Identifying Mixtures of Bayesian Network Distributions

Spencer L. Gordon∗ Bijan Mazaheri† Yuval Rabani‡ Leonard J. Schulman§

December 23, 2021

Abstract

A Bayesian Network is a directed acyclic graph (DAG) on a set of \( n \) random variables (identified with the vertices); a Bayesian Network Distribution (BND) is a probability distribution on the \( n \)’s that is Markovian on the graph. A finite mixture of such models is the projection on these variables of a BND on the larger graph which has an additional “hidden” (or “latent”) rv \( U \), ranging in \( \{1, \ldots, k\} \), and a directed edge from \( U \) to every other vertex.

Models of this type are fundamental to research in Causal Inference, where \( U \) models a confounding effect. One extremely special case has been of longstanding interest in the theory literature: the empty graph. Such a distribution is simply a mixture of \( k \) product distributions. A longstanding problem has been, given the joint distribution of a mixture of \( k \) product distributions, to identify each of the product distributions, and their mixture weights. Our results are:

(1) We improve the sample complexity (and runtime) for identifying mixtures of \( k \) product distributions from \( \exp(O(k^2)) \) to \( \exp(O(k \log k)) \). This is almost best possible in view of a known \( \exp(O(k)) \) lower bound.

(2) We give the first algorithm for the case of non-empty graphs. The complexity for a graph of maximum degree \( \Delta \) is \( \exp(O(k(\Delta^2 + \log k))) \).

(The above complexities are approximate and suppress dependence on secondary parameters.)
Contents

1 Introduction ................................................................. 1
  1.1 Background .............................................................. 2
  1.2 Bayesian networks and causal identification ....................... 3
  1.3 Related work ........................................................... 4

2 Mixtures of Product Distributions ..................................... 4
  2.1 Problem statement. Hadamard extensions and their condition number .... 4
  2.2 Algorithm for mixtures of product distributions: overview and theorem .......... 7

3 General Bayesian Networks ............................................. 8
  3.1 Describing Bayesian networks ......................................... 8
  3.2 Applying a $k$-MixProd algorithm ..................................... 9
    3.2.1 Describing runs .................................................... 9
    3.2.2 Assumptions ....................................................... 10
  3.3 Using different runs to learn source-specific parameters ............... 11
  3.4 Aligning source labels across different runs ........................ 11
  3.5 Bayesian unzipping: recovering parameters per source ................ 12
    3.5.1 Recovering the distribution on sources ........................ 13
  3.6 A good collection of runs ............................................ 14
  3.7 Outline of the algorithm ............................................ 15
  3.8 Statement of main result for $k$-MixBND ............................ 15

4 Analyzing the $k$-MixBND algorithm ................................ 16
  4.1 Applying the $k$-MixProd algorithm on each run .................... 16
  4.2 Analysis of alignment ............................................... 16
  4.3 Recovering Parameters .............................................. 16

Appendices ........................................................................ 21

A Finding good collections of runs .................................... 21
  A.1 Constructing a good collection of runs from $2k$ centers ............ 21
  A.2 Degree bounds ......................................................... 23
  A.3 Mixtures of paths ...................................................... 23

B Extending mixtures of binary product distributions to larger alphabets .... 25
  B.1 Larger alphabets for mixtures of DAGs ............................. 27

C Mixtures of Product Distributions .................................... 28
  C.1 The Algorithm ......................................................... 29
  C.2 Analyzing the Algorithm ............................................. 32
  C.3 Miscellaneous Proofs .................................................. 37

References ......................................................................... 38
1 Introduction

A Bayesian Network is a directed acyclic graph $G = (\mathcal{V}, E)$, on a set of $n$ random variables (identified with the vertices); a Bayesian Network Distribution (BND) is a probability distribution on the rv’s that is Markovian on the graph. That is to say, the joint distribution on the variables can be factored as $\prod_{i=1}^{n} \mathcal{P}[V_i = v_i \mid \text{pa}(V_i)]$ where $\text{pa}(V_i)$ is the assignment to the parents of $V_i$. A $k$-MixBND on $G$ is a convex combination of $k$ BNDs. Such a distribution arises naturally if there is an additional, “confounding” random variable $U$ with range $\{1, \ldots, k\}$. Then the distribution on the variables in $G$ (the “observables”) is the BND of a larger graph in which $U$ is an extra vertex having directed edges to every vertex of $G$. The main complexity parameter of the problem is $k$.

Models of this type (and even more general) are fundamental to research in Causal Inference, where the chief difficulty has to do with performing statistical inference for collections of random variables which, besides some known potential direct effects on each other (the directed edges), are also affected by unseen confounders.

One extremely special case has been of longstanding interest in the theory literature: where $G$ is empty. Such a distribution is a mixture of $k$ product distributions or $k$-MixProd. See Fig. 1.

![Figure 1: (a) A small Bayesian Network, with latent variable $U$. (b) In the empty graph, a $k$-MixBND is a $k$-MixProd (mixture of product distributions).](image)

In this paper we study the Identification problem for $k$-MixBNDs. Specifically, given the graph $G$, and given a joint distribution $\mathcal{P}$ on the variables (vertices), recover up to small statistical error (a) the mixture weights, up to a permutation of the constituents, and (b) for every mixture constituent and for every vertex $V$, its conditional distribution given each possible setting to its parents. We assume that each individual random variable (vertex) ranges in a finite set of size $\leq d$, which we show w.l.o.g. (and at mild cost in complexity), can be taken as $2$. Clearly, $k$-MixBND models are not always identifiable. For instance, if $k > 1$ and $G$ consists of a single vertex; more generally if two constituents of the mixture produce the same BND, or if one BND is the convex combination of two others. Thus, another contribution of our paper is to establish mild conditions on the size of $G$ and on the constituent BND’s, that suffice to guarantee identifiability.

The following theorem gives our main results in somewhat simplified form (see Theorems 10, 35 for more detail). Here every observable is binary; $\zeta$-separated and $\zeta$-informative are conditions (see later) enabling quantitative guarantees on identifiability; the mixture weights are $\pi_i$; and $\varepsilon$ represents the distance of the output model from the true model.

**Theorem 1.** The algorithms given later w.h.p. identify a $k$-MixBND, with the following runtime (including sample) complexity.

1. Suppose $\Delta = 0$ (i.e., $k$-MixProd) and the $n \geq 3k - 3$ observables are $\zeta$-separated.
   The runtime is $(1/\zeta)^{O(k \log k)} (n \log n) (1/\min \pi_i)^{O(\log k)} (1/\varepsilon)^2$.

2. Suppose $\Delta \geq 1$, $n \geq \Omega(k)$, and the $k$-MixBND is $\zeta$-informative.
   The runtime is $(1/\zeta)^{O(k^{(\Delta^2+1)\log k})} (n \log n) (1/\min \pi_i)^{O(\log k)} (1/\varepsilon)^2$. 

1
In the $k$-MixProd result it is actually sufficient that there exist a subset of $3k - 3$ $\zeta$-separated random variables. Under this weaker assumption, the runtime above is multiplied by $n^{O(k)}$. (The sample size is unaffected.)

1.1 Background

To the best of our knowledge, the only prior work on $k$-MixBND estimation problem in the theory literature deals with the special case of an empty graph $G$ (the joint distribution conditional on the confounder $U$ is a product distribution), see Fig. 1(b). This is better known as the problem of identifying a mixture of product distributions, $k$-MixProd. It has been studied for nearly 30 years [KMR+94, CGG01, FM99, FOS08, CR08, TMMA18, CM19, GMRS21] (and we do not even try to list the extensive analogous literature for parametrized distributions over $\mathbb{R}$). There are actually two versions of the problem: (1) Learning the model, namely, producing any model consistent with (or close to) the observations; (2) Identifying the model, namely, producing the true model (or one close to it). These are incomparable problems, as the second goal is stronger but not always possible. Our paper is concerned entirely with Identification, because the underlying motivation is to be able to predict the effect of interventions in a system (represented by the graph). Such predictions will not be valid when the model is not identifiable.

In [FOS08] a seminal algorithm for $k$-MixProd was given. Its running time, for mixtures on $n$ binary variables ($n$ suff. large), is $n^{O(k^3)}$. This was improved in [CM19] to $k^{O(k^3)}n^{O(k^2)}$. The most recent algorithm [GMRS21] identifies a mixture of $k$ product distributions on at least $3k-3$ variables in time $2^{O(k^2)}n^{O(k)}$, under a mild “separation” condition that excludes unidentifiable instances (see below). Under somewhat stricter separation, the time complexity improves to $2^{O(k^2)}n$. In this paper we improve the latter results, giving upper bounds with exponent $O(k \log k)$ rather than $O(k^2)$. This is nearly optimal, as it is known that even in the special case that the observables are known to be iid conditional on $U$—call this the $k$-MixIID problem—exponent $\Omega(k)$ is unavoidable [RSS14]. Moreover $n \geq 2k-1$ observable variables are generally necessary. (There are interesting special cases in which much better efficiency is possible. [CM19] study “mixtures of sub-cubes,” $k$-MixProd with all bit probabilities in $\{0, \frac{1}{2}, 1\}$, and achieve complexity $n^{O(\log k)}$.)

It turns out that the $k$-MixIID problem plays a special role at the base of a tower of reductions. On the one hand it can be solved using a relatively simple, two-century-old method of Prony [dP95], which connects it to the classical Hausdorff moment problem; it can also be solved by the Matrix Pencil Method [RSS14, LRSS15, KKM+19, GMRS20]. Both analyses are quite recent however. In turn, the $k$-MixProd problem was solved by a nontrivial reduction to $k$-MixIID [GMRS21]. And, in the present paper, we show that, by a rather complex reduction involving subjecting the model to many conditionings, the $k$-MixBND problem can in turn be reduced to $k$-MixProd.

Thus, one contribution of our paper is to add to the corpus of evidence of the amazing power of the $k$-MixIID problem. (Apart from its many direct applications—op. cit.) It turns out that if one wants to identify mixtures of Bayesian network distributions, then solving this extremely restricted case is (possibly up to a logarithmic factor in the exponent) the complexity bottleneck.

As already noted, a $k$-MixBND can be unidentifiable (i.e., there may be two distinct models mapping to the same distribution on the observables). There are two fundamental ways this can occur. One is that the particular distribution is degenerate. We will rule out such degeneracy by assumptions we call $\zeta$-separated, and later, $\zeta$-informative. The second is that the graph $G$ is too “impoverished”—the simplest way for this to occur is if $n$ is too small. (As noted, even in the empty graph, $n \geq 2k-1$ is needed information-theoretically.) But it can also happen if $G$ has too many edges. A sufficient condition to rule out such obstacles is that $n \geq \Omega(k(\Delta + 1)^3)$.

The parameter $k$ is vital. In a Bayesian network where there is an unobserved $U$ which can
directly affect every observed variable, causal identification is not possible in general. Each value \( u \in U \) could determine completely the values of all observable variables. There is no way to distinguish this model from models that include direct causal relationships among the observables. However, the additional assumption of bounded range for \( U \) changes the situation, and in this paper we show that there indeed are many graphs for which causal identification is possible.

1.2 Bayesian networks and causal identification

The primary application of Bayesian networks (a term coined in [Pea85]) is to model complex causal relations. Concrete applications abound: mining data on diverse populations or multi-purpose usage, bioinformatics, etc. In a wide range of complex natural and artificial systems causal relations are probabilistic, and moreover inquiry using controlled experiments is impossible or prohibitively expensive. In such settings researchers often have to resort to collecting samples of an assortment of measurements that are related in a pattern of probabilistic causal relations that govern the joint distribution. The conditional dependencies among the components of the joint distribution are traditionally modeled as a Bayesian network. Often, practitioners have domain-specific knowledge that leads to the construction of a Bayesian network representing the direct causal relationships among the variables under examination. However, if there are multiple populations for which the strength of the causal relationships differ, the observed statistics are a convex combination of the statistics of the different populations. Multiplicity of populations, or equivalently an unobserved discrete confounding variable \( U \), will create joint distributions on the observed variables that obscure the true causal behavior of the system. On the other hand, while such deviations of the observables from the assumed causal structure pose a challenge, they also contain valuable information about the presence and behavior of the hidden variable \( U \) that created them. For some critical algorithms in this literature see [SP06, HV08], and for a broader survey [Pea09, SGS00a, PJS17, KF09].

We briefly outline (for binary observables) how we solve \( k \)-MixBND using \( k \)-MixProd as a subroutine. The algorithm requires a set of \( 3k - 3 \) nodes of \( G \) with disjoint Markov Boundaries. See Figure 2 for an illustration. In particular, if the maximum degree in the undirected skeleton of \( G \) is \( \Delta \geq 2 \) and the number of observables is \( n \geq (\Delta + 1)^4(3k - 3) \), then this condition is satisfied. (If the skeleton of \( G \) happens to be a path, then we only need a milder condition that \( n \geq 2(3k - 3) \).) We then execute a sequence of “runs” of the \( k \)-MixProd algorithm on the selected set of variables, for all the possible restrictions (assignments of values) to their Markov boundaries. The main challenge is then to align the outcomes of all the runs, and then to propagate the conditional probabilities through the entire network, in order to recover the model, a procedure we call “unzipping.” Thus,

![Figure 2: A Bayesian network. Four vertices with disjoint Markov boundaries are indicated.](image-url)
the reduction to $k$-MixProd relies on three contributions of this paper: the construction of a good collection of runs, the alignment algorithm, and the Bayesian unzipping algorithm.

Each individual run of $k$-MixProd employs (a variation on) the method of bootstraping synthetic bits, introduced in [GMRS21]. Our contribution here which leads to the near-optimal bound of $\exp(k \log k)$ (rather than $\exp(O(k^2))$) is a method to bound the singular values of the matrix of multilinear moments of the distribution on observables.

1.3 Related work

Some loosely related work includes learning hidden Markov models [HKZ12, AHK12, SKLV17], an incomparable line of work to our question, but with somewhat similar motivation. In the same vein, some papers study learning mixtures of Markov chains from observations of random walks through the state space [BGK04, GKV16]. These models, too, differ substantially from the models addressed in this paper, and pose very different challenges. Literature on causal structure learning [SGS+00b, GZS19] answers the question of identifying the presence of hidden confounders. Fast Causal Inference (FCI) harnesses observed conditional independence to learn causal structure, which can detect the presence of unobserved variables when the known variables are insufficient to explain the observed behavior. This literature includes the MDAG problem in which the DAG structure may depend upon the hidden variable, see [TMCH98] for heuristic approaches to this problem. Other related works study causal inference in the presence of visible “proxy” variables which are influenced by a latent confounder [MGT18, KP14]. This has more recently given rise to multiple causal inference, which assumes multiple causes which are independent when conditioned on a confounder [Hec18, RP18, WB19]. These settings are given with respect to a known causal inference task and significantly less general assumptions on the behavior of the proxy variables. Finite mixture models have been the focus of intense research for well over a century, since pioneering work in the late 1800s [New86, Pea94], and doing justice to the vast literature that emanated from this endeavor is impossible within the scope of this paper. See, e.g., the surveys [EH81, TSM85, Lin95, MLR19].

2 Mixtures of Product Distributions

In this section, we describe and analyze an algorithm for identifying a mixture of product distributions (a.k.a. $k$-MixBNDs on the empty graph). It improves substantially the previously best sample size and time complexity in [GMRS21], and it nearly matches the known lower bound that holds even in the simpler case that the observables are iid [RSS14]. The chief novelty of this section is not the algorithm, which is similar to that in [GMRS21] although one significant change has been necessary, but the analysis, which is entirely different and stronger.

For this reason we arrange the section in a perhaps unusual order: after some necessary definitions, first we give the crucial new theorem which is essential to the sample complexity bound. Only afterward we give the algorithm and the theorem which analyzes its performance.

2.1 Problem statement. Hadamard extensions and their condition number

Problem statement and notation. We have a hidden variable $U$ ranging in $[k]$, with distribution $\pi_j := \Pr(U = j)$. There are $n$ binary observable variables $V_1, V_2, \ldots, V_n$; their distributions are given by the rows $m_i$, with entries $m_{ij} := \Pr[V_i = 1 \mid U = j]$, of an $n \times k$ matrix $m$. Thus, the identification problem is to recover $(\pi_j)_{j \in [k]}$ and $(m_{ij})_{i \in [n], j \in [k]}$ (up to permuting $[k]$) to within high accuracy.
Identification is not always possible, as there can be disparate models with the same observable distributions. To set the conditions for identifiability, we need the following definition.

**Definition 2.** An observable $V_i$ is $\zeta$-separated if $\min_{j \neq j'} |m_{ij} - m_{ij'}| > \zeta$.

It is known that a mixture of product distributions (with $\pi_{\min} > 0$) is identifiable (from perfect statistics) if it has at least $2k - 1$ $0$-separated observables [TMMA18]; in general, $2k - 1$ are necessary [RSS14]. Clearly, getting a sample size bound requires also a quantitative assumption, hence the $\zeta$-separation assumption. Our running time and sample complexity guarantee require a slightly larger number of observables: $3k - 3$. It should be noted that the post-sampling runtime, although exponential in $k$, is hardly affected by $\zeta$. The bottleneck resource is sample size.

(A similar algorithm works with just $2k - 1$ $\zeta$-separated observables, and its post-sampling runtime is about the same, but its sample size requirement is larger.)

As in earlier works [CM19, GMRS21], the algorithm and its analysis make extensive use of the Hadamard extension of a matrix $m$.

**Definition 3.** The Hadamard product for row vectors $u = (u_1, \ldots, u_k)$, $v = (v_1, \ldots, v_k)$ is the mapping $\circ : \mathbb{R}^k \times \mathbb{R}^k \to \mathbb{R}^k$ given by $u \circ v := (u_1v_1, \ldots, u_kv_k)$. Equivalently, using the linear operator $v_\circ = \text{diag}(v)$, the Hadamard product is $u \circ v = u \cdot v_\circ$. The identity element for the Hadamard product is the all-ones vector $1$.

**Definition 4.** For $n \in \mathbb{R}^{[n] \times [p]}$, the Hadamard Extension of $n$, written $\mathbb{H}(n)$, is the $2^n \times p$ matrix with rows $n_S$ for all $S \subseteq [n]$, where, for $S = \{i_1, \ldots, i_k\}$, $n_S = n_{i_1} \circ \cdots \circ n_{i_k}$; equivalently $n_S = \prod_{i \in S} n_i$. In particular $n_{\emptyset} = 1$, and for all $i \in [n]$, $(\mathbb{H}(n))_{\{i\}} = n_i$.

**Definition 5.** $\mathbb{H}_p = \{\mathbb{H}(n) : n \in \mathbb{R}^{[k-1] \times [p]}\}$. (So $\mathbb{H}_1$ consists of rank-1 tensors of order $k - 1$.)

In order to prevent some confusions, in the rest of this section, for a matrix $A$ we denote by $A_{i*}$ its $i$-th row $i$ and by $A_{*j}$ its $j$-th column. Thus e.g., row $m_i$ becomes $m_{i*}$.

