Learning latent causal graphs via mixture oracles

Bohdan Kivva
bkivva@uchicago.edu
University of Chicago

Goutham Rajendran
goutham@uchicago.edu
University of Chicago

Pradeep Ravikumar
pradeepr@cs.cmu.edu
Carnegie Mellon University

Bryon Aragam
bryon@chicagobooth.edu
University of Chicago

November 23, 2021

Abstract

We study the problem of reconstructing a causal graphical model from data in the presence of latent variables. The main problem of interest is recovering the causal structure over the latent variables while allowing for general, potentially nonlinear dependencies. In many practical problems, the dependence between raw observations (e.g. pixels in an image) is much less relevant than the dependence between certain high-level, latent features (e.g. concepts or objects), and this is the setting of interest. We provide conditions under which both the latent representations and the underlying latent causal model are identifiable by a reduction to a mixture oracle. These results highlight an intriguing connection between the well-studied problem of learning the order of a mixture model and the problem of learning the bipartite structure between observables and unobservables. The proof is constructive, and leads to several algorithms for explicitly reconstructing the full graphical model. We discuss efficient algorithms and provide experiments illustrating the algorithms in practice.

1 Introduction

Understanding causal relationships between objects and/or concepts is a core component of human reasoning, and by extension, a core component of artificial intelligence [Pea88, LM11]. Causal relationships are robust to perturbations, encode invariances in a system, and enable agents to reason effectively about the effects of their actions in an environment. Broadly speaking, the problem of inferring causal relationships can be broken down into two main steps: 1) The extraction of high-level causal features from raw data, and 2) The inference of causal relationships between these high-level features. From here, one may consider estimating the magnitude of causal effects, the effect of interventions, reasoning about counterfactuals, etc. Our focus in this paper will be the problem of learning causal relationships *between* latent variables, which is closely related to the problem of learning causal representations [SLB$^+$21]. This problem should be contrasted with the equally important problem of causal inference

in the presence of latent confounders [e.g. CMKR12, SMR13, AHJK13, HSK06, SSGS06]; see also Remark 2.1.

Causal graphical models [Pea09, Pea88] provide a natural framework for this problem, and have long been used to model causal systems with hidden variables [RS$^+$02, Eva16, ER14, E$^+$18, ER$^+$19, RERS17]. It is well-known that in general, without additional assumptions, a causal graphical model given by a directed acyclic graph (DAG) is not identifiable in the presence of latent variables [e.g., Pea88, SGS00]. In fact, this is a generic property of nonparametric structural models: Without assumptions, identifiability is impossible, however, given enough structure, identifiability can be rescued. Examples of this phenomenon include linearity [FNM17, CPW$^+$12, AHJK13, XCH$^+$20, And84], independence [AMR09, BJR16, XCH$^+$20], rank [FNM17, CPW$^+$12], sparsity [And84], and graphical constraints [AHJK13, AV$^+$13].

In this paper, we consider a general setting for this problem with *discrete* latent variables, while allowing otherwise arbitrary (possibly nonlinear) dependencies. The latent causal graph between the latent variables is also allowed to be arbitrary: No assumptions are placed on the structure of this DAG. We do not assume that the number of hidden variables, their state spaces, or their relationships are known; in fact, we provide explicit conditions under which all of this can be recovered uniquely. To accomplish this, we highlight a crucial reduction between the problem of learning a DAG model over these variables—given access only to the observed data—and learning the parameters of a finite mixture model. This observation leads to new identifiability conditions and algorithms for learning causal models with latent structure.

Overview Our starting point is a simple reduction of the graphical model recovery problem to three modular subproblems:

1. The bipartite graph Γ between hidden and observed nodes,
2. The latent distribution $\mathbb{P}(H)$ over the hidden variables H, and
3. A directed acyclic graph (DAG) Λ over the latent distribution.

From here, the crucial observation is to reduce the recovery problems for Γ and $\mathbb{P}(H)$ to the problem of learning a finite mixture over the observed data. The latter is a well-studied problem with many practical algorithms and theoretical guarantees. We do not require parametric assumptions on this mixture, which allows for very general dependencies between the observed and hidden variables. From this mixture model, we extract what is needed to learn the full graph structure.

This perspective leads to a systematic, modular approach for learning the latent causal graph via mixture oracles (see Section 2 for definitions). Ultimately, the application of these ideas requires a practical implementation of this mixture oracle, which is discussed in Section 6.

Contributions More precisely, we make the following contributions:

1. (Section 3) We provide general conditions under which the latent causal model G is identifiable (Theorem 3.2). Surprisingly, these conditions mostly amount to nondegen-

eracy conditions on the joint distribution. As we show, without these assumptions identifiability breaks down and reconstruction becomes impossible.

2. (Section 4) We carefully analyze the problem of reconstructing Γ under progressively weaker assumptions: First, we derive a brute-force algorithm that identifies Γ in a general setting (Theorem 4.2), and then under a linear independence condition we derive a polynomial-time algorithm based on tensor decomposition and Jennrich's algorithm (Theorem 4.8).

3. (Section 5) Building on top of the previous step, where we learn the bipartite graph and sizes of the domains of latent variables, we develop an efficient algorithm for learning the latent distribution $\mathbb{P}(H)$ from observed data (Theorem 5.4).

4. (Section 6-7) We implement these algorithms as part of an end-to-end pipeline for learning the full causal graph and illustrate its performance on simulated data.

A prevailing theme throughout is the fact that the hidden variables leave a recognizable "signature" in the observed data through the marginal mixture models induced over subsets of observed variables. By cleverly exploiting these signatures, the number of hidden variables, their states, and their relationships can be recovered exactly.

Previous work Latent variable graphical models have been extensively studied in the literature; as such we focus only on the most closely related work on causal graphical models here. Early work on this problem includes seminal work by Martin and VanLehn [MV95], Friedman et al. [F$^+$97] Elidan et al. [ELFK00]. More recent work has focused on linear models [AHJK13, FNM17, SSGS06, XCH$^+$20] or known structure [KSDV17, DDF21, SLD$^+$20]. When the structure is not known *a priori*, we find ourselves in the realm of *structure learning*, which is our focus. Less is known regarding structure learning between latent variables for nonlinear models, although there has been recent progress based on nonlinear ICA [MZH20, KKMH20]. For example, [YLC$^+$20] proposed CausalVAE, which assumes a linear structural equation model and knowledge of the concept labels for the latent variables, in order to leverage the iVAE model from [KKMH20]. By contrast, our results make no linearity assumptions and do not require these additional labels. While this paper was under review, we were made aware of the recent work [MGW20] that studies a similar problem to ours in a general, nonlinear setting under faithfulness assumptions. It is also worth noting recent progress on learning discrete Boltzmann machines [BKM19, BB20], which can be interpreted as an Ising model with a bipartite structure and a single hidden layer—in particular, there is no hidden causal structure. Nevertheless, this line of work shows that learning Boltzmann machines is computationally hard in a precise sense. More broadly, the problem of learning latent structure has been studied in a variety of other applications including latent Dirichlet allocation [AGM12, AGH$^+$13], phylogenetics [MR05, SS03], and hidden Markov models [AHK12, GCR13].

A prevailing theme in the causal inference literature has been negative results asserting that in the presence of latent variables, causal inference is impossible [GKS20, RSSW03, RW99, D'A19]. Our results do not contradict this important line of work, and instead adopts a more optimistic tone: We show that under reasonable assumptions—essentially that the latent variables are

discrete and well-separated—identifiability and exact recovery of latent causal relationships is indeed possible. This optimistic approach is implicit in recent progress on visual relationship detection [ND17], causal feature learning [CEP17, LPNC$^+$17], and interaction modeling [LTA$^+$20, KFW$^+$18]. In this spirit, our work provides theoretical grounding for some of these ideas.

Mixture models and clustering While our theoretical results in Sections 3-5 assume access to a mixture oracle (see Definition 2.5), in Section 6 we discuss how this oracle can be implemented in practice. To provide context for these results, we briefly mention related work on learning mixture models from data. Mixture models can be learned under a variety of parametric and nonparametric assumptions. Although much is known about parametric models [e.g. Lin95], of more interest to us are nonparametric models in which the mixture components are allowed to be flexible, such as mixtures of product distributions [HZ03, GS21], grouped observations [RVS20, VS16] and general nonparametric mixtures [ADXR20, SBY09]. In each of these cases, a mixture oracle can be implemented without parametric assumptions. In practice, we use clustering algorithms such as K-means or hierarchical clustering to implement this oracle. We note also that the specific problem of consistently estimating the order of a mixture model, which will be of particular importance in the sequel, has been the subject of intense scrutiny in the statistics literature [e.g. MK20, Kol00, DCG$^+$97, CK09].

Broader impacts and societal impact Latent variable models have numerous practical applications. Many of these applications positively address important social problems, however, these models can certainly be applied nefariously. For example, if the latent variables represent private, protected information, our results imply that this hidden private data can be leaked into publicly released data, which is obviously undesirable. Understanding how to infer unprotected data while safeguarding protected data is an important problem, and our results shed light on when this is and isn't possible.

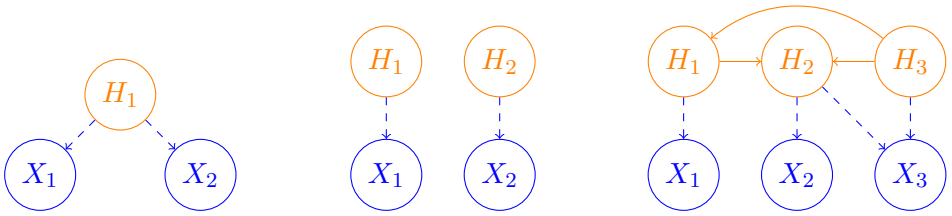
Notation We say that a distribution $\mathbb{P}(V)$ satisfies the *Markov property* with respect to a DAG $G = (V, E)$ if

$$\mathbb{P}(V) = \prod_{v \in V} \mathbb{P}(v \mid \mathrm{pa}_G(v)). \tag{1}$$

An important consequence of the Markov property is that it allows one to read off conditional independence relations from the graph G. More specifically, we have the following [see Pea88, SGS00, for details]:

- For each $v \in V$, v is independent of its non-descendants, given its parents.
- For disjoint subsets $V_1, V_2, V_3 \subset V$, if V_1 and V_2 are d-separated given V_3 in G, then $V_1 \perp\!\!\!\perp V_2 \,|\, V_3$ in $\mathbb{P}(V)$.

The concept of d-separation (see §3.3.1 in [Pea88] or §2.3.4 in [SGS00]) gives rise to a set of independence relations, often denoted by $\mathcal{I}(G)$. The Markov property thus implies that $\mathcal{I}(G) \subset \mathcal{I}(V)$, where $\mathcal{I}(V)$ is the collection of all valid conditional independence relations



(a) A single hidden state.

(b) Two independent hidden states.

(c) Three dependent hidden states.

Figure 1: Illustration of the basic model. Note that there are no edges between observed variables or edges oriented from observed to hidden. (a) A latent variable model with a single hidden state; i.e. a mixture model. (b)-(c) Two examples of latent variable models with more complicated hidden structure.

over V. When the reverse inclusion holds, we say that $\mathbb{P}(V)$ is *faithful* to G (also that G is a *perfect map* of V). Although the concepts of faithfulness and d-separation will not be needed in the sequel, we have included this short discussion for completeness and context (cf. Section 3).

Throughout this paper, we use standard notation such as $\mathrm{pa}(j)$ for parents, $\mathrm{ch}(j)$ for children, and $\mathrm{ne}(j)$ for neighbors. Specifically, we define

- The parents of a node $v \in V$ are denoted by $\mathrm{pa}(v) = \{u \in V : (u, v) \in E\}$;
- The children of a node $v \in V$ are denoted by $\mathrm{ch}(v) = \{u \in V : (v, u) \in E\}$;
- The neighborhood of a node $v \in V$ is denoted by $\mathrm{ne}(v) = \mathrm{pa}(v) \cup \mathrm{ch}(v)$.

Given a subset $V' \subset V$, $\mathrm{pa}(V') := \cup_{j \in V'} \mathrm{pa}(j)$ and given a subgraph $G' \subset G$, $\mathrm{pa}_{G'}(V') := \mathrm{pa}(V') \cap G'$, with similar notation for children and neighbors. We let $A \in \{0,1\}^{|X| \times |H|}$ denote the adjacency matrix of Γ and denote its columns by $a_j \in \{0,1\}^{|X|}$. Finally, we adopt the convention that H is identified with the indices $[m] = \{1, \ldots, m\}$, and similar X is identified with $[n] = \{1, \ldots, n\}$. In particular, we use $\mathrm{pa}(i)$ and $\mathrm{pa}(H_i)$ interchangeably when the context is clear.

2 Background

Let $G = (V, E)$ be a DAG with $V = (X, H)$, where $X \in \mathbb{R}^n$ denotes the observed part and $H \in \Omega := \Omega_1 \times \cdots \times \Omega_m$ denotes the hidden, or latent, part. Throughout this paper, we assume that each Ω_i is a discrete space with $|\Omega_i| \geq 2$. We assume further that there are no edges between observed variables and no edges from observed to hidden variables, and that the distribution of V satisfies the Markov property with respect to G (see the supplement for definitions). Under these assumptions, G decomposes as the union of two subgraphs $G = \Gamma \cup \Lambda$, where Γ is a directed, bipartite graph of edges pointing from H to X, and Λ is a DAG over the latent variables H. Similar assumptions have appeared in previous work

[AHJK13, XCH+20, MGW20], and although nontrivial, they encapsulate our keen interest in reconstructing the structure Λ amongst the latent variables, and captures relevant applications where the relationships between raw observations is less relevant than so-called "causal features" [CPE14, CEP17]. See Figure 1 for examples.

Throughout this paper, we use standard notation such as $\mathrm{pa}(j)$ for parents, $\mathrm{ch}(j)$ for children, and $\mathrm{ne}(j)$ for neighbors. Given a subset $V' \subset V$, $\mathrm{pa}(V') := \cup_{j \in V'} \mathrm{pa}(j)$ and given a subgraph $G' \subset G$, $\mathrm{pa}_{G'}(V') := \mathrm{pa}(V') \cap G'$, with similar notation for children and neighbors. We let $A \in \{0,1\}^{|X| \times |H|}$ denote the adjacency matrix of Γ and denote its columns by $a_j \in \{0,1\}^{|X|}$.

