Abstract

Suppose we are given an \( n \)-dimensional order-3 symmetric tensor \( T \in (\mathbb{R}^n)^{\otimes 3} \) that is the sum of \( r \) random rank-1 terms. The problem of recovering the rank-1 components is possible in principle when \( r \leq n^4 \) but polynomial-time algorithms are only known in the regime \( r \ll n^{3/2} \). Similar “statistical-computational gaps” occur in many high-dimensional inference tasks, and in recent years there has been a flurry of work on explaining the apparent computational hardness in these problems by proving lower bounds against restricted (yet powerful) models of computation such as statistical queries (SQ), sum-of-squares (SoS), and low-degree polynomials (LDP). However, no such prior work exists for tensor decomposition, largely because its hardness does not appear to be explained by a “planted versus null” testing problem.

We consider a model for random order-3 tensor decomposition where one component is slightly larger in norm than the rest (to break symmetry), and the components are drawn uniformly from the hypercube. We resolve the computational complexity in the LDP model: \( O(\log n) \)-degree polynomial functions of the tensor entries can accurately estimate the largest component when \( r \ll n^{3/2} \) but fail to do so when \( r \gg n^{3/2} \). This provides rigorous evidence suggesting that the best known algorithms for tensor decomposition cannot be improved, at least by known approaches. A natural extension of the result holds for tensors of any fixed order \( k \geq 3 \), in which case the LDP threshold is \( r \sim n^{k/2} \).

1 INTRODUCTION

Tensor decomposition is a basic computational primitive underlying many algorithms for a variety of data analysis tasks, including phylogenetic reconstruction [78], topic modeling [3], community detection [2, 4, 57, 60], learning mixtures of Gaussians [7, 19, 44, 58], independent component analysis [47], dictionary learning [15, 73], and multi-reference alignment [80]. For further discussion, we refer the reader to [8, 75, 82, 87].

In this work we will consider order-\( k \) tensors of the form \( T \in (\mathbb{R}^n)^{\otimes k} \) for an integer \( k \geq 3 \), that is, \( T \) is an \( n \times \cdots \times n \) (\( k \) times) array of real numbers with entries denoted by \( T_{i_1,\ldots,i_k} \) with \( i_1,\ldots,i_k \in [n] := \{1, 2, \ldots, n\} \). A vector \( u = (u_i)_{i\in[n]} \in \mathbb{R}^n \) can be used to form a rank-1 tensor \( u^{\otimes k} \) defined by \( (u^{\otimes k})_{i_1,\ldots,i_k} = u_{i_1}\cdot u_{i_2}\cdot \cdots u_{i_k} \), akin to the rank-1 matrix \( uu^\top \).

In the tensor decomposition problem, we observe a rank-\( r \) tensor of the form

\[
T = \sum_{j=1}^r a_j^{\otimes k}
\]

with unknown components \( a_j \in \mathbb{R}^n \). The goal is to recover the components \( a_1,\ldots,a_r \) up to the inherent symmetries in the problem, i.e., we cannot recover the ordering of the list \( a_1,\ldots,a_r \) and if \( k \) is even we cannot distinguish \( a_j \) from \(-a_j \). Our regime of interest will be \( k \) fixed and \( n \) large, with \( r \) possibly growing with \( n \).

A large number of algorithmic results for tensor decomposition now exist under various assumptions on \( n, k, r \) and \( \{a_j\} \) [5, 6, 15, 19, 29, 33, 45–47, 54, 55, 61, 70, 73, 85]. We will focus on the order-3 case \( (k = 3) \) to simplify the discussion. If \( a_1,\ldots,a_r \) are linearly independent, the decomposition problem can be solved by a classical method based on simultaneous diagonalization\footnote{This method is sometimes called Jenrich’s algorithm, although this may be a misnomer; see [63].} [70]. Most recent work focuses on the more difficult overcomplete regime where \( r > n \). For random order-3 tensors, where the components \( a_j \) are drawn uniformly from the unit sphere, the state-of-the-art polynomial-time algorithms succeed when \( r \ll n^{3/2} \) (where \( \ll \) hides a polylogarithmic factor \( (\log n)^{O(1)} \)): this was achieved first by a method based on the sum-of-squares hierarchy [73] and then later by a faster spectral method [33]. Under the weaker condition \( r \leq n^2 \) (where \( \leq \) hides a constant factor), the decomposition is unique \([21]\) and so the problem is solvable in principle. However, no polynomial-time algorithm is known in the regime \( r \geq n^{3/2} \).

For random tensors of any fixed order \( k \geq 4 \), the situation is similar. The decomposition is unique when \( r \leq n^{k-1} \) [21], but polynomial-time algorithms are only known for substantially smaller rank. For \( k = 4 \), poly-time decomposition is known when \( r \ll n^2 \) [29, 73], and it is expected that \( r \ll n^{k/2} \) is achievable for general \( k \); however, to

CCS CONCEPTS

• Mathematics of computing → Probabilistic inference problems; • Theory of computation → Computational complexity and cryptography.

keywords

tensor decomposition, low-degree polynomials

ACM Reference Format:
Alexander S. Wein. 2023. Average-Case Complexity of Tensor Decomposition for Low-Degree Polynomials. In Proceedings of the 55th Annual ACM Symposium on Theory of Computing (STOC ’23), June 20–23, 2023, Orlando, FL, USA. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3564246.3585232

This work is licensed under a Creative Commons Attribution 4.0 International License.
our knowledge, the best poly-time algorithms in the literature for $k \geq 5$ require $r \leq n^{(k-1)/2}$ [19, 70]. These algorithms for $k \geq 4$ work not just for random tensors but also under much weaker assumptions on the components.

**Statistical-computational gaps.** The motivation for this work is to understand whether the “statistical-computational gap” mentioned above is inherent, that is, we hope to investigate whether or not a poly-time algorithm exists in the apparent “possible but hard” regime $n^{k/2} \leq r \leq n^{k-1}$. For context, gaps between statistical and computational thresholds are a nearly ubiquitous phenomenon throughout high-dimensional statistics, appearing in settings such as planted clique, sparse PCA, community detection, tensor PCA, and more; see e.g. [40, 92] for exposition. In these average-case problems where the input is random, classical computational complexity theory has little to say: results on NP-hardness (including those for tensor decomposition [49, 50]) typically show hardness for worst-case instances, which does not imply hardness in the average case. Still, a number of frameworks have emerged to justify why the “hard” regime is actually hard, and to predict the location of the hard regime in new problems. One approach is based on average-case reductions which formally transfer suspected hardness of one average-case problem (usually planted clique) to another (e.g. [18, 23, 24, 48]). Another approach is to prove unconditional lower bounds against particular algorithms or classes of algorithms (e.g. [30, 59, 69]), and sometimes this analysis is based on intricate geometric properties of the solution space (e.g. [1, 41, 42]; see [37] for a survey). Perhaps some of the most powerful and well-established classes of algorithms to prove lower bounds against are statistical query (SQ) algorithms (e.g. [32, 36]), the sum-of-squares (SoS) hierarchy (e.g. [14, 65]; see [83] for a survey), and low-degree polynomials (LDP) (e.g. [52, 53, 57]; see [68] for a survey). In recent years, all of the above frameworks have been widely successful at providing many different perspectives on what makes statistical problems easy versus hard, and there has also been progress on understanding formal connections between different frameworks [11, 25, 39, 53]. For many conjectured statistical-computational gaps, we now have sharp lower bounds in one (or more) of these frameworks, suggesting that the best known algorithms cannot be improved (at least by certain known approaches).

Despite all these successes, the tensor decomposition problem is a rare example where (prior to this work) essentially no progress has been made on any kind of average-case hardness, even though the problem itself and the suspected threshold at $r \sim n^{k/2}$ have maintained a high profile in the community since the work of [45] in 2015. The lecture notes [67] highlight “hardness of tensor decomposition” as one of 15 prominent open problems related to sum-of-squares (Open Problem 7.1). In Section 1.3.2 we explain why tensor decomposition is more difficult to analyze than the other statistical problems that were understood previously, and in Section 1.4.1 we explain how to overcome these challenges.

**The LDP framework.** Our main result (Theorem 1.4) determines the computational complexity of tensor decomposition in the low-degree polynomial (LDP) framework, establishing an easy-hard phase transition at the threshold $r \sim n^{k/2}$. This means we consider a restricted class of algorithms, namely those that can be expressed as $O(\log n)$-degree multivariate polynomials in the tensor entries. The study of this class of algorithms in the context of high-dimensional statistics first emerged from a line of work on the sum-of-squares (SoS) hierarchy [14, 52, 53, 57] and is now a popular tool for predicting and explaining statistical-computational gaps; see [68] for a survey. Low-degree polynomials capture various algorithmic paradigms such as spectral methods and approximate message passing (see e.g. Section 4 of [68], Appendix A of [39], and Theorem 1.4 of [77]), and so LDP lower bounds allow us to rule out a large class of known approaches all at once. For the types of problems that typically arise in high-dimensional statistics (informally speaking), the LDP framework has a great track record for consistently matching the performance of the best known poly-time algorithms. As a result, LDP lower bounds can be taken as evidence for inherent hardness of certain types of average-case problems. While there are some settings where LDPs are outperformed by another algorithm, these other algorithms tend to be “brittle” algebraic methods with almost no noise tolerance, and so the LDP framework is arguably still saying something meaningful in these settings; see [31, 51, 62, 66, 93] for discussion.

Existing LDP lower bounds apply to a wide variety of statistical tasks, which can be classified as hypothesis testing (e.g. [52, 53, 57]), estimation (e.g. [62, 86]), and optimization (e.g. [26, 39, 90]). The techniques used to prove these lower bounds are quite different in the three cases, and the current work introduces a new technique for the case of estimation.

### 1.1 Largest Component Recovery

In order to formulate the random tensor decomposition problem in the LDP framework, we define a variant of the problem where one component has slightly larger norm than the rest, and the goal is to recover this particular component.

**Problem 1.1 (Largest component recovery).** Let $k \geq 3$ be odd and $\delta > 0$. In the largest component recovery problem, we observe

$$T = (1 + \delta) a_1^{\otimes k} + \sum_{j=2}^r a_j^{\otimes k}$$

where each $a_j$ (for $1 \leq j \leq r$) is drawn uniformly and independently from the hypercube $\{\pm 1\}^n$. The goal is to recover $a_1$ with high probability.

One can think of $\delta$ as a small constant, although our results will apply more generally. The purpose of increasing the norm of the first component is to give the algorithm a concrete goal, namely to recover $a_1$; without this, the algorithm has no way to break symmetry among the components. We have restricted to odd $k$ here because otherwise there is no hope to disambiguate between $a_1$ and $-a_1$; however, we give similar results for even $k$ in Section 2. The assumption that $a_j$ is drawn from the hypercube helps to make some of the proofs cleaner, but our approach can likely handle other natural distributions for $a_j$ such as $N(0, I_n)$ (and we do not expect this to change the threshold).