**Theorem 6.** Let $m$ have $k - 1$ rows, every row $\zeta$-separated, and let $M = \mathbb{H}(m)$. List the singular values of $M n_\circ$ in decreasing order: $\sigma_1 \geq \sigma_2 \geq \ldots$, and note $\sigma_{k+1} = \ldots = \sigma_{2^k-1} = 0$. Then

$$\sigma_k \geq \frac{(\min \pi_i)(\zeta/2\sqrt{5})^{k-1}}{\sqrt{k(k^2+1)^{k-1}}}.$$  

**Proof.** The proof relies on the following insight. Since $M$ has dimensions $2^{k-1} \times k$, $\sigma_k$ characterizes the least norm of $Mv$ ranging over any unit vector $v$, but it does not characterize the least norm of vectors of the form $hM$; the left-kernel of $M$ is of course very large. The insight is that it does become possible to bound $\sigma_k$ in terms of such vectors $h$, provided $h$ is restricted to rank 1 tensors.

Let $\tau(m, \pi) = \min_{0 \neq h \in H} \|h^TM \pi_\circ\|/\|h\|$. The proof relies on two lemmas.

**Lemma 7.** $\sigma_k(M \pi_\circ) \geq \tau(m, \pi)/\sqrt{k} \prod_{i=1}^{k-1}(1 + \|m_{i*}\|^2)$.

**Proof.** Consider $v \in \mathbb{R}^k$, $\|v\| = 1$, achieving $\sigma_k$, i.e., $r := \mathbb{H}(m) \pi_\circ v$ satisfies $\|r\| = \sigma_k$. W.l.o.g. $\|v_k\| \geq 1/\sqrt{k}$. Now $M_{ik} = \frac{1}{\sigma_k v_k} \left(r - \sum_{j=1}^{k-1} \pi_j v_j \mathbb{H}(m)_{*j}\right)$.
Now we carefully choose \( h \in \mathcal{H}_1 \) based on \( m \). Define \( n \in \mathbb{R}^{[k-1]} \) by \( n_i := -1/m_{ii} \); and let \( h := \left( \prod_{i=1}^{k-1} m_{ii} \right) \mathcal{H}(n) \). For \( j \neq k \) we have

\[
h^T \mathcal{H}(m)_{sj} = \left( \prod_{i=1}^{k-1} m_{ii} \right) \sum_S n_S m_{S,j} = \left( \prod_{i=1}^{k-1} m_{ii} \right) \sum_S (-1)^{|S|} \left( \prod_{i \in S} 1/m_{ii} \right) \left( \prod_{i \notin S} m_{ij} \right)
\]

\[
= \sum_S (-1)^{|S|} \left( \prod_{i \in S} m_{ii} \right) \left( \prod_{i \notin S} m_{ij} \right) = \prod_{i=1}^{k-1} (m_{ii} - m_{ij}) = 0.
\]

So, \( h^T \mathcal{H}(m)_{sj} \pi_j = 0 \) for \( j = 1, \ldots, k - 1 \). For \( j = k \) we can also show that \( h^T \mathcal{H}(m)_{sj} \pi_j \) is small:

\[
\left( h^T \mathcal{H}(m)_{\pi} \right)_k = \frac{1}{v_k} h^T \left( r - \sum_{j=1}^{k-1} \pi_j v_j \mathcal{H}(m)_{sj} \right) = \frac{1}{v_k} h^T r - \sum_{j=1}^{k-1} \pi_j v_j \left( h^T \mathcal{H}(m)_{sj} \right) = \frac{1}{v_k} h^T r.
\]

The norm of \( h^T \mathcal{H}(m)_{\pi} \) is then upper bounded by

\[
\| h^T \mathcal{H}(m)_{\pi} \| = \left| (h^T \mathcal{H}(m)_{\pi} )_k \right| = \left| \frac{1}{v_k} h^T \mathcal{H}(m) r \right| = \sqrt{k \| h \| \| r \|}
\]

\[
= \sqrt{k \| h \| \sigma_k} = \sigma_k \sqrt{k \prod_{i=1}^{k-1} (1 + m_{ii}^2)} \text{ due to } h \text{ being a rank 1 tensor } \square
\]

Next we lower bound \( \tau(m, \pi) \).

**Lemma 8.** \( \tau(m, \pi) \geq (\min \pi_i) (\zeta/2\sqrt{5})^{k-1} \).

**Proof.** Consider any \( G \in \mathcal{H}_1 \), say \( G = \mathcal{H}(g) \), \( g \in \mathbb{R}^{[k-1]} \). Then \( (G^T \mathcal{H}(m)_{\pi})_j = \pi_j \sum_S g_S m_{S,j} = \pi_j \prod_{i=1}^{k-1} (1 + g_i m_{ij}) \). We also note that \( \| G \| = \sqrt{\prod_{i=1}^{k-1} (1 + g_i^2)} \).

We now show that there is some \( j \) such that \( \prod_{i=1}^{k-1} \left| \frac{1 + g_i m_{ij}}{\sqrt{1 + g_i^2}} \right| \) is large. First, for any \( i \) for which \( g_i \geq \frac{1}{2} \), there is at most one \( j \) s.t. \( m_{ij} < \zeta \); exclude these \( j \)’s. Next, for each \( i \) for which \( g_i < \frac{1}{2} \), there is at most one \( j \) s.t. \( \left| \frac{1}{g_i} + m_{ij} \right| < \zeta/2 \); exclude these \( j \)’s. For the remainder of the argument fix any \( j \) which has not been excluded. Since \( m \) has \( k \) columns while \( g \in \mathbb{R}^{k-1} \), such a \( j \) exists. We consider three cases for \( i \). In each case we lower bound \( \left| \frac{1 + g_i m_{ij}}{\sqrt{1 + g_i^2}} \right| \).

1. \( g_i \geq 1/2, m_{ij} \geq \zeta \). Then \( \left| \frac{1 + g_i m_{ij}}{\sqrt{1 + g_i^2}} \right| \geq m_{ij} \geq \zeta \).
2. \(-1/2 \leq g_i < 1/2 \). Then \( \left| \frac{1 + g_i m_{ij}}{\sqrt{1 + g_i^2}} \right| \geq \sqrt{(m_{ij} - 2)^2} \geq 1/\sqrt{5} \). (This does not depend on \( j \).)
3. \( g_i < -1/2, |\frac{1}{g_i} + m_{ij}| \geq \zeta/2 \). Then \( \left| \frac{1 + g_i m_{ij}}{\sqrt{1 + g_i^2}} \right| = \left| \frac{g_i (\frac{1}{g_i} + m_{ij})}{|g_i|^{1/2} (1 + g_i^2)^{1/2}} \right| \geq \frac{\zeta}{2\sqrt{5}} \).

We therefore have \( \tau(m, \pi) \geq (\min \pi_i) (\zeta/2\sqrt{5})^{k-1} \). \( \square \)

Finally for the theorem combine the above lemmas and the fact that \( \| m_{ii} \| \leq k \) for any \( i \). \( \square \)
2.2 Algorithm for mixtures of product distributions: overview and theorem

We use the synthetic bits method introduced in [GMRS21]. Let \( V_S \) be the random variable \( \prod_{i \in S} V_i \). Our entire access to information about the source is that for every \( S \subseteq [n] \) we have an empirical estimate \( \hat{g}(S) \) of \( \mathbb{E}[V_S] \), defined as the fraction of samples in which \( \prod_{i \in S} V_i = 1 \). Notice that \( \mathbb{E}[V_S] = M_S \pi \circ \mathbbm{1} \) (where as previously, \( M = \oplus \{(m)\} \)). This is a multilinear moment of the \( m_S \). For the algorithm outlined here to produce \( m \) and \( \pi \) within additive error \( \varepsilon \) (in each coordinate), we shall need the \( \hat{g}(S) \) that we use to be accurate to within \( \varepsilon (\min \pi_i) O(\log k) C^{O(k \log k)} \). This can be achieved with high probability using a sample of size \( (1/\varepsilon)^2 (1/ \min \pi_i) O(\log k) (1/\zeta)^{O(k \log k)} \). To avoid introducing more notation here, the algorithm is described with respect to the exact statistics. The complete analysis in Appendix C includes the analysis of the effects of sampling error.

\( M \) has \( k \) columns. Thus, if \( n \) is not too small, it will have linear dependencies among its rows. We will create multiple “synthetic” copies of a single row of \( M \) by taking linear combinations of other rows. If each copy uses a distinct set of rows of \( m \) in the combinations that are added up, the corresponding synthetic variables are statistically independent (conditional on \( U \)). This enables computing higher moments of a random variable (in particular of the \( V_i \)) and ultimately reduces the mixtures of products problem to identification of mixtures of products of iid variables. This is an easier problem which can be solved within the required guarantees using either the matrix pencil method [KKM+19] or Prony’s method [GMRS20] (the differences between the two methods are inconsequential for the application here). Finally, after recovering a good estimate for one row and \( \pi \), it is a simple task to solve for the rest of the model.

The implementation of the above outline as follows. We need three disjoint sets \( S, T, T' \subseteq [n] \), each of size \( k-1 \), such that each of these \( 3k-3 \) variables are \( \zeta \)-separated. These can be found by enumeration over at most \( n^{O(k)} \) choices, and the rest of the algorithm needs to be repeated for each choice, until a solution is found.

Definition 9. For a matrix \( A \) and a set of row indices \( S \), denote by \( A[S] \) the sub-matrix derived from \( A \) by taking the rows in \( S \). The sum of two collections of subsets \( \mathcal{X}, \mathcal{Y} \subseteq 2^{[n]} \) is defined as \( \mathcal{X} + \mathcal{Y} := \{X \cup Y : X \in \mathcal{X}, Y \in \mathcal{Y}\} \). Let \( S, T, T' \subseteq [n] \) be two disjoint sets. Let \( x \) and \( y \), respectively, be vectors indexed by the subsets in \( X = 2^S \) and \( Y = 2^T \), respectively. The Kronecker product \( x \otimes y \) is the vector indexed by the subsets in \( \mathcal{X} + \mathcal{Y} \) given by \( (x \otimes y)_{X \cup Y} := x_X y_Y \), for all \( X \in \mathcal{X} \) and \( Y \in \mathcal{Y} \). Note that \( X + Y = 2^{S \cup T'} \). (Notice that every set \( Z \in \mathcal{X} + \mathcal{Y} \) can be written in this case uniquely as \( Z = X \cup Y \) for \( X \in \mathcal{X} \) and \( Y \in \mathcal{Y} \).)

Let \( A = 2^S \), \( B = 2^T \), and \( B' = 2^{T'} \). Obviously, Theorem 6 applies to all three matrices \( M[A], M[B], \) and \( M[B'] \). W.l.o.g., \( \{1\} \in B \). We generate the first \( 2k \) power moments of \( V_1 \) as follows. Define \( C_{B:A} = M[B] \pi \circ M[A]^\top \), and similarly \( C_{B':A} = M[B'] \pi \circ M[A]^\top \) and \( C_{B+B':A} = M[B+B'] \pi \circ M[A]^\top \). Notice that the entries of all of these matrices are multilinear moments, and hence we have made empirical estimates of these matrices with high probability. We will compute vectors \( v^{(1)}, v^{(2)}, \ldots, v^{(2k)} \in \mathbb{R}^A \) iteratively as follows. Initialize \( v^{(1)} = (g(S \cup \{1\}))_{S \in A} \). Notice that \( v^{(1)}_{\{1\}} = g(\{1\}) = m_1 \pi \circ \mathbbm{1} \). Then, for \( i = 2, 3, \ldots, 2k \), writing \( i = 2^r + j \) for \( j \in \{1, 2, \ldots, 2^r\} \), put

\[
v^{(i)} = \left( \left( v^{(j)} C_{B':A}^{-1} \right) \otimes \left( v^{(2^r)} C_{B:A}^{-1} \right) \right) C_{B+B':A}.
\]

A simple induction shows that \( v^{(1)}_{\{1\}} = m_1^{2^i} \pi \circ \mathbbm{1} \), the \( i \)-th power moment of \( V_1 \). The depth of the recurrence for computing the \( v^{(i)}_{\{1\}} \)'s does not exceed \( O(\log k) \), and this, along with Theorem 6, controls the accumulation of the error from the empirical values of the multilinear moments. Given the \( 2k \) moments of \( V_1 \), we can recover \( \pi \) and \( m_1 \) using, for instance, the above-mentioned result on
Prony’s method. The rest of $m$ can be recovered easily using this information and the empirical estimates of bilinear moments.

A detailed and rigorous description and analysis of the algorithm for mixtures of product distributions appears in Appendix C. In particular, it includes a proof of the following theorem, summarizing the result.

**Theorem 10.** There is an algorithm for identifying mixtures of $k$ product distributions on $n$ bits with the following guarantees. Let $\varepsilon > 0$ and suppose that at least $3k - 3$ observable bits are $\zeta$-separated. The algorithm runs in time $(1/\varepsilon)^2(1/\min \pi_i)^O(\log k)(1/\zeta)^O(k \log k)n^{O(k)}$. With high probability (over the distribution of the samples), the algorithm computes $\tilde{\pi}$ and $\tilde{m}$ such that $\max_i |\tilde{\pi}_i - \pi_i| \leq \varepsilon$ and $\max_{ij} |\tilde{m}_{ij} - m_{ij}| \leq \varepsilon$.

Comment: In Section 3, the general $k$-MixBND inference problem will rely on applications of this algorithm as a black box. The improvement given in this section is required in order to achieve the stated sample size and time complexity of solving the general case.

### 3 General Bayesian Networks

#### 3.1 Describing Bayesian networks

A Bayesian network consists of a directed acyclic graph (DAG) $G = (V, E)$, where $V$ is a set of random variables whose values are distributed according to a joint probability distribution $P$. The distribution $P$ must maintain the following consistency requirement with respect to $G$. For a vertex $V \in V$, let $\text{Pa}(V) = \{U \in V : (U, V) \in E\}$ be the set of the parents of $V$. (Similarly, let $\text{Ch}(V)$ denote the set of children of $V$.) Let $V = \{V_1, V_2, \ldots, V_n\}$ be an enumeration of the nodes of $G$. Then, $P$ factors into

$$P[V_1 = v_1, V_2 = v_2, \ldots, V_n = v_n] = \prod_{i=1}^{n} P[V_i = v_i | \text{pa}(V_i)],$$

where $\text{pa}(V_i)$ denotes the assignment $V_j = v_j$ for all $V_j \in \text{Pa}(V_i)$. In particular, for every $V \in V$, conditional on the values of $\text{Pa}(V)$, $V$ is independent of all other nodes which are non-descendants of $V$ in $G$.

To induce independence among vertices, we will use the concept of the Markov boundary [Pea14].

**Definition 11** (Markov Boundary). For a vertex $Y$ in a DAG $G = (V, E)$, the Markov boundary of $Y$, denoted $\text{Mb}(Y)$, is defined by

$$\text{Mb}(Y) := \text{Pa}(Y) \cup \text{Ch}(Y) \cup \text{Pa}(\text{Ch}(Y)) \setminus \{Y\}.$$  

**Definition 12.** We’ll introduce a parameter $\gamma(G)$ which will appear in the complexity of the identification procedure, which is defined by $\gamma(G) := \max_{V \in V} |\text{Mb}(V)|$.

**Observation 13.** For any $X, Y \in V$, $X \in \text{Mb}(Y) \iff Y \in \text{Mb}(X)$

Conditioning on the Markov boundary of a vertex allows us to ensure that it is $d$-separated, and thus independent, from the rest of the graph.

---

1This includes the sample size. The number of samples that is required is $(\log n/\varepsilon^2)(1/\min \pi_i)^O(\log k)(1/\zeta)^O(k \log k)$; each sample has $n$ bits.
Lemma 14 (See [Pea14]). For any vertex $V \in \mathcal{V}$ and subset $S \subseteq V \setminus (\text{Mb}(V) \cup \{V\})$, $\mathcal{P}(V \mid \text{Mb}(V), S) = \mathcal{P}(V \mid \text{Mb}(V))$.

The final definition will be useful when considering vertices in a topological ordering.

Definition 15 (Top). We will use $\text{Top}(V)$ to denote $\text{Mb}(V) \setminus \text{Ch}(V)$.

Observation 16. We can decompose

$$P(\text{mb}(Y), y) = P_u(\text{ch}^a(Y) \mid \text{top}^a(Y), y)P(y \mid \text{top}(Y))P_u(\text{top}^a(Y))$$

since $Y$ is $d$-separated from $\text{top}(Y) \setminus \text{pa}(Y)$ given $\text{pa}(Y)$.

Uppercase/lowercase conventions Following notation in causal inference literature, we will use lowercase letters to denote assignments. For example $P(v \mid u) = P(V = v \mid U = u)$. Following this convention, we will write $\text{pa}(V)$, $\text{ch}(V)$, and $\text{mb}(X)$ to denote assignment to the parents $\text{Pa}(V)$, children $\text{Ch}(V)$, and Markov boundary $\text{Mb}(V)$.

3.2 Applying a $k$-MixProd algorithm

Our algorithm will induce instances of independence through post-selected conditioning. We will then run a black box $k$-MixProd algorithm on this post-selected subset of the data. We will now formally define this concept.

3.2.1 Describing runs

Definition 17 (Run). A run over a graph $G = (\mathcal{V}, E)$ is a tuple $a = (\mathcal{I}^a, f^a)$ where $\mathcal{I}^a \subseteq \mathcal{V}$ are variables that we will focus on by conditioning on assignments to the set

$$\text{Cond}^a := (\cup_{X \in \mathcal{I}^a, d(X) < d_{\text{max}}^a} \text{Mb}(X)) \cup (\cup_{X \in \mathcal{I}^a, d(X) = d_{\text{max}}^a} \text{Pa}(X))$$

where $d_{\text{max}}^a := \max_{X \in \mathcal{I}^a} d(X)$. The value of the assignment is given by $f^a : \text{Cond}^a \rightarrow \{0, 1\}$. We will restrict our attention through the entire paper to well-formed runs, i.e. runs for which

$$\mathcal{I}^a \cap \text{Cond}^a = \emptyset.$$  

We’ll call $\mathcal{I}^a$ the independent set for $a$, and $\text{Cond}^a$ the conditioning set.

Definition 18 (Depth of a vertex). Given a DAG $G = (\mathcal{V}, E)$ and any vertex $V \in \mathcal{V}$, let $d_G(V)$ be the depth of $V$ in $G$, i.e. the length of the shortest path from a source vertex to $V$ in $G$. When $G$ is clear from context, we’ll omit the subscript.

Definition 19 (Bottom vertex in a run). Given a run $a = (\mathcal{I}^a, f^a)$, we’ll define the set of bottom vertices of the run to be the subset $\mathcal{F}^a \subseteq \mathcal{I}^a$ of vertices with maximal depth among the vertices in $\mathcal{I}^a$. 


Superscript notation. We’ll write \( mb^a(V) \), \( pa^a(V) \), \( ch^a(V) \) to refer to the assignment to the Markov boundary of \( V \), parents of \( V \), and children of \( V \) as set by run \( a \). In a similar spirit, we’ll occasionally write \( v^0 \) to denote the assignment \( V = 0 \).

**Definition 20** (Distribution induced by a run). For any run well-formed run \( a \), there is a distribution on the variables in \( \mathcal{I}^a \) induced by the run, which we’ll denote \( Pa(\cdot) \). The distribution is given by

\[
Pa(\cdot) = P(\cdot | cond^a),
\]

where \( cond^a \) is the assignment to \( Cond^a \) in keeping with our conventions.

