbounded) adversary observes $X_1, \ldots, X_n$ and replaces up to $\epsilon n$ of them with any vectors it chooses. The adversary may then reorder the vectors arbitrarily and output them as $Y_1, \ldots, Y_n$.

In this paper, we study the following problem. There is an unknown mixture of Gaussians

$$M = w_1 G_1 + \cdots + w_k G_k$$

where $G_i = N(\mu_i, \Sigma_i). We receive an $\epsilon$-corrupted sample $Y_1, \ldots, Y_n$ from $M$ where $n = \text{poly}(d/\epsilon)$. The goal is to output a set of parameters $\hat{\mu}_1, \ldots, \hat{\mu}_k$ and $(\hat{\Sigma}_1, \ldots, \hat{\Sigma}_k)$ that are poly($\epsilon$) close to the true parameters in the sense that there exists a permutation $\pi$ on $[k]$ such that for all $i$

$$|w_i - \hat{w}_{\pi(i)}| d_{\text{TV}}\left(N(\mu_i, \Sigma_i), N(\hat{\mu}_{\pi(i)}, \hat{\Sigma}_{\pi(i)})\right) \leq \text{poly}(\epsilon).$$

Throughout our paper, we will assume that all of the Gaussians that we consider have variance at least $\text{poly}(\epsilon/d)$ and at most $\text{poly}(d/\epsilon)$ in all directions i.e. they are not too flat. This implies that their covariance matrices are invertible so we may write expressions such as $\Sigma_i^{-1}$. We will also make the following assumptions about the mixture:

- The $w_i$ are rational with denominator at most $A$
- For all $i \neq j$, $d_{TV}(G_i, G_j) > b$

for some positive constants $A, b$. Note that a lower bound on the minimum mixing weight and a lower bound on the TV distance between components is necessary for parameter learning. Throughout this paper, we treat $k, A, b$ as constants — i.e. $A$ and $b$ could be any function of $k$ — and when we say polynomial, the exponent may depend on these parameters. We are primarily interested in dependence on $\epsilon$ and $d$ (the dimension of the space).

2.2 Sum of Squares Proofs

We will make repeated use of the Sum of Squares (SOS) proof system. We review a few basic facts here (see [8] for a more extensive treatment). We begin with a standard definition of pseudoexpectations. Pseudoexpectations will be used repeatedly later on.

Definition 2.2. Let $x_1, \ldots, x_n$ be indeterminates. A degree-$k$ pseudoexpectation $\mathbb{E}$ is a linear map

$$\mathbb{E} : \mathbb{R}[x_1, \ldots, x_n]_{\leq k} \to \mathbb{R}$$

from degree-$k$ polynomials to $\mathbb{R}$, such that $\mathbb{E}[p(x)^2] \geq 0$ for any $p$ of degree at most $k/2$ and $\mathbb{E}[1] = 1$. For a set of polynomial constraints $\mathcal{A} = \{p_1(x) \geq 0, \ldots, p_m(x) \geq 0\}$, we say that $\mathbb{E}$ satisfies $\mathcal{A}$ if

$$\mathbb{E}[p_i(x)] \geq 0$$

for all polynomials $s(x)$ and $i \in [m]$ such that $s(x)^2 p_i(x)$ has degree at most $k$.

The key fact is that given a set of polynomial constraints, we can solve for a constant-degree pseudoexpectation that satisfies those constraints (or determine that none exist) in polynomial time as it reduces to solving a polynomially sized SDP.

Theorem 2.3 (SOS Algorithm [8]). There is an algorithm that takes a natural number $k$ and a satisfiable system of polynomial inequalities $\mathcal{A}$ in variables $x_1, \ldots, x_n$ with coefficients at most $2^\omega$ containing an inequality of the form $\|x\|^2 \leq M$ for some real number $M$ and returns in time $n^{O(k)}$ a degree-$k$ pseudoexpectation $\mathbb{E}$ which satisfies $\mathcal{A}$ up to error $2^{-n}$.

Note that there are a few technical details with regards to only being able to compute a pseudoexpectation that nearly satisfies the constraints. These technicalities do not affect our proof (as $2^{-n}$ errors will be negligible) so we will simply assume that we can compute a pseudoexpectation that exactly satisfies the constraints. See [8] for more details about these technicalities.

Finally, we state a few simple inequalities that will be used repeatedly later on.

Claim 2.4 (Cauchy Schwarz for Pseudo-distributions). Let $f, g$ be polynomials of degree at most $k$ in indeterminates $x = (x_1, \ldots, x_n)$. Then for any degree $k$ pseudoexpectation,

$$\mathbb{E}[fg] \leq \sqrt{\mathbb{E}[f^2]} \sqrt{\mathbb{E}[g^2]}.$$
We now introduce the generating function machinery that we will use in our learning algorithm. We begin with a standard definition.

**Definition 3.1.** Let $\mathcal{H}_m(x)$ be the univariate Hermite polynomials defined by the recurrence

$$\mathcal{H}_m(x) = x\mathcal{H}_{m-1}(x) - (m-1)\mathcal{H}_{m-2}(x).$$

Note that $\mathcal{H}_m(x)$, the degree of each nonzero monomials has the same parity as $m$. In light of this, we can write the following:

**Definition 3.2.** Let $\mathcal{H}_m(x, y^2)$ be the homogenized Hermite polynomials e.g. $\mathcal{H}_2(x, y^2) = x^2 - y^2$, $\mathcal{H}_3(x, y^2) = x^3 - 3xy^2$.

It will be important to note the following fact:

**Claim 3.3.** We have

$$e^{xx-\frac{1}{2}y^2} = \sum_{m=0}^{\infty} \frac{1}{m!} \mathcal{H}_m(x, y^2) z^m$$

where the RHS is viewed as a formal power series in $z$ whose coefficients are polynomials in $x, y$.

Now we define a multivariate version of the Hermite polynomials.

**Definition 3.4.** Let $H_m(X, z) = (z_1 X_1 + \cdots + z_d X_d, X_1^2 + \cdots + X_d^2)$ be a formal polynomial in variables $X = (X_1, \ldots, X_d)$ whose coefficients are polynomials in $d$ variables $z_1, \ldots, z_d$ which is given by

$$H_m(X, z) = H_m(z_1 X_1 + \cdots + z_d X_d, X_1^2 + \cdots + X_d^2)$$

Note that $H_m$ is homogeneous of degree $m$ as a polynomial in $X_1, \ldots, X_d$.

**Definition 3.5.** For a distribution $D$ on $\mathbb{R}^d$, we let

$$h_{m,D}(X) = \mathbb{E}(z_1, \ldots, z_d) - D[H_m(X, z)]$$

where we take the expectation of $H_m$ over $(z_1, \ldots, z_d)$ drawn from $D$. Note that $h_{m,D}(X)$ is a polynomial in $(X_1, \ldots, X_d)$. We will omit the $D$ in the subscript when it is clear from context. Moreover for a mixture of Gaussians

$$M = \frac{w_1 N(\mu_1, \Sigma_1) + \cdots + w_k N(\mu_k, \Sigma_k)}{w_1 + \cdots + w_k}$$

we will refer to the Hermite polynomials $h_{m,M}$ as the Hermite polynomials of the mixture.

We remark that if there is a mixture

$$M = \frac{w_1 N(\mu_1, \Sigma_1) + \cdots + w_k N(\mu_k, \Sigma_k)}{w_1 + \cdots + w_k}$$

where instead of real numbers, the $w_i, \mu_i, \Sigma_i$ are given in terms of indeterminates, the Hermite polynomials will be polynomials in those indeterminates. We will repeatedly make use of this abstraction later on.

The first important observation is that the Hermite polynomials for Gaussians can be written in a simple closed form via generating functions.

**Claim 3.6.** Let $D = N(\mu, I + \Sigma)$. Let $a(X) = \mu \cdot X$ and $b(X) = X^T \Sigma X$.

Then

$$e^{a(X)y + \frac{1}{2}b(X)y^2} = \sum_{m=0}^{\infty} \frac{1}{m!} h_{m,D}(X)y^m$$

as formal power series in $y$.