**Connection to decomposition.** While Problem 1.1 does not have formal implications for the canonical random tensor decomposition model (i.e., (1) with $a_j$ uniform on the sphere), it does have implications for a mild generalization of this problem.
Problem 1.2 (Semirandom tensor decomposition). Let \( k \geq 3 \) be odd and \( \delta > 0 \). In the semirandom tensor decomposition problem, we observe

\[
T = \sum_{j=1}^{r} \lambda_j a_j^{\otimes k}
\]

where \( \lambda_j \in [1, 1+\delta] \) are arbitrary and \( a_j \) are drawn uniformly and independently from the hypercube \((\pm 1)^n\). The goal is to recover \( a_1, \ldots, a_r \) (up to re-ordering) as well as the corresponding \( \lambda_j \)'s, with high probability.

Note that any algorithm for Problem 1.2 can be used to solve Problem 1.1 (with the same parameters \( k, n, r, \delta \)); simply run the algorithm to recover all the \((\lambda_j, a_j)\) pairs and then output the \( a_j \) with the largest corresponding \( \lambda_j \). When \( r \gg n^{k/2} \), our main result (Theorem 1.4) will give rigorous evidence for hardness of Problem 1.2 via a two-step argument: we will show LDP-hardness of Problem 1.1, justifying the conjecture that there is no poly-time algorithm for Problem 1.1; this conjecture, if true, formally implies that there is no poly-time algorithm for Problem 1.2.

We see Problem 1.2 as a natural variant of random tensor decomposition. In particular, we expect that existing algorithms \([33, 73]\) can be readily adapted to solve this variant when \( r \ll n^{k/2} \), at least for \( k \in (3, 4) \). As such, understanding the complexity of Problem 1.2 (via the connection to Problem 1.1 described above) can be viewed as the end-goal of this work. From this point onward, we will study Problem 1.1. In Section 1.5 we discuss possible extensions of our result that would more directly address the canonical model for random tensor decomposition.

Remark 1.3. In line with the discussion in the Introduction, our results should really only be considered evidence for hardness of Problems 1.1 and 1.2 in the presence of noise. Indeed, if \( \delta \) is not an integer, there is a trivial algorithm for Problem 1.1 by examining the non-integer part of \( T \) (the author thanks Bruce Hajek for pointing this out). This algorithm is easily defeated by adding Gaussian noise of variance \( \sigma^2 \gg 1 \) to each entry of \( T \). This is a low level of noise, as we believe our low-degree upper bound tolerates \( \sigma \ll n^{k/4} \), akin to tensor PCA.

1.2 Main Results

We now show how to cast Problem 1.1 in the LDP framework and state our main results. The class of algorithms we consider are (multivariate) polynomials \( f \) of degree (at most) \( D \) with real coefficients, whose input variables are the \( n^k \) entries of \( T \in (\mathbb{R}^n)^{\otimes k} \) and whose output is a real scalar value; we write \( \mathbb{E}[T] \leq D \) for the space of such polynomials. We will be interested in whether such a polynomial can accurately estimate \( a_{11} := (a_1)_1 \), the first entry of the vector \( a_1 \). Following \([86]\), define the degree-\( D \) minimum mean squared error

\[
\text{MMSE}_{\leq D} := \inf_{f \in \mathbb{E}[T] \leq D} \mathbb{E}[(f(T) - a_{11})^2]
\]

where the expectation is over \( \{a_j\} \) and \( T \) sampled as in Problem 1.1.

Note that the above “scalar MMSE” is directly related to the “vector MMSE”: by linearity of expectation and symmetry among the coordinates,

\[
\inf_{f_i, \ldots, f_n \in \mathbb{E}[T] \leq D} \sum_{i=1}^{n} (f_i(T) - (a_1)_1)^2 = n \cdot \text{MMSE}_{\leq D}.
\]

It therefore suffices to study the scalar MMSE, with \( \text{MMSE}_{\leq D} \to 0 \) indicating near-perfect estimation and \( \text{MMSE}_{\leq D} \to 1 \) indicating near-trivial estimation (no better than the constant function \( f(T) \equiv 0 \)).

Theorem 1.4. Fix any \( k \geq 3 \) odd and \( \delta > 0 \). As in Problem 1.1, let

\[
T = (1+\delta)a_1^{\otimes k} + \sum_{j=2}^{r} a_j^{\otimes k}
\]

where each \( a_j \) (for \( 1 \leq j \leq r \)) is drawn uniformly and independently from the hypercube \((\pm 1)^n\). Let \( r = n^\alpha \) grow with \( n = \Theta(n^\alpha) \) for an arbitrary fixed \( \alpha > 0 \).

- ("Easy") If \( \alpha < k/2 \) then \( \lim_{n \to \infty} \text{MMSE}_{\leq C \log n} = 0 \) for a constant \( C = C(k, \delta) > 0 \).
- ("Hard") If \( \alpha > k/2 \) then \( \lim_{n \to \infty} \text{MMSE}_{\leq n^{\alpha}} = 1 \) for a constant \( c = c(k, \alpha) > 0 \).

More precise and more general statements of the lower bound ("hard" regime) and upper bound ("easy" regime) can be found in Section 2 (Theorems 2.1 and 2.2), where the case of \( k \) even is also handled. The lower bound shows failure of LDP algorithms when \( r \gg n^{k/2} \). Since no one has managed to find a poly-time algorithm in the LDP-hard regime for many other problems of a similar nature (including planted clique \([52]\), community detection \([57]\), tensor PCA \([53, 68]\), and more), this suggests that there is no poly-time algorithm for Problem 1.1 (and therefore also for Problem 1.2) when \( r \gg n^{k/2} \), at least without a new algorithmic breakthrough. This is the first average-case hardness result of any type for tensor decomposition, and it gives a concrete reason to believe that the existing algorithms for \( k = 3 \) (which succeed when \( r \ll n^{k/2} \) \([33, 73]\)) are essentially optimal. Following the heuristic correspondence between polynomial degree and runtime in Hypothesis 2.1.5 of \([52]\) (see also \([34, 68]\))—namely, degree \( D \) corresponds to runtime \( \exp(\Theta(D)) \) where \( \Theta \) hides factors of \( \log n \)—the lower bound suggests that runtime \( \exp(n^{O(1)}) \) is required in the hard regime.

The upper bound shows that a degree-\( O(\log n) \) polynomial successfully estimates \( a_1 \) in the regime \( r \ll n^{k/2} \). Such a polynomial has \( n^{O(\log n)} \) terms and can thus be evaluated in time \( n^{O(\log n)} \) (quasi-polynomial). This is in stark contrast to the \( \exp(n^{O(1)}) \) runtime that we expect to need in the hard regime. We are confident that for \( k \in (3, 4) \) and \( r \ll n^{k/2} \), Problems 1.1 and 1.2 can in fact be solved in polynomial time by adapting existing algorithms \([33, 73]\), but we have not attempted this here as our main focus is on establishing the phase transition for polynomials. We consider the lower bound to be our main contribution, and the upper bound serves to make the lower bound more meaningful.

1.3 Discussion

1.3.1 Why the LDP Framework? Since there are many different frameworks for exploring statistical-computational gaps, we briefly...
discuss the possibility of applying some of the others to tensor decomposition, and explain why the LDP framework seems especially suited to this task.

Statistical query (SQ) model. The SQ model (e.g. \cite{32,36}) is applicable only to settings where the observation consists of i.i.d. samples from some unknown distribution. There does not seem to be a clear way to cast tensor decomposition in this form.

Sum-of-squares (SoS) hierarchy. SoS is a powerful algorithmic tool, and SoS lower bounds (e.g. \cite{14,65,83}) are sometimes viewed as a gold standard in average-case hardness. However, it is important to note that SoS lower bounds show hardness of certification problems, whereas Problems 1.1 and 1.2 are recovery problems. Hardness of certification does not necessarily imply hardness of recovery, as discussed in \cite{9,12,13}. Therefore, while there are some natural certification problems related to tensor decomposition that would be nice to have SoS lower bounds for (e.g. Open Problem 7.2 in \cite{67}), it is not clear how to use an SoS lower bound to directly address the decomposition problem.

Optimization landscape. Prior work \cite{6,46} has studied the landscape of a certain non-convex optimization problem related to tensor decomposition, leading to some algorithmic results. In other settings, recent work has studied structural properties of optimization problems as a means to prove failure of certain algorithms (e.g., local search, Markov chains, and gradient descent \cite{16,17,38,42}) to recover a planted structure. If results of this type were obtained for tensor decomposition, they would complement ours by ruling out different types of algorithms. One caveat is that recent work on tensor PCA \cite{16,20,74,84,91} and planted clique \cite{27,43} has revealed that the choice of which function to optimize can be very important. Thus, "hardness" of one particular canonical optimization landscape need not indicate inherent hardness of the recovery problem.

Reductions. Reductions from planted clique (e.g. \cite{18,23,24,48}) are among the strongest forms of average-case hardness that exist for statistical problems, and it would be great to have reduction-based evidence for hardness of tensor decomposition. This would likely require new ideas beyond those in the current literature, as Problem 1.1 is somewhat different in structure from planted clique and the problems it has been reduced to thus far. These differences are explained in Section 1.3.2 below.

Summary. In summary, there are many complementary approaches that could be explored in future work in order to give new perspectives on hardness of tensor decomposition. In this work we have chosen to pursue the LDP framework, some strengths of which include its stellar track record of predicting statistical-computational gaps in the past, and its flexibility to directly address the decomposition problem (Problem 1.2) rather than a related certification or optimization problem.

1.3.2 Difficulties of Tensor Decomposition. Although the suspected threshold $r \sim n^{k/2}$ for random tensor decomposition has been around for some time now \cite{45,73}, and despite the many recent successes in computational lower bounds for other statistical problems, there are (prior to this work) effectively no results that point to whether $r \sim n^{k/2}$ is a fundamental barrier or not. Here we will discuss what makes tensor decomposition more difficult to analyze than the other statistical-computational gaps that have received recent attention in the literature.

As a point of comparison, we briefly introduce the tensor principal component analysis (tensor PCA) problem \cite{84}: here we observe an order-$k$ tensor $Y = \lambda x_{\otimes k} + Z$ where $\lambda > 0$ is a signal-to-noise ratio, $x \in \mathbb{R}^n$ is drawn uniformly from the unit sphere, and $Z \in (\mathbb{R}^n)^{\otimes k}$ is a tensor with i.i.d. $N(0,1)$ entries; the goal is to (approximately) recover the vector $x$ (up to sign, if $k$ is even). Tensor PCA also has a statistical-computational gap, and it is now well-understood: the best known poly-time algorithms require $\lambda \gtrsim n^{k/4}$, and there are matching lower bounds in both the SoS and LDP frameworks \cite{53,56,68,81}. While tensor decomposition and tensor PCA may look superficially similar, tensor decomposition is much harder to analyze for the reason described below.