The outputs of applying a \( k \)-MixProd algorithm to \( Pa(\mathcal{I}^a) \) are a matrix \( m^a \in [0,1]^{\mathcal{I}^a \times k} \) and a mixture vector \( \pi^a \in [0,1]^k \) (satisfying \( \sum_u \pi^a_u = 1 \)) defined by

\[
\begin{align*}
m^a_{iu} &:= P^a(X_i = 1 | U_a = u) = P(X_i = 1 | U_a = u, Cond^a), & \forall i \in \mathcal{I}^a, u \in [k] \\
\pi^a_u &:= P^a(U_a = u) = P(U_a = u | Cond^a), & \forall j \in [k],
\end{align*}
\]

where \( U_a \) is a source over \( [k] \) with a distribution that can be permuted to equal the distribution of \( U | Cond^a \). Note that because a mixture of products is invariant to a permutation of the labels of the mixture components, a mixture of products algorithm has total freedom to output any permutation of the labels for the source variable it’s considering. In other words, the labels of \( U_a \) correspond to a unique unknown label of \( U \). We will see how to handle this this in Section 3.4.

### 3.2.2 Assumptions

In order to apply a \( k \)-MixProd algorithm on the distribution \( Pa(\mathcal{I}^a) \) induced by a run, we need to ensure that the requirements of \( k \)-MixProd identifiability are satisfied.

Moreover, in order for us to be able to stitch together the information gleaned from multiple runs into a coherent mixture distribution over all of \( \mathcal{V} \), we’ll need to be able to uniquely determine a mapping from mixture constituents to probability assignments at each variable. Both of these requirements will be met by distributions satisfying the following criterion:

**Definition 21** (\( \zeta \)-informative mixture). Let \( P \) be distribution over a set \( \{0,1\}^\mathcal{V} \) that is a mixture of \( k \) distributions that are individually Markov with respect to a DAG \( G = (\mathcal{V}, E) \), i.e., \( \mathcal{P}_u(V) \) is Markov with respect to \( G \) for a variable \( U \in [k] \) with some unknown distribution. We’ll say that \( \mathcal{P} \) is \( \zeta \)-informative if for every variable \( V \in \mathcal{V} \),

1. for every assignment \( mb^a(V) \) to the Markov boundary of \( V \) we have that \( V \) is \( \zeta \)-separated in the distribution \( P(V | mb^a(V)) \) and that \( P(V | mb^a(V)) \in (\zeta, 1 - \zeta) \) for all \( u \in [k] \), and

2. for every assignment \( pa^a(V) \) to the parents of \( V \) we have that \( V \) is \( \zeta \)-separated in the distribution \( P(V | pa^a(V)) \)

**Lemma 22.** If \( \mathcal{P} \) is \( \zeta \)-informative, then for any assignment \( s \) to a subset of vertices \( S \subseteq V \), \( \mathcal{P}(s) \geq \zeta^{|S|} \).

**Proof.** Fix a topological ordering \( (X_1, \ldots, X_n) \) on the vertices in \( \mathcal{V} \). We’ll use induction on the latest variable appearing in \( S \), \( X_i \). Assume the claim holds for all subsets \( R \subseteq \{X_1, \ldots, X_{i-1}\} \), and write

\[
\mathcal{P}(s) = \mathcal{P}(s(X_i) | s_{(i-1)}) \mathcal{P}(s_{(i-1)}).
\]

\footnote{Any quantities parameterized by a run will take the parameter as a superscript.}
Now \[
P(s(X_i) \mid s_{(i-1)}) = \sum_{\text{pa}(X_i) \in \{0,1\} \text{pa}(X_i)} P(s(X_i) \mid \text{pa}(X_i)) P(\text{pa}(X_i) \mid s_{(i-1)}) \geq \zeta \sum_{\text{pa}(X_i) \in \{0,1\} \text{pa}(X_i)} P(\text{pa}(X_i) \mid s_{(i-1)}) = \zeta.
\]

Applying the induction hypothesis, we get that
\[
P(s) \geq \zeta \cdot \zeta^{|S|-1}.
\]

\[\square\]

**Definition 23.** We’ll say that a distribution over \(\mathcal{V}\) is *positive* if the probability of any event is > 0.

**Observation 24.** If \(P\) is \(\zeta\)-informative, then \(P\) is positive.

We will assume henceforth that our distribution \(P\) is a \(\zeta\)-informative mixture. This guarantees that the variables in \(\mathcal{T}^a\) are separated in \(P^a\) for any run \(a\) with \(|\mathcal{T}^a| \geq 3k - 3\), allowing us to run a \(k\)-MixProd algorithm.

Our algorithm will begin by executing a collection of runs of the \(k\)-MixProd algorithm \(A\). In order to succeed, we need to ensure that each of these runs is on at least 3\(k\) independent variables, motivating the following definition:

**Definition 25.** An individual run \(a = (\mathcal{T}^a, f^a)\) is \(N\)-independent if \(|\mathcal{T}^a| \geq N\).

### 3.3 Using different runs to learn source-specific parameters

In contrast to the \(k\)-MixProd setting, in the \(k\)-MixBND setting we are not always fortunate enough to have a sufficient number of independent variables. Instead, we will have to condition on variables in order to induce an independent set using \(d\)-separation rules. Specifically, our algorithm will consist of many calls to a black box \(k\)-MixProd algorithm, where each call looks at the distribution on a subset \(\mathcal{I} \subseteq \mathcal{V}\) conditioned on an assignment to the Markov boundaries of the variables in \(\mathcal{I}\) such that we \(d\)-separate each pair of variables.

### 3.4 Aligning source labels across different runs

Each run of the \(k\)-MixProd algorithm will return \(P^a(V \mid U_a = u)\) for some arbitrary permutation \(U_a\) of the variable. We need to align all of the outputs so that different outputs share the same permutation of \(U\). This can be done if we know ahead of time that a variable in one run’s independent set should have the same probability distribution as it does in the other run.

**Definition 26** (Aligned mixture distributions). A pair of \(k\)-component mixture distributions \(P^{(1)}, P^{(2)}\) over tuples of random variables \(U^{(1)} = (X_1^{(1)}, \ldots, X_m^{(1)})\) and \(U^{(2)} = (X_1^{(2)}, \ldots, X_m^{(2)})\), respectively, is alignable if there exist \(i, i \in [m]\) such that \(P_u(X_{i_1}^{(1)}) = P_u(X_{i_2}^{(2)})\) for all \(u \in [k]\) and \(X_1^{(1)}\) is separated with respect to \(U\). We’ll call the random variable \(X_{i_1}^{(1)}\) (which is the same as \(X_{i_2}^{(2)}\)) the *alignment variable*.

**Definition 27** (Alignment spanning tree). We say a set of \(\ell\) mixture distributions \(P^{(1)}, \ldots, P^{(\ell)}\), is alignable if there exists an undirected spanning tree with edges \(E \subseteq \{(P^{(1)}, \ldots, P^{(\ell)}) \times \{P^{(1)}, \ldots, P^{(\ell)}\}\) where each edge corresponds to \(\text{AV}(P^{(i)}, P^{(j)})\), variables at which \(P^{(i)}, P^{(j)}\) are aligned.
The alignment step will take the output from a number of alignable runs and permute the mixture labels, assigning parameters to a global set of sources. Pseudocode for this step is given in Algorithm 1.

**Algorithm 1 Alignment**

**Input:** A set of distributions $P^{(1)}, \ldots, P^{(t)}$ with $P^{(i)}$ a distribution with parameters $P^{(i)}(x^{(i)} | u^{(i)})$ for $X^{(i)} \in \mathcal{X}^{(i)}$ and $P(u^{(i)})$. In addition, we have a spanning tree with edges $E$ and alignment variables $AV(P^{(i)}, P^{(j)}) \in \mathcal{X}^{(i)} \times \mathcal{X}^{(j)}$.

**Output** $P_u(X_i^{(j)})$ for $X_i^{(j)} \in \mathcal{X}^{(j)}$ for $j \in [\ell]$. Also $P(u)$ for each $u \in [k]$.

1. Let $T$ be an oriented tree on the undirected spanning tree with edges $E$ with all vertices having a directed path to an arbitrary fixed vertex $P^{(t)}$.
2. for each edge $P^{(i)} \rightarrow P^{(j)}$ in $T$ and corresponding alignment variables $AV(P^{(i)}, P^{(j)}) = (X_{AV}^{(i)}, X_{AV}^{(j)})$ do
   3. Let $\sigma^{i \rightarrow j}$ be the permutation on the sources that minimizes $\left\| P^{(i)}(X_{AV}^{(i)} | \sigma^{i \rightarrow j}(u^{(i)})) - P^{(j)}(X_{AV}^{(j)} | u^{(j)}) \right\|_\infty$.
4. end for
5. for each vertex $P^{(s)}$ do
6. Let $p = (s, p_1, \ldots, p_r, t)$ be the directed path in $T$ from $P^{(s)}$ to $P^{(t)}$.
7. Let $\sigma^{s \rightarrow t} = \sigma^{s \rightarrow p_1} \cdot \sigma^{p_1 \rightarrow p_2} \cdots \sigma^{p_r \rightarrow t}$ be the composition of permutations along the path from $P^{(s)}$ to $P^{(t)}$.
8. Set $P_u(X_i^{(s)}) \leftarrow P^{(s)}(X_i^{(s)} | \sigma^{s \rightarrow t}(u^{(s)}))$ for all $i \in [n]$.
9. Set $P(u) \leftarrow P^{(s)}(\sigma^{s \rightarrow t}(u^{(s)}))$
10. end for

### 3.5 Bayesian unzipping: recovering parameters per source

Recall that our algorithm uses Markov boundary conditioning to induce independent variables. Hence, after aligning the sources in runs of the $k$-MixProd algorithm we will have access to $P_u(Y \mid Mb(Y))$ for each $Y \in V$. Our goal is to obtain $P_u(Y \mid Pa(Y))$. Note that

$$P_u(y^1 \mid mb(Y)) = \frac{P_u(y^1, mb(Y))}{P_u(y^1, mb(Y)) + P_u(y^0, mb(Y))} \quad (3)$$

The terms in this fraction are all of the same form and can be factored according to the DAG into

$$P_u(y, mb^a(Y)) = P_u(top(Y))P_u(y \mid pa^a(Y)) \prod_{V \in Ch(Y)} P_u(u^a \mid f^a(Pa(V) \setminus \{Y\}), y).$$

See Figure 3 for a concrete example of this decomposition. After substitution into (3) we see that $P_u(top(Y))$ appears in both the numerator and denominator because it is independent of the assignment to $Y$. We can cancel these terms out leaving only the following terms:

1. $P_u(y^0 \mid pa^a(Y))$ and $P_u(y^1 \mid pa^a(Y))$, which must sum to 1.

2. $P_u(ch^a(Y) \mid top^a(Y), y^0)$ and $P_u(ch^a(Y) \mid top^a(Y), y^1)$ which are both the product of the desired parameters of variables later in the topological ordering. We can ensure we have access to these terms by solving for the parameters of $V \in \mathcal{V}$ in a reverse-topological ordering.
We can substitute $1 - \mathbb{P}_u(y^1 \mid \text{pa}^a(Y))$ for $\mathbb{P}_u(y^0 \mid \text{pa}^a(Y))$ in the expanded version of (3) to obtain a single equation with only $\mathbb{P}_u(y^1 \mid \text{pa}^a(Y))$ as an unknown, which we can then solve. The pseudocode for this process is given in Algorithm 2.

Figure 3: We can decompose $\mathbb{P}_u(v_1, v_2, v_3, y, v_4, v_5) = \mathbb{P}_u(v_1, v_2, v_3) \mathbb{P}_u(y \mid v_1, v_2) \mathbb{P}_u(v_4 \mid y, v_3) \mathbb{P}_u(v_5 \mid y, v_4)$. This figure is restricted to $Mb(Y) \cup \{Y\}$. $U$ and any other variables in the graph are omitted for clarity.

**Algorithm 2 Bayesian Unzipping**

**Input:** A collection of runs $\mathcal{A}$ of size at most $2^O(\Delta^2)$ and their aligned output. For each $V_i$ and assignment to its parents $\text{pa}(V_i)$ there must be some run with $V_i$ in its independent set with parents conditioned to $\text{pa}(V_i)$.

**Output** $\mathbb{P}_u(Y \mid \text{Pa}(Y))$

1: Fix a topological ordering on the vertices in $V$, $\langle X_1, X_2, \ldots, X_n \rangle$.
2: for $i = n, n-1, \ldots, 1$ do
3: for each assignment $\text{pa}(X_i)$ to $\text{Pa}(X_i)$ do
4: Let $a$ be a run in $\mathcal{A}$ with $\text{pa}^a(X_i) = \text{pa}(X_i)$.
5: for $u = 1, \ldots, k$ do
6: for $b = 0, 1$ do
7: if $X_i$ is a bottom vertex then
8: Set $\mathbb{P}_u(x^b_i \mid \text{pa}^a(X_i)) \leftarrow \mathbb{P}_u^a(x^b_i)$.
9: else
10: Set $\mathbb{P}_u(x^b_i \mid \text{pa}(X_i)) \leftarrow \frac{\mathbb{P}_u^a(x^b_i) \mathbb{P}_a(\text{ch}^a(X_i) \mid \text{top}^a(X_i), x_i^{1-b})}{\mathbb{P}_u^a(x^b_i) \mathbb{P}_a(\text{ch}^a(X_i) \mid \text{top}^a(X_i), x_i^{1-b}) + \mathbb{P}_u(x_i^{1-b}) \mathbb{P}_a(\text{ch}^a(X_i) \mid \text{top}^a(X_i), x_i^1)}$.
11: end if
12: end for
13: end for
14: end for
15: end for

3.5.1 Recovering the distribution on sources

Now consider some arbitrary run $a$ of the $k$-MixProd algorithm with conditioning $\text{cond}^a$. Since $\mathbb{P}_a(V) = \prod_{V \in Y} \mathbb{P}_u(V \mid \text{Pa}(V))$, knowing $\mathbb{P}_a(V \mid \text{Pa}(V))$ gives us access to all of $\mathbb{P}_a(V)$ after Bayesian unzipping. Thus, we can compute $\mathbb{P}_a(\text{cond}^a) = \mathbb{P}(\text{cond}^a \mid u)$. The $k$-MixProd algorithm will return $\mathbb{P}^a(U) = \mathbb{P}(U \mid \text{cond}^a)$ when run on $a$ (after source alignment). Finally, $\mathbb{P}(\text{cond}^a)$ is directly observable. Combining these terms in Bayes’ rule lets us compute the distribution on $U$:

$$\mathbb{P}(u) = \frac{\mathbb{P}(u \mid \text{cond}^a) \mathbb{P}(\text{cond}^a)}{\mathbb{P}(\text{cond}^a \mid u)}$$

This equation is well defined under the assumption of positivity.
3.6 A good collection of runs

We must also be careful to ensure all of these runs can have their sources aligned and that all the desired parameters can be obtained via Bayesian unzipping.

**Definition 28** (Runs aligned at $X$/alignable pair of runs). A pair of runs $a, b$ are **aligned at $X \in \mathcal{V}$** if the distributions on $\mathcal{I}^{a} | \text{cond}^{a}$ and $\mathcal{I}^{b} | \text{cond}^{b}$ are aligned mixture distributions with alignment variable $X | \text{cond}^{a}$. We say that $a, b$ are **alignable** if there is any variable aligning them.

**Observation 29.** Two runs $a, b$ are aligned at $X \in \mathcal{V}$ if and only if

1. $X \in \mathcal{I}^{a} \cap \mathcal{I}^{b}$,
2. $\text{mb}^{a}(X) = \text{mb}^{b}(X)$, i.e, $f^{a}(\text{Mb}(X)) = f^{b}(\text{Mb}(X))$, and
3. $X$ is separated given $\text{mb}^{a}(X)$ (equivalently, given $\text{mb}^{b}(X)$).

**Definition 30.** A collection of runs $\mathcal{A}$

- is **alignable** if the undirected alignment graph $H(\mathcal{A}) = (\mathcal{A}, E)$ with edges given by $E = \{(a, b) : a$ and $b$ are alignable$\}$ has a single connected component
- **covers** $X \in \mathcal{V}$ if for every assignment $\mathcal{pa}(X)$ to $\mathcal{Pa}(X)$ there exists a run $a \in \mathcal{A}$ with $X \in \mathcal{I}^{a}$ and $\mathcal{pa}(X) = \mathcal{pa}^{a}(X)$.

Our algorithm will require **good** collection of runs:

**Definition 31** (A good collection of runs). A collection of well-formed runs $\mathcal{A}$ is **good** if it is

- alignable,
- every run is $(3k - 3)$-independent,
- the collection covers every vertex in $\mathcal{V}$,
- no vertex appears both as a bottom vertex and a non-bottom vertex, and
- every non-bottom vertex has depth at most $9k$.\(^4\)

**Observation 32.** In a good collection of runs, we can partition all vertices into bottom vertices and non-bottom vertices for the entire collection.

**Definition 33** (Centers, Central Runs). A set of vertices $\mathcal{X} = \{X_{1}, \ldots, X_{3k-3}\} \subseteq \mathcal{V}$ will be called **centers** if the Markov boundaries of the vertices in $\mathcal{X}$ are disjoint. Given a set of centers $\mathcal{X}$, a run $a$ is called a **central run** if $\mathcal{I}^{a} = \mathcal{X}$.

The main result on good collections of runs that we’ll utilize is the following:

**Lemma 34.** Given a graph with max degree $\Delta$ satisfying $n \geq (3k - 3) \times O(\Delta^4)$, we can find a set of centers $\mathcal{X} = \{X_{1}, \ldots, X_{3k-3}\} \subseteq \mathcal{V}$ of size $(3k - 3)$ and depth at most $9k$, such that by running Algorithm 4, we obtain a good collection of runs $\mathcal{A}$ of size $O(2 \Delta^2 n)$ assuming that the mixture $\mathcal{P}$ is $\zeta$-informative.

The proof of this lemma is in Appendix A, along with other sufficient conditions for the existence of good collections of runs.

\(^4\)The choice of $9k$ can be tightened to $9k - 9$, which we suppress.
3.7 Outline of the algorithm

Our algorithm will take in a good collection of runs $\mathcal{A}$ satisfying the size constraint obtained by Lemma 34. While any good collection of runs will suffice for the correctness of our algorithm for sufficiently large sample complexity, having this bound on the collection size ensures that our sample complexity bounds are sufficient.

Essentially our algorithm has four steps:

1. Use a $k$-MixProd algorithm on $P(\mathcal{I}^a \mid \text{Cond}^a)$ for each run $a \in \mathcal{A}$ to compute $P(Y \mid \text{Mb}(Y), U_a = u)$ for all variables $Y \in \mathcal{V}$.

2. Align the parameters obtained from the previous step to ensure that $U$ means the same thing across different runs, giving $P_u(Y \mid \text{Mb}(Y))$.

3. Work backwards in topological order to recover $P_u(Y \mid Pa(Y))$ for each vertex $Y \in \mathcal{V}$ via Bayesian unzipping.

4. Compute $P(U)$ by applying Bayes’ law.

The full procedure appears as Algorithm 3.

Algorithm 3 The $k$-MixBND algorithm

**Input:** A good collection of runs $\mathcal{A}$ of size at most $2^{O(\Delta^2)}$; a target accuracy $\varepsilon$.

**Output** $\tilde{P}_u(Y \mid \text{Pa}(Y))$ and $\tilde{P}(U)$.

1. Estimate $\tilde{P}^{\text{a}}(\mathcal{I}^a)$ for all runs $a \in \mathcal{A}$ using

   $$n \log n \cdot \varepsilon^{-2(\Delta + 1)}2^{O(\Delta^2 + \Delta k)}(\min \pi_i)^{-O(\log k)}(1/\zeta)^{O(k \log k + \Delta^2 k)}$$

   samples from $P$.

