Beyond the Low-Degree Algorithm: Mixtures of Subcubes and Their Applications

Sitan Chen*
Massachusetts Institute of Technology
United States of America
sitanc@mit.edu

Ankur Moitra†
Massachusetts Institute of Technology
United States of America
moitra@mit.edu

ABSTRACT

We introduce the problem of learning mixtures of k subcubes over $\{0,1\}^n$, which contains many classic learning theory problems as a special case (and is itself a special case of others). We give a surprising $n^{O(\log k)}$-time learning algorithm based on higher-order multilinear moments. It is not possible to learn the parameters because the same distribution can be represented by quite different models. Instead, we develop a framework for reasoning about how multilinear moments can pinpoint essential features of the mixture, like the number of components.

We also give applications of our algorithm to learning decision trees with stochastic transitions (which also capture interesting scenarios where the transitions are deterministic but there are latent variables). Using our algorithm for learning mixtures of subcubes, we can approximate the Bayes optimal classifier within additive error ϵ on k-leaf decision trees with at most s stochastic transitions on any root-to-leaf path in $n^{O(s+\log k)} \cdot \mathrm{poly}(1/\epsilon)$ time. In this stochastic setting, the classic $n^{O(\log k)} \cdot \mathrm{poly}(1/\epsilon)$-time algorithms of Rivest, Blum, and Ehrenfreucht-Haussler for learning decision trees with zero stochastic transitions break down because they are fundamentally Occam algorithms. The low-degree algorithm of Linial-Mansour-Nisan is able to get a constant factor approximation to the optimal error (again within an additive ϵ) and runs in time $n^{O(s+\log(k/\epsilon))}$. The quasipolynomial dependence on $1/\epsilon$ is inherent to the low-degree approach because the degree needs to grow as the target accuracy decreases, which is undesirable when ϵ is small.

In contrast, as we will show, mixtures of k subcubes are *uniquely* determined by their $2\log k$ order moments and hence provide a useful abstraction for simultaneously achieving the polynomial dependence on $1/\epsilon$ of the classic Occam algorithms for decision trees and the flexibility of the low-degree algorithm in being able to accommodate stochastic transitions. Using our multilinear moment techniques, we also give the first improved upper and lower bounds since the work of Feldman-O'Donnell-Servedio for the related but harder problem of learning mixtures of binary product distributions.

*This work was supported in part by an MIT Presidential Fellowship, a Paul and Daisy Soros Fellowship, and NSF CAREER Award CCF-1453261 and NSF Large CCF-1565235.
†This work was supported in part by NSF CAREER Award CCF-1453261, NSF Large CCF-1565235, a David and Lucile Packard Fellowship, and an Alfred P. Sloan Fellowship.

STOC '19, June 23–26, 2019, Phoenix, AZ, USA

ACM ISBN 978-1-4503-6705-9/19/06…$15.00
https://doi.org/10.1145/3313276.3316375

CCS CONCEPTS

• **Theory of computation → Unsupervised learning and clustering**; *Boolean function learning*; • **Mathematics of computing** → *Density estimation.*

KEYWORDS

Mixture models, method of moments, statistical query learning, Fourier approximation

ACM Reference Format:
Sitan Chen and Ankur Moitra. 2019. Beyond the Low-Degree Algorithm: Mixtures of Subcubes and Their Applications. In *Proceedings of the 51st Annual ACM SIGACT Symposium on the Theory of Computing (STOC '19), June 23–26, 2019, Phoenix, AZ, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3313276.3316375

1 INTRODUCTION

1.1 Background

In this paper, we introduce and study the following natural problem: A *mixture of subcubes* is a distribution on the Boolean hypercube where each sample is drawn as follows:

(1) There are k mixing weights $\pi^1, \pi^2, \cdots, \pi^k$ corresponding to k centers $\mu^1, \mu^2, \cdots, \mu^k \in \{0, 1/2, 1\}^n$.
(2) We choose a center proportional to its mixing weight and sample a point uniformly at random from its corresponding subcube. More precisely, if we choose the i^{th} center, each coordinate is independent and the j^{th} has expectation μ_i^j.

Our goal is to give efficient algorithms for estimating the distribution in the PAC-style model of Kearns et al. [16]. It is not always possible to learn the parameters because two mixtures of subcubes[1] can give rise to identical distributions. Instead, the goal is to output a distribution close to the true one in total variation distance.

The problem of learning mixtures of subcubes contains various classic problems in computational learning theory as a special case, and is itself a special case of others. For example, for any k-leaf decision tree, the uniform distribution on assignments that satisfy it is a mixture of k subcubes. Likewise, for any function that depends on just j variables (a j-junta), the uniform distribution on assignments that satisfy it is a mixture of 2^j-subcubes. And when we allow the centers μ^i to instead be in the set $[0,1]^n$ it becomes the problem of learning mixtures of binary product distributions.

[1] Even with different numbers of components.

These problems all have a long history of study. Ehrenfeucht and Haussler [8] gave an $n^{O(\log k)}$ time algorithm for learning k-leaf decision trees. Blum [3] showed that k-leaf decision trees can be represented as $\log k$-width decision lists and Rivest [21] gave an algorithm for learning ℓ-width decision lists in time $n^{O(\ell)}$. Mossel, O'Donnell and Servedio [20] gave an $n^{j\frac{\omega}{\omega+1}}$ time algorithm for learning j-juntas where ω is the matrix multiplication constant, and Valiant [23] improved this to $n^{j\frac{\omega}{4}}$ time. Freund and Mansour [11] gave the first algorithm for learning mixtures of two product distributions. Feldman, O'Donnell and Servedio [9] gave an $n^{O(k^3)}$ time algorithm for learning mixtures of k product distributions.

What makes the problem of learning mixtures of subcubes an interesting compromise between expressive power and structure is that it admits surprisingly efficient learning algorithms. The main result in our paper is an $n^{O(\log k)}$ time algorithm for learning mixtures of subcubes. We also give applications of our algorithm to learning k-leaf decision trees with at most s stochastic transitions on any root-to-leaf path (which also capture interesting scenarios where the transitions are deterministic but there are latent variables). Using our algorithm for learning mixtures of subcubes, we can approximate the error of the Bayes optimal classifier within an additive ϵ in $n^{O(s+\log k)} \cdot \mathrm{poly}(1/\epsilon)$ time with an inverse polynomial dependence on the accuracy parameter ϵ. The classic algorithms of [3, 8, 21] for learning decision trees with zero stochastic transitions achieve this runtime, but because they are Occam algorithms, they break down in the presence of stochastic transitions. Alternatively, the low-degree algorithm [18] is able to get a constant factor approximation to the optimal error (again within an additive ϵ), while running in time $n^{O(s+\log(k/\epsilon))}$. The quasipolynomial dependence on $1/\epsilon$ is inherent to the low-degree approach because the degree needs to grow as the target accuracy decreases, which is undesirable when ϵ is small as a function of k.

In contrast, we show that mixtures of k subcubes are uniquely identified by their $2\log k$ order moments. Ultimately our algorithm for learning mixtures of subcubes will allow us to simultaneously match the polynomial dependence on $1/\epsilon$ of Occam algorithms and achieve the flexibility of the low-degree algorithm in being able to accommodate stochastic transitions. We emphasize that proving identifiability from $2\log k$ order moments is only a first step in a much more technical argument: There are many subtleties about how we can algorithmically exploit the structure of these moments to solve our learning problem.

1.2 Our Results and Techniques

Our main result is an $n^{O(\log k)}\mathrm{poly}(1/\epsilon)$ time algorithm for learning mixtures of subcubes.

Theorem 1.1. *Let $\epsilon, \delta > 0$ be given and let $\mathcal{D}$ be a mixture of k subcubes. There is an algorithm that given samples from $\mathcal{D}$ runs in time $O_k(n^{O(\log k)}(1/\epsilon)^{O(1)} \log 1/\delta)$ and outputs a mixture $\mathcal{D}'$ of $f(k)$ subcubes that satisfies $d_{TV}(\mathcal{D}, \mathcal{D}') \le \epsilon$ with probability at least $1-\delta$. Moreover the sample complexity is $O_k((\log n/\epsilon)^{O(1)} \log 1/\delta)$.*[2]

[2]Throughout, the hidden constant depending on k will be $O(k^{k^3})$, which we have made no attempt to optimize.

The starting point for our algorithm is the following simple but powerful identifiability result:

Lemma 1.1 (Informal). *A mixture of k subcubes is uniquely determined by its $2\log k$ order moments.*

In contrast, for many sorts of mixture models with k components, typically one needs $\Theta(k)$ moments to establish identifiability [19] and this translates to algorithms with running time at least $n^{\Omega(k)}$ and sometimes even much larger than that. In part, this is because the notion of identifiability we are aiming for needs to be weaker and as a result is more subtle. We cannot hope to learn the subcubes and their mixing weights because there are mixtures of subcubes that can be represented in many different ways, sometimes with the same number of subcubes. But as distributions, two mixtures of subcubes are the same if they match on their first $2\log k$ moments. It turns out that proving this is equivalent to the following basic problem in linear algebra:

Question 1.1. *Given a matrix $M \in \{0, 1/2, 1\}^{n\times k}$, what is the minimum d for which the set of all entrywise products of at most d rows of M spans the set of all entrywise products of rows of M?*

We show that d can be at most $2\log k$, which is easily shown to be tight up to constant factors. We will return to a variant of this question later when we discuss why learning mixtures of product distributions requires much higher-order moments.

Unsurprisingly, our algorithm for learning mixtures of subcubes is based on the method of moments. But there is an essential subtlety. For any distribution on the hypercube, $x_i^2 = x_i$. From a technical standpoint, this means that when we compute moments, there is never any reason to take a power of x_i larger than one. We call these *multilinear moments*, and characterizing the way that the multilinear moments determine the distribution (but cannot determine its parameters) is the central challenge. Note that multilinearity makes our problem quite different from typical settings where tensor decompositions can be applied.

Now collect the centers $\mu^1, \mu^2, \cdots, \mu^k$ into a $n\times k$ size matrix that we call the *marginals matrix* and denote by $\mathbf{m}$. The key step in our algorithm is constructing a basis for the entrywise products of rows from this matrix. However we cannot afford to simply brute-force search for this basis among all sets of at most k entrywise products of up to $2\log k$ rows of $\mathbf{m}$ because the resulting algorithm would run in time $n^{O(k\log k)}$. Instead we construct a basis incrementally.

The first challenge that we need to overcome is that we cannot directly observe the entrywise product of a set of rows of the marginals matrix. But we can observe its weighted inner-product with various other vectors. More precisely, if u, v are respectively the entrywise products of subsets S and T of rows of some marginals matrix $\mathbf{m}$ that realizes the distribution and π is the associated vector of mixing weights, then the relation

$$\sum_{i=1}^{k} \pi^i u_i v_i = \mathbb{E}\left[\prod_{i\in S\cup T} x_i\right]$$

holds if S and T are disjoint. When S and T intersect, this relation is no longer true because in order to express the left hand side in terms of the x_i's we would need to take some powers to be larger than one, which no longer correspond to multilinear moments that can be estimated from samples.