With a few exceptions (\cite{62,64,86}), essentially all existing lower bounds for recovering a planted signal in the SQ, SoS, or LDP frameworks leverage hardness of testing between a "planted" distribution and a "null" distribution, where the null distribution has independent entries. In the case of tensor PCA, in order to show hardness of recovering $x$ when $\lambda \ll n^{k/4}$, the SoS and LDP lower bounds crucially leverage the fact that it is hard to even detect the planted structure when $\lambda \ll n^{k/4}$, that is, it is hard to distinguish between a spiked tensor $Y$ and an i.i.d. Gaussian tensor $Z$. Now for the decomposition problem, we are hoping to show hardness of decomposition whenever $r \gg n^{k/2}$, but the problem of distinguishing between a random rank-$r$ tensor and the appropriate Gaussian tensor$^2$ is actually easy when $r \ll n^{k/2}$; this can be achieved by thresholding a particular degree-$4$ polynomial in the tensor entries, where each monomial forms a double cover of $2k$ elements of $[n]$. This "detection-recovery gap" creates difficulty because the standard tools for proving lower bounds cannot show hardness of recovery unless detection is also hard. The tools with this limitation include the SDA (statistical dimension on average) approach for SQ \cite{36}, the pseudo-calibration approach for SoS \cite{14}, and the low-degree likelihood ratio \cite{52,53,57,68} (as well as its conditional variants \cite{11,28}) for LDP. Reductions from planted clique also typically require hardness of detection \cite{24}.

The method of \cite{86} overcomes this challenge in some settings and gives LDP lower bounds for recovery problems even in regimes where detection is easy. This methods applies to problems of the form (roughly speaking) "signal plus noise" where the noise has independent entries. For instance, this gives LDP-hardness of recovery for variants of tensor PCA with detection-recovery gaps \cite{72}. However, tensor decomposition does not take this "signal plus noise" form: Problem 1.1 has a "signal" term $(1 + \delta)a_{1\otimes k}$ but the remaining "noise" term is a rank-$(r-1)$ tensor, which does not have independent entries. As a result, we will need to introduce a new method in order to prove a lower bound for tensor decomposition. The proof strategy is outlined in Section 1.4.1 below.

---

$^2$The Gaussian tensor matches moments of order 1 and 2 with the rank-$r$ tensor. (One could choose a different null distribution, but the analysis may be difficult; for instance, see Section 1.5.)
1.4 Proof Techniques

1.4.1 Lower Bound. Instead of mean squared error it will be convenient to work with an equivalent quantity, the degree-$D$ maximum correlation

$$\text{Corr}_{\leq D} := \sup_{f \in \{T\}_{\leq D}} \frac{\mathbb{E}[f(T) \cdot a_{11}]}{\sqrt{\mathbb{E}[f(T)^2]}}.$$ 

This is directly related to MMSE$_{\leq D}$ via the identity MMSE$_{\leq D} = \mathbb{E}[\sigma^2_{11}] - \text{Corr}^2_{\leq D} = 1 - \text{Corr}^2_{\leq D}$ (see [86, Fact 1.1]), so our objective is to show Corr$_{\leq D} = o(1)$ when $r \gg n^{3/2}$ and $D = n^{2(1)}$.

A first attempt is to compute Corr$_{\leq D}$ explicitly. Expand an arbitrary degree-$D$ polynomial in the monomial basis: $f(T) = \sum_{|S| \leq D} \hat{f}_S \mathcal{T}^S$, where $S$ ranges over multi-sets of tensor entries and $\mathcal{T}^S := \prod_{i \in S} T_i$. The numerator of Corr$_{\leq D}$ is a linear functional of the coefficient vector $\hat{f} = (\hat{f}_S)_{|S| \leq D}$:

$$\mathbb{E}[f(T) \cdot a_{11}] = \sum_S \hat{f}_S \mathbb{E}[\mathcal{T}^S \cdot a_{11}] = c^T \hat{f} \quad \text{where } c_S := \mathbb{E}[\mathcal{T}^S \cdot a_{11}].$$

For the denominator, $\mathbb{E}[f(T)^2]$ is a quadratic form:

$$\mathbb{E}[f(T)^2] = \sum_{S,S'} \hat{f}_S \hat{f}_{S'} \mathbb{E}[\mathcal{T}^S \cdot \mathcal{T}^{S'}] = \hat{f}^T \hat{f} \quad \text{where } P_{S,S'} := \mathbb{E}[\mathcal{T}^S \cdot \mathcal{T}^{S'}].$$

This means we have the explicit formula

$$\text{Corr}_{\leq D} = \sup_f \frac{c^T \hat{f}}{\sqrt{\hat{f}^T P \hat{f}}},$$

where

$$c := \sqrt{\mathbb{E}[f(T)^2]} \frac{\mathbb{E}[f(T) \cdot a_{11}]}{c^T \hat{f}}.$$ 

The difficulty here is that, while we can write down an explicit formula for the vector $c$ and matrix $P$, it does not seem tractable to write down an explicit formula for the inverse matrix $P^{-1}$. We will instead find an upper bound on Corr$_{\leq D}$ that is manageable to work with.

Our difficulties stem from the fact that we do not have an orthogonal basis of polynomials to work with. After all, if $\{T^S\}_{|S| \leq D}$ were orthogonal polynomials—i.e., if $P$ were a diagonal matrix—then we would have no problem inverting $P$. Since the distribution of $T$ is complicated (in particular, it is not a product measure), it seems difficult to construct an explicit basis of orthogonal polynomials. Instead, we will think of $f(T)$ as a function of the underlying i.i.d. Rademacher random variables $A = (a_{ij}) := (a_{ij})_i$ and work with an orthogonal basis of polynomials in those variables. For any $f(T)$ there is an associated polynomial $g(A)$ such that $f(T) = g(A)$, and we can expand these as

$$\sum_{|S| \leq D} \hat{f}_S \mathcal{T}^S = f(T) = g(A) = \sum_{U \subseteq \mathcal{U}} \hat{g}_U A^U,$$

where $\mathcal{U} \subseteq [n] \times [r]$, $A^U := \prod_{i,j \in U} a_{ij}$, and $\hat{g}_U = (\hat{g}_U)_{|U| \leq D}$ is some vector of coefficients. Here $\{A^U\}$ is the standard Fourier basis for functions on the hypercube, which crucially does have the desired orthogonality property: $\mathbb{E}[A^U A^{U'}] = \mathbf{1}_{U = U'}$. As a result, we can express the denominator of Corr$_{\leq D}$ as simply

$$\mathbb{E}[f(T)^2] = \mathbb{E}[g(A)^2] = \|g\|^2.$$ 

To exploit this, we will need to understand the relation between $f$ and the corresponding $\hat{g}$. We will write down an explicit linear transformation, i.e., a matrix $M$ such that $\hat{g} = M \hat{f}$. The crux of our new strategy is that to obtain an upper bound on Corr$_{\leq D}$ it suffices to construct a left-inverse for $M$, i.e., a matrix $M^*$ such that $M^* M = I$. To see why this suffices,

$$\text{Corr}_{\leq D} = \sup_f \frac{c^T \hat{f}}{\|M \hat{f}\|} = \sup_f \frac{c^T M^* M \hat{f}}{\|M \hat{f}\|} \leq \sup_h \frac{c^T M^* h}{\|h\|} = \|c^T M^*\|,$$

where in the inequality step, $h$ plays the role of $\hat{g} = M \hat{f}$ except we have relaxed the problem to allow $h$ to be any vector, not necessarily in the image of $M$.

In light of the above, our remaining task is to construct an explicit left-inverse $M^*$. There are many possible choices, some of which will yield better bounds on Corr$_{\leq D}$ than others. We will need one that yields a good bound but is also analytically tractable to write down explicitly. We now explain the intuition behind our construction for $M^*$. Note that the left-inverse property is equivalent to $M^* \hat{g} = f$ whenever $\hat{g} = M \hat{f}$. That is, $M^*$ is a procedure to recover the (coefficients of the) polynomial $f$ when given the (coefficients of the) polynomial $g$ that satisfies $f(T) = g(A)$. Luckily there is a methodical process for this task which starts from the highest-degree terms and works downward. At each stage, there is always a particular monomial in $A$ whose coefficient in $g$ allows us to immediately deduce some particular coefficient in $f$. To illustrate, if $k = 3$ and $D = 2$ then the coefficient of the monomial

$$a_2a_3a_4a_7a_8a_9a_{35}$$

in $g(A)$ allows us to immediately deduce the coefficient of the monomial $T_{234}T_{235}$ in $f(T)$, since this is the only monomial in $T$ (of degree at most 2) whose expansion in terms of $A$ contains the term (3). Once we have deduced the highest-degree coefficients of $f$, we can subtract off the corresponding terms in $g$ and repeat the process recursively. Our choice of $M^*$ is inspired by this process; the full details can be found in Section 3.4.3.

Recall now that the final step is to bound $\|c^T M^*\|$ for our choice of $M^*$. Due to the recursive structure of $M^*$, this boils down to analyzing certain recursively-defined values $\omega_k$ (see Section 3.6). As above, $S$ is a multi-set of tensor entries, which can be thought of as a hypergraph on vertex set $[n]$ (with a hyperedge for each tensor entry). The value $\omega_k$ is computed by summing over all possible ways to reduce $S$ to a smaller hypergraph by replacing some collection of hyperedges by a new hyperedge equal to their symmetric difference. Analyzing this recurrence is the most technologically involved part of the proof, since there are subtle cancellations that need to be understood in order to get the right bound.

1.4.2 Upper Bound. To construct a polynomial that accurately estimates the largest component, we take inspiration from a line of algorithmic work that builds spectral methods from tensor networks [33, 55, 76, 79]. A tensor network is a diagram that specifies how to “multiply” together a collection of tensors to form another tensor; see [76] for details. Some existing algorithms for order-3 tensor decomposition [33, 55] (including the fastest known method that reaches the threshold $r \sim n^{3/2}$ [33]) are based on the idea of building a tensor network from multiple copies of the input tensor, flattening the result to a matrix, and running a spectral method on this matrix. Morally speaking, these are all steps that one should be able to capture using low-degree polynomials, but the algorithm of [33] is actually a multi-stage process that seems challenging to write as a single polynomial. Also, since our goal is to estimate the
largest component, we do not need to inject randomness to break symmetry among the components as in [33].

To construct our polynomial, we multiply $O(\log n)$ copies of $T$ together in a tensor network whose shape is an expander graph. The output is a single vector, which we take as our estimate for $a_1$. There is one key trick we use to greatly simplify the analysis: we deviate slightly from the standard notion of a tensor network by disallowing repeated “edge labels.” The precise definition of the polynomial appears in the full version of this paper [89].

1.5 Future Directions

We collect here a list of possible extensions of our main result and related open problems.

- **Distribution of $a_j$:** We have taken $a_j$ drawn uniformly from the hypercube in the interest of keeping the proofs as clean as possible. We expect our approach could be adapted to other distributions for $a_j$ such as $\mathcal{N}(0, I_n)$.

