Learning Linear Causal Representations

Using Higher-Order Cumulants

A THESIS PRESENTED
BY
PAULA LEYES CARRENO
TO
THE DEPARTMENT OF COMPUTER SCIENCE

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
BACHELOR OF ARTS
IN THE JOINT SUBJECTS OF
COMPUTER SCIENCE AND MATHEMATICS

HARVARD UNIVERSITY
CAMBRIDGE, MASSACHUSETTS

APRIL 2024

Thesis advisor: Anna Seigal Paula Leyes Carreno

Learning Linear Causal Representations
Using Higher-Order Cumulants

Abstract

Causal representation learning seeks to extract a representation of data that captures causal relationships, allowing for better understanding, prediction, and manipulation of the underlying processes. Such a representation is identifiable if the transformation from the latent representation to the observed variables and the latent model are both unique. In this thesis, we study the identifiability of causal representation learning in the linear setting. We prove that one perfect intervention per latent variable is both sufficient and necessary for identifiability given access to finitely many cumulants of the observed variables. We further show that one soft intervention per latent variable does not suffice for identifiability. The proof for the sufficiency of perfect interventions is constructive. We implement our algorithm for causal representation learning and verify its performance on synthetic data.

Contents

Listing of figures

List of symbols

Symbol	Meaning/Description
$\overline{G}$	Transitive closure of the graph G
$M_{:l} = [M]_l$	l^{th} column of matrix M
$M_{l:} = \mathbf{m}_l$	l^{th} row of matrix M
M^T	transpose of matrix M
I_n	Identity matrix of size $n \times n$
$\mathbf{0}_{m \times n}$	$m \times n$ zero matrix
$\mathrm{diag}(T)$	$m \times m$ diagonal matrix of diagonal $m \times m \times m$ tensor T, satisfying $\mathrm{diag}(T)_{ii} = T_{iii}$
$[n]$	Integer set $\{0, 1, \ldots, n-1\}$
σ	Given a permutation matrix P, we define its associated permutation σ as $\sigma(i) = j \Leftrightarrow P_{ij} = 1$
$\mathbb{S}^n$	unit n-sphere: $\{x \in \mathbb{R}^{n+1} \mid \|x\| = 1\}$
$\approx$	Approximately equal

Acknowledgments

I would like to express my sincere gratitude to my thesis advisor Professor Anna Seigal, for her unwavering support and patience throughout this project and for being a wonderful mentor during my last two years at Harvard.

I am also deeply indebted to Doctor Chiara Meroni. I could have not carried out this research project without her guidance and advice, and I feel very fortunate to have had the opportunity to collaborate with and learn from her.

I would also like to thank Professor Weiwei Pan and Professor David Alvarez-Melis for generously agreeing to act as readers for this thesis and for their helpful suggestions.

Moreover, I'd like to extend my gratitude to Álvaro Ribot, Ada Wang and Aida Maraj, for their insightful advice and feedback.

Lastly, I would like to thank my parents, for always believing in me and supporting me throughout the years.

1 Introduction

1.1 Motivation

A key determinant of the performance of machine learning methods is the choice of data representation they are applied on [5]. Modern methods like deep learning and large language models learn high-level representations of data by themselves, which leads to more effective and robust performance across a wide range of tasks [8]. The representations they learn, however, do not necessarily reflect the true data generating process, which can make it difficult to interpret and understand them. Being able to reason about these learned representations is not only key in diagnosing why the model may be making certain mistakes or failing to generalize to unseen data, but also to explain model predictions to stakeholders in fields where interpretability is crucial, such as healthcare [51]. Causal representation learning, also referred to as causal disentanglement, aims to enhance the interpretability of

learned representations by simultaneously learning latent high-level representations of data and capturing their causal structure.

A representation is identifiable if both the latent causal structure and the transformation from the latent representation to the observed data are unique. Identifiability is crucial to achieve meaningful and well-founded disentanglement, and in applications such as causal discovery [28]. It is, however, impossible without imposing structure on the data generation process or auxiliary information [26, 34]. In [48], Squires et al. study identifiability by considering observed variables that are a linear transformation of a linear latent causal model. They show that access to the covariance matrix of the observed variables alone is not sufficient to recover the model and the transformation from latent to observed variables, and prove that interventional data is needed for identifiability. We build on their work by taking a method of moments approach [40] to the study of identifiability. We notice that, unlike the covariance matrix, the third-order cumulant of the observed variables exhibits a structure that renders its decomposition unique. In view of this, we examine whether working with the third-order cumulant instead of the covariance matrix offers advantages for identifiability, in particular regarding the quantity and type of interventional data required. We use the terminology in [49] for interventions: given a variable Z_k, a *perfect* intervention at Z_k removes the causal dependencies of Z_k on its parents and changes its stochasticity, and a *soft* intervention modifies the magnitude of the causal dependencies of Z_k on its parents and changes its stochasticity. It is also possible to have an intervention that sets Z_k to a deterministic value, but we do not consider such case here as it is more restrictive than the former two and overly simple for certain applications [36].

1.2 Our contributions

Our contributions are both theoretical and practical. On the theoretical front, we show that, given finitely many cumulants of the observed variables, interventional data is necessary for identifiability. In particular, we prove that in the setting of perfect single-node interventions, one intervention per latent node is both sufficient

and in the worst case necessary for identifiability. We further show that one soft intervention per latent node is not sufficient to recover the graph or the mixing from latent to observed variables in the worst case.

On the practical front, we propose a method for causal disentanglement using the third-order cumulant of the observed variables as input. The algorithm is algebraically simpler than the one proposed in [48], presents similar sample complexity and accuracy of recovery, and demonstrates robustness to a certain type of misspecification of the latent model.

The subsequent chapters of the thesis are structured as follows: Chapter 2 provides some theoretical background and an overview of related work in the field. In Chapter 3 we present our theoretical results. In Chapter 4 we apply our method for causal disentanglement to synthetic data and analyze its performance and robustness to model misspecification. Finally, Chapter 5 suggests some directions for future work. The code for our computational results can be found at: https://github.com/paulaleyes14/linear-causal-representations

2 Background

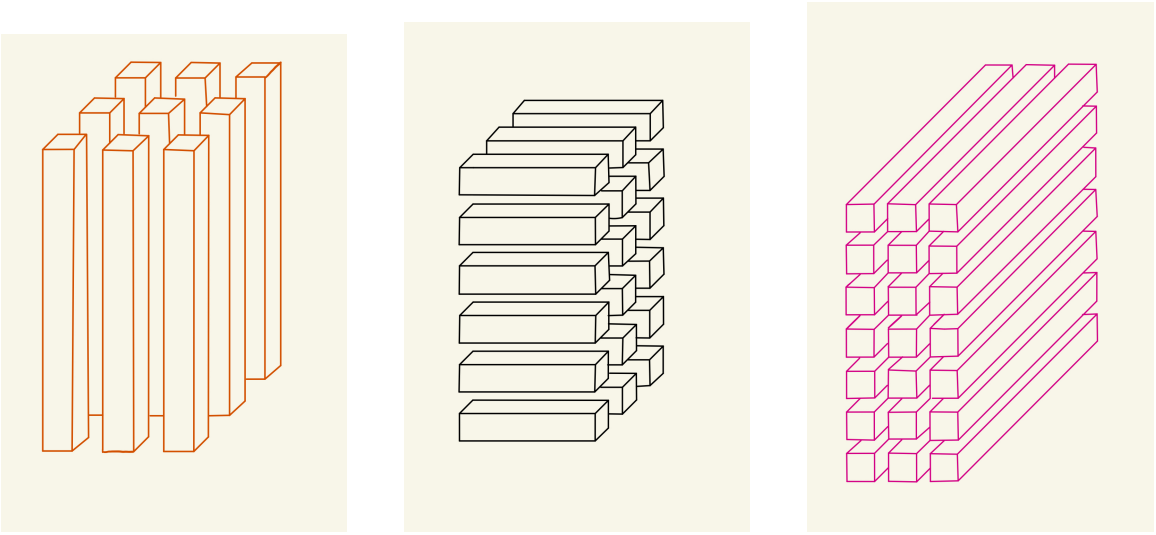
2.1 Preliminaries

Tensors

A *tensor* is an element of the tensor product of N vector spaces. After fixing a basis for each vector space, it can be represented as a multidimensional array. The *order* of a tensor is the number of dimensions of such array. A vector is thus an order-1 tensor, a matrix is an order-2 tensor, an order-3 tensor has three indices, etc.

Fibers extend the concepts of rows and columns of a matrix to higher dimensions. They are obtained by fixing all indices of a tensor but one. As an example, matrix columns are mode-1 fibers, since they are obtained by fixing all indices of a matrix but the first one, and matrix rows are mode-2 fibers. The fibers obtained for order-3 tensors when fixing the third index are called tube fibers. For higher-order tensors,

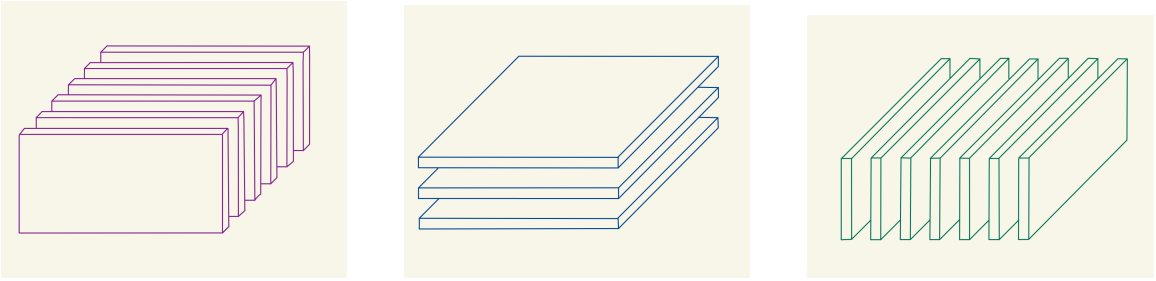
we just use the terminology of mode-k fibers. Figure 2.1.1 shows the fibers of an order-3 tensor.



(a) Mode-1 fibers **(b)** Mode-2 fibers **(c)** Mode-3 fibers

Figure 2.1.1: Fibers of an order-3 tensor

Similarly to fibers, tensor *slices* are obtained by fixing all indices of a tensor but two. They are thus two-dimensional objects. Figure 2.1.2 shows the slices of an order-3 tensor.



(a) Frontal slices **(b)** Horizontal slices **(c)** Lateral slices

Figure 2.1.2: Slices of an order-3 tensor

Using fibers, we may flatten a tensor, that is, reshape it into a matrix. The

mode-n flattening of a tensor T, denoted by $T_{(n)}$, is a matrix whose columns are the mode-n fibers of T. As an example, let $T \in \mathbb{R}^{3\times3\times3}$, with frontal slices:

$$T_1 = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 2 & 3 \end{bmatrix}, \quad T_2 = \begin{bmatrix} 4 & 5 & 6 \\ 4 & 5 & 6 \\ 4 & 5 & 6 \end{bmatrix}, \quad T_3 = \begin{bmatrix} 7 & 8 & 9 \\ 7 & 8 & 9 \\ 7 & 8 & 9 \end{bmatrix}. \tag{2.1}$$

Then we have that

$$T_{(1)} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{bmatrix},$$

$$T_{(2)} = \begin{bmatrix} 1 & 1 & 1 & 4 & 4 & 4 & 7 & 7 & 7 \\ 2 & 2 & 2 & 5 & 5 & 5 & 8 & 8 & 8 \\ 3 & 3 & 3 & 6 & 6 & 6 & 9 & 9 & 9 \end{bmatrix}, \tag{2.2}$$

$$T_{(3)} = \begin{bmatrix} 1 & 1 & 1 & 2 & 2 & 2 & 3 & 3 & 3 \\ 4 & 4 & 4 & 5 & 5 & 5 & 6 & 6 & 6 \\ 7 & 7 & 7 & 8 & 8 & 8 & 9 & 9 & 9 \end{bmatrix}.$$

The flattenings of a tensor can be used to multiply it by one or several matrices. We denote the n-mode product of a tensor $T \in \mathbb{R}^{K_1\times K_2\times\cdots\times K_N}$ with a matrix $M \in \mathbb{R}^{J\times K_n}$ by $T \times_n M$. The resulting tensor lives in $\mathbb{R}^{K_1\times\cdots\times K_{n-1}\times J\times K_{n+1}\times\cdots\times K_N}$ and can be expressed using T's mode-n flattening by $MT_{(n)}$. Given a tensor $T \in \mathbb{R}^{K\times K\times\cdots\times K}$ and a matrix $M \in \mathbb{R}^{J\times K}$, we let $M \cdot T$ denote the $J \times J \times \cdots \times J$ tensor resulting from multiplying T by M along every mode.

Just as with matrices, we can define what it means for higher-order tensors to be *symmetric*. An order-N tensor T is said to be symmetric if all of its modes have the

same size and it is invariant under permutation of its indices. Mathematically, we require that

1. $T \in \mathbb{R}^{K \times K \cdots \times K}$,
2. $T_{k_1 k_2 \cdots k_N} = T_{k_{\sigma(1)} k_{\sigma(2)} \cdots k_{\sigma(N)}}$ for all permutations σ on $\{1, 2, \cdots, N\}$.

A tensor $T \in \mathbb{R}^{K_1 \times K_2 \times \cdots \times K_N}$ is said to have *rank* one if it can be expressed as the outer product of N vectors:

$$T = v_1 \otimes v_2 \otimes \cdots \otimes v_N. \tag{2.3}$$

Elementwise, we have

$$T_{k_1 k_2 \cdots k_N} = (v_1)_{k_1} (v_2)_{k_2} \cdots (v_N)_{k_N} \quad \text{for all } 1 \leq k_n \leq K_n. \tag{2.4}$$

More generally, a tensor is said to have rank R if the minimum number of rank-one terms we need to add to express it is R.

The CANDECOMP/PARAFAC (CP) [11, 22] decomposition of a tensor T factorizes it into a sum of rank-one tensors. Thus, decomposing a third-order tensor $T \in \mathbb{R}^{K_1 \times K_2 \times K_3}$ using CP yields:

$$T \approx \sum_{r=1}^{R} u_r \otimes v_r \otimes w_r, \tag{2.5}$$

where $u_r \in \mathbb{R}^{K_1}$, $v_r \in \mathbb{R}^{K_2}$ and $w_r \in \mathbb{R}^{K_3}$. When R equals the rank of the tensor, the decomposition is called rank decomposition.

We can group the u, v and w vectors appearing in the decomposition into matrices:

$$U = \begin{bmatrix} u_1 & u_2 & u_3 & \cdots & u_R \end{bmatrix}, \quad V = \begin{bmatrix} v_1 & v_2 & v_3 & \cdots & v_R \end{bmatrix}, \quad W = \begin{bmatrix} w_1 & w_2 & w_3 & \cdots & w_R \end{bmatrix}. \tag{2.6}$$

These are called the *factor matrices* of the decomposition. Imposing constraints on their form yields different versions of the CP decomposition. Of interest to us will

be the symmetric CP decomposition, in which $U = V = W$. Whenever the decomposition of a tensor is discussed in this thesis, we will be referring specifically to its symmetric CP decomposition.

Unlike matrices, which can be expressed in various ways as a sum of rank-one matrices when their rank is greater than 1, higher-order tensors often have a unique decomposition up to significantly higher ranks. We highlight the result by Robert Jennrich, documented by Richard A. Harshman [22]:

Theorem 2.1 (Jennrich's theorem [22]). *If $T = \sum_{i=1}^{R} \alpha_i v_i^{\otimes 3}$ with $v_1, \ldots, v_R$ linearly independent, then T has rank R and its decomposition is unique up to permutation and scaling.*

An algorithm to recover the factor matrix of such a decomposition up to permutation and scaling of columns is presented in [32]. It proceeds as follows:

Algorithm 1 Simultaneous diagonalization [32]

Input: $T \in \mathbb{R}^{K \times K \times K}$

Output: $V = \begin{bmatrix} \uparrow & \uparrow & & \uparrow \\ v_1 & v_2 & \ldots & v_R \\ \downarrow & \downarrow & & \downarrow \end{bmatrix}$ such that $T = \sum_{i=1}^{R} \alpha_i v_i^{\otimes 3}$

$a, b \sim \mathbb{S}^{K-1}$

$M_a \leftarrow \sum_{i=1}^{R} \langle v_i, a \rangle \alpha_i v_i v_i^T$

$M_b \leftarrow \sum_{i=1}^{R} \langle v_i, b \rangle \alpha_i v_i v_i^T$

eigvalues, eigvectors $\leftarrow$ eigendecompose$(M_a (M_b)^\dagger)$

return eigvectors

M_a and M_b are weighted sums of the horizontal slices of T, each slice weighted by the i^{th} entry of a and the i^{th} entry of b, respectively. Equivalently, $M_a = V D_a V^T$ and $M_b = V D_b V^T$, where D_a is a diagonal matrix satisfying that $(D_a)_{ii} = \langle v_i, a \rangle \alpha_i$, and similarly for D_b. We thus have that $M_a (M_b)^\dagger = V D_a (D_b)^\dagger V^\dagger$. The columns of V are precisely the eigenvectors of this matrix, as desired.