Now suppose we are given a collection $\mathcal{B} = \{T_1, T_2, \cdots, T_k\}$ of subsets of rows of $\mathbf{m}$ and we want to check whether the vectors $\{v_1, v_2, \cdots, v_k\}$ (where v_i is the entrywise product of the rows in T_i) are linearly independent. Set $J = \cup_i T_i$. We can define a helper matrix whose columns are indexed by the T_i's and whose rows are indexed by subsets of $[n]\backslash J$. The entry in column i, row S is $\mathbb{E}[\prod_{j \in S \cup T_i} x_j]$ and it is easy to show that if this helper matrix has full row rank then the vectors $\{v_1, v_2, \cdots, v_k\}$ are indeed linearly independent.

The second challenge is that this is an imperfect test. Even if the helper matrix is not full rank, $\{v_1, v_2, \cdots, v_k\}$ might still be linearly independent. Even worse, we can encounter situations where our current collection $\mathcal{B}$ is not yet a basis, and yet for any set we try to add, we cannot certify that the associated entrywise product of rows is outside the span of the vectors we have so far. Our algorithm is based on a win-win analysis. We show that when we get stuck in this way, it is because there is some $S \subseteq [n]$ with $|S| \leq 2 \log k$ where the order $2 \log k$ entrywise products of subets of rows from $[n]\backslash(J \cup S)$ do not span the full k-dimensional space. We show how to identify such an S by repeatedly solving systems of linear equations. Once we identify such an S it turns out that for any string $s \in \{0,1\}^{|J \cup S|}$ we can condition on $x_{J \cup S} = s$ and the resulting conditional distribution will be a mixture of strictly fewer subcubes, which we can then recurse on.

1.3 Applications

We demonstrate the power of our $n^{O(\log k)}$ time algorithm for learning mixtures of subcubes by applying it to learning decision trees with stochastic transitions. Specifically suppose we are given a sample x that is uniform on the hypercube, but instead of computing its label based on a k-leaf decision tree with deterministic transitions, some of the transitions are stochastic — they read a bit and based on its value proceed down either the left or right subtree with some unknown probabilities. Such models are popular in medicine [13] and finance [14] when features of the system are partially or completely unobserved and the transitions that depend on these features appear to an outside observer to be stochastic. Thus we can also think about decision trees with deterministic transitions but with latent variables as having stochastic transitions when we marginalize on the observed variables.

With stochastic transitions, it is no longer possible to perfectly predict the label even if you know the stochastic decision tree. This rules out many forms of learning like Occam algorithms such as [3, 8, 21] that are based on succinctly explaining a large portion of the observed samples. It turns out that by accurately estimating the distribution on positive examples — via our algorithm for learning mixtures of subcubes — it is possible to approach the Bayes optimal classifier in $n^{O(\log k)}$ time and with only a polylogarithmic number of samples:

Theorem 1.2. *Let $\epsilon, \delta > 0$ be given and let $\mathcal{D}$ be a distribution on labelled examples from a stochastic decision tree under the uniform distribution. Suppose further that the stochastic decision tree has k leaves and along any root-to-leaf path there are at most s stochastic transitions. There is an algorithm that given samples from $\mathcal{D}$ runs in time $O_{k,s}(n^{O(s+\log k)}(1/\epsilon)^{O(1)} \log 1/\delta)$ and with probability at least $1 - \delta$ outputs a classifier whose probability of error is at most $opt + \epsilon$ where opt is the error of the Bayes optimal classifier. Moreover the sample complexity is $O_{k,s}((\log n/\epsilon)^{O(1)} \log 1/\delta)$.*

Recall that the low-degree algorithm [18] is able to learn k-leaf decision trees in time $n^{O(\log(k/\epsilon))}$ by approximating them by $O(\log(k/\epsilon))$ degree polynomials. These results also generalize to stochastic settings [1]. Recently, Hazan, Klivans and Yuan [12] were able to improve the sample complexity even in the presence of adversarial noise using the low-degree Fourier approximation approach together with ideas from compressed sensing for learning low-degree, sparse Boolean functions [22]. Although our algorithm is tailored to handle stochastic rather than adversarial noise, our algorithm has a much tamer dependence on ϵ which yields much faster algorithms when ϵ is small as a function of k. Moreover we achieve a considerably stronger (and nearly optimal) error guarantee of opt+ϵ rather than $c \cdot$opt+ϵ for some constant c. Our algorithm even works in the natural variations of the problem [5, 6, 17] where it is only given positive examples.

Lastly, we remark that [5] studied a similar setting where the learner is given samples from the uniform distribution $\mathcal{D}$ over satisfying assignments of some Boolean function f and the goal is to output a distribution close to $\mathcal{D}$. Their techniques seem quite different from ours and also the low-degree algorithm. Among their results, the one most relevant to ours is the incomparable result that there is an $n^{O(\log(k/\epsilon))}$-time learning algorithm for when f is a k-term DNF formula.

1.4 More Results

As we discussed earlier, mixtures of subcubes are a special case of mixtures of binary product distributions. The best known algorithm for learning mixtures of k product distributions is due to Feldman, O'Donnell and Servedio [9] and runs in time $n^{O(k^3)}$. A natural question which a number of researchers have thought about is whether the dependence on k can be improved, perhaps to $n^{O(\log k)}$. This would match the best known statistical query (SQ) lower bound for learning mixtures of product distributions, which follows from the fact that the uniform distribution over inputs accepted by a decision tree is a mixture of product distributions and therefore from Blum et al.'s $n^{O(\log k)}$ SQ lower bound [4].

As we will show, it turns out that mixtures of product distributions require much higher-order moments even to distinguish a mixture of k product distributions from the uniform distribution on $\{0,1\}^n$. As before, this turns out to be related to a basic problem in linear algebra:

Question 1.2. *For a given k, what is the largest possible collection of vectors $v_1, v_2, \cdots, v_m \in \mathbb{R}^k$ for which (1) the entries in the entrywise product of any $t < m$ vectors sum to zero and (2) the entries in the entrywise product of all m vectors do not sum to zero?*[3]

We show a rather surprising construction that achieves $m = c\sqrt{k}$. An obvious upper bound for m is k. It is not clear what the correct answer ought to be. In any case, we show that this translates to the following negative result:

[3] In Section 4 we discuss the relationship between Questions 1.1 and 1.2.

Lemma 1.2 (Informal). *There is a family of mixtures of product distributions that are all different as distributions but which match on all $c\sqrt{k}$ order moments.*

Given a construction for Question 1.2, the idea for building this family is the same idea that goes into the $n^{\Omega(s)}$ SQ lower bound for s-sparse parity [15] and the $n^{\Omega(k)}$ SQ lower bound for density estimation of mixtures of k Gaussians [7], namely that of hiding a low-dimensional moment-matching example inside a high-dimensional product measure. We leverage Lemma 1.2 to show an SQ lower bound for learning mixtures of product distributions that holds for small values of ϵ, which is exactly the scenario we are interested in, particularly in applications to learning stochastic decision trees.

Theorem 1.3 (Informal). *Any algorithm with statistical query access to a mixture $\mathcal{D}$ of k binary product distributions to within accuracy $\Omega(n^{-\sqrt{k}/3})$ which outputs a distribution $\mathcal{D}'$ satisfying $d_{TV}(\mathcal{D},\mathcal{D}') \le \epsilon$ for $\epsilon \le k^{-c\sqrt{k}}$ must make at least $n^{c'\sqrt{k}}$ queries.*

This improves upon the previously best known SQ lower bound of $n^{\Omega(\log k)}$, although for larger values of ϵ our construction breaks down. In any case, in a natural dimension-independent range of parameters, mixtures of product distributions are substantially harder to learn using SQ algorithms than the special case of mixtures of subcubes.

Finally, we leverage the insights we developed for reasoning about higher-order multilinear moments to give improved algorithms for learning mixtures of binary product distributions:

Theorem 1.4. *Let $\epsilon,\delta > 0$ be given and let $\mathcal{D}$ be a mixture of k binary product distributions. There is an algorithm that given samples from $\mathcal{D}$ runs in time $O_k((n/\epsilon)^{O(k^2)}\log 1/\delta)$ and outputs a mixture $\mathcal{D}'$ of $f(k)$ binary product distributions that satisfies $d_{TV}(\mathcal{D},\mathcal{D}') \le \epsilon$ with probability at least $1-\delta$.*

Here we can afford to brute-force search for a basis. However a different issue arises. In the case of mixtures of subcubes, when a collection of vectors that come from entrywise products of rows are linearly independent we can also upper bound their condition number, which allows us to get a handle on the fact that we only have access to the moments of the distribution up to some sampling noise. But when the centers are allowed to take on arbitrary values in $[0,1]^n$ there is no a priori upper bound on the condition number. To handle sampling noise, instead of finding just any basis, we find a barycentric spanner.[4] We proceed via a similar win-win analysis as for mixtures of subcubes: in the case that condition number poses an issue for learning the distribution, we argue that after conditioning on the coordinates of the barycentric spanner, the distribution is *close* to a mixture of fewer product distributions. A key step in showing this is to prove the following *robust* identifiability result that may be of independent interest:

Lemma 1.3 (Informal). *Two mixtures of k product distributions are ϵ-far in statistical distance if and only if they differ on some moment of order at most $2k$ by an additive $\mathrm{poly}(n,1/\epsilon,2^k)^{-O(k)}$ amount.*

[4]Specifically, we find a barycentric spanner for just the rows of the *marginals matrix*, rather than for the set of entrywise products of rows of the marginals matrix.

In fact this is tight in the sense that $o(k)$-order moments are insufficient to distinguish between some mixtures of k product distributions (see the discussion in Section 4). Another important point is that in the case of mixtures of subcubes, exact identifiability by $O(\log k)$-order moments (Lemma 1.1) is non-obvious but, once proven, can be bootstrapped in a black-box fashion to robust identifiability using the abovementioned condition number bound. On the other hand, for mixtures of product distributions, exact identifiability by $O(k)$-order moments is straightforward, but without a condition number bound, it is much more challenging to turn this into a result about robust identifiability.

1.5 Organization

The rest of this paper is organized as follows:

- Section 2 — we set up basic definitions, notation, and facts about mixtures of product distributions and give an overview of our techniques.
- Section 3 — we describe our algorithm for learning mixtures of subcubes and give the main ingredients in the proof of Theorem 1.1.
- Section 4 — we sketch a proof of the statistical query lower bound of Theorem 1.3.
- Section 5 — we describe our algorithm for learning general mixtures of product distributions, give the main ingredients in the proof of Theorem 1.4, and conclude in Section 5.1 with a comparison of our techniques to those of [9].
- Appendix A — we make precise the connection between mixtures of subcubes and various classical learning theory problems, including stochastic decision trees, juntas, and sparse parity with noise, and prove Theorem 1.2.

2 PRELIMINARIES

2.1 Notation and Definitions

Given a matrix A, we denote by A_i^j the entry in A in row i and column j. For a set S, we denote $A|_S$ as the restriction of A to rows in S. And similarly $A|^T$ is the restriction of A to columns in T. We will let $\|A\|_{\max}$ denote the maximum absolute value of any entry in A and $\|A\|_\infty$ denote the induced L_∞ operator norm of A, that is, the maximum absolute row sum. We will also make frequent use of entrywise products of vectors and their relation to the multilinear moments of the mixture model.