- **Canonical model:** While we feel that the semirandom tensor decomposition model (Problem 1.2) is quite natural, there may be interest in showing hardness for the more canonical model where every $\lambda_j$ is equal to 1. One way to approach this could be to study the task of hypothesis testing between a random rank-$r$ tensor and a random rank-$(r+1)$ tensor. Hardness (for poly-time algorithms) of this testing problem implies hardness of decomposition in the canonical model. It may be possible to use our proof technique to show LDP-hardness of this testing problem. Another alternative approach could be to consider a variant of Problem 1.1 where symmetry between the tensor components is broken by giving the algorithm a small amount of side-information rather than increasing the norm of one component.

- **Tensors with algebraic structure:** Hopefully the proof techniques introduced here will open the door to other hardness results for related problems. For instance, *orbit recovery problems*—a class of statistical problems involving group actions [10, 35]—give rise to variants of tensor decomposition with algebraic structure. One example is *multi-reference alignment (MRA)*, where the tensor components are cyclic shifts of one another [80]. A statistical-computational gap in heterogeneous MRA was conjectured in [22]; the positive side was resolved in [88] using [73], and the negative side should now be approachable using our techniques. Tensors with continuous group symmetry are even less well understood [71, 76].

- **Smoothed order-3 tensor decomposition:** For $k = 3$, the algorithms achieving the optimal condition $r \ll n^{3/2}$ seem to crucially exploit randomness in the components [33, 73]. In contrast, other algorithms succeed in the smoothed analysis model (which imposes minimal assumptions on the components) but require $r \ll n$ [19, 47]. It remains unclear whether better algorithms for the smoothed model exist, or whether there is an inherent gap between the random and smoothed settings. For $k = 4$ there is no such gap, with both random and smoothed algorithms achieving $r \sim n^2$ [29, 73].

- **Other LDP estimation lower bounds:** Proving LDP lower bounds for estimation problems remains a difficult task in many settings, with relatively few tools available compared to detection (hypothesis testing) problems. A few open problems are to resolve the LDP recovery threshold for sparse linear regression and group testing; these problems have detection-recovery gaps and the LDP detection thresholds were resolved in [11, 28].

2 GENERAL SETTING AND RESULTS

It will be convenient to generalize Problem 1.1 in a few different ways. First, we allow all the components to have potentially different norms. This way, there is nothing distinguished about the first component, eliminating some tedious casework. To this end, we will consider tensors of the form

$$\sum_{j=1}^r \lambda_j a_j^{\otimes k}$$

where each component $a_j$ is drawn uniformly and independently from the hypercube $\{\pm1\}^n$, and $\lambda_j \in \mathbb{R}$ are deterministic nonzero scalars that are known to the algorithm. Without loss of generality we assume $1 = \lambda_1 \geq |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_r| =: \lambda_{\text{min}} > 0$. The goal is to recover the largest component $a_j$. The case $\lambda_2 = \lambda_3 = \cdots = \lambda_r = (1+\delta)^{-1}$ is equivalent to Problem 1.1.

Second, we will potentially allow the algorithm access not just to an order-$k$ tensor but also to tensors of lower order. For $I \subseteq [n]$ write

$$T_I := \sum_{j=1}^r \lambda_j a_j^I$$

with $\lambda_j$ and $a_j$ as above, and where for a vector $v \in \mathbb{R}^n$, $v^I := \prod_{i \in I} v_i$. Note that since the $a_j$ have $\{\pm1\}$-valued entries, each entry of the tensor (4) takes the form $T_I$ for some $0 \leq |I| \leq k$.

Let $\Omega$ be a collection of subsets of $[n]$. We consider the task of estimating $a_{11} := (a_1)_1$ using a degree-$D$ polynomial $f : \Omega^\mathbb{R} \rightarrow \mathbb{R}$ whose input variables are $\{T_I\}_{I \subseteq \Omega}$. Accordingly, we define

$$\text{MMSE}_{ED}^{\Omega} := \inf_{f : \mathbb{R}^{\sum_{|I|=D} T_I} \rightarrow \mathbb{R}} \mathbb{E}[(f(T) - a_{11})^2].$$

To make our lower bound for order-$k$ tensors as strong as possible, we will choose $\Omega = \{I \subseteq [n] : 0 < |I| \leq k\}$. This is equivalent to giving the algorithm access to all the tensors

$$\sum_{j=1}^r \lambda_j a_j, \sum_{j=1}^r \lambda_j a_j^{\otimes 2}, \ldots, \sum_{j=1}^r \lambda_j a_j^{\otimes k},$$

(which is a situation that often arises when using the method of moments, e.g. [58, 80]). Note that a lower bound in this setting implies a lower bound in the original setting where only the order-$k$ tensor is revealed.

To make our upper bound as strong as possible, we will give the algorithm access to minimal information. For $k$ odd, we will take $\Omega = \{I \subseteq [n] : |I| = k\}$, that is, our polynomial only needs access to the “off-diagonal” entries of the order-$k$ tensor ($T_{i_1, i_1, \ldots, i_k}$ where $i_1, \ldots, i_k$ are all distinct). For $k$ even, the order-$k$ tensor alone does not suffice to disambiguate between $a_1$ and $-a_1$, so we additionally reveal the (off-diagonal part of the) order-$(k-1)$ tensor, that is, we take $\Omega = \{I \subseteq [n] : k - 1 \leq |I| \leq k\}$. (An alternative formulation for $k$ even could be to reveal only the order-$k$ tensor and ask the
algorithm to estimate \((a_1) \cdot (a_1)\); we expect this could be analyzed using our methods.

The following are non-asymptotic statements of our main results. Together, these immediately imply Theorem 1.4.

**Theorem 2.1 (Lower bound).** Consider the setting described above with any parameters \(k' \geq 3, n \geq 1, D \geq 0\), and set \(\Omega := \{I \subseteq [n] : 0 \leq |I| \leq k\}\). If
\[
\mathbb{E} f(T) - (a_1) \leq \frac{1}{2} \log n / |I|^2.
\]
then
\[
\text{MMSE}_{\leq D}^{\Omega} \leq 1 - n^{-1/2}. \tag{6}
\]

In particular, if \(k' \geq 3\) is fixed, \(\lambda_{\min} \geq \delta\) for fixed \(\delta > 0\), and \(r = r_n\) grows as \(r = \Theta(n^\alpha)\) for fixed \(\alpha > k'/2\), then \(\lim_{n \to \infty} \text{MMSE}_{\leq D}^{\Omega} = 1\) for a constant \(c = c(k, \alpha) > 0\).

**Theorem 2.2 (Upper bound).** Consider the setting described above with any \(k' \geq 3\). Let \(k' \in \{k - 1, k\}\) be odd, and set \(\Omega := \{I \subseteq [n] : k' \leq |I| \leq k\}\). Suppose \(n \geq n_0\) for some constant \(n_0 = n_0(k)\). Let \(D\) be the smallest odd integer such that
\[
D \geq k \log n / 1 - |a_2|. \tag{7}
\]
If \(\lambda_{|I|} \leq 1 - n^{-1/2}\) and
\[
\mathbb{E} f(T) - (a_1) \leq \frac{1}{2} k^{-1} D^{-\frac{1}{2}} \cdot 2^k k n^{-1/2}. \tag{8}
\]
then
\[
\text{MMSE}_{\leq D}^{\Omega} \leq 10 k^{-1} D^{5/2k} \frac{r}{n^{-1/2} - 1_k} \leq 10 k^{-1} D^{5/2k} \frac{r}{n^{-1/2}}. \tag{9}
\]
In particular, if \(k' \geq 3\) is fixed, \(\lambda_{|I|} \leq 1 - \Omega(|I|)\) for fixed \(\Omega = \{I \subseteq [n] : |I| \leq k\}\), and \(r = r_n\) grows as \(r = \Theta(n^\alpha)\) for fixed \(0 < \alpha < k'/2\), then \(\lim_{n \to \infty} \text{MMSE}_{\leq D}^{\Omega} = 0\) for a constant \(C = C(k, \delta) > 0\).

Note that (7) is the bottleneck, requiring \(r \ll n^{k/2}\). The intermediate step in (9) is a sharper bound on the MMSE that we use for the remark below, but it does not dictate the threshold for \(r\).

**Remark 2.3 (Exact recovery).** We remark that if \(\text{MMSE}_{\leq D}^{\Omega} = o(1/n)\) then exact recovery of \(a_1\) is possible with probability \(1 - o(1)\) by thresholding the values of certain degree-\(D\) polynomials \(f_1, \ldots, f_m\). To see this: combining (2) with Markov's inequality, we have \(f_1, \ldots, f_m\) such that \(\sum_{i=1}^m (f_i(T) - (a_1)_i)^2 < 1\) with probability \(1 - o(1)\). Furthermore, thresholding \(f_1, \ldots, f_m\) is guaranteed to exactly recover \(a_1\) whenever \(\sum_{i=1}^m (f_i(T) - (a_1)_i)^2 < 1\). As a result, when \(|\lambda_{|I|} | \leq 1 - \Omega(|I|)\) and \(r \leq n^{k/2}/(\log n)^{O(1)}\), our upper bound (Theorem 2.2) gives exact recovery for any fixed \(k' \geq 5\). We expect a similar result to hold for \(k' \in \{3, 4\}\) but this may require modifying the construction of the polynomial in the proof of Theorem 2.2.

**Organization.** The proof of Theorem 2.1 is given in Section 3. The proof of Theorem 2.2 is deferred to the full version of this paper [89].

### 3 PROOF OF THEOREM 2.1: LOWER BOUND

#### 3.1 Setup

Throughout, define \(\Omega := \{I \subseteq [n] : 0 \leq |I| \leq k\}\). Our goal will be to give an upper bound on
\[
\mathbb{E} f(T) - (a_1) = \mathbb{E} f(T) - (a_1)_1 = \mathbb{E}[f(T) - (a_1)_1].
\]
This will imply the desired result due to the following direct relation between \(\text{Corr}_{\leq D}\) and the associated MMSE.

**Fact 3.1 ([86, Fact 1.1]).** \(\text{MMSE}_{\leq D}^{\Omega} = \mathbb{E} f(T) - (a_1)_1 - \text{Corr}_{\leq D}^2 > 1 - \text{Corr}_{\leq D}^2\).

Any degree-\(D\) polynomial \(f : \mathbb{R}^\Omega \to \mathbb{R}\) admits an expansion of the form
\[
f(T) = \sum_{0 \leq |S| \leq D} f_S T^S
\]
for some coefficients \(f_S \in \mathbb{R}\), where \(S\) takes values \(S \in \mathbb{R}^\Omega := \{0, 1, 2, \ldots, D\}^{\lfloor |\Omega| / 2 \rfloor}\) with \(|S| := \sum_{s \in S} T^s\) and \(T^S := \prod_{s \in S} T^s\).