**Proof.** It follows from direct computation. \hfill \blacksquare

We now have two simple corollaries to the above.

**Corollary 3.7.** Let $M = w_1 N(\mu_1, I + \Sigma_1) + \cdots + w_k N(\mu_k, I + \Sigma_k)$. Let $a(X) = \mu \cdot X$ and $b(X) = X^T \Sigma X$. Then

$$\sum_{m=0}^{\infty} \frac{1}{m!} h_{m,M}(X)y^m = w_1 e^{a(X)y + b(X)/2} + \cdots + w_k e^{a(X)y + b(X)/2}$$

**Corollary 3.8.** Let $M = w_1 N(\mu_1, I + \Sigma_1) + \cdots + w_k N(\mu_k, I + \Sigma_k)$. Let $a(X) = \mu \cdot X$ and $b(X) = X^T \Sigma X$. Then the Hermite polynomials $h_{m,M}(X)$ can be written as a linear combination of products of the $a(X), b(X)$ such that the number of terms in the sum, the number of terms in each product, and the coefficients in the linear combination are all bounded as functions of $m, k$.

The next important insight is that the generating functions for the Hermite polynomials behave nicely under certain differential operators. We can use these differential operators to derive identities that the Hermite polynomials must satisfy and these identities will be a crucial ingredient in our learning algorithm.

The proceeding claims all follow from direct computation.

**Claim 3.9.** Let $\partial$ denote the differential operator with respect to $y$. If

$$f(y) = P(y, X)e^{a(X)y + \frac{1}{2}b(X)y^2}$$

where $P$ is a polynomial in $y$ of degree $k$ (whose coefficients are polynomials in $X$) then

$$(\partial - (a(X) + yb(X)))f(y) = Q(y, X)e^{a(X)y + \frac{1}{2}b(X)y^2}$$

where $Q$ is a polynomial in $y$ with degree exactly $k - 1$ whose leading coefficient is $k$ times the leading coefficient of $P$.

**Corollary 3.10.** Let $\partial$ denote the differential operator with respect to $y$. If

$$f(y) = P(y, X)e^{a(X)y + \frac{1}{2}b(X)y^2}$$

where $P$ is a polynomial in $y$ of degree $k$ then

$$(\partial - (a(X) + yb(X)))^k f(y) = 0.$$
and if \( b(X) = d(X) \) then

\[
(\partial - (c(X) + yd(X)))^k f(y) = Q(y, X)e^{a(X)y + \frac{1}{2}b(X)y^2}
\]

where \( Q \) is a polynomial of degree \( k \) in \( y \) with leading coefficient

\[
L(X)(a(X) - c(X))^k
\]

### 3.1 Polynomial Factorizations

The analysis of our SOS-based learning algorithm will rely on manipulations of Hermite polynomials. An important piece of our analysis is understanding how the coefficients of polynomials behave under addition and (polynomial) multiplication. Specifically, if we have two polynomials \( f(X), g(X) \) and we have bounds on the coefficients of \( f \) and \( g \), we now want to give bounds on the coefficients of the polynomials \( f(X) + g(X) \) and \( f(X)g(X) \). Due to space constraints, all of the proofs of the results in this section are omitted.

**Definition 3.12.** For a polynomial \( f(X) \) in the \( d \) variables \( X_1, \ldots, X_d \) with real coefficients define \( v(f) \) to be the vectorization of the coefficients. (We will assume this is done in a consistent manner so that the same coordinate of vectorizations of two polynomials corresponds to the same coordinate of monomials.) We will frequently consider expressions of the form \( \|v(f)\| \) i.e. the \( L^2 \) norm of the coefficient vector.

**Definition 3.13.** For a polynomial \( A(X) \) of degree \( k \) in \( d \) variables \( X_1, \ldots, X_d \) and a vector \( v \in \mathbb{R}^d \) with nonnegative integer entries summing to at most \( k \), we use \( A_v \) to denote the corresponding coefficient of \( A \).

First, we prove a simple result about the norm of the vectorization of a sum of polynomials.

**Claim 3.14.** Let \( f_1, \ldots, f_m \) be polynomials in \( X_1, \ldots, X_d \) whose coefficients are polynomials in formal variables \( u_1, \ldots, u_n \) of degree \( O_k(1) \). Then

\[
\|v(f_1 + \cdots + f_m)\|^2 \leq m(\|v(f_1)\|^2 + \cdots + \|v(f_m)\|^2)
\]

Furthermore, the difference can be written as a sum of squares of polynomials of degree \( O_k(1) \) in \( u_1, \ldots, u_n \).

Next, we upper bound the norm of the vectorization of a product of polynomials.

**Claim 3.15.** Let \( f, g, h_1, \ldots, h_k \) be polynomials in \( X_1, \ldots, X_d \) of degree at most \( k \) with coefficients that are polynomials in formal variables \( u_1, \ldots, u_n \) of degree \( O_k(1) \) Then for any pseudoexpectation \( \mathbb{E} \) of degree \( C_k \) for some sufficiently large constant \( C_k \) depending only on \( k \),

\[
\mathbb{E}[\|v(h_1)\|^2 \cdots \|v(h_k)\|^2 \|v(fg)\|^2] \\
\leq O_k(1)\mathbb{E}[\|v(h_1)\|^2 \cdots \|v(h_k)\|^2 \|v(f)\|^2 \|v(g)\|^2]
\]

where the pseudoexpectation operates on polynomials in \( u_1, \ldots, u_n \).

Now we present the final result in this section, a lower bound on the norm of the vectorization of the product of polynomials.

**Claim 3.16.** Let \( f, g, h_1, \ldots, h_k \) be polynomials in \( X_1, \ldots, X_d \) of degree at most \( k \) with coefficients that are polynomials in formal variables \( u_1, \ldots, u_n \) of degree \( O_k(1) \). Then for any pseudoexpectation \( \mathbb{E} \) of degree \( C_k \) for some sufficiently large constant \( C_k \) depending only on \( k \),

\[
\mathbb{E}[\|v(h_1)\|^2 \cdots \|v(h_k)\|^2 \|v(fg)\|^2] \\
\geq \Omega_k(1)\mathbb{E}[\|v(h_1)\|^2 \cdots \|v(h_k)\|^2 \|v(f)\|^2 \|v(g)\|^2]
\]

where the pseudoexpectation operates on polynomials in \( u_1, \ldots, u_n \).

### 4 COMPONENTS ARE NOT FAR APART

Now we are ready to present our main contribution: an algorithm that learns the parameters of a mixture of Gaussians \( M = w_1G_1 + \cdots + w_kG_k \) from an \( \epsilon \)-corrupted sample when the components are not too far apart. In this section, we will assume that the mixture is in nearly isotropic position and that we have estimates for the Hermite polynomials. We will show how to learn the parameters from these estimates. In the next section, Section 5, we show how to actually place the mixture in isotropic position and obtain estimates for the Hermite polynomials.

We use the following conventions:

- The true means and covariances are given by
  \[
  (\mu_1, I + \Sigma_1), \ldots, (\mu_k, I + \Sigma_k)
  \]
- The true mixing weights are \( w_1, \ldots, w_k \) and are all bounded below by some value \( w_{\min} \)
- \( \Delta \) is an upper bound that we have on \( \|\mu_i\| \) and \( \|\Sigma_i\| \) i.e. the components are not too far separated.
- \( \|\mu_i - \mu_j\|_2 + \|\Sigma_i - \Sigma_j\|_2 \geq \epsilon \) for all \( i \neq j \) i.e. no pair of components is too close.
- We should think of \( w_{\min} \) as being at least \( \epsilon^{\beta} \) and \( \Delta \) being at most \( \epsilon^{-r} \) for some sufficiently small value of \( r > 0 \).
- Let the Hermite polynomials for the true mixture be given by \( h_1 = h_1, h_2 = h_2, \ldots \) where
  \[
  M = w_1N(\mu_1, I + \Sigma_1) + \cdots + w_kN(\mu_k, I + \Sigma_k)
  \]

In this section we assume that we have the following:

- Estimates \( \mathbb{E}_h(X) \) for the Hermite polynomials such that
  \[
  \left\| v(\mathbb{E}_h(X) - h_i(X)) \right\|^2 \leq \epsilon' = \text{poly}(\epsilon)
  \]
  and our only interaction with the actual samples is through these estimates. We will show how to obtain these estimates in Section 5 (closely mirroring the method in [37]).