Remark 2.1. *Our goal is to learn the hidden variables H and the causal graph between them, defined above by Λ. To accomplish this, our main result (Theorem 3.2) shows how to identify $(\Gamma, \mathbb{P}(H))$, from which Λ can be recovered (see Section 3 for details). It is important to contrast this problem with problems involving latent confounders [e.g. CMKR12, SMR13, AHJK13, HSK06, SSGS06], where the goal is to learn the causal graph between the observed variables X. In our setting, there are no edges between the observed variables.*

2.1 Assumptions

It is well-known that without additional assumptions, the latent variables H cannot be identified from X, let alone the DAG Λ. For example, we can always replace a pair of distinct hidden variables H_i and H_j with a single hidden variable H_0 that takes values in $\Omega_i \times \Omega_j$. Similarly, a single latent variable can be split into two or more latent variables. In order to avoid this type of degeneracy, we make the following assumptions:

Assumption 2.2 (No twins)**.** *For any hidden variables $H_i \neq H_j$ we have $\mathrm{ne}_\Gamma(H_i) \neq \mathrm{ne}_\Gamma(H_j)$.*

Assumption 2.3 (Maximality)**.** *There is no DAG $G' = ((X, H'), E')$ such that:*

1. *$\mathbb{P}(X, H')$ is Markov with respect to G';*
2. *G' is obtained from G by splitting a hidden variable (equivalently, G is obtained from G' by merging a pair of vertices);*
3. *G' satisfies Assumption 2.2.*

These assumptions are necessary for the recovery of Λ in the sense that, without these assumptions, latent variables can be created or destroyed without changing the observed distribution $\mathbb{P}(X)$. Informally, the maximality assumption says that if there are several DAGs that are Markov with respect to the given distribution, we are interested in recovering the most informative among them. Finally, we make a mild assumption on the probabilities, in order to avoid degenerate cases where certain configurations of the latent variables have zero probability:

Assumption 2.4 (Nondegeneracy)**.** *The distribution over $V = (X, H)$ satisfies:*

(a) $\mathbb{P}(H = h) > 0$ *for all* $h \in \Omega_1 \times \ldots \times \Omega_k$.

(b) For all $S \subset X$ and $a \neq b$, $\mathbb{P}(S|\operatorname{pa}(S) = a) \neq \mathbb{P}(S|\operatorname{pa}(S) = b)$, where a and b are distinct configurations of $\operatorname{pa}(S)$.

Without this nondegeneracy condition, H cannot be identified; see Appendix A for details.

2.2 Mixture oracles

Let $S \subset X$ be a subset of the observed variables. We can always write the marginal distribution $\mathbb{P}(S)$ as

$$\mathbb{P}(S) = \sum_{h \in \Omega} \mathbb{P}(H = h)\mathbb{P}(S \mid H = h). \tag{2}$$

When $S = X$, this can be interpreted as a mixture model with $K := |\Omega|$ components. When $S \subsetneq X$, however, multiple components can "collapse" onto the same component, resulting in a mixture with fewer than K components. Let $k(S)$ denote this number, so that we may define a discrete random variable Z with $k(S)$ states such that for all $j \in [k(S)]$, we have

$$\mathbb{P}(S) = \sum_{j=1}^{k(S)} \underbrace{\mathbb{P}(Z = j)}_{:=\pi(S,j)} \underbrace{\mathbb{P}(S \mid Z = j)}_{:=C(S,j)} = \sum_{j=1}^{k(S)} \pi(S, j)C(S, j). \tag{3}$$

Then $\pi(S, j)$ is the weight of the jth mixture component over S, and $C(S, j)$ is the corresponding jth component. It turns out that these probabilities precisely encode the conditional independence structure of H. To make this formal, we define the following oracle:

Definition 2.5. *A* mixture oracle *is an oracle that takes $S \subset X$ as input and returns the number of components $k(S)$ as well as the weights $\pi(S, j)$ and components $C(S, j)$ for each $j \in [k(S)]$. This oracle will be denoted by* $\mathsf{MixOracle}(S)$.

Although our theoretical results are couched in the language of this oracle, we provide practical implementation details in Section 6 and experiments to validate our approach in Section 7.

A sufficient condition for the existence of a mixture oracle is that the mixture model over X is identifiable. This is because identifiability implies that the number of components K, the weights $\mathbb{P}(Z = j)$, and the mixture components $\mathbb{P}(X \mid Z = j)$ are determined by $\mathbb{P}(X)$. The marginal weights $\pi(S, j)$ and components $C(S, j)$ can then be recovered by simply projecting the full mixture over X onto S.

Remark 2.6. *In fact, we do not need the full power of* MixOracle. *For our algorithms it is sufficient to have access to $k(S)$ for a sufficiently large family of $S \subset X$, the list of weights $\pi(X, j)$, and a map that relates components in the full mixture over X to the components in the marginal mixtures over each variable X_i (see Section 5 for details).*

Before concluding this section, we note an important consequence of Assumption 2.4 that will be used in the sequel:

Observation 2.7. *Under Assumption 2.4, for any $S \subseteq X$*

$$k(S) = \prod_{H_i \in \mathrm{pa}(S)} \dim(H_i) =: \dim(\mathrm{pa}(S)).$$

Proof. By the Markov property, S is independent of $H \setminus \mathrm{pa}(S)$. There are $\dim(\mathrm{pa}(S))$ possible assignments to the hidden variables in $\mathrm{pa}(S)$ and by Assumption 2.4, distinct assignments to the hidden variables induce distinct components in the marginal distribution $P(S)$. Hence, by definition, $k(S) = \dim(\mathrm{pa}(S))$. □

3 Recovery of the latent causal graph

We first consider the oracle setting in which we have access to $\mathsf{MixOracle}(S)$.

Observe that the problem of learning G can be reduced to learning $(\Gamma, \mathbb{P}(H))$: Since we can decompose G into a bipartite subgraph Γ and a latent subgraph Λ, it suffices to learn these two components separately. We then further reduce the problem of learning Λ to learning the latent distribution $\mathbb{P}(H)$. First, we will show how to reconstruct Γ from $\mathsf{MixOracle}(S)$. Then, we will show how to learn the latent distribution $\mathbb{P}(H)$ from $\mathsf{MixOracle}(S)$.

Thus, the problem of learning G is reduced to the mixture oracle:

$$G \to (\Gamma, \mathbb{P}(H)) \to \mathsf{MixOracle}(S).$$

In the sequel, we focus our attention on recovering $(\Gamma, \mathbb{P}(H))$. In order to recover $\mathbb{P}(H)$, we will require the following assumption:

Assumption 3.1 (Subset condition)**.** *We say that the bipartite graph Γ satisfies the subset condition (SSC) if for any pair of distinct hidden variables H_i, H_j the set $\mathrm{ne}_\Gamma(H_i)$ is not a subset of $\mathrm{ne}_\Gamma(H_j)$.*

This assumption is weaker than the common "anchor words" assumption from the topic modeling literature. The latter assumption says that every topic has a word that is unique to this topic, and it is commonly assumed for efficient recovery of latent structure [AGM12, AGH$^+$13].

Under Assumption 3.1, we have the following key result:

Theorem 3.2. *Under Assumptions 2.2, 2.3, 2.4, and 3.1, $(\Gamma, \mathbb{P}(H))$ can be reconstructed from $\mathbb{P}(X)$ and $\mathsf{MixOracle}(S)$. Furthermore, if additionally the columns of the bipartite adjacency matrix A are linearly independent, there is an efficient algorithm for this reconstruction.*

The proof is constructive and leads to an efficient algorithm as alluded to in the previous theorem. An overview of the main ideas behind the proof of this result are presented in Sections 4 and 5; the complete proof of this theorem can be found in Appendices B-D.

As presented, Theorem 3.2 leaves two aspects of the problem unresolved: 1) Under what conditions does $\mathsf{MixOracle}(S)$ exist, and 2) How can we identify Λ from $\mathbb{P}(H)$? As it turns

out, each of these problems is well-studied in previous work, which explains our presentation of Theorem 3.2. For completeness, we address these problems briefly below.

Existence of $\mathsf{MixOracle}(S)$ A mixture oracle exists if the mixture model over X is identifiable. As discussed in Section 1, such identifiability results are readily available in the literature. For example, assume that for every $S \subseteq X$, the mixture model (2) comes from any of the following families:

1. a mixture of gaussian distributions [Tei63, YS68], or
2. a mixture of Gamma distributions [Tei63], or
3. an exponential family mixture [YS68], or
4. a mixture of product distributions [Tei67], or
5. a well-separated (i.e. in TV distance) nonparametric mixture [ADXR20].

Then $(\Gamma, \mathbb{P}(H))$ is identifiable. The list above is by no means exhaustive, and many other results on identifiability of mixture models are known (e.g., see the survey [MLR19]).

Identifiability of Λ Once we know $\mathbb{P}(H)$ (e.g. via Theorem 3.2), identifying Λ from $\mathbb{P}(H)$ is a well-studied problem with many solutions [SGS00, Pea88]. For simplicity, it suffices to assume that $\mathbb{P}(H)$ is faithful to Λ, which implies that Λ can be learned up to Markov equivalence. This assumption is *not* necessary, and any number of alternative identifiability assumptions on $\mathbb{P}(H)$ can be plugged in place of faithfulness, for example triangle faithfulness [SZ14], independent noise [SHHK06, PMJS14], post-nonlinearity [ZH09], equality of variances [PB13, GDA20], etc.

4 Learning the bipartite graph

In this section we outline the main ideas behind the recovery of Γ in Theorem 3.2. We begin by establishing conditions that ensure Γ is identifiable, and then proceed to consider efficient algorithms for its recovery.

4.1 Identifiability result

We study a slightly more general setup in which the identifiability of Γ depends on how much information we request from the $\mathsf{MixOracle}$. Clearly, we want to rely on $\mathsf{MixOracle}$ as little as possible. As the proofs in the supplement indicate, the only information required for this step are the number of components. Neither the weights nor the components are needed.

Definition 4.1. *We say that* Γ *is* t*-recoverable if* Γ *can be uniquely recovered from* X *and the sequence* $(\mathsf{MixOracle}(S) \mid |S| \leq t)$.

Theorem 4.2. *Let* Γ *be the bipartite graph between* X *and* H.

(a) Assume that $\mathrm{ne}_\Gamma(H_i) \neq \mathrm{ne}_\Gamma(H_j)$ *for any* $i \neq j$. *Then* Γ *and* $\dim(H_i)$ *are* n*-recoverable.*

(b) Let $t \geq 3$. Assume that for every $S \subseteq H$ with $|S| \geq 2$ we have

$$\dim \operatorname{span}\{a_j \mid j \in S\} \geq \frac{2}{t}|S| + 1,$$

then Γ and $\dim(H_i)$ are t-recoverable.

Note that Assumption 3.1 implies the assumption in Theorem 4.2(a). Finally, as in Section 2, we argue that in the absence of additional assumptions, this assumption is in fact necessary:

Observation 4.3. *If there is a pair of distinct variables $H_i, H_j \in H$ such that $\mathrm{ne}_\Gamma(H_1) = \mathrm{ne}_\Gamma(H_2)$, then Γ is not n-recoverable.*

4.2 Ideas behind the recovery

In Corollary 4.4 below, we recast Observation 2.7 as an additive identity. This transforms the problem of learning Γ into an instance of more general problem that is discussed in the appendix. The results of this section apply to this more general version.

Corollary 4.4. *Assume that Assumptions 2.4 hold. For $H_i \in H$ define $w(H_i) = \log(\dim(H_i))$. Then for every set $S \subseteq X$*

$$\log(k(S)) = \sum_{H_i \in \mathrm{pa}(S)} w(H_i). \tag{4}$$

In order to argue about the causal structure of the hidden variables we first need to identify the variables themselves. By Assumption 2.2, every hidden variable leaves a "signature" among the observed variables, which is the set $\mathrm{ne}_\Gamma(H_i)$ of observed variables it affects. In particular, note that $H_i \in \bigcap_{X_s \in \mathrm{ne}_\Gamma(H_i)} \mathrm{pa}(X_s)$, and if there is no H_j with $\mathrm{ne}_\Gamma(H_i) \subset \mathrm{ne}_\Gamma(H_j)$, then H_i is the unique element of the intersection. The lemma above allows us to extract information about the union of parent sets, and we wish to turn it into the information about intersections. This motivates the following definitions.

Definition 4.5. *Let Γ and w be as above. Define*

$$\mathrm{sne}_\Gamma(S) = \bigcap_{x \in S} \mathrm{ne}_\Gamma(x) \quad \textit{and} \quad \mathrm{Wsne}_\Gamma(S) = \sum_{v \in \mathrm{sne}_\Gamma(S)} w(v) \tag{5}$$

Lemma 4.6. *For a set $S \subseteq X$ we have*

$$\mathrm{Wsne}_\Gamma(S) = \sum_{U \subseteq S, U \neq \emptyset} (-1)^{|U|+1} W_\Gamma(U), \quad \textit{where} \quad W_\Gamma(S) = \sum_{v \in \mathrm{ne}_\Gamma(S)} w(v). \tag{6}$$

The proof of this lemma is a simple application of the Inclusion-Exclusion principle.

Remark 4.7. *The RHS of Eq.* (6) *only depends on W evaluated on subsets of S. Thus, in particular, if $|S| \leq t$ to compute* $\mathrm{Wsne}(S)$ *it is enough to know* MixOracle *on all sets of size $\leq t$.*

Finally, the values of the function Wsne_Γ can be organized into a tensor, and from here the problem of learning Γ can be cast as decomposition problem for this tensor. These proof details are spelled out in Appendix B; in the next section we illustrate this procedure for the special case of 3-recovery.