2. Set $\varepsilon' \leftarrow \varepsilon/(6(\Delta + 1))^{18k}$.

3. for each run $a \in \mathcal{A}$ do

4. Let $Q := \tilde{P}^{\text{a}}(\mathcal{I}^a)$.

5. Set $([m_{ui}^a])_{u \in [k]} \cdot (\pi_u^a)_{u \in [k]} := \text{LEARNPRODUCTMIXTURE}(Q, \varepsilon')$, where $m_{ij}^a = Q_a(x^a_i) = P_u(x^a_i \mid \text{cond}^a)$ and $\pi_u^a = P(U = u \mid \text{cond}^a)$.

6. end for

7. Run Algorithm 1 to align the sources in the output of all the runs in $\mathcal{A}$.

8. Run Algorithm 2 to unzip the parameters.

9. Fix any run $a \in \mathcal{A}$.

10. for $j = 1, 2, \ldots, k$ do

11. Set $\tilde{P}(u^j) \leftarrow \tilde{P}(u^j \mid \text{cond}^a) \tilde{P}(\text{cond}^a) / \tilde{P}(\text{cond}^a | u^j)$.

12. end for

3.8 Statement of main result for $k$-MixBND

**Theorem 35.** If $P(V) = \sum_u P_u(V)P(u)$ is a $\zeta$-informative mixture of $k$ distributions over a DAG $G = (\mathcal{V}, E)$ and there exists a set of $3k - 3$ centers, then there is an algorithm to compute estimates of all parameters to accuracy $\varepsilon$ with sample complexity (and runtime) bounded by

$$n \log n \cdot \varepsilon^{-2(\Delta + 1)}2^{O(\Delta^2 + \Delta k)}(\min \pi_i)^{-O(\log k)}(1/\zeta)^{O(k \log k + \Delta^2 k)}.$$
4 Analyzing the $k$-MixBND algorithm

4.1 Applying the $k$-MixProd algorithm on each run

We must collect the moments needed for each application of the $k$-MixProd algorithm and the accuracy with which the moments are estimated will show up in the output of the $k$-MixProd algorithm:

**Lemma 36.** Suppose that we fix an $\varepsilon$ target accuracy and that $\mathcal{P}$ is $\zeta$-informative. Then with total sample complexity

$$(n \log n/\varepsilon^2)(\Delta + 1)^{O(k)}2^{O(\Delta^2 + \Delta k)}(1/\min \pi_i)^{O(\log k)}(1/\zeta)^{O(k \log k + \Delta^2 k)},$$

we can ensure that we have estimates of $\tilde{\mathcal{P}}^a(\mathcal{I}^a)$ sufficiently accurate so that $\text{LearnProductMixture}(\mathcal{P}^a(\mathcal{I}^a), \varepsilon')$ (where $\varepsilon' = \varepsilon \zeta/(6(\Delta + 1))^{9k}$) will return $\tilde{\mathcal{P}}^a(\mathcal{I}^a)$ and $\mathcal{P}^a(u)$ satisfying

$$\left|\tilde{\mathcal{P}}^a(y^b \mid pa^a(Y)) - \mathcal{P}^a(y^b \mid pa^a(Y))\right| \leq \varepsilon \zeta/(6(\Delta + 1))^{9k}$$

and

$$\left|\tilde{\mathcal{P}}^a(u) - \mathcal{P}^a(u)\right| \leq \varepsilon \zeta/(6(\Delta + 1))^{9k}$$

for all $Y \in \mathcal{I}^a$, $b \in \{0, 1\}$, $u \in [k]$ in time $(6(\Delta + 1))^{9k}\varepsilon^{-2}(1/\min \pi_i)^{O(\log k)}(1/\zeta)^{O(k \log k)}n^{O(k)}$. Consequently, we will also have

$$\left|\tilde{\mathcal{P}}^a(y^b \mid pa^a(Y)) - \mathcal{P}^a(y^b \mid pa^a(Y))/\mathcal{P}^a(y^b \mid pa^a)\right| \leq \varepsilon/(6(\Delta + 1))^{9k}$$

and

$$\left|\tilde{\mathcal{P}}^a(u) - \mathcal{P}^a(u)/\mathcal{P}^a(u)\right| \leq \varepsilon/(6(\Delta + 1))^{9k}.$$

**Proof.** With the given sample complexity we have that the estimates of $\tilde{\mathcal{P}}(\mathcal{I}^a)$ are within accuracy $\varepsilon \zeta/(6(\Delta + 1))^{9k}$ for each run $a \in \mathcal{A}$ with high probability. The conclusion follows from the first part of Theorem 1 on the behavior of the algorithm $\text{LearnProductMixture}$. Note that we require Lemma 22 in order for the expected number of samples to observe one sample with $\text{cond}^a = \text{Cond}^a$ to be bounded above by $\zeta^{O(\Delta^2 k)}$ for any run $a \in \mathcal{A}$. \qed

4.2 Analysis of alignment

**Lemma 37.** Suppose $\mathcal{P}(V)$ is a $\zeta$-informative mixture of $k$ Bayesian networks. Given an alignable collection of runs $\mathcal{A}$ and the outputs of a $k$-MixProd algorithm $((m^a_u), (\pi^a_u))$ for each run $a \in \mathcal{A}$ satisfying

$$|\pi^a_u - \mathcal{P}^a(u)| < \zeta/2, \quad |m^a_{iu} - \mathcal{P}^a(u(x^1))| < \zeta/2 \quad \text{for all } u \in [k], a \in \mathcal{A}.$$ 

Algorithm 1 returns $\mathcal{P}^a(X_i = 1)$ for $i \in [n]$ and $\mathcal{P}^a(u)$.

**Proof.** There is a unique permutation $\sigma^{i\rightarrow j}$ such that

$$\left\|\mathcal{P}^{(i)}(X^{(i)}_{\mathcal{A}^i} \mid \sigma^{i\rightarrow j}(u^{(i)})) - \mathcal{P}^{(j)}(X^{(j)}_{\mathcal{A}^j} \mid u^{(j)})\right\|_{\infty} < \frac{\zeta}{2}$$

The algorithm chooses this permutation and aligns accordingly. \qed

4.3 Recovering Parameters

**Lemma 38.** Let $a$ be a run with $Y \in \mathcal{I}^a$ and let $\mathcal{P}$ be $\zeta$-informative for $\zeta > 0$. Then we can compute $\mathcal{P}^a(y^b \mid pa^a(Y))$ for $b \in \{0, 1\}$ as follows:

1. If $Y \in \mathcal{F}_a$, then $\mathcal{P}^a(y^b \mid pa^a(Y)) = \mathcal{P}^a(y^b)$. 

16
2. If \( Y \not\in \mathcal{J}_a \), then
\[
\mathcal{P}_u(y^b | pa^a(y)) = \frac{\mathcal{P}_u(y^b)\mathcal{P}_u(ch^a(y) | \text{top}^a(y), y^{1-b})}{\mathcal{P}_u(y^b | \text{top}^a(y), y^{1-b}) + \mathcal{P}_u(y^{1-b})\mathcal{P}_u(ch^a(y) | \text{top}^a(y), y^b)}
\] (4)

**Proof.** In the following we’ll fix \( ch^a := ch^a(Y), mb^a := mb^a(Y), pa^a := pa^a(Y) \), and \( \text{top}^a := \text{top}^a(Y) \). If \( Y \in \mathcal{J}_a \), then \( \mathcal{P}_u(y^b) = \mathcal{P}_u(y^b | pa^a(Y)) \), so the claim is trivially true. If \( Y \not\in \mathcal{J}_a \), then
\[
\mathcal{P}_u(y^b) = \frac{\mathcal{P}_u(y^b | mb^a)}{\mathcal{P}_u(mb^a)} = \frac{\mathcal{P}_u(y^b, mb^a)}{\mathcal{P}_u(mb^a)}
\]
and we can expand \( \mathcal{P}_u(y^b, mb^a) \) using Observation 16 as
\[
\mathcal{P}_u(y^b, mb^a) = \mathcal{P}_u(y^b | pa^a)\mathcal{P}_u(ch^a | \text{top}^a, y^b)\mathcal{P}_u(top^a).
\]
Substituting this back into the preceding equation, we obtain
\[
\mathcal{P}_u(y^b) = \frac{\mathcal{P}_u(y^b | pa^a)\mathcal{P}_u(ch^a | \text{top}^a, y^b)\mathcal{P}_u(top^a)}{\sum_{b' \in \{0,1\}} \mathcal{P}_u(y^{b'}, \text{pa}^a)\mathcal{P}_u(ch^a | \text{top}^a, y^{b'})}.
\]

We now multiply both sides by the denominator (which is non-zero since all terms are strictly positive by assumption) and then simplify to obtain
\[
\mathcal{P}_u(y^b | pa^a)\left(\mathcal{P}_u(y^{1-b})\mathcal{P}(ch^a | \text{top}^a, y^b)\right) + \mathcal{P}_u(y^b | pa^a)\left(-\mathcal{P}_u(y^{1-b})\mathcal{P}(ch^a | \text{top}^a, y^{1-b})\right) = 0.
\]

When augmented with the equation
\[
\mathcal{P}_u(y^b | pa^a) + \mathcal{P}_u(y^{1-b} | pa^a) = 1
\]
we have a system of two equations in \( \mathcal{P}_u(y^b | pa^a), \mathcal{P}_u(y^{1-b} | pa^a) \) which is non-singular whenever
\[
\left(\mathcal{P}_u(y^{1-b})\mathcal{P}(ch^a | \text{top}^a, y^b)\right) \neq \left(-\mathcal{P}_u(y^{1-b})\mathcal{P}(ch^a | \text{top}^a, y^{1-b})\right).
\]
Since all probabilities occurring are strictly positive, this will always be the case and we can solve the system for \( \mathcal{P}_u(y^b | pa^a) \). The resulting equation is
\[
\mathcal{P}_u(y^b | pa^a) = \frac{\mathcal{P}_u(y^b)\mathcal{P}(ch^a | \text{top}^a, y^{1-b})}{\mathcal{P}_u(y^b)\mathcal{P}(ch^a | \text{top}^a, y^{1-b}) + \mathcal{P}_u(y^{1-b})\mathcal{P}(ch^a | \text{top}^a, y^b)},
\]
proving the claim.

The following standard inequalities will be useful throughout the analysis:

**Observation 39.**
\[
\frac{1+x}{1-x} \leq 1 + 3x \quad \text{and} \quad 1 - 3x \leq \frac{1-x}{1+x} \quad \text{for } x \in (0,1/4);
\]
\[
1 - 2rx \leq (1 - x)^r \quad \text{and} \quad (1 + x)^r \leq 1 + 2rx \quad \text{for } x \in (0,1), r \geq 1, rx \leq 1.
\]
Lemma 40. Given access to \( \{ \tilde{\mathcal{P}}_a^u(Y) \}_{a \in A, i \in [k]} \) for a good collection of runs \( A \) from a \( \zeta \)-informative distribution \( \mathcal{P} \) satisfying 
\[
| \tilde{\mathcal{P}}_a^u(y^b) - \mathcal{P}_a^u(y^b) | / \mathcal{P}_a^u(y^b) \leq \varepsilon
\]
for all \( Y \in \mathcal{V} \), \( b \in \{0, 1\} \), \( u \in [k] \) for \( \varepsilon \) sufficiently small (and the estimated probabilities used as input to \( k \)-MixProd algorithms), the procedure in Algorithm 2 will output \( \tilde{\mathcal{P}}_a(y^b | \mathbf{Pa}(Y)) \) and \( \tilde{\mathcal{P}}(u) \) for all \( Y \in \mathcal{V} \), \( b \in \{0, 1\} \), \( u \in [k] \) satisfying
\[
\frac{| \tilde{\mathcal{P}}_u(y^b | \mathbf{pa}(Y)) - \mathcal{P}_u(y^b | \mathbf{pa}(Y)) |}{\mathcal{P}_u(y^b | \mathbf{pa}(Y))} \leq (6(\Delta + 1))^{\ell} \varepsilon
\]
where \( \ell \) is the distance from \( Y \) to the nearest bottom vertex if \( Y \) doesn’t appear as a bottom vertex and is 0 if \( Y \) is a bottom vertex.

Proof. We fix a topological ordering on \( \mathcal{V} \), let’s say \( X_1, X_2, \ldots, X_n \). Now starting with the last vertex in topological order, \( X_n \), and proceeding in decreasing order, we’ll compute \( \tilde{\mathcal{P}}_u(X_i | \mathbf{Pa}(X_i)) \). For bottom vertices, we set \( \tilde{\mathcal{P}}_u(x_i^b | \mathbf{pa}(X_i)) = \tilde{\mathcal{P}}_a^u(x_i^b) \) for some \( a \) with \( \mathbf{pa}(X_i) = \mathbf{pa}^a(X_i) \). It immediately follows that \( \mathcal{P}_u(X_i | \mathbf{Pa}(X_i)) \) satisfies the desired relative error bound. Inductively, assume we’ve already computed \( \tilde{\mathcal{P}}_u(V_m | \mathbf{Pa}(V_m)) \) for all \( m > i \) satisfying the stated bounds, and that we also have access to \( \tilde{\mathcal{P}}_a^u(V_i) = \tilde{\mathcal{P}}_u(V_i | \mathbf{Cond}^a) \) for a subset of the runs \( a \in A \) that cover \( V_i \). To streamline notation, let \( Y := V_i \). Now fix a run \( a \in A \) with \( Y \in T^a \).

Now we can bound each term above and below using the inductive hypothesis to get the desired result. In particular, we know that
\[
(1 - \varepsilon)\mathcal{P}_u^a(y^b) \leq \tilde{\mathcal{P}}_u^a(y^b) \leq (1 + \varepsilon)\mathcal{P}_u^a(y^b)
\]
for \( b \in \{0, 1\} \) and
\[
(1 - (6(\Delta + 1))^{\ell-1} \varepsilon)\mathcal{P}_u(v^b | \mathbf{pa}^a(v^b)) \leq \tilde{\mathcal{P}}_u(v^b | \mathbf{pa}^a(v^b)) \leq (1 + (6(\Delta + 1))^{\ell-1} \varepsilon)\mathcal{P}_u(v^b | \mathbf{pa}^a(v^b))
\]
for all \( V \in \mathbf{Ch}(Y) \) and \( b \in \{0, 1\} \) which implies that both the numerator and denominator of (4) are within a factor of \( (1 \pm \varepsilon)(1 \pm (6(\Delta + 1))^{\ell-1} \varepsilon)^\Delta \) of the correct value for those terms. It immediately follows that \( \tilde{\mathcal{P}}_u(y^b | \mathbf{pa}^a(Y)) \) is bounded by
\[
\mathcal{P}_u(y^b | \mathbf{pa}^a(Y)) \frac{1 - \varepsilon}{1 + \varepsilon} \left( \frac{1 - (6(\Delta + 1))^{\ell-1} \varepsilon}{1 + (6(\Delta + 1))^{\ell-1} \varepsilon} \right) \leq \tilde{\mathcal{P}}_u(y^b | \mathbf{pa}^a(Y)) \leq \mathcal{P}_u(y^b | \mathbf{pa}^a(Y)) \frac{1 + \varepsilon}{1 - \varepsilon} \left( \frac{1 + (6(\Delta + 1))^{\ell-1} \varepsilon}{1 - (6(\Delta + 1))^{\ell-1} \varepsilon} \right).
\]

We now use the inequalities from Observation 39 to simplify the bounds as follows:
\[
\tilde{\mathcal{P}}_u(y^b | \mathbf{pa}^a(Y)) \leq \mathcal{P}_u(y^b | \mathbf{pa}^a(Y)) \frac{1 + \varepsilon}{1 - \varepsilon} \left( \frac{1 + (6(\Delta + 1))^{\ell-1} \varepsilon}{1 - (6(\Delta + 1))^{\ell-1} \varepsilon} \right)^\Delta
\]
\[
\leq \mathcal{P}_u(y^b | \mathbf{pa}^a(Y))(1 + 3\varepsilon)(1 + 3(6(\Delta + 1))^{\ell-1} \varepsilon)^\Delta
\]
\[
\leq \mathcal{P}_u(y^b | \mathbf{pa}^a(Y))(1 + 3(6(\Delta + 1))^{\ell-1} \varepsilon)^{\Delta + 1}
\]
\[
\leq \mathcal{P}_u(y^b | \mathbf{pa}^a(Y))(1 + 2 \cdot 3(\Delta + 1)(6(\Delta + 1))^{\ell-1} \varepsilon)
\]
\[
\leq \mathcal{P}_u(y^b | \mathbf{pa}^a(Y))(1 + (6(\Delta + 1))^{\ell} \varepsilon).
\]
The lower bound is analogous. \( \square \)
Lemma 41. Given access to \( \{ \widetilde{P}^a_u(Y) \} \) for a good collection of runs \( A \) from a \( \zeta \)-informative distribution \( P \) satisfying \( | \widetilde{P}_u^a(y^b) - P_u^a(y^b) | / P_u^a(y^b) \leq \varepsilon \) for all \( Y \in \mathcal{Z} \), \( b \in \{0, 1\} \), \( u \in [k] \) for \( \varepsilon \) sufficiently small, the procedure in Algorithm 2 will output \( \widetilde{P}_u(y^b \mid PA(Y)) \) and \( \widetilde{P}(u) \) for all \( Y \in \mathcal{Y}, b \in \{0, 1\}, u \in [k] \) satisfying

\[
\frac{\left| \widetilde{P}_u(y^b \mid PA(Y)) - P_u(y^b \mid PA(Y)) \right|}{P_u(y^b \mid PA(Y))} \leq (6(\Delta + 1))^{9k}\varepsilon
\]

and

\[
\frac{\left| \widetilde{P}(u^j) - P(u^j) \right|}{P(u^j)} \leq 5\Delta^2\varepsilon.
\]

Since \( P_u(y^b \mid PA(Y)) \), \( P(u^j) \leq 1 \), we also have that

\[
\left| \widetilde{P}_u(y^b \mid PA(Y)) - P_u(y^b \mid PA(Y)) \right| \leq (6(\Delta + 1))^{9k}\varepsilon \quad \text{and} \quad \left| \widetilde{P}(u^j) - P(u^j) \right| \leq 5\Delta^2\varepsilon.
\]

Proof. The error bound on \( \widetilde{P}_u(y^b \mid PA(Y)) \) follows from Lemma 40 above and the depth bound of \( 9k \) on non-bottom vertices.

