Generative Neural networks to infer Causal Mechanisms: Algorithms and applications

Diviyan Kalainathan

NNT : 2019SACLS516

Thèse de doctorat

Generative Neural Networks to infer Causal Mechanisms: Algorithms and applications

Thèse de doctorat de l'Université Paris-Saclay
préparée à Equipe TAU, Laboratoire de recherche en Informatique (LRI), Université Paris-Sud, INRIA

Ecole doctorale n°580 Sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat : Informatique

Thèse présentée et soutenue à Gif-Sur-Yvette, le 17 Décembre 2019, par

DIVIYAN KALAINATHAN

Composition du Jury :

Jean-Pierre Nadal Directeur de Recherche au CNRS, Directeur d'Etudes à l'EHESS	Président
Kristin Bennett Professeur, Rensselaer Polytechnic Institute	Rapporteur
Kun Zhang Assistant Professeur, Carnegie Mellon University	Rapporteur
Julie Josse Professeur, CMAP, Ecole Polytechnique, INRIA	Examinateur
Isabelle Guyon Professeur, TAU, LRI, Université Paris-Sud	Directeur de thèse
Michèle Sebag Professeur, TAU, LRI, Université Paris-Sud, CNRS	Co-directeur de thèse

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Deep learning models have shown extraordinary predictive abilities, breaking records in computer vision (Krizhevsky et al., 2012), speech recognition (Hinton et al., 2012), language translation (Cho et al., 2014), and reinforcement learning (Silver et al., 2016). Decision makers accordingly tend to leverage those models to not only predict, but also control phenomena. However, current machine learning paradigms are blind to the distinction between correlation and causation, which might be harmless in terms of prediction, but may have very undesirable effects in terms of control. For example, consider the prediction of target variable Y given features X and Z, assuming that the underlying generative process is described by the equations:

$$\begin{aligned} X, E_Y, E_Z &\sim \text{Uniform}(0, 1), \\ Y &\leftarrow 0.5X + E_Y, \\ Z &\leftarrow Y + E_Z, \end{aligned}$$

with (E_Y, E_Z) additive noise variables. The above model states that the values of Y are computed as a function of the values of X (we say that X causes Y), and that the values of Z are computed as a function of the values of Y (Y causes Z). The "assignment arrows" emphasize the asymmetric relations between all three random variables. Accordingly, one willing to control Y only needs to control X.

However, as Z provides a stronger signal-to-noise ratio than X for the prediction of Y, the best regression solution in terms of least-square error is

$$\hat{Y} = 0.25X + 0.5Z$$

The above regression model, a typical case of inverse regression after[1] Goldberger (1984), would wrongly explain some changes in Y as a function of changes in Z, although Z does

[1] In this simple linear case, there exists approaches overcoming the inverse regression mistake and uncovering all true cause-effect relations (Hoyer et al., 2009). In the general case however, mainstream machine learning approaches fail to understand the relationships between all three distributions, and might attribute some effects on Y to changes in Z.

not cause Y. This model thus suggests that the value of Y can be influenced (i.e., Y can be controlled) by acting mostly on Z.

In brief, correlation-based models lead to wrong conclusions in terms of control: Mistaking correlation for causation can be catastrophic for agents who must plan, reason, and decide based on observations. Thus, discovering causal structures is of crucial importance. As detailed in Imbens and Rubin (2015), the discovery of causal relationships is at the core of many natural sciences, aiming to understand the world and its mechanisms. The gold standard to discover causal relations is to perform experiments (Pearl, 2003). However, experiments are in many cases expensive, unethical, or impossible to realize. In these situations, there is a need for *observational causal discovery*, that is, the estimation of causal relations from observations alone (Spirtes et al., 2000; Peters et al., 2017).

To perform observational causal discovery *for two or more variables*, leading researchers in causality have proposed methods that leverage conditional independences. All these methods have provable *consistency*, *i.e.,* the true underlying causal model can be asymptotically recovered (identifiability), under some assumptions (Section 2.1.3). Unfortunately, these assumptions might be overly restrictive (*e.g.,* assuming the absence of “confounding” variables resulting from hidden or unknown causes); and most domain experts ignore whether these assumptions hold in practice. For both reasons, the validity of the resulting causal models might be hard to ascertain.

In the case where *only two variables* are available, conditional independences cannot be leveraged (since the conditioning set is empty); this prompted the development of new approaches, relying on the simplicity of the causal mechanisms derived from Occam’s razor principle (Hoyer et al., 2009; Zhang and Hyvärinen, 2010; Stegle et al., 2010). More recently, Guyon (2013, 2014) has proposed a machine learning challenge in cause-effect pairs pattern recognition. In this setting, machine learning algorithms are provided with a training set consisting of joint distributions of pairs of variables labeled with their causal relationship. The algorithms must predict causal relationships between pairs of variables from a test set of new joint distributions never seen before. The algorithms developed by challenge participants produce good performance, provided that adequate training datasets are provided. Although using Occam’s razor may also be beneficial in the case of more than two variables, few approaches have tried to mix several causal discovery principles to infer causal relationships in a unified approach (Bühlmann et al., 2014).

Our contribution in this thesis is to exploit the modularity and expressiveness of *neural networks* for causal discovery leveraging both *conditional independences* and *simplicity of the causal mechanisms* (Occam’s razor principle).

1.1 Outline

The main goal of this thesis is to achieve structural causal discovery from observations; the proposed approach relies on recent machine learning techniques, chiefly neural network architectures and adversarial learning mechanisms.

Two main algorithmic contributions are made.

A first contribution is concerned with causal *structure* discovery, with the development of a neural network approach based on stochastic gradient descent capable of learning *the structure* of a causal graph from domain observations only (as opposed to learning from experimental data), using parsimony-enforcing regularization and exploiting conditional independences between variables.

A second contribution is concerned with causal model *parameter fitting.* Our regularized neural networks broaden the class of causal mechanisms involved in structural equation models (beyond linear causal mechanisms and/or additive noise (Spirtes et al., 2000; Hoyer et al., 2009; Bühlmann et al., 2014)) without adverse model over-fitting side-effects. As opposed to restricting beforehand the complexity of the sought mechanisms, a regularization scheme is proposed, as Stegle et al. (2010), to adjust the trade-off between the data fit and the model complexity.

On the theoretical side, the contribution made is an algorithm analysis establishing the identifiability of the sought causal model, under some assumptions.

The main software contribution consists in the *Causal Discovery Toolbox*, a Python package gathering many graph and pairwise approaches to observational causal discovery (more details in Appendix A).

This thesis is divided in 4 chapters, two on the state of the art in causal discovery (Chapter 2) and neural network architectures (Chapter 3), and two on the proposed contributions: Chapter 4 presents the causal generative neural networks, assuming that the graph skeleton is available and extending score-based methods to the generative neural network framework. Chapter 5 presents the structural agnostic model, relaxing the previous assumption and achieving the end-to-end identification of the causal graph and of the causal mechanisms from data.

In summary, the thesis contributions are along three axes: theoretical, algorithmic, and implementation. The theoretical and algorithmic contributions consist in two algorithms for observational causal discovery, and their theoretical analysis. These contributions rely on a thorough review of the state of the art on pairwise causal discovery, published as three chapters of the Guyon et al. (2019) book.

The list of published papers is given below:

- *Portraits de travailleurs*,
  Diviyan Kalainathan, Olivier Goudet, Philippe Caillou, Isabelle Guyon, Michèle Sebag, Emilie Bourdu-Swzedek, Thierry Weil, 2017, La Fabrique de l'industrie.

- *Learning Functional Causal Models with Generative Neural Networks*,
  Olivier Goudet[*], Diviyan Kalainathan*, Philippe Caillou, Isabelle Guyon, David Lopez-Paz, Michèle Sebag, 2018, Chapter in "Explainable Machine Learning", Springer Verlag.

[*] Equal contribution

- *Structural Agnostic Modeling: Adversarial Learning of Causal Graphs*,
  Diviyan Kalainathan, Olivier Goudet, Philippe Caillou, Isabelle Guyon, David Lopez-Paz, Michèle Sebag, 2018, ArXiv.

- *[Re] World Models*,
  Corentin Tallec, Léonard Blier, Diviyan Kalainathan, 2019, ReScience.

- *Causal Discovery Toolbox: Uncover causal relationships in Python*,
  Diviyan Kalainathan, Olivier Goudet, 2019, Journal of Machine Learning Research - Open Source Software.

- *Evaluation methods of cause-effect pairs*,
  Isabelle Guyon, Olivier Goudet, Diviyan Kalainathan 2019, Chapter II in "Cause effect pairs", Springer Verlag.

- *Learning Bivariate Functional Causal Models*,
  Olivier Goudet, Diviyan Kalainathan, Michèle Sebag, Isabelle Guyon, 2019, Chapter III in "Cause effect pairs", Springer Verlag.

- *Predicting Pairwise Causality with Discriminative Methods*,
  Diviyan Kalainathan, Olivier Goudet, Michèle Sebag, Isabelle Guyon, 2019, Chapter IV in "Cause effect pairs", Springer Verlag.

Chapter 2

Causal discovery and models

In this chapter, the state of the art on observational causal discovery is reviewed and discussed, referring the reader to (Spirtes et al., 2000; Peters et al., 2017; Guyon et al., 2019) for a comprehensive survey. First, notions on causal discovery and assumptions are introduced. Then, the state of the art on observational causal discovery algorithms is presented, distinguishing local search algorithms (Section 2.3), global search algorithms (Section 2.4), methods leveraging distributional asymmetries (Section 2.5), and finally methods combining approaches (Section 2.6).

2.1 Observational causal discovery: Formal background

In the considered setting, we assume that data are generated from an underlying causal model, with a well defined "true" causal graph.

Formally, let $\mathbf{X} = [X_1, ..., X_d]$ be a vector of d real valued features and $P(\mathbf{X})$ its associated probability distribution. From $P(\mathbf{X})$, an *observational* empirical dataset, of n sample, is drawn (independently and identically distributed). Data with controlled interventions and time-series data are excluded in the following.

Variables X_i are causally linked along true causal graph $\mathcal{G}$. By abuse of notations, X_i denotes a variable and its corresponding node in the causal graph. A **Causal graph** $\mathcal{G}$ is a directed graph where an edge represents a causal relationship between the connected nodes. Considering an edge $X_i \rightarrow X_j$, X_i refers to the cause, and X_j to the effect of the causal relationship. A d-variable causal graph can also be represented by a $d \times d$ binary adjacency matrix A representing all the connections between all the variables: $A_{ij} = 1$ if and only if X_i causes X_j and 0 otherwise. The number of possible graphs is super-exponential in d: $O(2^{d \times d})$, highlighting the difficulty of causal discovery and preventing exhaustive search for non-toy problems.

Let intervention *do(X=x)* be defined as the operation on distribution obtained by clamping variable X to value x, while the rest of the system remains unchanged (Pearl, 2009). It is said that variable X_i is a **direct cause** of X_j with respect to $X_1, ..., X_d$ *iff* different interventions on variable X result in different marginal distributions on X_j, everything else being equal:

$$P_{X_j|\mathrm{do}(X_i=x,\mathbf{X}_{\backslash ij}=\mathbf{c})} \neq P_{X_j|\mathrm{do}(X_i=x',\mathbf{X}_{\backslash ij}=\mathbf{c})} \tag{2.1}$$

with $\mathbf{X}_{\backslash ij} := X_{\{1,\ldots,d\}\backslash i,j}$ the set of all variables except X_i and X_j, scalar values $x \neq x'$, and vector value $\mathbf{c}$. Distribution $P_{X_j|\mathrm{do}(X_i=x,\mathbf{X}_{\backslash ij}=c)}$ is the resulting interventional distribution of the variable X_j when the variable X_i is clamped to value x, while keeping all other variables at a fixed value (Mooij et al., 2016).

Example 1. *Considering the 3 variable example $X \rightarrow Y \rightarrow Z$ such that:*

$$\begin{cases} X, E_Y, E_Z \sim \mathit{Uniform}(0,1), \\ Y \leftarrow 0.5X + E_Y, \\ Z \leftarrow Y + E_Z, \end{cases}$$

Applying the interventions $Y = 1$ and $Y = 0$ does not affect in any way the distribution of the X variable $P_{X|do(Y=1,Z=cst)} = P_{X|do(Y=0,Z=cst)} = U(0,1)$, whereas the distribution of Z is shifted: $P_{Z|do(Y=1,X=cst)} = U(1,2)$ and $P_{Z|do(Y=0,X=cst)} = U(0,1)$.

2.1.1 Functional Causal Models

A Functional Causal Model (FCM) upon a random variable vector $\mathbf{X} = [X_1, \ldots, X_d]$ is a triplet $(\mathcal{G}, f, \mathcal{E})$, representing a set of equations:

$$X_i \leftarrow f_i(X_{\mathrm{Pa}(i;\mathcal{G})}, E_i), E_i \sim \mathcal{E}, \text{ for } i = 1, \ldots, d \tag{2.2}$$

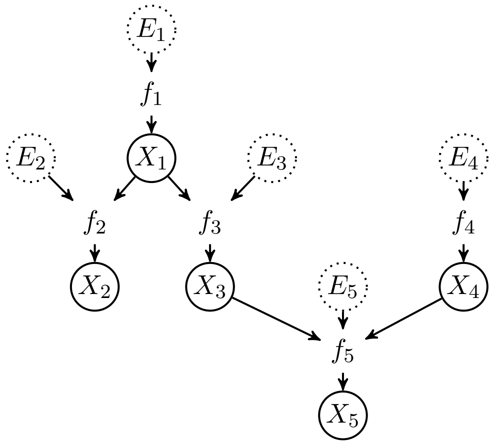
Each equation characterizes the direct causal relation explaining variable X_i from the set of its causes $X_{\mathrm{Pa}(i;\mathcal{G})} \subset \{X_1, \ldots, X_d\}$, based on the so-called *causal mechanism* f_i involving besides $X_{\mathrm{Pa}(i;\mathcal{G})}$ some random variable E_i drawn after distribution $\mathcal{E}$, meant to account for all unobserved variables. The causal mechanism f_i represents a function that takes as input all the causes and the noise variable E_i to output the effect variable.

Letting $\mathcal{G}$ denote the causal graph obtained by drawing arrows from causes $X_{\mathrm{Pa}(i;\mathcal{G})}$ towards their effects X_i, we restrict ourselves to directed acyclic graphs (DAG), where the propagation of interventions to end nodes is assumed to be instantaneous. An example of functional causal model with five variables is illustrated on Fig. 2.1.

In causal discovery, we seek a Functional Causal Model (FCM), also known as Structural Equation Model (SEM), that best matches the underlying data-generating mechanism(s) in the following sense: under relevant manipulations/interventions/experiments the FCM would produce data distributed similarly to the real data obtained in similar conditions.

2.1.2 Notations and Definitions

$\mathbf{X}_{\backslash i}$: denotes the set of all variables but X_i.
$\widehat{\mathbf{U}}$: represents the approximation of a variable $\mathbf{U}$.



$$\begin{cases} X_1 = f_1(E_1) \\ X_2 = f_2(X_1, E_2) \\ X_3 = f_3(X_1, E_3) \\ X_4 = f_4(E_4) \\ X_5 = f_5(X_3, X_4, E_5) \end{cases}$$

Figure 2.1: Example of a Functional Causal Model (FCM) on $\mathbf{X} = [X_1, \ldots, X_5]$: Left: causal graph $\mathcal{G}$; right: causal mechanisms.

Conditional independence: $(X_i \perp\!\!\!\perp X_j | X_k)$ means that variables X_i and X_j are independent conditionally to X_k, i.e. $P(X_i, X_j | X_k) = P(X_i | X_k) P(X_j | X_k)$.
Markov blanket: a Markov blanket $\mathrm{MB}(X_i)$ of a variable X_i is a minimal subset of variables in $\mathbf{X}_{\backslash i}$ such that any disjoint set of variables in the network is independent of X_i conditioned on $\mathrm{MB}(X_i)$. In a Bayesian network, the Markov blanket of a node corresponds to its parents, children and spouses (parents of its children), as shown in Figure 2.2.

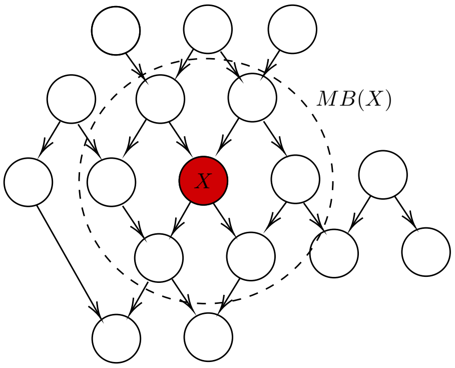


Figure 2.2: Illustration of a Markov blanket of a node X (in red). The Markov blanket corresponds to the nodes inside the dotted line, except from X.

V-structure: Variables $\{X_i, X_j, X_k\}$ form a v-structure iff their causal structure is:

$$X_i \rightarrow X_k \leftarrow X_j$$

Confounder: Variable Z is a confounder of X_i and X_j if their causal structure is :

$$X_i \leftarrow Z \rightarrow X_j$$

Z is called *hidden confounder* if $Z \notin \mathbf{X}$.
Skeleton of the DAG: the skeleton of the DAG is the undirected graph obtained by replacing all edges by undirected edges. In our setting, that corresponds to $A_{ij} = A_{ji} = 1$, A denoting the adjacency matrix of the skeleton.
Markov equivalent DAG: two DAGs with same skeleton and same v-structures are said to be *Markov equivalent* (Pearl and Verma, 1991). A *Markov equivalence class* is represented by a *Completed Partially Directed Acyclic Graph* (CPDAG) having both directed and undirected edges (refer to Section 2.2 for more detailed information).
Adjacent nodes: X_i and X_j are said to be adjacent according to a CPDAG *iff* there exists an edge between both nodes. If directed, this edge models either the causal relationship $X_i \to X_j$ or the opposite relation $X_j \to X_i$. If undirected, it models a causal relationship in either direction.

2.1.3 Causal Assumptions and Properties

In this work, the recovery of the underlying causal graph $\mathcal{G}$ from observational data relies on the following assumptions:

Acyclicity: The d-variable causal graph $\mathcal{G}$ is assumed to be a Directed Acyclic Graph (DAG): there exists no $i \in [1, d]$, such that a causal path $X_i \to \ldots \to X_i$ is present in $\mathcal{G}$. This translates in terms of adjacency matrix A to $A^d = \mathbf{0}$.

Causal Markov Assumption (CMA): Noise variables E_j (Eq. (2.2)) are assumed to be independent from each other. This assumption together with the above DAG assumption yields the classical causal Markov property, stating that all variables are independent of their non-effects (non descendants in the causal graph) conditionally to their direct causes (parents) (Spirtes et al., 2000). Under the causal Markov assumption, the distribution described by the FCM satisfies all conditional independence relations[1] among variables in $\mathbf{X}$ via the notion of d-separation (Pearl, 2009). Accordingly, the joint distribution $p(\mathbf{x})$ can be factorized as the product of the distributions of each variable conditionally on their parents in the graph:

$$p(\mathbf{x}) = \prod_{j=1}^{d} p(x_j | x_{\mathrm{Pa}(j;\mathcal{G})}) \tag{2.3}$$

Causal Faithfulness Assumption (CFA): The joint distribution $p(\mathbf{x})$ is assumed to be *faithful* to graph $\mathcal{G}$, that is, every conditional independence relation that holds true according to p is entailed by $\mathcal{G}$ (Spirtes and Zhang, 2016).

It follows from causal Markov and faithfulness assumptions that every causal path in the graph corresponds to a dependency between variables, and vice versa.

[1] It must be noted however that the data might satisfy additional independence relations beyond those in the graph; see the faithfulness assumption.

Causal Sufficiency assumption (CSA): $\mathbf{X}$ is assumed to be *causally sufficient*, that is, a pair of variables $\{X_i, X_j\}$ in $\mathbf{X}$ has no common cause external to $\mathbf{X}_{\setminus i,j}$. This assumption is often made because of *hidden confounding* variables: if an external variable Z causes X_i and X_j, both these variables will be dependent as they possess the same parent. However, no edge exists between these variables. This highlights a tricky problem of dependencies that are not explainable using only the set of observable variables.

Selection bias : The data sampling procedure introduces a bias if the samples were selected from the population depending on the values of any of the variables in the data. The assumption of having no selection bias in the data is made in all algorithms and methods presented in this work.

2.2 How to infer causality from observational data ?

Identifying causal relationships with only observational data mainly relies on two key steps: Conditional statistics in the data, with the identification of v-structures, and modeling the data distribution given a constraint on complexity of the proposed model, following Occam's razor principle.

2.2.1 Notions of independence and conditional independence

Independence and conditional independence between variables are characterized by summary statistics stemming from information theory. Letting (X, Y) denote a pair of continuous random variables with joint probability density $p(x, y)$, X and Y are independent ($X \perp\!\!\!\perp Y$) if and only if the $p(x, y)$ can be decomposed into the product of marginals:

$$X \perp\!\!\!\perp Y \iff p(x, y) = p(x)p(y)$$

. Likewise, X and Y are considered independent conditionally to a third variable Z if and only if:

$$X \perp\!\!\!\perp Y|Z \iff p(x, y|z) = p(x|z)p(y|z)$$

.

The mutual information between X and Y is defined as:

$$I(X, Y) = \int_{\mathbb{R}} \int_{\mathbb{R}} p(x, y) \log \frac{p(x, y)}{p(x)p(y)} d_x d_y \tag{2.4}$$

The mutual information measures the information shared by X and Y and how much knowing either variable reduces uncertainty about the other one. It can be expressed in term of entropy:

$$I(X, Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) = H(X) + H(Y) - H(X, Y) \tag{2.5}$$

Mutual information $I(X, Y)$ is equal to zero iff X and Y are independent. In the case where X and Y follow Gaussian distributions, it holds:

$$I(X, Y) = -\frac{1}{2}\log(1 - \rho_{x,y}^2) \tag{2.6}$$

where $\rho_{x,y}$ denotes the Pearson correlation coefficient between X and Y. Mutual information can also be expressed with the Kullback-Leibler divergence:

$$I(X, Y) = D_{KL}(p(x, y) \,\|\, p(x)p(y)) \tag{2.7}$$

Let us now consider a third variable Z. The conditional mutual information between the variables X and Y conditionally to Z is:

$$I(X, Y|Z) = \int_{\mathbb{R}}\int_{\mathbb{R}}\int_{\mathbb{R}} p(x, y, z)\log\frac{p(x, y|z)}{p(x|z)p(y|z)}dxdydz \tag{2.8}$$

Likewise, $I(X, Y|Z) = 0$ iff X and Y are independent conditionally to Z. The Markov blanket $\mathrm{MB}(X)$ of X, defined in Section 2.1.2, can be reformulated in terms of conditional mutual information: for every Z in $\mathbf{X}\backslash\{X\} \cup \mathrm{MB}(X)$, it holds:

$$I(X, Z|\mathrm{MB}(X)) = 0 \tag{2.9}$$

The notion of Markov blanket is used in the causal discovery literature to prune irrelevant edges before recovering the Markov equivalence class of the DAG, see e.g. (Tsamardinos et al., 2003).

In the field of causal feature selection (Yu et al., 2018), recovering the Markov blanket of the target variable also is a key step in order to select relevant variables (Brown et al., 2012; Meyer and Bontempi, 2013), and remove variables that are independent of the target conditionally to the selected variables.

Empirically, independence between variables is evaluated with a test statistic (*e.g.,* MI, Pearson correlation, F-statistic, depending on the type of dependence/alternative distribution anticipated and the type of variable – continuous or categorical), and its significance is assessed with a p-value. Likewise, there are test procedures for conditional independence, such as the z-Fisher test:

z-Fisher test Considering variables X_1 and X_2, a set of conditioning variables $S = \{X_j\}, j \in [3, d]$ and noting the correlation matrix of the data R, the test is computed as follows:

- Compute the partial correlation[2] between X_1 and X_2 conditionally to S, noted $r_{1,2|S}$
  - If $Card(S) = 0$: $r_{1,2|S} = R_{12}$
  - Else if $Card(S) = 1$: $r_{1,2|S} = \frac{R_{12} - R_{1S} * R_{2S}}{\sqrt{(1-R_{1S}^2)(1-R_{2S}^2)}}$

[2] The partial correlation corresponds to the correlation between two variables as if the conditioning variables were constant. Partial correlation can be computed by regression, recursivity or matrix inversion.

  - Else: Let $R^{inv} = R^{-1}_{[12S],[12S]}$, the inverse of the sub-matrix with the coordinates $1, 2$ and S,
    $r_{1,2|S} = \frac{-R^{inv}_{01}}{\sqrt{R^{inv}_{11} * R^{inv}_{22}}}$

- Compute the z-value: $z = \frac{1}{2}\left(\ln(1 + r_{1,2|S} - \ln(1 - r_{1,2|S}))\right)$

Replacing the traditional z-Fisher conditional independence test with better performing and non-parametric tests has become increasingly popular. The following tests represent the main recent contributions in the field:

Kernel Conditional Independence test (KCI) Kernel methods are popular due to their representative power that brings improved accuracy at the cost of computational power. Zhang et al. (2012) leverages the kernel-based Hilbert-Schmidt Independence Criterion (HSIC) (Gretton et al., 2005b) to test conditional independence. Two alternatives are proposed to estimate the null distribution based on which the p-value is going to be computed: Sampling randomly points from the distributions to break dependencies, or using the gamma distribution, which proves itself to be quite costly.

Randomized Conditional Independence Test (RCIT) As the KCI test is computationally expensive, Strobl et al. (2017) propose to approximate this criterion in linear time. This approximation is made with randomized Fourier transformations, and the estimation of the null distribution is computed with the Lindsay-Pilla-Basak approximation.

Conditional Mutual Information Test (CMIT) To tackle the issue of test accuracy with small sample size or small conditioning set, Runge (2017) introduces a new test approximating the conditional mutual information using a k-nearest neighbor criterion. The null distribution estimation is also adapted using a nearest-neighbor random permutation.

2.2.2 Leveraging conditional statistics

Some methods to uncover the causal *structure* of graphs rely on Markov equivalence properties. To illustrate the principle of these methods, consider three variables X, Y, Z endowed with two Markov properties (conditional dependences/independences): $X \perp\!\!\!\perp Z|Y$ and $X \not\perp\!\!\!\perp Z$. The three following causal structures cannot be distiguished on the basis of such Markov properties:

$$\begin{cases} X \rightarrow Y \rightarrow Z \\ X \leftarrow Y \leftarrow Z \\ X \leftarrow Y \rightarrow Z \end{cases}$$

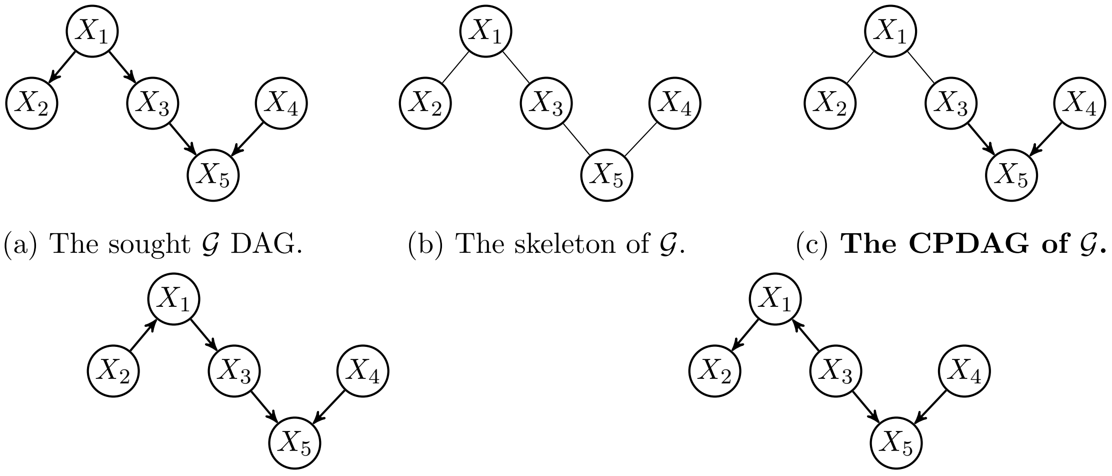
Hence they are called “Markov equivalent”. In contrast, the *v-structure* $X \rightarrow Y \leftarrow Z$ is uniquely identifiable if the Markov properties $X \perp\!\!\!\perp Z$ and $X \not\perp\!\!\!\perp Z|Y$ hold.

Leveraging conditional independences to detect v-structures allows to partially recover the causal graph: some remaining edges can be oriented using constraints on the graph: all the v-structures being detected, some Markov equivalent graphs are not admissible as they would create new v-structures or cycles. After this step, some edges remain undirected. The resulting set of admissible graphs, obtained by orienting the remaining edges with Markov properties, represent the *Markov equivalence class of the graph* called **Completed Partially Directed Acyclic Graph (CPDAG)** in its general formulation.