At the same time, \(T\) can be thought of as a function of \(A\), the \(n \times r\) matrix with columns \(a_j\). This means we can also expand \(f\) as
\[
f(T) = g(A) = \sum_U \hat{g}_U A^U
\]
where \(U\) ranges over subsets \(U \subseteq [n] \times [r]\) of cardinality \(|U| \leq kD\). This expansion will be useful because, since \(A\) has i.i.d. Rademacher entries, \(\{A^U\}\) forms an orthonormal basis in the sense \(E[A^U A^U'] = \mathbb{1}_{U=U'}\). As a consequence,
\[
\mathbb{E} f(T)^2 = \mathbb{E} g(A)^2 = \|g\|^2 := \sum_U \hat{g}_U^2. \tag{9}
\]
Any vector of coefficients \(\hat{f} := (\hat{f}_S)_{|S| \leq D}\) induces a unique choice of \(\hat{g} = (\hat{g}_U)_{|U| \leq kD}\) such that
\[
\sum_{|S| \leq D} f_S T^S = f(T) = g(A) = \sum_{|U| \leq kD} \hat{g}_U A^U. \tag{10}
\]
It will be important to understand this mapping from \(\hat{f}\) to \(\hat{g}\).

#### 3.2 Proof Overview

We will show that the mapping from \(\hat{f}\) to \(\hat{g}\) in (10) is a linear transformation that takes the form \(\hat{g} = M\hat{f}\) for an explicit matrix \(M = (M_{d,S})\). A key step in the proof will be to construct an explicit left inverse for \(M\), that is, a matrix \(M^+\) satisfying \(M^+ M = I\).

In other words, \(M^+\) is a matrix that recovers the coefficients \(\hat{f}\) from the coefficients \(\hat{g}\): \(M^+ \hat{g} = M^+ M\hat{f} = \hat{f}\).

The numerator of \(\text{Corr}_{\leq D}\) can be expressed as
\[
\mathbb{E} f(a_{11}) = \sum_{|S| \leq D} \hat{f}_S \mathbb{E}[T^S \cdot a_{11}] = c^T \hat{f}
\]
where the vector \(c = (c_S)_{|S| \leq D}\) is defined by
\[
c_S = \mathbb{E}[T^S \cdot a_{11}]. \tag{11}
\]
Using (9), the denominator can be expressed as \( \sqrt{\mathbb{E}[f(T)^2]} = \|\hat{g}\| = \|M\hat{f}\| \). This means we can write

\[
\text{Corr}_{\leq D} = \sup_{\hat{f}} \frac{c^T\hat{f}}{\|M\hat{f}\|} = \sup_{\hat{f}} \frac{c^TM^*\hat{f}}{\|M^*\hat{f}\|} \leq \sup_{h} \frac{c^TM^*h}{\|h\|} = \|c^TM^*\|. 
\]

(12)

In the crucial inequality above, \( h \) plays the role of \( \hat{g} = M\hat{f} \) except we have relaxed the problem to allow \( h \) to be any vector, not necessarily in the image of \( M \). After this simplification, the optimizer for \( h \) is \( h^* = (c^TM^*)^T \) (or any scalar multiple thereof), yielding the explicit expression \( \|c^TM^*\| \) for the optimum. So long as we can construct a left inverse \( M^* \), this gives an upper bound on \( \text{Corr}_{\leq D} \). While many choices for \( M^* \) are possible, we will need to find one that (i) is simple enough to work with explicitly, and (ii) results in a good bound on \( \text{Corr}_{\leq D} \) at the end of the day.

### 3.3 Computing \( M \)

The first step in writing down an explicit expression for \( M \) will be to express \( T^S \) in the basis \( \{A^I\} \).

**Definition 3.2 (List notation for \( S \)).** We will identify \( S \in \Omega \) with a multi-set, namely the multi-set containing \( S_I \) copies of each \( I \in \Omega \). With some abuse of notation, write \( S \) as an ordered list \( S = (I_1, \ldots, I_{|S|}) \) containing the elements of the associated multi-set sorted according to some arbitrary but fixed ordering on \( \Omega \). For a labeling \( \ell = (\ell_1, \ldots, \ell_{|S|}) \in [r]^{|S|} \), define \( S(\ell) \subseteq [n] \times [r] \) to be the subset containing all \((i,j)\) pairs with the following property: the element \( i \) occurs in an odd number of the sets \( \{I_d : \ell_d = j\} \).

(Informally, \( S(\ell) \) is produced by placing each \( I_d \) into column \( \ell_d \) of an \( n \times r \) grid and then XOR’ing the contents of each column.)

With this notation we can write

\[
T^S = \prod_{I \in \Omega} \left( \sum_{j=1}^r \lambda_j^I A^I_{S(I)} \right) = \sum_{\ell \in [r]^{|S|}} \lambda^\ell A^S(\ell) \quad (13)
\]

where \( \lambda^\ell := \prod_{d=1}^{|S|} \lambda_{\ell_d} \).

Next consider an arbitrary vector of coefficients \( \hat{f} \) and write down an expression for the corresponding \( \hat{g} \) in (10). We have

\[
f(T) = \sum_{|S| \leq D} \hat{f}_S T^S = \sum_{|S| \leq D} \hat{f}_S \sum_{\ell \in [r]^{|S|}} \lambda^\ell A^S(\ell) = \sum_{|U| \leq kD} A^U \sum_{|S| \leq D} \hat{f}_S \sum_{\ell \in [r]^{|S|}} \lambda^\ell \mathbb{I}_{S(\ell) = U}. 
\]

In other words, \( \hat{g} = M\hat{f} \) where

\[
M_{U,S} := \sum_{\ell \in [r]^{|S|}} \lambda^\ell \mathbb{I}_{S(\ell) = U} \quad (14)
\]

are the “generic” terms appearing in the expansion (13): \( U \) is generic if and only if there exists \( S \in \Omega \) with \( |S| \leq D \) and \( \ell \in [r]^{|S|} \) with distinct entries \( \ell_1, \ldots, \ell_{|S|} \) such that \( S(\ell) = U \). (Note that (6) implies \( D \leq r \), so it is possible for \( \ell \) to have distinct entries.) Furthermore, if \( U \) is generic then there is a unique corresponding \( S \), namely \( \text{cols}(U) \) defined as follows.

**Definition 3.4.** For a generic \( U \), define \( \text{cols}(U) \in \Omega \) by letting \( \text{cols}(U)_I \) be the number of columns \( j \) for which \( U_j = I \).

When viewing \( S \) as a multi-set as per Definition 3.2, \( \text{cols}(U) \) is simply the multi-set of columns \( U_j \). Recalling the definition \( |S| := \sum_{I \in \Omega} \text{cols}(U)_I \), note that \( |\text{cols}(U)| \) denotes the number of non-empty columns of \( U \).

Our left inverse \( M^* \) will satisfy \( M^*_{SU} = 0 \) whenever \( U \) is not generic; in other words, our procedure for recovering \( \hat{f} \) from \( \hat{g} \) only uses the values \( \hat{g}_U \) for which \( U \) is generic. Write \( M \) in block form

\[
M = \begin{bmatrix} G & N \end{bmatrix}
\]

where \( G \) ("generic") is indexed by generic \( U \)’s and \( N \) ("not") is indexed by the rest. It suffices to construct a left inverse \( G^* \) for \( G \) and then set

\[
M^* = \begin{bmatrix} G^* & 0 \end{bmatrix} \quad .
\]

(15)

Note that \( G = G(D) \) has the recursive structure

\[
G(D) = \begin{bmatrix} G(D-1) & R(D) \\ 0 & Q(D) \end{bmatrix}
\]

where the first block of columns is indexed by \( |S| \leq D-1 \) and the second is indexed by \( |S| = D \), and the first block of rows is indexed by \( |\text{cols}(U)| \leq D-1 \) and the second by \( |\text{cols}(U)| = D \).

Crucially, the lower-left block is 0; recall (14) and note that \( S(\ell) = U \) is only possible when \( |S| \geq |\text{cols}(U)| \). Given any left inverses \( G(D-1)^*, Q(D)^* \) for \( G(D-1), Q(D) \) respectively, one can verify that the following matrix is a valid left inverse for \( G(D) \):

\[
G(D)^* := \begin{bmatrix} G(D-1)^* & -G(D-1)^*R(D)Q(D)^* \\ 0 & Q(D)^* \end{bmatrix} \quad .
\]

(16)

The left inverse \( G(D-1)^* \) can be constructed by applying (16) recursively. The matrix \( Q(D) \) has only one nonzero entry per row and so we will be able to construct \( Q(D)^* \) by hand.

**Lemma 3.5.** The following is a valid left inverse for \( Q = Q(D) \): for \( U \) generic and \( |S| = |\text{cols}(U)| = D \), define \( Q^* = (Q_{SU}^*) \) by

\[
Q_{SU}^* := \frac{1}{\lambda_{U,r-1}} \mathbb{I}_{\text{cols}(U) = S} \quad (17)
\]

where

\[
\lambda_U := \prod_{j : |U_j| > 0} \lambda_j
\]

and

\[
r^D := r(r-1)(r-2) \cdots (r-d+1) \quad .
\]

(18)
where the last step is justified as follows: since $|S'| = |\text{cols}(U)| = D$, the indicator $\mathbb{I}_{S'(t) = U}$ can only be satisfied when $t$ has distinct entries. There are $r_{\text{alt}}$ such $t$’s, and for each there is exactly one term in the first sum satisfying $\mathbb{I}_{S'(t) = U}$, namely $U = S'(t)$, and this implies $\lambda^U = \lambda^t$ and $S' = \text{cols}(U)$. This means the indicator $\mathbb{I}_{\text{cols}(U) = S}$ becomes $\mathbb{I}_{S = S'}$, and the other factors cancel. This completes the proof. \qed

This completes the description of the left inverse $M^*$.

### 3.5 Recurrence for $w$

Ultimately, we are interested in an expression not for $M^*$ but for the vector $w^T := c^T M^*$ appearing in (12). We will use the calculations from the previous section to write down a self-contained recursive formula for the entries of $w$. Note that from (15), $w_{U} = 0$ whenever $U$ is not generic, so we will focus on computing $w_{\text{gen}} = (w_U : U \text{ generic}).$ Using (16),

$$w_{\text{gen}}^T = c^T G^* = \begin{bmatrix} c^T, G(D - 1)^* & \end{bmatrix} \begin{bmatrix} 0, R(D)Q(D)^* \end{bmatrix} =: [x^T, y^T],$$

where $x$ is indexed by $|\text{cols}(U)| \leq D - 1$ and $y$ is indexed by $|\text{cols}(U)| = D$. The expression for $x$ reveals that $w_U$ does not depend on $D$ (so long as $D \geq |\text{cols}(U)|$). By comparing the expressions for $x$ and $y$ we can write

$$y^T = c^T, 0, R(D)Q(D)^* = x^T R(D)Q(D)^*.$$

For any generic $U$, the case $D = |\text{cols}(U)|$ of the above gives

$$w_U = \sum_{S: |S| = |\text{cols}(U)|} c_S Q(D)^*_{SU} - \sum_{\text{generic } U'} \sum_{S: |S| = |\text{cols}(U)|} w_{U'} R(D)_{U'S} Q(D)^*_{SU}.$$

We treat the two terms (I), (II) separately. Using the definition (17) for $Q^*$,

$$w_{\text{gen}} = \sum_{S: |S| = |\text{cols}(U)|} c_S \mathbb{I}_{\text{cols}(U) = S} \mathbb{I}_{S = S'},$$

$$= \sum_{S: |S| = |\text{cols}(U)|} c_S \mathbb{I}_{\text{cols}(U) = S} \mathbb{I}_{S = S'},$$

$$= \sum_{S: |S| = |\text{cols}(U)|} \mathbb{I}_{\text{cols}(U) = S} \mathbb{I}_{S = S'},$$

where $S = \text{cols}(U)$.