Definition 2.1. The *entrywise product* $\bigodot_{j\in S} v^j$ of a collection of vectors $\{v^j\}_{j\in S}$ is the vector whose i^{th} coordinate is $\prod_{j\in S} v_i^j$. When $S=\emptyset$, $\bigodot_{j\in S} v_i$ is the all ones vector.

Given a set J, we use 2^J to denote the powerset of J. Let U_n be the uniform distribution over $\{0,1\}^n$. Also let $\mathcal{R}(J) = 2^{[n]\setminus J}$ for convenience. Let $\mathcal{D}(x)$ denote the density of $\mathcal{D}$ at x. Let 1^n be the all ones string of length n.

Definition 2.2. For $S\subseteq[n]$, the *S-moment* of $\mathcal{D}$ is $\Pr_{\mathcal{D}}[x_S = 1^{|S|}]$. We will sometimes use the shorthand $\mathbb{E}_{\mathcal{D}}[x_S]$.

There can be many choices of mixing weights and centers that yield the same mixture of product distributions $\mathcal{D}$. We will refer to any valid choice of parameters as a realization of $\mathcal{D}$.

Definition 2.3. A mixture of k product distributions $\mathcal{D}$ is a *mixture of k subcubes* if there is a realization of $\mathcal{D}$ with mixing weights $\pi^1, \pi^2, \cdots, \pi^k$ and centers $\mu^1, \mu^2, \cdots, \mu^k$ for which each center has only $\{0, 1/2, 1\}$ values.

In this paper, when referring to mixing weights, our superscript notation is only for indexing and never for powering.

There are three main matrices we will be concerned with.

Definition 2.4. The *marginals matrix* $\mathbf{m}$ is a $n \times k$ matrix obtained by concatenating the centers $\mu^1, \mu^2, \cdots, \mu^k$, for some realization. The *moment matrix* $\mathbf{M}$ is a $2^n \times k$ matrix whose rows are indexed by sets $S \subseteq [n]$ and

$$\mathbf{M}_S = \bigodot_{i \in S} \mathbf{m}_i$$

Finally the *cross-check matrix* $\mathbf{C}$ is a $2^n \times 2^n$ matrix whose rows and columns are indexed by sets $S, T \subseteq [n]$ and whose entries are in $[0,1] \cup \{?\}$ where

$$\mathbf{C}_S^T = \begin{cases} \mathbb{E}_{\mathcal{D}}[x_{S \cup T}] & \text{if } S \cap T = \emptyset \\ ? & \text{otherwise} \end{cases}$$

We say that an entry of $\mathbf{C}$ is *accessible* if it is not equal to ?.

It is important to note that $\mathbf{m}$ and $\mathbf{M}$ depend on the choice of a particular realization of $\mathcal{D}$, but that $\mathbf{C}$ does not because its entries are defined through the moments of $\mathcal{D}$. The starting point for our algorithms is the following observation about the relationship between $\mathbf{M}$ and $\mathbf{C}$:

Observation 2.1. *For any realization of $\mathcal{D}$ with mixing weights π and centers $\mu^1, \mu^2, \cdots, \mu^k$. Then*

(1) For any set $S \subseteq [n]$ we have $\mathbf{M}_S \cdot \pi = \mathbb{E}_{\mathcal{D}}[x_S]$

(2) For any pair of sets $S, T \subseteq [n]$ with $S \cap T = \emptyset$ we have

$$\mathbf{C}_S^T = \left(\mathbf{M} \cdot diag(\pi) \cdot \mathbf{M}^\top\right)_S^T$$

The idea behind our algorithms are to find a basis for the rows of $\mathbf{M}$ or failing that to find some coordinates to condition on which result in a mixture of fewer product distributions. The major complications come from the fact that we can only estimate the accessible entries of $\mathbf{C}$ from samples from our distribution. If we had access to all of them, it would be straightforward to use the above relationship between $\mathbf{M}$ and $\mathbf{C}$ to find a set of rows of $\mathbf{M}$ that span the row space.

2.2 Rank of the Moment Matrix and Conditioning

First we will show that without loss of generality we can assume that the moment matrix $\mathbf{M}$ has full column rank. If it does not, we will be able to find a new realization of $\mathcal{D}$ as a mixture of strictly fewer product distributions.

Definition 2.5. A realization of $\mathcal{D}$ is a *full rank realization* if $\mathbf{M}$ has full column rank and all the mixing weights are nonzero. Furthermore if $\text{rank}(\mathbf{M}) = k$ we will say $\mathcal{D}$ has rank k.

Lemma 2.1. *Fix a realization of $\mathcal{D}$ with mixing weights π and centers $\mu^1, \mu^2, \cdots, \mu^k$ and let $\mathbf{M}$ be the moment matrix. If* $\text{rank}(\mathbf{M}) = r < k$ *then there are new mixing weights π' such that:*

(1) π' has r nonzeros

(2) π' and $\mu^1, \mu^2, \cdots, \mu^k$ also realize $\mathcal{D}$.

Moreover the submatrix $\mathbf{M}'$ consisting of the columns of $\mathbf{M}$ with nonzero mixing weight in π' has rank r.

Proof. We will proceed by induction on r. When $r = k-1$ there is a vector $v \in \ker(\mathbf{M})$. The sum of the entries in v must be zero because the first row of $\mathbf{M}$ is the all ones vector. Now if we take the line $\pi + tv$ as we increase t, there is a first time t_0 when a coordinate becomes zero. Let $\pi' = \pi + t_0 v$. By construction, π' is nonnegative and its entries sum to one and it has at most $k-1$ nonzeros. We can continue in this fashion until the columns corresponding to the support of π' in $\mathbf{M}$ are linearly independent. Note that as we change the mixing weights, the moment matrix $\mathbf{M}$ stays the same. Also the resulting matrix $\mathbf{M}'$ that we get must have rank r because each time we update π we are adding a multiple of a vector in the kernel of $\mathbf{M}$ so the columns whose mixing weight is changing are linearly dependent. □

Thus when we fix an (unknown) realization of $\mathcal{D}$ in our analysis, we may as well assume that it is a full rank realization. This is true even if we restrict our attention to mixtures of subcubes where the above lemma shows that if $\mathbf{M}$ does not have full column rank, there is a mixture of $r < k$ subcubes that realizes $\mathcal{D}$. Next we show that mixtures of product distributions behave nicely under conditioning:

Lemma 2.2. *Fix a realization of $\mathcal{D}$ with mixing weights π and centers $\mu^1, \mu^2, \cdots, \mu^k$. Let $S \subseteq [n]$ and $s \in \{0,1\}^{|S|}$. Then $\mathcal{D}|_{x_S = s}$ can be realized as a mixture of k product distributions with mixing weights π' and centers $\mu^1|_{[n]\setminus S}, \mu^2|_{[n]\setminus S}, \cdots, \mu^k|_{[n]\setminus S}$.*

Proof. Using Bayes' rule we can write out the mixing weights π' explicitly as

$$\pi' = \frac{\pi \bigodot \left(\bigodot_{i \in S} \gamma^i\right)}{\Pr_{\mathcal{D}}[x_S = s]}$$

where we have abused notation and used $\bigodot$ as an infix operator and where $\gamma^i = \mu^i + (1 - s_i) \cdot (\vec{\mathbf{1}} - 2\mu^i)$. This follows because the map $x \mapsto x + (1-s) \cdot (1-2x)$ is the identity when $s = 1$ and $x \mapsto 1 - x$ when $s = 0$ □

We can straightforwardly combine Lemma 2.1 and Lemma 2.2 to conclude that if $\text{rank}(\mathbf{M}|_{2^{[n]\setminus S}}) = r$ then for any $s \in \{0,1\}^{|S|}$ there is a realization of $\mathcal{D}|_{x_S = s}$ as a mixture of r product distributions. Moreover if $\mathcal{D}$ was a mixture of subcubes then so too would the realization of $\mathcal{D}|_{x_S = s}$ be.

2.3 Linear Algebraic Relations between M and C

Even though not all of the entries of $\mathbf{C}$ are accessible (i.e. can be estimated from samples from $\mathcal{D}$) we can still use it to deduce linear algebraic properties among the rows of $\mathbf{M}$. All of the results in this subsection are elementary consequences of Observation 2.1.

Lemma 2.3. *Let $T_1, T_2, \cdots, T_r \subseteq [n]$ and set $J = \cup_i T_i$. If the columns*

$$\mathbf{C}^{T_1}|_{\mathcal{R}(J)}, \mathbf{C}^{T_2}|_{\mathcal{R}(J)}, \cdots, \mathbf{C}^{T_r}|_{\mathcal{R}(J)}$$

are linearly independent then for any realization of $\mathcal{D}$ the rows $\mathbf{M}_{T_1}, \mathbf{M}_{T_2}, \cdots, \mathbf{M}_{T_r}$ are also linearly independent.

Proof. Fix any realization of $\mathcal{D}$. Using Observation 2.1, we can write:

$$\mathbf{C}|_{\mathcal{R}(J)}^{T_1,\ldots,T_r} = \mathbf{M}|_{\mathcal{R}(J)} \cdot \mathrm{diag}(\pi) \cdot \left(\mathbf{M}^\top\right)|^{T_1,\ldots,T_r}$$

Now suppose for the sake of contradiction that the rows of $\mathbf{M}|_{T_1,\ldots,T_r}$ are not linearly independent. Then there is a nonzero vector u so that $(\mathbf{M}^\top)|^{T_1,\ldots,T_r} u = 0$ which by the above equation immediately implies that the columns of $\mathbf{C}|_{\mathcal{R}(J)}^{T_1,\ldots,T_r}$ are not linearly independent, which yields our contradiction. □

Next we prove a partial converse to the above lemma:

Lemma 2.4. *Fix a realization of $\mathcal{D}$ and let $\mathcal{D}$ have rank k. Let $T_1, T_2, \cdots, T_r \subseteq [n]$ and set $J = \cup_i T_i$. If $\mathrm{rank}(\mathbf{M}|_{\mathcal{R}(J)}) = k$ and there are coefficients $\alpha_1, \alpha_2, \cdots, \alpha_r$ so that*

$$\sum_{i=1}^{r} \alpha_i \mathbf{C}^{T_i}|_{\mathcal{R}(J)} = 0$$

then the corresponding rows of $\mathbf{M}$ are linearly dependent too — i.e. $\sum_{i=1}^r \alpha_i \mathbf{M}_{T_i} = 0$.

Proof. By the assumptions of the lemma, we have that

$$\mathbf{M}|_{\mathcal{R}(J)} \cdot \mathrm{diag}(\pi) \cdot \left(\mathbf{M}^\top\right)|^{T_1,\ldots,T_r} \alpha = 0$$

Now $\mathrm{rank}(\mathbf{M}|_{\mathcal{R}(J)}) = k$ and the fact that the mixing weights are nonzero implies that $\mathbf{M}|_{\mathcal{R}(J)} \cdot \mathrm{diag}(\pi)$ is invertible. Hence we conclude that $\left(\mathbf{M}^\top\right)|^{T_1,\ldots,T_r} \alpha = 0$ as desired. □

Of course, we don't actually have exact estimates of the moments of $\mathcal{D}$. In Section 3.4 we discuss how to handle sampling noise to get an actual learning algorithm.