Example 2. *This notion is illustrated on the 5 variables examples (Fig. 2.1): the sought DAG $\mathcal{G}$ (ground truth) and graph skeleton are respectively depicted on Fig. 2.3.a and Fig. 2.3.b.*

The sought DAG is unknown, but empirical data drawn using it may be used to recover the graph skeleton (using independence tests). Next, under the assumptions CSA, CMA and CFA (see Section 2.1.3), since $(X_3 \perp\!\!\!\perp X_4 | X_5)$ does not hold, a v-structure $X_3 \rightarrow X_5 \leftarrow X_4$ is identified (Fig. 2.3.c).

However, since $(X_1 \perp\!\!\!\perp X_5 | X_3)$ and $(X_2 \perp\!\!\!\perp X_3 | X_1)$ hold, the DAGs depicted on Fig. 2.3.d and Fig. 2.3.e encode the same conditional independences as the true DAG (Fig. 2.3.a). Therefore the true DAG cannot be fully identified on the basis of independence tests, and the edges between pairs of nodes $\{X_1, X_2\}$ and $\{X_1, X_3\}$ must be left undirected. The process thus terminates with a Completed Partially Directed Acyclic Graph *(CPDAG), depicted on Fig. 2.3.c.*



(a) The sought $\mathcal{G}$ DAG.

(b) The skeleton of $\mathcal{G}$.

(c) **The CPDAG of $\mathcal{G}$.**

(d) A Markov equivalent DAG of $\mathcal{G}$.

(e) Another Markov equivalent DAG of $\mathcal{G}$.

Figure 2.3: **A Markov equivalent class:** Given a graph skeleton (b), all three DAGS (a, d, e) are consistent with independence relations holding in empirical data. The set of these consistent graphs defines a Markov equivalent class represented as CPDAG (c).

.

2.2.3 Modeling with complexity

Complementary to methods leveraging conditional independence tests described in the previous section, *score-based methods* rely on a global score evaluating the relative merit of alternative causal models to explain the empirical data, subject to complexity constraints. Such constraints are either "hard constraints" restraining the range of acceptable causal mechanisms (to linear models for example) or "soft constraints" such as a L_1 regularization on the number of edges.
In the general multi-variate setting, irrespective of the model search space, the Occam's razor principle has been formalized by Janzing and Scholkopf (2010) in terms of Kolmogorov complexity.

Referring the reader to Li and Vitányi (2013) for a comprehensive introduction, the Kolmogorov complexity of a probability distribution p of the continuous variable X defined on its domain of definition $dom(X)$ is the description length of the shortest program that implements its sampling process (Grünwald et al., 2008) (Eq. 14), noted $K(p)$ (also noted $K(p(\mathbf{x}))$ in the following by abuse of notation):

$$K(p) = \underset{s}{min}\, \{|s| \ : \ \text{for all } m \ \in \{1, 2, ...\}, x \in dom(X) : |\mathbb{U}(s, x, m) - p(x)| \leq 1/m\}\,, \quad (2.10)$$

with $\mathbb{U}$ a Universal Turing machine. Taking inspiration from (Janzing and Scholkopf, 2010), the key working hypothesis for complexity-based approaches is that the sought causal models are those with minimum Kolmogorov complexity of their conditional probabilities:

Working Hypothesis 1 (Algorithmic independence of statistical properties)**.** *(Janzing and Scholkopf, 2010)*

A necessary condition for causal model $\mathcal{G}$ (i.e., a DAG) to hold is that the shortest description of the joint density p be the sum of the shortest description of its causal mechanisms, up to a constant:

$$K(p(\mathbf{x})) \overset{+}{=} \sum_{j=1}^{d} K(p(x_j | x_{Pa(j;\mathcal{G})})), \quad (2.11)$$

Minimum Description Length A tractable approximation of the Kolmogorov complexity, the Minimum Description Length (MDL) is often used in practice, in particular in relation with bivariate causal discovery (Stegle et al., 2010; Budhathoki and Vreeken, 2017).

Let joint distribution p be defined after a candidate causal graph $\mathcal{G}$. The MDL associated with p measured with respect to a class $\mathcal{Q}$ of computable probabilistic models (e.g. exponential models), and an $i.i.d.n$-sample drawn after $p(\mathbf{x})$ noted $D = \{\mathbf{x}^{(1)} \ldots \mathbf{x}^{(n)}\}$ is defined as

(Barron and Cover, 1991):

$$MDL_r(\mathcal{G}, D) := \underset{q \ in \ \mathcal{Q}}{min} \left[K(q(\mathbf{x}, \mathcal{G})) + \sum_{\ell=1}^{n} \log \frac{1}{q(\mathbf{x}^{(\ell)}, \mathcal{G})} \right] \tag{2.12}$$

with $K(q(\mathbf{x}, \mathcal{G}))$ the number of bits needed to describe model q (that is computable by definition of $\mathcal{Q}$) and $\sum_{\ell=1}^{n} \log \frac{1}{q(\mathbf{x}^{(\ell)}, \mathcal{G}))}$ the number of bits in the coding length of the dataset with respect to q.

The MDL used in the following is the normalized MDL, divided by the size n of the iid-sample D:

$$MDL(\mathcal{G}, D) := \underset{q \ in \ \mathcal{Q}}{min} \left[\frac{1}{n} \ K(q(\mathbf{x}, \mathcal{G})) + \frac{1}{n} \sum_{\ell=1}^{n} \log \frac{1}{q(\mathbf{x}^{(\ell)}, \mathcal{G})} \right] \tag{2.13}$$

Causal inference with Minimum Description Length Overall, the Working hypothesis 1 states that the Kolmogorov complexity of the true graph $\mathcal{G}$, and the MDL-based approximation MDL$(\mathcal{G}, D)$ thereof, are minimal. If the minimal MDL is reached for a *unique* DAG $\mathcal{G}^*$, this graph is therefore the sought causal model under the assumptions made. Note however that the unicity of the solution is not guaranteed.

A well-known example is the linear bivariate Gaussian model, with $Y = X + E$ and $X \perp\!\!\!\perp E$ with X and E Gaussian variables, illustrated on Figure 2.4. As established by Mooij et al. (2016), there exists two models q_1 and q_2 such that $p(x) = q_1(x)q_1(y|x) = q_2(y)q_2(x|y)$ with exact same *complexity* (same structure and same number of parameters). In such cases, MDL$(X \to Y, D)$ and MDL$(Y \to X, D)$ are equal in the large sample limit and the causal graph remains undetermined.

Approximations with information criteria (AIC, BIC) As the Kolmogorov complexity is not computable however, the fitness-complexity of a proposed model is usually estimated using the *Akaike Information Criterion* (AIC) or the *Bayesian Information Criterion* (BIC):

$$AIC = 2k - 2\ln(L) \tag{2.14}$$

$$BIC = \ln(n)k - 2\ln(L) \tag{2.15}$$

where n represents the number of samples, k represents the number of parameters of the model (proxy for the complexity of the model) and L represents the maximum of likelihood of the model:

$$L = L(X|\theta) = \max_{\theta} \frac{1}{n} \sum_{i=1}^{n} f(x_i|\theta) \tag{2.16}$$

with θ denoting the parameters of the model and f representing the density function. The maximum likelihood value accounts for how the model fits the data distribution.

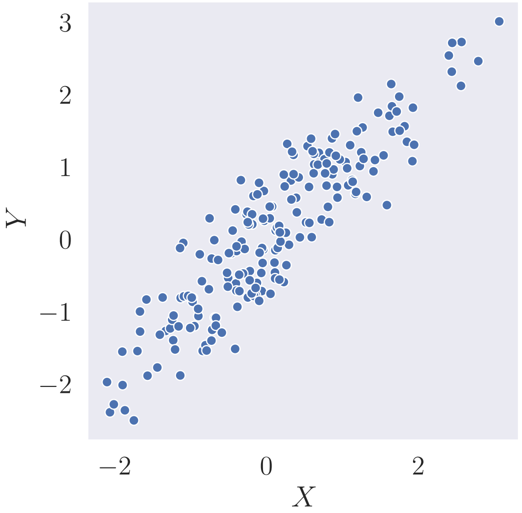


Figure 2.4: Illustration of an unidentifiable case by MDL, $Y = X + E$, with X, E sampled from a normal distribution and independent.

While considering Gaussian data, practitioners compare two candidate causal graphs $\mathcal{G}$ and $\mathcal{G}'$ using AIC by computing those criteria in the following manner, under the additional assumptions of model errors being i.i.d. following a normal distribution and the derivative of the log likelihood with respect to the true variance is zero:

- For both graphs, compute their AIC score:

  (a) Compute the residuals sum of squares[3] for all variables following the graph:

  $$R = \sum_{i=1}^{d} (X_i - f_i(Pa(X_i)))^2$$

  where $Pa(X_i)$ the parents of X_i in the graph, and $f_i(Pa(X_i))$ the approximation of X_i learned by regression on $Pa(X_i)$.

  (b) $AIC = 2k + n \ln(R/n)$
  where k corresponds to the number of edges in the graph and n the number of samples.

- Finally, the model with **minimal** AIC score (between $\mathcal{G}$ and $\mathcal{G}'$) is selected as the best one.

[3] the likelihood score is resumed to a sum of squares in the case of Gaussian data.

2.3 Learning Markov equivalence class with local search algorithms

We now move to describing state-of-the-art algorithms exploiting the principles introduced in the first part of this chapter.

The first category approach relies on CSA, CMA and CFA assumptions (Section 2.1.3) and uses conditional independence (CI) relations[4] in order to identify the Markov equivalence class of the sought Directed Acyclic Graph (CPDAG). This first family of algorithms recovering the CPDAG of a functional causal model from continuous data includes constraint-based methods, score-based methods, and hybrid methods (Drton and Maathuis, 2016).

2.3.1 Constraint-based methods

Constraint-based methods leverage conditional independence tests to identify a skeleton of the graph and v-structures. Then, constraint propagation is used to gradually orient other edges, and finally output the CPDAG of the graph as detailed in Section 2.2.2.

Spirtes-Glymour-Scheines (SGS) The SGS algorithm (Spirtes et al., 1993) is one of the first algorithms to leverage conditional independencies for causal discovery. It is known to be quite computationally expensive as almost all conditioning sets for all variables are considered in the skeleton phase. This algorithm led to the PC algorithm, which represents a computationally efficient version of SGS, thanks to its optimized arrangement of variables.

Peter-Clark (PC) Constraint-based methods are best exemplified with the celebrated PC algorithm (Spirtes et al., 1993): under CSA, CMA and CFA, and assuming *that all conditional independences have been identified*, PC returns the CPDAG of the functional causal model, respecting all v-structures (Fig. 2.3(c)). In practice, PC uses a threshold on p-value to select the edges in the graph (Algorithm 1).

Replacing the Pearson p-value test with non-parametric independence tests based on machine learning such as kernel-based conditional independence tests (Zhang et al., 2012; Strobl et al., 2017) is becoming increasingly popular. See Section 2.2.2 for more details.

Fast Causal Inference (FCI) The FCI algorithm extends PC and relaxes the *causal sufficiency* assumption (Spirtes et al., 1999). FCI starts with the PC algorithm (Initial skeleton recovery with conditional independencies and v-structure identification), but adds a second structure recovery phase meant to sharpen the structure by leveraging possible d-separation sets. Finally, more rules are added in the constraint propagation phase (Zhang, 2008), taking account of the relaxation of the causal sufficiency assumption.

[4] Proofs of model identifiability generally assume the existence of an “oracle” yielding the ground truth CIs.

Algorithm 1: The PC algorithm

```
Data: Observational data $\mathbf{X} = X_1, \dots, X_d$ sampled i.i.d. from $P(\mathbf{X})$
Input: p-value threshold $\alpha$
Result: CPDAG of the Causal Graph
Start from a fully connected graph $\mathcal{G}$
$n = 0$
while $n < d - 2$                    // Skeleton recovery with conditional independence
 do
    forall $i, j$ in $[1, d]$ such that $i < j$ and $X_i$ and $X_j$ are adjacent do
        forall Subsets of variables $S$ adjacent to $X_i$ and $X_j$ such that $Card(S) = n$ do
            Compute the p-value $\beta_{ij|S}$ for testing the independence of $X_i, X_j|S$
            if $\beta_{ij|S} > \alpha$ then
                Remove $X_i \rightarrow X_j$ from $\mathcal{G}$
                Record $S$ in SeparationSet$(X_i, X_j)$
            end
        end
    end
    n = n+1
end
forall $i, j, k \in [1, d]$ such that $(X_i, X_j)$ and $(X_k, X_k)$ are adjacent but not $(X_i, X_k)$ do
    if $X_j \notin SeparationSet(X_i, X_k)$                    // Orient v-structures
     then
        Orient $X_i - X_j - X_k$ as $X_i \rightarrow X_j \leftarrow X_k$
    end
end
while Edges can be oriented                    // Constraint propagation phase
 do
    forall $i, j$ in $[1, d]$ such that $X_i$ and $X_j$ are adjacent and the edge $X_i - X_j$ is not oriented
     do
        if $\exists k \neq i, j \in [1, d]$ such that $X_k \rightarrow X_i$ and $X_k, X_j$ are not adjacent in $\mathcal{G}$ then
            Orient $X_i - X_j$ as $X_i \rightarrow X_j$
        end
        if There exists a directed path from $X_i$ to $X_j$ in $\mathcal{G}$ then
            Orient $X_i - X_j$ as $X_i \rightarrow X_j$
        end
    end
end
return $\mathcal{G}$
```

Really Fast Causal Inference (RFCI) RFCI (Colombo et al., 2012) optimizes FCI for handling larger DAGs with latent variables. Indeed, FCI can show itself quite expensive due to the second structure recovery phase. In counterpart for weaker consistency results, RFCI replaces this latter phase using additional testing of soundness before orienting edges in the v-structure identification phase.

Limitations. Such constraint-based algorithms suffer from three drawbacks. First, their dependency on conditional independence tests make them data hungry as the required datasize exponentially increases with the number of variables; making them highly dependent on the complexity of the scoring function[5]. Secondly, the propagation rules used to direct edges are prone to error propagation. Finally, the number of conditional independence tests required grows exponentially with the number of variables, preventing their scalability beyond small problems (a few dozen variables).

2.3.2 Score-based methods

Score-based methods aim at finding the best CPDAG in the sense of some global score: using search heuristics, graph candidates are iteratively evaluated using a scoring criterion such as AIC or BIC (Section 2.2.3) and compared with the best graph obtained so far.

Greedy Equivalence Search (GES) The Greedy Equivalent Search (GES) algorithm (Chickering, 2002) aims to find the best CPDAG in the sense of the Bayesian Information Criterion (BIC). The CPDAG space is navigated using local search operators, e.g. *add edge*, *remove edge*, and *reverse edge*. GES starts with an empty graph. In a first forward phase, edges are iteratively added to greedily improve the global score. In a second backward phase, edges are iteratively removed to greedily improve the score (Algorithm 2). Under CSA, CMA and CFA assumptions, GES identifies the true CPDAG in the large sample limit, if the score used is decomposable, score-equivalent and consistent (Chickering, 2002).

Fast Greedy Equivalence Search (FGES) More recently, Ramsey (2015) proposed a GES extension called Fast Greedy Equivalence Search (FGES) algorithm aimed to alleviate the computational cost of GES. It leverages the decomposable structure of the graph to optimize all the subgraphs in parallel. This optimization greatly increases the computational efficiency of the algorithms, enabling score-based methods to run on millions of variables which is unfeasible for constraint-based methods.

Limitations These methods rely on exploration heuristics, which hardly support the efficient exploration in the graph space, as the scoring function might not be smooth according to the distance to the true graph. Other heuristics have though been developed to explore more efficiently the graph space (Glover and Taillard, 1993; Tsamardinos et al., 2006; Better et al., 2007).

[5] having a scoring function such as kernel-based independence tests leads to having algorithms unable to scale above 50 variables

```
Algorithm 2: The GES algorithm
Data: Observational data $\mathbf{X} = X_1, \dots, X_d$ sampled i.i.d. from $P(\mathbf{X})$
Input: Constraint parameter $\lambda$, Scoring criterion $C$
Result: CPDAG of the Causal Graph
Start from an empty graph $\mathcal{G}$
Init score $S = C(\mathcal{G})$
while $S$ does improve                                   // Forward Phase
 do
    for $i, j \in [1, d], i \neq j$ do
        if $X_i \to X_j$ and is not in $\mathcal{G}$ and adding it does not create any cycle then
            Let $\mathcal{G}'$ be the graph with $X_i \to X_j$
            if $S < C(\mathcal{G}')$ then
                $\mathcal{G} \leftarrow \mathcal{G}'$
                $S = C(\mathcal{G}$
            end
        end
    end
end
while $S$ does improve                                   // Backward Phase
 do
    for $i, j \in [1, d]$ such that $X_i \to X_j$ is in $\mathcal{G}$ do
        Let $\mathcal{G}'$ be the graph without $X_i \to X_j$
        if $S < C(\mathcal{G}')$ then
            $\mathcal{G} \leftarrow \mathcal{G}'$
            $S = C(\mathcal{G}$
        end
    end
end
return $\mathcal{G}$
```

2.3.3 Hybrid algorithms

Hybrid algorithms combine ideas from constraint-based and score-based algorithms. According to Nandy et al. (2015), such methods often use a greedy search like the GES method on a restricted search space for the sake of computational efficiency. This restricted space is defined using conditional independence tests.

Max-Min Hill climbing (MMHC) Tsamardinos et al. (2006) firstly build the skeleton of a Bayesian network using conditional independence tests (using constraint-based approaches) and then performs a Bayesian-scoring hill-climbing search to orient the edges (using score-based approaches). The skeleton recovery phase, called Max-min Parents and Children (MMPC) selects for each variable its parents and children in the dataset. Note that this task is different from recovering the Markov blanket of variables as the spouses are not selected.

The orientation phase is a hill-climbing greedy search involving 3 operators: add, delete and reverse edges.

Greedy Fast Causal Inference (GFCI) algorithm proceeds in the other way around, using FGES to get rapidly a first sketch of the graph (shown to be more accurate than those obtained with constraint-based methods), then using the FCI constraint-based rules to orient the edges in presence of potential hidden confounders (Sec. 2.1.2) (Ogarrio et al., 2016). GFCI assumes CMA, CFA and acyclicity of the causal graph.

Limitations The hybrid algorithms try to merge both score-based and constraint-based algorithms in order to combines the advantages from both types of approaches. This leads to hybrid algorithms seeking linear FCMs; but supposedly performing better than their other non-hybrid counterparts in the linear case.

2.4 Learning sparse linear Gaussian Bayesian Networks with global search algorithms

Addressing the above limitation, other methods have been proposed to simultaneously learn the causal mechanisms and the causal graph structure. These methods are restricted to linear functional causal models, referred to as *Functional Causal Models*(FCMs), and can be formulated in terms of linear algebra.

2.4.1 Matrix formulation of linear FCMs

The general formulation of the FCM (Eq. 2.2) in the case of linear causal mechanisms on centered Gaussian variables is (Aragam et al., 2017):

$$X_j = \beta_{\mathbf{j}}{}^T \mathbf{X} + E_j, \quad \text{for } j = 1, \ldots, d. \tag{2.17}$$

with $\beta_{\mathbf{j}} = (\beta_{1,j}, .., \beta_{d,j}) \in \mathbb{R}^d$, $\beta_{j,j} = 0$ to avoid feedback loops, and $E_j \sim \mathbb{N}(0, \omega_j^2)$.

Letting B denote the real-valued $d \times d$ matrix $(\beta_{i,j})$ and E the $d \times d$ diagonal matrix defined from the E_j, Eq. (2.17) is rewritten as:

$$\mathbf{X} = B^T \mathbf{X} + E \tag{2.18}$$

defining the so-called *functional causal model* (FCM) for $\mathbf{X}$, with B being the weighted adjacency matrix of the directed graph $\mathcal{G}$. This linear FCM formulation has been used in pioneering works focusing on relaxing the causal assumptions (Hoyer et al., 2006, 2008; Anandkumar et al., 2013).

2.4.2 Learning undirected graphical models with regularization

Such linear FCMs have first been solved in the literature to recover an undirected graph, encoding all conditional independence relations between variables. Let us consider the set of random variables $\mathbf{X} = (X_1, \dots, X_d)$. The information on conditional independence between variables can be schematized by an undirected graph $\mathcal{G}$ such that $X_i \perp\!\!\!\perp X_j | \mathbf{X}_{\setminus i,j}$ *iff* there is no edge between the nodes X_i and X_j. Now let $\mathbf{X} = (X_1, \dots, X_d)$ be centered and multivariate normal with positive definite covariance matrix Σ. Let the matrix $K = \{k_{i,j}\}_{i,j=1..d} = \Sigma^{-1}$ denote the inverse, referred to as the precision matrix of the Gaussian vector $\mathbf{X}$. In the multivariate Gaussian case, a well known property is that the entry $k_{i,j}$ of the precision matrix is equal to zero if and only if $X_i \perp\!\!\!\perp X_j | \mathbf{X}_{\setminus i,j}$ (Lauritzen, 1996) (more Section 2.2.1). Accordingly a Gaussian conditional independence graph can be estimated by determining the zero entries of the inverse covariance matrix. The inverse covariance matrix from the observational data is usually recovered in the literature by minimizing the negative log-likelihood of the matrix of observations $\mathbf{X} \in \mathbb{R}^{n \times d}$, available in closed form in the multivariate Gaussian case:

$$L(K|\mathbf{X}) = -\frac{n}{2}\log\ \det\ K + \frac{1}{2}\mathrm{tr}(K\ \mathbf{S}), \tag{2.19}$$

with $\mathbf{S} = \mathbf{X}^T\mathbf{X}$. Banerjee et al. (2008) proposed the graphical lasso (*glasso*) estimator of the inverse covariance matrix by adding a L_1 penalization term to this objective, in order to enforce the sparsity of matrix K.

$$\hat{K}^{gl} = \underset{K}{argmin}\ L(K|\mathbf{X}) + \lambda\|K\|_1, \tag{2.20}$$

where $\|K\|_1$ corresponds to the K matrix L_1 norm:

$$\|K\|_1 = \sum_{i,j} |k_{i,j}| \tag{2.21}$$

The optimization of Eq. 2.20 leads to efficient and scalable algorithms such as the coordinate-descent algorithm (Friedman et al., 2008), supporting computation of K^{gl} up to a few thousand variables.

2.4.3 From undirected to directed graphs

When searching for a directed graph, Eq. (2.18) must be solved subject to acyclicity constraints on B, yielding a much more difficult non-convex optimization problem. Letting Ω denote the covariance matrix of noise vector E (the diagonal matrix $(\omega_1, \dots \omega_d)$) and I the d-dimensional identity matrix, the multivariate Gaussian distribution of $\mathbf{X}$ is $\mathcal{N}(0, \Sigma)$ with (Aragam and Zhou, 2015):

$$\Sigma = (I - B)^{-1^T}\Omega(I - B)^{-1} \tag{2.22}$$

The $(I - B)$ intervenes in order to avoid self-loops, as all diagonal terms B_{ii} are set to 0 in this fashion.

Letting $K = \Sigma^{-1}$ be the inverse covariance matrix of $\mathbf{X}$, it follows:

$$K = (I - B)\Omega^{-1}(I - B)^T. \tag{2.23}$$

Letting $S = X^T X$ denote the empirical estimate of Σ, the likelihood estimator of this Gaussian graphical model to be minimized is:

$$L(K) = -\frac{n}{2}\text{log det}K + \frac{1}{2}\text{tr}(K\ S). \tag{2.24}$$

The FCM (matrix B and variances ω_j) is identified by minimizing Eq. 2.24, augmented with a regularization term $\rho_\lambda(B)$, set to an L_1 penalty (Tibshirani, 1996), a group norm penalty (Yuan and Lin, 2006) or an L_0 penalization (Van de Geer et al., 2013; Zheng et al., 2018b):

$$\underset{B \in \mathbb{B}}{min}\ L(K) + \rho_\lambda(B) \tag{2.25}$$

where $\mathbb{B} \subset \mathbb{R}^{d \times d}$ is the set of weighted adjacency matrices representing directed acyclic graphs. The L_1 formulation is the classic formulation corresponding to the lasso. Its optimization is rather smooth, but the edges have numerical values thus needing a threshold value in order to obtain a sparse graph. Group norm penalizations allow the models to select variables by groups and not individually, thus are more fit to variable selection problem for graphs. More recently, L_0 penalizations are becoming increasingly popular as the final state of the optimized model corresponds directly to the proposed graph, at the cost of a harder optimization procedure as the L_0 penalization is less smooth than other proposed counterparts.

The optimization problem described by Eq. 2.25, known to be NP-hard (Chickering et al., 2004), is tackled using either approximate or exact algorithms. Exact algorithms tackle the combinatorial optimization problem (see e.g. the linear integer programming approach proposed by Bartlett and Cussens (2017)); these hardly handle more than fifty variables even in the linear Gaussian case. By contrast, approximate methods can scale up to thousands of nodes (Aragam and Zhou, 2015; Scanagatta et al., 2015). Notably Zheng et al. (2018b) formulate the structure learning problem as a global continuous optimization problem over real matrices, avoiding the combinatorial search in the DAG space through a new characterization of acyclicity for the adjacency matrix B. This approach will be detailed in Section 5.1.3 as our proposed SAM takes inspiration from it.

Limitations. The methods based on constrained and sparse optimization mostly recover the Markov equivalence class of the sought DAG. In some domains such as biology, where the sought $\mathcal{G}$ graph is star-shaped and does not include v-structures, these methods are unable to orient the edges. Indeed, conditional independence does not allow to distinguish causal structures if v-structures are absent in the structure. Moreover, these methods are assuming linear FCMs, an assumption which is often not verified in the case of real-world data.

2.5 Exploiting asymmetry between cause and effect

New methods, taking into account the full information from the observational data (Spirtes and Zhang, 2016) such as data asymmetries induced by the causal directions, have been proposed and primarily applied to the bivariate DAG case[6], referred to as cause-effect pair problem (Hoyer et al., 2009; Daniusis et al., 2012; Mooij et al., 2016). The reader is referred to Statnikov et al. (2012); Mooij et al. (2016); Guyon et al. (2019) for a thorough presentation of the bivariate problem. Cause-effect pair algorithms most generally achieve model selection, and determine the best trade-off among the complexity of the model and its data fitting score. Four strategies are distinguished: i) imposing restrictions on the sought model; ii) computing a smooth trade-off between data fitting and complexity scores; iii) exploiting independence between cause and mechanism; iv) using machine learning methods.

2.5.1 Restricting the class of causal mechanisms

This family of pairwise models[7] rely on restricting oneself to a simple class of models, in order to favor the identifiability of the model, that is, a single one of the two models $X \to Y$ and $Y \to X$ is attached a good score. Otherwise, the dataset is either said to be unidentifiable (the model fits the data in both causal directions with good quality), or does not fit in either direction (the model does not fit the data with enough quality). These models allows for theoretical identifiability on pairs, but the considered class of models is often too restrictive for real-world data. This section takes inspiration from Guyon et al. (2019).

Additive Noise Model(ANM) The celebrated Additive Noise Model (Hoyer et al., 2009) models the data generative process as

$$Y = f(X) + E$$

with f a possibly non-linear function and E a noise independent of X. ANM is generally identifiable (i.e. ANMs $X \to Y$ and $Y \to X$ do not fit the data equally well in the large sample limit) except in specific cases, including linear FCMs with Gaussian distribution of the cause and Gaussian additive noise. More precisely, for the two alternatives $X \to Y$ and $Y \to X$, the estimated mechanisms $\hat{f}_Y$ and $\hat{f}_X$ are obtained via Gaussian process regressions. These estimated regression functions are used to estimate the residuals $\hat{n}_Y = y - \hat{f}_Y(x)$ and $\hat{n}_X = x - \hat{f}_X(y)$. The scores $S_{X \to Y}$ and $S_{Y \to X}$ correspond respectively to kernel HSIC independence test (Gretton et al., 2005a) between $\hat{n}_Y$ and x (for $X \to Y$) and between $\hat{n}_X$ and y (for $Y \to X$).

Example 3. *Considering the pair $Y = X + E$, with $X \sim U(0, 1)$ and $E \sim U(0, .9)$, the resulting joint distribution is represented on Figure 2.5.*

[6] Note that in the bivariate case, both $X \to Y$ and $Y \to X$ DAGs are Markov equivalent; the former methods do not apply.