Graphs and graphical models

A *graph* $G = (V, E)$ is a mathematical structure consisting of a set of vertices (also called nodes), denoted by V, and a set of pairs of vertices called edges, denoted by E. In an *undirected graph*, edges are unordered pairs: $(u, v) \in E$ denotes an edge from vertex u to vertex v and viceversa. In a *directed graph*, they are ordered: $(u, v) \in E$ denotes an edge from vertex u to vertex v. A vertex with at least two incoming edges is called a *collider* [53].

A directed graph $G = (V, E)$ is a *directed acyclic graph (DAG)* if it contains no directed cycles, that is, there does not exist a sequence of edges $(u_1, v_1), (u_2, v_2), \ldots, (u_t, v_t) \in E$ such that $v_n = u_{n+1}$ for all $n \in \{1, \ldots, t-1\}$ and $v_t = u_1$. Given two vertices $u_1, u_2 \in V$, we say that

1. u_1 is a *parent* of u_2, denoted by $u_1 \in \mathrm{pa}(u_2)$, if $(u_1, u_2) \in E$.
2. u_1 is a *child* of u_2, denoted by $u_1 \in \mathrm{ch}(u_2)$, if $(u_2, u_1) \in E$.
3. u_1 is an *ancestor* of u_2, denoted by $u_1 \in \mathrm{an}(u_2)$, if there exists a directed path from u_1 to u_2 in G.
4. u_1 is a *descendant* of u_2, denoted by $u_1 \in \mathrm{des}(u_2)$, if there exists a directed path from u_2 to u_1 in G.
5. u_1 is a *source vertex* if $\mathrm{pa}(u_1) = \emptyset$.
6. u_1 is a *sink vertex* if $\mathrm{ch}(u_1) = \emptyset$.

If indices suffice to identify vertices, we may replace u_1 by 1 and u_2 by 2 in definitions 1-4 above, i.e., $u_1 \in \mathrm{pa}(u_2) \Leftrightarrow 1 \in \mathrm{pa}(2)$, and similarly for children, ancestors, and descendants.

The *transitive closure* of a directed graph $G = (V, E)$, is the graph $\overline{G} = (V, \overline{E})$ satisfying that $(u, v) \in \overline{E}$ for all vertices $u, v \in V$ such that there exists a path from u to v in G [41].

An *Erdős–Rényi model* is a model for generating random undirected graphs. There exist two variations of the model; the one relevant for this thesis is the

$G(n, d)$ model. In such model, a graph is generated by considering n vertices and including each possible edge between pairs of vertices independently with probability d [42]. We refer to the parameter d as the density of the model.

Graphs can be used to represent relationships between entities: each vertex represents an entity and edges encode the relationships between them. When such representation takes the form of a directed acyclic graph, we refer to it as a DAG model. In a *probabilistic graphical model*, the entities are random variables, and edges encode conditional dependencies between them [37]. If the graph is a DAG, the random variables represented by two sets of vertices X and Y are conditionally independent given a third set Z if, for all $x \in X$ and $y \in Y$, there is no consecutive sequence of edges (regardless of their directions) between x and y containing no colliders other than those in Z or with descendants in Z, and no non-collider vertices in Z [31].

The *skeleton* of a graph G is the undirected graph with the same vertices and edges as G [24]. Given a graph $G = (V, E)$ and a set $S \subseteq V$, the *induced subgraph* of G over S is the graph whose set of vertices is S and whose edge set consists of all the edges in E connecting vertices in S [16]. A *v-structure* is a three-vertex induced subgraph of G. A *Markov equivalence class* is a set of DAGs that encode the same set of conditional independencies. Two DAGs are in the same Markov equivalence class only if they have the same skeleton and the same v-structures [56].

Cumulants

Cumulants are a set of quantities used to describe a probability distribution. The *cumulant generating function* of a random variable $X \in \mathbb{R}$ is the natural logarithm of its moment generating function:

$$K(t) = \log \mathbb{E}[e^{tX}] \tag{2.7}$$

The d^{th}-order cumulant of X, denoted by κ_d, is the result of evaluating the d^{th} derivative of the cumulant generating function at 0. Alternatively, just like moments can be obtained from the Taylor expansion of the moment generating function,

cumulants can be obtained from the Taylor expansion of the cumulant generating function:

$$K(t) = \sum_{d=0}^{\infty} \kappa_d \frac{t^d}{d!} \tag{2.8}$$

The first-order cumulant of X is its mean, the second-order cumulant is its variance, and the third-order cumulant equals its third-order central moment. Higher-order cumulants are polynomial functions of the central moments [35]. This relationship can be used to calculate cumulants if the moment generating function does not exist.

The *joint cumulant generating function* of a vector of random variables $\mathbf{X} = (X_1, \ldots, X_n)$ is the natural logarithm of its joint moment generating function:

$$K(\mathbf{t}) = \log \mathbb{E}[e^{\mathbf{tX}}] = \log \mathbb{E}[e^{t_1 X_1 + \ldots + t_n X_n}] \tag{2.9}$$

for $\mathbf{t} = (t_1, \ldots, t_n) \in \mathbb{R}^n$. Given a set of indices $\{i_1, i_2, \ldots, i_r\} \subseteq \{1, 2, \ldots, n\}$, the *joint cumulant* of $X_{i_1}, X_{i_2} \ldots, X_{i_r}$, denoted by $\kappa_r(i_1, i_2, \ldots, i_r)$, is defined as the coefficient of $\frac{t_{i_1} t_{i_2} \ldots t_{i_r}}{r!}$ in the Taylor series expansion of $K(\mathbf{t})$:

$$K(\mathbf{t}) = \sum_{d=0}^{\infty} \left(\sum_{1 \leq i_1, \ldots, i_d \leq n} \kappa_d(i_1, \ldots, i_d) \frac{t_{i_1} \ldots t_{i_d}}{d!} \right) \tag{2.10}$$

Note that $\kappa_r(i_1, i_2, \ldots, i_k) = \kappa_r(\sigma(i_1), \sigma(i_2), \ldots, \sigma(i_k))$ for all permutations σ of $\{i_1, \ldots, i_r\}$, which matches our intuition that the joint cumulant of a set of variables should not depend on their ordering.

The *d^{th}-order cumulant tensor* of $\mathbf{X}$, denoted by $\kappa_d(\mathbf{X})$, is a symmetric $n \times n \times \cdots \times n$ (d times) tensor satisfying $\kappa_d(\mathbf{X})_{i_1, \ldots, i_d} = \kappa_d(i_1, \ldots, i_d)$. Whenever we mention the d^{th}-order cumulant of a random vector in this thesis we will be referring to its d^{th}-order cumulant tensor. Cumulant tensors present several interesting properties [35]. Of interest to us will be the following:

1. Multilinearity: let $\mathbf{X}$ be an $\mathbb{R}^n-$valued random variable, M be a $m \times n$ matrix, and recall that we use $\cdot$ to denote the multiplication of a tensor by a

matrix along every mode. Then

$$\kappa_d(M\mathbf{X}) = M \cdot \kappa_d(\mathbf{X}). \tag{2.11}$$

2. Cumulant tensors of independent random variables are diagonal.

K-statistics are a family of formulas used to estimate cumulants from a data sample. They are minimum-variance unbiased estimators, meaning their expected value equals the true value of the cumulant (unbiased), and they have the lowest variance among all unbiased estimators (minimum variance) [20].

2.2 Related work

As mentioned in Section 1.1, the goals of causal representation learning are twofold: it aims to learn a representation of data and the causal dependencies relating the learned features. We present previous work in the identifiability of causal representation learning, as well as that of each of these aspects.

Identifiable representation learning

As its name indicates, the field of identifiable representation learning studies the identifiability of latent representations from observed data. Traditionally, identifiable representation learning has been studied within the framework of independent component analysis (ICA). Classical ICA assumes that observed signals are linear mixtures of independent source signals, and the goal is to separate these sources from the observed mixtures [14]. In [13], Comon shows that such representation is identifiable if and only if at most one source is normally distributed, under the assumptions that the number of sources equals the number of observed variables and the mixing matrix is invertible. Eriksson et al. [18] extend this result to the case when the number of sources can exceed the number of observed variables, under the assumption that there is no Gaussian source, and Wang et al. [58] to the general case. Various identifiability conditions have also

been obtained for the nonlinear setting, based on incorporating auxiliary variables [25] or introducing alternative assumptions on the mixing [62]. These models assume independence of the latent variables or conditional independence given the auxiliary variables, which is often too restrictive of an assumption. Recent work by Zimmerman et al. [63] manages to alleviate the independence assumption by assuming instead that the latent variables follow a known distribution. Similarly, in [1], Ahuja et al. assume knowledge of (at least part of) the mechanisms governing the evolution of the latent variables, and use the fact that such mechanisms constrain the set of possible latent representations that are consistent with the observed data to circumvent the independence assumption. These works, however, do not consider the case when the latent variables are causally related.

Causal structure learning

The goal of causal structure learning is to determine, from measurements of variables $\mathbf{X} = (X_0, \ldots, X_{p-1})$, the graph G relating the variables. It is usually assumed that the variables follow a linear DAG model, which we may study using a linear structural equation model:

$$\mathbf{X} = \Lambda \mathbf{X} + \boldsymbol{\epsilon}, \tag{2.12}$$

where $|\Lambda|_{ij} \neq 0$ if and only if there exists an edge $X_j \to X_i$ in G, and $\boldsymbol{\epsilon}$ is the noise term. Using observational data alone, full identifiability of G can only be achieved if the entries of $\boldsymbol{\epsilon}$ are non-Gaussian. Shimizu [44] and Améndola et al. [3] present methods for model recovery in such a setting, based on tensor decomposition of higher-order cumulants of $\mathbf{X}$, or on the vanishing of equations in entries of these cumulants, respectively. If the entries of $\boldsymbol{\epsilon}$ are Gaussian, only the Markov equivalence class of G can be recovered from observational data. This is because two graphs belonging to the same Markov equivalence class encode the same set of conditional independencies between their nodes, so the statistical dependencies captured by their respective covariance matrices are the same [56]. Interventional data makes it possible to reduce the size of the recovered Markov equivalence class

[23]. Reducing it to size 1 is equivalent to recovering G. In [17], Eberhardt et al. show that, in the worst case, $p-1$ interventions are required to do so.

Learning latent DAG models and causal representation learning

Although the field of causal disentanglement has garnered increased attention in recent years, the problem of learning the structure of latent variable models is not new. In [45], Silva et al. propose a two-step approach to learning the causal structure among latent variables. Such approach relies on the vanishing Tetrad conditions [47], and requires that every observed variable is an *anchor*, that is, has exactly one latent parent [21, 43]. Despite being less restrictive, many recent works also impose structural assumptions on the map from latent to observed variables: in [10], Cai et al. propose a method to estimate the structure over latent variables in the non-Gaussian setting. The method relies on the so-called Triad constraints, and requires that every latent variable has at least two corresponding anchors. In [30], Kivva et al. study the setting where the latent variables are discrete, and show that the latent model is identifiable under the assumption that no two latent nodes have the same set of observed children. Finally, Jiang et al. [27] impose structural assumptions in the setting where both the mixing function and causal model are nonparametric. They show that one intervention per latent node is sufficient for identifiability of the causal model up to *isolated edges*, as long as the latent graph lacks *imaginary subsets*. Isolated edges are edges $Z_i \to Z_j$ satisfying that $\text{pa}(Z_i) = \text{des}(Z_j) = \emptyset$, while imaginary subsets are subsets of observed variables that may not be the children of a single latent variable.

Ahuja et al. [2] and Brehmer et al. [7] do not restrict the map, but assume access to *paired* counterfactual data, which is often not realistic in applications such as image restoration [61] and biology [50]. In [48], Squires et al. consider the setting of *unpaired* data with a linear mapping from latent to observed variables. As mentioned in Section 1.1, they prove that, having access to the covariance matrix of the observed variables alone, one intervention per latent node is sufficient and necessary for identifiability. Their results strengthen some of those of Liu et al. [33],

another work considering unpaired data, and are extended to consider the setting of nonlinear transformations from latent to observed variables in [9].

In [60], Zhang et al. show that, in the infinite-data regime, where the observational and interventional distributions of the observed variables can be determined exactly, one soft intervention per latent node suffices for identifiability of the latent causal model. They require the transformation from latent to observed variables to be polynomial, but not necessarily linear. Varici et al. [54] consider the setting of a linear mixing with nonlinear causal relationships. They leverage this nonlinearity in latent space to show that one soft intervention per latent node is sufficient for identifiability of the causal model, though not the transformation from latent to observed variables. Finally, as in [27], von Kügelgen et al. [57] consider the setting where both the mixing function and causal model are nonparametric, but without structural assumptions on the mixing function. They prove that, in the setting of unpaired data, one intervention per latent variable is sufficent for identifiability if the number of latent variables is two, and extend their results to allow for an arbitrary number of latent variables by considering paired data.

3

Theoretical Results

3.1 Setup

We consider q latent variables $\mathbf{Z} = (Z_0, \ldots, Z_{q-1})$ and p observed variables $\mathbf{X} = (X_0, \ldots, X_{p-1})$. The latent variables follow a linear DAG model. We let G denote such DAG, and specify the model using a linear structural equation model:

$$\mathbf{Z} = \Lambda \mathbf{Z} + \boldsymbol{\epsilon}. \tag{3.1}$$

We let $\Omega^{[n]} = \kappa_n(\boldsymbol{\epsilon})$ and $w_l = \Omega^{[3]}_{lll} = \kappa_3(\epsilon_l)$, and we have that $|\Lambda|_{ij} \neq 0$ if and only if there exists an edge $Z_j \to Z_i$ in G. We assume that the observed variables are an injective linear mixing of the latent variables $\mathbf{X} = B\mathbf{Z} = B(I_q - \Lambda)^{-1}\boldsymbol{\epsilon}$, where the latter expression is obtained by isolating $\mathbf{Z}$ in equation (3.1). Our goal is to recover the model parameters B and Λ. We list our modeling assumptions below:

1. $2 \leq q \leq p$.
2. The noise vector $\boldsymbol{\epsilon}$ is assumed to have statistically independent and non-Gaussian entries.
3. B is full rank.
4. Let H denote the Moore-Penrose inverse of B. We assume that the rows of H have norm one and that the entry of largest absolute value in each row of H is positive. If multiple entries have the same absolute value, we assume that the leftmost one is positive.
5. The model parameters are generic.

Assumption 4 simply fixes the inherent scaling indeterminacy of the problem: $\mathbf{X} = ((I_q - \Lambda)H)^\dagger \boldsymbol{\epsilon}$ may be rewritten as $\mathbf{X} = ((I_q - \Lambda)MM^{-1}H)^\dagger \boldsymbol{\epsilon}$ for any diagonal matrix M with positive entries without changing $\mathbf{X}$'s distribution. Thus, it holds without loss of generality. Assumption 5 also does. Regarding assumption 2, as remarked in [48], the entries of $\boldsymbol{\epsilon}$ being independent holds for a causally sufficient structural equation model. Assuming that $q \leq p$ and that the entries of $\boldsymbol{\epsilon}$ are non-Gaussian facilitates compatibility of our setup with the identifiability guarantees of ICA, but these assumptions can be relaxed. See Chapter 5 and [14, 18, 29, 58].