3 MIXTURES OF SUBCUBES

With these basic linear algebraic relations in hand, we can explain the intuition behind our algorithms. Our starting point is the observation that if we know a collection of sets $T_1, \ldots, T_k \subset [n]$ indexing a row basis of $\mathbf{M}$, then we can guess one of the $3^{k \cdot |T_1 \cup \cdots \cup T_k|}$ possibilities for the entries of $\mathbf{m}|_{T_1 \cup \cdots \cup T_k}$. Using a correct guess, we can solve for the mixing weights using (1) from Observation 2.1. The point is that because $T_1, \ldots, T_k$ index a row basis of $\mathbf{M}$, the system of equations

$$\mathbf{M}_{T_j} \cdot \vec{\pi} = \mathbb{E}_{\mathcal{D}}[x_{T_j}], \quad j = 1, \ldots, k \tag{1}$$

has a unique solution which thus must be the true mixing weights in the realization $(\vec{\pi}, \mathbf{m})$. We can then solve for the remaining rows of $\mathbf{m}$ using part 2 of Observation 2.1, i.e. for every $i \notin T_1 \cup \cdots \cup T_k$ we can solve

$$\mathbf{M}_{T_j} \cdot \mathrm{diag}(\pi) \cdot \mathbf{m}_i^\top = \mathbb{E}_{\mathcal{D}}[x_{T_j \cup \{i\}}] \quad \forall j = 1, \ldots, k. \tag{2}$$

Again, because the rows $\mathbf{M}_{T_i}$ are linearly independent and π has no zero entries, we conclude that the true value of $\mathbf{m}_i$ is the unique solution.

There are three main challenges to implementing this strategy:

A **Identifiability**. How do we know whether a given guess for $\mathbf{m}|_{T_1 \cup \cdots \cup T_k}$ is correct? More generally, how do we efficiently test whether a given distribution is close to the underlying mixture of subcubes?

B **Building a Basis**. How do we produce a row basis for $\mathbf{M}$ without knowing $\mathbf{M}$, let alone one for which $T_1 \cup \cdots \cup T_k$ is small enough that we can actually try all $3^{k \cdot |T_1 \cup \cdots \cup T_k|}$ possibilities for $\mathbf{m}|_{T_1 \cup \cdots \cup T_k}$?

C **Sampling Noise**. Technically we only have approximate access to the moments of $\mathcal{D}$, so even from a correct guess for $\mathbf{m}|_{T_1 \cup \cdots \cup T_k}$ we only obtain approximations to $\vec{\pi}$ and the remaining rows of $\mathbf{m}$. How does sampling noise affect the quality of these approximations?

3.1 Identifiability

As our algorithms will be based on the method of moments, an essential first question to answer is that of identifiability: what is the minimum d for which mixtures of k subcubes are uniquely identified by their moments of degree at most d? As alluded to in Section 1.2, it is enough to answer Question 1.1, which we can restate in our current notation as:

Question 3.1. *Given a matrix $\mathbf{m} \in \{0, 1/2, 1\}^{n \times k}$ with associated $2^n \times k$ moment matrix $\mathbf{M}$, what is the minimum d for which the rows $\{\mathbf{M}_S\}_{|S| \le d}$ span all rows of $\mathbf{M}$?*

Let $d(k)$ be the largest d for Question 3.1 among all possible $\mathbf{m} \in \{0, 1/2, 1\}^{n \times k}$. Note that $d(k) = \Omega(\log k)$ just from considering a $O(\log k)$-sparse parity with noise instance as a mixture of k subcubes. The reason getting upper bounds on $d(k)$ is directly related to identifiability is that k subcubes are uniquely identified by their moments of degree at most $d(2k)$. Indeed, if $(\vec{\pi}_1, \mathbf{m}_1)$ and $(\vec{\pi}_2, \mathbf{m}_2)$ realize different distributions $\mathcal{D}_1$ and $\mathcal{D}_2$, then there must exist $S \subseteq [n]$ for which

$$(\mathbf{M}_1)_S \cdot \vec{\pi}_1 = \mathbb{E}_{\mathcal{D}_1}[x_S] \neq \mathbb{E}_{\mathcal{D}_2}[x_S] = (\mathbf{M}_2)_S \cdot \vec{\pi}_2.$$

In other words, the vector $(\vec{\pi}_1 | -\vec{\pi}_2) \in \mathbb{R}^{2k}$ does not lie in the right kernel of the matrix $2^n \times 2k$ matrix $(\mathbf{M}_1 | \mathbf{M}_2)$. But because $\mathbf{N} \triangleq (\mathbf{M}_1 | \mathbf{M}_2)$ is the moment matrix of the matrix $(\mathbf{m}_1 | \mathbf{m}_2) \in \{0, 1/2, 1\}^{n \times 2k}$, its rows are spanned by the rows $(\mathbf{N}_S)_{|S| \le d(2k)}$, so there in fact exists S' of size at most $d(2k)$ for which $\mathbb{E}_{\mathcal{D}_1}[x_{S'}] \neq \mathbb{E}_{\mathcal{D}_2}[x_{S'}]$. Finally, note also that the reverse direction of this argument holds, that is, if mixtures of k subcubes $\mathcal{D}_1$ and $\mathcal{D}_2$ agree on all moments of degree at most $d(2k)$, then they are identical as distributions.

We now show that $d(k) = \Theta(\log k)$. The idea is that there is a natural correspondence between (1) linear relations among the rows of $\mathbf{M}_S$ for $|S| \le d$ and (2) multilinear polynomials of degree at most d which vanish on the rows of $\mathbf{m}$. The bound on $d(k)$ then follows from carefully constructing an appropriate low-degree multilinear polynomial.

Lemma 3.1. *Let $\mathcal{D}$ be a mixture of k subcubes and fix a realization where the centers are $\{0, 1/2, 1\}$-valued. Let $\mathbf{M}$ be the corresponding moment matrix. Then*

$$\left\{\mathbf{M}_T \middle| |T| < 2 \log k\right\}$$

span the rows of $\mathbf{M}$.

Proof. Fix any set $S \subseteq [n]$ of size $m = 2 \log k$. Without loss of generality suppose that $S = \{1, 2, \cdots, m\}$. We want to show that

$\mathbf{M}_S$ lies in the span of $\mathbf{M}_T$ for all $T \subsetneq S$. Our goal is to show that there are coefficients α_T so that

$$\sum_{T \subseteq S} \alpha_T \mathbf{M}_T = 0$$

and that α_S is nonzero. If we can do this, then we will be done. First we construct a multilinear polynomial

$$p(x) = \prod_{i=1}^{m} \left(x_i - \lambda_i\right)$$

where each $\lambda_i \in \{0, 1/2, 1\}$ and with the property that for any j, $p(\mathbf{m}^j|_S) = 0$. If we had such a polynomial, we could expand

$$p(x) = \sum_{T \subseteq S} \alpha_T \prod_{i \in T} x_i$$

By construction $\alpha_S = 1$. And now for any j we can see that the j^{th} coordinate of $\sum_{T \subseteq S} \alpha_T \mathbf{M}_T$ is exactly $p(\mathbf{m}^j|_S)$, which yields the desired linear dependence.

All that remains is to construct the polynomial p. We will do this by induction. Suppose we have constructed a polynomial $p_t(x) = \prod_{i=1}^{t}(x_i - \lambda_i)$ and let

$$R_t = \left\{ j \middle| p_t(\mathbf{m}^j|_S) \neq 0 \right\}$$

In particular $R_t \subseteq [k]$ is the set of surviving columns. By the pigeonhole principle we can choose $\lambda_{t+1} \in \{0, 1/2, 1\}$ so that $|R_{t+1}| \leq \lfloor (2/3)|R_t| \rfloor$. For some $\ell \leq m$ we have that $R_\ell = \emptyset$ at which point we can choose

$$p(x) = \left(\prod_{i=1}^{\ell} (x_i - \lambda_i) \right) \cdot \prod_{i=\ell+1}^{m} x_i$$

which completes the proof. □

Note that the above discussion only pertains to *exact identifiability*. For the purposes of our learning algorithm, we want *robust identifiability*, i.e. there is some $d'(k)$ such that $\mathcal{D}_1$ and $\mathcal{D}_2$ are far in statistical distance if and only if they differ noticeably on some moment of degree at most $d'(k)$. It turns out that it suffices to take $d'(k)$ to be the same $\Theta(\log k)$, and in Section 3.4 below, we sketch how we achieve this.

Once we have robust identifiability in hand, we have a way to resolve Challenge A above: to check whether a given guess for $\mathbf{m}|_{T_1 \cup \cdots \cup T_k}$ is correct, compute the moments of degree at most $\Theta(\log k)$ of the corresponding candidate mixture of subcubes and compare them to empirical estimates of the moments of the underlying mixture. If they are close, then the mixture of subcubes we have learned is close to the true distribution.

As we will see below though, while the bound of $d(k) = \Theta(\log k)$ is a necessary first step to achieving a quasipolynomial running time for our learning algorithm, there will be many more steps and subtleties along the way to getting an actual algorithm.

3.2 Building a Basis

We now describe how we address Challenge B. The key issue is that we do not have access to the entries of $\mathbf{M}$ (and $\mathbf{M}$ itself depends on the choice of a particular realization). Given the preceding discussion about Question 3.1, a naive way to circumvent this is simply to guess a basis from among all combinations of at most k rows from $\{\mathbf{M}_S\}_{|S| \leq d(k)}$, but this would take time $n^{\Theta(k \log k)}$.

As we hinted at in Section 1.2, we will overcome the issue of not having access to $\mathbf{M}$ by using the accessible entries of $\mathbf{C}$, which we can easily estimate by drawing samples from $\mathcal{D}$, as a surrogate for $\mathbf{M}$ (see Lemmas 2.3 and 2.4). To this end, one might first try to use $\mathbf{C}$ to find a row basis for $\mathbf{M}$ by looking at the submatrix of $\mathbf{C}$ consisting of entries $\{\mathbf{C}_S^T\}_{S,T:|S|,|T| \leq d(k)}$ and simply picking out a column basis $\{T_1, ..., T_m\}$ for this submatrix. Of course, the crucial issue is that we can only use the accessible entries of $\mathbf{C}$.

Instead, we will incrementally build up a row basis. Suppose at some point we have found a list of subsets $T_1, ..., T_m$ indexing linearly independent rows of $\mathbf{M}$ for some realization of $\mathcal{D}$ and are deciding whether to add some set T to this list. Recall that Lemma 2.3 and Lemma 2.4 give us a way to efficiently certify whether we should do so. Motivated by this, we introduce the following key definitions:

Definition 3.1. Given a collection $\mathcal{B} = \{T_1, T_2, \cdots, T_m\}$ of subsets we say that $\mathcal{B}$ is *certified full rank* if $\mathbf{C}|_{\mathcal{R}'(J)}^{T_1, T_2, \cdots, T_m}$ has full column rank, where $J = \cup_i T_i$ and $\mathcal{R}'(J) = \{T \subseteq [n] \backslash J : |T| < 2 \log k\}$.