The main theorem that we prove in this section is as follows.

**Theorem 4.1.** Let \( \epsilon' \) be a parameter that is sufficiently small in terms of \( k \). There is a sufficiently small function \( f(k) \) and a sufficiently large function \( F(k) \) such that if

\[
M = w_1N(\mu_1, I + \Sigma_1) + \cdots + w_kN(\mu_k, I + \Sigma_k)
\]

is a mixture of Gaussians with

- \( \|\mu_i\|_2, \|\Sigma_i\|_2 \leq \Delta \) for all \( i \)
- \( \|\mu_i - \mu_j\|_2 + \|\Sigma_i - \Sigma_j\|_2 \geq \epsilon \) for all \( i \neq j \)
- \( w_1, \ldots, w_k \geq w_{\min} \)
We will let $D$ be the Hermite polynomials for all $i \leq F(k)$ such that
\[
\|\ell(T_i(X) - h_i(X))\|^2 \leq \epsilon'
\]
where $h_i$ are the Hermite polynomials for the true mixture $M$, then there is an algorithm that returns poly($1/\epsilon'$)$D_i(k)$ candidate mixtures, at least one of which satisfies
\[
\|w_i - \bar{w}_i\| + \mu_i - \bar{\mu}_i\| + \|\Sigma_i - \bar{\Sigma}_i\|_2 \leq \epsilon'\epsilon(k)
\]
for all $i$.

Informally, assuming that the parameters of the components of the mixture are bounded by poly($1/\epsilon'$) and that their separation is at least poly($\epsilon'$), given $\epsilon'$-accurate estimates for the Hermite polynomials, we can learn the parameters of the mixture to within Frobenius error poly($\epsilon'$).

### 4.1 Reducing to All Pairs of Parameters Equal or Separated

We claim that it suffices to work under the following assumption. All pairs of parameters are either separated of equal. More specifically, for each pair of parameters $\mu_i, \mu_j$ (and same for $\Sigma_i, \Sigma_j$), either $\mu_i = \mu_j$ or
\[
\|\mu_i - \mu_j\|_2 \geq c
\]

Essentially, we can prove that there is some scale for which we can treat parameters whose separation is smaller than that scale as equal and otherwise the parameters will be sufficiently separated.

The details of the reduction are omitted due to space constraints.

### 4.2 SOS Program Setup

Our algorithm for learning the parameters when given estimates of the Hermite polynomials involves solving an SOS program. Here we set up the SOS program that we will solve.

We will let $D = \binom{d}{2} + d$. We think of mapping between symmetric $d \times d$ matrices and $\mathbb{R}^D$ as
\[
\begin{bmatrix}
  a_{11} & \cdots & a_{1d} \\
  \vdots & \ddots & \vdots \\
  a_{d1} & \cdots & a_{dd}
\end{bmatrix} \leftrightarrow (a_{11}, 2a_{12}, 2a_{13}, \ldots, a_{dd})
\]

**Definition 4.2 (Parameter Solving Program $S$).** We will have the following variables
- $u_i = (u_{i1}, \ldots, u_{id}), \ldots, u_k = (u_{k1}, \ldots, u_{kd})$
- $v_i = (v_{i1(1)}, \ldots, v_{i1(d,d)}, \ldots, v_{ik(1)}, \ldots, v_{ik(d,d)})$

In the above $u_i, \ldots, u_k \in \mathbb{R}^d$ and $v_i, \ldots, v_k \in \mathbb{R}^{D}$. Our goal will be to solve for these variables in a way so that the solutions form orthonormal bases for the span of the $\mu_i$ and the span of the $\Sigma_i$. Note $u_{i1}, \ldots, u_{ik}$ live in $\mathbb{R}^D$ because the $\Sigma_i$ must be symmetric.

We guess coefficients $a_{ij}, b_{ij}$ where $i, j \in [k]$ expressing the means and covariances in this orthonormal basis. We ensure that the guesses satisfy the property that for every pair of vectors $A_i = (a_{i1}, \ldots, a_{ik}), A_j = (a_{j1}, \ldots, a_{jk})$ either $A_i = A_j$ or
\[
\|A_i - A_j\|_2 \geq \frac{c}{2}
\]

and similarly for $B_i, B_j$. We ensure that
\[
\|A_i\|_2 \leq 2\Delta
\]

Ensure similar conditions for the $\{B_i\}$. We also guess the mixing weights $\bar{w}_1, \ldots, \bar{w}_k$ and ensure that our guesses are all at least $\min w/2$.

Now we set up the constraints. Let $C$ be a sufficiently large integer depending only on $k$. Define $\bar{\mu}_i = a_{i1}u_{i1} + \cdots + a_{ik}u_{ik}$ and define $\Sigma_i$ similarly. These are linear expressions in the variables that we are solving for. Now consider the hypothetical mixture with mixing weights $\bar{w}_i$, means $\bar{\mu}_i$, and covariances $I + \Sigma_i$. The Hermite polynomials for this hypothetical mixture $h_i(x)$ can be written as formal polynomials in $X = (X_1, \ldots, X_d)$ with coefficients that are polynomials in $u, v$. Note that we can explicitly write down these Hermite polynomials. The set of constraints for our SOS system is as follows:

- $\|u_i\|_2^2 = 1$ for all $1 \leq i \leq k$
- $\|v_i\|_2^2 = 1$ for all $1 \leq i \leq k$
- $u_i \cdot u_j = 0$ for all $i \neq j$
- $v_i \cdot v_j = 0$ for all $i \neq j$
- For all $p = 1, 2, \ldots, C$
  \[
  \|\ell(h_p(X) - \bar{h}_p(X))\|^2 \leq 100\epsilon'
  \]

Note that we can explicitly write down the last set of constraints because we have estimates $h_i$.

It is important to note that the $\bar{w}_i, A_i, B_i$ are real numbers. We will attempt to solve the system for each of our guesses and show that for some set of guesses, we obtain a solution from which we can recover the parameters. We can brute-force search over an $\epsilon'$-net because there are only $O_k(1)$ parameters to guess. We call the SOS program that we set up $S$.

### 4.3 Analysis

We now prove a set of properties that must be satisfied by any pseudoexpectation of degree $C$, satisfying $S$ where $C_k$ is a sufficiently large constant depending only on $k$. What we would ideally want to show is that
- The span of the $\Sigma_i$ is close to the span of the $\Sigma_i$
- The span of the $\mu_i$ is close to the span of the $\mu_i$

However, it appears to be difficult to prove a statement of the above form within an SOS framework. Instead, we will look at the pseudoexpectations of the matrices
\[
M_i = \mathbb{E}[\Sigma_i \Sigma_i^T]
\]

(Where $\Sigma_i$ is viewed as a length-$D$ vector so $\Sigma_i \Sigma_i^T$ is a $D \times D$ matrix.)

The two key properties that we will prove about these matrices are in Lemmas 4.11 and 4.12.

Roughly Lemma 4.11 says that any singular vector that corresponds to a large singular value of $M_i$ must be close to the span of the $\Sigma_i$. Lemma 4.12 says that any vector $v$ that has large projection onto the subspace spanned by the $\{\Sigma_i\}$ must have the property that $v^TM_i v$ is large for some $i$. Putting these together, we can take the top-$k$ principal components of each of $M_1, \ldots, M_k$ and show that the span of these essentially contains the span of the $\{\Sigma_i\}$ (this last step is done outside the SOS framework). We can now brute-force over an
Consider the power series

\[ P_0(X) + P_1(X)h_1(X) + \cdots + P_m(X)h_m(X) = 0 \]

as an algebraic identity over formal variables \(X_1, \ldots, X_d\) and \(\{u_i\}, \{v_i\}\).