Now for the second term (II), suppressing the constraints on $U'$ for ease of notation,

$$(II) = \sum_{U'} \sum_{S: |S| = |\text{cols}(U)|} w_{U'} R(D)_{U'S} Q(D)^*_{SU}$$

$$= \sum_{U'} \sum_{S: |S| = |\text{cols}(U)|} w_{U'} \left( \sum_{\ell \in [r]|S|} \lambda^\ell \mathbb{I}_{S(\ell) = U'} \right) \frac{\mathbb{I}_{\text{cols}(U) = S}}{\lambda^U r_{\text{alt}}},$$

where $S = \text{cols}(U)$.

Putting it together, we have now shown

$$\text{Corr}_{\leq D}^2 \leq \sum_{\text{generic } U} \frac{w_U^2}{\mathbb{I}_{\text{cols}(U) = S}} \text{ where } S = \text{cols}(U).$$

where, for generic $U$, the $w_U$ are defined by the recurrence

$$w_U = \frac{1}{\lambda^U r_{\text{alt}}} \mathbb{I}_{S = S'},$$

where $S = \text{cols}(U)$.

### 3.6 Recurrence for $v$

It will be convenient to rewrite the recurrence (20) in terms of a different quantity indexed by $S \in \mathbb{N}^D$ instead of $U \subseteq [n] \times [r]$, namely

$$v_S := \lambda^U r_{\text{alt}} w_U \text{ for } U \text{ satisfying } \text{cols}(U) = S.$$  

(21)

It can be seen from (20) that $v_S$ is well-defined in the sense that it does not depend on the choice of $U$ in (21). In particular,

$$v_S = c_S - \sum_{\text{generic } U} \sum_{|\text{cols}(U')| < |\text{cols}(U)|} \lambda^\ell \mathbb{I}_{S(\ell) = U'}.$$

Using (21) we can turn this into a self-contained recurrence for $v$ (not involving $w$):

$$v_S = c_S - \sum_{\text{generic } U} \sum_{|\text{cols}(U')| < |\text{cols}(U)|} \frac{v_{U'}}{\lambda^U r_{\text{alt}}} \sum_{\ell \in [r]|S|} \lambda^\ell \mathbb{I}_{S(\ell) = U'}.$$

(22)

where the predicate $\text{cols}(S(\ell)) = S'$ in particular requires $S(\ell)$ to be generic. For any $S \in \mathbb{N}^D$ there are at most $r_{\text{alt}}$ corresponding generic $U$’s for which $S = \text{cols}(U)$. For any such $U$, we have from (21) that $|w_U| \leq v_S / (\lambda^U r_{\text{alt}})$. This means (19) gives

$$\text{Corr}_{\leq D}^2 \leq \sum_{S \in \mathbb{N}^D} v_S^2 \left( \frac{v_S \lambda^U r_{\text{alt}}}{\lambda^U r_{\text{alt}}} \right) \sum_{S \in \mathbb{N}^D} \frac{w_U^2}{\mathbb{I}_{\text{cols}(U) = S}} \text{ where } v_S \text { is defined by the recurrence (22).}$$

(23)
3.7 Grouping by Patterns

The core challenge remaining is to analyze the recurrence (22) and establish an upper bound on $|c_S|$. Naively bounding the terms in (22) will not suffice, as there are subtle cancellations that occur. To understand the nature of these cancellations, we will rewrite (22) in a different form. First we will group the terms in (22) by their type, defined as followed. We use $\oplus$ to denote the XOR (symmetric difference) operation on sets.

Definition 3.6. Fix $S \in \mathcal{P}_0$, viewed as a list $S = (I_1, \ldots, I_{|S|})$ as per Definition 3.2. Define a pattern $\pi = (\pi_1, \ldots, \pi_{|S|})$ to be an element of $(\{r\} \cup \{\ast\})^{|S|}$ satisfying the following rules:

(i) "No singletons": if $\pi_d = j \in \{r\}$ then there must exist $d' \neq d$ such that $\pi_{d'} = j$.

(ii) "Not all stars": there must exist $d$ such that $\pi_d \neq \ast$.

(iii) "Every column valid": for every $j \in \{r\}$, we have $|\oplus_d: \pi_d = j| I_d \leq k$.

Let $\Pi(S)$ denote the set of all such patterns. We let $S(\pi)_j := \oplus_d: \pi_d = j I_d$ and define $S(\pi) \subseteq \{r\} \times \{r\}$ to have $j$th column $S(\pi)_j$ for all $j \in \{r\}$.

Note that if $\pi$ has no stars, it is simply a labeling $\ell \in \{r\}^{|S|}$, in which case the definitions $S(\pi)$ and $S(\ell)$ coincide. Intuitively, a pattern $\pi$ describes a class of possible labelings $\ell$, where the stars are "wildcards" that may represent any element of $\{r\}$ (subject to some restrictions). More formally, we now define which $\ell$'s belong to a pattern $\pi$.

Definition 3.7. Fix $S \in \mathcal{P}_0$, viewed as a list $S = (I_1, \ldots, I_{|S|})$. For $\ell \in \{r\}^{|S|}$ and $\pi \in \Pi(S)$, write $\pi \vdash \ell$ if the following conditions hold:

(i) For each $d$, if $\pi_d = j \in \{r\}$ then $\ell_d = j$.

(ii) The "starred" columns $\ell_\ast := \{\ell_d : \pi_d = \ast\}$ are distinct.

(iii) For every $j \in \ell_\ast$ we have $S(\pi)_j = \emptyset$.

Let $\mathcal{L}(S)$ denote the set of labelings $\ell \in \{r\}^{|S|}$ for which there exists $S' \in \mathcal{P}_0$ such that $|S'| < |S|$ and $\text{cols}(S(\ell)) = S'$; these are the $\ell$'s that contribute to the sum in (22). For any $\ell \in \mathcal{L}(S)$, there is at least one (and possibly more than one) pattern $\pi \in \Pi(S)$ for which $\pi \vdash \ell$. The following inclusion-exclusion formula will allow us to sum over $\pi \in \Pi(S)$ in a way that counts every $\ell \in \mathcal{L}(S)$ exactly once.

Lemma 3.8. Fix $S \in \mathcal{P}_0$, viewed as a list $S = (I_1, \ldots, I_{|S|})$. Fix a function $\phi : \{r\}^{|S|} \to \mathbb{R}$. For $\pi \in \Pi(S)$ and $j \in \{r\}$, define

$$m_\pi(j) = \left|\{d : \pi_d = j \text{ and } I_d = S(\pi)_j\}\right|$$

and

$$m_\pi = \sum_{j \in \{r\}} (1 - m_\pi(j)).$$

Then we have

$$\sum_{\ell \in \mathcal{L}(S)} \phi(\ell) = \sum_{\pi \in \Pi(S)} m_\pi \sum_{\ell \in \{r\}^{|S|} : \pi \vdash \ell} \phi(\ell). \tag{24}$$

Proof. First note that $\pi \vdash \ell$ implies $\ell \in \mathcal{L}(S)$ and so there are no "extra" terms $\phi(\ell)$ on the right-hand side that are not present on the left-hand side. For any fixed $\ell \in \mathcal{L}(S)$, the term $\phi(\ell)$ appears exactly once on the left-hand side of (24). Our goal is to show that it also appears exactly once on the right-hand side, that is, it suffices to prove

$$\sum_{\pi \in \Pi(S)} m_\pi = 1.$$

For a fixed $\ell$, we need to enumerate the possible patterns $\pi$ for which $\pi \vdash \ell$. The rules for these patterns are as follows:

- (Case 1) For any $j$, if there is exactly one $d$ for which $\ell_d = j$ then $m_\pi(j) = 0$
- (Case 2) For any $j$, if $S(\ell)_j = \emptyset$ then there are no stars among $\{\pi_d : \ell_d = j\}$. In this case, $m_\pi(j) = 0$
- (Case 3) For any $j$ not in Case 1, if $S(\ell)_j \neq \emptyset$ then among $\{\pi_d : \ell_d = j\}$ there are either no stars or exactly one star of the form $\pi_d = \ast$ where $\ell_d = j$ and $I_d = S(\ell)_j$. If there are no stars then $m_\pi(j) = m_\pi(\ast) = |\{d : \ell_d = j\} - S(\ell)_j|$. There are $m_\pi(j)$ ways to have one star, and each yields $m_\pi(j) = 0$.

Now we have

$$\sum_{\pi \vdash \ell} m_\pi = \sum_{j \in \{r\}} (1 - m_\pi(j)) = \prod_{j \in \{r\}} (1 - m_\pi(j)) = 1$$

as desired.