[7] applicable on datasets of only two variables X and Y

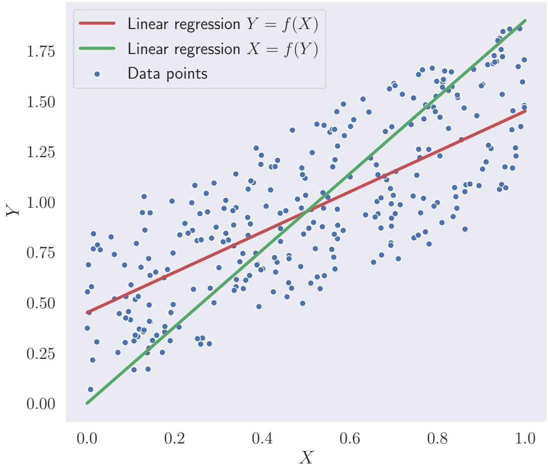
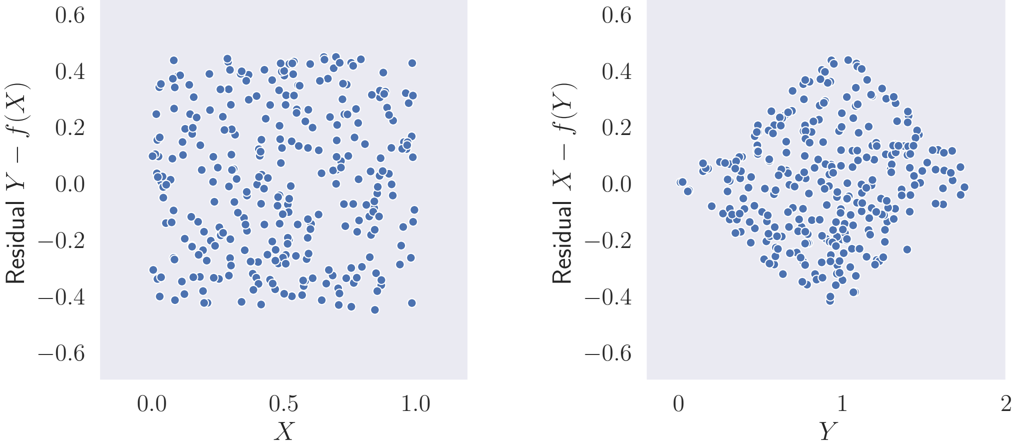


Figure 2.5



(a)

(b)

Figure 2.6: Scatter plots of residuals of the regression with the considered cause.

For each causal direction evaluated ($Y \to X$ and $X \to Y$), the regression of the effect by the considered cause is performed (Fig. 2.5, red and green lines), and the independence between the residuals and the cause are computed, as shown on Fig. 2.6. The residuals from the $X \to Y$ model is independent from the cause (Fig 2.6a), compared to the $Y \to X$ model (Fig 2.6b), thus showing that the ANM model identifies the pair as being an $X \to Y$ causal pair.

Post-Nonlinear Model (PNL) The Post-NonLinear model (PNL) generalizing ANM and taking into account nonlinear interactions has been proposed by Zhang and Hyvärinen (2010, 2009) to handle non-additive noise and achieve identifiability for more and more complex models, thus being able to explain a broader number of causal pairs:

$$Y = g(f(X) + E)$$

with g an invertible function on the top of the additive noise.

The counterpart of having a more complex model is to increase the number of non-identifiable cases: as the model gets more and more complex, the number of pairs that could fit the model in either way increases. From the ANM to the the PNL model, the number of unindentifiable cases gets from 1 (linear Gaussian with Gaussian noise) to 5 (Zhang and Hyvärinen, 2009).

For example, the causal relationship $Y = e^{X+E}, X, E \sim U(0,1)^2$, unindentifiable with the ANM model because of the non-linearity of the noise, is identifiable with the PNL model as the exponential function is invertible and $X + E$ correponds to an additive model.

2.5.2 Smooth complexity/fit tradeoff

Instead of restricting the class of models, another approach is to achieve a tradeoff between the fitness of the model and the complexity of the model. The model complexity can either be measured in various ways, such as the number of parameters of the model, or controlled through a regularization of the causal mechanisms. The benefit of these approaches is that they allow all kinds of data, as they avoid any sharp restriction on the model search space, implicitly controlled from the complexity.

Gaussian Process Inference (GPI) GPI (Stegle et al., 2010) is a non-parametric Bayesian approach, optimizing a sum of a data fitting term and a complexity term. Specifically two Bayesian generative models, one for $X \rightarrow Y$ and one for $Y \rightarrow X$, are built, where the distribution of the cause is modeled with a Gaussian mixture model, and the causal mechanism f is a Gaussian process. The causal direction associated with the model with minimal code length according to the Minimum Message Length principle (MML) is retained.

Classifier Two-Sample Tests (C2ST) C2ST (Lopez-Paz and Oquab, 2016) represents a neural network follow-up of GPI: it proceeds by training conditional generative adversarial networks in both causal directions and retain the one that best fits the observational data, thus replacing the fit score with a neural network. Our approach SAM (Chapter 5) takes inspiration from this approach to extend it to the graph setting.

2.5.3 Exploiting independence between cause and mechanism

Other approaches are based on the exploitation of conditional distributions. Formally, assuming that $X \rightarrow Y$, Sgouritsa et al. (2015) conjecture that the marginal probability distribution of the cause $P(X)$ is independent of the causal mechanism $P(Y|X)$; hence estimating $P(Y|X)$ from $P(X)$ should be "harder“ than estimating $P(X|Y)$ based on $P(Y)$. The same conjecture underlies Mitrovic et al. (2018)' approach, considering that the conditional distribution $\{Q_{Y|X=x^i}\}_{i=1}^n$ should be less sensitive to the different values x^i taken by the variable X, compared to the conditional distribution $\{Q_{X|Y=y^i}\}_{i=1}^n$ depending on the different values y^i, where the conditional distributions are estimated using conditional RBF kernel embeddings into the Hilbert space of infinitely differentiable functions.

2.5.4 Machine learning approaches

The two Causality challenges (Guyon, 2013, 2014) pioneered the formalization of causal inference as a supervised machine learning problem. During the two challenges, 16,200 labelled pairs of variables $S_i = \{(X_i, Y_i, \ell_i)\} = \{(x_{ij})_{j=1}^{n_i}, (x_{ij})_{j=1}^{n_i}, \ell_i\}$ were released, where each pair is associated with the label of the associated causal relation ranging in $X_i \rightarrow Y_i$, $Y_i \rightarrow X_i$, $X_i \perp\!\!\!\perp Y_i$, $X_i \leftrightarrow Y_i$ (presence of a confounder). Each (X_i, Y_i) pair is described by a sample of the underlying joint distribution. The classifier trained from these examples was used to estimate the causal relation for new pairs of variables, with good results.

Kaggle Cause-Effect Pair Challenge The Cause-Effect Pair Challenge organized in 2013 on the Kaggle platform (Guyon, 2013) is the first competition focusing on pairwise causal discovery, pioneering the supervised learning setting for pairwise causality. The training data involves 12,081 pairs of variables (Examples on Fig. 2.7); the test data involves 4,050 other pairs of variables. Each pair of variables is associated its ground truth causal label, ranging in four classes respectively corresponding to $X \rightarrow Y$, $X \leftarrow Y$, $X \perp\!\!\!\perp Y$ and $\exists Z, X \leftarrow Z \rightarrow Y$.

The training and test pairs of variables included circa 18% real pairs and 82% artificial pairs with continuous, categorical and binary variables.

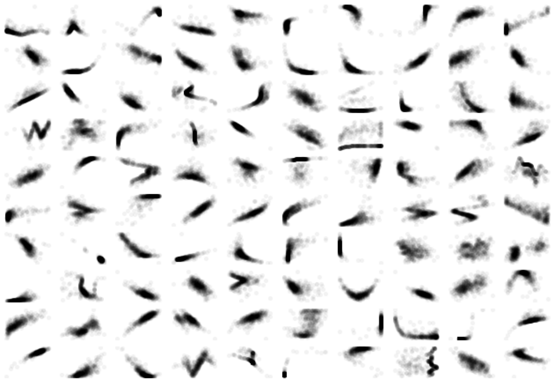


Figure 2.7: Example of bivariate causal datasets from the challenge

Codalab Fast Causation Coefficient Challenge Most approaches submitted to the Cause-Effect Pair Challenge involve a heavy feature construction process, associating to each sample of any joint distribution $P(X, Y)$ a real-valued vector of feature values (up to 20,000 features), on the top of which a standard learning algorithm is used. Due to the high computational effort required to achieve this statistical feature construction, a follow-up two-month challenge, the Fast Causation Coefficient challenge has been proposed (Guyon, 2014), aimed at algorithms achieving a reasonable trade-off between predictive causal accuracy and computational efficiency. The assessment of algorithms was made possible as the Fast Causation Coefficient challenge (with same setting as the previous challenge) was hosted on the Codalab challenge platform. This Codalab platform allows participants to submit an executable code, that can therefore be assessed in a fair and reproducible way.

Randomized Causation Coefficient (RCC) To leverage the representative power of kernel embeddings, RCC (Lopez-Paz et al., 2015) combines kernel-based embedding for feature construction with pairwise causal discovery. Considering the dataset of empirical distributions $S = \{S_i\}_{i=1}^n$, a kernel mean embedding allows to project these empirical distributions into the same Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}_k$. To obtain a homogeneous and low dimension embedding, Lopez-Paz et al. (2015) uses random cosine based embeddings that approximate empirical kernel mean embeddings in low dimension:

$$\mu_{k,m}(P_{S_j}) = \frac{2C_k}{|S|} \sum_{x_{ij}, y_{ij} \in S_j} (\cos(w_j^x * x_{ij} + w_j^y * y_{ij} + b_j))_{j=1}^m \in \mathbb{R}^m \tag{2.26}$$

where $\{w_j, b_j\}_{j=1}^m$ are the kernel parameters sampled i.i.d. in $\mathcal{N}_{0,2} \times [0, 2\pi]$, as well as their number m defining the number of dimensions of the output space, P_S is the empirical distribution, and $C_k = \int_{\mathcal{Z}} p_k(w)dw$, with $p_k : \mathbb{R}^d \mapsto \mathbb{R}$ the positive and integrable Fourier transform of the chosen kernel k, equal to 1 in this case. Next, a random forest classifier is trained on the built features and predict the causal direction of unseen distributions. This approach took the second place on the fast causation challenge (Guyon, 2014).

Jarfo (Fonollosa, 2016), one of the best performing algorithms over both challenges, operates as follows:

1. A type-dependent preprocessing of the input variables is applied ; Numerical variables are normalized and binned along 19 intervals to compute features such as discrete mutual information or discrete entropy. Categorical variables are relabelled with sorted probabilities to obtain numerical variables.

2. Information theoretic measures and other causally relevant features are computed; including discrete entropy, mutual information, divergence, and standard deviation on conditional distributions (CDS). Extra features, commonly used in conditional discovery, are computed: Hilbert Schmit Independence Criterion (HSIC), moments, a number

of pairwise discovery scores (IGCI score (Janzing et al., 2012), ANM, PNL, etc), a Pearson correlation and a polynomial fit on the variables, and the obtained residual of the fit.

3. The computed features are mapped from onto $\mathbb{R}^p$, noting the mapping function $\phi(S_i)$. The problem is shifted from a learning problem over distributions to a regular classification problem of data points.
4. A gradient boosting classifier based on the computed previous features is trained on the examples $\{\phi(S_i), \ell_i\}$ using a 10-fold cross-validation. This approach is rather popular due to its robust performance/computational cost ratio.

GEne Network Inference with Ensemble of trees (GENIE3) GENIE3 (Irrthum et al., 2010) approaches the problem of network inference as a feature selection problem by using random forests to perform the node selection: for each variable, a tree ensemble method is performed to select the node that help to model the target variable. By combining all the results, a final network is provided. Note that the resulting graph might contain cycles except for self-interactions, as the tree ensembles results are simply concatenated.

Limitations. On the one hand, bivariate methods can be used to independently orient each edge (with no propagation and thus no risk of error propagation). On the other hand, bivariate methods do not have a global view of the variable set, and specifically cannot take advantage of v-structures. Typically, when considering the v-structure $X \rightarrow Z \leftarrow Y$, a bivariate model based on cause-effect asymmetry would miss both causal relationships in the linear Gaussian case (linear mechanism, Gaussian distribution of causes and noise).

2.6 Exploiting conditional independence and distributional asymmetries

Most interestingly, the linear Gaussian case is the easiest one to deal with for the local and global learning algorithms (Sections 2.3, 2.4), while it is impossible to solve for the category of cause-effect pair algorithms (Section 2.5). On the contrary, when data employs non-linear causal mechanisms with complex interactions, causality modeling is most generally easier for the 3rd category and more difficult for the 1st and 2nd categories due in particular to the complexity of evaluating log-likelihood scores for non-Gaussian data.

These remarks inspire some hybrid approaches, exploiting the complementarity of the methods, and specifically returning partially directed graphs *and* exploiting the interactions between all variables.

An extension of the bivariate post-nonlinear model (PNL) proposed by Zhang and Hyvärinen (2009) illustrates such a hybrid approach: an FCM is trained for any plausible causal structure, and each model is tested *a posteriori* for the required independence between errors and causes. Its main limitation is its super-exponential cost with the number of variables (Zhang and Hyvärinen, 2009).

Another hybrid approach, proposed by Zhang and Hyvärinen (2009), uses a constraint based algorithm to identify a Markov equivalence class, and thereafter uses bivariate modelling to orient the remaining edges. For example, the constraint-based PC algorithm can identify the v-structure $X_3 \rightarrow X_5 \leftarrow X_4$ in an FCM (Fig. 2.3), enabling the bivariate PNL method to further infer the remaining arrows $X_1 \rightarrow X_2$ and $X_1 \rightarrow X_3$. Note that an effective combination of constraint-based and bivariate approaches requires a final verification phase to test the consistency between the v-structures and the edge orientations.

Causal Additive Models (CAM) The CAM algorithm (Bühlmann et al., 2014) leverages both conditional independence relations and pairwise asymmetries. CAM extends the pairwise additive model (ANM) (Hoyer et al., 2009) to the multi-variate setting, modeling the FCM as:

$$X_i = \sum_{k \in \mathrm{Pa}(i;\mathcal{G})} f_k(X_k) + E_i, \text{ for } i = 1, \ldots, d \tag{2.27}$$

CAM models the causal mechanisms with Gaussian Processes, supporting the identification of complex mechanisms. It also involves an initial feature selection step that allows to restrict the search space of the algorithm.

Partial Conclusion

The state of the art in observational causal discovery without interventional data relies on the notion of simplicity: no other leverage is available from the data without experiments. It is either considered in the structural form represented by the number of edges in the graph (Section 2.3, Section 2.4, Section 2.6) or in the functional form, i.e. the complexity of the causal mechanisms (Section 2.5, Section 2.6). The approaches employ various strategies to recover the causal graph: use a local or a global optimization procedure as a structure learning approach; or leverage the complexity of the causal functions without restricting causal models to linear functions (Section 2.5), as above-mentioned structure learning approaches do. Finally, more recent approaches combine structural and functional learning to increase their predictive performance.

From a scalability the point of view, local learning methods (Section 2.3) are limited to a few hundred variables for constraint-based methods and much more for score-based methods: up to a million variables for FGES (Ramsey et al., 2017). Global learning methods scale fairly well, up to a few thousand variables (Section 2.4), whereas methods only leveraging asymmetries (Section 2.5) are limited to two variables. Finally, hybrid methods extending these methods to graphs (Section 2.6) can usually scale up to a few hundred variables.

Most linear approaches are quite computationnally efficient, but the assumption of linear mechanisms hurts the accuracy of the algorithms. Relaxing this assumption by allowing for more causal mechanisms often leads to adding significant computational complexity to the algorithms: non-linear independence tests or non-linear regressors. However, even the recent approaches are still restrictive towards the distribution of the data and the causal mechanisms; those assumptions often not verified nor verifiable in the case of real data. Therefore, the performance of the algorithms may vary depending on the nature of the data. Introducing neural networks to causal discovery allows to have a really large class of supported causal mechanisms and data distributions; the modularity of neural networks allows at the same time to keep relative computational efficiency.

Chapter 3

Artificial Neural Networks

In this thesis, one of the objective is to leverage the expressivity and modularity of neural networks for causal discovery, not as traditional classification or regression problems, but as generative models: the quality of a causal graph candidate is evaluated through the quality of the data generated by the neural network modeling the causal graph. After having given a general explanation of neural networks, this chapter introduces all the tools and techniques that will be used in the proposed approaches.

Deep artificial Neural networks (NNs) represent the major breakthrough in recent machine learning as highlighted in various results (Krizhevsky et al., 2012; Goodfellow et al., 2014). Their popularity also comes from their good performance, their computational efficiency because of their portability to Graphical Processing Units (GPUs) (Raina et al., 2009) and their ease of use: any differentiable type of operation can be added in the computational graph.

In a nutshell, a NN is an algorithm composed of multiple layers of weighted sums, made out of learnable weights, and non-linear activation functions. All the learnable weights are optimized to satisfy the objective function (typically linked to prediction accuracy) using gradient descent. In this chapter, we will present the basics on artificial neural networks with multilayer perceptrons (Section 3.1), explain stochastic gradient descent (Section 3.2), describe new types of architecture in neural networks (Section 3.3), and finally present generative adversarial networks (Section 3.4).

Recent artificial neural network architectures stem from the Perceptron (Rosenblatt, 1958), a classifier linear in its paramaters, representing the simplest architecture of neural network, with only one trainable neuron.

Considering a d dimensional input $\mathbf{x}$, the output $\widehat{\mathbf{y}}$ of a linear Perceptron is:

$$\widehat{\mathbf{y}} = \sigma(\mathbf{x} \cdot W)$$

with σ being an activation function, such as hyperbolic tangent or the Heaviside function. Perceptrons can be trained in a number of manners. Weight updates W_i are usually proportional to the correlation between input x_i and output $\mathbf{y}$, as in the biologically-inspired rule of Hebb. One of the most successful algorithms in that family is the Widrow-Hoff algorithm, with update rule $W_i' = W_i + \alpha * (\mathbf{y} - \widehat{\mathbf{y}})x_i$, which can be derived as a stochastic gradient method to minimize the mean-square-error for linear units (no σ activation function.

One of the limits of the linear Perceptron is that it is unable to solve the XOR problem, which only non-linear classifiers can tackle. This led to the creation of non-linear neural network architectures, namely the multilayer perceptron in the end of the 80s.

3.1 Multilayer Perceptrons

Multi-layer perceptrons (MLPs) stack several layers f_i, each composed of many neurons with learning weights W_i and separated by non-linear functions σ_i.

σ_i is called **activation function** of the neuron, and is the element introducing non-linearity in the architecture. Examples of popular activation functions are the Rectified Linear Unit (ReLU), Exponential Linear Unit (ELU), Hyperbolic Tangent (Tanh), or Sigmoid.

Given an input $\mathbf{x}$, the output of a MLP with k layers $\widehat{\mathbf{y}}$ is:

$$\begin{cases} f_i(\mathbf{z}) = (\sigma_i(\mathbf{z} \cdot W_i + b_i)) \\ \widehat{\mathbf{y}} = f_k \circ f_{k-1} \circ \ldots \circ f_1(\mathbf{x}) \end{cases} \tag{3.1}$$

Notice here the versatility of the neural network architecture towards the input and output dimensions: the W_i weight matrices and biases b_i can have their dimensions tweaked according to the considered problem. Figure 3.1 illustrates this architecture.

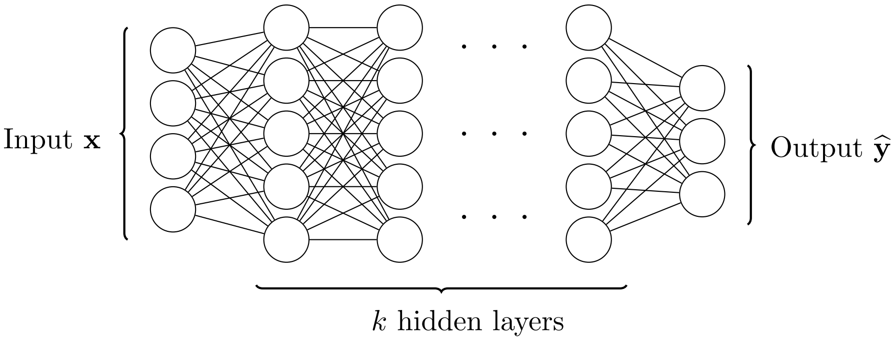


Figure 3.1: Illustration of a multilayer perceptron. Each node of a hidden layer represent an activation function, and each edge represent a weight W^i_{kl}.

Each neural network such as MLPs is endowed with hyper-parameters, i.e. parameters that are fixed at the conception of the algorithm and that are not optimized during the training phase of the algorithm. Examples of these hyper-parameters are the number of hidden units of the neural network or the number of layers of the neural network. Those hyper-parameters are often chosen by doing a hyper-parameter search, such as grid search (the space of hyper-parameters is tested incrementally in a given range) or a random search (where hyper-parameters are sampled randomly in the search space). In the automatic machine learning field of research, works have been done to learn those hyper-parameters during the training (Gordon et al., 2018; Golovin et al., 2017; Feurer et al., 2015).

Neural networks with only a single hidden layer, called **shallow networks**, already possess strong representational power. Given enough neurons in the hidden layer, they can approximate any continuous function over a compact:

Theorem 1 (Universal approximation theorem - Cybenko (1989)). *Let $\sigma : \mathbb{R} \to \mathbb{R}$ be a nonconstant, bounded, and continuous function.*

For any $\varepsilon > 0$ and any continuous function f defined over a compact subset of $\mathbb{R}^m$, there exist an integer n_h, real constants $w'_i, b_i \in \mathbb{R}$ and real vectors $w_i \in \mathbb{R}^m$, such that:

$$\begin{cases} F(x) = \sum_{i=1}^{n_h} w'_i \sigma(x \cdot w_i + b) \\ |F(x) - f(x)| < \varepsilon \end{cases} \tag{3.2}$$

for any x defined in the compact subset of $\mathbb{R}^m$.

Note that this theorem requires the activation function to be bounded, thus excluding popular activation functions such as ReLU, in favor of others such as hyperbolic tangent.

Activation functions Activation functions introduce non-linearities into the neural network architecture: all other operations are linear between inputs and the weights. They place themselves at the end of a layer, and are represented by the neurons (circles) in Figure 3.1. For a function to be considered as an activation function, it must be differentiable and defined over the domain of the inputs. Here are some examples of activation functions used throughout this thesis:

- **Hyperbolic Tangent (Tanh)** is a bounded, continuous and monotonic activation function, defined over $\mathbb{R}$:

$$\begin{cases} tanh(x) = \frac{2}{1+e^{-2x}} - 1 \\ tanh'(x) = 1 - tanh^2(x) \end{cases} \tag{3.3}$$

  One benefit of the Tanh function is its linearity near 0, making regularized models on weights near-linear. However, Tanh is on the computationally expensive activation function compared to alternatives.

- **Rectified Linear Unit (ReLU)** represents one of the most popular activation functions ;

$$\begin{cases} ReLU(x) = \begin{cases} x \text{ if } x > 0 \\ 0 \text{ otherwise} \end{cases} \\ ReLU'(x) = \begin{cases} 1 \text{ if } x > 0 \\ 0 \text{ otherwise} \end{cases} \end{cases} \tag{3.4}$$

ReLU isn't bounded and piecewise-linear and continuous; its gradient is 0 if the input is negative. However, the simplicity of the function and its derivative allows for quick computation, making it one of the most popular activation functions. A variant, LeakyReLU, introduces a non-constant linearity for the negative range of the input:

$$\begin{cases} LeakyReLU_a(x) = \begin{cases} x \text{ if } x > 0 \\ ax \text{ otherwise, } a \in \mathbb{R} \end{cases} \\ LeakyReLU'_a(x) = \begin{cases} 1 \text{ if } x > 0 \\ a \text{ otherwise} \end{cases} \end{cases} \tag{3.5}$$

with usually $a \leq 0.2$. LeakyReLU is a popular alternative for neural network architectures such as adversarial neural networks (Section 3.4).

- **Softmax** is a particular activation function as it isn't traditionally used in the middle of the neural network, but at the end of the neural network for classification tasks. Indeed, the softmax function allows to transition from real values of the neural network (logits) to probabilities; considering a K dimensional input x the softmax corresponds to:

$$softmax(x)_k = \frac{e^{x_k}}{\sum_{i=1}^{K} e^{x_i}} \tag{3.6}$$

This function corresponds to the sigmoid function in the case of binary classification with an output dimension of 1:

$$sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{3.7}$$

3.2 Learning with gradient descent

For a neural network to be able to learn, i.e. to be optimized, one must first define a differentiable criterion for success: a metric that denotes how the predictions provided by the neural network are close to the expected predictions (i.e. the labels). This metric is called loss, as the neural network is directly trained using backpropagation on this value, thus the need for differentiability. In this section, we will discuss the some standard losses of a neural network, the optimizers and finally the learning procedure of a neural network. Let $P(X, y)$ denote the joint distribution of the data X and the corresponding labels y and let $\widehat{P}(X, f(X))$ the joint distribution of the data and its output of the neural network f.

3.2.1 Losses

The Cross-Entropy loss is a standard loss for classification problems. It estimates the how $\widehat{P}$ is different from P by computing its cross-entropy:

$$\begin{aligned} H_P(\widehat{P}) &= H(P) + D_{KL}(P||\widehat{P}) \qquad (3.8) \\ &= -\sum_{j=1}^{m} P(\mathbf{y}_j) log(\widehat{P}(\mathbf{y}_j))) \qquad (3.9) \end{aligned}$$

where H denotes the entropy, D_{KL} denotes the Kullback-Leibler divergence and m denotes the number of classes. In practical applications, we do not have access to the distributions P and $\widehat{P}$, but to their respective sets of samples $\mathbf{y}$ and $\widehat{\mathbf{y}}$. We can therefore approximate the cross-entropy with the following loss:

$$\mathcal{L}_{CE} = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m} \delta_{ij} log(\widehat{p}_{i,j}) \qquad (3.10)$$

where n denotes the number of samples in the dataset and $\delta_{ij} = 1$ if $y_i = j$ and 0 otherwise. y_i represents the the label of the i^{th} sample, and $\widehat{p}_{i,j}$ is the probability associated with the j^{th} output of the neural network, obtained by applying the softmax function to the raw (logit) output $\widehat{\mathbf{y}}$.

The L_k losses are standard regression losses, they apply the L_k norm to the difference between the expected values and the predicted values by the NN:

$$\mathcal{L}_k = \frac{1}{n}\sqrt[k]{\sum_{i=1}^{n}(y_i - \widehat{y}_i)^k} \qquad (3.11)$$

The most commonly used are the $\mathcal{L}_1$ and $\mathcal{L}_2$ losses for their stability and robustness.

The Maximum Mean Discrepancy statistic (MMD) (Gretton et al., 2007) The MMD loss is a kernel-based loss that allows to compare predictions and expected values in distribution for continous data unlike the L_k losses that operate pointwise, thus making MMD fit for generative models (Li et al., 2015, 2017).

Let k be a real-valued symmetric kernel defined over P and $\widehat{P}$, and let $\mu_k = \int k(x,\cdot)\mathrm{d}P(x)$ be the *kernel mean embedding* of the distribution P, according to the kernel function $k(x,x') = \langle k(x,\cdot), k(x',\cdot)\rangle_{\mathcal{H}_k}$ with associated reproducing kernel Hilbert space $\mathcal{H}_k$. Therefore, μ_k summarizes P as the expected value of the features computed by k over samples drawn from P. Let $\mathbf{y}$ and $\widehat{\mathbf{y}}$ samples respectively from the distributions P and $\widehat{P}$ with $Card(\mathbf{y}) = n_1$ and $Card(\widehat{\mathbf{y}}) = n_2$

$$\mathrm{MMD}_k(P,\widehat{P}) = \left\|\mu_k(P) - \mu_k(\widehat{P})\right\|_{\mathcal{H}_k}.$$

Then in the finite sample case, we approximate the kernel mean embedding $\mu_k(P)$ by the *empirical kernel mean embedding* $\mu_k(\mathbf{y}) = \frac{1}{|\mathbf{y}|}\sum_{y_n \in \mathbf{y}} k(x, \cdot)$, and respectively for $\widehat{P}$. Then, the empirical MMD statistic is

$$\widehat{\text{MMD}}_k(\mathbf{y}, \widehat{\mathbf{y}}) = \frac{1}{n_1^2}\sum_{i,j}^{n} k(y_i, y_j) + \frac{1}{n_2^2}\sum_{i,j}^{n} k(\hat{y}_i, \hat{y}_j) - \frac{2}{n_1 n_2}\sum_{i,j}^{n} k(y_i, \hat{y}_j). \tag{3.12}$$

Importantly, the empirical MMD tends to zero as $n \to \infty$ if and only if $P = \widehat{P}$, as long as k is a characteristic kernel (Gretton et al., 2007). This property makes the MMD an excellent choice to model how close the observational distribution P is to the estimated observational distribution $\widehat{P}$. In terms of computation however, the evaluation of $\widehat{\text{MMD}}_k(\mathbf{y}, \widehat{\mathbf{y}})$ takes $O(n_1 n_2)$ time, which is prohibitive for very large n.