**Proof.** The desired result follows from direct computation and Corollary 3.10. The details are omitted here. \(\square\)

**Claim 4.6.** Consider the power series

\[ D_{k-1}^{2k} \cdots D_k^{2k} \cdots D_1^1 \left( F \right) \]

For any \(m\), the coefficient of \(y^m\) when the above is written as a formal power series can be written in the form

\[ P_0(X) + P_1(X)h_1(X) + \cdots + P_m(X)h_m(X) = 0 \]

where \(m'\) depends only on \(m\) and \(k\), each of the \(P_i\) is \(m\)-simple, and we have

\[ P_0(X) + P_1(X)h_1(X) + \cdots + P_m(X)h_m(X) = 0 \]

as an algebraic identity over formal variables \(X_1, \ldots, X_d\) and \(\{u_i\}, \{v_i\}\).

**Proof.** The proof is identical to the proof of Claim 4.5. \(\square\)

Note that the polynomials \(P_i\) in Claim 4.5 and Claim 4.6 are not necessarily the same.

**4.3.2 Warm-up: All Pairs of Parameters are Separated.** As a warm-up, we first analyze the case where all pairs of true parameters \(\mu_i, \mu_j\) and \(\Sigma_i, \Sigma_j\) satisfy \(\|\mu_i - \mu_j\|_2 \geq c\) and \(\|\Sigma_i - \Sigma_j\|_2 \geq c\). We will show how to deal with the general case where parameters may be separated or equal in Section 4.3.4.

We can assume that our guesses satisfy \(\|A_i - A_j\|_2 \geq c/2\) and \(\|B_i - B_j\|_2 \geq c/2\) for all \(i, j\). The key expressions to consider are applying the following differential operators

\[ \mathcal{D} = \mathcal{D}_k^{2k} \cdots \mathcal{D}_1^1 \]

where \(m'\) depends only on \(m\) and \(k\), each of the \(P_i\) is \(m\)-simple, and we have

\[ P_0(X) + P_1(X)h_1(X) + \cdots + P_m(X)h_m(X) = 0 \]

as an algebraic identity over formal variables \(X_1, \ldots, X_d\) and \(\{u_i\}, \{v_i\}\).

**Proof.** The proof is identical to the proof of Claim 4.5. \(\square\)

Note that the polynomials \(P_i\) in Claim 4.5 and Claim 4.6 are not necessarily the same.
as a formal power series in $y$. Its evaluation at $y = 0$ is

$$C_k \delta_k \kappa \prod_{i=1}^{k} (\Sigma_k(X) - \Sigma_1(X)\omega_i)^{2i - 1} \prod_{i=1}^{k - 1} (\Sigma_k(X) - \Sigma_1(X))^{2k+i - 1}$$

where $C_k$ is a constant depending only on $k$.

**Proof.** The desired result follows by combining Claim 3.11, Corollary 3.10, and Claim 3.9.

**Claim 4.8.** Write

$$D_k^2 \delta_k \kappa \prod_{i=1}^{k} (\Sigma_k(X) - \Sigma_1(X))^{2i - 1} \prod_{i=1}^{k - 1} (\Sigma_k(X) - \Sigma_1(X))^{2k+i - 1}$$

as a formal power series in $y$. Its evaluation at $y = 0$ is

$$C_k \delta_k \kappa \prod_{i=1}^{k} (\Sigma_k(X) - \Sigma_1(X))^{2i - 1} \prod_{i=1}^{k - 1} (\Sigma_k(X) - \Sigma_1(X))^{2k+i - 1}$$

where $C_k$ is a constant depending only on $k$.

**Proof.** This can be proved using the same method as Claim 4.7.

Combining the previous two claims with Claim 4.5 and Claim 4.6, we can write the expressions for the leading coefficients as polynomial combinations of the Hermite polynomials.

**Lemma 4.9.** Consider the polynomial

$$\frac{\delta_k}{\kappa} \prod_{i=1}^{k} (\Sigma_k(X) - \Sigma_1(X))^{2i - 1} \prod_{i=1}^{k - 1} (\Sigma_k(X) - \Sigma_1(X))^{2k+i - 1}$$

It can be written in the form

$$P_0(X) + P_1(X)\tilde{h}_1(X) + \cdots + P_m(X)\tilde{h}_m(X)$$

where $m$ is a function of $k$, each of the $P_i$ is $m$-simple, and we have $P_0(X) + P_1(X)h_1(X) + \cdots + P_m(X)h_m(X) = 0$

as an algebraic identity over formal variables $X_1, \ldots, X_d$ and $\{u_i\}, \{v_i\}$.

**Proof.** Consider the power series

$$\delta_k \kappa \prod_{i=1}^{k} (\Sigma_k(X) - \Sigma_1(X))^{2i - 1} \prod_{i=1}^{k - 1} (\Sigma_k(X) - \Sigma_1(X))^{2k+i - 1}$$

Now using Claim 4.7 and repeating the proof of Claim 4.5, we get the desired.

Similarly, we have:

**Lemma 4.10.** Consider the polynomial

$$\frac{\delta_k}{\kappa} \prod_{i=1}^{k} (\Sigma_k(X) - \Sigma_1(X))^{2i - 1} \prod_{i=1}^{k - 1} (\Sigma_k(X) - \Sigma_1(X))^{2k+i - 1}$$

It can be written in the form

$$P_0(X) + P_1(X)h_1(X) + \cdots + P_m(X)h_m(X)$$

where $m$ is a function of $k$, each of the $P_i$ is $m$-simple, and we have $P_0(X) + P_1(X)\tilde{h}_1(X) + \cdots + P_m(X)\tilde{h}_m(X) = 0$

as an algebraic identity over formal variables $X_1, \ldots, X_d$ and $\{u_i\}, \{v_i\}$.

Everything we’ve done so far has been symbolic manipulations and the claims in this section are all true as algebraic identities. We are now ready to analyze the SOS program. Note the polynomials $P_0, \ldots, P_m$ in Lemma 4.9 are unknown because they depend on the true parameters. This is fine because we will simply use their existence to deduce properties of pseudoexpectations that solve the SOS-system $S$.

Let $U$ be the subspace spanned by the true $\mu_1, \ldots, \mu_k$ and let $V$ denote the subspace spanned by the true (flattened) $\Sigma_1, \ldots, \Sigma_k$. We will use $\mathcal{F}_1, \mathcal{F}_V, \mathcal{F}_V^⊥$ to denote projections onto $V$ and the orthogonal complement of $V$ (and similar for $U, U^⊥$). Note that these are linear maps.

Our goal now will be to show that $V$ is essentially contained within the span of the union of the top $k$ principal components of the matrices

$$\mathbb{E}[\Sigma_k^1 \Sigma_k^T], \ldots, \mathbb{E}[\Sigma_k^k \Sigma_k^T]$$

This gives us a $k^2$-dimensional space that essentially contains $V$ and then we can guess the true covariance matrices via brute force search. In the first key lemma, we prove that the matrix $\mathbb{E}[\Sigma_k^i \Sigma_k^j]$ lives almost entirely within the subspace $V$.

**Lemma 4.11.** Let $\mathbb{E}$ be a pseudoexpectation of degree $C_k$ for some sufficiently large constant $C_k$ depending only on $k$ that solves $S$. Consider the matrix

$$M = \mathbb{E}[\Sigma_k \Sigma_k^T]$$

where by this we mean we construct the $D \times D$ matrix $\Sigma_k \Sigma_k^T$ whose entries are quadratic in the variables $\{u_i\}, \{v_i\}$ and then take the entry-wise pseudoexpectation. Then

$$\text{Tr}_{V^⊥}(M) \leq e^{C_k} \text{O}_k(1) \left( \frac{\Delta}{\text{w}_{\text{min}}^C} \right)^{O_k(1)}$$

where $\text{Tr}_{V^⊥}(M)$ denotes the trace of $M$ on the subspace $V^⊥$.