For the error bound on \( \widetilde{P}(u^j) \) we write

\[
\widetilde{P}(u^j) = \frac{\widetilde{P}(u^j \mid \text{cond}^a)\widetilde{P}(\text{cond}^a)}{\widetilde{P}(\text{cond}^a \mid u^j)}
\]

and analyze each term separately. First, define \( Z \) := \text{An}((\text{cond}^a) \setminus \text{cond}^a) \) to be all ancestors of vertices conditioned upon in \( a \) minus \( \text{cond}^a \). Fix a topological order on the vertices in \( Z \cup \text{cond}^a \), \( \langle X_1, \ldots, X_m \rangle \). We can write

\[
\widetilde{P}(\text{cond}^a \mid u^j) = \sum_{z \in \{0,1\}^Z} \prod_{i=1}^m \widetilde{P}(z(X_i) \mid z(X_1, \ldots, X_{i-1}), \text{cond}^a)
\]

\[
= \sum_{z \in \{0,1\}^Z} \prod_{X_i \in Z \cup \text{cond}^a} \widetilde{P}(z(X_i) \mid z(Pa(X_i), \text{pa}^a(X_i))
\]

\[
= \sum_{z \in \{0,1\}^Z} \left( \prod_{X_i \in Z} \widetilde{P}(z(X_i) \mid z(Pa(X_i), \text{pa}^a(X_i)) \right) \left( \prod_{X_i \in \text{cond}^a} \widetilde{P}(z(X_i) \mid z(Pa(X_i), \text{pa}^a(X_i)) \right).
\]

Now let \( C_z \) denote the value in the first grouped product for a given \( z \in \{0,1\}^Z \). Since \( \sum_{z \in \{0,1\}^Z} C_z = 1 \), we can bound the error in the result by the error in the second grouped product:

\[
(1 - \varepsilon)^{|\text{cond}^a|}P(\text{cond}^a \mid u^j) \leq \widetilde{P}(\text{cond}^a \mid u^j) \leq (1 + \varepsilon)^{|\text{cond}^a|}P(\text{cond}^a \mid u^j).
\]

Finally using the bound \( |\text{cond}^a| < 2\Delta^2 \) and the inequalities from Observation 39 we obtain

\[
(1 - 2\Delta^2\varepsilon)P(\text{cond}^a \mid u^j) \leq \widetilde{P}(\text{cond}^a \mid u^j) \leq (1 + 2\Delta^2\varepsilon)P(\text{cond}^a \mid u^j).
\]

By assumption \( \widetilde{P}(u^j \mid \text{cond}^a) \in (1 \pm \varepsilon)P(u^j \mid \text{cond}^a \); \( \widetilde{P}(\text{cond}^a) \) is also known up to multiplicative accuracy \( \varepsilon \) since the sampling needed to estimate the distributions that are input to \( k\text{-MixProd} \).
algorithms far exceed that needed to get an ε estimate of this quantity. Thus, the resulting bound on \( \tilde{P}(u^j) \) is

\[
(1 - 2\Delta^2 \varepsilon)(1 - 3\varepsilon)P(u^j) \leq \tilde{P}(\text{cond}^a | u^j) \leq (1 + 2\Delta^2 \varepsilon)(1 + 3\varepsilon)P(\text{cond}^a | u^j)
\]

which can be simplified to

\[
(1 - 2\Delta^2 \varepsilon - 3\varepsilon)P(u^j) \leq \tilde{P}(\text{cond}^a | u^j) \leq (1 + 2\Delta^2 \varepsilon + 3\varepsilon)P(\text{cond}^a | u^j)
\]

using \((1 - x)(1 - y) \geq (1 - x - y)\) for \(x, y \in [0, 1]\) and \(2\Delta^2 \varepsilon + 3\varepsilon \leq 5\Delta^2 \varepsilon\) for \(\Delta \geq 1\). \(\square\)
Appendices

A Finding good collections of runs

In this section we prove Lemma 34 and give a few different sufficient conditions for the existence of a good collection of runs.

A.1 Constructing a good collection of runs from 2\(k\) centers

One sufficient condition for the existence of a good collection of runs is the presence of 3\(k-3\) variables with disjoint Markov boundaries. We will now present a good collection of runs for this case.

Lemma 42. Given a set \(\mathcal{X} = \{X_1, \ldots, X_{3k-3}\} \subseteq \mathcal{V}\) of 3\(k-3\) variables with depth at most 9\(k\) and disjoint Markov boundaries, Algorithm 4 finds a good collection of runs \(\mathcal{A}\), satisfying \(|\mathcal{A}| = O(2^\gamma n)\) where \(\gamma = \max_{V \in \mathcal{V}} |\text{Mb}(V)|\), assuming that the mixture \(\mathcal{P}\) is \(\zeta\)-informative.

We’ll build our collection of good runs, \(\mathcal{A} = \mathcal{A}_C \cup \mathcal{A}_Y\), incrementally as follows: We will start with a set of 3\(k-3\) vertices \(\mathcal{X} = \{X_1, X_2, \ldots, X_{3k-3}\}\) with disjoint Markov boundaries. An example of four such vertices is given in Figure 2.

We will give a set of central runs \(\mathcal{A}_C\). First, we fix a run \(a_0\) with \(I_{a_0} = \mathcal{X}\) and \(\text{mb}^{a_0}(\mathcal{X})\) being chosen arbitrarily where \(\text{Mb}(\mathcal{X}) := \cup_{X_i \in \mathcal{X}} \text{Mb}(X_i)\). We will refer to this assignment \(\text{mb}^{a_0}(\mathcal{X})\) as the default assignment. Each run in \(a \in \mathcal{A}_C\) will have independent set \(I_a = \mathcal{X}\) and will agree with \(a_0\) on the assignment to \(\text{Mb}(\mathcal{X}) \setminus \text{Mb}(X_i)\) for some \(X_i \in \mathcal{X}\), i.e., \(f^a(\text{Mb}(\mathcal{X}) \setminus \text{Mb}(X_i)) = \text{mb}^{a_0}(\mathcal{X}) \setminus \text{mb}^{a_0}(X_i)\), as well as some arbitrary assignment \(\text{mb}(X_i)\) to \(\text{Mb}(X_i)\). We’ll write each such run as \(a_0[\text{Mb}(X_i) \mapsto \text{mb}(X_i)]\), and \(\mathcal{A}_C\) will be the collection of all such runs as we range over the \(X_i\) and assignments to \(\text{Mb}(X_i)\).

Definition 43. \(\mathcal{A}_C := \{a_0\} \cup \left\{a_0[\text{Mb}(X_i) \mapsto \text{mb}(X_i)] : i \in [3k-3], \text{mb}(X_i) \in \{0,1\}^{\text{Mb}(X_i)}\right\}\).

Claim 44. By construction, \(\mathcal{A}_C\) covers \(\mathcal{X}\) and is alignable.

See Figure 4 for an example of a set of central runs and a visualization of how they are alignable.

![Figure 4: An alignment graph of the default assignment \(a_0\) (\(\text{Cond}^{a_0}\) arbitrarily assigns all Markov boundaries to 0) and representations of the six other central runs. The three runs represented on the left cover all possible assignments to \(\text{Mb}(X_2)\) \((v_1, v_2) = (0,0), (0,1), (1,0)\) while maintaining the default assignment to \(\text{Mb}(X_1)\) to allow alignment with \(a_0\). The right runs similarly cover all possible assignments to \(\text{Mb}(X_1)\), aligned at \(X_2\). The remaining runs in \(\mathcal{A}\) (namely, \(\mathcal{A}_Y\)) will be chosen to cover \(\mathcal{V} \setminus \mathcal{X}\) subject to all of \(\mathcal{A}\) being alignable. The runs in \(\mathcal{A}_Y\) will be the union of two sets as follows:](image-url)
1. For each $Y \in \mathcal{V} \setminus \mathcal{X}$ in a Markov boundary of a center $\text{Mb}(X_i)$, we exclude $X_i$ to form $\mathcal{I}^a = \mathcal{X} - X_i + Y$ and $\text{Cond}^a = \text{Mb}(\mathcal{I}^a)$.

2. For each $Y \notin \text{Mb}(\mathcal{X}) \cap \mathcal{X}$, $\mathcal{I}^a = \mathcal{X} + Y$, and $\text{Cond}^a = \text{Mb}(\mathcal{I}^a)$.

For either independence set we will form $2^{\left|\text{Pa}(Y)\right|}$ runs each associated with a single assignment to $\text{pa}^a(Y)$, with the remaining variables in $\text{Cond}^a \cap \text{Cond}^a$ conditioned on their defaults given by $f^a$. Any leftover variables in $\text{Cond}^a$ can be chosen arbitrarily.

**Algorithm 4 Building a good collection of runs**

**Input:** Vertices $\mathcal{X} = \{X_1, \ldots, X_{3k-3}\} \subseteq \mathcal{V}$ having disjoint Markov boundaries.

**Output** A good collection of runs $\mathcal{A}$.

1. Let $a_0$ be a run with $\mathcal{I}_{a_0} = \{X_1, \ldots, X_{3k-3}\}$ and $\text{Cond}^{a_0}$ chosen arbitrarily.
2. Set $\mathcal{A} \leftarrow \{a_0\}$.
3. for $i = 1, \ldots, 3k-3$ do
4. \hspace{0.5cm} $\mathcal{A} \leftarrow \mathcal{A} \cup \left\{a_0[\text{Mb}(X_i) \mapsto \text{mb}(X_i)]: \text{mb}(X_i) \in \{0,1\}^{\text{Mb}(X_i)}\right\}$.
5. for $Y \in \mathcal{V} \setminus \mathcal{X}$ do
6. \hspace{1cm} if $Y \in \text{Mb}(X)$ for some $X \in \mathcal{X}$ then
7. \hspace{1.5cm} $\mathcal{I}^a = \mathcal{X} - X_i + Y$
8. \hspace{1cm} else
9. \hspace{1.5cm} $\mathcal{I}^a = \mathcal{X} + Y$
10. end if
11. for $\text{pa}(Y) \in \{0,1\}^{\left|\text{Pa}(Y)\right|}$ do
12. \hspace{1cm} if $Y \in \text{An}(\mathcal{I}^a - Y)$ then
13. \hspace{1.5cm} $\text{Cond}^a = \text{Mb}(\mathcal{I}^a)$
14. \hspace{1cm} else
15. \hspace{1.5cm} $\text{Cond}^a = \text{Mb}(\mathcal{I}^a - Y) \cup \text{Pa}(Y)$
16. \hspace{1cm} end if
17. \hspace{1cm} $f^a(\text{pa}(Y)) = \text{pa}(Y)$
18. \hspace{1cm} $\text{Defaults} = \text{Cond}^{a_0} \cap \text{Cond}^a \setminus \text{Pa}(Y)$
19. \hspace{1cm} $f^a(\text{Defaults}) = f^{a_0}(\text{Defaults})$
20. \hspace{1cm} $f^a(\text{Cond}^a \setminus \text{Cond}^{a_0} \setminus \text{Pa}(Y))$ are chosen arbitrarily.
21. \hspace{1cm} $\mathcal{A} \leftarrow \mathcal{A} \cup \{a\}$, where $a$ is given by $a = (\mathcal{I}^a, f^a)$.
22. end for
23. end for
24. end for
25. end for

**Claim 45.** $A_Y$ covers $\mathcal{V} \setminus \mathcal{X}$ and each run in $A_Y$ can be aligned with some run in $A_C$.

**Proof.** The fact that $A_Y$ covers $\mathcal{V} \setminus \mathcal{X}$ follows immediately from $A_Y$ containing a run assigning for independent variable $Y \in \mathcal{V} \setminus \mathcal{X}$ each possible assignment $\text{pa}(Y)$. Fix any run $a \in A_Y$ with $\mathcal{I}^a = \mathcal{X} - X_i + Y$ or $\mathcal{I}^a = \mathcal{X} + Y$ depending on whether $Y$ overlaps with $\text{Mb}(\mathcal{X})$. Now if $\text{Mb}(Y) \cap \text{Mb}(X_j) = \emptyset$ for any $j \neq i$, $a$ and $a_0$ are aligned at $X_j$. If instead $\text{Mb}(Y) \cap \text{Mb}(X_j) \neq \emptyset$ for all $j \neq i$, pick any $j$ and consider the central run $a_0[\text{Mb}(X_j) \mapsto \text{mb}^a(X_j)] \in A_C$. Clearly, $a$ and $a_0[\text{Mb}(X_j) \mapsto \text{mb}^a(X_j)]$ are aligned at $X_j$. In either case, we’ve aligned $a$ to a run in $A_C$. \(\square\)

\[4\text{This is a well-formed run since } Y \in \text{Mb}(X_i) \implies X_j \notin \text{Mb}(Y) \text{ for any } j \neq i \text{ by Observation 13.}\]
Claim 46. Every run $a \in \mathcal{A}$ is at least $(3k - 3)$-independent.

Proof of Lemma 42. This follows immediately from Claims 44, 45, and 46. 

Lemma 47. Let $G = (V, E)$ be a DAG with $n = |V|$, and let $\gamma := \gamma(G)$ be as in Definition 12. Then if $n \geq (3k - 3)(\gamma^2 - \gamma + 1)$ we can find a set of $(3k - 3)$ centers for $G$.

Proof. Let $n \geq (3k - 3)(\gamma^2 - \gamma + 1)$. Fix any source vertex $Y \in V$. Then

$$|\text{Mb}(\text{Mb}(Y)) \cup \text{Mb}(Y) \cup \{Y\}| = |\bigcup_{V \in \text{Mb}(Y)} \text{Mb}(V)|$$

$$\leq |\text{Mb}(Y)| \left( \max_{V \in \text{Mb}(Y)} |\text{Mb}(V)| - 1 \right) + 1$$

$$\leq \gamma^2 - \gamma + 1$$

so we can remove a source vertex $Y \in V$ along with $\text{Mb}(Y) \cup \text{Mb}(\text{Mb}(Y))$ from $G$ at least $(3k - 3)$ times before no vertices remain. The $i$th vertex removed will have depth at most $3i$. Now by Observation 13, if any subsequent vertex $Z$ has a Markov boundary that overlaps $\text{Mb}(Y)$, it must be the case that $Z \in \text{Mb}(\text{Mb}(Y))$, which is not possible since $\text{Mb}(\text{Mb}(Y))$ was removed from $G$. Thus, we can find $(3k - 3)$ centers at depth at most $9k$. 

A.2 Degree bounds

We can ensure that $3k - 3$ centers can be found on certain degree-bounded graphs, which in turn bound bound $\gamma$. Let $\Delta_{in}$ upper bound on the in-degree of any vertex in $G$ and let $\Delta_{out}$ upper bound the out-degree. Then

$$\gamma \leq \Delta_{in} + \Delta_{out} + \Delta_{out}(\Delta_{in} - 1) = \Delta_{in} + \Delta_{out}\Delta_{in}.$$  

If we have a bound $\Delta$ on the degree of the undirected skeleton of $G$, we get that

$$\gamma \leq \Delta(\Delta - 1) = \Delta^2 - \Delta.$$  

Corollary 48. If either of the following conditions hold, we can find $3k - 3$ centers for $G$ with depth at most $9k$:

1. $n \geq (3k - 3)(\Delta_{in}^2 + 2\Delta_{out}\Delta_{in} + \Delta_{out}^2 - \Delta_{in} - \Delta_{out} + 1) = (3k - 3) \cdot O(\Delta_{out}^2\Delta_{in}^2).$

2. $n \geq (3k - 3)(\Delta - 2\Delta^3 + \Delta + 1) = (3k - 3) \cdot O(\Delta^4).$

Corollary 49. Let $G = (V, E)$ be a DAG with $n = |V|$, and let $\gamma := \gamma(G)$ be as in Definition 12. Then if $n \geq (3k - 3)(\gamma^2 - \gamma + 1)$ we can find a set of $(3k - 3)$ centers for $G$ with depth at most $9k$.

Proof of Lemma 34. This follows immediately from Corollary 48 and Lemma 42. 

A.3 Mixtures of paths

Special cases do not require the condition of $3k - 3$ disjoint Markov boundaries. One of these special cases is a mixture of paths $v_1 \to v_2 \to v_3 \to \cdots v_n$ over $k$ sources that is positive and $\zeta$-informative. We will give a good set of runs for $n \geq 6k - 6$.

First, we will specify three default runs, which we will call ODD, EVEN, and LINK.

Definition 50 (ODD default run for Markov chains). Run ODD is specified by an independence set of vertices with odd indices $Z^{\text{ODD}} = \{V_1, V_3, V_5, \ldots, V_{6k-7}\}$. The conditioning set is given by the evenly indexed vertices $\text{Cond}^{\text{ODD}} = \{V_2, V_4, V_6, \ldots, V_{6k-8}\}$. $f^{\text{ODD}}$ may be chosen arbitrarily, but for simplicity we will give $f^{\text{ODD}}(V_i) = 0$ for all $V_i \in \text{Cond}^{\text{ODD}}$. 

23
Definition 51 (EVEN default run for Markov chains). Run EVEN is defined the same way for evenly indexed vertices. That is, \( \mathcal{Z}^{\text{EVEN}} = \{V_2, V_4, V_6, \ldots, V_{6k-6}\} \), \( \text{Cond}^{\text{EVEN}} = \{V_1, V_3, V_5, \ldots, V_{6k-7}\} \), and \( f^{\text{EVEN}}(v_i) = 0 \) for all \( V_i \in \text{Cond}^{\text{EVEN}} \).

Definition 52 (LINK default run for Markov chains). Run LINK is part evenly indexed vertices and part oddly indexed vertices. \( \mathcal{Z}^{\text{LINK}} = \mathcal{Z}^{\text{EVEN}} \cup \{V_1\} \setminus \{V_2\} \). Similarly, we condition on the complement \( \text{Cond}^{\text{LINK}} = \{V_2, V_3, V_5, \ldots, V_{6k-7}\} \), \( f^{\text{LINK}}(v_i) = 0 \) for all \( V_i \in \text{Cond}^{\text{LINK}} \).

Claim 53. If \( n \geq 6k - 6 \), then ODD, EVEN, and LINK are well-formed runs.

Claim 54. ODD and LINK are aligned at \( V_1 \). EVEN and LINK are aligned at evenly indexed vertices beyond \( V_2 \).

Now, we enumerate a set of runs that cover all possible entries to parents of vertices.

Definition 55. Let ODD\([v_i]\) denote a run on \( \mathcal{Z}^{\text{ODD}[V_i]} = \mathcal{Z}^{\text{ODD}} \), \( \text{Cond}^{\text{ODD}[V_i]} = \text{Cond}^{\text{ODD}} \) with \( f^{\text{ODD}[V_i]}(V_j) = 1 \) if \( i = j \) and 0 if \( i \neq j \). Similarly define EVEN\([V_i]\) to be a run on \( \mathcal{Z}^{\text{EVEN}[V_i]} = \mathcal{Z}^{\text{EVEN}} \), \( \text{Cond}^{\text{EVEN}[V_i]} = \text{Cond}^{\text{EVEN}} \) with \( f^{\text{EVEN}[V_i]}(v_j) = 1 \) if \( i = j \) and 0 if \( i \neq j \).

Claim 56. Any run ODD\([V_i]\) is aligned with ODD at \( V_j \) for \( j < i - 1 \) or \( j > i + 1 \). Similarly, any run EVEN\([V_i]\) is aligned with EVEN at \( V_j \) for \( j < i - 1 \) or \( j > i + 1 \).

Definition 57. Let TAIL\(^0\)[\(V_i\)] for \( i > 6k - 6 \) give runs which have
\[
\mathcal{Z}^{\text{T A I L}[V_i]} = \{V_1, V_3, V_5, \ldots, V_{6k-7}, V_i\}
\]
\[
\text{Cond}^{\text{T A I L}[V_i]} = \{V_2, V_4, \ldots, V_{6k-8}, V_{i-1}\}
\]
with
\[
f^{\text{T A I L}[V_i]}(V_j) = \begin{cases} 0 & \text{if } i - 1 \neq j \\ b & \text{if } i - 1 = j \end{cases}
\]

Claim 58. Any run LINK\([V_i]\) is aligned with ODD at \( V_j \) for \( j < 6k - 6 \) with even \( j \).