3.1.1 Connection to ICA

As mentioned in Section 2.2, classical ICA posits that the observed variables are a linear mixing of latent sources. If the number of sources is less than or equal to the number of observed variables, the sources are non-Gaussian and statistically independent, and the mixing matrix has full rank, ICA is identifiable and the sources and mixing matrix can be recovered up to permutation and column scaling [13, 18]. This is precisely our setting: recall that $\mathbf{X} = B(I_q - \Lambda)^{-1}\boldsymbol{\epsilon}$, we assume that $q \leq p$ and that the entries of $\boldsymbol{\epsilon}$ are independent and non-Gaussian and, by assumption 5, the columns of $B(I_q - \Lambda)^{-1}$ are linearly independent. Hence, we can use ICA to recover $B(I_q - \Lambda)^{-1}$ up to column permutation and scaling. We proceed

to outline how such recovery may be achieved using tensor decomposition on the third-order cumulant of $\mathbf{X}$.

3.1.2 Product recovery using tensor decomposition

We use symmetric tensor decomposition to decompose the third-order cumulant of $\mathbf{X}$ and recover $B(I_q - \Lambda)^{-1}(\text{diag}(\Omega^{[3]}))^{1/3}$ up to permutation of the columns. Since we have p observed variables and q latent variables, the third-order cumulant of $\mathbf{X}$ is a symmetric $p \times p \times p$ tensor of rank q. We have that

$$\begin{aligned}
\kappa_3(\mathbf{X}) &= \mathbb{E}[(\mathbf{X} - E(\mathbf{X}))^{\otimes 3}] \\
&= [B(I_q - \Lambda)^{-1}] \cdot \Omega^{[3]} \\
&= \sum_{l=1}^{q} \left(\sqrt[3]{w_l} [B(I_q - \Lambda)^{-1}]_l \right)^{\otimes 3} \\
&= \sum_{l=1}^{q} w_l \left([B(I_q - \Lambda)^{-1}]_l \right)^{\otimes 3}.
\end{aligned} \tag{3.2}$$

The columns of $B(I_q - \Lambda)^{-1}$ are linearly independent by assumption 5, so Theorem 2.1 guarantees that the decomposition of $\kappa_3(\mathbf{X})$ will be unique up to permutation and scaling. Note that that is the best we can hope for. The permutation indeterminacy is inevitable as reordering the summands yields the same decomposition, while the scaling indeterminacy cannot be avoided either because we may rewrite $w_l \left([B(I_q - \Lambda)^{-1}]_l \right)^{\otimes 3}$ as $\frac{w_l}{\mu} \left(\sqrt[3]{\mu} \cdot [B(I_q - \Lambda)^{-1}]_l \right)^{\otimes 3}$ for any $\mu \neq 0$ and still obtain the same decomposition.

Using Algorithm 1, we can recover the vectors up to norm one, that is, adjusting the scaling appropriately so that $\kappa_3(\mathbf{X}) = \sum_{l=1}^{q} \alpha_l (v_l)^{\otimes 3}$ with $\|v_l\|^2 = 1$, Algorithm 1 allows us to recover the vectors v_l. We note that in this step we also learn the latent dimension: q equals the rank of $\kappa_3(\mathbf{X})$ which, using the notation from Algorithm 1, is precisely the number of nonzero eigenvalues of $M_a(M_b)^\dagger$. We may therefore assume without loss of generality that q is known. To find the vector of coefficients

$\alpha = (\alpha_1, \ldots, \alpha_q)$, we solve the least squares problem

$$\min_{\alpha_1,\ldots,\alpha_q} \|\kappa_3(\mathbf{X}) - \sum_{l=1}^{q} \alpha_l (v_l)^{\otimes 3}\|. \tag{3.3}$$

Note that this implies that we can recover the product $B(I_q - \Lambda)^{-1}(\mathrm{diag}(\Omega^{[3]}))^{1/3}$ up to permutation, by absorbing each coefficient into its respective vector: $\kappa_3(\mathbf{X}) = \sum_{l=1}^{q} \alpha_l (v_l)^{\otimes 3} = \sum_{l=1}^{q} (\sqrt[3]{\alpha_l} v_l)^{\otimes 3}$. Thus, we can conclude that from the third-order cumulant of $\mathbf{X}$ we can recover the product $B(I_q - \Lambda)^{-1}(\mathrm{diag}(\Omega^{[3]}))^{1/3} P$, where P is a permutation matrix.

We assume without loss of generality that $P = I_q$ in the observational context. This holds without loss of generality because it is a labeling of the nodes in the latent graph. From here onwards, we use this labeling, that is, Z_m is the m^{th} latent node as given by the permutation in the observational context, and we recover B and Λ up to this permutation. We now prove some facts about the entries of $B(I_q - \Lambda)^{-1}$ that will be useful in future sections.

Claim 3.1. *Assume that $i \neq j$. Then the i, j entry of $(I_q - \Lambda)^{-1}$ equals the sum of the paths from Z_j to Z_i in our graph G, where a path $Z_j \to Z_{k_1} \to \ldots \to Z_{k_m} \to Z_i$ is represented by the product of its corresponding entries in $\Lambda : \lambda_{ik_m} \cdot \lambda_{k_m k_{m-1}} \cdot \ldots \cdot \lambda_{k_1 j}$.*

Proof. We prove by induction that the i, j entry of Λ^r contains the sum of paths from Z_j to Z_i in G of length exactly r (if any such path exists). The base case with $r = 1$ holds by definition of Λ. Assume that the proposition holds for Λ^n. We have that

$$\Lambda^{n+1}_{ij} = \sum_{k=1}^{q} \Lambda^n_{ik} \cdot \Lambda_{kj}. \tag{3.4}$$

By our inductive hypothesis, Λ^n_{ik} contains the sum of paths from Z_k to Z_i in G of length exactly n. We have that $\Lambda_{kj} \neq 0$ if and only if $j \in \mathrm{pa}(k)$ in G, in which case multiplying each path from Z_k to Z_i of length exactly n by Λ_{kj} will yield a path from Z_j to Z_i of length exactly $n + 1$, as desired.

We now show that

$$(I_q - \Lambda)^{-1} = I_q + \sum_{n=1}^{q-1} \Lambda^n. \tag{3.5}$$

This follows because

$$\begin{aligned}(I_q - \Lambda)(I_q + \sum_{n=1}^{q-1} \Lambda^n) &= (I_q + \sum_{n=1}^{q-1} \Lambda^n)(I_q - \Lambda) \\ &= I_q - \sum_{n=1}^{q-1} (\Lambda^n - \Lambda^n) - \Lambda^q \\ &= I_q,\end{aligned} \tag{3.6}$$

where $\Lambda^q = \mathbf{0}_{q \times q}$ because the maximum length of any path in a graph with q nodes is $q - 1$. We can thus conclude that, if $i \neq j$

$$(I_q - \Lambda)^{-1}_{ij} = \sum_{n=1}^{q-1} \Lambda^n_{ij}, \tag{3.7}$$

which is precisely the sum of the paths from Z_j to Z_i in G. □

Corollary 3.2. *The i, j entry of $B(I_q - \Lambda)^{-1}$ equals the sum of the paths from Z_j to X_i in our graph G, where a path $Z_j \to Z_{k_1} \to \ldots \to Z_{k_m} \to X_i$ is represented by the product of its corresponding entries in Λ and B: $b_{ik_m} \cdot \lambda_{k_m k_{m-1}} \cdot \ldots \cdot \lambda_{k_1 j}$.*

3.1.3 Necessity of interventional data

Access to the product $B(I_q - \Lambda)^{-1}(\text{diag}(\Omega^{[3]}))^{1/3}$ alone is not sufficient to recover B and Λ. To see this, let $\hat{\Lambda} = \mathbf{0}_{q \times q}$, $\hat{H} = M(I_q - \Lambda)H$, where M is a diagonal matrix with positive entries making $\hat{H}$ satisfy assumption 4, and $\hat{\Omega} = (\text{diag}(\Omega^{[3]}))^{-1/3} M^{-1}$. Then $\hat{\Omega}(I_q - \hat{\Lambda})\hat{H} = (\text{diag}(\Omega^{[3]}))^{-1/3}(I_q - \Lambda)H$. Thus, $\hat{\Lambda}$ and $\hat{H}$ are solutions to the causal disentanglement problem. Since $\hat{\Lambda}$ is the zero matrix, this implies that a solution with independent latent variables will always be consistent with $B(I_q - \Lambda)^{-1}(\text{diag}(\Omega^{[3]}))^{1/3}$. In addition, having access to a finite number of higher-order cumulants of $\mathbf{X}$ does not improve the situation.

Theorem 3.3. *It is not possible to separate B and Λ beyond the recovery of $B(I_q - \Lambda)^{-1}$ using finitely many cumulants of $\mathbf{X}$.*

Proof. We consider the expressions for the different cumulants of $\mathbf{X}$. We have that

$$\begin{aligned}
\kappa_1(\mathbf{X}) &= \mathbb{E}[\mathbf{X}] = B(I_q - \Lambda)^{-1}\Omega^{[1]}, \\
\kappa_2(\mathbf{X}) &= \mathbb{E}[(\mathbf{X} - \mathbb{E}[\mathbf{X}])^{\otimes^2}] = [B(I_q - \Lambda)^{-1}] \cdot \Omega^{[2]}, \\
\kappa_3(\mathbf{X}) &= \mathbb{E}[(\mathbf{X} - \mathbb{E}[\mathbf{X}])^{\otimes^3}] = [B(I_q - \Lambda)^{-1}] \cdot \Omega^{[3]}, \\
\kappa_4(\mathbf{X}) &= [B(I_q - \Lambda)^{-1}] \cdot \Omega^{[4]}, \\
&\vdots \\
\kappa_n(\mathbf{X}) &= [B(I_q - \Lambda)^{-1}] \cdot \Omega^{[n]}.
\end{aligned} \tag{3.8}$$

We can see that, regardless of the order of the cumulant tensor we consider, we only have access to the product $B(I_q - \Lambda)^{-1}$. Therefore, that is the best we can hope to recover. □

In order to recover B and Λ we will use interventional data. Recall our definition of interventions from Section 1.1: given a variable Z_k, a *perfect* intervention at Z_k removes the causal dependencies of Z_k on its parents and changes its stochasticity, while a *soft* intervention modifies the magnitudes of the causal dependencies of Z_k on its parents and changes its stochasticity. We now characterize what this means for our model parameters.

Definition 3.4 (Interventions)**.** A *perfect intervention* at Z_k sets the k^{th} row of Λ to 0, while a *soft intervention* modifies all nonzero entries of the k^{th} row of Λ. Both modify the distribution of ϵ_k, inducing a change in the value of $\kappa_3(\epsilon_k) = \operatorname{diag}(\Omega^{[3]})_{kk}$.

From here onwards we assume that we can observe $\mathbf{X}$ in various contexts, where a context is either observational or corresponds to an intervention at Z_k. We assume that every latent variable is intervened at and we can identify the observational context, but the intervention target of each interventional context is unknown. Given a tensor T in the observational context, we let $T^{(k)}$ denote such tensor after an intervention at Z_k and, to simplify notation, we let $D = \operatorname{diag}(\Omega^{[3]})^{1/3}$. Using tensor

decomposition, we can recover the products $B(I_q - \Lambda)^{-1}D$ and $B(I_q - \Lambda^{(k)})D^{(k)}P^{(k)}$ for all intervened variables Z_k. Our goal is to use this collection of closely-related matrices to find B and Λ. We note that the permutation matrices $P^{(k)}$ encode a relabeling of the nodes in the latent graph, arising from the permutation indeterminacy of tensor decomposition: $P^{(k)}_{ij} = 1$ indicates that node Z_i has been relabeled as node Z_j in the interventional context with intervention target Z_k.

3.2 Recovery of model parameters

In order to use the collection of closely-related matrices obtained from $\kappa_3(\mathbf{X})$ to recover B and Λ, we must first match each interventional context with its corresponding intervention target. That is, having recovered $B(I_q - \Lambda)^{-1}D$ from the third-order cumulant of $\mathbf{X}$ in the observational context, and $B(I_q - \tilde{\Lambda})^{-1}\tilde{D}\tilde{P}$ from the third-order cumulant in some interventional context, we want to find $k \in [q]$ such that $\tilde{\Lambda} = \Lambda^{(k)}$, $\tilde{D} = \tilde{D}^{(k)}$ and $\tilde{P} = \tilde{P}^{(k)}$. From here onwards, we let C denote the Moore-Penrose inverse of the product recovered in the observational context, and we use $C'^{(k)}$ to designate the result of reordering the rows of $C^{(k)}$ to match the labeling of the latent nodes given by the permutation in the observational context:

$$\begin{aligned} C &= (B(I_q - \Lambda)^{-1}D)^\dagger = D^{-1}(I_q - \Lambda)H, \\ C^{(k)} &= (B(I_q - \Lambda^{(k)})^{-1}D^{(k)}P^{(k)})^\dagger = (P^{(k)})^T(D^{(k)})^{-1}(I_q - \Lambda^{(k)})H, \\ C'^{(k)} &= P^{(k)}C^{(k)} = (D^{(k)})^{-1}(I_q - \Lambda^{(k)})H. \end{aligned} \tag{3.9}$$

The facts we present below underpin our algorithm to recover the intervention targets. They do not depend on the type of interventions considered, and as a consequence neither does the correctness of the algorithm.

Remark 3.5. *Claims 3.6 and 3.7, and Propositions 3.8, 3.9 and 3.10 hold regardless of whether the intervention at Z_k is perfect or soft.*

Claim 3.6. $\Lambda_{i:} = \Lambda^{(k)}_{i:}$ *for all* $i \neq k$.

Proof. By Definition 3.4, $\lambda_{kj} \neq \lambda^{(k)}_{kj}$ for all nonzero λ_{kj}. Entries of Λ not of this form remain unchanged under an intervention at Z_k, so $\Lambda_{i:} = \Lambda^{(k)}_{i:}$ for all $i \neq k$. □

Claim 3.7. *$D_{ii} \neq D_{ii}^{(k)}$ if and only if $i = k$.*

Proof. Recall that $D = (\mathrm{diag}(\Omega^{[3]}))^{1/3}$, that is, $D_{ii} = \kappa_3(\epsilon_i)^{1/3}$. The claim now follows from Definition 3.4 and from the fact that an intervention at Z_k does not affect the stochasticity of latent variables other than Z_k. □

Proposition 3.8. *Let $\sigma^{(k)}$ be the permutation associated to $P^{(k)}$. Then $\boldsymbol{c}_i \neq \boldsymbol{c}^{(k)}_{\sigma^{(k)}(i)}$ if and only if $i = k$.*

Proof. We have that

$$\begin{aligned} \mathbf{c}_i &= \sum_{j=1}^{q} \frac{1}{D_{ii}}(I_q - \Lambda)_{ij}\mathbf{h}_j = \frac{1}{D_{ii}}(\mathbf{h}_i - \sum_{j \in \mathrm{pa}(i)} \lambda_{ij}\mathbf{h}_j), \\ \mathbf{c}^{(k)}_{\sigma^{(k)}(i)} &= \sum_{j=1}^{q} \frac{1}{D^{(k)}_{ii}}(I_q - \Lambda^{(k)})_{ij}\mathbf{h}_j = \frac{1}{D^{(k)}_{ii}}(\mathbf{h}_i - \sum_{j \in \mathrm{pa}(i)} \lambda^{(k)}_{ij}\mathbf{h}_j). \end{aligned} \tag{3.10}$$

First assume that $\mathbf{c}_i \neq \mathbf{c}^{(k)}_{\sigma^{(k)}(i)}$. Then we must have that $D_{ii} \neq D^{(k)}_{ii}$ or $\Lambda_{i:} \neq \Lambda^{(k)}_{i:}$. It follows by Claims 3.6 and 3.7 that $i = k$. Similarly, assume instead that $i = k$. By Claim 3.7, $D_{ii} \neq D^{(k)}_{ii}$, so $\mathbf{c}_i \neq \mathbf{c}^{(k)}_{\sigma^{(k)}(i)}$. Note that assumption 5 guarantees that the difference between $\mathbf{c}_i$ and $\mathbf{c}^{(k)}_{\sigma^{(k)}(i)}$ due to the difference between D_{ii} and $D^{(k)}_{ii}$ is not offset by the potential difference between $\Lambda_{i:}$ and $\Lambda^{(k)}_{i:}$. □

We now present the algorithm to recover the intervention targets.