**Proof.** We provide a sketch of the proof here with details deferred to the full version. start from Lemma 4.9 which gives

$$\frac{\delta_k}{\kappa} \prod_{i=1}^{k} (\Sigma_k(X) - \Sigma_1(X))^{2i - 1} \prod_{i=1}^{k - 1} (\Sigma_k(X) - \Sigma_1(X))^{2k+i - 1} = P_1(X)h_1(X) + \cdots + P_m(X)h_m(X)$$

where $m = O_k(1)$ and the above holds as a symbolic identity. Note that the vectorization of the above quantity is a vector whose entries are polynomials in the indeterminates of the SOS system $S$. We look at the pseudoexpectation of the squared $L^2$ norm of this vector. We rely on the properties from Lemma 4.9 combined with Claims 3.14 and 3.15 to upper bound the pseudoexpectation on the RHS. We use Claim 3.16 and the fact that

$$\|\text{v}(\Sigma_k(X) - \Sigma_1(X))\|^2 \geq \text{SOS} \|\Gamma_{V^⊥}(\Sigma_k)\|^2$$

(recall that $\Gamma_{V^⊥}$ is a projection map with unknown but constant coefficients) to lower bound the pseudoexpectation on the LHS. Finally, noting that

$$\text{Tr}_{V^⊥}(M) = \mathbb{E} \left[ \|\Gamma_{V^⊥}(\Sigma_k)\| \right]$$

we get the desired inequality.
In the next key lemma, we prove that any vector that has nontrivial projection onto $V$ must also have nontrivial projection onto $\overline{E}[\Sigma_i \Sigma_i^T]$ for some $i$.

**Lemma 4.12.** Let $\overline{E}$ be a pseudoexpectation of degree $C_k$ for some sufficiently large constant $C_k$ depending only on $k$ that solves $\mathcal{S}$. Consider the matrix

$$N \equiv \sum_{i=1}^{k} \overline{E}[\Sigma_i \Sigma_i^T]$$

where by this we mean we construct the $D \times D$ matrix whose entries are quadratic in the variables $\{u\}, \{v\}$ and then take the entry-wise pseudoexpectation. Then for any unit vector $z \in \mathbb{R}^D$,

$$z^T N z \geq \left( \frac{w_{\min}(z \cdot \Sigma_k)O_k(1) - O_k(1)e^\epsilon\Delta O_k(1)}{O_k(1)\Delta O_k(1)} \right)^2$$

as long as

$$w_{\min}(z \cdot \Sigma_k)O_k(1) > O_k(1)e^\epsilon\Delta O_k(1).$$

**Proof.** The proof follows the same rough outline as the proof of Lemma 4.11 except we start from the symbolic identity in Lemma 4.10 (instead of Lemma 4.9).

Putting Lemmas 4.11 and 4.12 together, we now prove that $V$ is essentially contained within the span of the union of the top principal components of $\overline{E}[\Sigma_i \Sigma_i^T]$ over all $i$.

**Lemma 4.13.** For each $i$, let $M_i$ be the $D \times D$ matrix given by $M_i = \overline{E}[\Sigma_i \Sigma_i^T]$. Assume that for a sufficiently small function $f$ depending only on $k$,

$$\Delta \leq e^{-f(k)}$$

and $w_{\min} \cdot \epsilon \geq e^\epsilon f(k)$

Let $V_i$ be the subspace spanned by the top $k$ singular vectors of $M_i$. Then for all $i$, the projection of the true covariance matrix $\Sigma_i$ onto the orthogonal complement of $\text{spn}(V_1, \ldots, V_k)$ has length at most $e^{O(1)}$.

**Proof.** We combine Lemma 4.11 and Lemma 4.12. The details are omitted due to space constraints.

### 4.3.3 Finishing Up: Finding the Covariances and then the Means

Now we can brute-force search over the subspace spanned by the union of the top $k$ singular vectors of $M_1, \ldots, M_k$. Note that the SOS system $\mathcal{S}$ is clearly feasible as it is solved when the $u_i, v_i$ form orthonormal bases for the true subspaces and the $w_i, A_i, B_i$ are within $\epsilon^{O(1)}$ of the true values (i.e. the values needed to express the true means and covariances in the orthonormal basis given by the $u_i, v_i$).

Thus, brute forcing over an $\epsilon^{O(1)}$-net for the $w_i, A_i, B_i$, we will find a feasible solution. By Lemma 4.12 and Lemma 4.13, once we find any feasible solution, we can obtain a set of $(1/\epsilon')^{O(1)}$ estimates at least one of which, say $\Sigma_{i_1}, \ldots, \Sigma_{i_m}$ satisfies

$$\|\Sigma_i - \Sigma_{i_j}\|_2^2 \leq \epsilon^{O(1)}$$

for all $i$. Note we can assume that our covariance estimates are exactly correct because we can pretend that the true mixture is actually $N(\mu_i, \Sigma_{i_1}), \ldots, N(\mu_i, \Sigma_{i_m})$ and our estimates for the Hermite polynomials of this mixture will be off by at most $O_k(1)e^{O(1)}$.

Thus, making this assumption will only affect the dependence on $\epsilon'$ that we get at the end. From now on we can write $\mathcal{E}_i$ to denote the true covariances and treat these as known quantities.

To solve for the means, we set up essentially the same system as in Section 4.2 except we no longer have the variables $v_1, \ldots, v_k$ and no longer have the $\Sigma_i$. These will instead be replaced by real values from $\Sigma_i$. We can repeat the same arguments from Section 4.3.2 to prove that once we find a feasible solution, we can recover the span of the $\mu_i$.

### 4.3.4 All Pairs of Parameters are Equal or Separated

In the case where some pairs of parameters may be equal (but pairs $(\mu_i, \Sigma_i)$ and $(\mu_j, \Sigma_j)$ cannot be too close), we can repeat essentially the same arguments from the previous section but with minor adjustments in the number of times we are applying each differential operator.

We can assume that our guesses for the coefficients $A_i, B_i$ satisfy the correct equality pattern in the sense that $A_i = A_j$ if and only if $\mu_i = \mu_j$ and otherwise $\|A_i - A_j\| \geq c/2$ and similar for the parameters $B_i$. This is because there are only $O_k(1)$ different equality patterns.

Now without loss of generality let $\{\Sigma_1, \ldots, \Sigma_J\}$ $(j < k)$ be the set of covariance matrices that are equal to $\Sigma_i$. The key differential operators to consider are

$$\overline{D}_k^{2^{j-1} - 1} \cdots \overline{D}_k^{2^{j-1}} \overline{D}_{k-1}^{2^{j-2}} \cdots \overline{D}_1^{2^{j-2}} \overline{D}_{k-1}^{2^{j-1}} \cdots \overline{D}_1^{2^{j-1}}$$

We can prove analogs of Claim 4.7 and Claim 4.8 involving the above differential operators. We can then proceed with the remainder of the proof in the same way as Section 4.3.2 and Section 4.3.3 except with appropriate modifications to the number of times we apply each differential operator.

5 ROBUST HERMITE MOMENT ESTIMATION

In Section 4, we showed how to learn the parameters of a mixture of Gaussians $M$ with components that are not too far apart when we are given estimates for the Hermite polynomials. It remains to show how to estimate the Hermite polynomials from an $\epsilon$-corrupted sample. Putting the results together, we will get a robust learning algorithm in the case when the components are not too far apart. The algorithm for estimating the Hermite polynomials closely mirrors the work in [37] so we omit most of the details here.

While the closeness of components in Section 4 is defined in terms of parameter distance, we will need to reason about TV-distance between components in order to integrate our results into our full learning algorithm. We begin with a definition.