Definition 3.2. Let $\mathcal{B} = \{T_1, T_2, \cdots, T_m\}$ be certified full rank. Let $J = \cup_i T_i$. Suppose there is no

(1) $T' \subseteq J$ or
(2) $T' = T_i \cup \{j\}$ for $j \notin J$

for which $\mathbf{C}|_{\mathcal{R}'(J')}^{T_1, T_2, \cdots, T_m, T'}$ has full column rank, where $J' = J \cup T'$. Then we say that $\mathcal{B}$ is *locally maximal*.

We will show that any certified full rank and locally maximal $\mathcal{B}$ spans a particular subset of the rows of $\mathbf{M}$ and then give a simple algorithm for producing such a $\mathcal{B}$. First we will show the following helper lemma:

Lemma 3.2. *Let $\mathcal{B} = \{T_1, T_2, \cdots, T_m\}$ and $J = \cup_i T_i$ as usual. Suppose that*

(1) the rows of $\mathbf{M}|_{\mathcal{B}}$ are a basis for the rows of $\mathbf{M}|_{2^J}$ and
(2) for any T_i and any $j \notin J$, the row $\mathbf{M}_{T_i \cup \{j\}}$ is in the row span of $\mathbf{M}|_{\mathcal{B}}$

Then the rows of $\mathbf{M}|_{\mathcal{B}}$ are a basis for the rows of $\mathbf{M}$.

Proof. We will proceed by induction. Suppose that the rows of $\mathbf{M}|_{\mathcal{B}}$ are a basis for the rows of $\mathbf{M}|_{2^{J'}}$ for some $J' \supseteq J$. Consider any $j \notin J'$. Then the rows

$$\mathbf{M}_{T_1}, \mathbf{M}_{T_2}, \cdots, \mathbf{M}_{T_m} \text{ and } \mathbf{M}_{T_1 \cup \{j\}}, \mathbf{M}_{T_2 \cup \{j\}}, \cdots, \mathbf{M}_{T_m \cup \{j\}}$$

are a basis for the rows of $\mathbf{M}|_{2^{J' \cup \{j\}}}$. But by assumption each row $\mathbf{M}_{T_i \cup \{j\}}$ is in the row span of $\mathbf{M}|_{\mathcal{B}}$. Thus the rows of $\mathbf{M}|_{\mathcal{B}}$ are also a basis for the rows of $\mathbf{M}|_{2^{J' \cup \{j\}}}$, as desired. □

Now we are ready to prove the main lemma in this subsection:

Lemma 3.3. *Let $\mathcal{D}$ have rank k and fix a full rank realization of $\mathcal{D}$. Let $\mathcal{B} = \{T_1, T_2, \cdots, T_m\}$ be certified full rank and locally maximal. Let $J = \cup_i T_i$ and*

$$K = \left\{ i \middle| i \notin J \text{ and } \operatorname{rank}(\mathbf{M}|_{\mathcal{R}'(J \cup \{i\})}) = k \right\}$$

If $K \neq \emptyset$ then the rows of $\mathbf{M}|_{\mathcal{B}}$ are a basis for the rows of $\mathbf{M}|_{2^{J \cup K}}$.

Proof. Our strategy is to apply Lemma 3.2 to the set $J \cup K$ which will give the desired conclusion. To do this we just need to verify that the conditions in Lemma 3.2 hold. We will need to pay special attention to the distinction between $\mathcal{R}(J)$ and $\mathcal{R}'(J)$. First take any $i \in K$. Then

$$k = \text{rank}(\mathbf{M}|_{\mathcal{R}'(J\cup\{i\})}) = \text{rank}(\mathbf{M}|_{\mathcal{R}(J\cup\{i\})}) = \text{rank}(\mathbf{M}|_{\mathcal{R}(J)})$$

The first equality follows from how we constructed K. The second equality follows from Lemma 3.1 when applied to the set $[n]\backslash J \cup \{i\}$. The third equality follows because the rows of $\mathbf{M}|_{\mathcal{R}(J\cup\{i\})}$ are a subset of the rows of $\mathbf{M}|_{\mathcal{R}(J)}$ and $\mathbf{M}$ has rank k.

Now the first condition of local maximality implies that there is no $T' \subseteq J$ where $\mathbf{C}|^{T_1,T_2,\cdots,T_m,T'}_{\mathcal{R}'(J)}$ has full column rank. Lemma 3.1 implies that $\mathbf{C}|^{T_1,T_2,\cdots,T_m,T'}_{\mathcal{R}(J)}$ also does not have full column rank because the additional rows of the latter can be obtained as linear combinations of the rows in the former. Now we can invoke Lemma 2.4 which implies that $\mathbf{M}_{T'}$ is in the span of $\mathbf{M}|_{\mathcal{B}}$. Thus the rows of $\mathbf{M}|_{\mathcal{B}}$ are indeed a basis for the rows of $\mathbf{M}|_{2^J}$, which is the first condition we needed to check.

For the second condition, the chain of reasoning is similar. Consider any $i \in K$ and any $T_{i'} \in \mathcal{B}$. Set $T' = T_{i'} \cup \{i\}$ and $J' = J \cup \{i\}$. Then $\text{rank}(\mathbf{M}|_{\mathcal{R}'(J')}) = k$. Now the second condition of local maximality implies that $\mathbf{C}|^{T_1,T_2,\cdots,T_m,T'}_{\mathcal{R}'(J')}$ does not have full column rank. Lemma 3.1 implies that $\mathbf{C}|^{T_1,T_2,\cdots,T_m,T'}_{\mathcal{R}(J')}$ does not have full column rank either. We can once again invoke Lemma 2.4 which implies that $\mathbf{M}_{T'}$ is in the span of $\mathbf{M}|_{\mathcal{B}}$, which is the second condition we needed to verify. This completes the proof. □

The algorithm for producing such a certified full rank and locally maximal $\mathcal{B}$, which we call GrowByOne, is simple. Given $T_1, ..., T_m$ which have already been added to $\mathcal{B}$, consider all subsets of the form $T_j \cup \{i\}$ for $1 \le j \le m$ and $i \notin T_1 \cup \cdots \cup T_m$. If $T_1, ..., T_m$ have up to this point been constructed in this incremental fashion, we can use ideas from the proof of Lemma 3.3 to prove that if no such $T_j \cup \{i\}$ can be added to our list and moreover we have that $\text{rank}(\mathbf{M}|_{\mathcal{R}(J)}) = \text{rank}(\mathbf{M}|_{\mathcal{R}(T_1\cup\cdots\cup T_m\cup\{i\})}) = k$ for every i, then $T_1, ..., T_m$ indexes a row basis for $\mathbf{M}$. We refer the reader to the full version for the details of this.

3.3 Making Progress When Basis-Building Fails

The main subtlety is that the set K in Lemma 3.3 need not be all of $[n]$. In particular, we may run into a point where for any new candidate subset T we are trying to add to $\mathcal{B} = \{T_1, ..., T_m\}$, we have that $\mathbf{C}^T_{\mathcal{R}(J)}$ lies in the span of $\mathbf{C}^{T_1}_{\mathcal{R}(J)}, ..., \mathbf{C}^{T_m}_{\mathcal{R}(J)}$. In this case we cannot definitively conclude whether any $\mathbf{M}_T$ lies in the span of $\mathbf{M}_{T_1}, ..., \mathbf{M}_{T_m}$ and therefore decide to add nothing more to $\mathcal{B}$.

The key idea is that if this is the case, then there must have been some candidate $T = T_j \cup \{i\}$ such that $\text{rank}(\mathbf{M}|_{\mathcal{R}(T_1\cup\cdots\cup T_m\cup\{i\})}) < k$. We call the set of all such i the set of *impostors*. Note that the complement of this set in $[n]\backslash J$ is precisely the set K in Lemma 3.3. By Lemma 2.1, if i is an impostor, the conditional distribution $(\mathcal{D}|x_{T_1\cup\cdots\cup T_m\cup\{i\}} = s)$ can be realized as a mixture of strictly fewer than k subcubes for any bitstring s. The upshot is that even if the list $T_1, ..., T_m$ output by GrowByOne does not correspond to a row basis of $\mathbf{M}$, we can make progress by conditioning on the coordinates $T_1 \cup \cdots \cup T_m \cup \{i\}$ for an impostor i and recursively learning mixtures of fewer subcubes.

On the other hand, the issue of actually identifying an impostor $i \notin T_1 \cup \cdots \cup T_m$ is quite delicate. Because there may be up to k levels of recursion, we cannot afford to simply brute force over all $n - |T_1 \cup \cdots \cup T_m|$ possible coordinates. Instead, the idea will be to pretend that $T_1, ..., T_m$ actually corresponds to a row basis of $\mathbf{M}$ and use this to attempt to learn the parameters of the mixture. It turns out that either the resulting mixture will be close to $\mathcal{D}$ on all low-degree moments and robust identifiability will imply we have successfully learned $\mathcal{D}$, or it will disagree on some low-degree moment, and we show that this low-degree moment must contain an impostor i:

Lemma 3.4. *Let $\mathcal{D}$ have rank k and fix a full rank realization of $\mathcal{D}$. Let $\mathcal{B} = \{T_1, T_2, \cdots, T_m\}$ be certified full rank and locally maximal. Let $J = \cup_i T_i$. Let I be the set of impostors and K be the set of non-impostors.*

There is a guess $\mathbf{m}'|_J \in \{0, 1/2, 1\}^{|J|\times m}$ so that if we solve (1) and solve (2) for each $i \in K$ we get parameters that generate a mixture of subcubes $\mathcal{D}'$ on $J \cup K$ that satisfy $\mathbb{E}_{\mathcal{D}'}[x_S] = \mathbb{E}_{\mathcal{D}}[x_S]$ for all $S \subseteq J \cup K$.

Proof. For any $i \in K$ we have $\text{rank}(\mathbf{M}|_{\mathcal{R}'(J\cup\{i\})}) = k$. By Lemma 3.3 we know that $\mathbf{M}_{\mathcal{B}}$ is a row basis for $\mathbf{M}_{2^{J\cup K}}$. In particular $\text{rank}(\mathbf{M}_{2^{J\cup K}}) = m$. Thus using Lemma 2.1 there is a mixture of m subcubes with mixing weights π' and marginals matrix $\mathbf{m}' \in \{0, 1/2, 1\}^{|J\cup K|\times m}$ that realizes the same distribution as projecting $\mathcal{D}$ onto coordinates in $J \cup K$ (i.e. without conditioning on any coordinates outside of this set).