Algorithm 2 Recover intervention target (recover_int)

Input: C and $\tilde{C} = \tilde{P}^T \tilde{D}^{-1}(I_q - \tilde{\Lambda})H$, the Moore-Penrose inverses of the products recovered via tensor decomposition of $\kappa_3(\mathbf{X})$ in the observational context and in an interventional context with unknown intervention target, respectively.

Output: (k, j) such that $\tilde{C} = C^{(k)}$ and $\tilde{P}_{kj} = 1$. k is the intervention target of the context where $\tilde{C}^\dagger$ was recovered. $\tilde{P}$ encodes a relabeling of the nodes in the latent graph, arising from the permutation indeterminacy of tensor decomposition. $\tilde{P}_{kj} = 1$ indicates that node Z_k has been relabeled as Z_j in the interventional context where $\tilde{C}^\dagger$ was recovered.

$q \leftarrow$number of rows of C
$\text{matched}_{\text{obs}} \leftarrow \text{set}()$
$\text{matched}_{\text{int}} \leftarrow \text{set}()$
for $i = 0$ to $q - 1$ **do**
    if $\mathbf{c}_i$ has matching row in $\tilde{C}$ **then**
        $m \leftarrow$ index of matching row
        Add i to $\text{matched}_{\text{obs}}$
        Add m to $\text{matched}_{\text{int}}$
    end if
end for
$k \leftarrow [q] \backslash \text{matched}_{\text{obs}}$
$j \leftarrow [q] \backslash \text{matched}_{\text{int}}$
return (k, j)

Proposition 3.9. *Given C and $\tilde{C} = \tilde{P}^T \tilde{D}^{-1}(I_q - \tilde{\Lambda})H$ as input, the output of Algorithm 2 is the value of (k, j) such that $\tilde{C} = C^{(k)}$ and $\tilde{P}_{kj} = 1$.*

Proof. Let (k, j) be the output of Algorithm 2, $\tilde{\sigma}$ be the permutation associated to $\tilde{P}$, and n be such that $\tilde{\Lambda} = \Lambda^{(n)}$, $\tilde{D} = D^{(n)}$, and $\tilde{P} = P^{(n)}$. We wish to show that $n = k$ and $P^{(n)}_{nj} = 1$. By Proposition 3.8, we have that every row in C has a matching row in $\tilde{C}$ except for $\mathbf{c}_n$. Thus, at the end of the `for` loop, $[q] \backslash \text{matched}_{\text{obs}} = n$ and $[q] \backslash \text{matched}_{\text{int}} = \tilde{\sigma}(n)$. It follows that $n = k$ and $\tilde{P}_{nj} = 1$. □

Even with the intervention targets identified, using the products recovered in the different contexts to determine the values of B and Λ remains challenging if the labeling of nodes in the latent graph is not consistent across contexts. To address this, it suffices to recover the permutation matrices $\{P^{(k)}\}_{k\in[q]}$. We now present an algorithm to do so.

Algorithm 3 Recover permutation matrix (recover_perm)

Input: C and $C^{(k)}$. C is as in Algorithm 2. $C^{(k)}$ is the Moore-Penrose inverse of the product recovered via tensor decomposition of $\kappa_3(\mathbf{X})$ in the interventional context with intervention target Z_k.
Output: $P^{(k)}$, the permutation matrix encoding the relabeling of the latent nodes in the interventional context corresponding to an intervention at Z_k.

$q \leftarrow$number of rows of C
$P \leftarrow \mathbf{0}_{q\times q}$
$(a, b) \leftarrow$ recover_int$(C, C^{(k)})$
$P[a, b] \leftarrow 1$
for $i = 0$ to $q - 1$ **do**
    if $i = a$ **then**
        continue
    else
        $j \leftarrow$ index of matching row in $C^{(k)}$ to $\mathbf{c}_i$
        $P[i, j] \leftarrow 1$
    end if
end for
return P

Proposition 3.10. *Given C and $C^{(k)}$ as input, the output of Algorithm 3 is $P^{(k)}$.*

Proof. Let P be the output of Algorithm 3. We wish to show that $P = P^{(k)}$. By Proposition 3.9 we have that $\mathbf{p}_k = \mathbf{p}_k^{(k)}$. It remains to show that $\mathbf{p}_i = \mathbf{p}_i^{(k)}$ for all $i \neq k$. Let $i \in [q]$ with $i \neq k$. By Proposition 3.8, we have that $\mathbf{c}_i = \mathbf{c}^{(k)}_{\sigma^{(k)}(i)}$ and, by

assumption 5, $\mathbf{c}_i \neq \mathbf{c}_l^{(k)}$ for all $l \neq \sigma^{(k)}(i)$. The index of the row of $C^{(k)}$ matching $\mathbf{c}_i$, denoted by j in Algorithm 3, is thus equal to $\sigma^{(k)}(i)$. Since $\sigma^{(k)}(i) = j$, $P_{ij}^{(k)} = 1$, so $\mathbf{p}_i = \mathbf{p}_i^{(k)}$. □

3.2.1 Perfect interventions

Sufficiency

The goal of this section is to prove the following theorem:

Theorem 3.11. *One perfect intervention per latent variable is sufficient to recover B and Λ from $\kappa_3(\mathbf{X})$.*

The proof is constructive: we present algorithms to recover B and Λ from $\kappa_3(\mathbf{X})$. Having performed tensor decomposition to recover our products of interest, matched each context with its intervention target, and recovered the corresponding permutation matrices by Algorithms 2 and 3, we have access to $B(I_q - \Lambda)^{-1}D$ and $B(I_q - \Lambda^{(k)})^{-1}D^{(k)}$ for all $k \in [q]$, and thus to C and $C'^{(k)}$ for all $k \in [q]$ too. We now present an algorithm to recover H from these products. This suffices to recover B as, by definition, $B = H^\dagger$.

Algorithm 4 Recover mixing matrix (recover_H)

Input: $C'^{(k)}$ for all $k \in [q]$, the Moore-Penrose inverses of the products recovered in the interventional contexts after undoing the relabeling of the latent nodes resulting from tensor decomposition.
Output: H, the Moore-Penrose inverse of the mixing matrix B.

$q \leftarrow$ number of rows of $C'^{(0)}$
$p \leftarrow$ number of columns of $C'^{(0)}$
$\hat{H} \leftarrow \mathbf{0}_{q \times p}$
for $i = 0$ to $q - 1$ **do**
    $\hat{\mathbf{h}}_i = \text{normalize}(\mathbf{c}_i'^{(i)})$
    $idx \leftarrow$ index of leftmost element of greatest absolute value in $\hat{\mathbf{h}}_i$
    if $\hat{\mathbf{h}}_i[idx] < 0$ **then**
        $\hat{\mathbf{h}}_i = (-1) \cdot \hat{\mathbf{h}}_i$
    end if
end for
return $\hat{H}$

Proposition 3.12. *Given $C'^{(k)}$ for all $k \in [q]$ as input, the output of Algorithm 4 is H.*

Proof. Let $\hat{H}$ be the output of Algorithm 4. We wish to show that $\hat{\mathbf{h}}_i = \mathbf{h}_i$ for all $i \in [q]$. Let $i \in [q]$. We have that

$$\mathbf{c}_i'^{(i)} = \sum_{j=1}^{q} \frac{1}{D_{ii}^{(i)}} (I_q - \Lambda^{(i)})_{ij} \mathbf{h}_j = \frac{1}{D_{ii}^{(i)}} (\mathbf{h}_i - \sum_{j \in \text{pa}(i)} \lambda_{ij}^{(i)} \mathbf{h}_j). \tag{3.11}$$

Recall that, by Definition 3.4, $\lambda_{ij}^{(i)} = 0$ for all $j \in [q]$. We can thus simplify the above expression and obtain

$$\mathbf{c}_i'^{(i)} = \frac{1}{D_{ii}^{(i)}} \mathbf{h}_i. \tag{3.12}$$

By assumption 4, $\mathbf{h}_i$ has norm 1 and satisfies that its leftmost entry with largest

absolute value is positive. Thus, by normalizing and adjusting the sign appropriately, we get that $\hat{\mathbf{h}}_i = \mathbf{h}_i$, as desired. □

Finally, we present an algorithm to recover Λ, assuming access to H, C and $C'^{(k)}$ for all $k \in [q]$. Recall from equation (3.10) that $\mathbf{c}_i$ is a linear combination of $\{\mathbf{h}_i, \mathbf{h}_j : j \in \text{pa}(i)\}$, and the coefficients of $\{\mathbf{h}_j : j \in \text{pa}(i)\}$ in this linear combination are scalar multiples of entries of Λ:

$$\mathbf{c}_i = \frac{1}{D_{ii}}(\mathbf{h}_i - \sum_{j \in \text{pa}(i)} \lambda_{ij}\mathbf{h}_j). \tag{3.13}$$

The algorithm leverages this fact, iteratively recovering the entries of Λ by first identifying source nodes in the latent graph and proceeding in a breadth-first manner [15].

Algorithm 5 Recover the latent graph (recover_Λ)

Input: C, $C'^{(k)}$ for all $k \in [q]$ and H. C is as in Algorithms 2 and 3, and $C'^{(k)}$ and H are as in Algorithm 5.
Output: Λ, the matrix encoding the latent graph.

$q \leftarrow$ number of rows in C
nodes $\leftarrow$set($[q]$)
$V_0 \leftarrow [\,]$
$\hat{\Lambda} \leftarrow \mathbf{0}_{q \times q}$
for $i = 0$ to $q - 1$ **do**
    if $\mathbf{c}_i \parallel \mathbf{c}_i'^{(i)}$ **then**
        Add i to V_0
        Remove i from nodes
    end if
end for
for $j = 1$ to $q - 1$ **do**
    if nodes $\neq \emptyset$ **then**
        $V_j = V_{j-1}$
    else
        break
    end if
    for $n \in$ nodes **do**
        if $\mathbf{c}_n \in \text{span}\{\mathbf{h}_n, \mathbf{h}_m : m \in V_{j-1}\}$ **then**
            Add n to V_j
            Remove n from nodes
            Solve for $\gamma_n, \{\gamma_m\}_{m \in V_{j-1}}$ in the system $\mathbf{c}_n = \gamma_n \mathbf{h}_n - \sum_{m \in V_{j-1}} \gamma_m \mathbf{h}_m$
            $\hat{\Lambda}[n, m] \leftarrow \frac{\gamma_m}{\gamma_n}$ for all γ_m in $\{\gamma_m\}_{m \in V_{j-1}}$
        end if
    end for
end for
return $\hat{\Lambda}$

To prove the correctness of the algorithm, we first prove the following lemma:

Lemma 3.13. *Let level(i) denote the minimum number of steps required to reach Z_i from a source node in the latent graph, and V_j be as in Algorithm 5. Then for all $j \in [q]$, $V_j = \{i \mid \textit{level}(i) \leq j\}$.*

Proof. We proceed by induction. First consider the case $j = 0$. By construction, $V_0 = \{i \mid \mathbf{c}_i \parallel \mathbf{c}_i'^{(i)}\}$. Recall from equations (3.10) and (3.12) that

$$\begin{aligned} \mathbf{c}_i &= \frac{1}{D_{ii}}(\mathbf{h}_i - \sum_{m \in \mathrm{pa}(i)} \lambda_{im}\mathbf{h}_m), \\ \mathbf{c}_i'^{(i)} &= \frac{1}{D_{ii}^{(i)}}\mathbf{h}_i. \end{aligned} \tag{3.14}$$

Thus, $\mathbf{c}_i \parallel \mathbf{c}_i'^{(i)} \Leftrightarrow \mathrm{pa}(i) = \emptyset \Leftrightarrow \mathrm{level}(i) = 0$, as desired. Now assume that the proposition holds for $j = t$, that is, $V_t = \{i \mid \mathrm{level}(i) \leq t\}$. By construction,

$$V_{t+1} = \{i \mid \mathbf{c}_i \in \mathrm{span}\{\mathbf{h}_i, \mathbf{h}_t : t \in V_t\}\}. \tag{3.15}$$

Since H is full row rank by assumption 3, we have by equation (3.14) that

$$\mathbf{c}_i \in \mathrm{span}\{\mathbf{h}_i, \mathbf{h}_t : t \in V_t\} \Leftrightarrow \mathrm{pa}(i) \subseteq V_t. \tag{3.16}$$

Thus, $V_{t+1} = \{i \mid \mathrm{pa}(i) \subseteq V_t\} = \{i \mid \mathrm{level}(i) \leq t + 1\}$. □

We can use this fact to prove that the output of Algorithm 5 is Λ.

Proposition 3.14. *Given C, $C'^{(k)}$ for all $k \in [q]$, and H as input, the output of Algorithm 5 is Λ.*

Proof. Let $\hat{\Lambda}$ be the output of Algorithm 5. We wish to show that $\hat{\Lambda} = \Lambda$. Let $\mathrm{level}(i)$ be as in Lemma 3.13. We prove that after t iterations of the `for` loop starting on line 13, $\hat{\Lambda}_{n:} = \Lambda_{n:}$ for all n such that $\mathrm{level}(n) \leq t$. Since $\mathrm{level}(i) \leq q - 1$ for all nodes Z_i in the latent graph, this shows that $\hat{\Lambda} = \Lambda$. We proceed by induction. The case $t = 0$ follows from the fact that $\Lambda_{n:} = \mathbf{0}_{1\times q}$ for all n satisfying

that level$(n) = 0$ and $\hat{\Lambda} = \mathbf{0}_{q\times q}$ before the first iteration of the `for` loop. Now assume that the proposition holds for $t = r$. That is, after r iterations of the `for` loop starting on line 13, $\hat{\Lambda}_{n:} = \Lambda_{n:}$ for all n such that level$(n) \leq r$. Consider iteration $r + 1$. By lines 21 and 22, we have that $n \notin$ nodes for all $n \in V_r$ which, by Lemma 3.13, implies that level$(n) \geq r + 1$ for all $n \in$ nodes. Only rows $\hat{\Lambda}_{n:}$ for $n \in$ nodes can be modified in iteration $r + 1$ of the `for` loop, so it suffices to show that $\hat{\Lambda}_{n:} = \Lambda_{n:}$ for all n such that level$(n) = r + 1$. Let $n \in [q]$ be such that level$(n) = r + 1$. Then, by Lemma 3.13, pa$(n) \subseteq V_r$ so, by equation (3.14), $\mathbf{c}_n \in \text{span}\{\mathbf{h}_n, \mathbf{h}_m : m \in V_r\}$. Since H is full row rank by assumption 3, we have by the same equation that solving the system on line 23 yields $\gamma_n = 1/D_{nn}$, $\gamma_m = \lambda_{nm}/D_{nn}$ for all $m \in$ pa(n), and $\gamma_m = 0$ for all $m \notin$ pa(n). Thus, on line 24 we set $\hat{\Lambda}_{nm} = \lambda_{nm}$ for all $m \in$ pa(n), while leaving $\hat{\Lambda}_{nm} = 0$ otherwise, so $\hat{\Lambda}_{n:} = \Lambda_{n:}$. □

We have presented algorithms to recover B and Λ from $B(I_q - \Lambda)D$ and $B(I_q - \Lambda^{(k)})D^{(k)}P^{(k)}$ for all $k \in [q]$, so Theorem 3.11 holds.

Worst-case necessity

Having shown that one perfect intervention per latent variable suffices for recovery of B and Λ from $\kappa_3(\mathbf{X})$, we now prove that such set of interventions is also *necessary* for recovery. The proof follows the approach in [48].