Using Lemma 3.8, we can rewrite (22) as

$$v_S = c_S - \sum_{S' : |S'| < |S|} v_{S'} \sum_{\ell \in \{r\}^{|S'|}} \lambda^{\ell}_{S'(r_S)} \mathbb{I}_{\text{cols}(S(\ell)) = S'}.$$

The number of labelings $\ell$ such that $\pi \vdash \ell$ is $r_\pi |_{S'}^{|\pi|}$ where $r_\pi$ is the number of columns $j \in \{r\}$ for which $S(\pi)_j = \emptyset$ and $s_\pi$ is the number of stars in $\pi$ (i.e., the number of indices $d$ such that $\pi_d = \ast$). Note that the ratio $\frac{\lambda_{S'}}{\lambda_{S}}$ depends only on $\pi$ (not on $\ell$), and so we define $\frac{\lambda_{S'}}{\lambda_{S}} = \frac{\lambda_{S'}}{\lambda_{S}}$. Also, the predicate $\text{cols}(S(\ell)) = S'$ depends only on $\pi$ (not on $\ell$), and we write this predicate as $S \overset{\pi}{\rightarrow} S'$. With this notation, the recurrence for $v_S$ becomes

$$v_S = c_S - \sum_{S' : |S'| < |S|} v_{S'} \sum_{\pi \in \Pi(S)} m_\pi \left(\frac{\lambda_{S'}}{\lambda_{S}}\right)^{|\pi|} \delta_{S \overset{\pi}{\rightarrow} S'} \cdot \delta_{S' \overset{\pi}{\rightarrow} S''}. \tag{25}$$

3.8 Unravelling the Recurrence

Next we will rewrite (25) in closed form (without recursion) by expanding the recursion tree as a sum over "paths." We first unpack the definition (11) for $c_S$. Using (13) and recalling that $A$ has i.i.d. Rademacher entries,

$$c_S = \mathbb{E}[S_{a_{11}}] = \sum_{\ell \in \{r\}^{|S|}} \lambda^{\ell} \mathbb{E}[A(\ell) \cdot a_{11}] = \sum_{\ell \in \{r\}^{|S|}} \lambda^{\ell} \mathbb{I}_{S(\ell) = \{(1,1)\}}.$$

By expanding the recursion tree of (25) we can write $v_S$ as a sum over paths which we denote by

$$S = S^{0} \overset{\pi_1}{\rightarrow} S^{1} \overset{\pi_2}{\rightarrow} \cdots \overset{\pi_{p-1}}{\rightarrow} S^{p} \overset{\pi_p}{\rightarrow} \perp.$$

1694
Formally, a path consists of $S^0$ and $\pi^1, \ldots, \pi^p$ (which then determine $S^1, \ldots, S^p$) with $S^t \in \mathbb{N}^r$ for all $t$, $\pi^t \in \Pi(S^t)$ for $t \leq p - 1$, and $\pi^p \in [r]^{\delta(p)}$ ("no stars at the final step") such that the following properties hold:

- For all $0 \leq t \leq p - 1$, we require $|S^{t+1}| < |S^t|$ and the predicate $S^t \overset{\pi^t}{\longrightarrow} S^{t+1}$ holds.
- For the final step $S^p \overset{\pi^p}{\longrightarrow} \perp$, we require $S^p(\pi^p) = \{(1, 1)\}$.

With this notation, the expression for $v_S$ in (25) can be written as

$$
\sum_{p \geq 0} \sum_{S^0 = S^p, \pi^1, \ldots, \pi^{p-1} \in \text{good}} (-1)^p \left( \prod_{t=0}^{p-1} m_{\pi^t} (r_{\pi^t}) \sum_{r \in S^{t+1}} \lambda(\pi^{t+1}) \right) \lambda^{\pi^p},
$$

where the special rule for the final step comes from (26).

### 3.9 Excluding Bad Paths

The next step is the crux of the argument: we will show that only certain types of paths contribute to (28), due to cancellations among the remaining paths.

**Definition 3.9 (Event).** For a path of the form (27), we say an event occurs at timestep $t \in \{0, 1, \ldots, p\}$ on column $j$ if there exists $d$ for which $\pi^d_j = j$.

Note that Definition 3.6 requires every timestep $t$ to have an event on at least one column $j$.

**Definition 3.10 (Deletion event).** An event at timestep $t$ on column $j$ is called a deletion event if $S^t(\pi^j)_j = \emptyset$.

**Definition 3.11 (Good/bad paths).** A path of the form (27) is called bad if there exists a column $j \in [r]$ such that the last event (i.e., with the largest $t$) on that column is a deletion event. If a path is not bad, it is called good.

**Lemma 3.12.** The total contribution from bad paths to (28) is 0. That is, $v_S$ is equal to

$$
\sum_{p \geq 0} \sum_{S^0 = S^p, \pi^1, \ldots, \pi^{p-1} \in \text{good}} (-1)^p \left( \prod_{t=0}^{p-1} m_{\pi^t} (r_{\pi^t}) \sum_{r \in S^{t+1}} \lambda(\pi^{t+1}) \right) \lambda^{\pi^p}.
$$

Proof. We will show that the bad paths can be paired up so that within each pair, the two paths contribute the same term to (28) but with opposite signs. The pairing is described by the following procedure, an involution that maps each bad path to its partner (and vice versa).

(1) Given a bad path as input, let $j^*$ denote the largest column index for which the last event is a deletion event. Let $\tau^*$ denote the timestep on which this deletion event occurs.

(2a) If there exists another event at timestep $t^*$ (on some column $j \neq j^*$), "promote" the $(t^*, j^*)$ deletion event to its own timestep. That is, replace

$$
\ldots S^{t^*} \overset{\pi^{t^*}}{\longrightarrow} \ldots \text{ by } \ldots S^{t^*} \overset{\tau}{\longrightarrow} S^\tau \overset{\sigma}{\longrightarrow} \ldots
$$

where $\tau, \sigma$ are defined as follows. For all $d$ such that $\pi^d_j = j^*$, set $\tau_d = j^*$; for all other $d$, set $\tau_d = \bigstar$. Now $S^\tau$ (viewed as a list) is produced from $S^{t^*}$ by removing the elements $I_d$ for which $\pi^d_j = j^*$; similarly, $\sigma$ is produced from $\pi^\tau$ by removing the elements $I_d$ that are equal to $j^*$.

(2b) If instead there is no other event at timestep $t^*$ (which cannot happen if $t^* = p$ due to the non-deletion event on column 1), "merge" timestep $t^*$ with the subsequent timestep. That is, replace

$$
\ldots S^{t^*} \overset{\pi^{t^*}}{\longrightarrow} S^{t^*+1} \overset{\pi^{t^*+1}}{\longrightarrow} \ldots \text{ by } \ldots S^{\tau} \overset{\tau}{\longrightarrow} \ldots
$$

where $\tau$ is defined as follows. For all $d$ such that $\pi^d_j = j^*$, set $\tau_d = j^*$. The number of remaining entries of $\tau$ is exactly $|S^{t^*+1}|$, and we denote these entries as "unassigned" for each $1 \leq i \leq |S^{t^*+1}|$. Set the ith unassigned entry of $\tau$ to be $\pi_i^{t^*+1}$. Note that since the $(t^*, j^*)$ event was the last event in its column, the merge operation will not cause it to "collide" with another event.

A few claims remain to be checked before the proof is complete.

First note that the above procedure maps any bad path to a different bad path (its "partner"), and applying the procedure again on the partner recovers the original path. For instance, applying the procedure to the original path resulted in a "promote" operation on column $j^*$, applying the procedure on the partner will undo this using a "merge" operation on the same column $j^*$.

We furthermore claim that for any bad path and its partner, both paths have the same value for the factor $\prod_{r \in \mathbb{N}^r} \lambda(\pi^r)$ in (28). However, the lengths of the two paths differ by one, causing the two corresponding terms in (28) to cancel due to the $(-1)^p$ factor. To prove the claim, compare the cases $S^t \overset{\pi^t}{\longrightarrow} S^{t+1}$ and $S^t \overset{\tau}{\longrightarrow} S^{\tau} \overset{\sigma}{\longrightarrow} S^{\tau+1}$ from (2a) above, where for now we assume $S^{\tau+1} \neq \perp$. For the $m$ factors, note that $m_{\pi^t}(j) = 0$ whenever $\pi$ has either a deletion event on column $j$ or no event on column $j$, and so $m_{\tau} = m_{\sigma}$. For the $r_\pi$ factors, note that $r_{\pi^t} = r_\tau$ and $r_{\pi^\tau} = r_\sigma$; also, $r_\tau = r$ and $s_\tau = |S^\tau|$, so $(r_\tau) \lambda^{\tau}$ cancels with the existing factor of $|r_\tau| \lambda^{\tau}$ in (28). Finally, for the $\lambda$ factors we have $\lambda(\pi^\tau) = \lambda(\tau)\lambda(\sigma)$. The other case $S^t \overset{\pi^t}{\longrightarrow} S^{t^*} \overset{\pi^{t^*}}{\longrightarrow} S^{t^*+1}$ versus $S^t \overset{\tau}{\longrightarrow} S^{\tau} \overset{\sigma}{\longrightarrow} S^{\tau+1}$ is treated similarly, where now we have $m_{\pi^t} = 1$, $(r_\tau) \lambda^{\tau} = r_\tau \lambda^{\tau}$, and $\lambda^{\pi^\tau} = \lambda(\tau)\lambda(\sigma)$. This completes the proof.

### 3.10 Bounding $|v_S|$

Now that we have identified the crucial cancellations between pairs of bad paths, the rest of the proof will follow by bounding the terms in (29) in a straightforward way. We start by collecting some bounds on the individual pieces of (29).

**Lemma 3.13.** For any step $S \overset{\pi}{\longrightarrow} S'$, $|m_{\pi}| \leq 2^{|S|-|S'|}$.

Proof. Note that

$$
|m_{\pi}(j)| \leq |\{d : \pi_d = j\}| = |\pi^{-1}(j)|.
$$

The number of stars plus the number of distinct columns in $\pi$ must be at least $|S'|$. This leaves at most $|S| - |S'|$ entries of $\pi$ that repeat a previous column, i.e.,

$$
\sum_{j \in [r] : |\pi^{-1}(j)| \geq 2} |\pi^{-1}(j)| - 1 \leq |S| - |S'|.
$$

1695
This means

\[
|m_\pi| = \prod_{j \in [r]} |m_\pi(j) - 1| \leq \prod_{j \in [r] : |m_\pi(j)| \geq 2} (m_\pi(j) - 1) \\
\leq \prod_{j \in [r] : |m_\pi^{-1}(j)| \geq 2} (|m_\pi^{-1}(j)| - 1) \leq \prod_{j \in [r] : |m_\pi^{-1}(j)| \geq 2} 2^{(|m_\pi^{-1}(j)| - 1)} \\
= 2^{\sum_{j \in [r]} |m_\pi^{-1}(j)| \geq 2 (|m_\pi^{-1}(j)| - 1)} \leq 2^{|S| - |S'|}
\]

where the final step uses \((30)\). \(\Box\)

**Lemma 3.14.** For any step \(S \xrightarrow{\pi} S'\), \(|\lambda(S')| \leq 1\).

**Proof.** Recall \(\lambda(S') = \frac{\lambda'}{\lambda(\ell)}\) for any \(\ell\) such that \(\pi \upharpoonright \ell\). Recall that \(\lambda^{S(\ell)}\) is the product of \(\lambda_j\) over the non-empty columns \(j\) of \(S(\ell)\). For any such non-empty column \(j\), there must exist \(d\) for which \(\ell_d = j\).

Thus, every factor of \(\lambda_j\) in the denominator of \(\frac{\lambda'}{\lambda(\ell)}\) is cancelled by the numerator, and the result now follows because \(|\lambda_j| \leq 1\). \(\Box\)

We now state the main conclusion of this section.

**Lemma 3.15.** For any \(S \in \mathcal{M}^{|\Omega|}\) with \(|S| \geq 1\), we have \(|v_S| \leq (3|S|^2)|S|\).

Note that for \(|S| = 0\), it can be verified directly that \(v_0 = 0\).

**Proof.** Proceed by induction on \(|S|\). We will bound the sum of absolute values of terms in \((29)\); it will no longer be important to exploit cancellations between positive and negative terms. First consider paths of the form \(S \xrightarrow{\pi} \perp\). There is at most one such path that is good, namely \(\pi^0 = (1, 1, \ldots, 1)\); the value of this term is \(|\lambda^\pi| \leq 1\).