Let $\mathbf{M}'$ be the corresponding moment matrix. Then by construction $\mathbf{M}'$ consists of a subset of the columns of $\mathbf{M}_{2^{J\cup K}}$. Thus the rows of $\mathbf{M}'_{\mathcal{B}}$ still span the rows of $\mathbf{M}'$. Also by construction $\mathbf{M}'$ has rank m and hence the rows of $\mathbf{M}'_{\mathcal{B}}$ are linearly independent. Now if we take our guess to be $\mathbf{m}'|_J$ where $\mathbf{m}'$ is as above, (1) has a unique solution, namely π'. Also for each $i \in K$, (2) has a unique solution namely $\mathbf{m}'_i$. Now if we take our learned parameters we get a mixture of subcubes $\mathcal{D}'$ on $J \cup K$ that satisfies $\mathbb{E}_{\mathcal{D}'}[x_S] = \mathbb{E}_{\mathcal{D}}[x_S]$ for all $S \subseteq J \cup K$ because $\mathcal{D}'$ and projecting $\mathcal{D}$ onto coordinates in $J \cup K$ realize the same distribution. This completes the proof. □

So once we have a certified full rank and locally maximal $\mathcal{B} = \{T_1, ..., T_m\}$ with $J = \cup T_i$, we can guess $\mathbf{m}'|_J \in \{0, 1/2, 1\}^{|J|\times m}$ and solve (1) and solve (2) for each $i \in [n]\backslash J$ (because we do not know the set of impostors). We can then check whether the parameters we get generate a mixture of subcubes $\mathcal{D}'$ that satisfies

$$\mathbb{E}_{\mathcal{D}'}[x_S] = \mathbb{E}_{\mathcal{D}}[x_S]$$

for all S with $|S| \le c \log k$. If it does, then $\mathcal{D}' = \mathcal{D}$ and we are done. But if there is an S where the equation above is violated (and our guess was correct) then S cannot be a subset of $J \cup K$ which means that it contains an impostor. Thus the fact that we can check the equation above only up to logarithmic sized moments gives us a way to trace an impostor down to a logarithmic sized set, so that we can condition on $S \cup J$ and make progress without needing to fix too many coordinates.

3.4 Sampling Noise

The above discussion assumed exact access to the entries of $\mathbf{C}$, but obviously we only have access to empirical estimates of the entries of $\mathbf{C}$. So for instance, instead of checking whether a column of $\mathbf{C}$ lies in the span of other columns of $\mathbf{C}$, we must look at the corresponding L_∞ regression problem. In this setting, the above arguments still carry over provided that the submatrices of $\mathbf{M}$ and $\mathbf{C}$ used are well-conditioned. In the full version, we show that the former are well-conditioned by Cramer's, as they are matrices whose entries are low-degree powers of 1/2, and this on its own can already be used to show robust identifiability. By Observation 2.1, the submatrices of $\mathbf{C}$ used in the above arguments are also well-conditioned provided that π has no small entries. But if π has small entries, intuitively we might as well ignore these entries and only attempt to learn the subcubes of the mixture which have non-negligible mixing weight.

We refer the reader to the full version for further details on the subtleties that go into dealing with these issues of sampling noise.

4 A STATISTICAL QUERY LOWER BOUND

In this section we sketch our argument for exhibiting a lower bound against learning mixtures of general product distributions over $\{0,1\}^n$ under the powerful statistical query (SQ) model, first introduced in [15].

Definition 4.1. Fix a distribution $\mathcal{D}$ over $\{0,1\}^n$. For tolerance parameter $\tau > 0$, the STAT(τ) oracle answers any query $h : \{0,1\}^n \to [-1,1]$ with a value v such that

$$|\mathbb{E}_{x\sim\mathcal{D}}[h(x)] - v| \le \tau.$$

We show that learning mixtures of product distributions requires many calls to STAT by bounding the *SQ dimension*, first defined in [4] for learning Boolean functions and subsequently extended by [10] to learning distributions. To do so, it turns out that it suffices to construct a family of mixtures of k product distributions such that each mixtures matches U_n up to some large number $d^*(k)-1$ of moments and differs noticably on exactly one degree-$(d^*(k)+1)$ moment. We refer the reader to the full version of our paper for a proof that this suffices and proceed to sketch how to construct such a family.

We first emphasize that constructing such a family is strictly harder than simply constructing a pair of mixtures of product distributions which match on a large number of moments. To answer the latter, one could start by asking Question 3.1 more generally for arbitrary matrices $\mathbf{m} \in \mathbb{R}^{n\times k}$. It is not hard to see that the minimum d for which the rows $\{\mathbf{M}_S\}_{|S|\le d}$ span all rows of $\mathbf{M}$ can be as high as $k-1$. Simply take $\mathbf{m}$ to have identical rows, each of which consists of k distinct entries $z_1,\dots,z_k \in [0,1]$. Then $\mathbf{M}_S = (z_1^{|S|},\dots,z_k^{|S|})$, so by usual properties of Vandermonde matrices, the rows $\{\mathbf{M}_S\}_{|S|\le d}$ will not span the rows of $\mathbf{M}$ until $d \ge k-1$.[5]

From such an $\mathbf{m}$, we immediately get a pair of mixtures $(\vec{\mu}_1, \mathbf{m}_1)$ and $(\vec{\mu}_2, \mathbf{m}_2)$ that agree on all moments of degree at most $k-2$ but differ on moments of degree $k-1$: let $\vec{\mu}_1$ and $-\vec{\mu}_2$ up to scaling be the positive and negative parts of an element in the kernel of $\{M_S\}_{|S|<k-1}$, and let $\mathbf{m}_1$ and $\mathbf{m}_2$ be the corresponding disjoint submatrices of $\mathbf{m}$.

We now turn to the more difficult task of constructing a *family* of mixtures which match U_n on many moments. The SQ lower bound given in [9] essentially stems from the SQ lower bound for learning $\log k$-sparse parities (an example of a mixture of k subcubes), where the idea is that U_n and the uniform distribution over positive examples of a $\log k$-sparse parity agree on all moments of degree less than $\log k$ and differ on exactly one moment of degree $\log k + 1$, corresponding to the coordinates of the parity. The observation that leads to our SQ lower bound is that for general mixtures of k product distributions, we can come up with much harder instances which agree with U_n even on moments of degree at most $O(\sqrt{k})$.

Our general approach is to construct a mixture $\mathcal{A}$ of product distributions over $\{0,1\}^{d^*(k)}$ whose top-degree moment differs noticeably from $2^{-d^*(k)}$ but whose other moments agree with that of $U_{d^*(k)}$. The collection C of mixtures will then consist of all product measures given by $\mathcal{A}$ in some $d^*(k)$ coordinates S and $U_{n-d^*(k)}$ in the remaining coordinates $[n]\backslash S$. This general strategy of embedding a low-dimensional moment-matching distribution $\mathcal{A}$ in some hidden set of coordinates is the same principle behind SQ lower bounds for learning sparse parity [15], robust estimation and density estimation of mixtures of Gaussians [7], etc.

4.1 A Moment Matching Example

The main challenge is to actually construct the mixture $\mathcal{A}$. Here we reduce this problem to Question 1.2.

Definition 4.2. Given $\pi \in \Delta^k$, a collection of vectors $v_1,\dots,v_m \in \mathbb{R}^k$ is *d-wise superorthogonal with respect to π* if for any $S \subseteq [m]$ of size at most d, $\langle \bigodot_{i\in S} v_i, \pi\rangle = 0$. Note that if $\pi = \frac{1}{k}\cdot\vec{\mathbf{1}}$ and $d = 2$, this is just the usual notion of orthogonality.

Lemma 4.1. *Let $d \le m$ and suppose A is a mixture of product distributions with mixing weights π and marginals matrix $\mathbf{m}$. Then A and U_m agree on moments of degree at most d if and only if the rows of $\mathbf{m} - \frac{1}{2}\cdot\mathbf{J}_{m\times k}$ are d-wise superorthogonal with respect to π, where $\mathbf{J}_{m\times k}$ is the $m\times k$ all-ones matrix.*

Proof. For any $S \subseteq [m]$ of size at most d,

$$\left\langle \bigodot_{i\in S}\left(\mathbf{m}_i - \frac{1}{2}\cdot\vec{\mathbf{1}}\right), \pi\right\rangle = \sum_{T\subseteq S}(-1/2)^{|S|-|T|}\langle \mathbf{M}_T, \pi\rangle. \tag{3}$$

So if A and U_m agree on moments of degree at most d so that $\langle \mathbf{M}_T, \pi\rangle = 1/2^{|T|}$ for all $|T| \le d$, this is equal to

$$(1/2)^{|S|}\cdot\sum_{T\subseteq S}(-1)^{|S|-|T|} = 0.$$

Conversely, if the rows of $\mathbf{m} - \frac{1}{2}\cdot\mathbf{J}_{m\times k}$ are indeed d-wise superorthogonal with respect to π, then by induction on degree, the fact that (3) vanishes forces $\langle \mathbf{M}_S, \pi\rangle$ to be $2^{|S|}$. □

Because we insist that A and U_m agree on their moments of degree less than m and differ on their m-th moment, Lemma 4.1 reduces the task of constructing A to that of constructing a collection of vectors that is $(m-1)$-wise but not m-wise superorthogonal with respect to π.

[5] Note that by the connection between linear relations among rows of $\mathbf{M}_S$ and multilinear polynomials vanishing on the rows of $\mathbf{m}$, this example is also tight, i.e. $\{\mathbf{M}_S\}_{|S|\le k-1}$ will span the rows of $\mathbf{M}$ for any $m \in \mathbb{R}^{n\times k}$.

Definition 4.3. A collection of vectors $v_1, ..., v_\ell \in \mathbb{R}^k$ is *non-top-degree-vanishing* if $v_1 \odot \cdots \odot v_\ell$ does not lie in the span of $\{\odot_{i \in S} v_i\}_{S \subsetneq [\ell]}$.

Observation 4.1. *Suppose $v_1, ..., v_{m-1} \in \mathbb{R}^k$ are $(m-1)$-wise superorthogonal with respect to π and non-top-degree-vanishing. Denote the span of $\{\odot_{i \in S} v_i\}_{S \subsetneq [m-1]}$ by V. If $v_m \in \mathbb{R}^k$ satisfies*

$$v_m \cdot diag(\pi) \cdot v^\top = 0 \; \forall \, v \in V \tag{4}$$

$$v_m \cdot diag(\pi) \cdot (v_1 \odot \cdots \odot v_{m-1})^\top \neq 0, \tag{5}$$

then $v_1, ..., v_m$ are $(m-1)$- but not m-wise superorthogonal with respect to π.

Note that any collection of vectors that are $(m-1)$-wise but not m-wise superorthogonal with respect to π must arise in this way. By Observation 4.1, we can focus on finding the largest ℓ for which there exist vectors $v_1, ..., v_\ell$ which are ℓ-wise superorthogonal and non-top-degree-vanishing.

Construction. *Let $k = (\ell+1)^2$ and $\pi = \frac{1}{k}\vec{1}$, and fix any distinct scalars $x_1, ..., x_{\ell+1} \in \mathbb{R}$. Define matrices*

$$\mathbf{a} = \begin{pmatrix} x_1 & x_2 & \cdots & x_{\ell+1} \\ x_1 & x_2 & \cdots & x_{\ell+1} \\ \vdots & \vdots & \ddots & \vdots \\ x_1 & x_2 & \cdots & x_{\ell+1} \end{pmatrix} \quad \mathbf{b_i} = \begin{pmatrix} -x_i & 0 & 0 & \cdots & 0 \\ x_i & -2x_i & 0 & \cdots & 0 \\ x_i & x_i & -3x_i & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_i & x_i & x_i & \cdots & -\ell x_i \end{pmatrix}$$

with ℓ rows each. Define the $\ell \times k$ matrix

$$\mathcal{E}(x_1, ..., x_{\ell+1}) := (\mathbf{a} \| \mathbf{b_1} \| \cdots \| \mathbf{b}_{\ell+1}).$$

We refer the reader to the full version for an analysis of this construction.