4.3 Efficient 3-recovery

Under a simple additional assumption Γ can be recovered efficiently. We are primarily interested in the case $t = 3$. The main idea is to note that a rank-three tensor involving the columns of A can be written in terms of Wsne_Γ. We can then apply Jennrich's algorithm [Har70] to decompose the tensor and recover these columns, which yield Γ. To see this, let $I = (i_1, i_2, i_3) \subseteq X$ be a triple of indices, and note that

$$\sum_{j \in H} w(j)(a_j)_{i_1}(a_j)_{i_2}(a_j)_{i_3} = \Big(\sum_{j \in H} w(j) a_j \otimes a_j \otimes a_j \Big)_{(i_1,i_2,i_3)} = \mathrm{Wsne}_\Gamma(I). \tag{7}$$

Theorem 4.8. *Assume that the columns of A are linearly independent. Then Γ and $\dim(H_i)$, for all i, are 3-recoverable in $O(n^3)$ space and $O(n^4)$ time.*

Proof. It takes $O(n^3)$ space and $O(n^3)$ time to compute M_3 and then Jennrich's algorithm can decompose the tensor in $O(n^3)$ space and $O(n^4)$ time. □

5 Learning the latent distribution

In this section we outline the main ideas behind the recovery of $\mathbb{P}(H)$ in Theorem 3.2.

Remark 5.1. *Since the variables H are not observed,* $\mathsf{MixOracle}(S)$ *only tells us the set*

$$\{(i, \pi(S, i), C(S, i)) \mid i \in [k(S)]\}.$$

But the correspondence $\Omega \ni h \leftrightarrow j \in [K]$ between a possible tuple h of values of hidden variables and the corresponding mixture component is unknown.

Since the values of H are not observed, we may learn this correspondence only up to a relabeling of Ω_i. By definition, the input distribution has $K = |\Omega|$ mixture components over X and $k_i = k(X_i)$ mixture components over X_i. Fix any enumeration of these components by $[K]$ and $[k_i]$, respectively. To recover the correspondence $\Omega \ni h \leftrightarrow j \in [K]$, we will need access to the map

$$L : [K] \to [k_1] \times \cdots \times [k_n], \tag{8}$$

defined so that $[L(j)]_i$ equals to the index of the mixture component $C(X, j)$ (marginalized over X_i) in the marginal distribution over X_i. Crucially, this discussion establishes that L can be computed from a combination of $\mathsf{MixOracle}(X)$ and $\mathsf{MixOracle}(X_i)$ for each i.

The map L encodes partial information about the causal structure in G. Indeed, if $h_1, h_2 \in \Omega$ are a pair of states of hidden variables H that coincide on $\mathrm{pa}(X_i)$ for some $X_i \in X$, then by

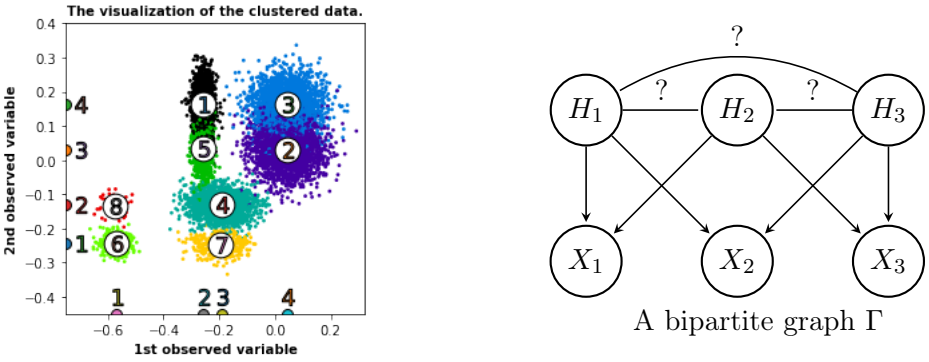


Figure 2: Example of a latent DAG and corresponding mixture distribution

the Markov property the components that correspond to h_1 and h_2 should have the same marginal distribution over X_i.

Example 5.2. *Consider the DAG on Figure 2. We do not make any assumptions about the causal structure between hidden variables. This DAG has* 3 *hidden variables, and we assume that each of them takes values in the set* $\{0,1\}$*. Then by Assumption 2.4, every observed variable is a mixture of* 4 *components, while the distribution on* X *is a mixture of* 8 *components. Note that the anchor word assumption is violated here, while (SSC) assumption is satisfied. The map* $L : [8] \to [4] \times [4] \times [4]$ *for an example as in Fig. 2 has form*

$i:$	1	2	3	4	5	6	7	8
$L(i):$	$(2,4,3)$,	$(4,3,4)$,	$(4,4,2)$,	$(3,2,4)$,	$(2,3,1)$,	$(1,1,3)$,	$(3,1,2)$,	$(1,2,1)$

Our goal is to find the correspondence between $h \in \Omega = \{0,1\}^3$ *and* $i \in [8]$*. (The projection on the third variable is not shown on Figure 2, so the third coordinate of* L *cannot be deduced from the plot.)*

We now show that there is an algorithm that exactly recovers $\mathbb{P}(H)$ from the bipartite graph Γ, the map $L : [K] \to [k_1] \times \cdots \times [k_n]$, and the mixture weights (probabilities) $\{\pi(X,i) \mid i \in [K]\} = \{\mathbb{P}(Z = i) \mid i \in [K]\}$. Each of these inputs can be computed from MixOracle.

Definition 5.3. *Let* J *be an order-*m *tensor whose* i*-th mode is indexed by values of* H_i*, such that* $J(h_1, h_2, \ldots, h_m) = \mathbb{P}(H = h)$*. That is,* J *is the joint probability table of* H*.*

Theorem 5.4. *Suppose Assumptions 2.4 and 3.1 hold. Then the correspondence* $\Omega \ni h \leftrightarrow C(X,i)$ *and the tensor* $J(h_1, h_2, \ldots, h_m) = \mathbb{P}(H = (h_1, h_2, \ldots, h_m))$ *can be efficiently reconstructed from* L*,* Γ *and* $\{\pi(X,i)\}_{i \in [K]}$*.*

Remark 5.5. *If Assumption 3.1 is violated, then in general* J *cannot be reconstructed uniquely and moreover,* G *cannot be uniquely identified. See Appendix C for details.*

5.1 Examples of Algorithm 1

To illustrate this algorithm, in this section we illustrate how it works on Example 5.2. The basic idea is the following: We start by arbitrarily assigning a component $C(X, i)$—and hence its corresponding probability $\pi(X, i)$ to some hidden state $h^* = (h_1, \ldots, h_m)$. This assignment amounts to declaring $\mathbb{P}(H_1 = h_1, \ldots, H_m = h_m) = \pi(X, i)$ and $\mathbb{P}(X \mid H_1 = h_1, \ldots, H_m = h_m) = C(X, i)$. The choice of initial state h^* here is immaterial; this can be done without loss of generality since the values of the hidden variables can be relabeled without changing anything. From here we proceed inductively by considering hidden states that differ from the previously identified states by in exactly one coordinate. In the example below, we start with $h^* = (0, \ldots, 0)$ and then use this as a base case to identify $h^* + e_i$ for each $i = 1, \ldots, m$, where

$$(e_i)_j = \begin{cases} 1 & i = j \\ 0 & i \neq j. \end{cases}$$

Note that h^* and e_i differ in exactly one coordinate. We then repeat this process until all states have been exhausted. The following example illustrates the procedure and explains how Lemma C.1 helps to resolve the ambiguity regarding the assignment of components to hidden states in each step.

Example 5.6. *Consider the DAG G in Fig. 2. It has 3 hidden variables, each of which takes values in $\{0, 1\}$. By Assumption 2.4 every observed variable is a mixture of 4 components, while the distribution on X is a mixture of 8 components. Note that the anchor word assumption is violated here, while SSC (Assumption 3.1) is satisfied. The map $L : [8] \to [4] \times [4] \times [4]$ can be written as:*

i:	1	2	3	4	5	6	7	8
$L(i)$:	$(2,4,3)$,	$(4,3,4)$,	$(4,4,2)$,	$(3,2,4)$,	$(2,3,1)$,	$(1,1,3)$,	$(3,1,2)$,	$(1,2,1)$

We want to find the correspondence between $h \in \Omega = \{0, 1\}^3$ and $i \in [8]$.

We start by picking an arbitrary component, say 1, and assign it to $(H_1, H_2, H_3) = (0, 0, 0)$. Next, we make use of Lemma C.1. Since we know Γ, we know $\mathrm{ch}(H_i)$ for each i. In particular, for the hidden variable H_1, we know $\mathrm{ch}(H_1) = \{X_1, X_2\}$. This implies that if H_2, H_3 are fixed while H_1 changes its value, then the component of X_3 is unchanged. It follows that the third coordinate of L is also unchanged. This gives us a way to pair up the components that have the same third coordinate $L(i)_3$; the pairs are $(1, 6)$, $(2, 4)$, $(3, 7)$ and $(5, 8)$. By our previous observation, these pairs are in one-to-one correspondence with unique states of $(H_1, H_2) = (h_1, h_2)$, and each pair identifies the pair of components $(P(X \mid H_1 = 0, H_2 = h_2, H_3 = h_3), P(X \mid H_1 = 1, H_2 = h_2, H_3 = h_3))$. Note that at this stage, there is still ambiguity as to which coordinate of each pair corresponds to which component.

Similarly, we can pair up the components that correspond to assignments of hidden variables that differ only in the value of H_2. The pairs are $(1, 3)$, $(2, 5)$, $(4, 8)$ and $(6, 7)$. Finally, for H_3 the pairs are $(1, 5)$, $(2, 3)$, $(4, 7)$ and $(6, 8)$.

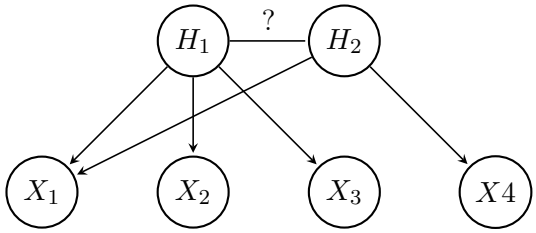


Figure 3: A bipartite graph Γ in Example 5.7

Since component 1 is assigned to $(H_1, H_2, H_3) = (0, 0, 0)$ *we can deduce that*

(H_1, H_2, H_3) :	$(0,0,0)$,	$(1,0,0)$,	$(0,1,0)$,	$(1,1,0)$,	$(0,0,1)$,	$(1,0,1)$,	$(0,1,1)$,	$(1,1,1)$
comp.# :	1	6	3	?	5	?	?	?

Assume that we know which components correspond to the hidden variable state $(H_1, H_2, H_3) = (h_1, h_2', h_3)$ *and* $(H_1, H_2, H_3) = (h_1', h_2, h_3)$*, with* $h_1 \neq h_1'$ *and* $h_2 \neq h_2'$*. Then we can use the information above to deduce which components correspond to the hidden state* (h_1', h_2', h_3) *since it differs from them in just 1 position. Hence, we can deduce*

(H_1, H_2, H_3) :	$(0,0,0)$,	$(1,0,0)$,	$(0,1,0)$,	$(1,1,0)$,	$(0,0,1)$,	$(1,0,1)$,	$(0,1,1)$,	$(1,1,1)$
comp.# :	1	6	3	7	5	8	2	?

Note that since $(1, 1, 1)$ *differs from the four states identified in the first step in two entries, this has not been determined yet. However, repeating this argument a third time we can deduce that component 4 corresponds to* $(H_1, H_2, H_3) = (1, 1, 1)$*.*

To illustrate how this algorithm works in the case of non-binary latent variables we provide one more example.

Example 5.7. *Assume that* $\mathbb{P}(V)$ *is Markov with respect to the DAG G in Figure 3 where we make no assumption about causal relation between* H_1 *and* H_2*. Assume that* $\dim(H_1) = \dim(H_2) = 3$*.*

Suppose that the map $L : [9] \to [9] \times [3] \times [3] \times [3]$ *is given by:*

i :	1	2	3	4	5
$L(i)$:	$(1,2,1,3)$,	$(3,3,3,1)$,	$(4,1,2,2)$,	$(2,2,1,1)$,	$(7,2,1,2)$,
i :	6	7	8	9	
$L(i)$:	$(5,1,2,1)$,	$(9,1,2,3)$,	$(8,3,3,3)$	$(6,3,3,2)$	

We want to find the correspondence between $h \in \Omega = \{0, 1, 2\}^2$ *and* $i \in [9]$*.*

As in the previous example, in order to see which components correspond to the states of latent variables where H_2 *is fixed and* H_1 *takes all values in* $\{0, 1, 2\}$ *we group together the components that have the same value of* L *on* $X \setminus \mathrm{ch}(H_1) = \{X_4\}$*. We get the following groups* $(1, 7, 8)$*,* $(2, 4, 6)$ *and* $(3, 5, 9)$*.*

Similarly, by comparing the values of L on $X \setminus \mathrm{ch}(H_2) = \{X_2, X_3\}$ we get that the following groups correspond to a fixed value of H_1, while H_2 vary: $(1, 4, 5)$, $(2, 8, 9)$ *and* $(3, 6, 7)$.

Since values of H_i are determined up to relabeling we can arbitrarily assign a component, say 1, to $(H_1 = 0, H_2 = 0)$. Now, using Lemma C.1, we know that components that correspond to $(H_1 = 1, H_2 = 0)$ and $(H_1 = 2, H_2 = 0)$ are 7 and 8, and again because values of H_i can be relabeled, at this point the choice is arbitrary. Using the similar argument for H_2, we can deduce the following correspondence:

(H_1, H_2) :	$(0,0)$,	$(1,0)$,	$(2,0)$	$(0,1)$,	$(1,1)$,	$(2,1)$,	$(0,2)$,	$(1,2)$,	$(2,2)$
comp.# :	1	7	8	4	?	?	5	?	?