Theorem 3.15. *Let n denote the number of intervened variables and assume that $n < q$. Then there exist matrices H and Λ and a set K of n distinct intervention targets such that H and Λ are not identifiable up to permutation of the latent nodes.*

Proof. Assume that variable Z_0 is not intervened at. It suffices to find $\hat{H}$, $\hat{D}$, $\{\hat{D}^{(k)}\}_{k\in K}, \hat{\Lambda}$ and $\{\hat{\Lambda}^{(k)}\}_{k\in K}$ such that $\hat{D}^{-1}(I_q - \hat{\Lambda})\hat{H} = D^{-1}(I_q - \Lambda)H$, $(\hat{D}^{(k)})^{-1}(I_q - \hat{\Lambda}^{(k)})\hat{H} = (D^{(k)})^{-1}(I_q - \Lambda^{(k)})H$ for all $k \in K$ and such that there is no

permutation matrix P and diagonal matrix S for which $PS\hat{H} = H$. Let

$$\hat{D} = \begin{bmatrix} \leftarrow e_1 \rightarrow \\ D_{1:q,0:q} \end{bmatrix}, \qquad \hat{D}^{(k)} = \begin{bmatrix} \leftarrow e_1 \rightarrow \\ D^{(k)}_{1:q,0:q} \end{bmatrix},$$

$$\hat{\Lambda} = \begin{bmatrix} \mathbf{0}_{1\times q} \\ \Lambda_{1:q,0:q} \end{bmatrix}, \qquad \hat{\Lambda}^{(k)} = \begin{bmatrix} \mathbf{0}_{1\times q} \\ \Lambda^{(k)}_{1:q,0:q} \end{bmatrix}, \tag{3.17}$$

$$\hat{H} = \begin{bmatrix} (D^{-1}(I_q - \Lambda))_{0:}H \\ H_{1:q,0:p} \end{bmatrix}.$$

Then $\hat{D}^{-1}(I_q - \hat{\Lambda})\hat{H} = D^{-1}(I_q - \Lambda)H$ and
$(\hat{D}^{(k)})^{-1}(I_q - \hat{\Lambda}^{(k)})\hat{H} = (D^{(k)})^{-1}(I_q - \Lambda^{(k)})H$ for all $k \in K$. Suppose that Z_0 has at least one parent, that is, $\exists j > 0$ such that $\Lambda_{0j} \neq 0$ and thus $(D^{-1}(I_q - \Lambda))_{0j} \neq 0$ too. Then the first row of $\hat{H}$ is a linear combination of at least two rows of H. Since H is full row rank by assumption 3, $PS\hat{H} \neq H$ for all permutation matrices P and diagonal matrices S. □

3.2.2 Soft interventions

In this section, we prove that the additional information obtained by considering $\kappa_3(\mathbf{X})$ instead of $\kappa_2(\mathbf{X})$ is not enough to offset the loss of inferring power resulting from studying soft interventions instead of perfect ones. We first prove non-identifiability in the case $p = q = 2$, and then use this result to prove non-identifiability for arbitrary p and q in the worst case.

Proposition 3.16. *Let $p = q = 2$. Then H and Λ cannot be recovered from $\kappa_3(\mathbf{X})$ in the general case.*

Proof. Let $p = q = 2$ and

$$H = \begin{bmatrix} h_{00} & h_{01} \\ h_{10} & h_{11} \end{bmatrix}, \quad B = H^{-1}, \quad \Lambda = \Lambda^{(1)} = \begin{bmatrix} 0 & \lambda_{01} \\ 0 & 0 \end{bmatrix}, \quad \Lambda^{(0)} = \begin{bmatrix} 0 & \lambda_{01}^{(0)} \\ 0 & 0 \end{bmatrix},$$
$$D^{-1} = \begin{bmatrix} d_{00} & 0 \\ 0 & d_{11} \end{bmatrix}, \quad (D^{(0)})^{-1} = \begin{bmatrix} d_{00}^{(0)} & 0 \\ 0 & d_{11} \end{bmatrix}, \quad (D^{(1)})^{-1} = \begin{bmatrix} d_{00} & 0 \\ 0 & d_{11}^{(1)} \end{bmatrix}. \tag{3.18}$$

with $\sqrt{h_{00}^2 + h_{01}^2} = \sqrt{h_{10}^2 + h_{11}^2} = 1$. Then from the third-order cumulant of $\mathbf{X}$ we have access to

$$D^{-1}(I_2 - \Lambda)H = \begin{bmatrix} d_{00}h_{00} - d_{00}h_{10}\lambda_{01} & d_{00}h_{01} - d_{00}h_{11}\lambda_{01} \\ d_{11}h_{10} & d_{11}h_{11} \end{bmatrix},$$

$$(D^{(0)})^{-1}(I_2 - \Lambda^{(0)})H = \begin{bmatrix} d_{00}^{(0)}h_{00} - d_{00}^{(0)}h_{10}\lambda_{01}^{(0)} & d_{00}^{(0)}h_{01} - d_{00}^{(0)}h_{11}\lambda_{01}^{(0)} \\ d_{11}h_{10} & d_{11}h_{11} \end{bmatrix}, \tag{3.19}$$

$$(D^{(1)})^{-1}(I_2 - \Lambda^{(1)})H = \begin{bmatrix} d_{00}h_{00} - d_{00}h_{10}\lambda_{01} & d_{00}h_{01} - d_{00}h_{11}\lambda_{01} \\ d_{11}^{(1)}h_{10} & d_{11}^{(1)}h_{11} \end{bmatrix}.$$

For identifiability to hold, we would want the values of the parameters in equation (3.18) to be the only ones giving rise to the products in equation (3.19) (up to permutation of the latent nodes). This is, however, not the case. Computations in `Julia` [6] using `Oscar.jl` [38] show that there is in fact a nine-dimensional family of parameters compatible with assumptions 1-5 giving rise to these same products. □

To prove non-identifiability for arbitrary p and q in the worst case, it suffices to embed this 4-variable configuration into a larger graph.

Theorem 3.17. *One soft intervention per latent node is not sufficient to recover B and Λ from $\kappa_3(\mathbf{X})$ in the worst case.*

Proof. Let $\hat{B}$, $\hat{\Lambda}$, $\{\hat{\Lambda}^{(k)}\}_{k\in[2]}$, $\hat{D}$, $\{\hat{D}^{(k)}\}_{k\in[2]}$ be as in equation (3.18), albeit with the

addition of the hat notation, and consider the model with parameters

$$B = \left[\begin{array}{c|c} \hat{B} & \mathbf{0}_{2\times(q-2)} \\ \hline \mathbf{0}_{(p-2)\times 2} & \begin{array}{c} I_{q-2} \\ \mathbf{0}_{(p-q)\times(q-2)} \end{array} \end{array}\right], \tag{3.20}$$

$$\Lambda^{(0)} = \left[\begin{array}{c|c|c} 0 & \hat{\Lambda}_{01}^{(0)} & \mathbf{0}_{1\times(q-2)} \\ \hline \multicolumn{3}{c}{\mathbf{0}_{(q-1)\times q}} \end{array}\right], \quad \Lambda = \{\Lambda^{(k)}\}_{k\in[q]\setminus\{0\}} = \left[\begin{array}{c|c|c} 0 & \hat{\Lambda}_{01} & \mathbf{0}_{1\times(q-2)} \\ \hline \multicolumn{3}{c}{\mathbf{0}_{(q-1)\times q}} \end{array}\right].$$

Since the values of $\hat{B}$ and $\hat{\Lambda}$ cannot be determined uniquely up to scaling and permutation of the latent nodes, it follows that those of B and Λ cannot either. □

One soft intervention per latent node is, however, sufficient to recover $\overline{G}$. The following section presents a constructive proof. Recall from Section 2.1 that $\overline{G} = (V, \overline{E})$ denotes the transitive closure of the graph $G = (V, E)$, i.e., $(u, v) \in \overline{E}$ for all vertices $u, v \in V$ such that there exists a path from u to v in G.

3.3 Recovery of $\overline{G}$

3.3.1 Perfect interventions

Section 3.2.1 proves that one perfect intervention per latent node is sufficient to recover B and Λ from $\kappa_3(\mathbf{X})$. It thus follows that such set of interventions is sufficient to recover $\overline{G}$ too. We now prove that, in the worst case, recovery of $\overline{G}$ is not possible with fewer perfect interventions.

Theorem 3.18. *One intervention per latent node is necessary to recover $\overline{G}$ from $\kappa_3(\mathbf{X})$.*

Proof. Assume the setup in the example given in the proof of Theorem 3.15. We have that $\mathrm{pa}(0) = \emptyset$ according to $\hat{\Lambda}$, while $\mathrm{pa}(0) \neq \emptyset$ according to Λ. The products $D^{-1}(I_q - \Lambda)H$ and $\{(D^{(k)})^{-1}(I_q - \Lambda^{(k)})H\}_{k\in K}$ are thus consistent with multiple transitive closures. □

3.3.2 Soft interventions

As proved in Section 3.2.2, one soft intervention per latent node is not sufficient to recover B and Λ. Such set of interventions does suffice, however, to recover $\overline{G}$:

Theorem 3.19. *One soft intervention per latent node is sufficient to recover $\overline{G}$ from $\kappa_3(\mathbf{X})$.*

The proof of Theorem 3.19 is constructive and follows a similar approach to [52]. By Remark 3.5, we may assume we have access to $B(I_q - \Lambda)^{-1}D$ and $B(I_q - \Lambda^{(k)})^{-1}D^{(k)}$ for all $k \in [q]$. The algorithm to recover $\overline{G}$ from these products relies on the following fact:

Proposition 3.20. $[B(I_q - \Lambda)^{-1}]_l \neq [B(I_q - \Lambda^{(k)})^{-1}]_l$ *if and only if* $l \in an(k)$.

Proof. First assume that $[B(I_q - \Lambda)^{-1}]_l \neq [B(I_q - \Lambda^{(k)})^{-1}]_l$. By Corollary 3.2, this implies that intervening at Z_k induces a change in a path from Z_l to at least one X_t for some $t \in [p]$ in G. Denote that path by P. Since B remains unchanged under an intervention, P is a product of entries of Λ and B and, by Claim 3.6, $\Lambda_{i:} = \Lambda^{(k)}_{i:}$ for all $i \neq k$, we must have that P contains a factor of the form λ_{kj}. That is, there exists $j \in \mathrm{pa}(k)$ such that there exists a path from Z_l to Z_k in G going through Z_j. The existence of such a path proves that $l \in \mathrm{an}(k)$.

Now assume that $l \in \mathrm{an}(k)$. Then there exists a path P from Z_l to Z_k in G. By Claim 3.1, $(I_q - \Lambda)^{-1}_{kl} \neq 0$. Let Z_j be the parent of Z_k in P. Then λ_{kj} is a factor in P that changes value after an intervention at Z_k, so

$$\begin{aligned}(I_q - \Lambda)^{-1}_{tl} \neq (I_q - \Lambda^{(k)})^{-1}_{tl} &\Leftrightarrow (B(I_q - \Lambda)^{-1})_{tl} \neq (B(I_q - \Lambda^{(k)})^{-1})_{tl} \\ &\Leftrightarrow [B(I_q - \Lambda)^{-1}]_l \neq [B(I_q - \Lambda^{(k)})^{-1}]_l.\end{aligned} \tag{3.21}$$

□

We now present an algorithm to recover $\overline{G}$; we store $\overline{G}$ as a dictionary satisfying that $\overline{G}[m] = \{n \mid n \in \mathrm{an}(m)\}$ for all $m \in [q]$.

Algorithm 6 Recover transitive closure (recover_TS)

Input: $B(I_q - \Lambda)^{-1}D$ and $B(I_q - \Lambda^{(k)})^{-1}D^{(k)}$ for all $k \in [q]$
Output: $\overline{G}$, the transitive closure of the latent graph.

$q \leftarrow$number of columns in $B(I_q - \Lambda)^{-1}D$
$\hat{\overline{G}} = \{\}$
for $k = 0$ to $q - 1$ **do**
    ancestor_set $\leftarrow$ set()
    for $l = 0$ to $q - 1$ **do**
        if $l = k$ **then**
            continue
        else
            if $[B(I_q - \Lambda)^{-1}D]_l \neq [B(I_q - \Lambda^{(k)})^{-1}D^{(k)}]_l$ **then**
                Add l to ancestor_set
            end if
        end if
    end for
    $\hat{\overline{G}}[k]$ = ancestor_set
end for
return $\hat{\overline{G}}$

Proposition 3.21. *Given $B(I_q - \Lambda)^{-1}D$ and $B(I_q - \Lambda^{(k)})^{-1}D^{(k)}$ for all $k \in [q]$ as input, the output of Algorithm 6 is $\overline{G}$.*

Proof. Let $\hat{\overline{G}}$ be the output of Algorithm 6. We wish to show that $\hat{\overline{G}} = \overline{G}$. Let $k \in [q]$. By Proposition 3.20, we have that $[B(I_q - \Lambda)^{-1}]_l \neq [B(I_q - \Lambda^{(k)})^{-1}]_l$ if and only if $l \in \text{an}(k)$. By Claim 3.7, $D_{ii} = D_{ii}^{(k)}$ for all $i \neq k$. Thus, $[B(I_q - \Lambda)^{-1}D]_l \neq [B(I_q - \Lambda^{(k)})^{-1}D^{(k)}]_l$ and $l \neq k$ both hold if and only if $l \in \text{an}(k)$. Since line 12 is run precisely when these conditions are met, we have that $\hat{\overline{G}}[k] = \{l \mid l \in \text{an}(k)\} = \overline{G}[k]$, as desired. □

Finally, we note that the proof of Theorem 3.15 did not assume the type of

interventions considered. Thus, by Theorem 3.18, one soft intervention per latent node is necessary to recover $\overline{G}$ from $\kappa_3(\mathbf{X})$ in the worst case.

4

Computational Results

We evaluate the performance of Algorithms 1-5 on synthetic data. The implementation of the algorithms adapts them to minimize susceptibility to noise, as follows:

- To find a single row that differs between matrices M and M' (up to permutation of the rows), we solve for i in

$$\max_i \min_j \|\mathbf{m}_i - \mathbf{m}'_j\|. \tag{4.1}$$

- To determine whether a vector v_r is in the span of $\{v_1, v_2, \ldots, v_n\}$, we let

$$V = \begin{bmatrix} \uparrow & \uparrow & & \uparrow \\ v_1 & v_2 & \cdots & v_n \\ \downarrow & \downarrow & & \downarrow \end{bmatrix} \tag{4.2}$$

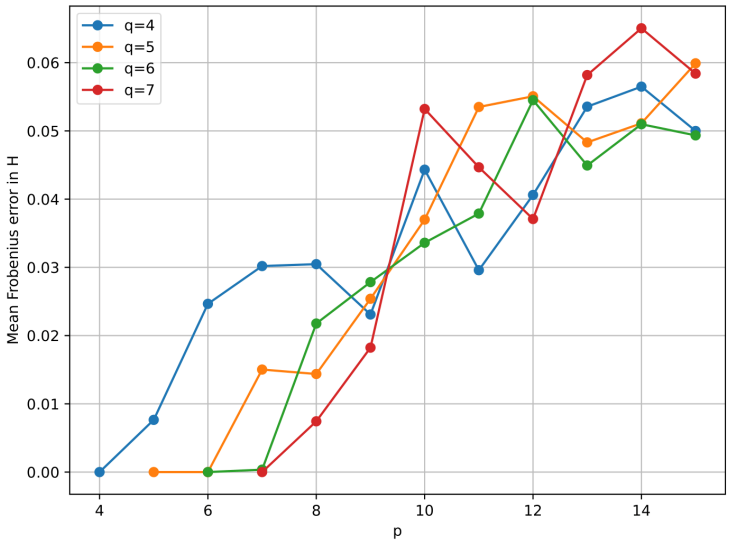
and solve for x in $Vx = v_r$ using ordinary least squares. We consider v_r to be in the span of $\{v_1, \ldots, v_n\}$ if the residuals are below a threshold. We find an appropriate value for the threshold using randomized search.

The adapted algorithms are presented in more detail in Appendix A, and a comprehensive computational complexity analysis is included in Appendix B.