5 MIXTURES OF PRODUCT DISTRIBUTIONS

The main difficulty with learning mixtures of general product distributions is that moment matrices can be arbitrarily ill-conditioned, which makes it far more difficult to handle sampling noise. Indeed, with exact access to the accessible entries of $\mathbf{C}$, one can in fact show there exists a $n^{O(d^*(k))}$ algorithm for learning mixtures of general product distributions, where $d^*(k)$ is the answer to Question 1.2, though we omit the proof of this in this work. In the presence of sampling noise, it is not immediately clear how to adapt the approach from Section 3. The three main challenges are:

A **Robust Identifiability**. For mixtures of subcubes, robust identifiability essentially followed from exact identifiability and a condition number bound on $\mathbf{M}$. Now that $\mathbf{M}$ can be arbitrarily ill-conditioned, how do we still show that two mixtures of product distributions that are far in statistical distance must differ noticeably on some low-degree moment?

B **Using C as a Proxy for M**. Without a condition number bound, can approximate access to $\mathbf{C}$ still be useful for deducing (approximate) linear algebraic relations among the rows of $\mathbf{M}$?

C **Guessing Entries of m**. Entries of $\mathbf{m}$ are arbitrary scalars now, rather than numbers from $\{0, 1/2, 1\}$. We can still try discretizing by guessing integer multiples $0, \eta, 2\eta, ..., 1$ of some small scalar η, but how small must η be for this to work?

For Challenge A, we show that if two mixtures of k product distributions are far in statistical distance, they must differ noticeably on some moment of degree at most $2k$. Specifically, we prove the following:

Lemma 5.1. *Let $\mathcal{D}_1, \mathcal{D}_2$ respectively be mixtures of k_1 and k_2 product distributions in $\{0,1\}^n$ for k_1, k_2. If $d_{TV}(\mathcal{D}_1, \mathcal{D}_2) > \epsilon$, there is some S for which $|S| < k_1 + k_2$ and $|\mathbb{E}_{\mathcal{D}_1}[x_S] - \mathbb{E}_{\mathcal{D}_2}[x_S]| > \eta$ for some $\eta = \eta(n, k_1 + k_2, \epsilon) \triangleq \exp(-O(k_1 + k_2)^2) \cdot poly(k_1 + k_2, n, \epsilon)^{-k_1 - k_2}$.*

Roughly, the proof is by induction on the total number of product distributions in the two mixtures; the inductive step is rather involved and we refer the reader to the full version.

Next, we make Challenges B and C more manageable by shifting our goal: instead of a row basis for $\mathbf{M}$, we would like a row basis for $\mathbf{m}$ that is well-conditioned in an appropriate sense. Specifically, we want a row basis $J \subset [n]$ for $\mathbf{m}$ such that if we express any other row of $\mathbf{m}$ as a linear combination of this basis, the corresponding coefficients are small. This is precisely the notion of *barycentric spanner* introduced in [2], where it was shown that any collection of vectors has a barycentric spanner. We can find a barycentric spanner for the rows of $\mathbf{m}$ by simply guessing all $\binom{n}{k}$ possibilities. We then show that if $J = \{i_1, ..., i_r\}$ is a barycentric spanner and $\mathbf{M}|_{\mathcal{R}(J \cup i_j)}$ is well-conditioned in an L_∞ sense for all $1 \leq j \leq r$, then in analogy with Lemma 2.4, one can learn good approximations to the true coefficients expressing the remaining rows of $\mathbf{m}$ in terms of $\mathbf{m}_{i_1}, ..., \mathbf{m}_{i_r}$. Concretely, for every remaining row i of $\mathbf{m}$, we may solve the regression problem

$$\tilde{\alpha}_i := \operatorname*{argmin}_{\alpha \in [-1,1]^r} \left\| \tilde{\mathbf{C}}|^{\{i_1\},...,\{i_r\}}_{\mathcal{R}'(J \cup \{i\})} \alpha - \tilde{\mathbf{C}}|^{\{i\}}_{\mathcal{R}'(J \cup \{i\})} \right\|_\infty. \tag{6}$$

and compute an approximation $\overline{\mathbf{m}}_i = \tilde{\alpha}_i \cdot \overline{\mathbf{m}}|_J$ to $\mathbf{m}_i$. Furthermore, this approximation is good enough that it suffices to pick the discretization parameter in Challenge C to be $\eta = \text{poly}(\epsilon/n)$, in which case the k^2 entries of $\mathbf{m}|_J$ can be guessed in time $(n/\epsilon)^{O(k^2)}$.

As usual, one complication is that it could be the case that $\text{rank}(\mathbf{M}|_{\mathcal{R}'(J \cup \{i\})}) < k$ for some realization of $\mathcal{D}$, but as in our algorithm for learning mixtures of subcubes, we can handle this by conditioning on $J \cup \{i\}$ and recursing. The more problematic issue that comes up here but not in the subcube setting is that $\mathbf{M}|_{\mathcal{R}'(J \cup \{i\})}$ might be full rank but very badly conditioned, in which case we cannot ensure that $\overline{\mathbf{m}}_i$ defined above is close to the true $\mathbf{m}_i$.

It turns out this issue is not so different from the situation where $\text{rank}(\mathbf{M}|_{\mathcal{R}'(J \cup \{i\})}) < k$ for some realization of $\mathcal{D}$, and we can effectively treat ill-conditioned moment matrices as degenerate-rank moment matrices. As we only insist on a runtime of $n^{O(k^2)}$, we can afford now to simply brute-force search for the impostor, but we cannot appeal to Lemma 2.1 to argue as before that each of the conditional distributions $(\mathcal{D}|x_{J \cup \{i_j\}} = s)$ is a mixture of fewer than k product distributions, because $\mathbf{M}|_{\mathcal{R}(J \cup \{i_j\})}$ might still have rank k. Instead, we show that robust identifiability implies that these conditional distributions are *close* to mixtures of at most $k-1$ product distributions, and this is enough for us to make progress and recursively learn:

Lemma 5.2. *The following holds for any $\eta > 0$. Let $\mathcal{D}$ be a mixture of k product distributions realized by mixing weights π and marginals*

matrix $\mathbf{m}$ *such that*

$$\sigma^{\infty}_{\min}(\mathbf{M}) \le \frac{\eta \cdot \sqrt{2}}{3k^2}.$$

Then there exists $\mathcal{D}'$ *a mixture of at most* $k-1$ *product distributions realized by mixing weights* π' *and marginals matrix* $\mathbf{m}'$ *such that* $|\mathbb{E}_{\mathcal{D}}[x_S] - \mathbb{E}_{\mathcal{D}'}[x_S]| \le \eta$ *for all* $|S| \le k$. *In particular, if we take* $\eta = \eta(n, 2k, \epsilon)$, *then by Lemma 5.1,* $d_{TV}(\mathcal{D}, \mathcal{D}') \le \epsilon$.

5.1 Comparison to Feldman et al.'s Algorithm

The algorithm of Feldman, O'Donnell and Servedio [9] also uses brute-force search to find a basis for the rows of $\mathbf{m}$. However instead of constructing a barycentric spanner they construct a basis that is approximately as well-conditioned as $\mathbf{m}$. Their algorithm proceeds by gridding the entries $\mathbf{m}|_J$. The key difference between their approach and ours is that their gridding requires granularity $O((\epsilon/n)^k)$ while ours requires only $O(\epsilon/n)$. The reason is that they try to solve for the other rows of $\mathbf{m}$ in the same way that we do in (2) when learning mixtures of subcubes, that is, by solving a system of equations for each $i \notin J$ with coefficients given by row $\mathbf{m}_i$. They require granularity $O((\epsilon/n)^k)$ to account for $\mathbf{m}$ being ill-conditioned. Just as we showed we could assume in our algorithm for mixtures of subcubes that the mixture weights had a gap of $\rho = 2^{-O(k^2)}$, [9] showed they can assume that $\mathbf{m}$ has a *spectral* gap of $O(\epsilon/n)$ by brute-forcing singular vectors of $\mathbf{m}$ and appending them to $\mathbf{m}$ to make it better conditioned. Such a spectral gap corresponds in the worst case to an $\mathbf{m}$ that is $O((\epsilon/n)^k)$-well-conditioned, which in turn ends up as the granularity in their gridding procedure. As a result, the bottleneck in the algorithm of [9] is the $(n/\epsilon)^{O(k^3)}$ time spent just to grid the entries of $\mathbf{m}|_J$.

In comparison, we save a factor of k in the exponent of the running time by only $O(\epsilon/n)$-gridding the entries of $\mathbf{m}|_J$. The reason is that we solve for the remaining rows of $\mathbf{m}$ not by solving systems of equations with coefficients in the rows $\mathbf{m}_i$ for $i \notin J$, but by expressing these rows $\mathbf{m}_i$ as linear combinations of the rows of $\mathbf{m}|_J$, where the linear combinations have bounded coefficients. This leverages higher order multilinear moments to make the linear system better conditioned. We estimate these coefficients by solving the regression problem (6), and the coefficients are accurate so long as the sampling error is $O(\epsilon/n)$ times the condition number of $\mathbf{M}|_{\mathcal{R}'(J\cup\{i\})}$ for J the barycentric spanner of the rows of $\mathbf{m}$ and any $i \notin J$. So in our algorithm, the bottlenecks leading to a k^2 dependence in the exponent are (1) $O(\epsilon/n)$-gridding all $O(k^2)$ entries of $\mathbf{m}|_J$, (2) brute-forcing $O(k)$ coordinates to condition in every one of the $\le k$ recursive steps, (3) using degree-$O(k^2)$ subsets in $\mathcal{R}'(J \cup \{i\})$ to ensure that when we condition on each of at most k subsequent subsets $J \cup \{i\}$, the resulting mixtures are all close in low-order moments to mixtures of fewer components.