Claim 59. The set
\[
\left\{ \text{ODD}[V_i] : v_i \in \mathcal{Z}^{\text{ODD}} \right\} \cup \left\{ \text{EVEN}[V_j] : V_j \in \mathcal{Z}^{\text{EVEN}} \right\} \cup \left\{ \text{T A I L}^0[V_j], \text{T A I L}^1[V_j] : j > 6k - 6 \right\}
\]
covers \( \mathcal{V} \).

Proof. For all \( V_i \) with \( i > 6k - 6 \) we have LINK\(^0\)[\(V_i\)] and LINK\(^1\)[\(V_i\)] to cover both possible assignments to \( V_{i-1} \).

Consider \( V_i \) with \( i \leq 6k - 6 \). If \( i > 1 \) is odd, then \( f^{\text{EVEN}[V_i]}(V_{i-1}) = 0 \) and \( f^{\text{EVEN}[V_i]}(V_{i-1}) = 1 \), so all possible assignments to the parent of \( V_i \) are covered. Similarly, if \( i \) is even, then \( f^{\text{ODD}[V_i]}(V_{i-1}) = 0 \) and \( f^{\text{ODD}[V_i]}(V_{i-1}) = 1 \), so all possible assignments to the parent of \( V_i \) are covered. Finally, if \( i = 1 \), then \( V_i \) has no parents.

We give the following example to illustrate this construction. Consider \( k = 2 \) and \( n = 8 \). We will express a run \( a \) as sequences of values 0, 1 or \(*\). A \(*\) in the \( i \)th location indicates \( V_i \in \mathcal{Z}^a \) and a
0 or 1 indicates $f^a(V_i) = 0$ or $f^a(V_i) = 1$, respectively. Finally, - indicates that the variable is not conditioned on and not in the independent set (not in Cond and not in $\mathcal{I}$). The default runs are:

```
EVEN = 0*0*0*---
ODD = *0*0**---
LINK = *00*0*---
```

In addition to these runs, we have:

```
ODD[V_1] = 1*0*0*---
ODD[V_3] = 0*1*0*---
ODD[V_5] = 0*0*1*---
EVEN[V_2] = *1*0*---
EVEN[V_4] = *0*1*---
EVEN[V_6] = *0*0*---
TAIL^0[V_7] = *0*0--0*
TAIL^1[V_7] = *0*0--1*
TAIL^0[V_8] = *0*0--0*
TAIL^1[V_8] = *0*0--1*
```

The construction of a good set of runs given does not use disjoint Markov boundaries, yielding greater efficiency than our more general algorithm for degree bounded graphs.

While similar in name and structure, this setting is different from that of hidden Markov models. Hidden Markov models are Bayesian networks consisting of a chain of unobserved variables, each affecting a unique observed variable, with no causal relations among the observed variables. For instance, in [GKV16], all chains in the mixture have the same state space. The sampling process selects a chain and a starting state from the mixture distribution, then generates a short observable path through the chain. In their model, under some conditions, a sufficiently large collection of paths of length 3 suffices to recover the parameters of the mixture, using spectral methods. A different model is considered in [BGK04]. There, the constituents of the mixture have disjoint state spaces and the observation is a long sequence of interleaved paths in the separate chains. The primary challenge is to cluster the observable states into the $k$ constituents. This model can be viewed as a special case of the hidden Markov model problem.

The natural extension of EVEN and ODD vertices is a two-coloring on a tree which denotes sets that are of even or odd distance from the root. One can construct a good set of runs for learning mixtures of trees of size $n \geq 6k - 6$ that is very similar to the one given for mixtures of paths.

B Extending mixtures of binary product distributions to larger alphabets

All algorithms presented in this paper assume that the observed variables are binary, taking values in \{0, 1\}. A simple reduction allows us to also handle larger alphabets of size $d$ while incurring only a linear (or logarithmic) factor in the runtime.
This algorithm will involve projecting the alphabet onto a binary space in order to call the binary $k$-MixProd-algorithm. For an alphabet $\{0,1,2,\ldots,d−1\}$ on a product distribution, consider the projection onto a binary alphabet given by the indicator function,

$$1_{\ell}(x) = \begin{cases} 0 & \text{if } x \neq i \\ 1 & \text{if } x = i \end{cases}$$

Lemma 60. If $V_1,\ldots,V_n$ is a $\zeta$-separated $k$-mixture of product distributions with alphabet size $d$, then for any assignment of $v_1,\ldots,v_n \in 0,\ldots,d−1$, we have that $1_{v_1}(V_1),\ldots,1_{v_n}(V_n)$ is a $k$-mixture of binary product distributions that is described by $P_u(1_{v_1}(V_1)) = 1 = P_u(1_{v_i}(V_i) = v_i)$ and is $\zeta$-separated.

Proof. First, observe that $1_{v_1}(V_1) = 1$ iff $V_1 = 1$. So, trivially $P_u(V_1 = v_i) = P_u(1_{v_1}(V_1) = 1) = P_u(1_{v_i}(V_i) = 1)$.

Now, $V_1,\ldots,V_n$ is $\zeta$-separated, so

$$P_u(V_i = v_i) − P_{u'}(V_i = v_i) \geq \zeta \quad \forall u \neq u' \in [k]$$

Hence, we also have that $P_u(1_{v_1}(V_1)) = 1 = P_u(1_{v_i}(V_i) = 1) \geq \zeta$.

Projecting each variables alphabet yields a $\zeta$-separated binary mixture of products that contains exactly the information we need. We need one run with alphabet projection $1_{v_1}$ for all possible values of $v_i$ that $V_i$ can take. A set of runs that achieves this and can be “aligned” will prove Theorem 63. We will first give our new definition of run more precisely.

Definition 61. We define a large-alphabet $k$-MixProd run (LAMP) to be given by a list of alphabet projection functions for each variable. These alphabet projections for run are denoted $g^a_i : [d] \mapsto \{0,1\}$ for all $i \in [n]$. Run $a$ applies the $k$-MixProd algorithm to the image of these alphabet projections on the visible variables: $g^a_1(V_1),\ldots,g^a_n(V_n)$.

Observation 62. Two LAMP runs $a,b$ are alignable at $V_i$ as given by Definition 26 if $g^a_i = g^b_i$.

Now, we give the main theorem for mixtures of product distributions on large alphabets.

Theorem 63. There is an algorithm for identifying mixtures of $k$ product distributions on $n$ variables in $\{0,\ldots,d−1\}$ with the following guarantees. Let $\varepsilon > 0$ and suppose that at least $3k−3$ observable variables are $\zeta$-separated. The algorithm runs in time $2d(1/\varepsilon)^2(1/\min \pi_i)^{O(\log k)}(1/\zeta)^{O(k \log k)}n^{O(k)}$.

With high probability (over the distribution of the samples), the algorithm computes $\hat{\pi}$ and $\hat{P}_u(V_i)$ such that $\max_i |\pi_i - \hat{\pi}_i| \leq \varepsilon$ and $\max_{ij} |\hat{P}_u(V_i) - P_u(V_i)| \leq \varepsilon$.

Proof. Now, consider the following two sets of LAMP runs:

1. A set of $d$ central runs $\{c_1,\ldots,c_d\}$ by choosing $g^c_i = 1_i$ for $i \in [d]$.

2. An additional set of $d$ runs $a_1,\ldots,a_d$ in which $g^a_j = 1_i$.

Note that all runs $c_1,\ldots,c_d$ are alignable at variables $V_2,\ldots,V_n$ because the runs share the same alphabet projection at all of these variables. We have that run $a_i$ aligns with run $c_j$ at $V_1$. In addition, we now have at that run $a_i$ identifies $P_u(V_j = i)$ for all $j$, meaning we can construct all possible variables.

$^5$Here, we have chosen to iterate over the alphabet mapping choice for variable $x_1$ which is completely arbitrary.

$^6$Again, the choice of $1_0$ is arbitrary.
Recall that for a $\zeta$-separated variable $V_i$, the choice of $g^K_i(V_i) = 1_{v_i}$ yields a $\zeta$-separated mixture of products distribution with $P_u(1_{v_i}(V_i) = 1) = P_u(V_i = v_i)$ (by Lemma 60). Hence, we must also have $3k - 3$ $\zeta$-separated variables in the projected alphabet distribution $1_{v_1}(V_1) \ldots 1_{v_n}(V_n)$.

The rest of the proof follows from Theorem 10, which allows us to recover $P_u(1_{v_i}(V_i) = 1)$.

Algorithm 1 is run on the runs $\{c_1, \ldots, c_d\}$ and $\{a_1, \ldots, a_d\}$ to align their output and obtain the parameters.

It is worth noting that a similar reduction to $2 \log(d)$ runs can be given using alphabet projections to the $i$th bit in the binary representation of the alphabet. In order for these runs to be successful, we would require more complicated separation conditions.

B.1 Larger alphabets for mixtures of DAGs

The simplest reduction for larger alphabets is to replace each vertex with a clique of $d$ binary vertices which represent the value of the nonbinary vertex. The pseudocode for this process is given in Algorithm 5.

Algorithm 5 Reduction for larger alphabets

| Input: A DAG $(V, E)$ on $n$ variables $V_1, \ldots, V_n \in [d]$. |
| Output: A DAG $(W, E_W)$ on $dn$ binary variables $W_1^1, \ldots W_1^d, \ldots W_n^1, \ldots W_n^d \in \{0, 1\}$. |

1: Start with $E_W \leftarrow \emptyset$
2: for each vertex $i \in [n]$ do
3: Form a clique among $W_i^1, \ldots W_i^d$ by adding directed edges $(W_i^a, W_i^b)$ to $E_W$ for all pairs $a < b$ with $a, b \in [d]$.
4: end for
5: for each directed edge $(V_i, V_j) \in E$ do
6: Add $\{W_i^1, \ldots, W_i^d\} \times \{W_j^1, \ldots, W_j^d\}$ to $E_W$.
7: end for

Observation 64. The maximum degree of $(W, E_W)$ outputted by Algorithm 5 is now at least $\Delta \geq d$.

Observation 65. The number of vertices $|W| = nd$.

We now give the function that translates data from the original graph to the new graph.

Definition 66 (One-hot encoding). We define $\chi(v) = (1_0(v), \ldots, 1_{d-1}(v))$ to give the one-hot encoding of the value of a variable $V$.

This allows us to give the full algorithm.
Algorithm 6 DAG reduction for larger alphabets

Input: A DAG \((\mathcal{V}, \mathcal{E})\) on \(n\) variables \(V_1, \ldots, V_n \in [d]\). And data with entries of the form \((v_1, v_2, \ldots, v_n)\).

Output: Parameters \(P_u(v_i | Pa(V_i))\) for \(i \in [n]\).

1: Use Algorithm 5 to create a larger DAG on binary variables, called \(G'\).
2: One-hot encode data on each \(V_i\) into binary variables \(\xi(V) = (W_{1i}, \ldots, W_{di})\).
3: Run the Mixture of DAGs algorithm for \(G'\) on the one-hot encoded data.
4: for each parameter \(P_u(v_i | Pa(V_i))\) do
5: Let \(Z\) indicate zero assignments to \(W_{bi}\) for \(b \neq v_i\).
6: One-hot encode each parent \(V_j \in Pa(V_i)\) to obtain assignments to \(Pa(W_{vi}) \setminus \{w^b : b \neq v_i\}\), denoted \(pa^{OH}(W_{vi})\).
7: Assign \(P_u(v_i | Pa(V_i)) = P_u(W_{vi} = 1 | Z, pa^{OH}(W_{vi}))\).
8: end for

Theorem 67. If \(P(V) = \sum_u P_u(V)P(u)\) is a \(\zeta\)-informative mixture of \(k\) distributions over a DAG \(G = (\mathcal{V}, \mathcal{E})\) with \(\mathcal{V} \in \{0, \ldots, d-1\}\) and there exists a set of \(3k-3\) centers, then there is an algorithm to compute estimates of all parameters to accuracy \(\varepsilon\). If \(D = \max(d, \Delta)\) then the sample complexity (and runtime) are bounded by

\[
nd \log nd \cdot \varepsilon^{-2}(\Delta + 1)^O(k) 2^{O(D^2+Dk)}(\min \pi_i)^{-O(\log k)}(1/\zeta)^{O(k \log k + D^2 k)}.
\]

Proof. The algorithm uses the reduction in Algorithm 6. This involves running the algorithm on a larger graph with \(nd\) vertices and \(\Delta \geq d\), which modifies the run-time and sample complexity as given.

C Mixtures of Product Distributions

In this section we describe the algorithm for mixtures of product distributions and analyze its performance. We provide pseudocode formalizing the algorithm, see Fig. 7. For a matrix \(Q\), we use \(Q^+\) to denote its Moore-Penrose inverse (a.k.a. pseudo-inverse).
Algorithm 7 Identifies a mixture of product distributions given $3k-3$ $\zeta$-separated observable bits

1. $A \leftarrow 2^S$, $B \leftarrow 2^T$, $B' \leftarrow 2^{T'}$ (w.l.o.g. $1 \in T$).
2. If $\min \{ \sigma_k(\tilde{C}_{B,A}), \sigma_k(\tilde{C}_{B',A}) \} < \pi \min \zeta O(k)$, terminate.
3. $\tilde{v}_0 \leftarrow (\tilde{g}(R))_{R \in A}$.
4. $\tilde{v}_1 \leftarrow (\tilde{g}(R \cup \{1\}))_{R \in A}$.
5. $\tilde{u}_1 \leftarrow \tilde{v}_1(\tilde{C}_{B,A})^{-1}$.
6. $\tilde{u}_1' \leftarrow \tilde{v}_1(\tilde{C}_{B',A})^{-1}$.
7. for $i = 1, \ldots, 1 + \lg k$ do
8. for $j = 1, \ldots, 2^{i-1}$ do
9. $\tilde{v}_{2i-1+j} \leftarrow (\tilde{u}_j \circ \tilde{u}_{2i-1}) \tilde{C}_{B+B',A}$.
10. $\tilde{u}_{2i-1+j} \leftarrow \tilde{v}_{2i-1+j}(\tilde{C}_{B,A})^{-1}$.
11. end for
12. $\tilde{u}_{2i} \leftarrow \tilde{v}_{2i}(\tilde{C}_{B',A})^{-1}$.
13. end for
14. Let $\mathcal{H}_{k+1}$ be the $(k+1) \times (k+1)$ Hankel matrix with entries given by $[H_{k+1}]_{i,j=0} = (\tilde{v}_{i+j})_{\{1\}}$.
15. If the second-smallest eigenvalue of $\mathcal{H}_{k+1}$ is below $\pi \min \zeta O(k)$, terminate.
16. $\tilde{m}_1, \tilde{\pi} \leftarrow \text{LEARNPOWERDISTRIBUTION}(\mathcal{H}_{k+1})$.
17. $\tilde{V} \leftarrow (\tilde{v}_0, \ldots, \tilde{v}_{k-1})$.
18. $\tilde{V} \leftarrow (\tilde{m}_1 \circ \tilde{V}, \ldots, \tilde{m}_1 \circ \tilde{V})$.
19. $\tilde{A} \leftarrow \tilde{V}^{-1}(\tilde{V} \circ \tilde{m}_1)^{-1} \tilde{V}$.
20. $\tilde{B} \leftarrow \tilde{C}_{B,A}(\tilde{A}) + \tilde{A}$.
21. For every $i \in [n] \setminus S$, $\tilde{m}_i \leftarrow ((\tilde{g}(R \cup \{i\}))_{R \in A})^T(\tilde{A})^+ \tilde{\pi}^{-1}$.
22. For every $i \in S$, $\tilde{m}_i \leftarrow ((\tilde{g}(R \cup \{i\}))_{R \in A})^T(\tilde{B})^+ \tilde{\pi}^{-1}$.

C.1 The Algorithm

Constructing higher moments of a row. In what follows we assume that we have three mutually disjoint sets $S,T,T'$, each containing $k-1$ $\zeta$-separated variables. We show how to compute moments of arbitrary degree of an observable in $T$, w.l.o.g. $V_i$. Equipped with good approximations of the moments $E[V_1], E[V_1^2], \ldots, E[V_1^{2k}]$ of $V_i$, we can use known results to recover $\pi$ and $m_1$.

The only thing that our statistics tell us about row $m_1$ in isolation is its first moment: $E[V_1] = m_1 \pi \circ 1^T$. The second moment is equal to $(m_1 \circ m_1) \pi \circ 1^T$, and more generally, the $r$-th moment is $m_1^{\circ r} \pi \circ 1^T$ for any $r$.

Put $A = 2^S$, $B = 2^T$, and $B' = 2^{T'}$. Also, put $A = M[A]$, $B = M[B]$, and $B' = M[B']$. Recall from Section 2 the definition

$$C_{B,A} := B \pi \circ A^T \in \mathbb{R}^{k-1 \times 2k-1}.$$ 

This is an observable matrix (meaning every entry of it is a function of the joint statistics of the observable random variables $V_1, \ldots, V_n$). Let

$$\tilde{C}_{B,A} := \tilde{g}(B \cup A)_{B \in B, A \in A}$$

be the corresponding matrix of empirical moments. Similarly, define as in Section 2 the matrices $C_{B',A}$ and $C_{B+B',A}$, and let the empirical versions be $\tilde{C}_{B',A}$ and $\tilde{C}_{B+B',A}$, respectively.
Now consider the vector
\[ v_1 := m_1 \pi \otimes A^T = (E[V_1 V_R])_{R \in A}. \]
Each coordinate \( E[V_1 V_S] = E[V_{S,U(1)}] \) is a multi-linear moment and is therefore observable. In the algorithm, we will have to make do with approximations to these multi-linear moments, and we will address that challenge shortly; for now we write down the algorithm as it would run with perfect statistics.

While we’d like to find \( m_1^{\otimes 2} \) directly, we won’t be able to do so, and instead we’ll settle for the following indirect calculation. Put \( u_1 = v_1 C_{BA}^{-1} \) and \( u'_1 = v_1 C_{BA'}^{-1} \). We will show that both matrices \( C_{BA}^{-1} \) and \( C_{BA'}^{-1} \) are invertible, hence \( u_1 \), \( u'_1 \) are well-defined. Now set
\[ v_2 := (u_1 \otimes u'_1) C_{BA}, \]
More generally, suppose that we’ve already computed \( v_1, v_2, \ldots, v_{\ell-1} \) and we want to compute \( v_\ell \).
Then write \( \ell = j + 2^i \) for \( j \in \{1,2,\ldots,2^i\} \) (so \( 2^i \) is the largest power of 2 strictly smaller than \( \ell \)). Put \( u_j = v_j C_{BA}^{-1} \) and \( u_{2i} = v_{2i} C_{BA'}^{-1} \). Then set
\[ v_\ell := (u_j \otimes u'_{2i}) C_{BA'}, \quad (6) \]
Intuitively, what we are doing here is writing \( m_1^{\otimes j} \) as a linear combination of the rows of \( B \), with coefficients \( u_j \), and \( m_1^{\otimes 2^i} \) as a linear combination of the rows of \( A \), with coefficients \( u_{2i} \), and as these two combinations of multilinear moments do not use a common variable, the two “synthetic” copies of Hadamard powers of \( m_1 \) are independent and we can multiply them to get a copy of \( m_1^{\otimes j + \otimes 2^i} \) whose expectation is observable. This is summarized more formally in the following lemma.