**Definition 5.1.** We say a mixture of Gaussians $w_1 G_1 + \cdots + w_k G_k$ is $\delta$-well-conditioned if

1. Let $G$ be the graph on $[k]$ obtained by connecting two nodes $i, j$ if $d_{TV}(G_i, G_j) \leq 1 - \delta$. Then $G$ is connected
2. $d_{TV}(G_i, G_j) \geq \delta$ for all $i \neq j$
3. $w_{\min} \geq \delta$

The main theorem of this section is stated below.

**Theorem 5.2.** There is a function $f(k) > 0$ depending only on $k$ such that given an $\epsilon$-corrupted sample from a $\delta$-well-conditioned
mixture of Gaussians

\[ M = w_1 N(\mu_1, \Sigma_1) + \cdots + w_k N(\mu_k, \Sigma_k) \]

where \( \delta \geq e^{f(k)} \), there is a polynomial time algorithm that outputs a set of \((1/\epsilon)^{O(k)}\) candidate mixtures \(\{w_1 N(\mu_1, \Sigma_1) + \cdots + w_k N(\mu_k, \Sigma_k)\}\) with high probability, at least one of them satisfies that for all \(i\):

\[ |w_i - \tilde{w}_i| + d_{TV}(N(\mu_i, \Sigma_i), N(\tilde{\mu}_i, \tilde{\Sigma}_i)) \leq \text{poly}(\epsilon) \]

Proof. We combine Theorem 4.1 with techniques for estimating the Hermite polynomials from [37].

6 ROUGH CLUSTERING

As mentioned earlier in the proof overview, the first step in our full algorithm will be to cluster the points. We present our clustering algorithm in this section.

The main theorem that we aim to prove in this section is that if the true mixture can be well-clustered into submixtures, then we can recover this clustering with constant-accuracy. First, we need a definition.

Definition 6.1. We say that two Gaussians \(N(\mu, \Sigma)\) and \(N(\mu', \Sigma')\) are \(C\)-close if all of the following conditions hold

1. (mean condition) For all unit vectors \(\mathbf{v} \in \mathbb{R}^d\), we have \(\langle \mathbf{v} \cdot \mu - \mathbf{v} \cdot \mu' \rangle^2 \leq C\mathbf{v}^T (\Sigma + \Sigma') \mathbf{v} \)
2. (variance condition) For all unit vectors \(\mathbf{v} \in \mathbb{R}^d\), we have \(\max(\sigma^T \Sigma \sigma, \sigma'^T \Sigma' \sigma) \leq C \min(\sigma^T \Sigma \sigma, \sigma'^T \Sigma' \sigma) \)
3. (covariance condition) Finally, we have \(\left\| I - \Sigma'^{-1/2} \Sigma^{-1/2} \right\|_2 \leq C \)

Theorem 6.2. Let \(k, D, \gamma\) be parameters. Assume we are given \(\epsilon\)-corrupted samples from a mixture of Gaussians \(w_1 G_1 + \cdots + w_k G_k\) where the mixing weights \(w_i\) are all rational numbers with denominator bounded by a constant \(A\). Let \(A_1, \ldots, A_t\) be a partition of the components such that

1. For any \(j_1, j_2\) in the same piece of the partition, \(G_{j_1}, G_{j_2}\) are \(D\)-close
2. For any \(j_1, j_2\) in different pieces of the partition, \(G_{j_1}, G_{j_2}\) are not \(D'\)-close

where \(D' = F(k, A, D, \gamma)\) for some sufficiently large function \(F\). Assume that \(t \geq f(k, A, D, \gamma)\) and \(\eta, \delta < f(k, A, D, \gamma)\) for some sufficiently small function \(f\). Then with probability at least \(1 - \gamma\), if \(X_1, \ldots, X_n\) is a \(\epsilon\)-corrupted sample from the mixture \(w_1 G_1 + \cdots + w_k G_k\) with \(n \geq \text{poly}(1/\epsilon, 1/\eta, 1/\delta, 1/D, 1)\), then one of the clusterings returned by Rough Clustering gives a \(\gamma\)-corrupted sample of each of the submixtures given by \(A_1, \ldots, A_t\).

Remark. Note that the last statement is well defined because the assumption about the partition essentially implies that all pairs of components in different submixtures are separated so \(\gamma\)-corrupted sample simply means correctly recovering a \(1 - \gamma\)-fraction of the original points that were drawn from the corresponding submixture.

Since the proof of Theorem 6.2 is very similar to the work in [20], we omit the details here.

7 PUTTING EVERYTHING TOGETHER

We can now combine our clustering results and our results for learning mixtures of Gaussians that are not too separated to get a learning algorithm in the fully general case. Our main theorem is stated below.

Theorem 7.1. Let \(k, A, B > 0\) be constants. There is a sufficiently large function \(G\) and a sufficiently small function \(g\) depending only on \(k, A, B\) (with \(G(k, A, B), g(k, A, B) > 0\)) such that given an \(\epsilon\)-corrupted sample \(X_1, \ldots, X_n\) from a mixture of Gaussians \(M = w_1 G_1 + \cdots + w_k G_k \in \mathbb{R}^d\) where the \(G_i\) have variance at least \(\text{poly}(\epsilon/d)\) and at most \(\text{poly}(\epsilon/d)\) in all directions and

- The \(w_i\) are all rational with denominator at most \(A\)
- \(d_{TV}(G_i, G_j) \geq b\)

and \(n \geq (d/\epsilon)^{G(k, A, B)}\), then there is an algorithm that runs in time \(\text{poly}(n)\) and with 0.99 probability outputs a mixture \(\tilde{M} = \tilde{w}_1 G_1 + \cdots + \tilde{w}_k G_k\) such that \(d_{TV}(\tilde{M}, M) \leq \epsilon^9(k, A, B)\).

7.1 Distance between Gaussians

We will need a few preliminary results. The following two lemmas give a few important properties for Gaussians that are not too far separated in TV distance.

Lemma 7.2. Let \(\lambda \) be a constant. Let \(A, B, C\) be Gaussian distributions. Assume that \(d_{TV}(A, B) \leq 1 - \lambda\). If \(d_{TV}(A, C) \geq 1 - \epsilon\) and \(\epsilon\) is sufficiently small then

\[ d_{TV}(B, C) \geq 1 - \text{poly} (\epsilon) \]

(where the RHS may depend on \(\lambda\)).

Lemma 7.3 (Restated from [20]). Let \(A\) and \(B\) be two Gaussians with \(d_{TV}(A, B) \leq 1 - \lambda\) where \(\lambda\) is a constant. If \(D \in \{A, B\}\) then

\[ P_{X \sim D} \left| \frac{A(x)}{B(x)} - 1 \right| \leq 1 - \epsilon \geq 1 - \text{poly} (\epsilon) \]

7.2 Full Algorithm

We are now ready to prove Theorem 7.1. We begin by describing the algorithm. Our full algorithm consists of several phases.

1. Cluster with constant accuracy into constant-separated submixtures
2. Learn parameters of submixtures to constant accuracy
3. Reccluster all points and form new poly(\(\epsilon\))-separated submixtures
4. Learn parameters of submixtures to poly(\(\epsilon\)) accuracy

The algorithm LEARN PARAMETERS (WELL-CONDITIONED) for learning the parameters of a well-conditioned mixture of Gaussians (see Theorem 5.2) is summarized in Algorithm 1.