At this point the labeling of the values of hidden variables is fixed. Now let us consider an index of hamming weight 2, say $(1, 1)$. *We know that the component, that corresponds to this state of latent variables, differs from the component* 4, *that corresponds to* $(0, 1)$, *only due to the change of H_1. Hence, the component that corresponds to* $(1, 1)$ *is in the set* $\{2, 4, 6\}$. *At the same time, we know that it differs from the component* 7 *that corresponds to* $(1, 0)$ *only due to the change of H_2. Hence, the desired component is in the set* $\{3, 6, 7\}$. *By taking the intersection of sets* $\{2, 4, 6\}$ *and* $\{3, 6, 7\}$ *we deduce that the value that corresponds to* $(1, 1)$ *is 6. Similarly we can determine the rest of the values.*

(H_1, H_2) :	$(0,0)$,	$(1,0)$,	$(2,0)$	$(0,1)$,	$(1,1)$,	$(2,1)$,	$(0,2)$,	$(1,2)$,	$(2,2)$
comp.# :	1	7	8	4	6	2	5	3	9

6 Implementation details

The results in Section 3 assume access to the mixture oracle $\mathsf{MixOracle}(S)$. Of course, in practice, learning mixture models is a nontrivial problem. Fortunately, many algorithms exist for approximating this oracle: In our implementation, we used K-means. A naïve application of clustering algorithms, however, ignores the significant structure *between* different subsets of observed variables. Thus, we also enforce internal consistency amongst these computations, which makes estimation much more robust in practice. In the remainder of this section, we describe the details of these computations; a complete outline of the entire pipeline can be found in Appendix E.

Estimating the number of marginal components In order to estimate the number of components in a marginal distribution for a subset S of observed variables with $|S| \leq 3$, we use K-means combined with agglomerative clustering to merge nearby cluster centers, and then select the number of components that has the highest silhouette score. Done independently, this step ignores the structure of the global mixture, and is not robust. In order to make learning more robust we observe that the assumptions on the distribution imply the following properties:

- *Divisibility condition:* The number of components we expect to observe over a set S of observed variables is divisible by a number of components we observe on the subset $S' \subset S$ of observed variables (see Obs. 2.7).
- *Structure of means:* Observe that the projections of the means of mixture clusters in the marginal distribution over S are the same as the means of mixture components over variables S' for every $S' \subseteq S$. Hence, if we learn the mixture models over S and S' with the correct numbers of components $k(S)$ and $k(S')$, we expect the projections to be close.

Example 6.1. *Suppose we are confident that the number of components in the mixture over X_1 is in the set $\{6, 7, 8\}$, over X_2 is in $\{4, 5, 6\}$ and the number of components in the mixture over $\{X_1, X_2\}$ is in the set $\{20, 21, 22, 23, 24, 25, 26\}$. Using divisibility condition between X_1 and $\{X_1, X_2\}$ we may shrink the set of candidates to $\{21, 24\}$. Next using the divisibility condition for X_2 and $\{X_1, X_2\}$ we may determine that the number of components should be 24.*

With these observations in mind, we use a weighted voting procedure, where every set S votes for the number of components in every superset and every subset based on divisibility or means alignment. We then predict the true number of components by picking the candidate with the most votes.

Constructing L In order to estimate L from samples we learn the mixture over the entire set of variables (using K-means and the number of components predicted on the previous step) and over each variable separately (again, using previous step). After this we project the mean of each component to a space over which X_i is defined and pick the closest mean in L_2 distance (see Figure 2).

Reconstructing the latent graphical model Once we obtain the joint probability table of H, the final piece is to learn the latent DAG Λ on H. This is a standard problem of learning the causal structure among m discrete variables given samples from their joint distribution. For this a multitude of approaches have been proposed in the literature, for instance the PC algorithm [SG91] or the GES algorithm [Chi02]. In our experiments, we use the Fast Greedy Equivalence Search [RGSRG17] with the discrete BIC score, without assuming faithfulness. The final graph G is therefore obtained from Γ and Λ.

7 Experiments

We implemented these algorithms in an end-to-end pipeline that inputs observed data and outputs an estimate of the causal graph G and an estimate for the joint probability table $\mathbb{P}(H)$. To test this pipeline, we ran experiments on synthetic data. Full details about these experiments, including a detailed description of the entire pipeline, can be found in Appendix F.

Data generation We start with a causal DAG G generated from the Erdös-Rényi model, for different settings of m, n and $|\Omega_i|$. We then generate samples from the probability distribution

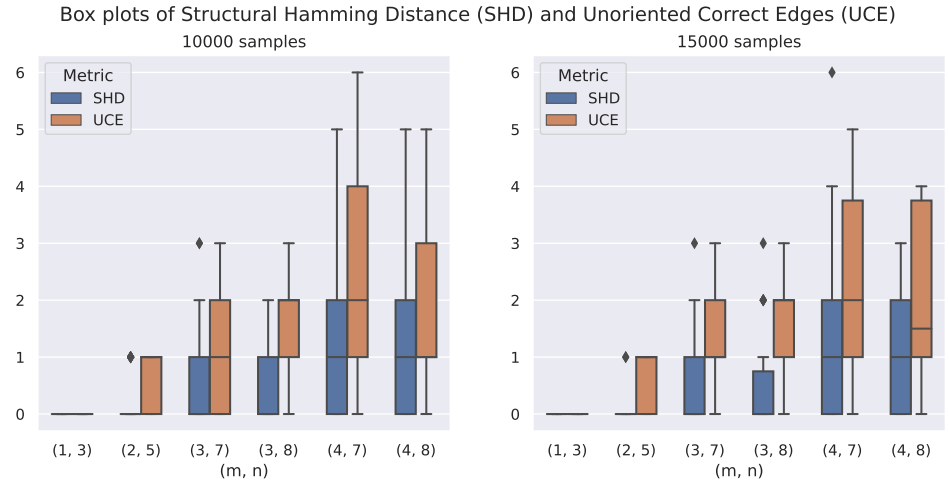


Figure 4: Average Structural Hamming distance for recovery of G, where $m = |H|$ and $n = |X|$.

that corresponds to G. We take each mixture component to be a Gaussian distribution with random mean and covariance (we do not force mixture components to be well-separated, aside from constraining the covariances to be small). Additionally, we do not impose restrictions on the weights of the components, which may be very small. As a result, it is common to have highly unbalanced clusters (e.g. we may have less than 30 points in one component and over 1000 in another). Figure 4 reports the results of 600 simulations; 300 each for $N = 10000$ samples and $N = 15000$ samples.

Results To compare how well our model recovers the underlying DAG, we compute the Structural Hamming Distance (SHD) between our estimated DAG and the true DAG. Since GES returns a CPDAG instead of a DAG, we also report the number of correct but unoriented edges in the estimated DAG. The average SHD across different problems sizes ranged from zero to 1.33. The highest SHD for any single run was 6. For context, the simulated DAGs had between 3 and 25 edges. Note that any errors are entirely due to estimation error in the K-means implementation of MixOracle, which we expect can be improved significantly. In the supplement we also report on experiments with much smaller sample size $N = 1000$ (Fig. 6). These results indicate that the proposed pipeline is surprisingly effective at recovering the causal graph.

8 Discussion

In this paper, we established general conditions under which the latent causal model G is identifiable (Theorem 3.2). We show that these conditions are essentially necessary, and mostly amount to non-degeneracy conditions on the joint distribution. Under a linear independence condition on columns of the bipartite adjacency matrix of Γ, we propose a polynomial time algorithm for recovering Γ and $\mathbb{P}(H)$. Our algorithms work by reduction to the mixture

oracle, which exists whenever the mixture model over X, naturally induced by discrete latent variables, is identifiable. Experimental results show effectiveness of our approach. Even though identifiability of mixture models is a long-studied problem, a good mixture oracle implementation is a bottleneck for scalability of our approach. We believe that it may be improved significantly, and consider this as a promising future direction. In this paper, we work under the measurement model that does not allow direct causal relationships between observed variables. We believe that this condition may be relaxed and are eager to explore this direction in future work.

9 Acknowledgements

G.R. thanks Aravindan Vijayaraghavan for pointers to useful references. B.K. was partially supported by advisor László Babai's NSF grant CCF 1718902. G.R. was partially supported by NSF grant CCF-1816372. P.R. was supported by NSF IIS-1955532. B.A. was supported by NSF IIS-1956330, NIH R01GM140467, and the Robert H. Topel Faculty Research Fund at the University of Chicago Booth School of Business.

References

[ADM+18] Nima Anari, Constantinos Daskalakis, Wolfgang Maass, Christos Papadimitriou, Amin Saberi, and Santosh Vempala. Smoothed analysis of discrete tensor decomposition and assemblies of neurons. *Advances in Neural Information Processing Systems*, 31:10857–10867, 2018.

[ADXR20] Bryon Aragam, Chen Dan, Eric P. Xing, and Pradeep Ravikumar. Identifiability of nonparametric mixture models and bayes optimal clustering. *Ann. Statist.*, 48(4):2277–2302, 2020. arXiv:1802.04397.

[AGH+13] Sanjeev Arora, Rong Ge, Yonatan Halpern, David Mimno, Ankur Moitra, David Sontag, Yichen Wu, and Michael Zhu. A practical algorithm for topic modeling with provable guarantees. In *International Conference on Machine Learning*, pages 280–288. PMLR, 2013.

[AGM12] Sanjeev Arora, Rong Ge, and Ankur Moitra. Learning topic models–going beyond svd. In *2012 IEEE 53rd annual symposium on foundations of computer science*, pages 1–10. IEEE, 2012.

[AHJK13] Animashree Anandkumar, Daniel Hsu, Adel Javanmard, and Sham Kakade. Learning linear Bayesian networks with latent variables. In *Proceedings of The 30th International Conference on Machine Learning*, pages 249–257, 2013.

[AHK12] A. Anandkumar, D. Hsu, and S.M. Kakade. A method of moments for mixture models and hidden markov models. *Journal of Machine Learning Research*, 23:33.1–33.34, 2012. cited By 26.

[AMR09] Elizabeth S Allman, Catherine Matias, and John A Rhodes. Identifiability of parameters in latent structure models with many observed variables. *Annals of Statistics*, pages 3099–3132, 2009.

[And84] Theodore Wilbur Anderson. Estimating linear statistical relationships. *The Annals of Statistics*, pages 1–45, 1984.

[AV+13] Animashree Anandkumar, Ragupathyraj Valluvan, et al. Learning loopy graphical models with latent variables: Efficient methods and guarantees. *Annals of Statistics*, 41(2):401–435, 2013.

[BB20] Guy Bresler and Rares-Darius Buhai. Learning restricted boltzmann machines with sparse latent variables. *Advances in Neural Information Processing Systems*, 33, 2020.

[BJR16] Stéphane Bonhomme, Koen Jochmans, and Jean-Marc Robin. Estimating multivariate latent-structure models. *Annals of Statistics*, 44(2):540–563, 2016.

[BKM19] Guy Bresler, Frederic Koehler, and Ankur Moitra. Learning restricted boltzmann machines via influence maximization. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 828–839, 2019.

[CEP17] Krzysztof Chalupka, Frederick Eberhardt, and Pietro Perona. Causal feature learning: an overview. *Behaviormetrika*, 44(1):137–164, 2017.

[Chi02] David Maxwell Chickering. Optimal structure identification with greedy search. *Journal of machine learning research*, 3(Nov):507–554, 2002.

[CK09] Jiahua Chen and Abbas Khalili. Order selection in finite mixture models with a nonsmooth penalty. *Journal of the American Statistical Association*, 104(485):187–196, 2009.

[CMKR12] Diego Colombo, Marloes H Maathuis, Markus Kalisch, and Thomas S Richardson. Learning high-dimensional directed acyclic graphs with latent and selection variables. *Annals of Statistics*, 40(1):294–321, 2012.

[CPE14] Krzysztof Chalupka, Pietro Perona, and Frederick Eberhardt. Visual causal feature learning. *arXiv preprint arXiv:1412.2309*, 2014.

[CPW+12] Venkat Chandrasekaran, Pablo A Parrilo, Alan S Willsky, et al. Latent variable graphical model selection via convex optimization. *The Annals of Statistics*, 40(4):1935–1967, 2012.

[D'A19] Alexander D'Amour. On multi-cause causal inference with unobserved confounding: Counterexamples, impossibility, and alternatives. *arXiv preprint arXiv:1902.10286*, 2019.

[DCG+97] Didier Dacunha-Castelle, Elisabeth Gassiat, et al. The estimation of the order of a mixture model. *Bernoulli*, 3(3):279–299, 1997.

[DDF21] Mucong Ding, Constantinos Daskalakis, and Soheil Feizi. Gans with conditional independence graphs: On subadditivity of probability divergences. In *International Conference on Artificial Intelligence and Statistics*, pages 3709–3717. PMLR, 2021.

[E+18] Robin J Evans et al. Margins of discrete bayesian networks. *The Annals of Statistics*, 46(6A):2623–2656, 2018.

[ELFK00] Gal Elidan, Noam Lotner, Nir Friedman, and Daphne Koller. Discovering hidden variables: a structure-based approach. In *Proceedings of the 13th International Conference on Neural Information Processing Systems*, pages 458–464, 2000.

[ER14] Robin J Evans and Thomas S Richardson. Markovian acyclic directed mixed graphs for discrete data. *The Annals of Statistics*, pages 1452–1482, 2014.

[ER+19] Robin J Evans, Thomas S Richardson, et al. Smooth, identifiable supermodels of discrete dag models with latent variables. *Bernoulli*, 25(2):848–876, 2019.

[Eva16] Robin J Evans. Graphs for margins of bayesian networks. *Scandinavian Journal of Statistics*, 43(3):625–648, 2016.

[F+97] Nir Friedman et al. Learning belief networks in the presence of missing values and hidden variables. In *ICML*, volume 97, pages 125–133. Citeseer, 1997.

[FNM17] Benjamin Frot, Preetam Nandy, and Marloes H Maathuis. Robust causal structure learning with some hidden variables. *arXiv preprint arXiv:1708.01151*, 2017.

[GCR13] Elisabeth Gassiat, Alice Cleynen, and Stéphane Robin. Finite state space non parametric hidden markov models are in general identifiable. *arXiv preprint arXiv:1306.4657*, 2013.

[GDA20] Ming Gao, Yi Ding, and Bryon Aragam. A polynomial-time algorithm for learning nonparametric causal graphs. *Advances in Neural Information Processing Systems*, 33, 2020.

[GKS20] Justin Grimmer, Dean Knox, and Brandon M Stewart. Na\" ive regression requires weaker assumptions than factor models to adjust for multiple cause confounding. *arXiv preprint arXiv:2007.12702*, 2020.

[GS21] Spencer L Gordon and Leonard J Schulman. Hadamard extensions and the identification of mixtures of product distributions. *arXiv preprint arXiv:2101.11688*, 2021.