**Lemma 68.** For all \( \ell \), \( (v_\ell)_{\{1\}} = E[V_1^\ell] \).

**Proof.** We show by induction on \( \ell \) that \( v_\ell = m_1 \otimes \pi \otimes A^T \), hence \( (v_\ell)_{\{1\}} = m_1 \otimes \pi \otimes 1^T = E[V_1^\ell] \). The base case of \( \ell = 1 \) follows trivially from the definition of \( v_1 \). So suppose that this is true of \( v_\ell \) for \( \ell' < \ell \). Let \( \ell = 2^i + j \) for \( j \in \{1,2,\ldots,2^i\} \). Now, \( A, B \), and \( B' \) have full column rank, so
\[
\begin{align*}
v_\ell &= (v_j C_{BA}^{-1} \otimes v_{2i} C_{BA'}^{-1}) C_{BA}, \\
&= (m_1^{\otimes j} \pi \otimes A^T (B \pi \otimes A^T)^{-1} \otimes m_1^{\otimes 2^i} \pi \otimes A^T (B' \pi \otimes A^T)^{-1})(B \otimes B') \pi \otimes A^T \\
&= (m_1^{\otimes j} (B)^+ \otimes m_1^{\otimes 2^i} (B')^+)(B \otimes B') \pi \otimes A^T \\
&= m_1^{\otimes \pi \otimes A^T},
\end{align*}
\]
where \( B \otimes B' \) is the matrix with rows \( b \otimes b' \) for every row \( b \) of \( B \) and every row \( b' \) of \( B' \).

In the algorithm, we’ll only have access to the approximations \( \tilde{C}_{BA}, \tilde{C}_{BA'} \), and \( \tilde{C}_{BA+BA'}, \) and, starting with \( \tilde{v}_1 \), we will compute approximate vectors \( \tilde{u}_j = \tilde{v}_j \tilde{C}_{BA}^{-1} \) and \( \tilde{u}_{2i} = \tilde{v}_{2i} \tilde{C}_{BA'}^{-1} \) and use those to compute an approximate vector \( \tilde{v}_\ell = (\tilde{u}_j \otimes \tilde{u}_{2i}) \tilde{C}_{BA}, \tilde{B}_{BA} \). Notice the advantage of the recurrence in Equation (6); we are able to get away with performing at most \( 1 + \log k \) iterations to compute any of \( \tilde{v}_1, \ldots, \tilde{v}_{2k} \). Each iteration at most two matrix multiplications by \( \tilde{C}_{BA}^{-1} \) and \( \tilde{C}_{BA'}^{-1} \), followed by a convolution, followed by multiplication by \( \tilde{C}_{BA+BA'} \). Each such step can increase the initial sampling error by at most a factor of \( \zeta^{-O(k)} \pi_{\min}^{-O(1)} \). By starting with empirical moments accurate to within \( \epsilon \zeta^{O(k \log k)} \pi_{\min}^{O(\log k)} \), we can ensure that the resulting vectors \( \tilde{v}_i \) are sufficiently close to the vectors \( v_i \) to start solving for \( m_1 \) and \( \pi \). We note that a simpler version of the above procedure uses only \( 2k - 1 \) \( \zeta \)-separated variables, but needs \( 2k - 1 \) iterations to compute the \( \tilde{u}_i \)-s, hence the required initial accuracy would be exponential in \( k^2 \) rather than in \( k \log k \).
**Flagging a failure condition.** If the chosen rows $S \cup T \cup T'$ fail to all be $\zeta$-separated, the algorithm might fail. However, we will detect such failure. The conditions that we actually need so that the algorithm should work, are these: (a) $\tilde{C}_{\mathcal{B}A}$ and $\tilde{C}_{\mathcal{B}'}A$ should have a large least singular value. (It does not actually matter whether all rows we use are $\zeta$-separated, that was merely a sufficient condition for this well-conditioning.) We compute this singular value explicitly and simply dismiss the triple $S,T,T'$ if this condition fails. (b) The first row of $\mathcal{B}$, namely $\{1\}$ in the numbering used in the pseudocode for Algorithm 7, should be $\zeta$-separated. If condition (a) holds but this condition fails, we will detect the failure in line 15 of the algorithm, because the Hankel matrix will have insufficient eigenvalue gap (see Cor. 12 of [GMRS20]).

**Solving the power distribution problem.** Once we’ve computed $\tilde{v}_1, \ldots, \tilde{v}_{2k}$, we have by Lemma 68 access to all of the moments of the distribution corresponding to observable $V_1$. Recall, these are the moments of a mixture of $k$ Bernoulli random variables, where the $r$-th moment corresponds to drawing a mixture component $j$ with probability $\pi_j$, then setting the Bernoulli random variable to 1 with probability $m_{ij}^r$. The problem of recovering the parameters (i.e., the vectors $\mathbf{m}_1$ and $\pi$) from approximate moments of this form has been extensively studied, and many algorithms have been provided. We use the algorithm LEARNPOWERDISTRIBUTION from [GMRS20], which on inputs accurate to within $\varepsilon$, outputs parameters $\hat{\mathbf{m}}_1$ and $\hat{\pi}$ to within accuracy $\frac{1}{\pi_{\min}}(1 - 0(k))\varepsilon$ while running in time (arithmetic operations) $k^2\alpha(1)$.

**Recovering the remaining parameters.** Once we have estimates for $\mathbf{m}_1$ and $\pi$, we can simply solve for $\mathbf{A}$ and $\mathbf{B}$ (also $\mathbf{B}'$, but it’s not needed here). Write $v_0 = (\tilde{g}(R))_{R \in \mathcal{A}}$ and put

$$V = \begin{pmatrix} v_0 \\ \vdots \\ v_{k-1} \end{pmatrix},$$

the matrix with rows $v_i$, for $i = 0, 1, \ldots, k - 1$. Let $Vdm(\mathbf{m}_1)$ be the Vandermonde matrix with rows $\mathbf{m}_1^{\leq i}$ for $i = 0, \ldots, k - 1$. Then,

$$\mathbf{A}^T = \pi_{\circ}^{-1}(Vdm(\mathbf{m}_1))^{-1}V.$$

We can thus solve for $\mathbf{A}^T$. To solve for $\mathbf{B}$ we use

$$\mathbf{B} = \tilde{C}_{\mathcal{B}A}(\mathbf{A}^T)^+\pi_{\circ}^{-1}.$$

As $\mathbf{A}^T$ has full row rank, its Moore-Penrose inverse is given by $(\mathbf{A}^T)^+ = \mathbf{A}(\mathbf{A}^T)^{-1}$ and $\mathbf{A}^T(\mathbf{A}^T)^+ = I$. (Likewise for $\mathbf{B}'$ we can use $\mathbf{B}' = \tilde{C}_{\mathcal{B}'A}(\mathbf{A}^T)^+\pi_{\circ}^{-1}$. This is not really needed.)

Now for any row $i$ not already computed, we can solve for $\mathbf{m}_i$ as follows. If $i \in [n] \setminus S$, then

$$\mathbf{m}_i = ((E[V_iV_R])_{R \in \mathcal{A}})(\mathbf{A}^T)^+\pi_{\circ}^{-1}.$$

If $i \in S$, then use

$$\mathbf{m}_i = ((E[V_iV_R])_{R \in \mathcal{B}})(\mathbf{B}^T)^+\pi_{\circ}^{-1}.$$

**Runtime.** The algorithm has five main steps.

1. Collect a sufficiently large sample. We need to collect $(\log n/\varepsilon^2)(1/\min \pi_i)^{O(\log k)}(1/\zeta)^{O(k \log k)}$ independent samples, each containing $n$ bits, to achieve the required accuracy of the multilinear moments.
2. Find disjoint $S, T, T' \subset [n]$. This requires $n^{O(k)}$ iterations to check all possible disjoint $S, T, T'$. In each iteration, we need to aggregate the multilinear moments that we'll be using from the collected samples. If we know that all the bits are $\zeta$-separated, we can skip the enumeration and aggregate the statistics just once.

3. Compute the higher order moments of $V_1$. This step takes time $2^{O(k)}$ (on account of the dimensions of the vectors and matrices).

4. Apply the power distribution result to recover $m_1$ and $\pi$. This can be done in time $k^{2+o(1)} + O(k(\log^2 k) \log \log(\varepsilon^{-1}))$ (see Corollary 76).

5. Solve for the remaining parameters.

**C.2 Analyzing the Algorithm**

Define $\zeta_1 = \zeta / 9k^{3/2}$. This will simplify the statement of some of the bounds.

**Bounding $\|\tilde{u}_j - u_j\|$ for $j \leq 2k$.** The following lemma is a consequence of Theorem 6 in Section 2.

**Lemma 69.** The matrices $A$, $B$, and $B'$ satisfy

1. The first row of $A$, $B$, and $B'$ is the all-ones vector $\mathbb{1}$.
2. $\sigma_k(A), \sigma_k(B), \sigma_k(B') \geq \zeta_1^k$.
3. $\sigma_{\max}(A), \sigma_{\max}(B), \sigma_{\max}(B') \leq k2^{k-1}$.

Moreover, the matrices $C_{BA}$ and $C_{B'A}$ satisfy

1. $\sigma_{\max}(C_{BA}), \sigma_{\max}(C_{B'A}) \leq 2^{2k-2}$.
2. $\sigma_k(C_{BA}), \sigma_k(C_{B'A}) \geq \pi_{\min} \zeta_1^{2k}$.

**Proof.** This follows immediately from Theorem 6, the definitions of $C_{BA}$ and $C_{B'A}$, and the min-max characterization of the first and last singular values. \hfill $\square$

**Corollary 70.** $\|(C_{BA})^{-1}\|, \|(C_{B'A})^{-1}\| \leq \pi_{\min}^{-1} \zeta_1^{-2k}$.

**Lemma 71.** $\|u_i\|, \|u'_i\| \leq \pi_{\min}^{-1} \zeta_1^{-2k}$, and $\|v_i\| \leq \zeta_1^{-k}$.

**Proof.** Clearly, $\|v_i\| \leq 2^{k-1} \leq \zeta_1^{-k}$, as $v_i$ is a vector of $2^{k-1}$ moments of products of Bernoulli random variables. Now $\|u_i\| = \|v_i C_{BA}^{-1}\| \leq \zeta_1^{-k} \|C_{BA}^{-1}\| \leq \pi_{\min}^{-1} \zeta_1^{-2k}$. A similar argument bounds $\|u'_i\|$. \hfill $\square$

**Lemma 72.** If all multilinear moments are within $\varepsilon$ of their true values, then

$$\|\hat{C}_{BA} - C_{BA}\|_2, \|\hat{C}_{B'A} - C_{B'A}\|_2, \|\hat{C}_{B + B', A} - C_{B + B', A}\|_2 \leq 2^k \varepsilon < \zeta_1^{-k} \varepsilon,$$

and

$$\|\hat{C}_{BA}^{-1} - C_{BA}^{-1}\|_2, \|\hat{C}_{B'A}^{-1} - C_{B'A}^{-1}\|_2, \|\hat{C}_{B + B', A}^{-1} - C_{B + B', A}^{-1}\|_2 \leq 2 \pi_{\min}^{-2} \zeta_1^{-5k} \varepsilon.$$
Proof. The first two inequalities just use \( \| \cdot \|_2 \leq \| \cdot \|_F \) and the fact that every entry in the matrix of differences is at most \( \varepsilon \) in magnitude. For the final inequality we use Lemma 84 to get
\[
\left\| \hat{C}_{BA}^{-1} - C_{BA}^{-1} \right\| \leq 2 \| C_{BA}^{-1} \|^2 \| \hat{C}_{BA} - C_{BA} \| \leq 2 \pi_{min}^{-2} \varepsilon^{-5k} \varepsilon.
\]
The same argument bounds \( \left\| \hat{C}_{BA}^{-1} - C_{BA}^{-1} \right\|_2 \).

Lemma 73. If all multilinear moments are within \( \varepsilon \) of their true values, then we have for any \( i \in \lg(2k) \) and \( j \leq 2^i \),
\[
\| \tilde{v}_j - v_j \|, \| \tilde{u}_j - u_j \|, \| \tilde{u}_j' - u_j' \| \leq \pi^{-2i} \varepsilon^{-11i} \varepsilon.
\]

Proof. Recall that we initialize the algorithm with
\[
\tilde{v}_1 \leftarrow (\hat{g}(R \cup \{1\}))_{R \in A}, \quad \tilde{u}_1 \leftarrow \tilde{v}_1(\hat{C}_{BA})^{-1}, \quad \tilde{u}_1' \leftarrow \tilde{v}_1(\hat{C}_{BA})^{-1}.
\]
First, we observe that \( \| \tilde{v}_1 - v_1 \| \leq \varepsilon \) by assumption. Since \( \tilde{u}_1, \tilde{u}_1' \) are computed in the same manner here as in the loop, we’ll bound that error in the induction. Now assume that the claim holds up to \( i - 1 \), and let \( \varepsilon_{i-1} \) be the bound obtained in the \( i - 1 \)th step. Recall that in each iteration of the outer loop we compute
\[
\tilde{v}_{2i} \leftarrow (\tilde{u}_{2i-1} \otimes \tilde{u}_{2i-1}) \hat{C}_{BA + BA}, \quad \tilde{u}_{2i} \leftarrow \tilde{v}_{2i}(\hat{C}_{BA})^{-1}, \quad \tilde{u}_{2i}' \leftarrow \tilde{v}_{2i}(\hat{C}_{BA})^{-1}.
\]
We also observe that we choose \( \varepsilon \) so that \( \| \tilde{u}_j - u_j \| \ll \| u_j \| \) and \( \| \tilde{v}_j - v_j \| \ll \| v_j \| \) for all \( i, j \) so that we can upper bound \( \| \tilde{u}_j \| \leq 2 \| u_j \| \leq 2 \pi_{min}^{-1} \varepsilon^{-2k} \) and \( \| \tilde{v}_j \| \leq 2 \| v_j \| \leq 2 \varepsilon^{-1} \). We’ll first focus on bounding \( \| \tilde{v}_{2i} - v_{2i} \| \). To do this we write
\[
\tilde{v}_{2i} - v_{2i} = (\tilde{u}_{2i-1} \otimes \tilde{u}_{2i-1}) \hat{C}_{BA + BA} - (u_{2i-1} \otimes u_{2i-1}) C_{BA + BA}.
\]
We now let \( w = u_{2i-1} - u_{2i-1} \), \( u' = \tilde{u}_{2i-1} - u_{2i-1} \), and \( E = \hat{C}_{BA + BA} - C_{BA + BA} \), and use the triangle inequality to obtain
\[
\| \tilde{v}_{2i} - v_{2i} \| \leq \left\| (w \otimes u_{2i-1}) \hat{C}_{BA + BA} \right\| + \left\| (\tilde{u}_{2i-1} \otimes w') \hat{C}_{BA + BA} \right\| + \left\| (\tilde{u}_{2i-1} \otimes \tilde{u}_{2i-1}) E \right\|.
\]
The first two terms can each be bounded by
\[
\| w \| 2 \| u_{2i-1} \| \hat{C}_{BA + BA} \| \leq 2 \pi_{min}^{-1} \varepsilon^{-2k} 2^{3k} \varepsilon^{-1},
\]
and the final term is bounded by
\[
4 \| u_{2i-1} \|^2 \| E \| \leq 4 \pi_{min}^{-2} \varepsilon^{-4k} \varepsilon^{-k} \varepsilon.
\]
As a result we get that
\[
\| \tilde{v}_{2i} - v_{2i} \| \leq 4 \pi_{min}^{-1} \varepsilon^{-5k} (\varepsilon^{-1} + \varepsilon).
\]
Now we can bound \( \| u_{2i} - u_{2i} \| \) by observing that
\[
\tilde{u}_{2i} - u_{2i} = \tilde{v}_{2i} \hat{C}_{BA}^{-1} - u_{2i} C_{BA}^{-1}.
\]
Let \( z = \tilde{v}_{2i} - v_{2i} \) and \( D = \hat{C}_{BA}^{-1} - C_{BA}^{-1} \). The above equation becomes
\[
\tilde{u}_{2i} - u_{2i} = (v_{2i} + z)(\hat{C}_{BA}^{-1} + D) - v_{2i} C_{BA}^{-1} = v_{2i} D + z C_{BA}^{-1} + z D,
\]

and after taking norms and using the triangle inequality we obtain
\[ \| \tilde{u}_{2i} - u_{2i} \| \leq \| v_{2i} \| D + \| z \tilde{C}_{G,A}^{-1} \| + \| zD \|. \]

By Corollary 70, Lemma 71 and the induction hypothesis, we get
\[ \| \tilde{u}_{2i} - u_{2i} \| \leq \pi_{\min}^{-2} \zeta_1^{-i} 2 \zeta_1^{-5} \varepsilon + \| \tilde{v}_{2i} - v_{2i} \| \left( \pi_{\min}^{-1} \zeta_1^{-i} + 2 \pi_{\min}^{-2} \zeta_1^{-5} \varepsilon \right) \]
\[ \leq 2 \pi_{\min}^{-2} \zeta_1^{-10} \varepsilon + 4 \pi_{\min}^{-2} \zeta_1^{-5} (\varepsilon_{i-1} + \varepsilon) \left( \pi_{\min}^{-1} \zeta_1^{-i} + 2 \pi_{\min}^{-2} \zeta_1^{-5} \varepsilon \right) \]
\[ \leq 32 \pi_{\min}^{-2} \zeta_1^{-10} \varepsilon_{i-1} \]
\[ \leq \pi_{\min}^{-2} \zeta_1^{-11} \varepsilon_{i-1}, \]

where we use the fact that \( \varepsilon_{i-1} \geq \varepsilon \). For \( j \) not a power of 2, we can do the same analysis, and since the error bound is increasing in \( j \), the result will follow. \( \square \)

**Corollary 74.** Algorithm 7 will produce vectors \( \tilde{v}_i \) for \( i \leq 2k \) satisfying
\[ \| \tilde{v}_i - v_i \| \leq \pi_{\min}^{-O(\log k)} \zeta^{-O(\log k)} \varepsilon. \]

**Applying the power distribution result.** We use the following theorem.

**Theorem 75** (Theorem 17 from [GMRS20]). Given a mixture \( \mathcal{M} = (m, \pi) \) of \( k \) Bernoulli random variables with probabilities \( m_1, \ldots, m_k \) and mixing probabilities \( \pi_1, \ldots, \pi_k \), respectively, let \( [\mathcal{H}_{k+1}]_{i,j=0}^{k} = \mu_{i+j} \) be the matrix of moments of the distribution. If \( m \) is \( \zeta \)-separated, then there is an algorithm, LearnPowerDistribution, that takes as input a Hankel matrix \( [\tilde{H}_{k+1}]_{i,j=0}^{k} = \tilde{\mu}_{i+j} \) of approximate moments of \( \mathcal{M} \) satisfying \( \| \tilde{H}_{k+1} - \mathcal{H}_{k+1} \|_2 \leq \pi_{\min}^{-2} \gamma \zeta^{16k} \) (for some \( \gamma \geq 1 \)), and outputs a model \( \tilde{\mathcal{M}} = (\tilde{m}, \tilde{\pi}) \) satisfying
\[ \| \tilde{m} - m \|_\infty, \| \tilde{\pi} - \pi \|_\infty \leq 2^{-\gamma} \]
using \( O(k^2 \log k + k \log^2 k \cdot \log(\log \zeta^{-1} + \log \pi_{\min}^{-1} + \gamma)) \) arithmetic operations.