A APPLICATION: LEARNING STOCHASTIC DECISION TREES

In this section, we prove Theorem 1.2. We begin with a warmup:

Example A.1 (Parity and juntas). *The uniform distribution* $\mathcal{D}$ *over the positive examples of a* k*-junta* $f : \{0,1\}^n \to \{0,1\}$ *is a mixture of at most* 2^k *subcubes in* $\{0,1\}^n$. *Let* $I \subseteq [n]$ *be the* k *coordinates that* f *depends. Every* $s \in \{0,1\}^{|I|}$ *for which* $f(x) = 1$ *for all* x *satisfying* $x_I = s$ *corresponds to a subcube with mixture weight* $1/N$, *where* $N \le 2^k$ *is the number of such* s *(e.g. when* f *is a parity,* $N = 2^{k-1}$*). In the same way we can show that the uniform distribution over the negative examples is also a mixture of at most* 2^k *subcubes.*

So given access to examples $(x, f(x))$ *where* x *is uniformly distributed over* $\{0,1\}^n$, *we can learn* f *as follows. With high probability, we can determine* $b^* \in \{0,1\}$ *for which* f *outputs* b^* *on at least* $1/3$ *of the inputs. As we have shown in this work, our algorithm can then learn some* $\mathcal{D}'$ *that is* ϵ*-close to the uniform distribution over* $\{x : f(x) = b^*\}$. *We then output the hypothesis* g *given by* $g(x) = b^*$ *if* $\mathcal{D}'(x) \le 1/2^{n+1}$ *and* $g(x) = 1 - b^*$ *otherwise. It is easy to see that* g *is* ϵ*-accurate.*

This approach can handle mild random classification noise γ*: if we take the distribution over examples* (x, b) *where* x *is drawn from the uniform distribution over* $\{0,1\}^n$ *and* b *is labeled by* $f(x)$ *with probability* $1-\gamma$ *and* $1 - f(x)$ *with probability* γ, *and we condition on* $b = 1$, *the resulting distribution is still a mixture of subcubes: every* s *for which* $f(x) = 1$ *for all* $x_I = s$ *corresponds to a subcube of weight* $(1-\gamma)/N$, *and every other* s *corresponds to a subcube of weight* γ/N. *This mixture is* $O(\gamma)$*-far from the uniform distribution over* $\{x : f(x) = 1\}$, *so in the above analysis, our algorithm would give an* $(\epsilon + O(\gamma))$*-accurate hypothesis.*

Finally, note that if mixing weights π *and marginals matrix* $\mathbf{m}$ *realize* $\mathcal{D}$, *then* $\mathbf{m}_i \in \{0,1\}^k$ *if* f *depends on coordinate* k, *and* $\mathbf{m}_i = (1/2, \ldots, 1/2)$ *otherwise, meaning the rows of* $\mathbf{M}$ *are spanned by all entrywise products of degree less than* $\log_2(N) \le k$, *rather than* $2\log(N)$ *as is required in general by N-List. So the algorithm we described above has the same performance as the brute-force algorithm.*

The above example serves simply to suggest the naturality of the problem of learning mixtures of subcubes, but because there are strong SQ lower bounds against learning sparse noisy parity [4], it's inevitable that our algorithm gives no new improvements over such problems. We now describe an application of N-List which does achieve a new result on a classical learning theory problem.

Definition A.1. A *stochastic decision tree* T on n bits is a tree with leaves labeled by 0 or 1 and with internal nodes of two types: *decision nodes* and *stochastic nodes*. Each decision node is labeled with some $i \in [n]$ and has two outgoing edges, one labeled with 0 and the other with 1. Each stochastic node u has some number of outgoing edges uv each labeled with a probability p_{uv} such that $\sum_v p_{uv} = 1$.

T defines a joint probability distribution D_T on $\{0,1\}^n \times \{0,1\}$. The $x \in \{0,1\}^n$ is sampled uniformly at random. Then given x, the conditional distribution can be sampled from by walking down the tree as follows. At a decision node labeled with i, traverse along the edge labeled by x_i. At a stochastic node u with outgoing edges labeled p_{uv}, pick edge uv with probability p_{uv} and traverse along that edge. When we reach a leaf node, output its value b. In this case we say that x *evaluates to* b along this path.

If T has m decision nodes and some stochastic nodes u each with some outdegree d_u, then T has $m + \sum_u (d_u - 1)$ leaves.

Lemma A.1. *For any* k*-leaf stochastic decision tree* T *on* n *bits, the distribution of* $(x, b) \sim D_T$ *conditioned on* $b = 1$ *is a mixture of* k *subcubes.*

Proof. Consider any path p in T from the root to a leaf labeled with 1. If along this path there are m decision nodes corresponding to some variables $i_1, \ldots, i_m \in [n]$ and with outgoing edges $b_1, \ldots, b_m$, then any $x \in \{0,1\}^n$ from the subcube corresponding to the conjunction $(x_{i_1} = b_1) \wedge \cdots \wedge (x_{i_m} = b_m)$ evaluates to 1 along this path with probability equal to the product μ_p of the edge weights along this path which emanate from stochastic nodes. So the distribution of $(x, b) \sim D_T$ conditioned on $b = 1$ is a mixture of k such subcubes, where the p-th subcube has mixture weight proportional to $\mu_p / 2^{d_p}$, where d_p is the number of decision nodes along path p. □

The following immediately implies our algorithm for learning stochastic decision trees (Theorem 1.2).

Lemma A.2. *Let T be any k-leaf stochastic decision tree corresponding to a joint probability distribution D_T on $\{0,1\}^n \times \{0,1\}$. Given access to samples from D_T, D_T can be learned to within total variation distance ϵ with probability at least $1 - \delta$ in time*

$$O_{k,s}(n^{O(s + \log k)}(1/\epsilon)^{O(1)} \log 1/\delta)$$

and with sample complexity $O_{k,s}((\log n/\epsilon)^{O(1)} \log 1/\delta)$

Proof. Denote by A our algorithm for learning mixtures of subcubes, given by Theorem 1.1. To learn D_T, by a Chernoff bound we can first estimate $\pi(b) := \Pr_{(x,b') \sim D_T}[b' = b] \geq 1/3$ for each $b \in \{0,1\}$ to within accuracy ϵ and confidence $1 - \alpha/3$ by drawing $O((1/\epsilon)^2 \log(1/\alpha))$ samples. We pick $b^* \in \{0,1\}$ for which $\Pr_{(x,b) \sim D_T}[b = b^*] \geq 1/3$ and denote our estimate for $\pi(b^*)$ by $\pi'(b^*)$.

By Lemma A.1, $\mathcal{D}$ is a mixture of k subcubes, so we can run A with error parameter $\epsilon/2$ and confidence parameter $\alpha/3$ on $\mathcal{D}$ and get a distribution $\mathcal{D}'$ for which $d_{\mathrm{TV}}(\mathcal{D}, \mathcal{D}') \leq \epsilon/4$. Our algorithm outputs the distribution D' given by $D'(x, b^*) = \pi'(b^*) \cdot \mathcal{D}(x)$ and $D'(x, 1 - b^*) = 1 - \pi'(b^*) \cdot \mathcal{D}(x)$.

Now because $D_T(x, b^*) = \pi_{b^*} \cdot \mathcal{D}(x)$, we have that

$$\sum_{x \in \{0,1\}^n} |D_T(x, b^*) - D'(x, b^*)| \leq \frac{\epsilon}{2} \sum_{x \in \{0,1\}^n} \mathcal{D}(x) + \pi'(b^*) \cdot \sum_{x \in \{0,1\}^n} |\mathcal{D}(x) - \mathcal{D}'(x)| \leq \frac{\epsilon}{2} + 2 \cdot \frac{\epsilon}{4} = \epsilon.$$

We thus also get that $\sum_{x \in \{0,1\}^n} |D_T(x, 1 - b^*) - D'(x, 1 - b^*)| = \sum_{x \in \{0,1\}^n} |D_T(x, b^*) - D'(x, b^*)| \leq \epsilon$, so $d_{\mathrm{TV}}(D_T, D') \leq O(\epsilon)$ as claimed. □

REFERENCES

[1] William Aiello and Milena Mihail. 1991. Learning the Fourier Spectrum of Probabilistic Lists and Trees.

[2] Baruch Awerbuch and Robert Kleinberg. 2008. Online linear optimization and adaptive routing. *J. Comput. System Sci.* 74, 1 (2008), 97–114.

[3] Avrim Blum. 1992. Rank-r decision trees are a subclass of r-decision lists. *Inform. Process. Lett.* 42, 4 (1992), 183–185.

[4] Avrim Blum, Merrick Furst, Jeffrey Jackson, Michael Kearns, Yishay Mansour, and Steven Rudich. 1994. Weakly learning DNF and characterizing statistical query learning using Fourier analysis. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*. ACM, 253–262.

[5] Anindya De, Ilias Diakonikolas, and Rocco A Servedio. 2014. Learning from satisfying assignments. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 478–497.

[6] François Denis. 1998. PAC learning from positive statistical queries. In *International Conference on Algorithmic Learning Theory*. Springer, 112–126.

[7] Ilias Diakonikolas, Daniel M Kane, and Alistair Stewart. 2016. Statistical query lower bounds for robust estimation of high-dimensional gaussians and gaussian mixtures. *arXiv preprint arXiv:1611.03473* (2016).

[8] Andrzej Ehrenfeucht and David Haussler. 1989. Learning decision trees from random examples. *Information and Computation* 82, 3 (1989), 231–246.

[9] Jon Feldman, Ryan ODonnell, and Rocco A Servedio. 2005. Learning mixtures of product distributions over discrete domains. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*. IEEE Computer Society, 501–510.

[10] Vitaly Feldman, Elena Grigorescu, Lev Reyzin, Santosh Vempala, and Ying Xiao. 2013. Statistical algorithms and a lower bound for detecting planted cliques. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*. ACM, 655–664.

[11] Yoav Freund and Yishay Mansour. 1999. Estimating a mixture of two product distributions. In *Proceedings of the twelfth annual conference on Computational learning theory*. ACM, 53–62.

[12] Elad Hazan, Adam Klivans, and Yang Yuan. 2017. Hyperparameter Optimization: A Spectral Approach. *arXiv preprint arXiv:1706.00764* (2017).

[13] Gordon B Hazen, James M Pellissier, and Jayavel Sounderpandian. 1998. Stochastic-tree models in medical decision making. *Interfaces* 28, 4 (1998), 64–80.

[14] Richard F Hespos and Paul A Strassmann. 1965. Stochastic decision trees for the analysis of investment decisions. *Management Science* 11, 10 (1965), B–244.

[15] Michael Kearns. 1998. Efficient noise-tolerant learning from statistical queries. *Journal of the ACM (JACM)* 45, 6 (1998), 983–1006.

[16] Michael Kearns, Yishay Mansour, Dana Ron, Ronitt Rubinfeld, Robert E Schapire, and Linda Sellie. 1994. On the learnability of discrete distributions. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*. ACM, 273–282.

[17] Fabien Letouzey, François Denis, and Rémi Gilleron. 2000. Learning from positive and unlabeled examples. In *International Conference on Algorithmic Learning Theory*. Springer, 71–85.

[18] Nathan Linial, Yishay Mansour, and Noam Nisan. 1993. Constant depth circuits, Fourier transform, and learnability. *Journal of the ACM (JACM)* 40, 3 (1993), 607–620.

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

[20] Elchanan Mossel, Ryan O'Donnell, and Rocco P Servedio. 2003. Learning juntas. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*. ACM, 206–212.

[21] Ronald L Rivest. 1987. Learning decision lists. *Machine learning* 2, 3 (1987), 229–246.

[22] Peter Stobbe and Andreas Krause. 2012. Learning fourier sparse set functions. In *Artificial Intelligence and Statistics*. 1125–1133.

[23] Gregory Valiant. 2012. Finding correlations in subquadratic time, with applications to learning parities and juntas. In *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*. IEEE, 11–20.