To alleviate this issue, Lopez-Paz (2016) proposed a linear time approximation of the MMD. When using a shift-invariant kernel[1], such as the Gaussian kernel, one can invoke Bochner's theorem (Edwards, 1964) to obtain a linear-time approximation of the Gaussian kernel:

$$\hat{\mu}_k^m(\mathbf{y}) = \sqrt{\frac{2}{m}}\frac{1}{n_1}\sum_{x \in \mathcal{D}} \left[\cos(\langle w_1, x\rangle + b_1), \ldots, \cos(\langle w_m, x\rangle + b_m)\right], \tag{3.13}$$

where w_i is drawn from the normalized Fourier transform of the Gaussian kernel, and $b_i \sim U[0, 2\pi]$, for $i = 1, \ldots, m$. It can be shown that the approximation converges pointwise as the number of random features $m \to \infty$.

This approximation allows to approximate the empirical MMD (Lopez-Paz et al., 2015), with complexity $O(mn)$:

$$\widehat{\text{MMD}}_k^m(\mathbf{y}, \widehat{\mathbf{y}}) = \|\hat{\mu}_k^m(\mathbf{y}) - \hat{\mu}_k^m(\widehat{\mathbf{y}})\|_{\mathbb{R}^m} \tag{3.14}$$

3.2.2 Gradient Descent

Gradient descent is an iterative optimization method to adapt the learnable weights of an algorithm to optimize a smooth learning criterion, i.e. the loss through backpropagation. One main constraint for this approach is the need of a differentiable computation graph with respect to the data. We will consider neural networks (NNs) in the following as NN are quite well-suited to gradient descent: the gradient computation through the whole graph becomes straightforward.

Assuming a fully differentiable computation graph, the output of the NN is evaluated using a loss criterion $\mathcal{L}$, which defines the objective function to optimize. Thus, the gradient backpropagation algorithm retraces the algorithm backwards (in the sense of algorithmic operations) while computing the respective gradients for each weight θ_i with respect to the

[1] kernel insensitive to translations, i.e. $k(x, x') = k(x + a, x' + a)$, with $a \in \mathbb{R}^d$

input. Considering the following k-layer neural network:

$$\begin{cases} f_i(\mathbf{z}) = (\sigma_i(\mathbf{z} \cdot W_i + b_i)) \\ \widehat{\mathbf{y}} = f_k \circ f_{k-1} \circ \ldots \circ f_1(\mathbf{x}) \\ \widehat{\mathbf{y}}_i = f_i \circ f_{i-1} \circ \ldots \circ f_1(\mathbf{x}), \text{ for } i \in [1, k] \end{cases} \tag{3.15}$$

The gradient of the weight matrix of the i^{th} layer is computed using the chain rule:

$$\begin{aligned} \nabla\mathcal{L}(\theta_i) &= \frac{\partial\mathcal{L}(\widehat{\mathbf{y}}, \mathbf{y})}{\partial\theta_i} \\ &= \frac{\partial\mathcal{L}(\widehat{\mathbf{y}}, \mathbf{y})}{\partial\widehat{\mathbf{y}}_{k-1}} \frac{\partial\widehat{\mathbf{y}}_{k-1}}{\partial\widehat{\mathbf{y}}_{k-2}} \cdots \frac{\partial\widehat{\mathbf{y}}_{i+1}}{\partial\theta_i} \end{aligned} \tag{3.16}$$

where θ_i accounts for the weights W_i and biases b_i. Next, the weights of the neural network are adjusted according to the computed gradient in a linear fashion:

$$\theta_i \leftarrow \theta_i - \eta\frac{1}{n}\sum_{j=1}^{n} \nabla\mathcal{L}_j(\theta_i) \tag{3.17}$$

with $\mathcal{L}_j$ representing the value of the loss $\mathcal{L}$ for the j^{th} example, and η denoting the learning rate of the weights.

Training a neural network The learning procedure of an artificial neural network procedure is iterative: at each epoch, the data (usually divided in mini-batches) is passed through the network, and the gradient is computed for each weight using backpropagation of the loss. Finally, the weights are adjusted using gradient descent. The training phase of a NN is resumed in Algorithm 3.

Algorithm 3: Neural network learning procedure (Stochastic Gradient Descent)

```
for number of epochs do
    for number of batches do
        • Sample a batch $\mathbf{x}_i$ out of available data
        • Compute the output $\widehat{\mathbf{y}}_i$ of the NN
        • Compute the loss w.r.t the true labels $\mathbf{y}_i$: $\mathcal{L}_i = \mathcal{L}(\widehat{\mathbf{y}}_i, \mathbf{y}_i)$
        • Compute the gradients for all the weights $\nabla\mathcal{L}_i(\theta_k)$
        • Update all the trainable weights with the gradient and the optimizer:
              $\theta_k \leftarrow \theta_k - \eta\frac{1}{n}\sum_{j=1}^{n} \nabla\mathcal{L}_j(\theta_k)$
    end
end
```

3.3 Hyper-parameter optimization and regularization

In this section, some specific constraints and tricks that improve the learning process of neural networks and structures that address precise needs in the computational graph will be presented.

3.3.1 Adjusting the learning rate

The learning rate represents one of the most important hyper-parameters of a neural network. Noted η in Algorithm 3, it controls how the gradient affects the weights: the higher the value, the more the weights value are shifted according to the gradient, thus the learning rate sets how quickly the model is adapted to the task defined by the loss. The impact of the learning rate on the neural network is quite significant, as setting at a value to high refrains the network from converging, and setting it too low refrains it from escaping local minima. In order to solve this issue, a solution is to apply a learning rate schedule: As soon as the loss does not decrease any more for a number of epochs at a given learning rate, the learning rate is decreased to allow the weights to be optimized at a finer scale. Learning rate scheduling provides a good trade-off between accuracy and computational efficiency.

Another alternative is to rely on momentum-based optimization: leveraging more moments allows the optimizer to have more context on the optimization and adapt its learning rate. One of the most popular momentum based optimizers are Adam:

Adam (Kingma and Ba, 2014) is a momentum based optimizer based on gradient descent. Instead of linearly updating the neural network's weights with the gradient's mean value, it takes into account of higher order statistics of the gradient, using decaying averages of gradients m and squared gradients v in the following fashion:

$$\begin{cases} m(\theta_i) \leftarrow \beta_1 m(\theta_i) + (1-\beta_1)\nabla\mathcal{L}(\theta_i) \\ v(\theta_i) \leftarrow \beta_2 v(\theta_i) + (1-\beta_2)\nabla\mathcal{L}(\theta_i)^2 \\ \hat{m}(\theta_i) \leftarrow m(\theta_i)/(1-\beta_1) \\ \hat{v}(\theta_i) \leftarrow v(\theta_i)/(1-\beta_2) \end{cases} \tag{3.18}$$

$$\theta_i \leftarrow \theta_i - \frac{\eta}{\sqrt{\hat{v}(\theta_i)} + \epsilon}\,\hat{m}(\theta_i) \tag{3.19}$$

with β_1 and β_2 being the decay rate hyper-parameters of Adam.

3.3.2 Initialization of the weights

The initial state of a neural network, i.e. the initial values of the parameters do have a strong impact on the training of the neural network, and ultimately, its final accuracy. The initialization should not be with zeros or constant values in all the network, as the gradient computation will be identical throughout the network, making all the weights identical. On the other hand setting the weights at too high values ($|W_{ij}| > 5$) might disturb the training, making it diverge or saturate. Therefore, one working heuristic is to set randomly the weights, following a uniform or Gaussian distribution: they allow the network to possess enough distinct values to train correctly. Other initialization heuristics have been developed, known to improve performance, such as Xavier or He initializations (Glorot and Bengio, 2010; He et al., 2015). In this thesis, we will be using a variant of those initializations, using a uniform distribution:

$$W_i \sim \mathcal{U}\left[-\sqrt{\frac{1}{\text{fan\_in}}}, \sqrt{\frac{1}{\text{fan\_in}}}\right] \tag{3.20}$$

where fan_in represents the number of incoming network connections.

3.3.3 Regularization

As many machine learning models are over-parameterized[2], they might content themselves with memorizing the training data instead of generalizing, i.e. being able to handle properly unseen data. This phenomenon, known as **overfitting**, has the effect of increasing the accuracy on training data but ultimately reducing the predictive power of the model on unseen data (ex. Figure 3.2).

In order to prevent overfitting, the regularization of the models have proved itself to be quite effective: the idea is to limit the capacity of the models by adding constraints to the loss. Traditional constraints for regularization are the L_1 and L_2 losses on the weights of the model. For the neural network described in Eq. 3.15, this gives:

$$L_1 : \sum_{i=1}^{k} \sum_{j,l}^{p,q} W_{i,jl} + b_{i,l} \tag{3.21}$$

$$L_2 : \sum_{i=1}^{k} \sum_{j,l}^{p,q} {W_{i,jl}}^2 + {b_{i,l}}^2 \tag{3.22}$$

Another approach for regularization, called **dropout**, consists in randomly cutting off temporarily some neural connections at each epoch during the training phase (some neurons have their value set to 0, deactivating the neuron in both forward and backward pass). Dropout forbids weights of the neural network to co-adapt to themselves, and is known to efficiently reduce overfitting[3].

In the setting of generative models for causality, this notion of regularization is of utmost importance. In fact, the goal to model as closely as possible the true causal mechanisms with our generative models; relying on Occam's razor principle. Having overfitting models to model the causal mechanisms might introduce biases in the model, leading ultimately to wrong conclusions and results.

[2] Having much more parameters than needed to learn, more than the dataset itself (Welling, 2018).

[3] Dropout can be assimilated to ensemble methods, where sub-networks are trained separately towards the same objective, to be aggregated in the end. (Hara et al., 2016; Dietterich, 2000)

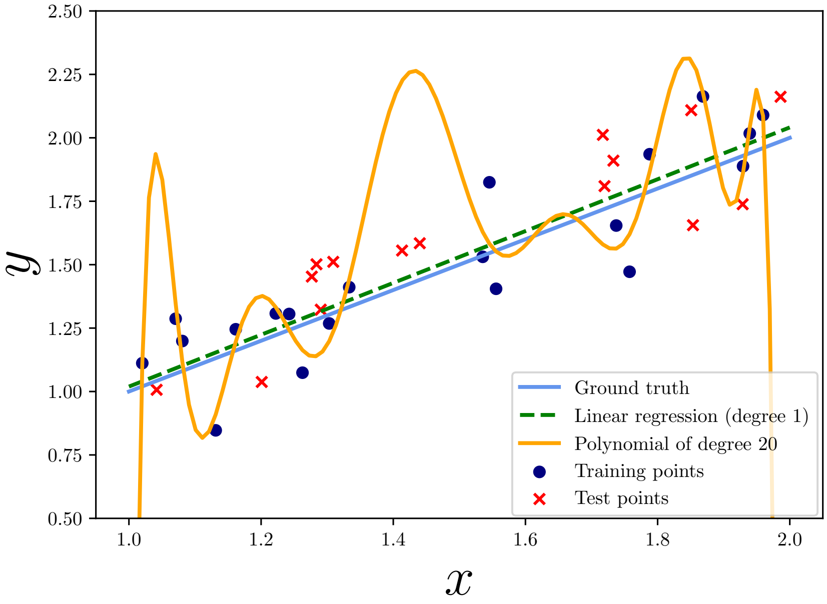


Figure 3.2: Illustration of overfitting with a underlying linear mechanism and an over-parameterized polynomial regression where the task is to predict y from x. The over-parameterized polynomial regression fits well the training data but has low accuracy on new data.

3.3.4 Batch Normalization

Batch normalization (Ioffe and Szegedy, 2015) is a neural network layer takes into account batch statistics to accelerate training of neural network. The batch-normalization layer normalizes the incoming data following minibatch statistics with two parameters γ and β to improve the learning process with information throughout the batch, and adjusting the variance of the gradients. Considering $\mathbf{x} = \{x_1, \dots, x_n\}$ a mini-batch of data given as input and $\mathbf{y}$ as the output, the batch-normalization layer computes the following:

$$\begin{cases} \bar{\mathbf{x}} & \leftarrow \frac{1}{n}\sum_{i=1}^{n} x_i \\ \widetilde{\mathbf{x}} & \leftarrow \frac{1}{n}\sum_{i=1}^{n} x_i^2 \\ \sigma_{\mathbf{x}}^2 & \leftarrow \widetilde{\mathbf{x}} - n\bar{\mathbf{x}}^2 \\ \hat{x}_i & \leftarrow \frac{x_i - \bar{\mathbf{x}}}{\sqrt{\sigma_{\mathbf{x}}^2 + \epsilon}} \\ y_i & \leftarrow \gamma \hat{x}_i + \beta \end{cases} \tag{3.23}$$

with γ and β being learnable parameters.

This approach is known to increase the stability of the training procedure of the neural network, as well as diminishing the number of epochs required at the cost of slower epochs, because of the computation of batch statistics.

3.3.5 Learning discrete values with the Gumbel softmax trick

Adjusting discrete hyperparameters such as the number of hidden units of a neural network or the dropout connections can prove itself to be quite tricky : if naively done, the neural nets must be trained anew for every considered categorical value as the *argmax* function is not differentiable, thus preventing the gradient from backpropagating. Alternatively, learning the parameters of Bernoulli distributions is infeasible as the sampling process does also break the differenciability of the computational graph.

The Binary Concrete relaxation approach (Maddison et al., 2016; Jang et al., 2016), has been proposed to alleviate this issue. It bases itself on an approximation of the categorical distribution with the Gumbel-Max trick (Gumbel, 1954), considering class probabilities[4] $p_1, \ldots, p_m$:

$$y_i = \frac{\exp{(\log p_i + g_i)}/\tau}{\sum_{j=1}^{m} \exp{(\log p_j + g_j)}/\tau}, \quad \text{for } i \in [1, m] \tag{3.24}$$

where the g_i values are samples drawn from the Gumbel(0, 1) distribution, and τ is a temperature parameter that controls the trade-off between the differenciability and the accuracy of the approximation. Another alternative is to separate the forward pass from the backward pass using the Gumbel trick: on the forward pass an argmax is considered (by setting $\tau = 0$), whereas a softmax is taken into account during the backward pass.

3.3.6 Recovering directed acyclic graphs

In the setting of graph learning, many problems require a directed acyclic graph (DAG) as an output. However, it is often too complex to test independently all DAGs ; thus the idea of imposing a constraint enforcing the acyclicity of the learned graph, named NOTEARS (Zheng et al., 2018b).

Letting A denote the binary adjacency matrix of the evaluated graph (1 if an edge is present, 0 otherwise), A represents a DAG if and only if

$$A^d = \mathbf{0}^d \tag{3.25}$$

This however appears to be impractical to use as a loss for numerical reasons. Zheng et al. (2018b) introduces a new smoother acyclicity constraint to augment the learning criterion with:

$$\mathcal{L}_{DAG} = \sum_{k=1}^{d} \frac{\text{tr } A^k}{k!} \tag{3.26}$$

Indeed, without any re-weighting of the number of cycles of size k by $k!$, the entries of A^k can easily exceed machine precision for even small values of d, which makes both function and

[4] These probabilities are obtainable from logit values by using a softmax function.

gradient evaluations highly unstable. This criterion is minimal (equal to zero) if and only if the considered graph is acyclic.

This differenciable constraint allows include a DAG regularization into the learning process of neural network directly on the adjacency matrix of the graph. It is extensively used in the SAM approach, described in Chapter 5.

3.4 Adversarial neural networks

Adversarial Neural Networks (Goodfellow et al., 2014) are a new paradigm for generative neural networks: The metric used to evaluate the generation quality is also a neural network called discriminator. This brings forward an concurrent optimization game between both the generator and the discriminator.

Generative Adversarial Neural Networks (GANs) allows for training generative neural networks of good quality (realistic samples and distributions) while being linear in computational complexity. GANs will be extensively used in SAM (Chapter 5) in order to train a neural network following the predicted causal graph and evaluate the data generated with the candidate model with respect to the real data distribution.

3.4.1 Description

As metrics for data generation are either not very relevant for high-dimensional data or computationally expensive (such as MMD (Gretton et al., 2007)), using a neural network as a metric proved itself to be a viable and efficient alternative, as shown by the DC-GAN (Radford et al., 2015).

The Generative Adversarial Network (GAN) setting relies on two main blocks: the generator G and the discriminator D. The architecture is depicted on Fig. 3.3. The discriminator has to distinguish true data from fake data coming from the generator along a binary classification scheme, and it is trained to do so during the learning process. On the opposite, the generator has to generate data from noise to fool the generator. In other words, its loss function corresponds to the opposite of the generator loss on fake data. Therefore, both the generator and the discriminator evolve concurrently during the learning process, which might lead to some instability. Considering the finite and real data $\mathbf{x} = \{x_i\}_{i=1}^n$ and noise variables $\mathbf{z} = \{z_i\}_{i=1}^n$, the discriminator D and the generator G respective losses are the following:

$$\mathcal{L}_D = \frac{1}{n}\sum_{i=1}^{n}[\log D(x_i) + \log(1 - D(G(z_i)))] \tag{3.27}$$

$$\mathcal{L}_G = \frac{1}{n}\sum_{i=1}^{n}\log\left(1 - D(G(z_i))\right) \tag{3.28}$$

While the GAN is quite efficient, it presents some issues:

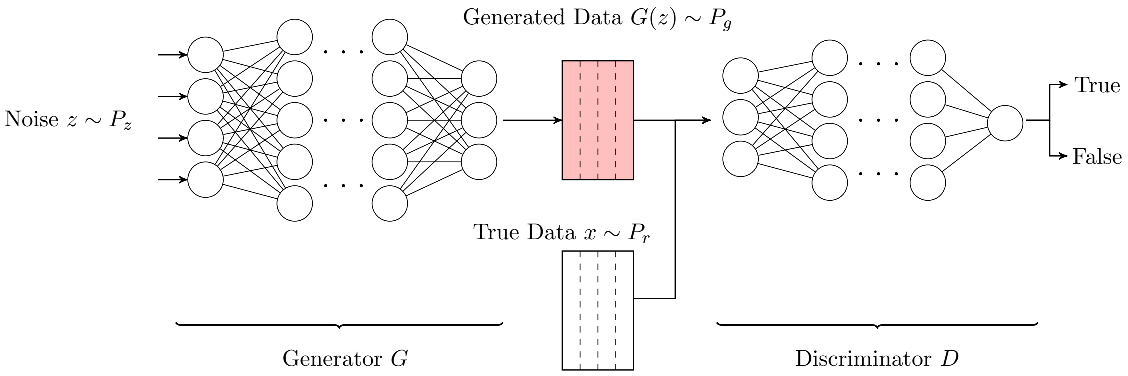


Figure 3.3: Architecture of a Generative Adversarial Network

- The GAN architecture never converges: as the generator gets better, the discriminator has better samples to train itself, and so on.

- The learning process can fail when the discriminator gets too successful as its gradient vanishes, preventing the generator from improving itself. This phenomenon is known as mode-collapse.

- Mode-dropping: The generator has no real incentive to reproduce all the distribution of the data; it can manage to generate only some limited variety of samples.

- The architecture is highly sensitive to hyper-parameter settings.

3.4.2 Approximations of f-divergences using GAN

The f-divergence framework, introduced by Nguyen et al. (2010) defines the family of f-divergence between two distributions $\mathcal{P}$ and $\mathcal{Q}$ as:

$$D_f(\mathcal{P}||\mathcal{Q}) = \int_{\mathcal{X}} q(x) f\left(\frac{p(x)}{q(x)}\right) dx \tag{3.29}$$

with $\mathcal{X}$ being the domain of x. They can be seen as a difference measure between distributions; examples of popular f-divergences are the Kullback-Leibler Divergence (D_{KL}), or the Jensen-Shannon Divergence (D_{JS}). In this section, the estimation of f-divergences with the GAN architecture will be explained.

Estimation of the Jensen-Shannon Divergence In the traditional GAN setting, considering the optimal discriminator G^* for any generator D, Goodfellow et al. (2014) shows that the discriminator estimates the Jensen-Shannon Divergence (D_{JS}) up to a constant between the real and generated data distributions P_r and P_g:

$$\begin{aligned} D_{JS} &= \frac{1}{2}D_{KL}\left(P_r \middle\| \frac{P_r + P_g}{2}\right) + \frac{1}{2}D_{KL}\left(P_g \middle\| \frac{P_r + P_g}{2}\right) \\ &= \frac{1}{2}\left(\log 2 + \mathbb{E}_{x\sim P_r}\left[P_r(x)\log\frac{P_r(x)}{P_g(x)+P_r(x)}\right]\right) \\ &+ \frac{1}{2}\left(\log 2 + \mathbb{E}_{x\sim P_g}\left[P_r(x)\log\frac{P_g(x)}{P_g(x)+P_r(x)}\right]\right) \\ &= \log 2 + \frac{1}{2}\left(\mathbb{E}_{x\sim P_r}\left[\log D^*(x)\right] + \mathbb{E}_{x\sim P_g}\left[\log D^*(x)\right]\right) \\ &= \log 2 + \frac{1}{2}\left(\mathbb{E}_{x\sim P_r}\left[\log D^*(x)\right] + \mathbb{E}_{z\sim P_z}\left[\log D^*(G(z))\right]\right) \\ &= \log 2 + \frac{1}{2}\left(\mathcal{L}_D\right) \end{aligned} \tag{3.30}$$

with P_z being the noise distribution.

***f*- GAN** Considering two distributions $\mathcal{P}$ and $\mathcal{Q}$ defined over $\mathbb{R}^d$ and letting T denote a set of functions defined over $\mathbb{R}^d$, Nguyen et al. (2010) establishes the following bound:

$$D_{KL}[\mathcal{P} \parallel \mathcal{Q}] \geq \sup_{T\in\mathcal{T}} \mathbb{E}_{x\sim\mathcal{P}}[T(x)] - \mathbb{E}_{x\sim\mathcal{Q}}[e^{T(x)-1}], \tag{3.31}$$

with a tight bound for sufficiently large families $\mathcal{T}$.

The idea proposed by Nowozin et al. (2016) named *f-gan* is then to choose $\mathcal{T}$ to be the family of functions $T_\omega : \mathbb{R}^d \to \mathbb{R}$ parameterized by a deep neural network with parameter $\omega \in \Omega$, and proposes new activation functions to extend the GAN framework to estimate other f-divergences. These activation functions are resumed in Table 3.1. These various output activations allows the GAN framework to estimate traditional divergences; the discriminator of the Structural Agnostic Model (Chapter 5) approaches the reverse Kullback-Leibler divergence in order to compare the quality of the generated data and the true data.

Table 3.1: Recommended final layer activation functions and critical variational function level defined by $f'(1)$. The critical value $f'(1)$ can be interpreted as a classification threshold applied to $T(x)$ to distinguish between true and generated samples. Table taken from Nowozin et al. (2016).

Name	Output activation g_f	dom_{f^*}	Conjugate $f^*(t)$	$f'(1)$
Kullback-Leibler (KL)	v	$\mathbb{R}$	$\exp(t-1)$	1
Reverse KL	$-\exp(-v)$	$\mathbb{R}_-$	$-1-\log(-t)$	-1
Pearson χ^2	v	$\mathbb{R}$	$\frac{1}{4}t^2+t$	0
Squared Hellinger	$1-\exp(-v)$	$t<1$	$\frac{t}{1-t}$	0
Jensen-Shannon	$\log(2)-\log(1+\exp(-v))$	$t<\log(2)$	$-\log(2-\exp(t))$	0
GAN	$-\log(1+\exp(-v))$	$\mathbb{R}_-$	$-\log(1-\exp(t))$	$-\log(2)$

Partial conclusion

The popularity of neural networks compared to traditional learning machines comes from their adaptability: they can be easily customized, as long as the operators used are differentiable (Section 3.1, 3.2). Therefore, many caveats have been circumvented with particular architectures or operators (Section 3.3), such as Batchnorm, Adam, and so on. Moreover, their architecture allows for portability for Graphical Processing Units (GPUs) and enables the efficient processing of significant amounts of data. A useful particularity of neural networks that we will leverage throughout this thesis is the simple control of the complexity of the neural networks: the number of hidden units and hidden layers is a direct proxy for the complexity of the functions.

Adversarial neural networks (Section 3.4) represent a new paradigm, in which the complex metric of success if replaced by a neural network; these two neural network train in tandem, in an adversarial game. This architecture brings not only better performance, but also computational efficiency (generally linear in the number of examples compared to other metrics such as MMD).

Chapter 4

Generative neural networks for score-based methods

This chapter describes the first contribution of the thesis, the *Causal Generative Neural Networks* (CGNN), published as a chapter in Goudet et al. (2018) (equal contribution). CGNN leverages the power of neural networks to learn a generative model of the joint distribution of the observed variables, by minimizing the Maximum Mean Discrepancy between generated and observed data. An approximate learning criterion is proposed to scale the computational cost of the approach to linear complexity in the number of observations. The performance of CGNN is studied throughout three experiments. First, CGNN is applied to cause-effect inference, where the task is to identify the best causal hypothesis out of "$X \rightarrow Y$" and "$Y \rightarrow X$". Secondly, CGNN is applied to the problem of identifying v-structures and conditional independences. Third, CGNN is applied to multivariate functional causal modeling: given a skeleton describing the direct dependencies in a set of random variables $\mathbf{X} = [X_1, \ldots, X_d]$, CGNN orients the edges in the skeleton to uncover the directed acyclic causal graph describing the causal structure of the random variables. On all three tasks, CGNN is extensively assessed on both artificial and real-world data, comparing favorably to the state-of-the-art. Finally, CGNN is extended to handle the case of confounders, where latent variables are involved in the overall causal model.

4.1 Modeling continuous functional causal models of a given structure with CGNN

This section first presents the modeling of continuous Functional Causal Models (FCMs)[1] with generative neural networks when **the causal structure is given *a priori***,

We first show that there exists a (non necessarily unique) *continuous* Functional Causal

[1] Sometimes also referred to as Structural Equation Models or SEMs

Model $(\mathcal{G}, f, \mathcal{E})$ such that the associated data generative process fits the distribution P of the observational data.

Theorem 2. Existence of continuous causal mechanisms *Let $\boldsymbol{X} = [X_1, \ldots, X_d]$ denote a set of continuous random variables with joint distribution P, and further assume that the joint density function h of P is continuous and strictly positive on a compact and convex subset of $\mathbb{R}^d$, and zero elsewhere. Letting $\mathcal{G}$ be a DAG such that P can be factorized along $\mathcal{G}$,*

$$P(X) = \prod_i P(X_i | X_{Pa(i;\mathcal{G})})$$

there exists $f = (f_1, \ldots, f_d)$ with f_i a continuous function with compact support in $\mathbb{R}^{|Pa(i;\mathcal{G})|} \times [0,1]$ such that $P(X)$ equals the generative model defined from FCM $(\mathcal{G}, f, \mathcal{E})$, with $\mathcal{E} = \mathcal{U}[0,1]$ the uniform distribution on $[0,1]$.

Proof. In Appendix 4.8.1 □

In order to model such continuous FCM $(\mathcal{G}, f, \mathcal{E})$ on d random variables $\mathbf{X} = [X_1, \ldots, X_d]$, we introduce the CGNN (Causal Generative Neural Network) depicted on Figure 4.1.

Definition 1. CGNN definition. A CGNN over d variables $[\widehat{X}_1, \ldots, \widehat{X}_d]$ is a triplet $\mathcal{C}_{\widehat{\mathcal{G}}, \widehat{f}} = (\widehat{\mathcal{G}}, \widehat{f}, \mathcal{E})$ where:

1. $\widehat{\mathcal{G}}$ is a Directed Acyclic Graph (DAG), the nodes of which are the variables $X_1, \ldots, X_d$, and such that there exists an edge $X_i \to X_j$ iff $X_i \in Pa(X_j)$, the set of parents of X_j.