[Har70] Richard A. Harshman. Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multimodal factor analysis. 1970.

[HSK06] Patrik O Hoyer, Shohei Shimizu, and Antti J Kerminen. Estimation of linear, non-gaussian causal models in the presence of confounding latent variables. *arXiv preprint cs/0603038*, 2006.

[HZ03] Peter Hall and Xiao-Hua Zhou. Nonparametric estimation of component distributions in a multivariate mixture. *Annals of Statistics*, pages 201–224, 2003.

[KFW+18] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. In *International Conference on Machine Learning*, pages 2688–2697. PMLR, 2018.

[KKMH20] Ilyes Khemakhem, Diederik Kingma, Ricardo Monti, and Aapo Hyvarinen. Variational autoencoders and nonlinear ica: A unifying framework. In *International Conference on Artificial Intelligence and Statistics*, pages 2207–2217. PMLR, 2020.

[Kol00] Vladimir I Koltchinskii. Empirical geometry of multivariate data: a deconvolution approach. *Annals of statistics*, pages 591–629, 2000.

[KSDV17] Murat Kocaoglu, Christopher Snyder, Alexandros G Dimakis, and Sriram Vishwanath. Causalgan: Learning causal implicit generative models with adversarial training. *arXiv preprint arXiv:1709.02023*, 2017.

[Lin95] Bruce G Lindsay. Mixture models: theory, geometry and applications. In *NSF-CBMS regional conference series in probability and statistics*, pages i–163. JSTOR, 1995.

[LM11] Pedro Larrañaga and Serafín Moral. Probabilistic graphical models in artificial intelligence. *Applied soft computing*, 11(2):1511–1528, 2011.

[LP21] Benjamin Lovitz and Fedor Petrov. A generalization of kruskal's theorem on tensor decomposition. *arXiv preprint arXiv:2103.15633*, 2021.

[LPNC+17] David Lopez-Paz, Robert Nishihara, Soumith Chintala, Bernhard Scholkopf, and Léon Bottou. Discovering causal signals in images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6979–6987, 2017.

[LTA+20] Yunzhu Li, Antonio Torralba, Animashree Anandkumar, Dieter Fox, and Animesh Garg. Causal discovery in physical systems from videos. *arXiv preprint arXiv:2007.00631*, 2020.

[MGW20] Alex Markham and Moritz Grosse-Wentrup. Measurement dependence inducing latent causal models. In *Conference on Uncertainty in Artificial Intelligence*, pages 590–599. PMLR, 2020.

[MK20] Tudor Manole and Abbas Khalili. Estimating the number of components in finite mixture models via the group-sort-fuse procedure. *arXiv preprint arXiv:2005.11641*, 2020.

[MLR19] Geoffrey J McLachlan, Sharon X Lee, and Suren I Rathnayake. Finite mixture models. *Annual review of statistics and its application*, 6:355–378, 2019.

[Moi14] Ankur Moitra. Algorithmic aspects of machine learning. *Lecture notes*, 2014.

[MR05] Elchanan Mossel and Sébastien Roch. Learning nonsingular phylogenies and hidden markov models. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 366–375, 2005.

[MV95] J Martin and Kurt VanLehn. Discrete factor analysis: Learning hidden variables in bayesian networks. Technical report, 1995.

[MZH20] Ricardo Pio Monti, Kun Zhang, and Aapo Hyvärinen. Causal discovery with general non-linear relationships using non-linear ica. In *Uncertainty in Artificial Intelligence*, pages 186–195. PMLR, 2020.

[ND17] Alejandro Newell and Jia Deng. Pixels to graphs by associative embedding. *arXiv preprint arXiv:1706.07365*, 2017.

[PB13] Jonas Peters and Peter Bühlmann. Identifiability of Gaussian structural equation models with equal error variances. *Biometrika*, 101(1):219–228, 2013.

[Pea88] Judea Pearl. *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. Morgan Kaufmann, 1988.

[Pea09] Judea Pearl. *Causality*. Cambridge university press, 2009.

[PMJS14] Jonas Peters, Joris M Mooij, Dominik Janzing, and Bernhard Schölkopf. Causal discovery with continuous additive noise models. *Journal of Machine Learning Research*, 15(1):2009–2053, 2014.

[RERS17] Thomas S Richardson, Robin J Evans, James M Robins, and Ilya Shpitser. Nested markov properties for acyclic directed mixed graphs. *arXiv preprint arXiv:1701.06686*, 2017.

[RGSRG17] Joseph Ramsey, Madelyn Glymour, Ruben Sanchez-Romero, and Clark Glymour. A million variables and more: the fast greedy equivalence search algorithm for learning high-dimensional graphical causal models, with an application to functional magnetic resonance images. *International journal of data science and analytics*, 3(2):121–129, 2017.

[RS+02] Thomas Richardson, Peter Spirtes, et al. Ancestral graph markov models. *The Annals of Statistics*, 30(4):962–1030, 2002.

[RSSW03] James M Robins, Richard Scheines, Peter Spirtes, and Larry Wasserman. Uniform consistency in causal inference. *Biometrika*, 90(3):491–515, 2003.

[RVS20] Alexander Ritchie, Robert A Vandermeulen, and Clayton Scott. Consistent estimation of identifiable nonparametric mixture models from grouped observations. *arXiv preprint arXiv:2006.07459*, 2020.

[RW99] James M Robins and Larry Wasserman. On the impossibility of inferring causation from association without background knowledge. *Computation, causation, and discovery*, 1999:305–21, 1999.

[SBY09] Tao Shi, Mikhail Belkin, and Bin Yu. Data spectroscopy: Eigenspaces of convolution operators and clustering. *Annals of Statistics*, pages 3960–3984, 2009.

[SG91] Peter Spirtes and Clark Glymour. An algorithm for fast recovery of sparse causal graphs. *Social Science Computer Review*, 9(1):62–72, 1991.

[SGS00] Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, prediction, and search*, volume 81. The MIT Press, 2000.

[SHHK06] Shohei Shimizu, Patrik O Hoyer, Aapo Hyvärinen, and Antti Kerminen. A linear non-Gaussian acyclic model for causal discovery. *Journal of Machine Learning Research*, 7:2003–2030, 2006.

[SLB+21] Bernhard Schölkopf, Francesco Locatello, Stefan Bauer, Nan Rosemary Ke, Nal Kalchbrenner, Anirudh Goyal, and Yoshua Bengio. Toward causal representation learning. *Proceedings of the IEEE*, 109(5):612–634, 2021.

[SLD+20] Xinwei Shen, Furui Liu, Hanze Dong, Qing Lian, Zhitang Chen, and Tong Zhang. Disentangled generative causal representation learning. *arXiv preprint arXiv:2010.02637*, 2020.

[SMR13] Peter L Spirtes, Christopher Meek, and Thomas S Richardson. Causal inference in the presence of latent variables and selection bias. *arXiv preprint arXiv:1302.4983*, 2013.

[SS03] Charles Semple and Mike Steel. *Phylogenetics*, volume 24. Oxford University Press on Demand, 2003.

[SSGS06] Ricardo Silva, Richard Scheine, Clark Glymour, and Peter Spirtes. Learning the structure of linear latent variable models. *Journal of Machine Learning Research*, 7(Feb):191–246, 2006.

[SZ14] Peter Spirtes and Jiji Zhang. A uniformly consistent estimator of causal effects under the k-triangle-faithfulness assumption. *Statistical Science*, pages 662–678, 2014.

[Tei63] Henry Teicher. Identifiability of finite mixtures. *The annals of Mathematical statistics*, pages 1265–1269, 1963.

[Tei67] Henry Teicher. Identifiability of mixtures of product measures. *The Annals of Mathematical Statistics*, 38(4):1300–1302, 1967.

[Vij20] Aravindan Vijayaraghavan. Efficient tensor decomposition. *Beyond the Worst-Case Analysis of Algorithms*, page 424, 2020.

[VS16] Robert A. Vandermeulen and Clayton D. Scott. An operator theoretic approach to nonparametric mixture models, 2016.

[WHE+19] Chirayu (Kong) Wongchokprasitti, Harry Hochheiser, Jeremy Espino, Eamonn Maguire, Bryan Andrews, Michael Davis, and Chris Inskip. bd2kccd/py-causal v1.2.1, December 2019.

[XCH+20] Feng Xie, Ruichu Cai, Biwei Huang, Clark Glymour, Zhifeng Hao, and Kun Zhang. Generalized independent noise condition for estimating latent variable causal graphs. *Advances in Neural Information Processing Systems*, 33, 2020.

[YLC+20] Mengyue Yang, Furui Liu, Zhitang Chen, Xinwei Shen, Jianye Hao, and Jun Wang. Causalvae: Disentangled representation learning via neural structural causal models. *arXiv preprint arXiv:2004.08697*, 2020.

[YS68] Sidney J Yakowitz and John D Spragins. On the identifiability of finite mixtures. *The Annals of Mathematical Statistics*, 39(1):209–214, 1968.

[ZH09] Kun Zhang and Aapo Hyvärinen. On the identifiability of the post-nonlinear causal model. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 647–655. AUAI Press, 2009.

A Non-identifiability if Assumption 2.4 is violated

In this appendix we are going to show that Assumptions 2.2 and 2.3 on the graph G are not sufficient for identifiability, and therefore additional assumptions on the distribution of H over Ω are required as well.

Definition A.1. *For distributions D_1, D_2, let $D_1 \otimes D_2$ denote the product of the distributions D_1 and D_2.*

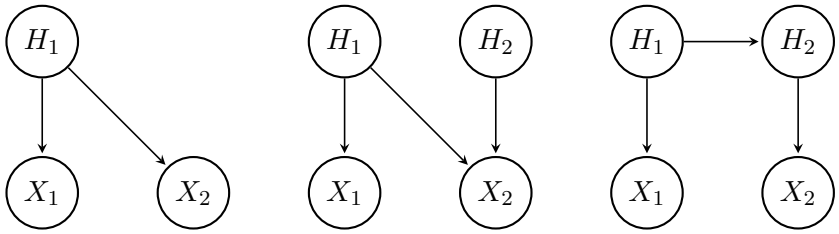
That is, if $X \sim D_1$ and $Y \sim D_2$ are independent, then their joint distribution is $D_1 \otimes D_2$.

The following example illustrates an important case of non-identifiability and motivates the need for Assumption 2.4.

Example A.2. *Let N_0, N_1, N_0', N_1' be independent Gaussian distributions with distinct parameters (means and variances). Consider*

$$(X_1, X_2) \sim \frac{1}{2} N_0 \otimes N_0' + \frac{1}{4} N_1 \otimes N_0' + \frac{1}{4} N_1 \otimes N_1' \tag{9}$$

We claim that (X_1, X_2) is consistent with (i.e., satisfies Markov property with respect to) each of the following three models below. Here, in the model S_3 the hidden variable H_1 can take three values $\{0, 1, 2\}$, and in models A and B, hidden variables take values in $\{0, 1\}$.



Note that all these models satisfy "no-twins" Assumption 2.2 and minimality Assumption 2.3, while Assumptions 2.4 are violated by models A and B.

1. *Consistency with S_3. Let H_1 be a random variable that takes values $0, 1, 2$ with probabilities $(1/2, 1/4, 1/4)$. Then*

$$(X_1, X_2) \sim \sum_{j \in \{0,1,2\}} \mathbb{P}(X | H_1 = j)\, \mathbb{P}(H_1 = j), \text{ where}$$
$$\mathbb{P}(X | H_1 = 0) = N_0 \otimes N_0', \quad \mathbb{P}(X | H_1 = 1) = N_1 \otimes N_0', \quad \mathbb{P}(X | H_1 = 2) = N_1 \otimes N_1'$$

2. *Consistency with A. Let H_1 and H_2 be i.i.d random variables that take values $0, 1$ with*

probabilities $(1/2, 1/2)$. *Then*

$$(X_1, X_2) \sim \sum_{i\in\{0,1\}} \sum_{j\in\{0,1\}} \mathbb{P}(X|H_1 = i, H_2 = j)\,\mathbb{P}(H_1 = i)\,\mathbb{P}(H_2 = j), \text{ where}$$
$$\mathbb{P}(X|H_1 = 0) = N_0 \otimes N_0', \quad \mathbb{P}(X|H_1 = 1, H_2 = 0) = N_1 \otimes N_0',$$
$$\mathbb{P}(X|H_1 = 1, H_2 = 1) = N_1 \otimes N_1'$$

3. *Consistency with* B. *Let* H_1 *be a random variable that takes values* $0, 1$ *with probabilities* $(1/2, 1/2)$. *Let* H_2 *be a dependent random variable that takes values* $0, 1$ *with probabilities* $(1, 0)$, *if* $H_1 = 0$, *and with probabilities* $(1/2, 1/2)$, *if* $H_1 = 1$.

$$(X_1, X_2) \sim \sum_{i\in\{0,1\}} \sum_{j\in\{0,1\}} \mathbb{P}(X_1|H_1 = i)\,\mathbb{P}(X_2|H_2 = j)\,\mathbb{P}(H_1 = i)\,\mathbb{P}(H_2 = j|H_1 = i),$$

where

$$\mathbb{P}(X_1|H_1 = 0) = N_0, \quad \mathbb{P}(X_1|H_1 = 1) = N_1,$$
$$\mathbb{P}(X_2|H_2 = 0) = N_0', \quad \mathbb{P}(X_2|H_2 = 1) = N_1'$$

Remark A.3. *Observe that among the models* A, B *and* S_3, *only* S_3 *satisfies Assumption 2.4. Observe that the model* A *satisfies part (a), but not (b), and the model* B *satisfies part (b), but not (a), of Assumption 2.4. This shows that only one of these assumptions is still not sufficient for identifiability of a latent causal model.*

B Reconstructing bipartite part Γ. Proofs for Sections 4

Recall that (cf. Section 4.2), that for $w(H_i) = \log(\dim(H_i))$ and every subset $S \subseteq X$ the parameters of the latent DAG satisfy

$$\log(k(S)) = \sum_{H_i\in \mathrm{pa}(S)} w(H_i). \tag{10}$$

Recall also the definitions of sne and Wsne in (5), reproduced here for ease of reference:

$$\mathrm{sne}_\Gamma(S) = \bigcap_{x\in S} \mathrm{ne}_\Gamma(x) \quad \text{and} \quad \mathrm{Wsne}_\Gamma(S) = \sum_{v\in \mathrm{sne}_\Gamma(S)} w(v).$$

B.1 Learning a bipartite graph with a hidden part from an additive score

We start our discussion of the proof of results in Section 4 by reducing learning of the causal graph Γ to a more general learning problem.