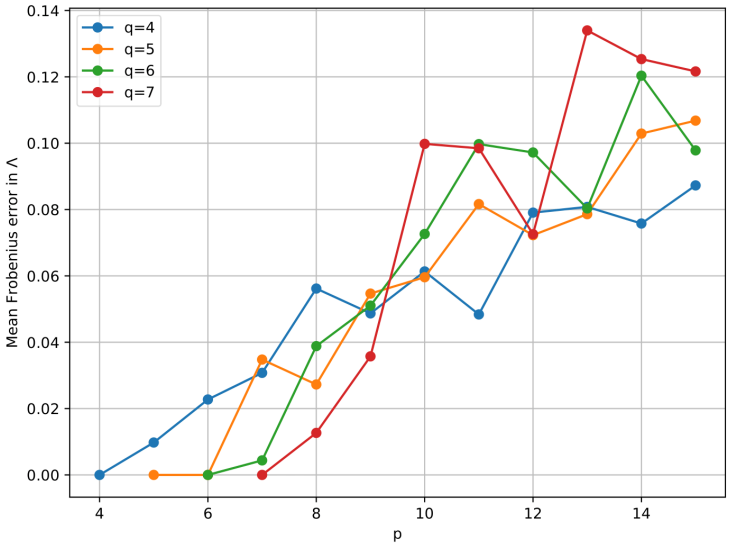
We evaluate the performance of the algorithms using both population and sample cumulants as inputs. We calculate the sample cumulants using k-statistics, and use the Python package `causaldag` [12] to sample our graphs. It implements an extension of the Erdős–Rényi model to DAGs: given a density d, the edge $i \to j$ is added to the graph with probability d and if and only if $i > j$. As in [48], we fix $d = 0.75$, sample the entries of H independently from $\texttt{Unif}([-2, 2])$, and the nonzero entries of Λ independently from $\texttt{Unif}(\pm[0.25, 1])$. We assume that the entries of $\boldsymbol{\epsilon}$ follow an exponential distribution, with rate parameter 1 in the observational context. The rate parameter in an interventional context is sampled from $\texttt{Unif}([0.25, 0.8])$.

4.1 Population cumulants

We consider values of q in $\{4, 5, 6, 7\}$, and let p range from q to 15 for each value of q. We generate 500 models for each value of (q, p) and measure the mean Frobenius error in estimating the parameters H and Λ. The results are presented in Figure 4.1.1. We can see that the algorithm exhibits good performance in recovering the parameters, particularly when $p = q$. This is not surprising, as when $p > q$, the vectors recovered in the tensor decomposition step are the result of calculating the eigendecomposition of a rank-deficient matrix: using the notation from Algorithm 1, we have that $M_a(M_b)^\dagger$ is a $p \times p$ matrix of rank q.



(a) Error in estimating H



(b) Error in estimating Λ

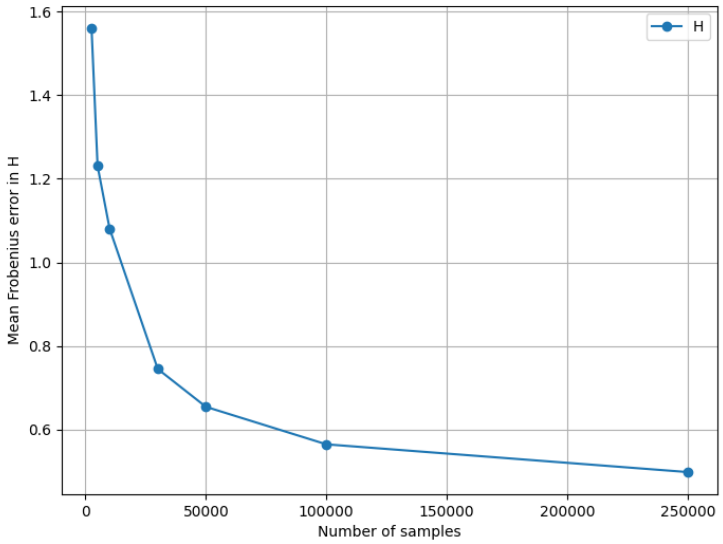
Figure 4.1.1: Parameter estimation error from population cumulants

4.2 Sample cumulants

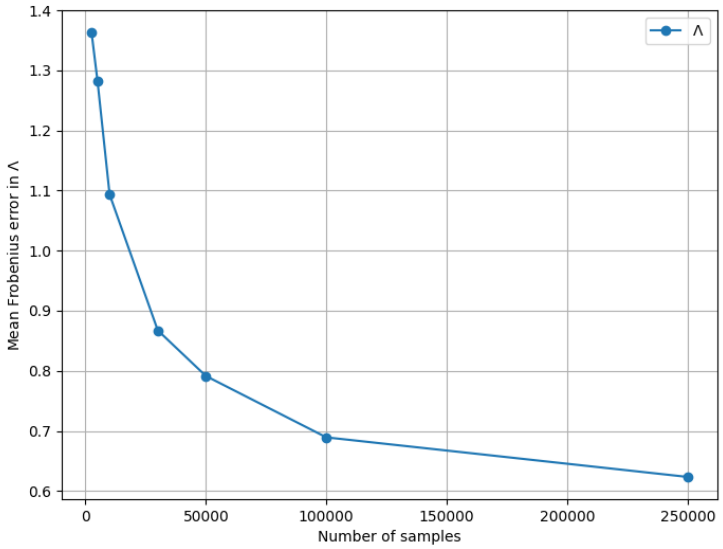
To construct the sample cumulants, we fix $p = 5$ and $q = 4$, and consider sample sizes ranging from 2500 to 250000. As in Section 4.1, we generate 500 models, and measure the mean Frobenius error in estimating the parameters H and Λ for each number of samples. The results are presented in Figure 4.2.1. We find that the method is consistent for recovering H and Λ from noisy data, i.e., the estimates approach the true parameters as the number of samples increases.

We point out that the performance of the model is limited by how well the sample cumulants approximate the population cumulants. That is, if, as the number of samples increases, the difference between the true parameters and the original parameters follows the same trend as the difference between the sample cumulants and the population cumulants, we can assert that the method is doing as well as could be expected. To check whether this is the case, at each number of samples and for each of the 500 models we generate, we calculate the mean Frobenius norm of the difference between the population cumulant and the sample cumulant across contexts. We then take the mean across models and plot that against the mean difference in H and Λ. See Figure 4.2.2.

We posit that a significant part of the error in recovering H is due to assumption 4: normalizing brings the absolute value of the entries in a given row of H closer together; if several entries have an absolute value close to the maximum of the row, noisy data can lead to the wrong entry being deemed as that having the maximum absolute value, potentially causing the row to be recovered with the incorrect sign. Computational results validate this hypothesis: letting $\text{abs}(H)$ denote the matrix obtained by taking the entrywise absolute value of H, we find that the mean Frobenius error in estimating $\text{abs}(H)$ is smaller than that in estimating H. See Figure 4.2.3.

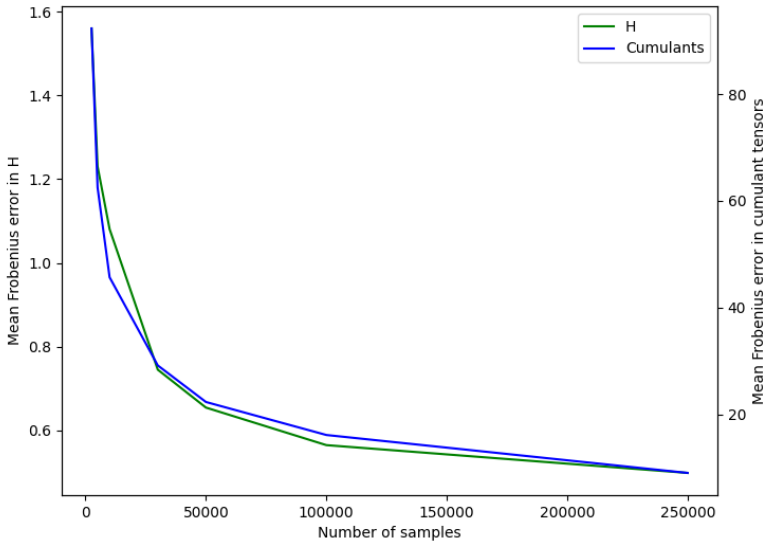


(a) Error in estimating H

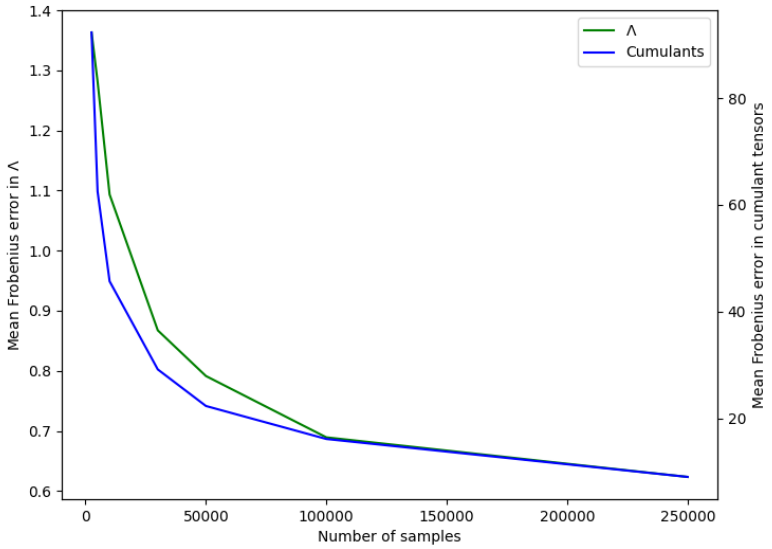


(b) Error in estimating Λ

Figure 4.2.1: Parameter estimation error from sample cumulants



(a) Error in estimating H vs error in estimating population cumulants



(b) Error in estimating Λ vs error in estimating population cumulants

Figure 4.2.2: Parameter vs population cumulants estimation

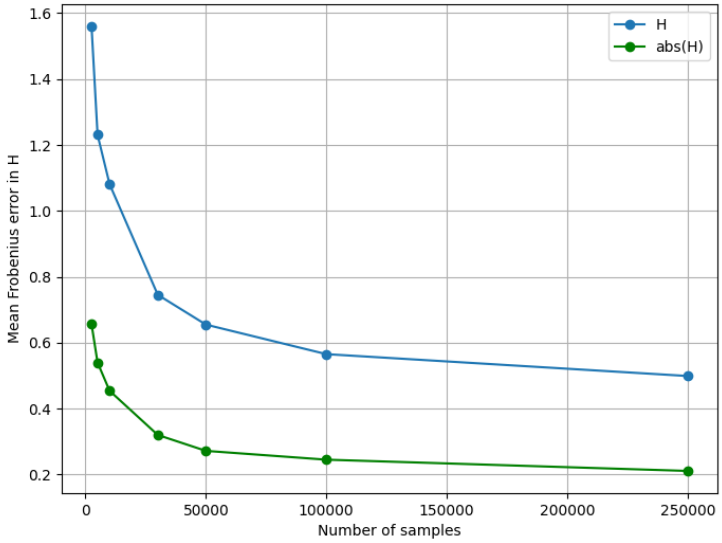


Figure 4.2.3: Error in estimating the mixing matrix from sample cumulants

To test robustness of the method to model misspecification, we consider adding two kinds of nonlinearity:

1. Nonlinearity in the transformation from latent to observed variables, by letting

$$\mathbf{X} = B\mathbf{Z} + \alpha \begin{bmatrix} \mathbf{Z}^2 \\ \mathbf{0}_{(p-q)\times 1} \end{bmatrix}, \quad \alpha \in \mathbb{R}. \tag{4.3}$$

2. Nonlinearity in the latent space, by letting

$$\mathbf{Z} = (I_q - \Lambda)^{-1}\boldsymbol{\epsilon} + \alpha\boldsymbol{\epsilon}^2, \quad \alpha \in \mathbb{R}. \tag{4.4}$$

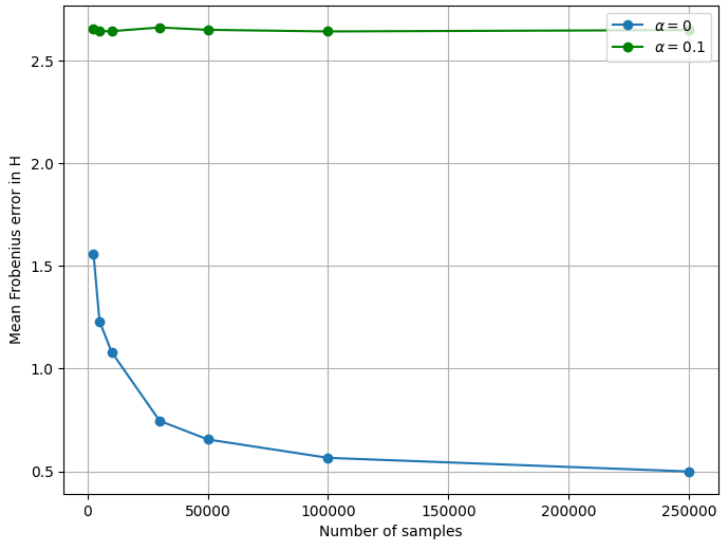
We find that the method is sensitive to nonlinearity in the transformation from latent to observed variables: the errors in estimating B and Λ increase significantly and do not improve with the number of samples for $\alpha = 0.1$. See Figure 4.2.4.

On the other hand, the method demonstrates remarkable robustness to

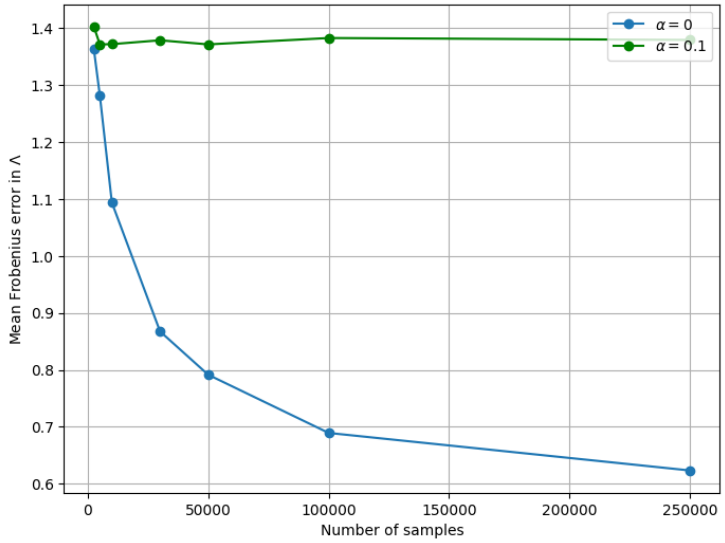
nonlinearity in the latent space. Although the accuracy in recovering Λ decreases considerably for $\alpha = 0.1$, the method remains consistent at estimating B for various values of α. See Figure 4.2.5. It is not surprising that the method does better at recovering B than Λ: despite $\mathbf{Z}$ now being a nonlinear function of $\boldsymbol{\epsilon}$, the linear nature of B's transformation is preserved, as we still have that $\mathbf{X} = B\mathbf{Z}$. Λ, on the other hand, is now involved in the nonlinear relationship introduced by $\boldsymbol{\epsilon}^2$. Finally, we note that the method does better at recovering B as the value of α increases. We hypothesize that this might be the case because, with nonlinearity in the latent space, the third-order cumulant of $\mathbf{X}$ becomes

$$\kappa_3(\mathbf{X}) = B(I - \Lambda)^{-1} \cdot \kappa_3(\boldsymbol{\epsilon}) + \alpha B \cdot \kappa_3(\boldsymbol{\epsilon}^2). \tag{4.5}$$

Unlike in the linear case, the expression for $\kappa_3(\mathbf{X})$ now contains a summand in which B appears isolated from Λ. The larger the value of α, the larger the contribution of this summand to the value of $\kappa_3(\mathbf{X})$. Such larger contribution can in turn make it easier to discern the influence of B on $\kappa_3(\mathbf{X})$, potentially leading to a better recovery.



(a) Error in estimating H



(b) Error in estimating Λ

Figure 4.2.4: Parameter estimation error from sample cumulants with nonlinearity in the transformation from latent to observed variables

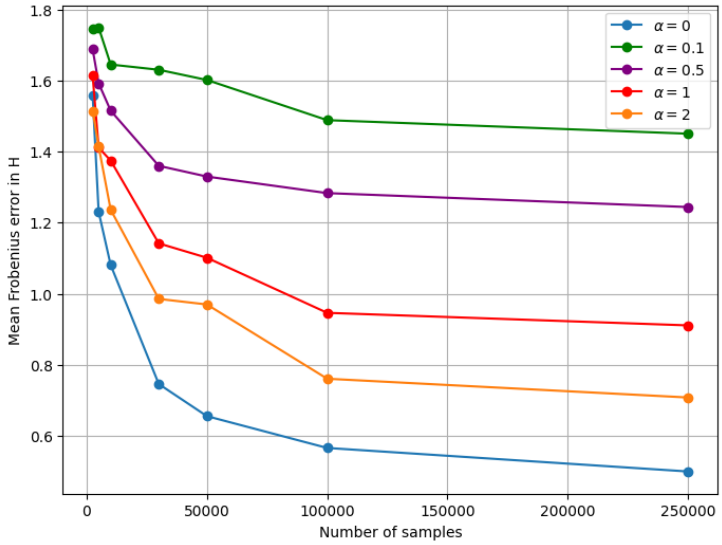


Figure 4.2.5: Error in estimating the mixing matrix from sample cumulants with nonlinearity in the latent space

5

Conclusion and Future Work

In this thesis, we have proved that one perfect intervention per latent variable is sufficient and necessary for the identifiability of linear causal disentanglement. The proof is constructive, yielding an algorithm for causal disentanglement that demonstrates good performance with noisy data, and robustness to model misspecification in the latent space. Finally, we have established the worst-case insufficiency of one soft intervention per latent node for identifiability. We propose several directions for future work.