2. For $i \in [\![1, d]\!]$, causal mechanism $\widehat{f}_i$ is a 1-hidden layer regression neural network with n_h hidden neurons:

$$\widehat{X}_i = \widehat{f}_i(\widehat{X}_{\mathrm{Pa}(i;\widehat{\mathcal{G}})}, E_i) = \sum_{k=1}^{n_h} \bar{w}_k^i \sigma \left(\sum_{j \in \mathrm{Pa}(i;\mathcal{G})} \widehat{w}_{jk}^i \widehat{X}_j + w_k^i E_i + b_k^i \right) + \bar{b}^i \qquad (4.1)$$

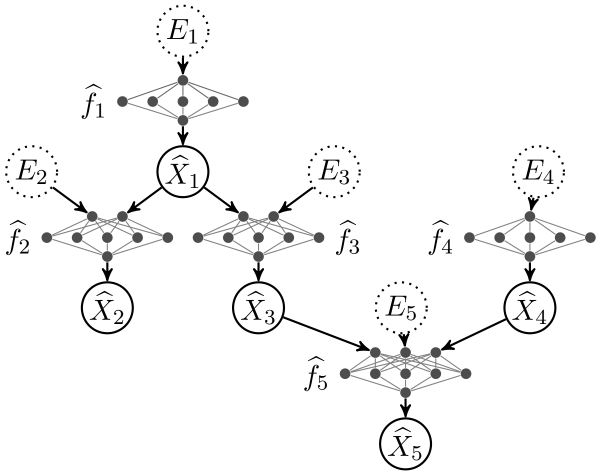
   with $n_h \in \mathbb{N}^*$ the number of hidden units, $\bar{w}_k^i, \widehat{w}_{jk}^i, w_k^i, b_k^i, \bar{b}^i \in \mathbb{R}$ the weights of the neural network, and σ a continuous activation function .

3. Each noise variable E_i is independent of the *cause* X_i. Furthermore, all noise variables are mutually independent and drawn after same distribution $\mathcal{E}$.

It is clear from its definition that a CGNN defines a continuous FCM.

4.1.1 Generative model and interventions

A CGNN $\mathcal{C}_{\widehat{\mathcal{G}}, \widehat{f}} = (\widehat{\mathcal{G}}, \widehat{f}, \mathcal{E})$ is a **generative** model in the sense that any sample $[e_1, \ldots, e_d]$ of the "noise" random vector $\mathbf{E} = [E_1, \ldots, E_d]$ can be used as "input" to the network to generate a data sample $[\widehat{x}_1, \ldots, \widehat{x}_d]$ of the estimated distribution $\widehat{P}(\widehat{\mathbf{X}}, \widehat{\mathbf{X}} \in \mathbb{R}^d$ as follows:



$$\begin{cases} \widehat{X}_1 = \widehat{f}_1(E_1) \\ \widehat{X}_2 = \widehat{f}_2(\widehat{X}_1, E_2) \\ \widehat{X}_3 = \widehat{f}_3(\widehat{X}_1, E_3) \\ \widehat{X}_4 = \widehat{f}_4(E_4) \\ \widehat{X}_5 = \widehat{f}_5(\widehat{X}_3, \widehat{X}_4, E_5) \end{cases}$$

Figure 4.1: Left: Causal Generative Neural Network over variables $\widehat{\mathbf{X}} = (\widehat{X}_1, \ldots, \widehat{X}_5)$. Right: Corresponding Functional Causal Model equations.

1. Draw n samples $\{[e_1, \ldots, e_d]\}$ independent identically distributed from the joint distribution of independent noise variables $\mathbf{E} = [E_1, \ldots, E_d]$.

2. Generate n samples $\{[\widehat{x}_1, \ldots, \widehat{x}_d]\}$, where each estimate sample $\widehat{x}_i$ of variable $\widehat{X}_i$ is computed as $\widehat{X_i} = \widehat{f}_i(\widehat{X_l}, E_i), l \in Pa(X_i))$

Notice that a CGNN generates a probability distribution $\widehat{P}$ which is Markov with respect to $\widehat{\mathcal{G}}$, as the graph $\widehat{\mathcal{G}}$ is acyclic and the noise variables E_i are mutually independent.

Importantly, CGNN supports interventions (c.f. Chapter 2), that is, freezing a variable X_i to some constant v_i. The resulting joint distribution noted $\widehat{P}_{\text{do}(\widehat{X}_i = v_i)}(\widehat{X})$, called *interventional distribution* (Pearl, 2009), can be computed from CGNN by discarding all causal influences on $\widehat{X}_i$ and clamping its value to v_i. It is emphasized that intervening is different from conditioning (*correlation does not imply causation*). The knowledge of interventional distributions is essential for e.g., public policy makers, wanting to estimate the overall effects of a decision on a given variable.

4.2 Model evaluation

The goal is to associate to each candidate solution $\mathcal{C}_{\widehat{\mathcal{G}},\widehat{f}} = (\widehat{\mathcal{G}}, \widehat{f}, \mathcal{E})$ a score reflecting how well this candidate solution describes the observational data. First we define the model scoring function (Section 4.2.1), then we show that this model scoring function allows us to build a CGNN generating a distribution $\widehat{P}(\widehat{X})$ that approximates $P(X)$ with arbitrary accuracy (Section 4.2.2).

4.2.1 Scoring metric

The score to be minimized ideally is the distance between the joint distribution P associated with the ground truth FCM, and the joint distribution $\widehat{P}$ defined by the CGNN candidate $\mathcal{C}_{\widehat{\mathcal{G}},\widehat{f}} = (\widehat{\mathcal{G}}, \widehat{f}, \mathcal{E})$. A tractable approximation thereof is given by the Maximum Mean Discrepancy (MMD) (Gretton et al., 2007) between the n-sample observational data $\mathcal{D}$, and an n'-sample $\widehat{\mathcal{D}}$ sampled after $\widehat{P}$[2]. Overall, the CGNN $\mathcal{C}_{\widehat{\mathcal{G}},\widehat{f}}$ is trained by minimizing

$$S(\mathcal{C}_{\widehat{\mathcal{G}},\widehat{f}}, \mathcal{D}) = \widehat{\mathrm{MMD}}_k(\mathcal{D}, \widehat{\mathcal{D}}) + \lambda|\widehat{\mathcal{G}}|, \tag{4.2}$$

with $\widehat{\mathrm{MMD}}_k(\mathcal{D}, \widehat{\mathcal{D}})$ defined as:

$$\widehat{\mathrm{MMD}}_k(\mathcal{D}, \widehat{\mathcal{D}}) = \frac{1}{n^2}\sum_{i,j=1}^{n} k(x_i, x_j) + \frac{1}{n^2}\sum_{i,j=1}^{n} k(\widehat{x}_i, \widehat{x}_j) - \frac{2}{n^2}\sum_{i,j=1}^{n} k(x_i, \widehat{x}_j) \tag{4.3}$$

where kernel k usually is taken as the Gaussian kernel ($k(x, x') = \exp(-\gamma\|x - x'\|_2^2)$). The MMD statistic, with quadratic complexity in the sample size, has the good property that as n goes to infinity, it goes to zero iff $P = \widehat{P}$ (Gretton et al., 2007). For scalability, a linear approximation of the MMD statistics based on m random features (Lopez-Paz, 2016), called $\widehat{\mathrm{MMD}}_k^m$ will also be used in the experiments with $m = 100$ (c.f. Section 3.2.1).

Due to the Gaussian kernel being differentiable, $\widehat{\mathrm{MMD}}_k$ and $\widehat{\mathrm{MMD}}_k^m$ are differentiable, and backpropagation can be used to learn the CGNN made of networks $\hat{f}_i$ structured along $\widehat{\mathcal{G}}$.

In order to compare candidate solutions with different structures in a fair manner, the evaluation score of Equation 4.2 is augmented with a penalization term $\lambda|\widehat{\mathcal{G}}|$, with $|\widehat{\mathcal{G}}|$ the number of edges in $\widehat{\mathcal{G}}$. Penalization weight λ is a hyper-parameter of the approach.

4.2.2 Representational power of CGNN

Let $\mathcal{D} = \{[x_{1,j}, \ldots, x_{d,j}]\}_{j=1}^{n}$ denote the data samples independently and identically distributed after the (unknown) joint distribution $P(\mathbf{X} = [X_1, \ldots, X_d])$, also referred to as observational data.

Under same conditions as in Theorem 2, ($P(X)$ being decomposable along graph $\mathcal{G}$, with continuous and strictly positive joint density function on a compact in $\mathbb{R}^d$ and zero elsewhere), there exists a CGNN $(\widehat{\mathcal{G}}, \widehat{f}, \mathcal{E})$, that approximates $P(X)$ with arbitrary accuracy:

Theorem 3. CGNN consistency lemma. *For $m \in [[1, d]]$, let Z_m denote the set of variables with topological order less than m and let d_m be its size. For any d_m-dimensional vector of noise values $e^{(m)}$, let $z_m(e^{(m)})$ (resp. $\widehat{z_m}(e^{(m)})$) be the vector of values computed in topological order from the FCM $(\mathcal{G}, f, \mathcal{E})$ (resp. the CGNN $(\mathcal{G}, \widehat{f}, \mathcal{E})$). For any $\epsilon > 0$, there exists a set of networks $\widehat{f}$ with architecture $\mathcal{G}$ such that*

$$\forall e^{(m)}, \|z_m(e^{(m)}) - \widehat{z_m}(e^{(m)})\| < \epsilon \tag{4.4}$$

[2] In this chapter, we will consider $n = n'$, for computational efficiency.

Proof. In Appendix 4.8.2 □

Using this Theorem and the $\widehat{\mathrm{MMD}}_k$ scoring criterion presented in Equation 4.3, it is shown that the distribution $\hat{P}$ of the CGNN can estimate the true observational distribution of the (unknown) FCM up to an arbitrary precision, under the assumption of an infinite observational sample:

Theorem 4. CGNN asymptotic consistency theorem. *Let $\mathcal{D}$ be an infinite observational sample generated from $(\mathcal{G}, f, \mathcal{E})$. With the same notations as in Prop. 2, for every sequence ϵ_t, such that $\epsilon_t > 0$ and goes to zero when $t \to \infty$, there exists a set $\widehat{f_t} = (\hat{f}_1^t \dots \hat{f}_d^t)$ such that $\widehat{MMD}_k$ between $\mathcal{D}$ and an infinite size sample $\widehat{\mathcal{D}}_t$ generated from the CGNN $(\mathcal{G}, \widehat{f}_t, \mathcal{E})$ is less than ϵ_t.*

Proof. In Appendix 4.8.3 □

Under these assumptions, as $\widehat{\mathrm{MMD}}_k(\mathcal{D}, \widehat{\mathcal{D}_t}) \to 0$, as $t \to \infty$, it implies that the sequence of generated $\widehat{P}_t$ converges in distribution toward the distribution P of the observed sample (Gretton et al., 2007). This result highlights the generality of this approach as we can model any kind of continuous FCM from observational data (assuming access to infinite observational data). Our class of model is not restricted to simplistic assumptions on the data generative process such as the additivity of the noise or linear causal mechanisms. But this strength comes with a new challenge relative to identifiability of such CGNNs, since the result of Theorem 4 holds for any DAG $\widehat{\mathcal{G}}$ such that P can be factorized along $\mathcal{G}$ and then for any any DAG in the Markov equivalence class of $\mathcal{G}$ (under classical assumption of CMA, CFA and CSA).

In particular in the pairwise setting, when only 2 variables X and Y are observed, the joint distribution $P(X, Y)$ can be factorized in two Markov equivalent DAGs $X \to Y$ or $Y \to X$ as $P(X, Y) = P(X)P(Y|X)$ and $P(X, Y) = P(Y)P(X|Y)$. Then the CGNN can reproduce equally well the observational distribution in both directions (under the assumption of Theorem 2). We refer the reader to Zhang and Hyvärinen (2009) for more details on this problem of identifiability in the bivariate case.

As shown in Section 4.3.3, the proposed approach enforces the discovery of causal models in the Markov equivalence class. Within this class, the non-identifiability issue is empirically mitigated by restricting the class of CGNNs considered, and specifically limiting the number n_h of hidden neurons in each causal mechanism (Eq. 4.1). Formally, we restrict ourselves to the sub-class of CGNNs, noted $\mathcal{C}_{\widehat{\mathcal{G}}, \widehat{f}^{n_h}} = (\widehat{\mathcal{G}}, \widehat{f}^{n_h}, \mathcal{E})$ with exactly n_h hidden neurons in each $\widehat{f}_i$ mechanism. Accordingly, any candidate $\widehat{\mathcal{G}}$ with number of edges $|\widehat{\mathcal{G}}|$ involves the same number of parameters: $(2d + |\widehat{\mathcal{G}}|) \times n_h$ weights and $d \times (n_h + 1)$ bias parameters. As shown experimentally in Section 4.5, this parameter n_h is crucial as it governs the CGNN ability to model the causal mechanisms: too small n_h, and data patterns may be missed; too large n_h, and overly complicated causal mechanisms may be retained.

4.3 Model optimization

Model optimization consists in finding a (nearly) optimum solution $(\widehat{\mathcal{G}}, \widehat{f})$ in the sense of the score defined in the previous section. The so-called *parametric* optimization of the CGNN, where structure estimate $\widehat{\mathcal{G}}$ is fixed and the goal is to find the best neural estimates $\widehat{f}$ conditionally to $\widehat{\mathcal{G}}$ is tackled in Section 4.3.1. The *non-parametric* optimization, aimed at finding the best structure estimate, is considered in Section 4.3.2. In Section 4.3.3, we present an identifiability result for CGNN up to Markov equivalence classes.

4.3.1 Parametric (weight) optimization

Given the acyclic structure estimate $\widehat{\mathcal{G}}$, the neural networks $\widehat{f}_1, \ldots, \widehat{f}_d$ of the CGNN are learned end-to-end using backpropagation with Adam optimizer (Kingma and Ba, 2014) by minimizing losses $\widehat{\mathrm{MMD}}_k$ (Eq. 4.3, referred to as **CGNN** ($\widehat{\mathrm{MMD}}_k$)) or $\widehat{\mathrm{MMD}}_k^m$ (see Section 3.2.1), **CGNN** ($\widehat{\mathrm{MMD}}_k^m$).

The procedure closely follows that of supervised continuous learning (regression), except for the fact that the loss to be minimized is the MMD loss instead of the mean squared error. A MMD loss is preferred to a generative adversarial network (GAN) setting because of the stability required in the training: the final value of the loss is used as the score of the graph candidate; the fluctuating training and value of the loss of a GAN makes it unfit for this kind of score-based approach for causality.

Neural nets $\widehat{f}_i$, $i \in [[1, d]]$ are trained during n_{train} epochs, where the noise samples, independent and identically distributed, are drawn in each epoch. In the $\widehat{\mathrm{MMD}}_k^m$ variant, the parameters of the random kernel are resampled from their respective distributions in each training epoch. After training, the score is computed and averaged over n_{eval} estimated samples of size n. Likewise, the noise samples are re-sampled anew for each evaluation sample. The overall process with training and evaluation is repeated nb_{run} times to reduce stochastic effects relative to random initialization of neural network weights and stochastic gradient descent. The procedure is detailed in Algorithm 4.

4.3.2 Non-parametric (structure) optimization

The number of directed acyclic graphs $\widehat{\mathcal{G}}$ over d nodes is super-exponential in d, making the non-parametric optimization of the CGNN structure an intractable computational and statistical problem. Taking inspiration from Tsamardinos et al. (2006); Nandy et al. (2015), we start from a graph skeleton recovered by other methods such as feature selection (Yamada et al., 2014). We focus on optimizing the edge orientations. Letting L denote the number of edges in the graph, it defines a combinatorial optimization problem of complexity $\mathcal{O}(2^L)$ (note however that not all orientations are admissible since the eventual oriented graph must be a DAG).

The motivation for this approach is to decouple the edge selection task and the causal modeling (edge orientation) tasks, and enable their independent assessment.

Algorithm 4: Evaluate candidate graph using generative neural networks

```
Data: Observational data $\mathbf{X} = X_1, \dots, X_d$ sampled i.i.d. from $P(\mathbf{X})$, candidate acyclic
      graph $\widehat{\mathcal{G}}$
Input: Number of hidden units $n_h$; learning rate $l_r$
$\mathcal{L}_t$ =0; for number of runs $nb_{runs}$ do
    forall $X_i \in \mathbf{X}$                          // Build a hierarchical neural network
     do
        Create a 1-hidden layer neural network $\widehat{f_i}$ with $n_h$ hidden units, 1 output neuron and
          $Card(\text{Pa}(X_i; \widehat{\mathcal{G}})) + 1$ inputs
    end
    for number of train epochs $n_{train}$                                     // Training phase
     do
        for $X_i$ in the topological order of $\widehat{\mathcal{G}}$ do
            Compute $\widehat{X_i} = \widehat{f_i}(\widehat{\text{Pa}}(X_i, \widehat{\mathcal{G}}), E_i)$, $E_i \sim \mathcal{N}(0,1)$
        end
        $\widehat{\mathbf{X}} = \{\widehat{X}_1, \dots \widehat{X}_d\}$
        Compute loss $\mathcal{L} = MMD(\mathbf{X}, \widehat{\mathbf{X}})$
        Backpropagate the loss $\mathcal{L}$ in the neural network
        Adjust the weights with the Adam Optimizer
    end
    for number of test epochs $n_{eval}$                                           // Test phase
     do
        for $X_i$ in the topological order of $\widehat{\mathcal{G}}$ do
            Compute $\widehat{X_i} = \widehat{f_i}(\widehat{\text{Pa}}(X_i, \widehat{\mathcal{G}}), E_i)$, $E_i \sim \mathcal{N}(0,1)$
        end
        $\widehat{\mathbf{X}} = \{\widehat{X}_1, \dots \widehat{X}_d\}$
        $\mathcal{L}_t \leftarrow \mathcal{L}_t + MMD(\mathbf{X}, \widehat{\mathbf{X}})$
    end
end
return $\mathcal{L}_t / (n_{eval} * nb_{run})$
```

Any $X_i - X_j$ edge in the graph skeleton stands for a direct dependency between variables X_i and X_j. Given Causal Markov and Faithfulness assumptions, such a direct dependency either reflects a direct causal relationship between the two variables ($X_i \rightarrow X_j$ or $X_i \leftarrow X_j$), or is due to the fact that X_i and X_j admit a latent (unknown) common cause ($X_i \leftrightarrow X_j$). Under the assumption of *causal sufficiency*, the latter does not hold. Therefore the $X_i - X_j$ link is associated with a causal relationship in one or the other direction. The causal sufficiency assumption will be relaxed in Section 4.7.

The edge orientation phase proceeds as follows:

- Each $X_i - X_j$ edge is first considered in isolation, and its orientation is evaluated using CGNN. Both score $S(\mathcal{C}_{X_i \rightarrow X_j, \widehat{f}}, \mathcal{D}_{ij})$ and $S(\mathcal{C}_{X_j \rightarrow X_i, \widehat{f}}, \mathcal{D}_{ij})$ are computed, where $\mathcal{D}_{ij} = \{[x_{i,l}, x_{j,l}]\}_{l=1}^{n}$. The best orientation corresponding to a minimum score is retained.

After this step, an initial graph is built with complexity $2L$ with L the number of edges in the skeleton graph.

- The initial graph is revised to remove all cycles. Starting from a set of random nodes[3], all paths are followed iteratively until all nodes are reached; an edge pointing toward an already visited node and forming a cycle is reversed. The resulting DAG is used as initial DAG for the structured optimization, below.

- The optimization of the DAG structure is achieved using a hill-climbing algorithm aimed to optimize the global score $S(\mathcal{C}_{\widehat{\mathcal{G}},\widehat{f}}, \mathcal{D})$. Iteratively, i) an edge $X_i - X_j$ is uniformly randomly selected in the current graph; ii) the graph obtained by reversing this edge is considered (if it is still a DAG and has not been considered before) and the associated global CGNN is retrained; iii) if this graph obtains a lower global score than the former one, it becomes the current graph and the process is iterated until reaching a (local) optimum. More sophisticated combinatorial optimization approaches, e.g. Tabu search, will be considered in further work. In this paper, hill-climbing is used for a proof of concept of the proposed approach, achieving a decent trade-off between computational time and accuracy.

At the end of the process each causal edge $X_i \to X_j$ in $\mathcal{G}$ is associated with a score, measuring its contribution to the global score:

$$S_{X_i \to X_j} = S(\mathcal{C}_{\widehat{\mathcal{G}} - \{X_i \to X_j\}, \widehat{f}}, \mathcal{D}) - S(\mathcal{C}_{\widehat{\mathcal{G}},\widehat{f}}, \mathcal{D}) \tag{4.5}$$

During the structure (non-parametric) optimization, the graph skeleton is fixed; no edge is added or removed. The penalization term $\lambda|\widehat{\mathcal{G}}|$ entering in the score evaluation (eq. 4.2) can thus be neglected at this stage and only the MMD-losses are used to compare two graphs. The penalization term will be used in Section 4.7 to compare structures with different skeletons, as the potential confounding factors will be dealt with by removing edges.

4.3.3 Identifiability of CGNN up to Markov equivalence classes

In the large sample limit of observational data, and assuming further that the generative distribution belongs to the CGNN class $\mathcal{C}_{\mathcal{G},f}$, then there exists a DAG reaching an MMD score of 0 in the Markov equivalence class of $\mathcal{G}$:

Theorem 5. CGNN Markov-equivalent structure identifiability. *Let $\boldsymbol{X} = [X_1, \ldots, X_d]$ denote a set of continuous random variables with joint distribution P, generated by a CGNN $\mathcal{C}_{\mathcal{G},f} = (\mathcal{G}, f, \mathcal{E})$ with $\mathcal{G}$ a directed acyclic graph. Let $\mathcal{D}$ be an infinite observational sample generated from this CGNN. We assume that P is Markov and faithful to the graph $\mathcal{G}$, and that every pair of variables (X_i, X_j) that are d-connected in the graph are not independent.*

[3] This initialization introduces stochasticity in the algorithm, but all CGNN instances should converge to the same solution if given enough exploration.

Algorithm 5: Causal Generative Network Algorithm

```
Data: Observational data $\mathbf{X} = X_1, \ldots, X_d$ sampled i.i.d. from $P(\mathbf{X})$, graph skeleton $\bar{\mathcal{G}}$
Input: Number of hidden units $n_h$; learning rate $l_r$
Initialize empty graph $\mathcal{G}$
forall Edge $(X_i, X_j)$ in $\bar{\mathcal{G}}$                                   // Pairwise orientation
 do
    // Evaluate Pairwise structures using Algorithm 4
    if Evaluate$(X_i \to X_j) <$ Evaluate$(X_i \leftarrow X_j)$ then
        Add edge $X_i \to X_j$ to $\mathcal{G}$
    else
        Add edge $X_i \leftarrow X_j$ to $\mathcal{G}$
    end
end
$S =$ Evaluate$(\mathcal{G})$ // Using Algorithm 4

impr $\leftarrow$ True // Search while structure improves

while impr==True                                                      // Structure search
 do
    impr $\leftarrow$ False
    forall Edge $X_i \to X_j$ in $\mathcal{G}$ do
        if Reversing $X_i \to X_j$ in $\mathcal{G}$ does not create any cycle then
            Let $\mathcal{G}'$ equal to $\mathcal{G}$ except for $X_i \leftarrow X_j$
            $S' =$ Evaluate$(\mathcal{G}')$ // Algorithm 4

            if $S > S'$ then
                $S \leftarrow S'$
                $\mathcal{G} \leftarrow \mathcal{G}'$
                impr $\leftarrow$ True
            end
        end
    end
end
return $\mathcal{G}$
```

We note $\widehat{\mathcal{D}}$ an infinite sample generated by a candidate CGNN, $\mathcal{C}_{\widehat{\mathcal{G}},\widehat{f}} = (\widehat{\mathcal{G}}, \widehat{f}, \mathcal{E})$. Then,
(i) If $\widehat{\mathcal{G}} = \mathcal{G}$ and $\widehat{f} = f$, then $\widehat{MMD}_k(\mathcal{D}, \widehat{\mathcal{D}}) = 0$.
(ii) For any graph $\widehat{\mathcal{G}}$ characterized by the same adjacencies but not belonging to the Markov equivalence class of $\mathcal{G}$, for all $\widehat{f}$, $\widehat{MMD}_k(\mathcal{D}, \widehat{\mathcal{D}}) \neq 0$.

Proof. In Appendix 4.8.4 □

This result does not establish the CGNN identifiability *within* the Markov class of equivalence. Although, based on the results of the experimental Section 4.4, we believe that stronger identifiability results could be derived, at this stage, we only have formal proof of identifiability up to a Markov equivalence class. Stronger results imply making assumptions about data generating model class and capacity of hypothesis class.[4]

4.4 Experimental setting

This section discusses the experimental setting in order to compare CGNN with other state of the art algorithms. Thereafter, the results obtained in the bivariate case, where only asymmetries in the joint distribution can be used to infer the causal relationship, are discussed. The variable triplet case, where conditional independence can be used to uncover causal orientations, and the general case of $d > 2$ variables are then considered. All computational times are measured on Intel Xeon 2.7Ghz (CPU) or on Nvidia GTX 1080Ti graphics card (GPU).

The CGNN architecture is a 1-hidden layer network with ReLU activation function. The multi-scale Gaussian kernel used in the MMD scores has bandwidth γ ranging in {0.005, 0.05, 0.25,0.5, 1,5,50}. The number nb_{run} used to average the score is set to 32 for CGNN-MMD (respectively 64 for CGNN-Fourier). In this section the distribution $\mathcal{E}$ of the noise variables is set to $\mathcal{N}(0, 1)$. Table 4.1 summarizes all the hyperparameters chosen for CGNN.

Table 4.1: Values of the CGNN hyper-parameters

Hyper-parameter	Symbol	Value
MMD bandwidth	γ	{0.005, 0.05, 0.25,0.5, 1,5,50}
Number of bootstraps	nb_{run}	32
Distribution of noise	$\mathcal{E}$	$\mathcal{N}(0, 1)$
Initial learning rate	l_r	0.01
Number of hidden units	n_h	20

Calibration. The number n_h of neurons in the hidden layer, controlling the identifiability of the model, is the most sensitive hyper-parameter of the presented approach. Calibration experiments are conducted to adjust its value, as follows, using a calibration task. A 1,500 sample dataset is generated from the linear structural equation model with additive uniform noise $Y = X + \mathcal{U}(0, 0.5), X \sim U([-2, 2])$ (Fig. 4.2). Both CGNNs associated to $X \to Y$ and $Y \to X$ are trained until reaching convergence ($n_{epoch} = 1,000$) using Adam (Kingma and Ba, 2014) with an initial learning rate of 0.01 and evaluated over $n_{eval} = 500$ generated samples. The distributions generated from both generative models are displayed on Fig. 4.2 for $n_h = 2, 5, 20, 100$. The associated scores (averaged on 32 runs) are displayed on Fig. 4.3a, confirming that the model space must be restricted for the sake of identifiability (cf. Section

[4] In some specific cases, such as in the bivariate linear FCM with Gaussian noise and Gaussian input, even by restricting the class of functions considered, the DAG cannot be identified from purely observational data (Mooij et al., 2016), and additional assumptions will be needed.

4.3.3 above). By conducting additional experiments to other cause-effect pairs, the value of n_h is set to 20 to attain optimal results.

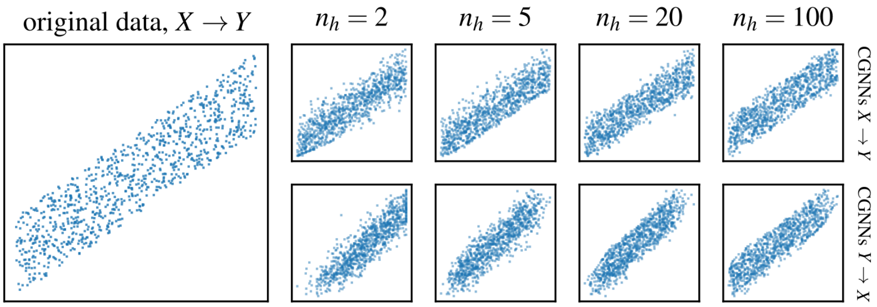
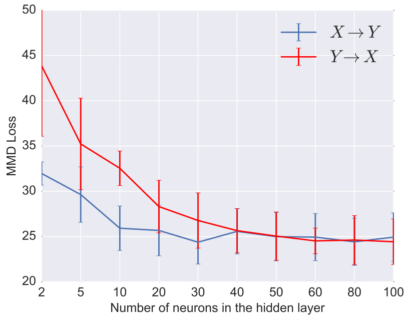


Figure 4.2: **Calibration data.** Leftmost: Data samples. Columns 2 to 5: Estimate samples generated from CGNN with direction $X \to Y$ (top row) and $Y \to X$ (bottom row) for number of hidden neurons $n_h = 2, 5, 20, 100$.



(a) $C_{X\to Y}$, $C_{Y\to X}$ with various n_h values.