Algorithm 1 LEARN PARAMETERS (WELL-CONDITIONED)

Input: \(\epsilon\)-corrupted sample \(X_1, \ldots, X_n\) from \(\delta\)-well-conditioned mixture of Gaussians \(M = w_1 G_1 + \cdots + w_k G_k\)

Estimate Hermite polynomials of \(M\)

Solve for parameters using SOS (see Section 4)

Our full algorithm is described in the next block Algorithm 2.
Algorithm 2 Full Algorithm

Input: \( e \)-corrupted sample \( X_1, \ldots, X_n \) from mixture of Gaussians \( M = w_1G_1 + \cdots + w_kG_k \)
Run ROUGH CLUSTERING Algorithm to split sample into submixtures where all pairs are \( D \)-close for constant \( D \)
for each candidate clustering do
  Run LEARN PARAMETERS (well-conditioned) for each sub-mixture
  Output candidate components
for each set of candidate components \( \widetilde{G}_1, \ldots, \widetilde{G}_k \) do
  Assign samples to components according to maximum likelihood to form sets of samples \( \{\widetilde{S}_1, \ldots, \widetilde{S}_k\} \)
for all partitions of \([k]\) into sets \( R_1, \ldots, R_l \) do
  Run LEARN PARAMETERS (well-conditioned) on each of \( \bigcup_{i \in R_j} S_i \) for all \( j \in [l] \)
Output candidate components
Hypothesis test over all candidate components to find a mixture \( \widetilde{M} \) that is \( \text{poly}(e) \)-close to \( M \)

7.3 Analysis of Full Algorithm

The first step will be to show that among the first set of candidate components that we output, there are some that are within constant distance (say \( < c(k) \) for some sufficiently small function \( c \)) of the true components.

Lemma 7.4. Let \( k, A, b > 0 \) be constants and \( \theta \) be a desired accuracy. There is a sufficiently large function \( G \) and a sufficiently small function \( g \) depending only on \( k, A, b, \theta \) such that given an \( e \)-corrupted sample \( X_1, \ldots, X_n \) from a mixture of Gaussians \( M = w_1G_1 + \cdots + w_kG_k \in \mathbb{R}^d \) where

- The \( w_i \) are all rational with denominator at most \( A \)
- \( d_{TV}(G_i, G_j) \geq b \)
- \( \epsilon < g(k, A, b, \theta) \)
- \( n \geq (d/e)^{G(k,A,b,\theta)} \)

then there is an algorithm that runs in time \( \text{poly}(n) \) and with 0.999 probability outputs a set of \( (1/\theta)^{G(k,A,b,\theta)} \) candidate mixtures at least one of which satisfies

\[
\max \left( d_{TV}(G_1, G_1), \ldots, d_{TV}(G_k, G_k) \right) \leq \theta \\
\max \left( |\tilde{w}_1 - w_1|, \ldots, |\tilde{w}_k - w_k| \right) \leq \theta
\]

Proof. We combine Theorem 5.2 and Theorem 6.2. The details are omitted due to space constraints.

Our next step is to show that if our algorithm starts with component estimates that are accurate within some constant and guesses a good set of clusters, then the resulting submixtures (after assigning according to maximum likelihood) are equivalent to \( \text{poly}(e) \)-corrupted samples from the corresponding submixtures. First, we need a simple preliminary claim which implies that a good set of clusters exists.

Claim 7.5. Let \( M = w_1G_1 + \cdots + w_kG_k \) be a mixture of Gaussians. For any constant \( \epsilon > 0 \) and parameter \( \epsilon \), there exists a function \( f(c, k) \) such that there exists a partition (possibly trivial) of \([k]\) into sets \( R_1, \ldots, R_l \) such that

- If we draw edges between all \( i, j \) such that \( d_{TV}(G_i, G_j) \leq 1 - \epsilon^k \) then each piece of the partition is connected
- For any \( i, j \) in different pieces of the partition \( d_{TV}(G_i, G_j) \geq 1 - \epsilon^k \)

and \( f(c, k) < k \). The details are omitted due to space constraints.

Proof. Omitted due to space constraints.

Now we can prove our lemma about obtaining a \( \text{poly}(e) \)-accurate clustering into submixtures when given constant-accuracy estimates for the components.

Lemma 7.6. Let \( M = w_1G_1 + \cdots + w_kG_k \in \mathbb{R}^d \) be a mixture of Gaussians where

- The \( w_i \) are all rational with denominator at most \( A \)
- \( d_{TV}(G_i, G_j) \geq b \)

There exists a sufficiently small function \( g(k, A, b) > 0 \) depending only on \( k, A, b \) such that the following holds. Let \( X_1, \ldots, X_n \) be an \( e \)-corrupted sample from \( M \) where \( e < g(k, A, b) \) and \( n = \text{poly}(d/e) \)

then each piece of the partition is connected with probability \( 0.999 \).

Proof. First, we upper bound the expected number of uncorrupted points that are mis-classified by relying on Lemma 7.3. We then use a VC dimension and uniform convergence bound to argue that for any \( \tilde{G}_1, \ldots, \tilde{G}_k \) that we are given, the number of points that we mis-classify is close to the expectation. The details are omitted due to space constraints.

Combining Lemma 7.4, Claim 7.5, Lemma 7.6 and Theorem 5.2, we can show that at least one of the sets of candidate parameters that our algorithm outputs is close to the true parameters.

Lemma 7.7. Let \( k, A, b > 0 \) be constants. There is a sufficiently large function \( G \) and a sufficiently small function \( g \) depending only on \( k, A, b \) such that given an \( e \)-corrupted sample \( X_1, \ldots, X_n \) from a mixture of Gaussians \( M = w_1G_1 + \cdots + w_kG_k \in \mathbb{R}^d \) where

- The \( w_i \) are all rational with denominator at most \( A \)
- \( d_{TV}(G_i, G_j) \geq b \)

and \( n \geq (d/e)^{G(k,A,b)} \), with 0.999 probability, among the set of candidates output by Full Algorithm, there is some

\( \{\tilde{w}_1, \tilde{G}_1, \ldots, \tilde{w}_k, \tilde{G}_k\} \)

such that for all \( i \) we have

\( |w_i - \tilde{w}_i| + d_{TV}(G_i, \tilde{G}_i) \leq \text{poly}(e) \)
Proof. This follows from combining Lemma 7.4, Claim 7.5, Lemma 7.6 and finally applying Theorem 5.2. Note we can choose $c$ in Claim 7.5 so that when combined with Lemma 7.6, the resulting accuracy that we get on each submixture is high enough that we can then apply Theorem 5.2 (we can treat the subsample corresponding to each submixture as a poly$(\epsilon')$-corrupted sample). We apply Lemma 7.6 with $\epsilon' = \epsilon^k$ where $k$ is obtained from Claim 7.5.

We have shown that our algorithm recovers a list of candidate mixtures, at least one of which is close to the true mixture. The last result that we need is a hypothesis testing routine. This is similar to the hypothesis testing result in [37]. However, there is a subtle difference that the samples we use to hypothesis test may not be independent of the hypotheses. First we need a definition.

Definition 7.8. Let $\mathcal{F}$ be a family of distributions on some domain $X$. Let $\mathcal{H}_{F,a}$ be the set of functions of the form $f_{M_1, M_2, \ldots, M_a}$ where $M_1, M_2, \ldots, M_a \in \mathcal{F}$ and

$$f_{M_1, M_2, \ldots, M_a}(x) = \begin{cases} 1 & \text{if } M_1(x) \geq M_2(x), \ldots, M_a(x) \\ 0 & \text{otherwise} \end{cases}$$

where $M_i(x)$ denotes the pdf of the corresponding distribution at $x$.

Lemma 7.9. Let $\mathcal{F}$ be a family of distributions on some domain $X$ with explicitly computable density functions that can be efficiently sampled from. Let $V$ be the VC dimension of $\mathcal{H}_{F,2}$. Let $D$ be an unknown distribution in $\mathcal{F}$. Let $n \geq \text{poly}(m, \epsilon, V)$ be an $\epsilon$-corrupted sample from $D$ with $n \geq \text{poly}(m, \epsilon, V)$ for some sufficiently large polynomial. Let $H_1, \ldots, H_m$ be distributions in $\mathcal{F}$ given to us by an adversary with the promise that $\min(d_{TV}(D, H_i)) \leq \epsilon$. Then there exists an algorithm that runs in time poly$(n, \epsilon)$ and outputs an $i$ with $1 \leq i \leq m$ such that with 0.999 probability

$$d_{TV}(D, H_i) \leq O(\epsilon).$$

Proof. The proof is very similar to the proof in [37] except we will need to use a VC dimension and uniform convergence bound to obtain a bound over all possible hypothesis distributions given to us by the adversary. The details are omitted due to space constraints.