Higher-dimensional latent space and Gaussian noise

As noted in assumptions 1 and 2, we have restricted our attention to the case $q \leq p$ and the entries of $\boldsymbol{\epsilon}$ being non-Gaussian. These assumptions have made our setup fit well with the identifiability guarantees of classical ICA. Identifiability would still hold if one entry of $\boldsymbol{\epsilon}$ was Gaussian [18], and recent work in the identifiability of

overcomplete ICA [58] suggests that the method could be adapted to the case when $q > p$ by using the subspace power method for tensor decomposition [29]. This would allow our method to handle situations where the latent space is higher-dimensional than the observed space, which arise for instance in image recognition and active shape models [59].

Nonlinear setting

We have demonstrated that the method presents robustness to a certain type of nonlinearity in the latent space, and provided a hypothesis for why this might be the case. Further research is needed to validate it and shed more light on the factors contributing to this performance.

In addition, despite the aforementioned robustness to nonlinearity, the theoretical guarantees of our method only hold when both the transformation from latent to observed variables and the latent model are linear. As mentioned in Section 2.2, current works propose various ways to relax these assumptions. We hope that the results derived in this thesis will help inform further research in this direction.

Partial identifiability with soft interventions

We have shown that one soft intervention per latent variable suffices to recover the transitive closure of the latent DAG but not the model parameters B and Λ in the worst case. In the future, it would be interesting to characterize more precisely what partial identifiability can be recovered from such a set of soft interventions.

A

Adapted Algorithms

Several of the algorithms presented rely on finding matching rows between matrices. Algorithm 7 is a helper function to accomplish this task: given a vector v of size $1 \times n$ and a matrix M of size $m \times n$, it finds the row of M minimizing the Euclidean distance from v. It returns the index of such row, denoted below by match below, and the corresponding Euclidean distance: $\|v - M_{\text{match:}}\|$.

Algorithm 7 Find matching row (match_row)

1: Input: v, M. v is a vector of size $1 \times n$ and M is a matrix of size $m \times n$.
2: Output: $(\arg\min_{i \in [m]} \|v - M_{i:}\|, \min_{i \in [m]} \|v - M_{i:}\|)$
3: $m \leftarrow$ number of rows in M
4: diff $\leftarrow$ max_float
5: match $\leftarrow -1$
6: **for** $i = 0$ to $m - 1$ **do**
7: current_diff $\leftarrow \text{norm}(\mathbf{m}_i - v)$
8: **if** current_diff $<$ diff **then**
9: diff $\leftarrow$ current_diff
10: match $\leftarrow i$
11: **end if**
12: **end for**
13: **return** (match, diff)

Recall that both Algorithm 2 and Algorithm 3 require comparing the rows of C, the Moore-Penrose inverse of the product recovered in the observational context, and those of $\tilde{C}$ or $C^{(k)}$ respectively, the Moore-Penrose inverse of the product recovered in an interventional context. We leverage this commonality and merge them into a single algorithm at the implementation level, Algorithm 8. The inputs to Algorithm 8 are those of Algorithm 2 together with a set which keeps track of the intervention targets that have already been matched with an interventional context. Such set is denoted by $\text{current}_{\text{ints}}$ below. The algorithm proceeds as follows:

1. For each $i \in [q]$, it matches $\mathbf{c}_i$ with $\tilde{\mathbf{c}}_j$ minimizing $\|\mathbf{c}_i - \tilde{\mathbf{c}}_j\|$. This is the `for` loop starting on line 10. $\text{matches}_{\text{int}}$ is a dictionary keeping track of the times each row of $\tilde{C}$ is deemed as a match.

2. By Proposition 3.8, every row of C has a corresponding matching row in $\tilde{C}$ except the row k for which $\tilde{C} = C^{(k)}$. We would thus expect step 1 to correctly

pair up all matching rows. It does not necessarily hold, however, that

$$\arg\min_j \|\mathbf{c}_k - \tilde{\mathbf{c}}_j\| = \tilde{\sigma}(k). \tag{A.1}$$

To account for this, as well for other row mismatches when the cumulants are estimated using fewer samples, the algorithm carries out another round of matching. It uses $\text{matches}_{\text{int}}$ to determine which rows of $\tilde{C}$ have been matched multiple times and which remain unmatched. The set of indices of unmatched rows is denoted by unmatched_rows below. For each j such that $\tilde{\mathbf{c}}_j$ has been matched multiple times, let I denote the set of indices of rows of C that have been matched with $\tilde{\mathbf{c}}_j$ (these are the first elements of the tuples in tuple_list). The algorithm deems $n = \arg\min_{i \in I} \|\mathbf{c}_i - \tilde{\mathbf{c}}_j\|$ as the correct match, and matches each $\mathbf{c}_i$ for $i \in I \backslash [n]$ with $\arg\min_{j \in \text{unmatched\_rows}} \|\mathbf{c}_i - \tilde{\mathbf{c}}_j\|$. We ensure that this step does not result again in multiple rows of C being matched with the same row of $\tilde{C}$ by removing the matched index from unmatched_rows in every iteration of the loop (line 25).

3. After steps 1 and 2, the algorithm has constructed a one-to-one correspondence between rows of C and rows of $\tilde{C}$. Such correspondence is stored in diff_list. We have that for each tuple $(i, j, \text{diff}) \in$ diff_list, i is the index of a row in C, j that of the row of $\tilde{C}$ with which $\mathbf{c}_i$ has been matched, and $\text{diff} = \|\mathbf{c}_i - \tilde{\mathbf{c}}_j\|$. By Proposition 3.8, we expect diff to be very close to 0 for all tuples in diff_list except the one whose first entry is the value of i such that $\tilde{C} = C^{(i)}$ and $\tilde{P} = P^{(i)}$. The algorithm uses this fact to determine such value, by finding the tuple $(i, j, \text{diff}) \in$ diff_list maximizing diff and such that $i \notin \text{current}_{\text{ints}}$. To construct $\tilde{P}$, it leverages the fact that a match between $\mathbf{c}_i$ and $\tilde{\mathbf{c}}_j$ indicates that node Z_i has been relabeled as node Z_j in the interventional context where $\tilde{C}^{\dagger}$ was recovered or, in other words, $\tilde{P}[i, j] = 1$.

Algorithm 8 Recover intervention target - adapted (recover_int_ad)

Input: C, $\tilde{C} = \tilde{P}^T \tilde{D}^{-1}(I_q - \tilde{\Lambda})H$ and current_ints. C and $\tilde{C}$ are as in Algorithm 2; current_ints is a set containing the intervention targets that have already been matched with an interventional context.
Output: $(k, j, P^{(k)})$ such that $\tilde{C} = C^{(k)}$, $\tilde{P} = P^{(k)}$ and $\tilde{P}_{kj} = 1$.

$q \leftarrow$number of rows of C
diff_ list $\leftarrow$ [ ]
$\text{matches}_{\text{int}} \leftarrow \{\}$
$P \leftarrow \mathbf{0}_{q \times q}$
for $k = 0$ to $q - 1$ **do**
    $\text{matches}_{\text{int}}[k] \leftarrow 0$
end for
for $i = 0$ to $q - 1$ **do**
    row_int, diff $\leftarrow$ match_row($\mathbf{c}_i, \tilde{C}$)
    $\text{matches}_{\text{int}}$[row_int] += 1
    Append $(i,$ row_int, diff) to diff_ list
end for
unmatched_rows $\leftarrow$ [key$'$ | (key$'$, value$'$) $\in \text{matches}_{\text{int}} \wedge$ value$' = 0$]
for (key, value) in $\text{matches}_{\text{int}}$ **do**
    if value > 1 **then**
        tuple_list $\leftarrow$ $[(i, j,$ diff$) \in$ diff_list $\mid j =$ key]
        Sort tuple_list by ascending order of the third element in each tuple
        Remove tuple_list[0] from tuple_list
        Remove all elements in tuple_list from diff_list
        for (obs_row_idx, _, _) in tuple_list **do**
            row_int_new, diff_new $\leftarrow$ match_row($\mathbf{c}_{\text{obs\_row\_idx}}, \tilde{C}_{[\text{unmatched\_rows}]:}$)
            new_match $\leftarrow$ (obs_row_idx, unmatched_rows[row_int_new], diff_new)
            Remove unmatched_rows[row_int_new] from unmatched_rows
            Append new_match to diff_list
        end for
    end if
end for

30: **for** $(i, j, \_) \in$ diff_list **do**
31: $\quad P[i, j] \leftarrow 1$
32: **end for**
33: Sort diff_list by descending order of the third element in each tuple
34: **for** tuple in diff_list **do**
35: $\quad$ **if** tuple[0] $\notin$ current_ints **then**
36: $\quad\quad$ **return** (tuple[0], tuple[1], P)
37: $\quad$ **end if**
38: **end for**

Algorithm 4 is implemented as presented. Algorithm 9 is the adapted version of Algorithm 5. As mentioned in Chapter 4, we use ordinary least squares to determine whether a vector v_r is in the span of a set of vectors $\{v_1, v_2, \ldots, v_n\}$, considering v_r to be in such span if the residuals are below a threshold. We use randomized search to find an appropriate value for such threshold, and set it to 10^{-8} when using population cumulants as inputs and to 0.01 in the finite-sample case. We use the least squares approach to determine whether the `if` blocks starting on lines 8 and 20 of Algorithm 5 should be executed. The algorithm undergoes no other changes.

Algorithm 9 Recover the latent graph - adapted (recover_Λ_ad)

Input: C, $C'^{(k)}$ for all $k \in [q]$ and H. C is as in Algorithms 2 and 3, and $C'^{(k)}$ and H are as in Algorithm 5.

Output: Λ, the matrix encoding the latent graph.

$q \leftarrow$ number of rows in C

nodes $\leftarrow$set($[q]$)

$V_0 \leftarrow [\,]$

$\hat{\Lambda} \leftarrow \mathbf{0}_{q \times q}$

for $i = 0$ to $q - 1$ **do**

  sol, residual $\leftarrow$ least_squares($\mathbf{c}_i^T x = (\mathbf{c}_i'^{(i)})^T$)

  if residual $<$ threshold **then**

    Add i to V_0

    Remove i from nodes

  end if

end for

for $j = 1$ to $q - 1$ **do**

  if nodes $\neq \emptyset$ **then**

    $V_j = V_{j-1}$

  else

    break

  end if

  for $n \in$ nodes **do**

    $A \leftarrow \begin{bmatrix} \uparrow & \uparrow & \uparrow & & \uparrow \\ \mathbf{h}_n & \mathbf{h}_{m_1} & \mathbf{h}_{m_2} & \cdots & \mathbf{h}_{m_t} \\ \downarrow & \downarrow & \downarrow & & \downarrow \end{bmatrix}$ where $V_{j-1} = \{m_1, m_2, \ldots, m_t\}$

    sol, residual $\leftarrow$ least_squares($Ax = \mathbf{c}_n^T$)

    if residual $<$ threshold **then**

      Add n to V_j

      Remove n from nodes

      Solve for $\gamma_n, \{\gamma_m\}_{m \in V_{j-1}}$ in the system $\mathbf{c}_n = \gamma_n \mathbf{h}_n - \sum_{m \in V_{j-1}} \gamma_m \mathbf{h}_m$

      $\hat{\Lambda}[n, m] \leftarrow \frac{\gamma_m}{\gamma_n}$ for all γ_m in $\{\gamma_m\}_{m \in V_{j-1}}$

    end if

  end for

end for

return $\hat{\Lambda}$

B

Computational Complexity

We analyze the worst-case time complexity of both the base and adapted algorithms, and use it to determine the overall time complexity of our method for causal disentanglement.

1. Algorithm 1 takes as input a $K \times K \times K$ tensor. Sampling the vectors a and b takes time $O(K)$. Calculating M_a requires performing K^2 multiplications K times and K^2 additions $K-1$ times, so its time complexity is $O(K^3)$. It follows that this is also the time complexity of calculating M_b. The Moore-Penrose inverse of M_b can be calculated in time $O(K^3)$ using singular value decomposition [55], and multiplying M_a times $M_b^\dagger$ using the elementary algorithm for matrix multiplication takes time $O(K^3)$ too [46]. Finally, calculating the eigendecomposition of $M_a(M_b)^\dagger$ also takes time $O(K^3)$ [39]. The overall runtime of Algorithm 1 is thus $O(K^3)$.

2. Algorithm 2 takes as input two $q \times p$ matrices. The bottleneck of the algorithm is the `for` loop starting on line 6, which runs for q iterations. In each iteration we traverse through the rows of $\tilde{C}$, comparing each to $\mathbf{c}_i$. The time complexity of the algorithm is thus $O(q^2 p)$.

3. Algorithm 3 also takes as input two $q \times p$ matrices. It first calls Algorithm 2. The loop starting on line 7 is then run for q iterations. In all iterations except for one, the algorithm traverses through the rows of $C^{(k)}$ and compares each one to $\mathbf{c}_i$. The work done per iteration is thus $O(qp)$, and the overall time complexity of the algorithm $O(q^2 p)$.

4. Algorithm 4 takes as input q matrices of size $q \times p$. It then picks a row of each of those matrices and modifies it to satisfy assumption 4. The time complexity of the algorithm is thus $O(qp)$.

5. Algorithm 5 takes as input $q + 2$ matrices of size $q \times p$. The `for` loop starting on line 7 runs for q iterations. Determining whether $\mathbf{c}_i \parallel \mathbf{c}_{\mathbf{i}}^{\prime(\mathbf{i})}$ requires traversing through the elements of the rows, and thus takes $O(p)$ time. Therefore, the time complexity of this first `for` loop is $O(qp)$. In the worst case, in which the latent graph contains a single source node and the number of steps to reach a node Z_i from such source node is different for each $i \in [q]$, the `for` loop starting on line 13 runs for $q - 1$ iterations. The rest of the analysis assumes we are in such worst case. We have that $|V_{j-1}| = j$, and the inner loop starting on line 19 runs for $q - j$ iterations. Checking whether $\mathbf{c}_n \in \text{span}\{\mathbf{h}_n, \mathbf{h}_m : m \in V_{j-1}\}$ amounts to solving the system of linear equations

$$\begin{bmatrix} \uparrow & \uparrow & \uparrow & & \uparrow \\ \mathbf{h}_n & \mathbf{h}_{m_1} & \mathbf{h}_{m_2} & \cdots & \mathbf{h}_{m_j} \\ \downarrow & \downarrow & \downarrow & & \downarrow \end{bmatrix} \begin{bmatrix} x_n \\ x_{m_1} \\ x_{m_2} \\ \vdots \\ x_{m_j} \end{bmatrix} = \mathbf{c}_n^T, \tag{B.1}$$

which can be done in time $O(p(j+1)^2)$ using Gaussian elimination [19]. The

work done within the `if` block that follows is then determined by the time required to set the entries of Λ to their correct value on line 24, as the solutions to the system on line 23 are obtained when checking the span condition. This operation takes time $O(j)$, so the overall runtime of the `for` loop starting on line 13 is $O\left(\sum_{j=1}^{q-1}(q-j)p(j+1)^2\right) = O(q^4p)$. This is also the runtime of the algorithm, as $O(qp + q^4p) = O(q^4p)$.

6. Algorithm 6 takes as input two $p \times q$ matrices. The `for` loops starting on lines 5 and 7 both run for q iterations. In all but one iteration of the inner loop, two vectors of size $p \times 1$ are compared, which takes time $O(p)$. The overall time complexity of the algorithm is thus $O(q^2p)$.