**Corollary 76.** The output \( (\tilde{m}_1, \tilde{\pi}) \) of LearnPowerDistribution in line 14 of Algorithm 7 satisfies
\[ \| \tilde{m}_1 - m_1 \|, \| \tilde{\pi} - \pi \| \leq \pi_{\min}^{-O(\log k)} \zeta^{-O(\log k)} \varepsilon. \]

**Proof.** Every entry \( (\tilde{v}_i)_{1} \) satisfies \( \| (\tilde{v}_i)_1 - (v_i)_1 \| \leq \pi_{\min}^{-O(\log k)} \zeta^{-O(\log k)} \varepsilon \) so
\[ \| \tilde{H}_{k+1} - \mathcal{H}_{k+1} \| \leq \pi_{\min}^{-O(\log k)} \zeta^{-O(\log k)} \varepsilon \]
which implies that
\[ \| \tilde{m}_1 - m_1 \|_\infty, \| \tilde{\pi} - \pi \|_\infty \leq \pi_{\min}^{-O(\log k)} \zeta^{-O(\log k)} \varepsilon. \]
Finally, we need to multiply by \( k \leq \zeta^{-1} \) to account for the conversion to the Euclidean norm to get the stated bound. \( \square \)
Solving for the rest of the model. Once we’ve computed $\tilde{\mathbf{m}}_1$ and $\tilde{\pi}$, we’ll use them to compute the remaining model parameters. In this section we bound the additional error introduced by these computations.

**Proposition 77.** $||V_{dm}(\tilde{\mathbf{m}}_1) - V_{dm}(\mathbf{m}_1)|| \leq k ||\tilde{\mathbf{m}}_1 - \mathbf{m}_1|| \leq \zeta^{-1} ||\tilde{\mathbf{m}}_1 - \mathbf{m}_1||$.

**Claim 78** (Claim 26 in [GMRS20]). $||V_{dm}(\mathbf{m}_1)^{-1}|| \leq 2^k / \zeta^{k-1} \leq \zeta^{-2k}$ when $\mathbf{m}_1$ is $\zeta$-separated.

**Lemma 79.** The computed $\tilde{\mathbf{A}}$ produced by Algorithm 7 will satisfy

$$\left\|\tilde{\mathbf{A}} - \mathbf{A}\right\| \leq \pi_{\min}^{-O(\log k)} \zeta^{-O(k \log k)} \epsilon.$$

**Proof.** Recall $\tilde{V} = (\tilde{v}_0; \ldots ; \tilde{v}_{k-1})$ from Algorithm 7 and $V = (v_0; \ldots ; v_{k-1})$ is its real-value analog. First, we observe that $||\tilde{V}|| \leq \zeta^{-3k}$. Also, by Lemma 84 and Claim 78, $||V_{dm}(\tilde{\mathbf{m}}_1)|| \leq \zeta^{-3k}$. Now, by Corollary 76, $||\tilde{\pi}_\otimes^{-1} - \pi_\otimes^{-1}|| \leq \pi_{\min}^{-O(\log k)} \zeta^{-O(k \log k)} \epsilon$. Thus,

$$||\tilde{\pi}_\otimes^{-1} - \pi_\otimes^{-1}|| \left\|\left(V_{dm}(\tilde{\mathbf{m}}_1)\right)^{-1}\right\| \tilde{V} \leq \pi_{\min}^{-O(\log k)} \zeta^{-O(k \log k)} \epsilon.$$

Now, by Lemma 84, $||\left(V_{dm}(\tilde{\mathbf{m}}_1)\right)^{-1} - \left(V_{dm}(\mathbf{m}_1)\right)^{-1}|| \leq \pi_{\min}^{-O(\log k)} \zeta^{-O(k \log k)} \epsilon$, so,

$$||\tilde{\pi}_\otimes^{-1}|| \left\|\left(V_{dm}(\tilde{\mathbf{m}}_1)\right)^{-1} - \left(V_{dm}(\mathbf{m}_1)\right)^{-1}\right\| \tilde{V} \leq \pi_{\min}^{-O(\log k)} \zeta^{-O(k \log k)} \epsilon.$$

Finally, $||\tilde{V} - V|| \leq \pi_{\min}^{-O(\log k)} \zeta^{-O(k \log k)} \epsilon$, so that

$$||\tilde{\pi}_\otimes^{-1}|| \left\|\left(V_{dm}(\tilde{\mathbf{m}}_1)\right)^{-1}\right\| ||\tilde{V} - V|| \leq \pi_{\min}^{-O(\log k)} \zeta^{-O(k \log k)} \epsilon.$$

Putting these together, we easily obtain

$$\left\|\tilde{\mathbf{A}} - \mathbf{A}\right\| = \left\|\tilde{\pi}_\otimes^{-1}\left(V_{dm}(\tilde{\mathbf{m}}_1)\right)^{-1}\tilde{V} - \pi_\otimes^{-1}\left(V_{dm}(\mathbf{m}_1)\right)^{-1}V\right\|
\leq \left\|\tilde{\pi}_\otimes^{-1} - \pi_\otimes^{-1}\right\| \left\|\left(V_{dm}(\tilde{\mathbf{m}}_1)\right)^{-1}\right\| \tilde{V} + \left\|\pi_\otimes^{-1}\right\| \left\|\left(V_{dm}(\tilde{\mathbf{m}}_1)\right)^{-1} - \left(V_{dm}(\mathbf{m}_1)\right)^{-1}\right\| \tilde{V}
+ \left\|\pi_\otimes^{-1}\right\| \left\|\left(V_{dm}(\tilde{\mathbf{m}}_1)\right)^{-1}\right\| \left\|\tilde{V} - V\right\| (\left\|V_{dm}(\mathbf{m}_1)^{-1}E_2\right\| + \left\|E_1V\right\|_{\infty})
+ \left\|\pi_\otimes^{-1}\right\| \left\|\left(V_{dm}(\tilde{\mathbf{m}}_1)\right)^{-1}\right\| \left\|\tilde{V}\right\|
\leq \pi_{\min}^{-O(\log k)} \zeta^{-O(k \log k)} \epsilon.$$

**Lemma 80.** $\left\|\left(\tilde{\mathbf{A}}^T\right)^+\right\| \leq \zeta^{-O(k)}$.

**Proof.** Notice that Lemma 79 together with the lower bound on $\sigma_k(\mathbf{A})$ from Theorem 6 imply that $\tilde{\mathbf{A}}^T$ has full row rank. Now,

$$\left\|\left(\tilde{\mathbf{A}}^T\right)^+\right\| \leq \left\|\left(\mathbf{A}^T\right)^+\right\| + \left\|\left(\tilde{\mathbf{A}}^T\right)^+ - \left(\mathbf{A}^T\right)^+\right\|.$$

We bound each term separately. We have

$$\left\|\left(\mathbf{A}^T\right)^+\right\| = \left\|\mathbf{A}(\mathbf{A}^T)^{-1}\right\| \leq \left\|\mathbf{A}\right\| ||\mathbf{A}(\mathbf{A}^T)^{-1}|| \leq k 2^{k-1}/(\sigma_k(\mathbf{A}))^2 \leq \zeta^{-O(k)}.$$

35
As for the second term, notice that
\[ \| \hat{A} \hat{A}^T - A A^T \| = \| \hat{A} (\hat{A}^T - A^T) + (\hat{A} - A) A^T \| \leq \| \hat{A} \| \| \hat{A}^T - A^T \| + \| \hat{A} - A \| \| A^T \| \leq \pi_{\min}^{-O(\lg k)} \zeta^{-O(k \lg k)} \varepsilon. \]

Therefore, by Lemma 84 also \( \| (\hat{A} \hat{A}^T)^{-1} - (A A^T)^{-1} \| \leq \pi_{\min}^{-O(\lg k)} \zeta^{-O(k \lg k)} \varepsilon. \) Thus, we get for the second term
\[ \| (\hat{A}^T)^+ - (A^T)^+ \| = \| \hat{A} (\hat{A} \hat{A}^T)^{-1} - A (A A^T)^{-1} \| \leq \| \hat{A} - A \| \| (\hat{A} \hat{A}^T)^{-1} - (A A^T)^{-1} \| \leq \pi_{\min}^{-O(\lg k)} \zeta^{-O(k \lg k)} \varepsilon. \]

Lemma 81. \( \| \tilde{B} - B \| \leq \pi_{\min}^{-O(\lg k)} \zeta^{-O(k \lg k)} \varepsilon. \)

Proof. We can bound \( \tilde{B} - B \) using the same tools as in the previous bounds. First, we bound
\[ \| \tilde{C}_{B,A} - C_{B,A} \| \| (\hat{A}^T)^+ \| \| \hat{\pi}_{\odot}^{-1} \| \leq \pi_{\min}^{-1} \zeta^{-O(k)} \varepsilon. \]

Next,
\[ \| \tilde{C}_{B,A} \| \| (\hat{A}^T)^+ - (A^T)^+ \| \| \hat{\pi}_{\odot}^{-1} \| \leq \pi_{\min}^{-O(\lg k)} \zeta^{-O(k \lg k)} \varepsilon. \]

Finally,
\[ \| \tilde{C}_{B,A} \| \| (\hat{A}^T)^+ \| \| \hat{\pi}_{\odot}^{-1} - \pi_{\odot}^{-1} \| \leq \pi_{\min}^{-O(\lg k)} \zeta^{-O(k \lg k)} \varepsilon. \]

The resulting bound is
\[ \| \tilde{B} - B \| = \| \tilde{C}_{B,A} (\hat{A}^T)^+ \hat{\pi}_{\odot}^{-1} - C_{B,A} (A^T)^+ \pi_{\odot} \| \leq \| \tilde{C}_{B,A} - C_{B,A} \| \| (\hat{A}^T)^+ \| \| \hat{\pi}_{\odot}^{-1} \| + \| \tilde{C}_{B,A} \| \| (\hat{A}^T)^+ - (A^T)^+ \| \| \hat{\pi}_{\odot}^{-1} \| + \| \tilde{C}_{B,A} \| \| (\hat{A}^T)^+ \| \| \hat{\pi}_{\odot}^{-1} - \pi_{\odot}^{-1} \| \leq \pi_{\min}^{-O(\lg k)} \zeta^{-O(k \lg k)} \varepsilon. \]

Corollary 82. \( \| (B^T)^+ \| \leq \zeta^{-O(k)} \).

Proof. It follows the footsteps of the proof of Lemma 80.

Theorem 83. Algorithm 7 will compute \( \hat{m}_i \) satisfying, for all \( i \in [n] \),
\[ \| \hat{m}_i - m_i \|_\infty \leq \pi_{\min}^{-O(\lg k)} \zeta^{-O(k \lg k)} \varepsilon. \]
Proof. We’ll compute the bound using the inversion of $\tilde{\mathbf{B}}^T$ since this will give us the worst case. Let $\tilde{y} = (\tilde{g}(R \cup \{i}\})_{R \in \mathcal{B}}$, and let $y = (g(R \cup \{i}\})_{R \in \mathcal{B}}$. We note that $\|\tilde{y} - y\| \leq \zeta^{-O(k)}\varepsilon$, by assumption on the sample size, and $\|\tilde{y}\| \leq 2^{k-1}$. Then,

$$\|\tilde{y} - y\| \|\tilde{\mathbf{B}}^T\|\|\tilde{\pi}^{-1}\| \leq \pi_{\min}^{-O(1)}\zeta^{-O(k)}\varepsilon,$$

and

$$\|\tilde{y}\| \|\tilde{\mathbf{B}}^T\| - \|\mathbf{B}^T\|\|\tilde{\pi}^{-1}\| \leq \pi_{\min}^{-O(\lg k)}\zeta^{-O(k\lg k)}\varepsilon,$$

So that we get

$$\|\tilde{m}_i - m_i\| = \|\tilde{y}(\tilde{\mathbf{B}}^T)^{+}\tilde{\pi}_{\odot}^{-1} - y(\mathbf{B}^T)^{+}\pi_{\odot}^{-1}\|$$

$$\leq \|\tilde{y} - y\| \|\tilde{\mathbf{B}}^T\|\|\tilde{\pi}_{\odot}^{-1}\| + \|\tilde{y}\| \|\tilde{\mathbf{B}}^T\| - \|\mathbf{B}^T\|\|\pi_{\odot}^{-1}\|$$

$$\leq \pi_{\min}^{-O(\lg k)}\zeta^{-O(k\lg k)}\varepsilon.$$

C.3 Miscellaneous Proofs

Lemma 84. For an invertible $n \times n$ matrix $M$ and a perturbed matrix $\tilde{M}$, if $\|\tilde{M} - M\| = \varepsilon \leq \sigma_n(M)/2$, then

$$\|\tilde{M}^{-1} - M^{-1}\| \leq 2\|M^{-1}\|^2 \varepsilon, \quad \text{and} \quad \|\tilde{M}^{-1}\| \leq 2\|M^{-1}\|.$$

Proof. First, we observe that

$$\|\tilde{M}^{-1}\| = \frac{1}{\sigma_n(M)} \leq \frac{1}{\sigma_n(M) - \sigma_n(M)/2} \leq 2\|M^{-1}\|.$$

We use the identity $\tilde{M}^{-1} - M^{-1} = M^{-1}(M - \tilde{M})M^{-1}$.

$$\|\tilde{M}^{-1} - M^{-1}\| = \|M^{-1}(M - \tilde{M})M^{-1}\| \leq 2\|M^{-1}\|^2\|M - \tilde{M}\|. \quad \square$$
References

[AHK12] A. Anandkumar, D. J. Hsu, and S. M. Kakade. A method of moments for mixture models and hidden Markov models. In Proc. 25th Ann. Conf. on Learning Theory - COLT, volume 23 of JMLR Proceedings, pages 33.1–33.34, 2012. URL: http://proceedings.mlr.press/v23/anandkumar12/anandkumar12.pdf.

[BGK04] T. Batu, S. Guha, and S. Kannan. Inferring mixtures of Markov chains. In Proc. 17th Conf. on Learning Theory, pages 186–199, 2004. doi:10.1007/978-3-540-27819-1_13.

[CGG01] M. Cryan, L. Goldberg, and P. Goldberg. Evolutionary trees can be learned in polynomial time in the two state general Markov model. SIAM J. Comput., 31(2):375–397, 2001. doi:10.1137/S0097539798342496.

[CM19] S. Chen and A. Moitra. Beyond the low-degree algorithm: mixtures of subcubes and their applications. In Proc. 51st Ann. ACM Symp. on Theory of Computing, pages 869–880, 2019. doi:10.1145/3313276.3316375.

[CR08] K. Chaudhuri and S. Rao. Learning mixtures of product distributions using correlations and independence. In Proc. 21st Ann. Conf. on Learning Theory - COLT, pages 9–20. Omnipress, 2008. URL: http://colt2008.cs.helsinki.fi/papers/7-Chaudhuri.pdf.

[dP95] R. de Prony. Essai expérimentale et analytique. J. Écol. Polytech., 1(2):24–76, 1795.

[EH81] B. S. Everitt and D. J. Hand. Mixtures of discrete distributions. In Finite Mixture Distributions, pages 89–105. Springer Netherlands, Dordrecht, 1981.

[FM99] Y. Freund and Y. Mansour. Estimating a mixture of two product distributions. In Proc. 12th Ann. Conf. on Computational Learning Theory, pages 53–62, July 1999. doi:10.1145/307400.307412.

[FOS08] J. Feldman, R. O’Donnell, and R. A. Servedio. Learning mixtures of product distributions over discrete domains. SIAM J. Comput., 37(5):1536–1564, 2008. doi:10.1137/060670705.

[GKV16] R. Gupta, R. Kumar, and S. Vassilvitskii. On mixtures of Markov chains. In Advances in Neural Information Processing Systems, volume 29, 2016. URL: https://proceedings.neurips.cc/paper/2016/file/8b5700012be65c9da25f49408d959ca0-Paper.pdf.

[GMRS20] S. L. Gordon, B. Mazaheri, Y. Rabani, and L. J. Schulman. The sparse Hausdorff moment problem, with application to topic models. 2020. URL: https://arxiv.org/abs/2007.08101.

[GMRS21] S. L. Gordon, B. Mazaheri, Y. Rabani, and L. J. Schulman. Source identification for mixtures of product distributions. In Proc. 34th Ann. Conf. on Learning Theory - COLT, volume 134 of Proceedings of Machine Learning Research, pages 2193–2216. PMLR, 2021. URL: http://proceedings.mlr.press/v134/gordon21a.html.

[GZS19] C. Glymour, K. Zhang, and P. Spirtes. Review of causal discovery methods based on graphical models. Frontiers in Genetics, 10:524, 2019. doi:10.3389/fgene.2019.00524.
[Pea14] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Elsevier, 2014.

[PJS17] J. Peters, D. Janzing, and B. Schölkopf. *Elements of Causal Inference*. MIT Press, 2017.

[RP18] R. Ranganath and A. Perotte. Multiple causal inference with latent confounding. 2018. URL: https://arxiv.org/abs/1805.08273.

[RSS14] Y. Rabani, L. J. Schulman, and C. Swamy. Learning mixtures of arbitrary distributions over large discrete domains. In *Proc. 5th Conf. on Innovations in Theoretical Computer Science*, pages 207–224, 2014. doi:10.1145/2554797.2554818.

[SGS00a] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction and Search*. MIT Press, second edition, 2000.

[SGS00b] P. Spirtes, C. Glymour, R. Scheines, S. Kauffman, V. Aimele, and F. Wimberly. Constructing Bayesian network models of gene expression networks from microarray data. 2000.

[SKLV17] V. Sharan, S. M. Kakade, P. Liang, and G. Valiant. Learning overcomplete HMMs. In *Advances in Neural Information Processing Systems*, pages 940–949, 2017. URL: https://arxiv.org/abs/1711.02309.

[SP06] I. Shpitser and J. Pearl. Identification of joint interventional distributions in recursive semi-Markovian causal models. In *Proc. 20th AAAI Conference on Artificial Intelligence*, pages 1219–1226, 2006. URL: https://dl.acm.org/doi/10.5555/1597348.1597382.

[TMCH98] B. Thiesson, C. Meek, D. M. Chickering, and D. Heckerman. Learning mixtures of DAG models. In *Proc. 14th Conf. on Uncertainty in Artificial Intelligence*, page 504–513, 1998. URL: https://dl.acm.org/doi/10.5555/2074094.2074154.

[TMMA18] B. Tahmasebi, S. A. Motahari, and M. A. Maddah-Ali. On the identifiability of finite mixtures of finite product measures. (Also in “On the identifiability of parameters in the population stratification problem: A worst-case analysis,” Proc. ISIT pp. 1051-1055), 2018. URL: https://arxiv.org/abs/1807.05444.

[TSM85] D. M. Titterington, A. F. M. Smith, and U. E. Makov. *Statistical Analysis of Finite Mixture Distributions*. John Wiley and Sons, Inc., 1985.

[WB19] Y. Wang and D. M. Blei. The blessings of multiple causes. *Journal of the American Statistical Association*, 114(528):1574–1596, 2019. doi:10.1080/01621459.2019.1686987.