(b) Scores $C_{X\to Y}$ and $C_{Y\to X}$ with their difference. *** denotes the significance at the 0.001 threshold with the t-test.

n_h	$C_{X\to Y}$	$C_{Y\to X}$	Diff.
2	32.0	43.9	11.9***
5	29.6	35.2	5.6***
10	25.9	32.5	6.6***
20	25.7	28.3	2.6***
30	24.4	26.8	2.4***
40	25.6	25.6	0.7
50	25.0	25.0	0.6
100	24.9	24.4	−0.5

Figure 4.3: CGNN sensitivity w.r.t. the number of hidden neurons n_h: Scores associated to both causal models (average and standard deviation over 32 runs).

Metrics of success All predictions of algorithms will be compared to the ground truth using three metrics:

- **Area Under the Precision Recall curve (AUPR)**: This metric account for the trade-off between the precision, the recall and the confidence score of the algorithms. All algorithms are bootstrapped and the confidence score of an edge corresponds to the ratio between the number of runs in which the edge is present in the prediction and the total number of runs.

- **Structural Hamming Distance (SHD)** corresponds to the traditional metric used in graph evaluation: it sums up to the number of edges that differ from the ground truth.
- **Structural Intervention Distance**: refers to an adaptation of the SHD metric for causal graphs (Peters and Bühlmann, 2013). It counts the number of different causal paths between connected variables.

Finally, a t-test is used to assess whether the score difference between the best methods is statistically significant with a p-value below 0.001.

Pairwise causal discovery baselines CGNN is assessed comparatively in the pairwise setting to the following algorithms:[5] i) ANM (Mooij et al., 2016) with Gaussian process regression and HSIC independence test of the residual; ii) a pairwise version of LiNGAM (Shimizu et al., 2006) relying on Independent Component Analysis to identify the linear relations between variables; iii) IGCI (Daniusis et al., 2012) with entropy estimator and Gaussian reference measure; iv) the post-nonlinear model (PNL) with HSIC test (Zhang and Hyvärinen, 2009); v) GPI-MML (Stegle et al., 2010); where the Gaussian process regression with higher marginal likelihood is selected as causal direction; vi) CDS, retaining the causal orientation with lowest variance of the conditional probability distribution; vii) Jarfo (Fonollosa, 2016), using a random forest causal classifier trained from the ChaLearn Cause-effect pairs on top of 150 features including ANM, IGCI, CDS, LiNGAM, regressions, HSIC tests. Details of the approaches are given in Section 2.5.3.

Multivariate causal discovery baselines We compare CGNN to the PC algorithm (Spirtes et al., 1993), the score-based methods GES (Chickering, 2002), LiNGAM (Shimizu et al., 2006), causal additive model (CAM) (Bühlmann et al., 2014) and with the pairwise methods ANM and Jarfo. For PC, we employ the better-performing, order-independent version of the PC algorithm proposed by (Colombo and Maathuis, 2014). PC needs the specification of a conditional independence test. We compare PC-Gaussian, which employs a Gaussian conditional independence test on Fisher z-transformations, and PC-HSIC, which uses the HSIC conditional independence test with the Gamma approximation (Gretton et al., 2005b). PC and GES are implemented in the *pcalg* package (Kalisch et al., 2012). All hyperparameters are set on the training graphs in order to maximize the Area Under the Precision/Recall score (AUPR). For the Gaussian conditional independence test and the HSIC conditional independence test, the significance level achieving best result on the training set are respectively 0.1 and 0.05. For GES, the penalization parameter is set to 3 on the training set. For CAM, the cutoff value is set to 0.001.

[5] Using the R program available at `https://github.com/ssamot/causality` for ANM, IGCI, PNL, GPI and LiNGAM.

4.5 Experimental validation on toy examples

This section reports on the empirical validation of CGNNs compared to state-of-the-art algorithms, under the "no confounding assumption", following the experimental setting defined in Section 4.4. First, we check experimentally in Section 4.5.1 that CGNNs can handle the bivariate case, hence do not need to rely on conditional independence to recover the causal direction in identifiable cases. We then check in Section 4.5.2 that CGNNs can identify v-structures when only conditional independence can be relied upon to detect the causal structure because linear Gaussian models are used (a known case in which the bivariate case is not identifiable).

4.5.1 Learning bivariate causal structures

As said, under the causal sufficiency assumption, a dependency between variables X and Y exists iff either X causes Y ($Y = f(X, E)$) or Y causes X ($X = f(Y, E)$). The identification of a *Bivariate Structural Causal Model* is based on comparing the model scores (Section 4.2) attached to both CGNNs.

Benchmarks. Five datasets with continuous variables are considered:[6]

- **CE-Cha**: 300 continuous variable pairs from the cause effect pair challenge (Guyon, 2013), restricted to pairs with label +1 ($X \to Y$) and -1 ($Y \to X$).
- **CE-Net**: 300 artificial pairs generated with a neural network initialized with random weights and random distribution for the cause (exponential, gamma, lognormal, laplace...).
- **CE-Gauss**: 300 artificial pairs without confounder sampled with the generator of Mooij et al. (2016): $Y = f_Y(X, E_Y)$ and $X = f_X(E_X)$ with $E_X \sim p_{E_X}$ and $E_Y \sim p_{E_Y}$. p_{E_X} and p_{E_Y} are randomly generated Gaussian mixture distributions. Causal mechanism f_X and f_Y are randomly generated Gaussian processes.
- **CE-Multi**: 300 artificial pairs generated with linear and polynomial mechanisms. The effect variables are built with post additive noise setting ($Y = f(X) + E$), post multiplicative noise ($Y = f(X) \times E$), pre-additive noise ($Y = f(X + E)$) or pre-multiplicative noise ($Y = f(X \times E)$).
- **CE-Tueb**: 99 real-world cause-effect pairs from the *Tuebingen cause-effect pairs* dataset, version August 2016 (Mooij et al., 2016). This version of this dataset is taken from 37 different data sets coming from various domain: climate, census, medicine data.

For all variable pairs, the size n of the data sample is set to 1,500 for the sake of an acceptable overall computational load.

Hyper-parameter selection. For a fair comparison, a leave-one-dataset-out procedure is used to select the key best hyper-parameter for each algorithm. To avoid computational explo-

[6] The first four datasets are available at `http://dx.doi.org/10.7910/DVN/3757KX`. The *Tuebingen cause-effect pairs* dataset is available at `https://webdav.tuebingen.mpg.de/cause-effect/`

sion, a single hyper-parameter per algorithm is adjusted in this way; other hyper-parameters are set to their default value.

For CGNN, n_h ranges over $\{5, \ldots, 100\}$. The leave-one-dataset-out procedure sets this hyper-parameter n_h to values between 20 and 40 for the different datasets. For ANM and the bivariate fit, the kernel parameter for the Gaussian process regression ranges over $\{0.01, \ldots, 10\}$. For PNL, the threshold parameter alpha for the HSIC independence test ranges over $\{0.0005, \ldots, 0.5\}$. For CDS, the $ffactor$ involved in the discretization step ranges over $[[1, 10]]$. For GPI-MML, its many parameters are set to their default value as none of them appears to be more critical than others. Jarfo is trained from 4,000 variable pairs datasets with same generator used for **CE-Cha-train**, **CE-Net-train**, **CE-Gauss-train** and **CE-Multi-train**; the causal classifier is trained on all datasets except the test set.

Empirical results. Figure 4.4 reports the area under the precision/recall curve for each benchmark and all algorithms.

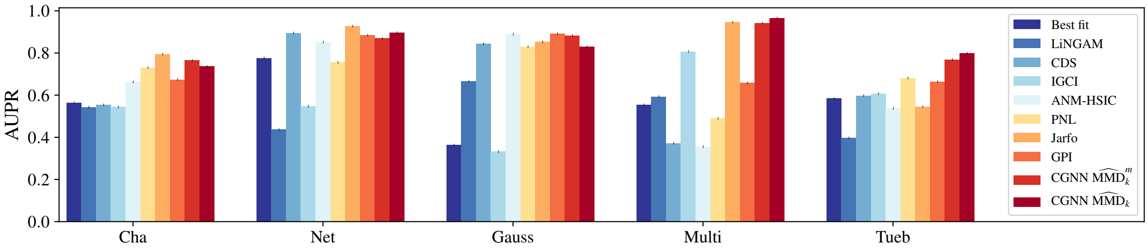


Figure 4.4: Bivariate Causal Modelling: Area under the precision/recall curve for the five datasets (the higher the better). A full table of the scores is given in Appendix 4.4. CGNN manages to obtain good scores for all datasets, and attain best performance on the polynomial and Tuebingen datasets.

Methods based on simple regression like the bivariate fit and Lingam are outperformed as they underfit the data generative process. CDS and IGCI obtain very good results on some datasets. Typically, IGCI takes advantage of some specific features of the dataset, (e.g. the cause entropy being lower than the effect entropy in **CE-Multi**), but remains at chance level otherwise. ANM-HSIC yields good results when the additive assumption holds (e.g. on **CE-Gauss**), but fails otherwise. PNL, less restrictive than ANM, yields overall good results compared to the former methods. Jarfo, a voting procedure, can in principle yield the best of the above methods and does obtain good results on artificial data. However, it does not perform well on the real dataset **CE-Tueb**; this counter-performance is blamed on the differences between all five benchmark distributions and the lack of generalization / transfer learning.

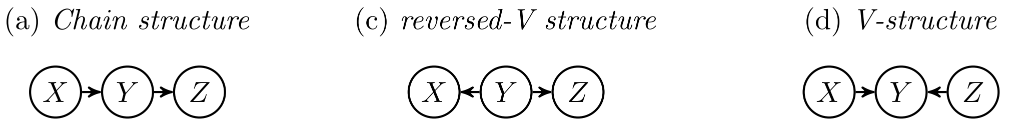
Lastly, generative methods GPI and **CGNN** ($\widehat{\mathrm{MMD}}_k$) perform well on most datasets, including the real-world cause-effect pairs CE-Tüb, in counterpart for a higher computational cost (resp. 32 min on CPU for GPI and 24 min on GPU for CGNN). Using the linear MMD approximation (Lopez-Paz, 2016), **CGNN** ($\widehat{\text{MMD}}_k^m$ as explained in Section 3.2.1) reduces the cost by a factor of 5 without hindering the performance.

Overall, CGNN demonstrates competitive performance on the cause-effect inference problem, where it is necessary to discover distributional asymmetries.

4.5.2 Identifying v-structures

A second series of experiments is conducted to investigate the method performances on variable triplets, where multivariate effects and conditional variable independence must be taken into account to identify the Markov equivalence class of a DAG. The considered setting is that of variable triplets (A, B, C) in the linear mechanisms and Gaussian input/noise case, where asymmetries between cause and effect cannot be exploited (Shimizu et al., 2006) and conditional independence tests are required. In particular strict pairwise methods can hardly be used due to un-identifiability (as each pair involves a linear mechanism with Gaussian input and additive Gaussian noise) (Hoyer et al., 2009).

With no loss of generality, the graph skeleton involving variables (A, B, C) is $A - B - C$. All three causal models (up to variable renaming) based on this skeleton are used to generate 500-sample datasets, where the random noise variables are independent centered Gaussian variables.



(a) *Chain structure* (c) *reversed-V structure* (d) *V-structure*

$$\begin{cases} X = E_X \\ Y = X + E_Y \\ Z = Y + E_Z \end{cases}$$

$$\begin{cases} Y = E_Y \\ X = Y + E_X \\ Z = Y + E_Z \end{cases}$$

$$\begin{cases} X = E_X \\ Z = E_Z \\ Y = X + Z + E_Y \end{cases}$$

Figure 4.5: Linear Gaussian datasets generated from the three DAG configurations with skeleton $A - B - C$

Given skeleton $X - Y - Z$, each dataset is used to model the possible four CGNN structures (Fig. 4.5, with generative SEMs):

- Chain structures XYZ ($X = f_1(E_1)$, $Y = f_2(X, E_2)$, $Z = f_3(Y, E_3)$ and ZYX ($Z = f_1(E_1)$, $Y = f_2(Z, E_2)$, $X = f_3(Y, E_3)$)
- V structure: $X = f_1(E_1)$, $Z = f_2(E_2)$, $Y = f_3(X, Z, E_3)$
- reversed V structure: $Y = f_1(E_1)$, $X = f_2(Y, E_2)$, $Z = f_3(Y, E_3)$

Let C_{XYZ}, C_{ZYX}, $C_{V-structure}$ and $C_{reversedV}$ denote the scores of the CGNN models respectively attached to these structures. The scores computed on all three datasets are displayed in Table 4.2 (average over 64 runs; the standard deviation is indicated in parenthesis).

	non V-structures		V structure
Score	Chain str.	Reversed-V str.	V-structure
C_{XYZ}	**0.122 (0.009)**	**0.124 (0.007)**	0.172 (0.005)
C_{ZYX}	**0.121 (0.006)**	**0.127 (0.008)**	0.171 (0.004)
$C_{reversedV}$	**0.122 (0.007)**	**0.125 (0.006)**	0.172 (0.004)
$C_{V structure}$	0.202 (0.004)	0.180 (0.005)	**0.127** (0.005)

Table 4.2: CGNN-MMD scores for all models on all datasets. Smaller scores indicate a better match. CGNN correctly identifies V-structure vs. other structures.

CGNN scores support a clear and significant discrimination between the V-structure and all other structures (noting that the other structures are Markov equivalent and thus can hardly be distinguished).

This second series of experiments thus shows that CGNN can effectively detect, and take advantage of, conditional independence between variables.

4.6 Experiments on multivariate causal modeling

This section reports on further experiments under the "no confounding assumption", this time investigating larger multivariate problems. Let $\mathbf{X} = [X_1, ..., X_d]$ be a set of continuous variables, satisfying the Causal Markov, faithfulness and causal sufficiency assumptions. In the following, the experiments provide algorithms with *the true graph skeleton*, so their ability to orient edges is compared in a fair way with state-of-the-art algorithms. This allows us to separate the task of orienting the graph from that of uncovering the skeleton.

4.6.1 Results on artificial graphs with additive and multiplicative noises

We draw 500 samples from 20 training artificial causal graphs and 20 test artificial causal graphs of 20 variables. Each variable has a number of parents uniformly drawn in $[[0, 5]]$; f_is are randomly generated polynomials involving additive/multiplicative noise.[7]

Figure 4.6 (left) displays the performance on the test set of artificial graphs of all algorithms obtained by starting from the exact skeleton and measured from the AUPR (Area Under the Precision/Recall curve), the Structural Hamming Distance (SHD, the number of edge modifications to transform one graph into another) and the Structural Intervention Distance (SID, the number of equivalent two-variable interventions between two graphs) Peters and Bühlmann (2013).

[7] The data generator is available at https://github.com/GoudetOlivier/CGNN. The datasets considered are available at http://dx.doi.org/10.7910/DVN/UZMB69.

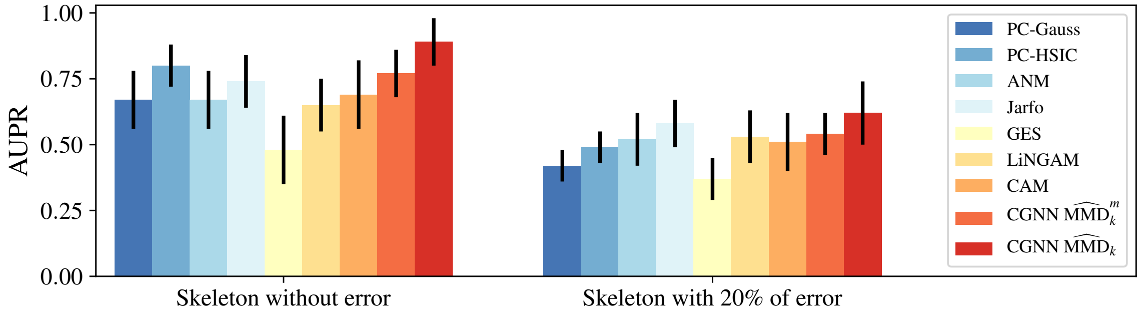


Figure 4.6: Average (std. dev.) AUPR results for the orientation of 20 artificial graphs given true skeleton (left) and artificial graphs given skeleton with 20% error (right). A full table of the scores, including the metrics Structural Hamming Distance (SHD) and Structural Intervention (SID) (Peters and Bühlmann, 2013) is given in Section 4.8.5.

True skeleton initialization CGNN obtains significant better results in terms of SHD and SID compared to the other algorithms when the algorithm start with the skeleton of the true graph. An example of result is displayed on Figure 4.7; there are 3 mistakes on this graph (red edges) (in lines with an SHD on average of 2.5). Constraints based method PC with powerful HSIC conditional independence test is the second best performing method. It highlights the fact that when the skeleton is known, exploiting the structure of the graph leads to good results compared to pairwise methods using only local information. Notably, as seen on Figure 4.7, this type of DAG has a lot of v-structure, as many nodes have more than one parent in the graph, but this is not always the case as shown in the next subsection. Overall CGNN and PC-HSIC are the most computationally expensive methods, taking an average of 4 hours on GPU and 15 hours on CPU, respectively.

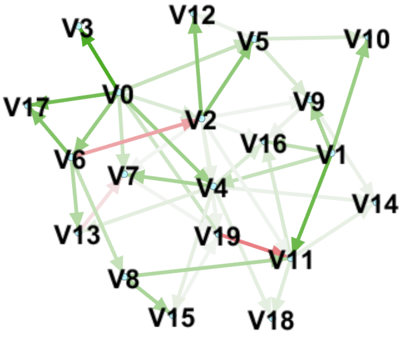


Figure 4.7: Orientation by CGNN of artificial graph with 20 nodes. The color indicates whether the edge is true (green) or false (red). 3 edges are red and 42 are green. The color brightness refers to the confidence of the algorithm.

Initialization with a skeleton containing errors The robustness of the approach is validated by randomly perturbing 20% edges in the graph skeletons provided to all algorithms (introducing about 10 false edges over 50 in each skeleton). As shown on Table 4 (right) in Appendix, and as could be expected, the scores of all algorithms are lower when spurious edges are introduced. Among the least robust methods are constraint-based methods; a tentative explanation is that they heavily rely on the graph structure to orient edges. By comparison pairwise methods are more robust because each edge is oriented separately. As CGNN leverages conditional independence but also distributional asymmetry like pairwise methods, it obtains overall more robust results when there are errors in the skeleton compared to PC-HSIC. However the best performance in terms of SHD score is obtained by CAM, on the skeletons with 20% error. This is due to the exclusive last edge pruning step of CAM, which removes spurious links in the skeleton. CGNN obtains overall good results on these artificial datasets. To explore the scalability of the approach, 5 artificial graphs with 100 variables have been considered, achieving an AUPRC of 85.5 ± 4, in 30 hours of computation on four NVIDIA 1080Ti GPUs.

4.6.2 Results on synthetic biological data

We now evaluate CGNN on biological networks data. First we apply it on simulated gene expression data and then on real protein network.

Syntren artificial simulator First we apply CGNN on SynTREN (Van den Bulcke et al., 2006) from sub-networks of E. coli (Shen-Orr et al., 2002)[8]. SynTREN creates synthetic transcriptional regulatory networks and produces simulated gene expression data that approximates experimental data. Interaction kinetics are modeled by complex mechanisms based on Michaelis-Menten and Hill kinetics (Mendes et al., 2003).

With Syntren, we simulate 20 subnetworks of 20 nodes and 5 subnetworks with 50 nodes. For the sake of reproducibility, we use the random seeds of $0, 1 \ldots 19$ and $0, 1 \ldots 4$ for each graph generation with respectively 20 nodes and 50 nodes. The default Syntren parameters are used: a probability of 0.3 for complex 2-regulator interactions and a value of 0.1 for Biological noise, experimental noise and Noise on correlated inputs. For each graph, Syntren give us expression datasets with 500 samples.

Figure 4.8 (left and middle) and Table 4.6 in Section 4.8.5 display the performance of all algorithms obtained by starting from the exact skeleton of the causal graph with same hyper-parameters as in the previous subsection[9].

Constraint based methods obtain low score on this type of graph dataset. It may be explained by the type of structure involved. Indeed as seen of Figure 4.9, there are very few v-structures in this type of network, making impossible the orientation of an important number of edges by using only conditional independence tests. Overall the methods CAM and

[8] available publicly as a R package named GRNdata.

[9] Except for PC-HSIC, that had to be stopped after 50 hours of running time for a single execution

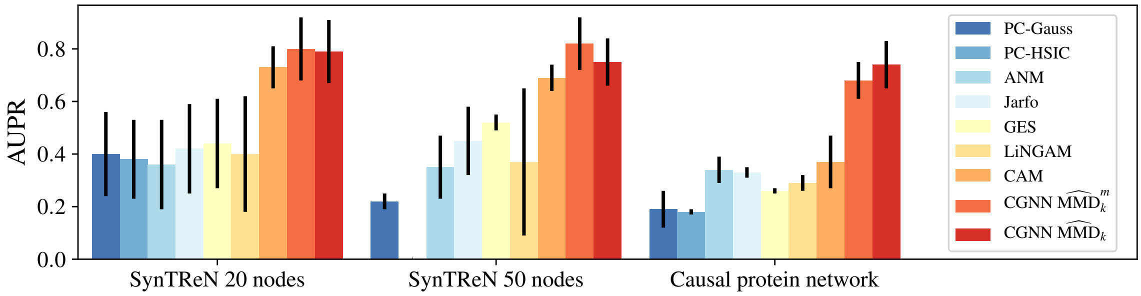


Figure 4.8: Average (std. dev.) AUPR results for the orientation of 20 artificial graphs generated with the SynTReN simulator with 20 nodes (left), 50 nodes (middle), and real protein network given true skeleton (right). A full table of the scores, including the metrics Structural Hamming Distance (SHD) and Structural Intervention (SID) (Peters and Bühlmann, 2013) is included in Section 4.8.5.

CGNN that take into account of both distributional asymmetry and multivariate interactions, get the best scores. CGNN obtain the best results in AUPR, SHD and SID for graph with 20 nodes and 50 nodes, showing that this method can be used to infer networks having complex distribution, complex causal mechanisms and interactions. The Figure 4.9 shows the resulting graph obtained with CGNN. Edges with good orientation are displayed in green and edge with false orientation in red.

4.6.3 Results on biological real-world data

CGNN is applied to the protein network problem (Sachs et al., 2005), using the Anti-CD3/CD28 dataset with 853 observational data points corresponding to general perturbations without specific interventions. All algorithms were given the skeleton of the causal graph (Sachs et al., 2005, Fig. 2) with same hyper-parameters as in the previous subsection. The results are measured after a 10-fold cross-validation (Detailed results in Appendix of this chapter, Table 6).

Constraint-based algorithms obtain surprisingly low scores, because they cannot identify many V-structures in this graph. Notably conditional independence tests for the adjacent tuples of nodes *pip3-akt-pka*, *pka-pmek-pkc*, *pka-raf-pkc* and do not yield strong evidences for V-structure. Therefore methods based on distributional asymmetry between cause and effect seem better suited to this dataset. CGNN obtains good results compared to the other algorithms. Notably, Figure 4.10 shows that CGNN, like CAM, is able to recover the strong signal transduction pathway *raf*→*mek*→*erk* reported in Sachs et al. (2005) and corresponding to clear direct enzyme-substrate causal effect. CGNN gives important scores for edges with good orientation (green line), and low scores (thinnest edges) to the wrong edges (red line), suggesting that false causal discoveries may be controlled by using the confidence scores defined in Eq. 4.5.

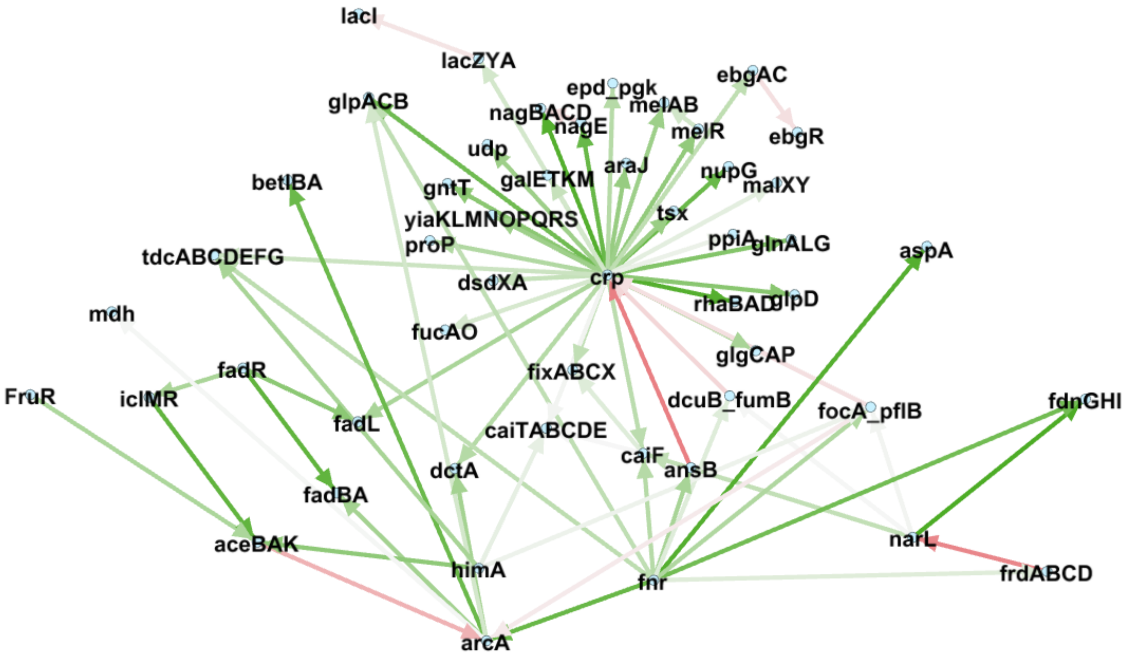


Figure 4.9: Orientation by CGNN of E. coli subnetwork with 50 nodes and corresponding to Syntren simulation with random seed 0. The color indicates whether the edge is true (green) or false (red). The color brightness refers to the confidence of the algorithm.

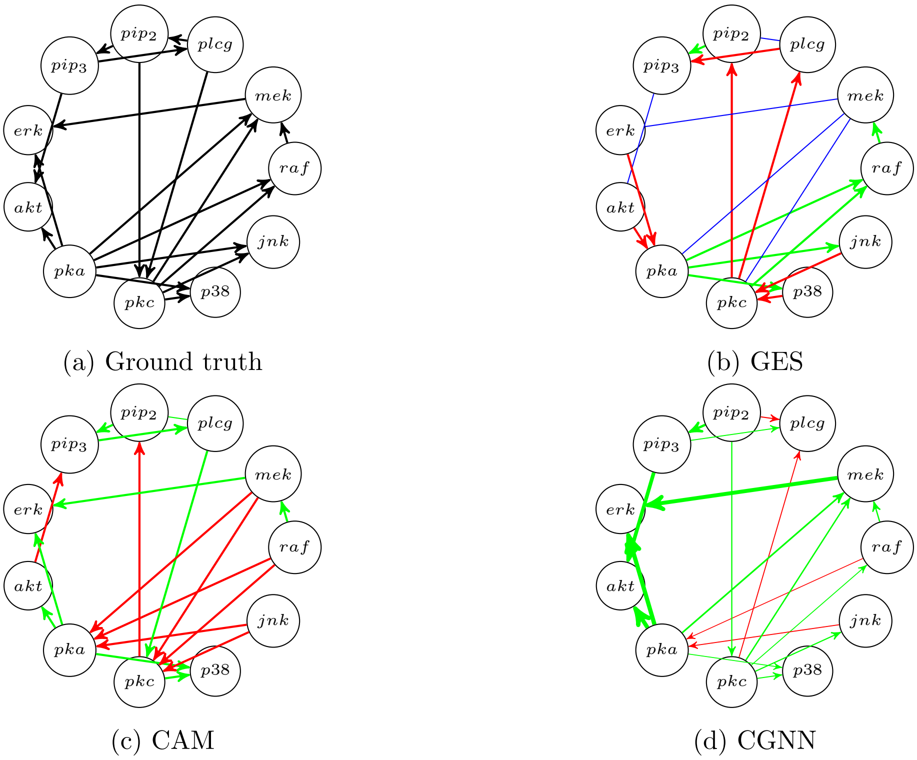
4.7 Towards predicting confounding effects

In this subsection we propose an extension of our algorithm relaxing the causal sufficiency assumption. We are still assuming the Causal Markov and faithfulness assumptions, thus three options have to be considered for each edge (X_i, X_j) of the skeleton representing a direct dependency: $X_i \to X_j$, $X_j \to X_i$ and $X_i \leftrightarrow X_j$ (both variables are consequences of common hidden variables).

4.7.1 Principle

Hidden common causes are modeled through correlated random noise. Formally, an additional noise variable $E_{i,j}$ is associated to each $X_i - X_j$ edge in the graph skeleton.