7. Algorithm 7 takes as input a vector of size $1 \times n$ and a matrix of size $m \times n$. The bottleneck of the algorithm is the `for` loop starting on line 6. It runs for m iterations and the work done per iteration is $O(n)$, the time required to compute norm$(\mathbf{m}_i - v)$. The overall runtime of the algorithm is thus $O(mn)$.

8. Algorithm 8 takes as input two $q \times p$ matrices and a set of size at most q. The `for` loop starting on line 7 runs for q iterations and the work done per iteration is constant, so its runtime is $O(q)$. The one starting on line 10 also runs for q iterations, and the work done per iteration is determined by the call to match_row, which takes time $O(qp)$. The runtime of this `for` loop is thus $O(q^2p)$. In the worst case, the initial round of matching will result in all rows of C being matched with the same row of $\tilde{C}$. The rest of the analysis assumes we are in such worst case. We have that $|\text{unmatched\_rows}| = q - 1$ and the `if` block starting on line 17 is entered once, in which $|\text{tuple\_list}| = q$. The bottleneck of this `if` block is the `for` loop starting on line 22, which runs for q iterations. The work per iteration is in turn determined by the call to match_row on line 24, which takes time $O(qp)$. The overall runtime of the `for` loop starting on line 16 is thus $O(q^2p)$. Setting the entries of P to their correct value takes time $O(q)$ as it requires traversing through diff_list, and sorting the latter takes time $O(q \log q)$ [4]. Finally, the `for` loop starting on

line 34 runs for at most q iterations and the work per iteration is $O(q)$, so its time complexity is $O(q^2)$. The time complexity of the algorithm is thus equal to the time required to run the `for` loops starting on lines 10 and 16: $O(q^2 p)$.

9. As mentioned in Appendix A, Algorithm 9 is the adapted version of Algorithm 5, and the only difference between them is the method used to determine whether a vector v_r is in the span of a set of vectors $\{v_1, v_2, \ldots, v_n\}$. Algorithm 9 uses least squares to accomplish this task. Solving the least squares problem on line 8 amounts to calculating

$$\begin{aligned} \text{sol} &= \frac{1}{\mathbf{c}_i \cdot \mathbf{c}_i^T} \cdot \mathbf{c}_i \cdot (\mathbf{c}_i'^{(i)})^T, \\ \text{residual} &= \|\mathbf{c}_i^T \cdot \text{sol} - (\mathbf{c}_i'^{(i)})^T\|. \end{aligned} \tag{B.2}$$

Using the elementary algorithm for matrix multiplication, the runtime of this operation is $O(p)$, as both $\mathbf{c}_i$ and $\mathbf{c_i'^{(i)}}$ are $1 \times p$ vectors [46]. Thus, the `for` loop starting on line 7 runs for q iterations and the work per iteration is $O(p)$, so its overall runtime is $O(qp)$. The `for` loop starting on line 14 is the bottleneck of the algorithm. As mentioned when analyzing the time complexity of Algorithm 5, it runs for $q - 1$ iterations in the worst case, in which case the inner loop starting on line 20 runs for $q - j$ iterations. The rest of the analysis assumes we are in such worst case. Solving the least squares problem on line 22 amounts to calculating

$$\begin{aligned} \text{sol} &= (A^T A)^{-1} A^T \mathbf{c}_n^T \\ \text{residual} &= \|A \cdot \text{sol} - \mathbf{c}_n^T\| \end{aligned} \tag{B.3}$$

A is a matrix of size $p \times (j+1)$, so multiplying A^T times A takes time $O(p(j+1)^2)$ using the elementary algorithm for matrix multiplication [46]. Inverting the product can then be done in time $O((j+1)^3)$ using Gaussian elimination [19]. The time complexity of multiplying A^T times $\mathbf{c}_n^T$ is in turn $O(p(j+1))$, and that of multiplying $(A^T A)^{-1}$ times $A^T \mathbf{c}_n^T$ is $O((j+1)^2)$. Since j goes from 1 to $q - 1$ and $q \leq p$, $p \geq j + 1$ for all values of j, so the

runtime of the least squares operation on line 22 is $O(p(j+1)^2)$. As in Algorithm 5, the work done within the `if` block starting on line 23 is determined by the time required to set the entries of Λ to their correct value on line 27, which takes $O(j)$ time. The runtime of the `for` loop starting on line 20 is therefore $O\left(\sum_{j=1}^{q-1}(q-j)p(j+1)^2\right) = O(q^4p)$, the same as in Algorithm 5. This is also the overall runtime of the algorithm.

We now analyze the worst-case time complexity of our method for causal disentanglement. Let p and q denote the number of observed and latent variables, respectively. Constructing the cumulant tensors in each context using t samples takes time $O(tp^3)$. We then use Algorithm 1 to decompose each of the $q+1$ tensors, one per context. This takes time $O(qp^3)$. The method then runs Algorithm 8 q times to match each context with its corresponding intervention target and recover the relabelings of the latent nodes arising from tensor decomposition. This step has time complexity $O(q^3p)$. Once the intervention targets and permutation matrices have been recovered, we use Algorithm 4 to recover the pseudoinverse of the mixing matrix B, which takes time $O(qp)$. We can then recover B by calculating the pseudoinverse of the output of Algorithm 4, which can be done in time $O(q^2p)$ using singular value decomposition [55]. Finally, we use Algorithm 9 to recover Λ, which takes time $O(q^4p)$. The total cost is therefore $O(tp^3 + q^4p + qp^3)$. Assuming that the number of latent variables is less than the number of samples used to calculate the cumulants, this reduces to $O(tp^3 + q^4p)$.

References

[1] Kartik Ahuja, Jason Hartford, and Yoshua Bengio. Properties from mechanisms: an equivariance perspective on identifiable representation learning. In *International Conference on Learning Representations*, 2022.

[2] Kartik Ahuja, Jason Hartford, and Yoshua Bengio. Weakly supervised representation learning with sparse perturbations. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[3] Carlos Améndola, Mathias Drton, Alexandros Grosdos, Roser Homs, and Elina Robeva. Third-order moment varieties of linear non-Gaussian graphical models. *Information and Inference: A Journal of the IMA*, 12(3):iaad007, 2023.

[4] Nicolas Auger, Vincent Jugé, Cyril Nicaud, and Carine Pivoteau. On the worst-case complexity of TimSort. *arXiv preprint arXiv:1805.08612*, 2018.

[5] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and machine intelligence*, 35(8):1798–1828, 2013.

[6] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 2017.

[7] Johann Brehmer, Pim De Haan, Phillip Lippe, and Taco S Cohen. Weakly

supervised causal representation learning. *Advances in Neural Information Processing Systems*, 35:38319–38331, 2022.

[8] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with GPT-4. *arXiv preprint arXiv:2303.12712*, 2023.

[9] Simon Buchholz, Goutham Rajendran, Elan Rosenfeld, Bryon Aragam, Bernhard Schölkopf, and Pradeep Ravikumar. Learning linear causal representations from interventions under general nonlinear mixing. *Advances in Neural Information Processing Systems*, 36, 2024.

[10] Ruichu Cai, Feng Xie, Clark Glymour, Zhifeng Hao, and Kun Zhang. Triad constraints for learning causal structure of latent variables. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[11] J Douglas Carroll and Jih-Jie Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of "Eckart-Young" decomposition. *Psychometrika*, 35(3):283–319, 1970.

[12] Chandler Squires. `causaldag`: creation, manipulation, and learning of causal models. https://github.com/uhlerlab/causaldag, 2018.

[13] Pierre Comon. Independent component analysis, a new concept? *Signal Processing*, 36:287–314, 1994.

[14] Pierre Comon and Christian Jutten. *Handbook of Blind Source Separation: Independent component analysis and applications*. Academic press, 2010.

[15] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*, chapter 22.2, pages 594–602. The MIT Press, 3rd edition, 2009.

[16] Reinhard Diestel. *Graph Theory*. Electronic library of mathematics. Springer, 2005.

[17] Frederick Eberhardt, Clark Glymour, and Richard Scheines. On the number of experiments sufficient and in the worst case necessary to identify all causal relations among n variables. In *Conference on Uncertainty in Artificial Intelligence*, 2005.

[18] Jan Eriksson and Visa Koivunen. Identifiability, separability, and uniqueness of linear ICA models. *IEEE Signal Processing Letters*, 11(7):601–604, 2004.

[19] Richard William Farebrother. *Linear least squares computation*, chapter 1, pages 11–13. Routledge, 1988.

[20] Ronald Aylmer Fisher. Moments and product moments of sampling distributions. *Proceedings of the London Mathematical Society*, 2(1):199–238, 1930.

[21] Yoni Halpern, Steven Horng, and David Sontag. Anchored discrete factor analysis. *arXiv preprint arXiv:1511.03299*, 2015.

[22] Richard A. Harshman. Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multimodal factor analysis. *UCLA Working Papers in Phonetics*, 16:1–84, 1970.

[23] Alain Hauser and Peter Bühlmann. Characterization and greedy learning of Markov equivalence classes of directed acyclic graphs. *Journal of Machine Learning Research*, 13:2409–2464, 2012.

[24] Yangbo He, Jinzhu Jia, and Bin Yu. Counting and exploring sizes of Markov equivalence classes of directed acyclic graphs. *Journal of Machine Learning Research*, 16(79):2589–2609, 2015.

[25] Aapo Hyvärinen, Hiroaki Sasaki, and Richard Turner. Nonlinear ICA using auxiliary variables and generalized contrastive learning. In Kamalika

Chaudhuri and Masashi Sugiyama, editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 859–868, 2019.

[26] Aapo Hyvärinen and Petteri Pajunen. Nonlinear independent component analysis: Existence and uniqueness results. *Neural Networks*, 12:429–439, 1999.

[27] Yibo Jiang and Bryon Aragam. Learning nonparametric latent causal graphs with unknown interventions. *Advances in Neural Information Processing Systems*, 36, 2024.

[28] Ilyes Khemakhem, Ricardo Monti, Diederik Kingma, and Aapo Hyvarinen. ICE-BeeM: Identifiable conditional energy-based deep models based on nonlinear ICA. *Advances in Neural Information Processing Systems*, 33:12768–12778, 2020.

[29] Joe Kileel and Joao M. Pereira. Subspace power method for symmetric tensor decomposition and generalized PCA. *arXiv preprint arXiv:1912.04007*, 2019.

[30] Bohdan Kivva, Goutham Rajendran, Pradeep Ravikumar, and Bryon Aragam. Learning latent causal graphs via mixture oracles. *Advances in Neural Information Processing Systems*, 34:18087–18101, 2021.

[31] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009.

[32] Sue E. Leurgans, Robert T. Ross, and Rebecca B. Abel. A decomposition for three-way arrays. *SIAM Journal on Matrix Analysis and Applications*, 14(4):1064–1083, 1993.

[33] Yuhang Liu, Zhen Zhang, Dong Gong, Mingming Gong, Biwei Huang, Anton van den Hengel, Kun Zhang, and Javen Qinfeng Shi. Identifying weight-variant latent causal models. *arXiv preprint arXiv:2208.14153*, 2022.

[34] Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *International Conference on Machine Learning*, pages 4114–4124. PMLR, 2019.

[35] Peter McCullagh. *Tensor methods in statistics: Monographs on statistics and applied probability*. Chapman and Hall/CRC, 2018.

[36] Nicolai Meinshausen, Alain Hauser, Joris M. Mooij, Jonas Peters, Philip Versteeg, and Peter Bühlmann. Methods for causal inference from gene perturbation experiments and validation. *Proceedings of the National Academy of Sciences*, 113(27):7361–7368, 2016.

[37] Kevin Murphy. An introduction to graphical models. *Rap. tech*, 96:1–19, 2001.

[38] `OSCAR`: Open source computer algebra research system, version 1.1.0-dev. https://www.oscar-system.org, 2024.

[39] Victor Y Pan and Zhao Q Chen. The complexity of the matrix eigenproblem. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing*, pages 507–516, 1999.

[40] Karl Pearson. Method of moments and method of maximum likelihood. *Biometrika*, 28(1/2):34–59, 1936.

[41] Paul Purdom Jr. A transitive closure algorithm. *BIT Numerical Mathematics*, 10(1):76–94, 1970.

[42] Erdos Renyi. On random graph. *Publicationes Mathematicate*, 6:290–297, 1959.

[43] Basil Saeed, Anastasiya Belyaeva, Yuhao Wang, and Caroline Uhler. Anchored causal inference in the presence of measurement error. In *Conference on Uncertainty in Artificial Intelligence*, 2019.

[44] Shohei Shimizu. LiNGAM: non-Gaussian methods for estimating causal structures. *Behaviormetrika*, 41:65–98, 2014.

[45] Ricardo Silva, Richard Scheine, Clark Glymour, and Peter Spirtes. Learning the structure of linear latent variable models. *Journal of Machine Learning Research*, 7(8):191–246, 2006.

[46] Steven S. Skiena. *The algorithm design manual*, chapter 2, pages 45–46. Springer, 1998.

[47] Charles Spearman. Pearson's contribution to the theory of two factors. *British Journal of Psychology*, 19(1):95, 1928.

[48] Chandler Squires, Anna Seigal, Salil Bhate, and Caroline Uhler. Linear causal disentanglement via interventions. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

[49] Chandler Squires and Caroline Uhler. Causal structure learning: A combinatorial perspective. *Foundations of Computational Mathematics*, 23(5):1781–1815, 2023.

[50] Stefan G Stark, Joanna Ficek, Francesco Locatello, Ximena Bonilla, Stéphane Chevrier, Franziska Singer, Tumor Profiler Consortium, Gunnar Rätsch, and Kjong-Van Lehmann. SCIM: universal single-cell matching with unpaired feature sets. *Bioinformatics*, 36(Supplement 2):919–927, 12 2020.

[51] Gregor Stiglic, Primoz Kocbek, Nino Fijacko, Marinka Zitnik, Katrien Verbert, and Leona Cilar. Interpretability of machine learning-based prediction models in healthcare. *WIREs Data Mining and Knowledge Discovery*, 10(5), June 2020.

[52] Jin Tian and Judea Pearl. Causal discovery from changes. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, pages 512–521, 2001.

[53] Thaddäus Tönnies, Sabine Kahl, and Oliver Kuss. Collider bias in observational studies: consequences for medical research part 30 of a series on evaluation of scientific publications. *Deutsches Ärzteblatt International*, 119(7):107, 2022.

[54] Burak Varici, Emre Acarturk, Karthikeyan Shanmugam, Abhishek Kumar, and Ali Tajer. Score-based causal representation learning with interventions. *arXiv preprint arXiv:2301.08230*, 2023.

[55] Vinita Vasudevan and M Ramakrishna. A hierarchical singular value decomposition algorithm for low rank matrices. *arXiv preprint arXiv:1710.02812*, 2017.

[56] Thomas Verma and Judea Pearl. Equivalence and synthesis of causal models. In *Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intelligence*, 1990.

[57] Julius von Kügelgen, Michel Besserve, Liang Wendong, Luigi Gresele, Armin Kekić, Elias Bareinboim, David Blei, and Bernhard Schölkopf. Nonparametric identifiability of causal representations from unknown interventions. *Advances in Neural Information Processing Systems*, 36, 2024.

[58] Kexin Wang and Anna Seigal. Identifiability of overcomplete independent component analysis. *arXiv preprint arXiv:2401.14709*, 2024.

[59] Quan Wang. Kernel principal component analysis and its applications in face recognition and active shape models. *arXiv preprint arXiv:1207.3538*, 2012.

[60] Jiaqi Zhang, Kristjan Greenewald, Chandler Squires, Akash Srivastava, Karthikeyan Shanmugam, and Caroline Uhler. Identifiability guarantees for causal disentanglement from soft interventions. *Advances in Neural Information Processing Systems*, 36, 2024.

[61] Dihan Zheng, Xiaowen Zhang, Kaisheng Ma, and Chenglong Bao. Learn from unpaired data for image restoration: A variational Bayes approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5):5889–5903, 2022.

[62] Yujia Zheng, Ignavier Ng, and Kun Zhang. On the identifiability of nonlinear ICA: sparsity and beyond. *Advances in Neural Information Processing Systems*, 35:16411–16422, 2022.

[63] Roland S Zimmermann, Yash Sharma, Steffen Schneider, Matthias Bethge, and Wieland Brendel. Contrastive learning inverts the data generating process. In *International Conference on Machine Learning*, pages 12979–12990. PMLR, 2021.