Let $\Gamma = (X \cup H, E)$ be a (not necessarily directed) bipartite graph on parts X and H, and let $w : H \to (0, \infty)$ be an arbitrary function that defines weights of variables in H.

Recall that for a weight function w and subset $S \subseteq X$ we define

$$W_\Gamma(S) = \sum_{v\in \mathrm{ne}_\Gamma(S)} w(v) \tag{11}$$

Problem B.1. *Assume that the vertices in H and the weight function w are unknown.*

Input: *Values $(W_\Gamma(S) \mid S \in \mathcal{F})$ indexed by a family of known subsets $\mathcal{F} \subseteq 2^X$*

Goal: *Reconstruct the number of unknown vertices H, the graph Γ between H and X (up to an isomorphism), and the weight function w from the input.*

Whether it is possible to reconstruct Γ and w from the input may depend on the family $\mathcal{F}$ or some additional assumptions about the structure of the graph Γ. To account for weights w, we slightly modify Definition 4.1 as follows:

Definition B.2. *We say that (Γ, w) is $\mathcal{F}$-recoverable if (Γ, w) can be uniquely recovered from X and the sequence $(W_\Gamma(S) \mid S \in \mathcal{F})$.*

In the sequel, we use this modified definition.

The most natural regime is when $\mathcal{F}$ contains the sets whose size is bounded:

Definition B.3. *We say that (Γ, w) is t-recoverable if (Γ, w) is $\binom{X}{\leq t}$-recoverable, where $\binom{X}{\leq t}$ denotes the collection of subsets of X of size at most t.*

B.2 Reconstructing Γ with full information about W

In this section we study Problem B.1, when full information about $W_\Gamma(\cdot)$ is provided, i.e. $\mathcal{F} = 2^X$.

Although the algorithm considered here will have exponential in $|X|$ runtime, it sheds light on the minimal theoretical assumptions we need for proving identifiability of Γ. We will consider more efficient algorithms in later sections.

We start by proving Observation 4.3, which notes that if $\mathrm{ne}_\Gamma(H_i) = \mathrm{ne}_\Gamma(H_j)$ for $H_i \neq H_j$, then (Γ, w) is not 2^X-recoverable.

Proof of Observation 4.3. Consider the graph Γ' obtained from Γ by replacing H_1 and H_2 with a single variable H^* and by connecting H^* by an edge to all vertices in X that are adjacent with H_1 or H_2 in Γ. Define $w(H^*) = w(H_1) + w(H_2)$. Then $W_\Gamma(S) = W_{\Gamma'}(S)$ for any $S \subseteq X$. □

Corollary B.4. *Let $\mathcal{F} \subseteq 2^X$. If there is a pair of distinct variables $H_i, H_j \in H$ such that $\mathrm{ne}_\Gamma(H_1) = \mathrm{ne}_\Gamma(H_2)$, then (Γ, w) is not $\mathcal{F}$-recoverable.*

We now prove that in the case $\mathcal{F} = 2^X$, this is the only obstacle. We start by showing that certain neighborhoods of hidden variables can be identified using $\mathrm{Wsne}(\cdot)$.

As explained in Section 4.2, in the case when $\mathrm{ne}(H_i) \not\subset \mathrm{ne}(H_j)$ for all H_j, we expect $\mathrm{Wsne}(\cdot)$ to have a clear "signature" of H_i. We make this intuition precise in the definition and lemma that follows.

Definition B.5. *We say that a set S of observed variables X is a* maximal neighborhood block *if $\mathrm{Wsne}(S) \neq 0$, but for any superset S' of S we have $\mathrm{Wsne}(S') = 0$.*

Lemma B.6. *A set $S \subseteq X$ is a maximal neighborhood block if and only if there exists a hidden vertex $H_i \in H$ such that $\mathrm{ne}_\Gamma(H_i) = S$ and for any other $H_j \in H$ we have $S \not\subseteq \mathrm{ne}_\Gamma(H_j)$.*

Proof. Assume that $S \subseteq X$ is a maximal neighborhood block. Since $\mathrm{Wsne}(S) > 0$ the set of common neighbours $\mathrm{sne}_\Gamma(S)$ is non-empty. If $\mathrm{sne}_\Gamma(S)$ contains a hidden vertex H_j that is connected to a vertex $x \notin S$ then, $H_j \in \mathrm{sne}_\Gamma(S \cup \{x\})$, and $\mathrm{Wsne}_\Gamma(S \cup \{x\}) \geq w(H_j) > 0$ which contradicts the assumption that S is a maximal neighborhood block. Therefore, for every H_j in $\mathrm{sne}_\Gamma(S)$, we have $\mathrm{ne}_\Gamma(H_j) \subset S$. Therefore, there exists a variable H_i such that $\mathrm{ne}_\Gamma(H_i) = S$ and for any other $H_j \in H$ we have $S \not\subseteq \mathrm{ne}_\Gamma(H_j)$.

The opposite implication can be verified in a similar way. □

Theorem B.7 (Theorem 4.2, part (a))**.** *Let Γ be a bipartite graph with parts X and H. Assume that no pair of vertices in H has the same set of neighbours (in X). Then Γ is 2^X-recoverable.*

Proof. We prove the claim of the theorem by induction on $|H|$. The statement for the base case $|H| = 0$ immediately follows from the fact that $W(S) = 0$ for all $v \in X$ if and only if $|H| = 0$ since $w(\cdot) > 0$. Assume that we proved the claim for all Γ with $|H| = t$ that satisfy the assumptions of the theorem. Let Γ be a graph with $|H| = t + 1$ that satisfies the assumptions of the theorem.

Using Lemma 4.6, compute values $\mathrm{Wsne}_\Gamma(S)$ for every $S \subseteq X$. Using values of $\mathrm{Wsne}(\cdot)$ we can find a maximal neighborhood block $Y \subseteq X$. By Lemma B.6, there exists a hidden vertex H_i such that $\{H_i\} = \mathrm{sne}_\Gamma(Y)$. Note that $w(H_i) = \mathrm{Wsne}(Y)$.

Denote by Γ' the graph obtained from Γ by deleting H_i.

Now we verify that Γ' satisfies the assumptions of the theorem. There is nothing to check if the set of hidden vertices of Γ' is empty. Assume that Γ' has a non-empty set of hidden vertices. First, note that all hidden vertices in Γ' still have distinct sets of neighbors. Second, note that (cf. (11)) $W_{\Gamma'}(S) = W_\Gamma(S)$ if $S \cap Y = \emptyset$ (i.e. $H_i \notin \mathrm{ne}_\Gamma(S)$), and

$$W_{\Gamma'}(S) = W_\Gamma(S) - w(H_i) = W_\Gamma(S) - \mathrm{Wsne}_\Gamma(Y)$$

if $S \cap Y$ is not empty. Thus, we can compute $W_{\Gamma'}$ from the values of W_Γ.

By the induction hypothesis $(\Gamma', w|_{\Gamma'})$ is uniquely recoverable from $W_{\Gamma'}(S)$. Let Γ^* be the graph obtained from Γ' by adding a new variable H_Y of weight $\mathrm{Wsne}_\Gamma(Y)$ and edges between H_Y and Y. Then Γ^* is isomorphic to Γ, and so Γ is 2^X-recoverable. □

B.3 Efficient t-recovery of Γ for $t \geq 3$

The approach proposed in Appendix B.2 is exponential in the number of observed variables in the worst case, since we need to compute the scores of all subsets of X. In this section, we show that with a mild additional assumption, there is an efficient algorithm to learn the bipartite graph between hidden and observed variables.

As before, let $\Gamma = (X \cup H, E)$ be the bipartite graph between hidden and observed variables.

Recall, that we defined A to be the $|X| \times |H|$ adjacency matrix of Γ (with $0, 1$ entries) and a_i to denote the i-th column of A.

For a sequence of indices $I = (i_1, i_2, \ldots, i_t) \subseteq [n]$ define

$$\mathrm{Wsne}_\Gamma(I) = \sum_{j \in H} w(j) \underbrace{(a_j)_{i_1}(a_j)_{i_2} \ldots (a_j)_{i_t}}_{t} = \Big(\sum_{j \in H} w(j) \underbrace{a_j \otimes a_j \otimes \ldots \otimes a_j}_{t} \Big)_{(I)}. \tag{12}$$

Recall, that as pointed out in Remark 4.7, for any $S \subseteq X$ with $|S| \leq t$ the value $\mathrm{Wsne}_\Gamma(S)$ can be computed from the $\{W_\Gamma(S) \mid S \subseteq X,\ |S| \leq t\}$ using Lemma 4.6. Therefore, we can make the following observation.

Observation B.8. *All entries of the the tensor* $M_t = \sum_{j \in H} w(j) (\underbrace{a_j \otimes a_j \otimes \ldots \otimes a_j}_{t})$ *can be computed as* $M_t(I) = \mathrm{Wsne}_\Gamma(I)$ *in* $O(2^t n^t)$ *time and space assuming access to* $\{W_\Gamma(S) \mid S \subseteq X,\ |S| \leq t\}$.

For fixed t this is a poly-time computation. Furthermore, in the settings we consider in Secrion 4 the values of W_Γ can be computed from MixOracle using Observation 2.7.

Now we want to recover the vectors a_j from M_t. Since a_j are the columns of the adjacency matrix of Γ this is equivalent to recovering the adjacency matrix of Γ or Γ itself up to an isomorphism.

Definition B.9. *For an order-t tensor M_t its rank is defined as the smallest r such that M_t can be written as*

$$M_t = \sum_{j=1}^{r} c_j \bigotimes_{i=1}^{t} x_j^{(i)}. \tag{13}$$

Such decomposition of M with precisely r components is called a minimum rank decomposition or a CP-decomposition.

Lemma B.10. *If the decomposition*

$$M_t = \sum_{j \in H} w(j) \underbrace{a_j \otimes a_j \otimes \ldots \otimes a_j}_{t}$$

is the unique minimum rank decomposition, then (Γ, w) is t-recoverable.

Proof. In order to recover Γ and w we compute M_t using $\{W_\Gamma(S) \mid S \subseteq X,\ |S| \leq t\}$. Then a_j and $w(j)$ can be uniquely (up to permutation) identified from minimum rank decomposition of M. □

The following simplified version of Kruskal's condition was proposed by Lovitz and Petrov.

Theorem B.11 ([LP21, Theorem 2]). *Let $m \geq 2$ and $t \geq 3$ be integers. Let $V = V_1 \otimes V_2 \otimes \ldots \otimes V_t$ be a multipartite vector space over a field $\mathbb{F}$ and let*

$$\{x_j^{(1)} \otimes x_j^{(2)} \otimes \ldots \otimes x_j^{(t)} \mid j \in [m]\} \subset V\}$$

be a set of m rank-1 (product) tensors. For a subset $S \subseteq [m]$ with $|S| \geq 2$ and $j \in [t]$ define

$$d_i(S) = \dim \operatorname{span}\{x_j^{(i)} \mid j \in S\}.$$

If $2|S| \leq \sum_{i=1}^{t} (d_i(S) - 1) + 1$ for every such S, then

$$\sum_{j \in [m]} x_j^{(1)} \otimes x_j^{(2)} \otimes \ldots \otimes x_j^{(t)}$$

constitutes a unique minimal rank decomposition.

In our settings the sufficient condition for having the unique minimal rank decomposition takes the following form.

Corollary B.12. *Assume that for every $S \subseteq H$ with $|S| \geq 2$ we have*

$$\dim \operatorname{span}\{a_j \mid j \in S\} \geq \frac{2}{t}|S| + 1,$$

then the decomposition $M_t = \sum_{j \in H} w(j) \underbrace{a_j \otimes a_j \otimes \ldots \otimes a_j}_{t}$ is the unique minimum rank decomposition and so (Γ, w) is t-recoverable.

Proof. Take $\mathbb{F} = \mathbb{R}$, then the result follows from B.11 for $x_j^{(1)} = w(j)a_j$ and $x_j^{(i)} = a_j$. □

Proof of Theorem 4.2 part (b). Follows by combining Corollary B.12 and Lemma B.10. □

Learning the components of the minimum rank decomposition is a very well-studied problem for which a variety of algorithms have been proposed in the literature (see the survey [Vij20] or the book [Moi14]). We can use Jennrich's algorithm [Har70] (see also [Vij20, Moi14] and the references therein) as an efficient algorithm with guarantees:

Theorem B.13 (Jennrich's algorithm [Har70]). *Assume that the components of the tensor $\mathcal{T} = \sum_{i=1}^{r} a_i \otimes b_i \otimes c_i$ satisfy the following conditions. The vectors $\{a_i \mid i \in [r]\}$ are linearly independent, the vectors $\{b_i \mid i \in [r]\}$ are linearly independent, and no pair of vectors c_i, c_j is linearly dependent for $i \neq j$. Then the components of the tensor can be uniquely recovered in $O(n^3)$ space and $O(n^4)$ time.*

Remark B.14. *Note that if all vectors a_i are linearly independent, then the assumptions of Corollary B.12 are satisfied.*

Remark B.15. *A similar problem for t-recovery (for weighted hypergraphs) arose in a completely different context [ADM+18]. While in both papers the problem is reduced to recovering the minimum rank decomposition of a carefully constructed tensor, we give better recovery guarantees for this problem by using more recent uniqueness guarantees [LP21].*

C Reconstruction of the probability distribution on H. Proofs for Section 5

In this section we discuss how one may reconstruct the hidden probability distribution on $\mathbb{P}(H)$ from

- the bipartite graph Γ, and
- the function $L : [K] \to [k_1] \times \cdots \times [k_n]$, and
- the mixture weights (probabilities) $\{\pi(X, i) \mid i \in [k(X)]\} = \{\mathbb{P}(Z = i) \mid i \in [k(X)]\}$

C.1 A key lemma

Below we formulate the key lemma that allows us to relate the structure present in the map L with the causal structure in G.