We use such new models with correlated noise to study the robustness of our graph reconstruction algorithm to increasing violations of causal sufficiency, by occluding variables from our datasets. For example, consider the FCM on $\mathbf{X} = [X_1, \ldots, X_5]$ that was presented on Figure 2.1. If variable X_1 would be missing from data, the correlated noise $E_{2,3}$ would be responsible for the existence of a double headed arrow connection $X_2 \leftrightarrow X_3$ in the skeleton of our new type of model. The resulting FCM is shown in Figure 4.11. Notice that direct causal effects such as $X_3 \to X_5$ or $X_4 \to X_5$ may persist, even in presence of possible confounding effects.



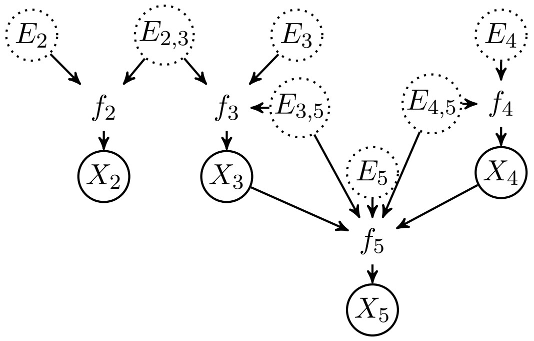
(a) Ground truth

(b) GES

(c) CAM

(d) CGNN

Figure 4.10: Causal protein network



$$\begin{cases} X_2 = f_2(E_2, E_{2,3}) \\ X_3 = f_3(E_3, E_{2,3}, E_{3,5}) \\ X_4 = f_4(E_4, E_{4,5}) \\ X_5 = f_5(X_3, X_4, E_5, E_{3,5}, E_{4,5}) \end{cases}$$

Figure 4.11: The Functional Causal Model (FCM) on $\mathbf{X} = [X_1, \ldots, X_5]$ with the missing variable X_1

Formally, given a graph skeleton $\mathcal{S}$, the FCM with correlated noise variables is defined as:

$$X_i \leftarrow f_i(X_{\mathrm{Pa}(i;\mathcal{G})}, E_i, E_{\mathrm{Ne}(i;\mathcal{S})}), \tag{4.6}$$

where $\mathrm{Ne}(i;\mathcal{S})$ is the set of indices of all the variables adjacent to variable X_i in the skeleton $\mathcal{S}$.

One can notice that this model corresponds to the most general formulation of the FCM with potential confounders for each pair of variables in a given skeleton (representing direct dependencies) where each random variable $E_{i,j}$ summarizes all the unknown influences of (possibly multiple) hidden variables influencing the two variables X_i and X_j.

Here we make a clear distinction between the directed acyclic graph denoted $\mathcal{G}$ and the skeleton $\mathcal{S}$. Indeed, due to the presence of confounding correlated noise, any edge in $\mathcal{G}$ can be removed without altering $\mathcal{S}$ (by replacing an edge $X_i - X_j$ with $X_i \leftarrow E_{ij} \rightarrow X_j$) if a common noise can explain the dependencies between the variables. We use the same generative neural network to model the new FCM presented in Equation 4.6. The difference is the new noise variables having effect on pairs of variables simultaneously. However, since the correlated noise FCM is still defined over a directed acyclic graph $\mathcal{G}$, the model can still be learned end-to-end using backpropagation based on the CGNN loss.

All edges are evaluated with these correlated noises, the goal being to see whether introducing a correlated noise explains the dependence between the two variables X_i and X_j.

As mentioned before, the score used by CGNN is:

$$S(\mathcal{C}_{\widehat{\mathcal{G}},\widehat{f}}, \mathcal{D}) = \widehat{\mathrm{MMD}}_k(\mathcal{D}, \widehat{\mathcal{D}}) + \lambda|\widehat{\mathcal{G}}| \tag{4.7}$$

where $|\widehat{\mathcal{G}}|$ is the total number of edges in the DAG. In the graph search, for any given edge, we compare the score associated to the graph considered with and without this edge, by replacing it with connections to a correlated noise. If the contribution of this edge is negligible compared to a given threshold lambda, the edge is considered as spurious.

The non-parametric optimization of the $\widehat{\mathcal{G}}$ structure is also achieved using a Hill-Climbing algorithm; in each step an edge of $\mathcal{S}$ is randomly drawn and modified in $\widehat{\mathcal{G}}$ using one out of the possible three operators: reverse the edge, add an edge and remove an edge. Other algorithmic details are as in Section 4.3.2: the greedy search optimizes the penalized loss function (Eq. 4.7). For CGNN, the hyperparameter λ is set to 5×10^{-5} fitted on the training graph dataset.

The algorithm stops when no improvement is obtained. Finally, each causal edge $X_i \rightarrow X_j$ in the final graph $\mathcal{G}$ is associated with a score, measuring its contribution to the global score[10]:

$$S_{X_i \rightarrow X_j} = S(\mathcal{C}_{\widehat{\mathcal{G}}-\{X_i \rightarrow X_j\}, \widehat{f}}, \mathcal{D}) - S(\mathcal{C}_{\widehat{\mathcal{G}}, \widehat{f}}, \mathcal{D}) \tag{4.8}$$

4.7.2 Experimental validation

Benchmarks. The empirical validation of this extension of CGNN is conducted on same benchmarks as in Section 4.6 ($\mathcal{G}_i$, $i \in [[2, 5]]$), where 3 variables (causes for at least two other variables in the graph) have been randomly removed.[11] The true graph skeleton is augmented with edges $X - Y$ for all X, Y that are consequences of a same removed cause. All algorithms are provided with the same graph skeleton for a fair comparison. The task is to both orient the edges in the skeleton, and remove the spurious direct dependencies created by latent causal variables.

Baselines. CGNN is compared with state of art methods: i) constraint-based RFCI (Colombo et al., 2012), extending the PC method equipped with Gaussian conditional independence test (RFCI-Gaussian) and the gamma HSIC conditional independence test (Gretton et al., 2005b) (RFCI-HSIC). We use the order-independent constraint-based version proposed by Colombo and Maathuis (2014) and the majority rules for the orientation of the edges. For CGNN, we set the hyperparameter $\lambda = 5 \times 10^{-5}$ fitted on the training graph dataset. Jarfo is trained on the 16,200 pairs of the cause-effect pair challenge (Guyon, 2013, 2014) to detect for each pair of variable if $X_i \rightarrow Y_i$, $Y_i \rightarrow X_i$ or $X_i \leftrightarrow Y_i$.

Table 4.3: AUPR (the higher the better), SHD and SID (the lower the better) on causal discovery with confounders. Significantly better results(t-test with p-value $p < 10^{-2}$) are underlined.

method	AUPR	SHD	SID
RFCI-Gaussian	0.22 (0.08)	21.9 (7.5)	174.9 (58.2)
RFCI-HSIC	0.41 (0.09)	17.1 (6.2)	124.6 (52.3)
Jarfo	0.54 (0.21)	20.1 (14.8)	98.2 (49.6)
CGNN ($\widehat{\mathrm{MMD}}_k$)	<u>**0.71**</u> (0.13)	<u>**11.7**</u> (5.5)	<u>**53.55**</u> (48.1)

[10] Edges finally not present in $\mathcal{G}$ are associated with a confidence score 0 .

[11] The datasets considered are available at http://dx.doi.org/10.7910/DVN/UZMB69

Results. Comparative performances are shown in Table 4.3, reporting the area under the precision/recall curve. Overall, these results confirm the robustness of the CGNN proposed approach w.r.t. confounders, and its competitiveness w.r.t. RFCI with powerful conditional independence test (RFCI-HSIC). Interestingly, the effective causal relations between the visible variables are associated with a high score; spurious links due to hidden latent variables get a low score or are removed.

Partial conclusion

This chapter introduced CGNN, a new framework and methodology for functional causal model learning, leveraging the power and non-parametric flexibility of Generative Neural Networks. CGNN provides good and consistent performance when starting the algorithm with a true skeleton but also when starting with a perturbed skeleton. Once the model is learned, the CGNNs present the advantage to be fully parametrized and may be used to simulate interventions on one or more variables of the model and evaluate their impact on a set of target variables. This usage is relevant in a wide variety of domains, typically among medical and sociological domains.

CGNN seamlessly accommodates causal modeling in presence of confounders, and its extensive empirical validation demonstrates its merits compared to the state of the art on medium-size problems. We believe that our approach opens new avenues of research, both from the point of view of leveraging the power of deep learning in causal discovery and from the point of view of building deep networks with better structure *interpretability*.

The main limitation of CGNN is its computational cost, due to the quadratic complexity of the CGNN learning criterion w.r.t. the data size, based on the Maximum Mean Discrepancy between the generated and the observed data. A linear approximation thereof has been proposed, with comparable empirical performances.

The main perspective for further research aims at a better scalability of the approach from medium to large problems. On the one hand, the computational scalability could be tackled by using embedded framework for the structure optimization, inspired by Lasso methods (Friedman et al., 2008), as studied in the next chapter.

4.8 Appendix

4.8.1 Proof of Theorem 2

Theorem 1. *Let $\boldsymbol{X} = [X_1, \dots, X_d]$ denote a set of continuous random variables with joint distribution P, and further assume that the joint density function h of P is continuous and strictly positive on a compact and convex subset of $\mathbb{R}^d$, and zero elsewhere. Letting $\mathcal{G}$ be a DAG such that P can be factorized along $\mathcal{G}$,*

$$P(X) = \prod_i P(X_i | X_{Pa(i;\mathcal{G})})$$

there exists $f = (f_1, \dots, f_d)$ *with* f_i *a continuous function with compact support in* $\mathbb{R}^{|Pa(i;\mathcal{G})|} \times [0,1]$ *such that* $P(X)$ *equals the generative model defined from FCM* $(\mathcal{G}, f, \mathcal{E})$*, with* $\mathcal{E} = \mathcal{U}[0,1]$ *the uniform distribution on* $[0,1]$*.*

Proof. By induction on the topological order of $\mathcal{G}$. Let X_i be such that $|\mathrm{Pa}(i;\mathcal{G})| = 0$ and consider the cumulative distribution $F_i(x_i)$ defined over the domain of X_i:

$$F_i(x_i) = Pr(X_i < x_i)$$

F_i is strictly monotonous as the joint density function is strictly positive therefore its inverse, the quantile function $Q_i : [0,1] \mapsto dom(X_i)$ is defined and continuous. By construction, $Q_i(e_i) = F_i^{-1}(e_i)$ and setting $Q_i = f_i$ yields the result.

Assume f_i be defined for all variables X_i with topological order less than m. Let X_j with topological order m and Z the vector of its parent variables.

For any noise vector $e = (e_i, i \in \mathrm{Pa}(j;\mathcal{G}))$ let $z = (x_i, i \in \mathrm{Pa}(j;\mathcal{G}))$ be the value vector of variables in Z defined from e. The conditional cumulative distribution $F_j(x_j|Z = z) = Pr(X_j < x_j|Z = z)$ is strictly continuous and monotonous wrt x_j, and can be inverted using the same argument as above. Then we can define

$$f_j(z, e_j) = F_j^{-1}(z, e_j)$$

Let $K_j = dom(X_j)$ and $K_{\mathrm{Pa}(j;\mathcal{G})} = dom(Z)$. We will show now that the function f_j is continuous on $K_{\mathrm{Pa}(j;\mathcal{G})} \times [0,1]$, a compact subset of $\mathbb{R}^{|\mathrm{Pa}(j;\mathcal{G})|} \times [0,1]$.

By assumption, there exists $a_j \in \mathcal{R}$ such that:

$$F(x_j|z) = \int_{a_j}^{x_j} \frac{h_j(u,z)}{h_j(z)} \mathrm{d}u \quad \text{for } (x_j, z) \in K_j \times K_{\mathrm{Pa}(j;\mathcal{G})},$$

with h_j the continuous and strictly positive joint density function. For $(a,b) \in K_j \times K_{\mathrm{Pa}(j;\mathcal{G})}$, as the function $(u,z) \to \frac{h_j(u,z)}{h_j(z)}$ is continuous on the compact $K_j \times K_{\mathrm{Pa}(j;\mathcal{G})}$,

$$\begin{array}{ll} \lim\limits_{x_j \to a} F(x_j|z) = \int_{a_j}^{a} \frac{h_j(u,z)}{h_j(z)} \mathrm{d}u & \text{uniformly on } K_{\mathrm{Pa}(j;\mathcal{G})} \\ \lim\limits_{z \to b} F(x_j|z) = \int_{a_j}^{x_j} \frac{h_j(u,b)}{h_j(b)} & \text{on } K_j \end{array}$$

thus according to exchanging limits theorem, F is continuous on (a,b).

For any sequence $z_n \to z$, we have that $F(x_j|z_n) \to F(x_j|z)$ uniformly in x_j. Let define two sequences u_n and $x_{j,n}$, respectively on $[0,1]$ and K_j, such that:

$$\begin{cases} u_n \to u \\ x_{j,n} \to x_j \end{cases}$$

As $F(x_j|z) = u$ has unique root $x_j = f_j(z,u)$, the root of $F(x_j|z_n) = u_n$, that is, $x_{j,n} = f_j(z_n, u_n)$ converge to x_j. Then the function $(z,u) \to f_j(z,u)$ is continuous on $K_{\mathrm{Pa}(i;\mathcal{G})} \times [0,1]$. □

4.8.2 Proof of Theorem 3

Theorem 2. *For $m \in [[1, d]]$, let Z_m denote the set of variables with topological order less than m and let d_m be its size. For any d_m-dimensional vector of noise values $e^{(m)}$, let $z_m(e^{(m)})$ (resp. $\widehat{z_m}(e^{(m)})$) be the vector of values computed in topological order from the FCM $(\mathcal{G}, f, \mathcal{E})$ (resp. the CGNN $(\mathcal{G}, \widehat{f}, \mathcal{E})$). For any $\epsilon > 0$, there exists a set of networks $\widehat{f}$ with architecture $\mathcal{G}$ such that*

$$\forall e^{(m)}, \|z_m(e^{(m)}) - \widehat{z_m}(e^{(m)})\| < \epsilon \tag{4.9}$$

Proof. By induction on the topological order of $\mathcal{G}$. Let X_i be such that $|\mathrm{Pa}(i; \mathcal{G})| = 0$. Following the universal approximation theorem Cybenko (1989), as f_i is a continuous function over a compact of $\mathbb{R}$, there exists a neural net $\widehat{f_i}$ such that

$$\|f_i - \widehat{f_i}\|_\infty < \epsilon/d_1$$

Thus Eq. 4.4 holds for the set of networks $\widehat{f_i}$ for i ranging over variables with topological order 0.

Let us assume that Prop. 2 holds up to m, and let us assume for brevity that there exists a single variable X_j with topological order $m + 1$. Letting $\widehat{f_j}$ be such that

$$\|f_j - \widehat{f_j}\|_\infty < \epsilon/3$$

based on the universal approximation property, letting δ be such that for all u

$$\|\hat{f}_j(u) - \hat{f}_j(u + \delta)\| < \epsilon/3$$

by absolute continuity and letting $\hat{f}_i$ satisfying Eq. 4.4 for i with topological order less than m for $min(\epsilon/3, \delta)/d_m$, it comes:

$$\begin{aligned}\|(z_m, f_j(z_m, e_j)) - (\hat{z}_m, \widehat{f_j}(\widehat{z_m}, e_j))\| &\le \|z_m - \hat{z}_m\| + |f_j(z_m, e_j) - \widehat{f_j}(z_m, e_j)| \\ &\quad + |\widehat{f_j}(z_m, e_j) - \widehat{f_j}(\widehat{z_m}, e_j)| \\ &< \epsilon/3 + \epsilon/3 + \epsilon/3\end{aligned} \tag{4.10}$$

□

4.8.3 Proof of Theorem 4

Theorem 3. *Let $\mathcal{D}$ be an infinite observational sample generated from $(\mathcal{G}, f, \mathcal{E})$. With same notations as in Prop. 2, for every sequence ϵ_t such that $\epsilon_t > 0$ goes to zero when $t \to \infty$, there exists a set $\widehat{f_t} = (\hat{f}_1^t \dots \hat{f}_d^t)$ such that $\widehat{MMD_k}$ between $\mathcal{D}$ and an infinite size sample $\widehat{\mathcal{D}}_t$ generated from the CGNN $(\mathcal{G}, \widehat{f_t}, \mathcal{E})$ is less than ϵ_t.*

Proof. According to Prop. 3 and with same notations, letting $\epsilon_t > 0$ go to 0 as t goes to infinity, consider $\hat{f}_t = (\hat{f}_1^t \dots \hat{f}_d^t)$ and $\widehat{z}_t$ defined from $\hat{f}_t$ such that for all $e \in [0, 1]^d$,

$$\|z(e) - \widehat{z}_t(e)\| < \epsilon_t$$

Let $\{\widehat{\mathcal{D}_t}\}$ denote the infinite sample generated after $\widehat{f}_t$. The score of the CGNN $(\mathcal{G}, \widehat{f}_t, \mathcal{E})$ is

$$\widehat{\mathrm{MMD}}_k(\mathcal{D}, \widehat{\mathcal{D}_t}) = \mathbb{E}_{e,e'}\left[k\left(z(e), z(e')\right) - 2k\left(z(e), \widehat{z}_t(e')\right) + k\left(\widehat{z}_t(e), \widehat{z}_t(e')\right)\right] \tag{4.11}$$

As $\widehat{f}_t$ converges towards f on the compact $[0,1]^d$, using the bounded convergence theorem on a compact subset of $\mathbb{R}^d$, $\widehat{z}_t(e) \to z(e)$ uniformly for $t \to \infty$, it follows from the Gaussian kernel function being bounded and continuous that $\widehat{\mathrm{MMD}}_k(\mathcal{D}, \widehat{\mathcal{D}_t}) \to 0$, when $t \to \infty$. □

4.8.4 Proof of Theorem 5

Theorem 4. *Let $\boldsymbol{X} = [X_1, \ldots, X_d]$ denote a set of continuous random variables with joint distribution P, generated by a CGNN $\mathcal{C}_{\mathcal{G},f} = (\mathcal{G}, f, \mathcal{E})$ with $\mathcal{G}$, a directed acyclic graph. And let $\mathcal{D}$ be an infinite observational sample generated from this CGNN. We assume that P is Markov and faithful to the graph $\mathcal{G}$, and that every pair of variables (X_i, X_j) that are d-connected in the graph are not independent. We note $\widehat{\mathcal{D}}$ an infinite sample generated by a candidate CGNN, $\mathcal{C}_{\widehat{\mathcal{G}},\widehat{f}} = (\widehat{\mathcal{G}}, \widehat{f}, \mathcal{E})$. Then,*
(i) If $\widehat{\mathcal{G}} = \mathcal{G}$ and $\widehat{f} = f$, then $\widehat{MMD}_k(\mathcal{D}, \widehat{\mathcal{D}}) = 0$.
(ii) For any graph $\widehat{\mathcal{G}}$ characterized by the same adjacencies but not belonging to the Markov equivalence class of $\mathcal{G}$, for all $\widehat{f}$, $\widehat{MMD}_k(\mathcal{D}, \widehat{\mathcal{D}}) \neq 0$.

Proof. The proof of (i) is obvious, as with $\widehat{\mathcal{G}} = \mathcal{G}$ and $\widehat{f} = f$, the joint distribution $\widehat{P}$ generated by $\mathcal{C}_{\widehat{\mathcal{G}},\widehat{f}} = (\widehat{\mathcal{G}}, \widehat{f}, \mathcal{E})$ is equal to P, thus we have $\widehat{\mathrm{MMD}}_k(\mathcal{D}, \widehat{\mathcal{D}}) = 0$.

(ii) Let consider $\widehat{\mathcal{G}}$ a DAG characterized by the same adjacencies but that do not belong to the Markov equivalence class of $\mathcal{G}$. According to Verma and Pearl (1991), as the DAG $\mathcal{G}$ and $\widehat{\mathcal{G}}$ have the same adjacencies but are not Markov equivalent, there are not characterized by the same v-structures.

a) First, we consider that a v-structure $\{X, Y, Z\}$ exists in $\mathcal{G}$, but not in $\widehat{\mathcal{G}}$. As the distribution P is faithful to $\mathcal{G}$ and X and Z are not d-separated by Y in $\mathcal{G}$, we have that $(X \not\perp\!\!\!\perp Z|Y)$ in P. Now we consider the graph $\widehat{\mathcal{G}}$. Let $\widehat{f}$ be a set of neural networks. We note $\widehat{P}$ the distribution generated by the CGNN $\mathcal{C}_{\widehat{\mathcal{G}},\widehat{f}}$. As $\widehat{\mathcal{G}}$ is a directed acyclic graph and the variables E_i are mutually independent, $\widehat{P}$ is Markov with respect to $\widehat{\mathcal{G}}$. As $\{X, Y, Z\}$ is not a v-structure in $\widehat{\mathcal{G}}$, X and Z are d-separated by Y. By using the causal Markov assumption, we obtain that $(X \perp\!\!\!\perp Z|Y)$ in $\widehat{P}$.

b) Second, we consider that a v-structure $\{X, Y, Z\}$ exists in $\widehat{\mathcal{G}}$, but not in $\mathcal{G}$. As $\{X, Y, Z\}$ is not a v-structure in $\mathcal{G}$, there is an "unblocked path" between the variables X and Z, the variables X and Z are d-connected. By assumption, there do not exist a set D not containing Y such that $(X \perp\!\!\!\perp Z|D)$ in P. In $\widehat{\mathcal{G}}$, as $\{X, Y, Z\}$ is a v-structure, there exists a set D not containing Y that d-separates X and Z. As for all CGNN $\mathcal{C}_{\widehat{\mathcal{G}},\widehat{f}}$ generating a distribution $\widehat{P}$, $\widehat{P}$ is Markov with respect to $\widehat{\mathcal{G}}$, we have that $X \perp\!\!\!\perp Z|D$ in $\widehat{P}$.

In the two cases a) and b) considered above, P and $\widehat{P}$ do not encode the same conditional independence relations, thus are not equal. We have then $\widehat{\mathrm{MMD}}_k(\mathcal{D}, \mathcal{D}') \neq 0$. □

4.8.5 Detailed results of CGNN experiments

Table 4.4: Cause-effect relations: Area Under the Precision Recall curve on 5 benchmarks for the cause-effect experiments (weighted accuracy in parenthesis for Tüb). Underline values correspond to best scores.

method	Cha	Net	Gauss	Multi	Tüb
Best fit	56.4	77.6	36.3	55.4	58.4 (44.9)
LiNGAM	54.3	43.7	66.5	59.3	39.7 (44.3)
CDS	55.4	89.5	84.3	37.2	59.8 (65.5)
IGCI	54.4	54.7	33.2	80.7	60.7 (62.6)
ANM	66.3	85.1	88.9	35.5	53.7 (59.5)
PNL	73.1	75.5	83.0	49.0	68.1 (66.2)
Jarfo	$\underline{79.5}$	$\underline{92.7}$	85.3	94.6	54.5 (59.5)
GPI	67.4	88.4	$\underline{89.1}$	65.8	66.4 (62.6)
CGNN ($\widehat{\text{MMD}}_k$)	73.6	89.6	82.9	$\underline{96.6}$	$\underline{79.8}$ (74.4)
CGNN ($\widehat{\text{MMD}}_k^m$)	76.5	87.0	88.3	94.2	76.9 (72.7)

Table 4.5: Average (std. dev.) results for the orientation of 20 artificial graphs given true skeleton (left), artificial graphs given skeleton with 20% error (middle). * denotes statistical significance at $p = 10^{-2}$. Underline values correspond to best scores.

	Skeleton without error			Skeleton with 20% of error		
	AUPR	SHD	SID	AUPR	SHD	SID
Constraints						
PC-Gauss	0.67 (0.11)	9.0 (3.4)	131 (70)	0.42 (0.06)	21.8 (5.5)	191.3 (73)
PC-HSIC	0.80 (0.08)	6.7 (3.2)	80.1 (38)	0.49 (0.06)	19.8 (5.1)	165.1 (67)
Pairwise						
ANM	0.67 (0.11)	7.5 (3.0)	135.4 (63)	0.52 (0.10)	19.2 (5.5)	171.6 (66)
Jarfo	0.74 (0.10)	8.1 (4.7)	147.1 (94)	0.58 (0.09)	20.0 (6.8)	184.8 (88)
Score-based						
GES	0.48 (0.13)	14.1 (5.8)	186.4 (86)	0.37 (0.08)	20.9 (5.5)	209 (83)
LiNGAM	0.65 (0.10)	9.6 (3.8)	171 (86)	0.53 (0.10)	20.9 (6.8)	196 (83)
CAM	0.69 (0.13)	7.0 (4.3)	122 (76)	0.51 (0.11)	$\underline{15.6}$ (5.7)	175 (80)
CGNN ($\widehat{\text{MMD}}_k^m$)	0.77 (0.09)	7.1 (2.7)	141 (59)	0.54 (0.08)	20 (10)	179 (102)
CGNN ($\widehat{\text{MMD}}_k$)	$\underline{0.89}$* (0.09)	$\underline{2.5}$* (2.0)	$\underline{50.45}$* (45)	$\underline{0.62}$ (0.12)	16.9 (4.5)	$\underline{134.0}$* (55)

Table 4.6: Average (std. dev.) results for the orientation of 20 and 50 artificial graphs coming from Syntren simulator given true skeleton. * denotes statistical significance at $p = 10^{-2}$. Underline values correspond to best scores.

	Syntren network 20 nodes			Syntren network 50 nodes		
	AUPR	SHD	SID	AUPR	SHD	SID
Constraints						
PC-Gauss	0.40 (0.16)	16.3 (3.1)	198 (57)	0.22 (0.03)	61.5 (32)	993 (546)
PC-HSIC	0.38 (0.15)	23 (1.7)	175 (16)	-	-	-
Pairwise						
ANM	0.36 (0.17)	10.1 (4.2)	138 (56)	0.35 (0.12)	29.8 (13.5)	677 (313)
Jarfo	0.42 (0.17)	10.5 (2.6)	148 (64)	0.45 (0.13)	26.2 (14)	610 (355)
Score-based						
GES	0.44 (0.17)	9.8 (5.0)	116 (64)	0.52 (0.03)	21 (11)	462 (248)
LiNGAM	0.40 (0.22)	10.1 (4.4)	135 (57)	0.37 (0.28)	33.4 (19)	757 (433)
CAM	0.73 (0.08)	4.0 (2.5)	49 (24)	0.69 (0.05)	14.8 (7)	285 (136)
CGNN ($\widehat{\text{MMD}}_k^m$)	$\underline{0.80}$* (0.12)	3.2 (1.6)	45 (25)	$\underline{0.82}$* (0.1)	$\underline{10.2}$* (5.3)	$\underline{247}$ (134)
CGNN ($\widehat{\text{MMD}}_k$)	0.79 (0.12)	$\underline{3.1}$* (2.2)	$\underline{43}$ (26)	0.75 (0.09)	12.2 (5.5)	309 (140)

Table 4.7: Average (std. dev.) results for the orientation of the real protein network given true skeleton. * denotes statistical significance at $p = 10^{-2}$. Underline values correspond to best scores.

	Causal protein network		
	AUPR	SHD	SID
Constraints			
PC-Gauss	0.19 (0.07)	16.4 (1.3)	91.9 (12.3)
PC-HSIC	0.18 (0.01)	17.1 (1.1)	90.8 (2.6)
Pairwise			
ANM	0.34 (0.05)	8.6 (1.3)	85.9 (10.1)
Jarfo	0.33 (0.02)	10.2 (0.8)	92.2 (5.2)
Score-based			
GES	0.26 (0.01)	12.1 (0.3)	92.3 (5.4)
LiNGAM	0.29 (0.03)	10.5 (0.8)	83.1 (4.8)
CAM	0.37 (0.10)	8.5 (2.2)	78.1 (10.3)
CGNN ($\widehat{\text{MMD}}_k^m$)	0.68 (0.07)	5.7 (1.7)	56.6 (10.0)
CGNN ($\widehat{\text{MMD}}_k$)	$\underline{0.74}$* (0.09)	$\underline{4.3}$* (1.6)	$\underline{46.6}$* (12.4)

Chapter 5

Learning a graph end-to-end

This chapter, inspired from Kalainathan et al. (2019), presents the Structural Agnostic Model (SAM). It aims at addressing the computational limitation of CGNN, presented in Chapter 4.

An open-source implementation is available at `https://github.com/Diviyan-Kalainathan/SAM`. Leveraging both conditional independencies and distributional asymmetries in the data, the Structural Agnostic Model (SAM) aims at recovering full causal models from continuous observational data along a multivariate non-parametric setting. The approach is based on a game between d players estimating each variable distribution conditionally to the others as a neural net, and an adversary aimed at discriminating the overall joint conditional distribution, and that of the original data. An original learning criterion combining distribution estimation, sparsity and acyclicity constraints is used to enforce the end-to-end optimization of the graph structure and parameters through stochastic gradient descent. Besides the theoretical analysis of the approach in the large sample limit, SAM is extensively experimentally validated on synthetic and real-world datasets.