Proof of Theorem 7.1. Combining Lemma 7.7 and Lemma 7.9, we immediately get the desired bound (the necessary VC dimension bound to apply Lemma 7.9 follows from Theorem 8.14 in [2]).

8 Identifiability

Theorem 7.1 implies that we can learn a mixture that is close to the true mixture in TV distance. In order to prove that we recover the individual components, it suffices to prove identifiability. We have the following theorem.

Theorem 8.1. Let $M = w_1 G_1 + \cdots + w_k G_k$ and $M' = w_1' G_1' + \cdots + w_k' G_k'$ be mixtures of Gaussians such that $TV(M, M') \leq \epsilon$ and the $G_i, G_i'$ have variance at least poly$(\epsilon/d)$ and at most poly$(d/\epsilon)$ in all directions. Further assume,

- $d_{TV}(G_i, G_i') \geq b$,
- $d_{TV}(G_i', G_i') \geq b$ for all $i \neq j$,
- $w_i, w_i' \geq \omega_{\text{min}}$

where $b, \omega_{\text{min}} \geq \epsilon^k$ where $k = \max(k_1, k_2)$ and $f(k) > 0$ is sufficiently small function depending only on $k$. Then $k_1 = k_2$ and there exists a permutation $\pi$ such that

$$|w_i - w'_{\pi(i)}| + d_{TV}(G_i, G'_{\pi(i)}) \leq \alpha(\epsilon).$$

Theorem 8.1 can be proven by using the same techniques as the analysis of our algorithm in previous sections. Combining Theorem 7.1 and Theorem 8.1, we get our full theorem.

Theorem 8.2. Let $k, a, b > 0$ be constants. There is a sufficiently large function $G$ and a sufficiently small function $g$ depending only on $k, a, b$ with $G(k, a, b), g(k, a, b) > 0$ such that given an $\epsilon$-corrupted sample $X_1, \ldots, X_n$ from a mixture of Gaussians $M = w_1 G_1 + \cdots + w_k G_k \in \mathbb{R}^d$ where the $G_i$ have variance at least poly$(\epsilon/d)$ and at most poly$(d/\epsilon)$ in all directions and

- The $w_i$ are all rational with denominator at most $A$,
- $d_{TV}(G_i, G_j) \geq b$

and $n \geq (d/\epsilon)G(k, a, b)$, then there is an algorithm that runs in time poly$(n)$ and with 0.99 probability outputs a set of components $G_1, \ldots, G_k$ and mixing weights $w_1, \ldots, w_k$ such that there exists a permutation $\pi$ on $[k]$ such that for all $i$,

$$|w_i - w_{\pi(i)}| + d_{TV}(G_i, G_{\pi(i)}) \leq \alpha(\epsilon)^{G(k, a, b)}.$$

Acknowledgments

This work was supported in part by a Microsoft Trustworthy AI Grant, NSF CAREER Award CCF-1453261, NSF Large CCF1565235, a David and Lucile Packard Fellowship and an ONR Young Investigator Award.

References

[1] Dimitris Achlioptas and Frank McSherry. 2005. On spectral learning of mixtures of distributions. In Learning Theory. Springer, 458–469.
[2] Martin Anthony and Peter L Bartlett. 2009. Neural network learning: Theoretical foundations. Cambridge University Press.
[3] Sanjeev Arora and Ravi Kannan. 2001. Learning mixtures of arbitrary gaussians. In Proceedings of the thirty-third annual ACM symposium on Theory of computing. 257–267.
[4] Animesh Bakshi, Ilias Diakonikolas, He Jia, Daniel Michael Kane, Pravesh Kothari, and Santosh S Vempala. 2020. Robustly Learning Mixtures of $k$ Arbitrary Gaussians. arXiv:2012.02119 [cs.DS].
[5] Animesh Bakshi and Pravesh Kothari. 2020. Outlier-Robust Clustering of Non-Spherical Mixtures. arXiv preprint arXiv:2005.02970 (2020).
[6] Animesh Bakshi and Adarsh Prasad. 2020. Robust Linear Regression: Optimal Rates in Polynomial Time. arXiv preprint arXiv:2007.01394 (2020).
[7] Swaranan Balakrishnan, Simon S Du, Jerry Li, and Aarti Singh. 2017. Computationally efficient robust sparse estimation in high dimensions. In Conference on Learning Theory. 169–212.
[8] Boaz Barak. [n.d.]. Proofs, beliefs, and algorithms through the lens of sum-of-squares. (in d\).
[9] Boaz Barak, Jonathan A Kelner, and David Steurer. 2014. Rounding sum-of-squares relaxations. In Proceedings of the forty-sixth annual ACM symposium on Theory of computing. 31–40.
[10] Boaz Barak, Jonathan A Kelner, and David Steurer. 2015. Dictionary learning and tensor decomposition via the sum-of-squares method. In Proceedings of the forty-seventh annual ACM symposium on Theory of computing. 143–153.
[11] Boaz Barak and Ankur Moitra. 2016. Noisy tensor completion via the sum-of-squares hierarchy. In Conference on Learning Theory. 417–445.
[12] Mikhail Belkin and Kaushik Sinha. 2010. Polynomial learning of distribution families. In Conference on Learning Theory. 143–151.
[13] Thorsten Bernholt. 2006. Robust estimators are hard to compute. Technical Report. Technical Report.
[14] 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.
[15] S Charles Brubaker and Santosh S Vempala. 2008. Isotropic PCA and affine-invariant clustering. In Building Bridges Springer, 241–281.

[16] Moses Charikar, Jacob Steinhardt, and Gregory Valiant. 2017. Learning from untrusted data. In Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing. ACM, 47–60.

[17] Sitan Chen, Frederic Koehler, Ankur Moitra, and Morris Yau. 2020. Online and Distribution-Free Robustness: Regression and Contextual Bandits with Huber Contamination. arXiv preprint arXiv:2010.04157 (2020).

[18] Sanjoy Dasgupta. 1999. Learning mixtures of gaussians. In Foundations of Computer Science, 1999. 40th Annual Symposium on. IEEE, 634–644.

[19] Sanjoy Dasgupta and Leonard Schulman. 2013. A two-round variant of EM for gaussian mixtures. arXiv preprint arXiv:1301.8350 (2013).

[20] Ilias Diakonikolas, Samuel B Hopkins, Daniel Kane, and Sushrut Karmalkar. 2020. Robustly Learning any Clusterable Mixture of Gaussians. arXiv preprint arXiv:2005.06417 (2020).

[21] Ilias Diakonikolas, Gautam Kamath, Daniel Kane, Jerry Li, Ankur Moitra, and Alistair Stewart. 2019. Robust Estimators in High-Dimensions Without the Computational Intractability. SIAM J. Comput. 48, 2 (2019), 742–864.

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

[23] Ilias Diakonikolas, Gautam Kamath, Daniel M Kane, Jerry Li, Ankur Moitra, and Alistair Stewart. 2017. Being robust (in high dimensions) can be practical. In Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR. org, 999–1008.

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

[25] 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. ACM, 761–770.

[26] Frank B Hampel, Elvezio M Ronchetti, Peter J Rousseeuw, and Werner A Stahel. 2011. Robust statistics: the approach based on influence functions. Vol. 196. John Wiley & Sons.

[27] Morris Haridt and Ankur Moitra. 2013. Algorithms and hardness for robust subspace recovery. In Conference on Learning Theory. 354–375.

[28] Samuel B Hopkins, Pravesh K Kothari, Aaron Potechin, Prasad Raghavendra, Tselil Schramm, and David Steurer. 2017. The power of sum-of-squares for detecting hidden structures. In 2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS). IEEE, 726–731.

[29] 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. ACM, 1021–1034.

[30] 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.

[31] Samuel B Hopkins, Jonathan Shi, and David Steurer. 2015. Tensor principal component analysis via sum-of-squares proofs. In Conference on Learning Theory.