Given a state $H = (h_1, \ldots, h_m)$ and its corresponding component $P(X \mid H_1 = h_1, \ldots, H_m = h_m)$, we want to identify the components $P(X \mid H_1 = h_1', H_2 = h_2, \ldots, H_m = h_m)$ that result from changing just the first hidden variable while keeping every other hidden variable fixed. The next lemma says that we can identify such components by looking into the distribution of the observed variables that are not children of H_1.

Lemma C.1. *Let H_i be a hidden variable and let $C(X \setminus \mathrm{ne}_\Gamma(H_i), j)$ be an arbitrary mixture component observed in a marginal mixture distribution over the variables in $X \setminus \mathrm{ne}_\Gamma(H_i)$. Let $C(j_1), C(j_2), \ldots C(j_t)$ be all the mixture components in the distribution of X whose marginal distribution over $X \setminus \mathrm{ne}_\Gamma(H_i)$ is equal to $C(X \setminus \mathrm{ne}_\Gamma(H_i), j)$. In other words, $L(j_s)_i = j$ for all $s \in [t]$. Then $t = \dim(H_i)$ and every $C(j_s)$ for $s \in [t]$ corresponds to a distinct value of H_i.*

Proof. Observe that Assumption 3.1 implies that $\mathrm{ne}_\Gamma(X \setminus \mathrm{ne}_\Gamma(H_i)) = H \setminus \{H_i\}$. Therefore, by Assumption 2.4(b), $p(X \setminus \mathrm{ne}_\Gamma(H_i) \mid H = h_1) \sim p(X \setminus \mathrm{ne}_\Gamma(H_i) \mid H = h_2)$, if and only if h_1 and h_2 differ only in the value of H_i. □

C.2 Proof of Theorem 5.4

The algorithm described in the previous examples can be used to prove Theorem 5.4. For this, we present a general algorithm to recover the correspondence $\Omega \ni h \leftrightarrow i \in [K]$ using Lemma C.1.

Proof of Theorem 5.4. Without loss of generality, we may assume that H_i takes values from $\Omega_i = \{0, 1, \ldots, \dim(H_i) - 1\}$ for every i.

Recall that the Hamming weight of a vector is the number of non-zero coordinates of this vector. Denote by $\Omega^{(t)}$ the set of elements of $\Omega = \Omega_1 \times \Omega_2 \times \ldots \times \Omega_k$ of the Hamming weight at most t.

We start by recovering the entries of the tensor that correspond to the indicies in $\Omega^{(1)}$.

Let us pick an arbitrary mixture component C that participates in the observed mixture model and let us put it in correspondence to $h = (0, 0, \ldots 0)$. We assign the probability of observing C to the cell $J(0, 0, \ldots, 0)$.

Take any $i \in [m]$. Consider the set of $d(H_i)$ mixture components $\{C_{i,a} \mid a \in \Omega_i\}$, guaranteed by Lemma C.1, that have the same distribution as C in coordinates $X \setminus \mathrm{ch}(H_i)$ (here we take arbitrary indexing by a). Assign $C_{i,a}$ to the vector $h_{i,a} \in \Omega^{(1)}$ of Hamming weight 1, that has unique non-zero value a in coordinate i. And let $J(h_{i,a})$ be the probability of observing $C_{i,a}$.

Next, we claim that the (valid) correspondence $\Omega \ni h \leftrightarrow i \in [K]$ for $h \in \Omega^{(t)}$ can be uniquely extended to the (valid) correspondence $\Omega \ni h \leftrightarrow i \in [K]$ for $h \in \Omega^{(t+1)}$ for any $t = 1, \ldots, m-1$.

Indeed, let $h \in \Omega^{(t+1)}$ and let i and j be a pair of distinct non-zero coordinates of h. Let h_i and h_j be the vectors obtained by changing the i-th and j-th coordinates of h to 0. Let C_i and C_j be the mixture components that correspond to h_i and h_j.

Using Lemma C.1, for $s \in \{i, j\}$ we can find a set M_u of $\dim(H_u)$ mixture components that are equally distributed with C_s over $X \setminus \mathrm{ne}_\Gamma(H_s)$. We put into correspondence with h the unique component in the intersection of M_i and M_j. We define $J(h)$ to be the probability of observing this component. □

Next we show that our algorithm works in time that is almost linear in the output size (recall that $K \geq 2^m$ and K is the size of the output).

Observation C.2. *The algorithm described in Theorem 5.4 works in $O((nm + \max_i k_i)K)$ time.*

Proof. First, the algorithm in Theorem 5.4 computes the equivalence classes of components that correspond to states of latent variables that differ just in the value of H_j. Having access to Γ and L, computing these equivalence classes takes at most $O(nmK)$ time (for each of the m hidden variables we need to compare vectors of values of L of length n for K components).

Once these equivalence classes are computed, the algorithm in Theorem 5.4 sequentially fills in the joint probability table. If the entries with indices of Hamming weight t are filled in, in order to determine the value of a cell with an index of hamming weight $t+1$, we explore at most $2 \max_{i \in [m]} k_i$ elements of the corresponding equivalence classes. Since eventually we explore all K cells of the joint probability table, the total runtime of this phase is bounded by $O(\max_{i \in [m]} k_i)K$. □

C.3 Non-identifiability if Assumption 3.1 is violated

Finally, we prove the impossibility claim in Remark 5.5.

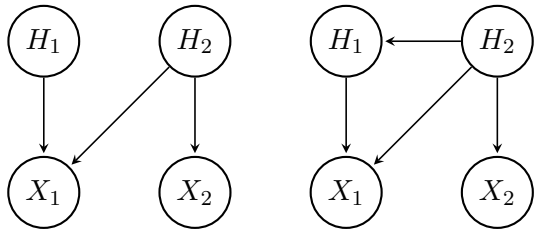


Figure 5: An example of the causal latent models that cannot be distinguished from observed data since Assumption 3.1 is violated

Proof of Remark 5.5. We claim that if Assumption 3.1 is violated, then $\mathbb{P}(H)$ cannot be recovered and moreover G is not identifiable. Consider a pair of models on Figure 5, where variables H_1 and H_2 are binary, i.e., they take values $\{0, 1\}$. Let N_0, N_1, N_2, N_3 and N'_0, N'_1 be independent Gaussian distributions with distinct means and variances.

Suppose that the observed distribution is equal to

$$(X_1, X_2) \sim \frac{1}{9} N_0 \otimes N'_0 + \frac{2}{9} N_1 \otimes N'_1 + \frac{2}{9} N_2 \otimes N'_0 + \frac{4}{9} N_3 \otimes N'_1 \tag{14}$$

Now we show that this distribution can be realized by both models A and B.

1. *Consistency with A.* Let H_1, H_2 be independent random variables that take values $\{0, 1\}$ with probabilities $(1/3, 2/3)$.

$$(X_1, X_2) \sim \sum_{i \in \{0,1\}} \sum_{j \in \{0,1\}} \mathbb{P}(X_2 | H_1 = i, H_2 = j)\, \mathbb{P}(H_1 = i)\, \mathbb{P}(H_2 = j), \text{ where}$$
$$\mathbb{P}(X_1 | H_1 = 0, H_2 = 0) = N_0, \quad \mathbb{P}(X_1 | H_1 = 0, H_2 = 1) = N_1,$$
$$\mathbb{P}(X_1 | H_1 = 1, H_2 = 0) = N_2, \quad \mathbb{P}(X_1 | H_1 = 1, H_2 = 1) = N_3,$$
$$\mathbb{P}(X_2 | H_2 = 0) = N'_0, \quad \mathbb{P}(X_2 | H_2 = 1) = N'_1$$

2. *Consistency with B.* Let H_1, H_2 be binary random variables with the following distribution

$$\begin{aligned} &\mathbb{P}(H_2 = 0) = 1/3 \quad , \mathbb{P}(H_1 = 0 | H_2 = 0) = 1/3, \quad \mathbb{P}(H_1 = 1 | H_2 = 0) = 2/3, \\ &\mathbb{P}(H_2 = 0) = 2/3, \quad \mathbb{P}(H_1 = 0 | H_2 = 1) = 2/3, \quad \mathbb{P}(H_1 = 1 | H_2 = 1) = 1/3 \end{aligned} \tag{15}$$

   Define components of the mixture distribution to be

$$(X_1, X_2) \sim \sum_{i \in \{0,1\}} \sum_{j \in \{0,1\}} \mathbb{P}(X_2 | H_1 = i, H_2 = j)\, \mathbb{P}(H_1 = i, H_2 = j), \text{ where}$$
$$\mathbb{P}(X_1 | H_1 = 0, H_2 = 0) = N_0, \quad \mathbb{P}(X_1 | H_1 = 0, H_2 = 1) = N_3,$$
$$\mathbb{P}(X_1 | H_1 = 1, H_2 = 0) = N_2, \quad \mathbb{P}(X_1 | H_1 = 1, H_2 = 1) = N_1,$$
$$\mathbb{P}(X_2 | H_2 = 0) = N'_0, \quad \mathbb{P}(X_2 | H_2 = 1) = N'_1$$

Since both models A and B realize distribution $\mathbb{P}(X)$, we get that G and $\mathbb{P}(H)$ are not identifiable. Observe that Assumption 3.1 is not satisfied for both A and B, while Assumptions 2.2, 2.3 and 2.4 are satisfied for each of A and B. □

D Proof of Theorem 3.2

Finally, we collect our results into a proof of the main theorem.

Proof of Theorem 3.2. Suppose that Assumptions 2.2, 2.3 and 2.4 hold, then by Theorem 4.2(a), Γ and $\dim(H_i)$, for all i, can be recovered from $\mathbb{P}(X)$. If additionally, the columns of the $|X| \times |H|$ adjacency matrix A are linearly independent, then by Theorem 4.8 (see Corollary B.12, Theorem B.13 and Observation B.8), Γ and $\dim(H_i)$, for all i, can be reconstructed efficiently in $O(n^4)$ time.

Now, suppose that Assumption 3.1 holds. We can extract the map L from the MixOracle (by taking appropriate projections of component distributions). Therefore, since we have Γ, $\dim(H_i)$, $\{\pi(X, i)\}_{i\in[K]}$ and L, by Theorem 5.4 and Observation C.2, we can reconstruct $\mathbb{P}(H)$ efficiently. □

E Algorithms

In this section we describe the full pipeline[1] for learning G from samples of the observed data X. As input we receive a set of samples and as output we return an estimated causal graph G and a joint probability distribution over H. The pipeline consists of the following blocks:

(Step a) **Learning number of components.** Estimates the number of components for all subsets of observed variables of size at most 3.

- Input: Samples from the distribution $\mathbb{P}(X)$
- Output: Estimated number of mixture components $k(S)$ in $\mathbb{P}(S)$ for all $S \subseteq X$, $|S| \leq 3$.

(Step b) **Reconstruction of the bipartite graph.** Implements the algorithm of Theorem 4.8 for learning the bipartite causal graph Γ.

- Input: The number of mixture components $k(S)$ in $\mathbb{P}(S)$ for all $S \subseteq X$, $|S| \leq 3$.
- Output: Estimated bipartite graph Γ and sizes of the domains of hidden variables $\dim(H_i)$.

(Step c) **Learning the projection map L.**

- Input: Samples from the distribution $\mathbb{P}(X)$ and the numbers of components $k(X)$ and $k(X_i)$ for every $i \in [n]$.

[1]The code used to run the experiments can be found at https://github.com/30bohdan/latent-dag

- Output: Estimated projection map L.

(Step d) **Learning the distribution $\mathbb{P}(H)$.** In this step we implement the algorithm described in Theorem 5.4, see also Algorithm 1.

- Input: L, Γ and $\dim(H_i)$ for all $i \in [m]$ and weights $\pi(X, j)$ of $k(X)$ mixture components.
- Output: Estimated joint probability table of $\mathbb{P}(H)$.

We take L, Γ and $\dim(H_i)$ for all i as an input and return the joint probability table for $\mathbb{P}(H)$ as an output.

(Step e) **Learning latent DAG Λ.** In this step we estimate the causal graph over latent variables.

- Input: The joint probability table of $\mathbb{P}(H)$.
- Output: Estimated causal graph Λ over H.

In this paper, we prove theoretical guarantees for Steps (b) and (d), which invoke the mixture oracle MixOracle. Step (a) implements MixOracle, and Steps (c) and (e) are intermediate steps of the pipeline. As long as the oracle is correct, Step (c) is guaranteed to output the correct graph. The correctness of Step (e) depends on the structure learning algorithm used. A nice feature of our algorithm is its modularity, if a better algorithm is developed for one of the steps, it can be incorporated without influencing other parts.

Below we discuss various implementation details for these steps.

Details of Step (a): Our implementation of Step (a) uses the following strategy.

1. We estimate the upper bound k_{max} on the number of components involved in the mixtures of single variables (this can be done using the silhouette score).
2. For every observed variable X_i we train K-means clustering with $k = k_{max}$. After this, we perform agglomerative clustering for every $t \in [2, k_{max}]$, and record the silhouette score for every t. We pick 5 values of t with the best silhouette score.
3. We use the divisibility condition to compute the sets S_{X_i,X_j} of possible numbers of components we expect to see over the pairs of variables X_i, X_j. We use the best 5 predictions from the previous step for every variable X_i and include the candidate for the number of components into S_{X_i,X_j} if it is divisible by one of the top-5 candidates for X_i and for X_j. This step is mainly needed for computational purposes in order to restrict the number of candidates for the number of components observed over the pairs of variables.
4. Next we learn the mixture of k components for every $k \in S_{X_i,X_j}$ over the pairs (X_i, X_j) of observed variables. Similarly as in 2., we train K-means for the largest candidate and perform agglomerative clustering after that.

5. We use divisibility and means voting (discussed in Sec. 6) to decide the best number of components for the single variables and the pairs of variables. In order to do this we make the predicted numbers of components for a pair $X_i, (X_i, X_j)$ to vote for each other if they satisfy the divisibility or means projection condition. We count the vote with the weight proportional to the silhouette score of the predicted number of components. For every X_i, and every pair (X_i, X_j), we take the component with the largest amount of votes as our best prediction.