All remaining paths take the form \(S \xrightarrow{\pi} S^1 \cdots \perp\) where \(|S^1| = i\) for some \(1 \leq i \leq |S| - 1\). In order to produce \(S^1\), \(\pi^0\) must have exactly \(i\) entries \(d\) that are either stars (i.e., \(\pi^d_0 = \ast\)) or first in a “combination” event (i.e., for some \(j \in [r]\) with \(S(\pi^0)_j \neq \emptyset\), \(d\) is the lowest index such that \(\pi^d_0 = j\)).

There are \([|S|]\) choices for which \(i\) entries play these roles; by default they will all be stars, and will be converted to "combinations" if another entry of \(\pi^0\) decides to join them on the same column.

Now there are \(|S| - i\) entries of \(\pi^0\) remaining, which have a few options. One option is to participate in a combination event by joining one of the \(i\) previously designated entries on the same column. The other option is to participate in a deletion event. Since we are only counting good paths, this can only happen on a column on which a later event will occur. Regardless of the remainder of the path \(S^1 \xrightarrow{\pi} \cdots \perp\), there are at most \(|S^1| = i\) such columns available. Since each of \(|S| - i\) entries of \(\pi^0\) has at most \(2i\) choices, this gives a total of at most \((2i)^{|S| - i}\) choices.

We also need to decide which columns the combination events occur on. If \(\pi^0\) has \(s_{\pi^0}\) stars and \(i - s_{\pi^0}\) combination events, there are \(r^{i - s_{\pi^0}}\) choices for the columns. Note that this exactly cancels the factor \((r_{\pi^0})^{s_{\pi^0}}/r_{|S|}\) in \((29)\) because \(r_{\pi^0} = r - (i - s_{\pi^0})\) and \(|S^1| = i\).

Recall that we aim to show \(|v_S| \leq b(|S|)\) where \(b(d) = (3d^2)^d\). Plugging the above calculations (along with Lemmas 3.13 and 3.14) into \((29)\) and using the induction hypothesis to handle the remainder of the path \(S^1 \xrightarrow{\pi} \cdots \perp\), we have that \(|v_S|\) is at most

\[
1 + \sum_{i=1}^{|S|-1} b(i) (\frac{|S|}{i}) (2i)^{|S|-i} 2^{|S|-i} = 1 + \sum_{i=1}^{|S|-1} 3(2^i)^i (4i)^{|S|-i} \\
\leq 1 + \sum_{i=0}^{|S|-1} (\frac{|S|}{i}) (3(|S| - 1)^2/4(|S| - 1)) |S|^{i-1} \\
= 1 + \sum_{i=0}^{|S|-1} (\frac{|S|}{i}) (3(|S| - 1)^2/4(|S| - 1)) |S|^{i-1} \\
= 3(|S|^2 - 6|S| + 3 + 4|S| - 4) |S| \leq (3|S|^2)|S|,
\]

completing the proof. \(\Box\)

### 3.11 Putting It Together

We now combine \((23)\) with Lemma 3.15 to complete the proof of Theorem 2.1. We first note that \(v_S \neq 0\) only when the elements of \(S\) (viewed as a multi-set) together with \(\{1\}\) form an even cover of \([n]\).

**Lemma 3.16.** Let \(P(S)\) denote the property that \(S(\ell) = \{(1, 1)\}\) for \(\ell = (1, 1, \ldots, 1)\). If \(P(S)\) fails to hold then \(v_\emptyset = 0\).

**Proof.** From \((26)\), note that if \(v_\emptyset \neq 0\) then \(S(\ell) = \{(1, 1)\}\) for some \(\ell\), which implies \(S(\ell) = \{(1, 1)\}\) for \(\ell = (1, 1, \ldots, 1)\). Thus \(v_\emptyset = 0\) whenever \(P(S)\) fails.

Also note that if \(P(S)\) fails and \(\text{cols}(S(\ell)) = S'\) for some \(\ell\), then \(P(S')\) fails. The result now follows from \((22)\) using induction on \(|S|\). \(\Box\)

**Lemma 3.17.** For any \(d \geq 1\), the number of multi-sets \(S \in \mathcal{M}^{|\Omega|}\) with \(|S| = d\) such that \(P(S)\) holds is at most \(n^{(kd-1)/2}((kd + 3)/2)^kd\).

**Proof.** Since \(P(S)\) holds, \(S\) together with \(\{1\}\) forms an even cover of \([n]\). Therefore the number of elements of \([n] \setminus \{1\}\) covered by \(S\) is at most \((kd - 1)/2\). The number of ways to choose this many elements is at most \(n^{(kd-1)/2}\). Once these are chosen, \(S\) has at most \((kd - 1)/2 + 1 = (kd + 1)/2\) elements to draw from, so the number of possibilities for \(S\) is at most \((kd + 1)/2 + 1 = (kd + 3)/2kd\). \(\Box\)

**Proof of Theorem 2.1.** Start from \((23)\) and use Lemmas 3.15, 3.16, and 3.17:

\[
\text{Corr}_D^{z_D} \leq \sum_{|S| \leq d} \frac{v_S}{\lambda^{|S|}} \frac{r_{|S|}}{r_{|S'|}} \\
\leq \sum_{d=1}^D n^{(kd-1)/2}((kd + 3)/2)^kd \frac{(3d^2)^{2d}}{\lambda_{\min}^{2d}(r - d + 1)^d} \\
\leq n^{-1/2} \sum_{d=1}^D (9D^4(KD)^k \frac{19n^{k/2}}{18\lambda_{\min}^2})^d
\]
since (6) implies $D \leq r/19$
\[= n^{-1/2} \sum_{d=1}^{D} \left( \frac{19}{2} \right)^{d/2} \binom{n^{d/2}}{d/2} \leq n^{-1/2} \sum_{d=1}^{D} \left( \frac{1}{2} \right)^{d} , \]
where we have used the assumption (6). This means $\text{Corr}_2^{\leq D} \leq n^{-1/2}$, which completes the proof using Fact 3.1.

ACKNOWLEDGMENTS
Part of this work was done while with the Algorithms and Randomness Center at Georgia Tech, supported by NSF grants CCF-2007443 and CCF-2166444.

The author is indebted to Jonathan Niles-Weed, Tselil Schramm, and Jerry Li for numerous detailed discussions about this problem. The author also thanks Jingqiu Ding, Bruce Hajek, Tim Kunisky, cris Moore, Aron Potechin, Bobby Shi, and anonymous reviewers, for helpful discussions and comments.

REFERENCES
[1] Dimitris Achlioptas and Amin Coja-Oghlan. 2008. Algorithmic barriers from phase transitions. In 2008 49th Annual IEEE Symposium on Foundations of Computer Science. IEEE, 793–802.
[2] Esha Al-Sharara, Mahmood Al-Khassawneh, and Selin Avyente. 2017. A tensor based framework for community detection in dynamic networks. In International conference on acoustics, speech and signal processing (ICASSP). IEEE, 2312–2316.
[3] Anima Anandkumar, Dean P. Foster, Daniel J. Hsu, and Sham M. Kakade. 2014. Learning latent variable models through tensor methods. In Conference on Learning Theory. PMLR, 787–881.
[4] Anima Anandkumar, Rong Ge, and Majid Janzamin. 2014. Analyzing tensor power method dynamics: Applications to learning overcomplete latent variable models. arXiv preprint arXiv:1411.1488 (2014).
[5] Anima Anandkumar, Rong Ge, and Majid Janzamin. 2015. Learning overcomplete latent variable models through tensor methods. In Conference on Learning Theory. PMLR, 36–112.
[6] Jesse Banks, Dmitriy Kunisky, Christopher Moore, and Alexander S. Wein. 2021. Smoothed analysis of tensor decompositions. In Conference on Learning Theory. PMLR, 868–875.
[7] Jesse Banks, Dmitriy Kunisky, Christopher Moore, and Alexander S. Wein. 2021. Spectral planted and the hardness of refuting cuts, colorability, and communities in random graphs. In Conference on Learning Theory. PMLR, 410–473.
[8] Jesse Banks, Dmitriy Kunisky, Christopher Moore, and Alexander S. Wein. 2020. Computational Hardness of Certifying Bounds on Constrained PCA Problems. In 11th Innovations in Theoretical Computer Science Conference (ITCS 2020). Vol. 151. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 78.
[9] Jess Banks, Sidhanth Mohanty, and Prasad Raghavendra. 2021. Local statistics, semidefinite programming, and community detection. In Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA). SIAM, 1298–1316.
[10] Boaz Barak, Samuel Hopkins, Jonathan Kelner, Pravesh K. Kothari, Aron Potechin, and Aaron Potechin. 2019. A nearly tight sum-of-squares lower bound for the planted clique problem. SIAM J. Comput. 48, 2 (2019), 687–735.
[11] 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, 143–151.
[12] Gerard Ben Arous, Reza Gheissari, and Aukosh Jagannath. 2020. Algorithmic thresholds for tensor PCA. The Annals of Probability 48, 4 (2020), 2022–2087.
[13] Gerard Ben Arous, Reza Gheissari, and Aukosh Jagannath. 2020. Algorithmic thresholds for tensor PCA. The Annals of Probability 48, 4 (2020), 2022–2087.
[14] Quentin Berthet and Philippe Rigollet. 2013. Complexity theoretic lower bounds for sparse principal component detection. In Conference on learning theory. PMLR, 1044–1066.
[15] Aditya Bhaskara, Moses Charikar, Anupam Gupta, and Aravindan Vijayaraghavan. 2014. Smoothed analysis of tensor decompositions. In Proceedings of the forty-sixth annual ACM symposium on Theory of computing, 594–603.
[16] Luca Bihoro, Chiara Camoriano, and Federico Ricci-Tersenghi. 2020. How to iron out rough landscapes and get optimal performances: averaged gradient descent and its application to tensor PCA. Journal of Physics A: Mathematical and Theoretical 53, 17 (2020), 174003.
[17] Cristiano Borei, Luca Chiantini, and Giorgio Ottaviani. 2014. Reduced methods for the identifiability of tensors. Annali di Matematica Pura ed Applicata (1923-) 193, 5 (2014), 1691–1702.
[18] Nicolas Boumal, Tahir Bendory, Ryan L. Bledsoe, and Ameet Singher. 2018. Heterogeneous multireference alignment: A single pass approach. In 52nd Annual Conference on Information Sciences and Systems (CISS). IEEE, 1–6.
[19] Matthew Brennan and Guy Bresler. 2020. Reducibility and statistical-computational gaps from secret leakage. In Conference on Learning Theory. PMLR, 648–647.
[20] Matthew Brennan, Guy Bresler, and Wasim Huleihel. 2018. Reducibility and computational lower bounds for problems with planted sparse structure. In Conference On Learning Theory. PMLR, 48–166.