6. We use means of the components predicted for pairs of variables (X_i, X_j) to estimate the locations of the means for the triples of observed variables. Instead of using K-means with the fresh start we initialize it with predicted locations. This improves the running time. We use K-means and silhouette score to predict the number of components for the triples of observed variables.

Details of Step (b): In this step we use Corollary 4.4, Eq. (7) and Lemma 4.6 to compute entries of the tensor M_3 using the output of Step (a). After this we apply Jennrich's algorithm to learn the components of the tensor. As discussed in Appendix B.3 this is sufficient to reconstruct Γ and $\dim(H_i)$. In case Jennrich's algorithm did not successfully execute due to numerical issues, alternating least squares (ALS) was used as a failsafe. In this case, the number of hidden variables m was used as input.[2]

Details of Step (c): We use Γ and $\dim(H_i)$ to compute the number of components we expect to observe in $\mathbb{P}(X_i)$ for every observed variable X_i and the number of components in the distribution $\mathbb{P}(X)$ over the entire set of observed variables. After this we use K-means to learn the components in the mixture distribution over every variable X_i and over the entire set of observed variables. For every i, and for every mixture component of $\mathbb{P}(X)$, we project its mean into the subspace over which X_i is defined. We use the closest in L_2 distance mean of the components in $\mathbb{P}(X_i)$ as a prediction for the projected component.

Details of Step (d): We implement the algorithm described in Theorem 5.4. See Algorithm 1 for details.

Details of Step (e): Once we obtain the estimated joint probability table, we run the Fast Greedy Equivalence Search [RGSRG17] to learn the edges of the Latent graph H, where we used the Discrete BIC score. FGES returns a CPDAG by default, so some edges may be undirected. We accordingly report both the Structural Hamming Distance (SHD) and the Unoriented Correct Edges (UCE) as metrics for our experiments. We remark that this step may be improved by using other algorithms such as PC [SG91] or other scores, which is an interesting direction for future work.

[2]This can easily be avoided by running ALS for multiple values of m and choosing the best fit. Since this issue arose in only a minority of cases, we did not implement this feature.

Algorithm 1: Learning $\mathbb{P}(H)$

Input:

- A bijective map $L : [k(X)] \to [k(X_1)] \times [k(X_2)] \times \ldots \times [k(X_n)]$;
- A bipartite graph Γ between X and H
- Values $\dim(H_i)$ for $i \in H$.
- Values $\mathbb{P}(Z = i)$ for $i \in [k(X)]$ (the probabilities of observing the mixture components)

Output: An $\dim(H_1) \times \ldots \times \dim(H_m)$ tensor such that $J \cong \mathbb{P}(H)$

`// Phase 1: use Lemma C.1 to compute the sets of components that correspond to a change in a single hidden variable`

arrows = {}
for $H_i \in H$ **do**
    S = $X \setminus ne_\Gamma(H_i)$
    for $c_1, c_2 \in [k(X)]$ **do**
        if $(L(c_2)_S == L(c_1)_S)$ *and* $c_1 \neq c_2$ **then**
            arrows$[H_i][c_1]$.append(c_2)

`// Phase 2: initialize` T `"along the edges"`

$A(0, \ldots 0) = 0, \quad T(0, \ldots 0) = \mathbb{P}(Z = 0)$
for $H_i \in H$ *and* $t \in \dim(H_i)$ **do**
    $A(0, \ldots, t, \ldots 0) = arrows[H_i][0][t]$ `// Note that an order does not matter`
    $J(0, \ldots, t, \ldots 0) = \mathbb{P}(Z = arrows[H_i][0][t])$

`// Phase 3: reconstruct all other entries of the tensor`

$r = 1$
while $r < m$ **do**
    for $ind \in \dim(H_1) \times \ldots \dim(H_r)$ **do**
        for $j = r + 1, \ldots, m$ *and* $t \in \dim(H_t)$ **do**
            Let i be the smallest index at which ind is non-zero.
            Let ind' be an index obtained from ind by changing j-th entry from 0 to t
            Let ind'' be obtained from ind' by changing i-th entry to 0.
            Let x be the unique entry in the intersection of arrows$[H_i][A(ind'')]$ and arrows$[H_t][A(ind)]$.
            $A(ind') = x$
            $J(ind') = \mathbb{P}(Z = x)$

return T

F Experiment details

Data generation For each experiment, the data generation process was as follows:

- (m, n): Chosen from among $(1, 3), (2, 5), (3, 7), (3, 8), (4, 7), (4, 8)$ in the ratio $1 : 2 : 2 : 3 : 1 : 1$

- Domain sizes $|\Omega_i|$: Sampled from $\{2, 3, 4, 5, 6\}$. If $|\Omega| = |\Omega_1| \dots |\Omega_m| > 50$, we skip the experiment.
- $\mathbb{P}(H)$: Generated via the Markov property. For each variable H_i, conditioned on its parents $H_{\mathrm{pa}(i)}$, a discrete distribution supported on Ω_i is chosen as follows: For each element i in Ω_i, a random integer c_i is picked from $[1, 4]$ and distribution picks i with probability proportional to c_i.
- Λ: Choose an arbitrary topological order uniformly at random and sample each directed edge independently with probability 0.6.
- Γ: Sample each directed edge from H to X with probability 0.5. Enforce assumption 3.1 and linear independence of the columns a_j of the adjacency matrix A.
- Components: We generate Gaussian components for every X_i in $\mathbb{R}^5$ with random means and covariances. We take the means of the components to be sampled uniformly at random from the unit sphere. We take random symmetric diagonally dominant covariance matrices with the largest eigenvalue being 0.01. (Note that for 50 points on a unit 5-dimensional sphere, we expect to observe a pair of points at distance of the same order of magnitude).
- Samples: We generate samples from the mixture components generated on the previous step with probabilities defined by $\mathbb{P}(H)$.

We do not enforce minimum probability sizes or cluster sizes. As a result, the data generating process is likely to generate models which are extremely difficult to learn (e.g. if a randomly generated probability is very small, a mixture component will have few samples, which makes learning difficult). As a result, some random configurations may fail. We ran a total of 724 experiments; out of these, 8.3% failed in the oracle learning phase and another 8.8% failed to produce a graph because of very high domain sizes or unfeasible L. In the cases when the Jennrich algorithm failed due to numerical issues, this was caught and replaced with ALS for practical purposes as described in Step (b) above. These errors are conveniently caught during runtime and can be attributed to either the data generation process or the finite sample size as described above. Fig. 4 reports the metrics for the remaining 600 experiments: 300 experiments each for $N = 10000$ samples and $N = 15000$ samples. The experiments were run on a single node of an internal cluster.

Experiments with smaller sample size. The number of samples in the experiments discussed above is chosen so that every cluster component has approximately 20 samples. We also explored the behaviour of our algorithms when the number of samples is much smaller. We ran a total of 136 experiments for $N = 1000$ samples, with (m, n) chosen from $(1, 3), (2, 5), (3, 7), (4, 7), (3, 8)$ in proportion $1 : 2 : 1 : 1 : 1$. Out of these, 4.4% failed in the oracle learning phase and another 8.8% failed to produce a graph because of very high domain sizes or unfeasible L. Furthermore, out of all failures, 25% happen for $(m, n) = (4, 7)$ and other 37.5% happen for $(m, n) = (3, 8)$. We report the metrics on Fig. 6.

We mention, that with $N = 1000$ samples, we were able to recover H and Ω even in the cases

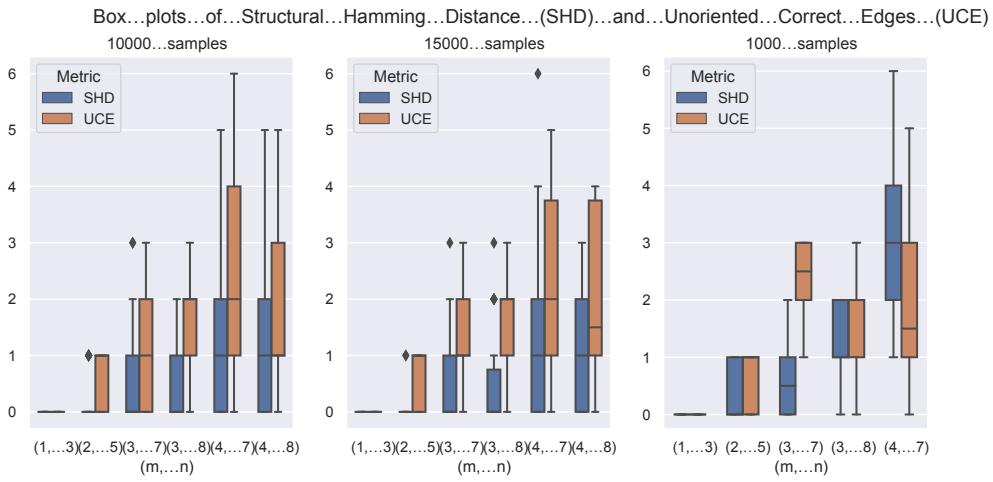


Figure 6: Average Structural Hamming distance for recovery of G, where $m = |H|$ and $n = |X|$.

when several latent states had fewer than five observations. Also, for comparison, to give an example where we were not able to recover H and Ω exactly: the mixture model had 48 components with 1, 2, 2, 3, 3, 5, 5, 5, 6 . . . , 53, 55 samples per component. This is clearly an extremely challenging setup: Some states had only a few observations and the true number of components is unknown to the procedure.

Choice of parameters for learning Λ. Once we have recovered the estimated joint probability table of H, to learn Λ, we use the Fast Greedy Equivalence Search algorithm [RGSRG17] with the Discrete BIC score. We use the PyCausal library [WHE+19]. We used the default parameters (no hyperparameter tuning) and in particular, we did not assume faithfulness.

Approximate Runtime The average runtimes for each experiment are in the following table.

Table 1: Average runtime in seconds

(m, n)	**10000 samples**	**15000 samples**
(1, 3)	30.64 s	53.06 s
(2, 5)	89.03 s	148.81 s
(3, 7)	288.25 s	385.27 s
(3, 8)	320.25 s	616.86 s
(4, 7)	297.32 s	400.04 s
(4, 8)	361.28 s	604.14 s

Average number of edges For our experiments, the average total number of edges in Λ, Γ (also known as NNZ of G) are in the following table.

Table 2: Average number of edges for different settings

(m, n)	**Number of Samples**	**Average number of edges in** $G = (\Lambda, \Gamma)$
(1, 3)	10000	3.0
(1, 3)	15000	3.0
(1, 3)	1000	3.0
(2, 5)	10000	7.15
(2, 5)	15000	6.95
(2, 5)	1000	6.98
(3, 7)	10000	13.52
(3, 7)	15000	13.2
(3, 7)	1000	13.7
(3, 8)	10000	15.27
(3, 8)	15000	15.16
(3, 8)	1000	15.3
(4, 7)	10000	17.43
(4, 7)	15000	18.17
(4, 7)	1000	18.35
(4, 8)	10000	19.87
(4, 8)	15000	20.13

Scatter plots The scatter plots for the Structural Hamming distance (SHD) versus the total number of edges $|E(G)|$ in G and that of the unoriented correct edges (UCE) vs $|E(G)|$ is given in Fig. 7.

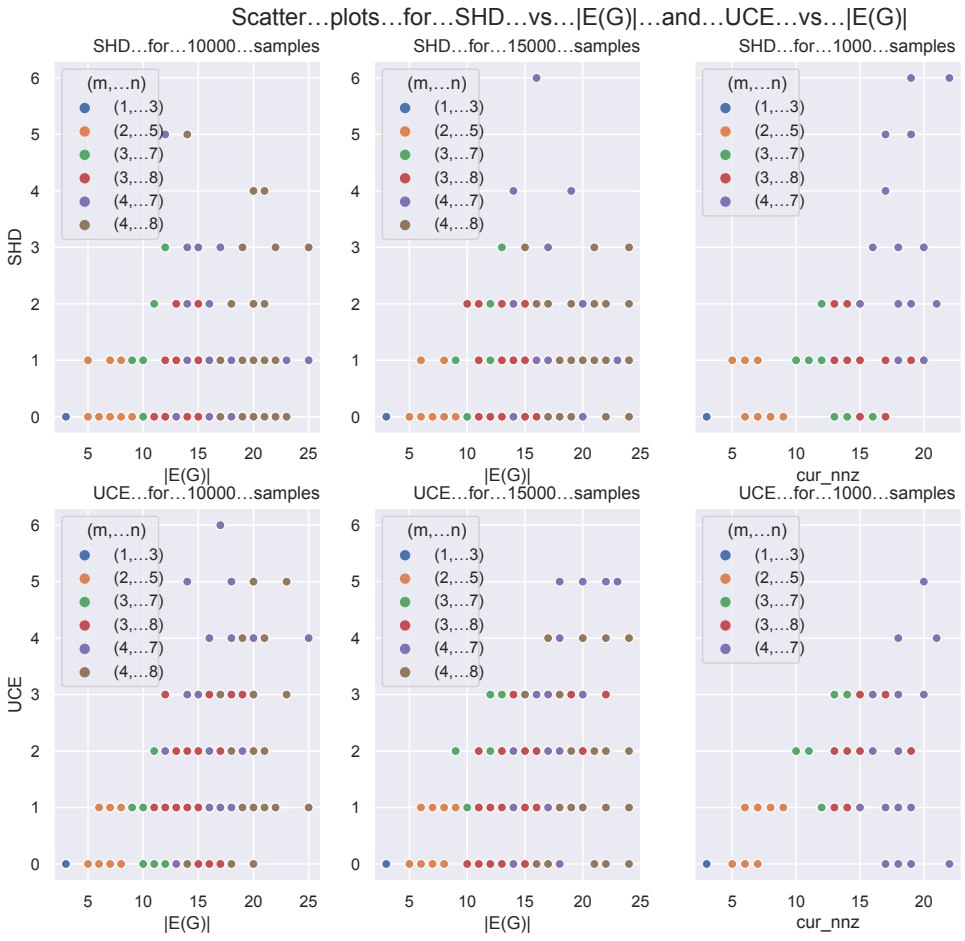


Figure 7: Scatterplots where $m = |H|$ and $n = |X|$.