5.1 The Structural Agnostic Model (SAM)

This section presents the *Structural Agnostic Model* (SAM), implementing the MDL framework presented in Section 2.2.3 within the space of generative neural networks (NN). The originality of the approach is to implement an end-to-end search for a Functional Causal Model (FCM, Eq. 2.2) with **no restrictive assumption on the underlying causal mechanisms and data distributions**. Specifically, SAM searches for the generative model of every X_i from X_{-i} (all variables from $\mathbf{X}$ but X_i itself), such that:

$$X_i = f_i(X_{-i}, E_i) \tag{5.1}$$

The sparsity of the structure is sought using a Lasso-inspired mechanism (Friedman et al., 2008) with binary coefficients a_{ij} defining the connection between X_j and X_i, such as :

$$X_i = f_i(a_{i1}X_1, \ldots, a_{id}X_d, E_i) \tag{5.2}$$

$$= f_i(\mathbf{a_i} \odot \mathbf{X}, E_i) \tag{5.3}$$

with $a_{ii} = 0$ to avoid X_i from generating itself.

5.1.1 Modeling causal mechanisms with conditional generative neural networks

The model search space includes all joint distributions $q(\mathbf{x})$ defined from a DAG $\widehat{\mathcal{G}}$ and causal mechanisms $\hat{f} = (\hat{f}_1, \ldots, \hat{f}_d)$, with $\hat{f}_j$ a 1-hidden layer NN yielding a generative model of X_j from all other variables in $\mathbf{X}$ (Fig. 5.1). Formally:

- The d-dimensional vector of variables $\mathbf{X}$ is elementwise multiplied with binary vector $\mathbf{a}_j = (a_{1,j}, \ldots a_{d,j})$ named *structural gate*. Coefficient $a_{i,j}$ is 1 *iff* variable X_i is used to generate X_j (with $a_{i,i}$ set to 0 to avoid self-loops), that is, edge $X_i \rightarrow X_j$ is present in graph $\widehat{\mathcal{G}}$, and X_i is considered to be a cause of X_j. Otherwise, $a_{i,j}$ is set to 0. A regularization term on $\mathbf{a}_j$ enforces the graph sparsity.

- The number of active hidden units in neural network $\hat{f}_j$ is controlled by a Boolean vector $\mathbf{z}_j$ of size n_h named *functional gate*, where the h-th entry noted $z_{h,j} \in \{0, 1\}$ corresponds to the activation of the h-th hidden unit of the neural network. Likewise, a regularization on the functional gates is used to limit the complexity of the functional mechanisms.

- At every evaluation of noise variable E_j, a value is drawn anew from distribution $\mathcal{N}(0, 1)$. As already mentioned, the restriction to Gaussian noise is not a limitation in terms of expressivity of the model.

As said, $\hat{f}_j$ is implemented as a 1-hidden layer NN, i.e. a linear combination of non-linear features $\phi_{i,k}$:

$$\begin{aligned} X_j = \hat{f}_j(\mathbf{X}, E_j) &= \sum_{k=1}^{n_h} m_{j,k}\phi_{j,k}(\mathbf{X}, E_j) z_{j,k} + m_{j,0} \\ \text{with } \phi_{j,k}(\mathbf{X}, E_j) &= tanh\left(\sum_{i=1}^{d} W_{j,ik} a_{ij} X_i + b_{j,k} + W_{j,d+1} E_j\right) \end{aligned} \tag{5.4}$$

For notational simplicity, each $\hat{f}_j$ is associated with a parameter vector $\theta_j = (\theta_{j,1}, \ldots, \theta_{j,p_j})$ (including vectors $\mathbf{m}_j$ and $W_{j,\cdot}$ but excluding the $\mathbf{a}_j$ and $\mathbf{z}_j$ gates). With E_j a Gaussian noise variable, each $\hat{f}_j$ thus encodes a generative model of X_j conditionally to variables in $x_{\text{Pa}(j;\widehat{\mathcal{G}})}$, with $\text{Pa}(j;\widehat{\mathcal{G}}) = \{i \in [1, \ldots, d] \ s.t. \ a_{i,j} = 1\}$.

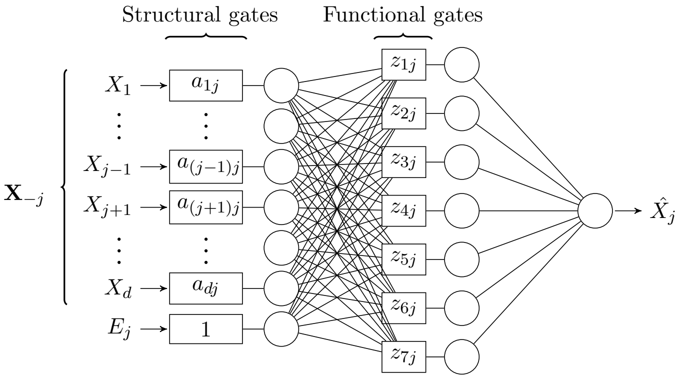


Figure 5.1: Example diagram of the conditional generative neural network modeling the causal mechanism $X_j = \hat{f}_j(\mathbf{X}, E_j)$ with $n_h = 7$.

Under the assumptions that noise variables E_j are independent of each other (Causal Sufficiency Assumption), and graph $\widehat{\mathcal{G}}$ is acyclic, noting θ the concatenation of parameters $\theta_1, \ldots, \theta_d$ and $Z = \{z_{h,j}\}$ the functional gate $n_h \times d$ matrix, the candidate model $(\widehat{\mathcal{G}}, \hat{f})$ defines a multivariate distribution $q(\mathbf{x}, \widehat{\mathcal{G}}, \theta, Z)$ after the global Markov property:

$$q(\mathbf{x}, \widehat{\mathcal{G}}, \theta, Z) = \prod_{j=1}^{d} q(x_j | x_{\mathrm{Pa}(j;\widehat{\mathcal{G}})}, \theta_j, \mathbf{z}_j) \tag{5.5}$$

Moreover, as the conditional densities $q(x_j | x_{\mathrm{Pa}(j;\widehat{\mathcal{G}})}, \theta_j, \mathbf{z}_j)$ can be computed independently,

$$K(q(\mathbf{x}, \widehat{\mathcal{G}}, \theta, Z)) \overset{+}{=} \sum_{j=1}^{d} K(q(x_j | x_{\mathrm{Pa}(j;\widehat{\mathcal{G}})}, \theta_j, \mathbf{z}_j)).$$

The normalized MDL for a candidate graph $\widehat{\mathcal{G}}$ (Eq. (2.12)) thus is rewritten as a sum of d local scores:

$$MDL(\widehat{\mathcal{G}}, \theta^*, D) = \min_{\theta, Z} \left[\underbrace{\frac{1}{n} \sum_{j=1}^{d} K(q(x_j | x_{\mathrm{Pa}(j;\widehat{\mathcal{G}})}, \theta_j, \mathbf{z}_j))}_{\text{model complexity}} + \underbrace{\frac{1}{n} \sum_{j=1}^{d} \sum_{\ell=1}^{n} \log \frac{1}{q(x_j^{(\ell)} | x^{(\ell)}_{\mathrm{Pa}(j;\widehat{\mathcal{G}})}, \theta_j, \mathbf{z}_j)}}_{\text{fit loss}} \right] \tag{5.6}$$

with θ^* the optimal set of parameters for the considered model.

5.1.2 SAM learning criterion

This section derives a principled loss function from the model complexity and data fitting terms in Eq. (5.6), defining SAM learning criterion.

Model complexity While $K(q(x_j|x_{\mathrm{Pa}(j;\widehat{\mathcal{G}})}, \theta_j, \mathbf{z}_j))$ could be estimated using the Akaike Information or the Bayesian Information Criterion, the complexity of the graph structure and of the causal mechanisms can by construction be assessed and controlled through respectively the L_0 norm of the structural and functional gates $\mathbf{a}_j$ and $\mathbf{z}_j$ (that is, the number of parents of X_j and the number of effective neurons in $\hat{f}_j$):

$$K(q(x_j|x_{\mathrm{Pa}(j;\widehat{\mathcal{G}})}, \theta_j, \mathbf{z}_j)) \stackrel{\mathrm{def}}{=} \lambda_S|\mathrm{Pa}(j;\widehat{\mathcal{G}})| + \lambda_F \sum_{h=1}^{n_h} z_{h,j} \tag{5.7}$$

with $\lambda_S > 0$ and $\lambda_F > 0$ the regularization weights. For notational simplicity we write $q(x_j|x_{\mathrm{Pa}(j;\widehat{\mathcal{G}})}, \theta_j)$ instead of $q(x_j|x_{\mathrm{Pa}(j;\widehat{\mathcal{G}})}, \theta_j, \mathbf{z}_j)$ in the following.

Data fitting loss As said, when the number of samples $\mathbf{x}^{(\ell)}$ goes to infinity, the data fitting term goes to data log-likelihood expectation under the sought generative distribution:

$$\lim_{n\to\infty} \frac{1}{n} \sum_{\ell=1}^{n} \log \frac{1}{q(x_j^{(\ell)}|x^{(\ell)}_{\mathrm{Pa}(j;\widehat{\mathcal{G}})}, \theta_j)} = - \mathbb{E}_p \log q(x_j|x_{\mathrm{Pa}(j;\widehat{\mathcal{G}})}, \theta_j) \tag{5.8}$$

For $j = 1 \ldots d$, for $\mathbf{x} = (x_1, \ldots, x_d)$ let $\mathbf{x}_{-j}$ be defined as $(x_1, \ldots x_{j-1}, x_{j+1}, \ldots, x_d)$. The distribution of x_j conditionally to $\mathbf{x}_{-j}$ is denoted as $q(x_j|\mathbf{x}_{-j})$. Considering FCM $(\widehat{G}, \hat{f})$, as variable X_j only depends on $X_{\mathrm{Pa}(j;\widehat{\mathcal{G}})}$, it follows that $q(x_j|\mathbf{x}_{\mathrm{Pa}(j;\widehat{\mathcal{G}})}, \theta_j) = q(x_j|\mathbf{x}_{-j}, \theta_j)$. Therefore:

$$\lim_{n\to\infty} \frac{1}{n} \sum_{\ell=1}^{n} \log \frac{1}{q(x_j^{(\ell)}|\mathbf{x}^{(\ell)}_{\mathrm{Pa}(j;\widehat{\mathcal{G}})}, \theta_j)} = \mathbb{E}_p \log \frac{1}{q(x_j|\mathbf{x}_{-j}, \theta_j)} \tag{5.9}$$

$$= \mathbb{E}_p \log \frac{p(x_j|\mathbf{x}_{-j})}{q(x_j|\mathbf{x}_{-j}, \theta_j)} - \mathbb{E}_p \log p(x_j|\mathbf{x}_{-j}) \tag{5.10}$$

$$= D_{KL}[p(x_j|\mathbf{x}_{-j}) \,\|\, q(x_j|\mathbf{x}_{-j}, \theta_j)] + H(X_j|\mathbf{X}_{-j}), \tag{5.11}$$

with $D_{KL}[p(x_j|\mathbf{x}_{-j}) \,\|\, q(x_j|\mathbf{x}_{-j}, \theta_j)]$ the Kullback-Leibler divergence between the true conditional distribution $p(x_j|\mathbf{x}_{-j})$ and $q(x_j|\mathbf{x}_{-j}, \theta_j)$, and $H(X_j|\mathbf{X}_{-j})$ the constant, domain-dependent entropy of X_j conditionally to $\mathbf{X}_{-j}$ (neglected in the following).

Taking inspiration from Nguyen et al. (2010); Nowozin et al. (2016), $D_{KL}[p(x_j|\mathbf{x}_{-j}) \,\|\, q(x_j|\mathbf{x}_{-j}, \theta_j)]$ is estimated using an adversarial approach. Formally, for $j = 1$ to d, for each initial sample $\mathbf{x}^{(\ell)}$ let pseudo-sample $\tilde{\mathbf{x}}_j^{(\ell)}$ be defined from $\mathbf{x}^{(\ell)}$ by replacing its j-th coordinate by $\hat{f}_j(\mathbf{x}^{(\ell)}, e_j^{(\ell)})$, with $e_j^{(\ell)}$ drawn from $\mathcal{N}(0, 1)$. Let dataset $\tilde{D}_j$ denote the set of all pseudo $\tilde{\mathbf{x}}_j^{(\ell)}$ for $\ell = 1$ to n.

Let T_ω be a neural net trained to discriminate between the original dataset D and the dataset $\tilde{D} = \bigcup_{j=1}^{d} \tilde{D}_j$, with ω ranging in the parameter space Ω. Then, the scaled log-likelihood of the data in the large sample limit can be approximated after Nguyen et al. (2010):

$$\frac{1}{n}\sum_{j=1}^{d}\sum_{\ell=1}^{n}\log\frac{1}{q(x_j^{(\ell)}|x_{\mathrm{Pa}(j;\widehat{\mathcal{G}})}^{(\ell)},\theta_j)} \approx \sup_{\omega\in\Omega}\left(\frac{d}{n}\sum_{\ell=1}^{n}T_\omega(\mathbf{x}^{(\ell)})+\frac{1}{n}\sum_{j=1}^{d}\sum_{\ell=1}^{n}[-\exp(T_\omega(\tilde{\mathbf{x}_j}^{(\ell)})-1)]\right) + \mathit{constant} \quad (5.12)$$

By reintroducing this result into Eq. 5.11, this gives for each generator/discriminator set considering X_j, X_{-j}:

$$D_{KL}[p(x_j,\mathbf{x}_{-j})\,\|\,q(x_j,\mathbf{x}_{-j},\theta_j)] \approx \sup_{\omega\in\Omega}\left(\frac{1}{n}\sum_{\ell=1}^{n}[\,T_\omega^j(\mathbf{x}^{(\ell)})\,] + \frac{1}{n}\sum_{\ell=1}^{n}[\,-\exp(T_\omega^j(\tilde{\mathbf{x}_j}^{(\ell)})-1)\,]\right) \quad (5.13)$$

Note that using a single discriminator T_ω to discriminate among D and $\tilde{D}$ is more computationally efficient than building d discriminators (among D and each $\tilde{D}_j$) and yields a more stable algorithm.[1]

Evaluation of the global loss min-max penalized optimization problem with SAM Overall, SAM is trained by solving a min-max penalized optimization problem (Eqs (5.7) for the model complexity and (5.12) for the data fitting term):

$$MDL(\widehat{\mathcal{G}},\theta^*,D) = \min_{Z,A,\theta}\left(\underbrace{\frac{\lambda_S}{n}\sum_{i,j}a_{i,j}+\frac{\lambda_F}{n}\sum_{h,j}z_{h,j}}_{\text{model complexity}} + \underbrace{\max_{\omega\in\Omega}\left(\frac{d}{n}\sum_{\ell=1}^{n}T_\omega(\mathbf{x}^{(\ell)})+\frac{1}{n}\sum_{j=1}^{d}\sum_{\ell=1}^{n}[-\exp(T_\omega(\tilde{\mathbf{x}_j}^{(\ell)})-1)]\right)}_{\text{fit loss}}\right), \quad (5.14)$$

where the minimization is carried over the set of parameters $\theta = (\theta_1,\dots,\theta_d)$ of the generators and over the matrices A and Z representing the structural and functional gates.

Fig. 5.2 illustrates a 4-variable SAM: on the left are the four generators corresponding to the causal mechanisms $\hat{f}_j^{\theta_j,a_j,z_j}$, for $j = 1\dots 4$. On the right is the shared neural network discriminator T_ω evaluating the global fit loss corresponding to the sum of the estimated fit terms $D_{KL}[p(x_j,\mathbf{x}_{-j})\,\|\,q(x_j,\mathbf{x}_{-j},\theta_j)]$ for $j = 1\dots 4$.

[1] It avoids the gradient vanishing phenomena that were empirically observed when building d discriminators.

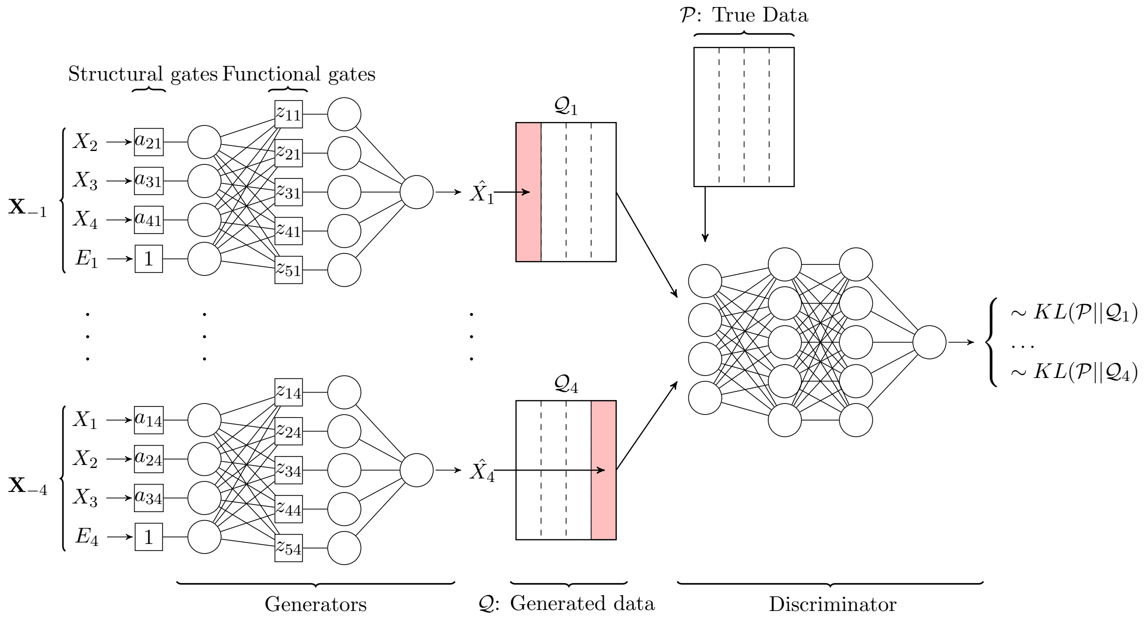


Figure 5.2: A four-variable example: Diagram of the SAM structure for variables $X_1, \ldots, X_4$

5.1.3 Enforcing the acyclicity of the causal graph

Note that Eq. (5.14) does not ensure that the optimal $\widehat{\mathcal{G}}$ be a DAG: the sparsity constraint on $\widehat{\mathcal{G}}$ through the model complexity term (minimizing $\|\mathbf{a}_j\|_0$) leads to independently identify the Markov blanket of each variable X_j, selecting all causes, effects and spouses thereof (Yu et al., 2018).

In order to ensure that the solution is a DAG and avoid the associated combinatorial optimization issues (Section 2.3), it is proposed to augment the learning criterion with an acyclicity term inspired from Zheng et al. (2018a). The use of other acyclicity characterizing criteria (Zheng et al., 2018a) is left for further work. Letting A denote the structural gate matrix (the adjacency matrix of the graph), $\widehat{\mathcal{G}}$ is a DAG iff

$$\sum_{k=1}^{d} \frac{\text{tr } A^k}{k!} = 0$$

Accordingly, the learning criterion is augmented with an acyclicity term, with:

$$\begin{aligned} MDL(\widehat{\mathcal{G}}^*, \theta^*, D) = \min_{A,Z,\theta} \max_{\omega \in \Omega} \Bigg(& \frac{1}{n} \sum_{\ell=1}^{n} \sum_{j=1}^{d} [T_\omega(\mathbf{x}^{(\ell)}) - \exp(T_\omega(\tilde{\mathbf{x}}_j^{(\ell)} - 1)] \\ & + \frac{\lambda_S}{n} \sum_{i,j} a_{i,j} + \frac{\lambda_F}{n} \sum_{j,h} z_{h,j} + \lambda_D \sum_{k=1}^{d} \frac{\text{tr } A^k}{k!} \Bigg), \end{aligned} \tag{5.15}$$

with $\lambda_D > 0$ a penalization weight.[2]

This acyclicity constraint creates a coupling among the d feature selection problems, implying that at most one arrow between pairs of variables can be selected, and more generally leading to remove effect variables from the set of parents of any X_i; the removal of effect variables in turn leads to removing spouse variables as well (section 5.2.1).

As the use of the L_0 norms of **a**s and **z**s, if naively done, could entail computational issues (retraining the network from scratch for every new graph structure or neural architecture), an approach based on the Bernoulli reparameterization trick is proposed to end-to-end train the SAM architecture and weights using stochastic gradient descent (Srivastava et al., 2014; Louizos et al., 2017) and the Binary Concrete relaxation approach (Maddison et al., 2016; Jang et al., 2016). This solution corresponds to a learned dropout of edges and hidden units of the neural network.

Overall, the optimization of the learning criterion in Eq.(5.15) with the acyclicity and sparsity constraints defines the *Structural Agnostic Model* SAM (Alg. 6, Fig. 5.2).

5.2 Theoretical Analysis of SAM

This section analyzes the MDL learning criterion, decomposed into two terms: a structural loss and a parametric loss. It is finally shown that under some mild assumptions SAM recovers the true underlying graph $\mathcal{G}$.

Using Eq. (5.7), Eq. (5.6) can be rewritten as:

$$MDL(\widehat{\mathcal{G}}, \theta^*, D) = \frac{\lambda_S}{n}|\widehat{\mathcal{G}}| + \frac{\lambda_F}{n}\sum_{j=1}^{d}\|\mathbf{z}_j\|_0 + \frac{1}{n}\sum_{j=1}^{d}\sum_{\ell=1}^{n}\log\frac{1}{q(x_j^{(\ell)}|x_{\mathrm{Pa}(j;\widehat{\mathcal{G}})}^{(\ell)}, \theta_j^*)} \tag{5.16}$$

According to (Brown et al., 2012), each scaled conditional log-likelihood term can be decomposed into three terms as:

$$\frac{1}{n}\sum_{\ell=1}^{n}\log\frac{1}{q(x_j^{(\ell)}|x_{\mathrm{Pa}(j;\widehat{\mathcal{G}})}^{(\ell)}, \theta_j^*)} = \frac{1}{n}\sum_{\ell=1}^{n}\log\frac{p(x_j^{(\ell)}|x_{\mathrm{Pa}(j;\widehat{\mathcal{G}})}^{(\ell)})}{q(x_j^{(\ell)}|x_{\mathrm{Pa}(j;\widehat{\mathcal{G}})}^{(\ell)}, \theta_j^*)} + \frac{1}{n}\sum_{\ell=1}^{n}\log\frac{p(x_j^{(\ell)}|\mathbf{x}_{-j}^{(\ell)})}{p(x_j^{(\ell)}|x_{\mathrm{Pa}(j;\widehat{\mathcal{G}})}^{(\ell)})}$$
$$+\frac{1}{n}\sum_{\ell=1}^{n}\log\frac{1}{p(x_j^{(\ell)}|\mathbf{x}_{-j}^{(\ell)})} \tag{5.17}$$

Note that term $\frac{1}{n}\sum_{\ell=1}^{n}\log p(x_j^{(\ell)}|\mathbf{x}_{-j}^{(\ell)})$ is a domain-dependent constant, converging toward $H(X_j|\mathbf{X}_{-j})$, the negative entropy of X_j conditionally to $\mathbf{X}_{-j}$ when n goes toward infinity. This term is neglected in the following.

[2] In practice, λ_D is small at the initialization and increases along time; in this way, the structural penalization term $\lambda_S \sum_{i,j} a_{i,j}$ can operate and prune the less relevant edges before considering the DAG constraint.

Algorithm 6: Structural Agnostic Modeling Algorithm

```
*bullet* Initialize $a'_{ij} = 2, z'_{ij} = 0,$ for $i, j \in [1, d]^2$
for number of iterations do
    for $j$ = 1, ..., $d$ do
        • sample the structural gate vector $\mathbf{a}_j$: for $i = 1, \dots, d$,
          $a_{i,j} = \text{cst}(H(l_{i,j} + a'_{i,j})) - \text{cst}(\text{sigmoid}(l_{i,j} + a'_{i,j})) + \text{sigmoid}(l_{i,j} + a'_{i,j})$ with $l_{i,j}$
          drawn from logistic distribution and $H$ the Heaviside step function; cst() represents
          the copy by value without gradient operator $(a_{ii} = 0, \forall i \in [1, d])$
        • sample the functional gate vector $\mathbf{z}_j$: for $h = 1, \dots, n_h$,
          $z_{h,j} = \text{cst}(H(l_{h,j} + z'_{h,j})) - \text{cst}(\text{sigmoid}(l_{h,j} + z'_{h,j})) + \text{sigmoid}(l_{h,j} + z'_{h,j})$ with $l_{h,j}$
          drawn from logistic distribution
        • sample noise variables, $e_j^{(\ell)} \sim \mathcal{N}(0, 1)$ for $\ell = 1 \dots n$, $j = 1 \dots d$
        • generate $n$ samples $\{\tilde{\mathbf{x}}_j^{(\ell)}\}_{l=1}^n$ such that for $\ell = 1 \dots, n$ :
          $$\begin{aligned} \tilde{x}_j^{(\ell)} &= \hat{f}_j^{\theta_j, a_j, z_j}(\mathbf{x}_{-j}^{(\ell)}, e_j^{(\ell)}) \\ &= \sum_{k=1}^{n_h} m_{j,k} tanh \left( \sum_{i=1}^{d} W_{j,ik} a_{ij} X_i + b_{j,k} + W_{j,d+1} E_j \right) + m_{j,0} \end{aligned}$$
    end
    • update the discriminator by ascending its stochastic gradient:
      $$\nabla_\omega \left[ \frac{d}{n} \sum_{\ell=1}^{n} T_\omega(\mathbf{x}^{(\ell)}) \; + \frac{1}{n} \sum_{j=1}^{d} \sum_{\ell=1}^{n} [ -\exp(T_\omega(\tilde{x}_j^{(\ell)}, \mathbf{x}_{-j}^{(\ell)}) - 1) ] \right]$$
    for $j$ = 1, ..., $d$ do
        • update the generator by descending its stochastic gradients w.r.t the set of
          parameters $\theta_j = (m_j, W_j, n_j, b_j, \beta_j)$, the set of parameters $\mathbf{a}'_j$ of the structural gates
          $\mathbf{a}_j$ and the set of parameters $\mathbf{z}'_j$ of the functional gates $\mathbf{z}_j$:
          $$\begin{aligned} \nabla_j =& \nabla_{\theta_j} \left[ \frac{1}{n} \sum_{\ell=1}^{n} [ -\exp(T_\omega(\tilde{x}_j^{(\ell)}, \mathbf{x}_{-j}^{(\ell)}) - 1) ] \right] \\ &+ \nabla_{\mathbf{a}'_j} \left[ \frac{1}{n} \sum_{\ell=1}^{n} [ -\exp(T_\omega(\tilde{x}_j^{(\ell)}, \mathbf{x}_{-j}^{(\ell)}) - 1) ] + \frac{\lambda_S}{n} \sum_{i,j} a_{i,j} + \lambda_D \sum_{k=1}^{d} \frac{\text{tr } A^k}{k!} \right] \\ &+ \nabla_{\mathbf{z}'_j} \left[ \frac{1}{n} \sum_{\ell=1}^{n} [ -\exp(T_\omega(\tilde{x}_j^{(\ell)}, \mathbf{x}_{-j}^{(\ell)}) - 1) ] + \frac{\lambda_F}{n} \sum_{j,h} z_{h,j} \right] \end{aligned}$$
    end
end
return $A$ and $\hat{f}_1, \dots, \hat{f}_d$
```

Let $X_{\overline{\mathrm{Pa}}(j;\widehat{\mathcal{G}})}$ denote the complementary set of X_j and its parent nodes in $\widehat{\mathcal{G}}$. Then, after Brown et al. (2012), $\frac{1}{n}\sum_{\ell=1}^{n} \log \frac{p(x_j^{(\ell)}|\mathbf{x}_{-j}^{(\ell)})}{p(x_j^{(\ell)}|x_{\mathrm{Pa}(j;\widehat{\mathcal{G}})}^{(\ell)})}$ is equal to:

$$\hat{I}^n(X_j, X_{\overline{\mathrm{Pa}}(j;\widehat{\mathcal{G}})}|X_{\mathrm{Pa}(j;\widehat{\mathcal{G}})}) = \frac{1}{n}\sum_{\ell=1}^{n} \log \frac{p(x_j^{(\ell)}, x_{\overline{\mathrm{Pa}}(j;\widehat{\mathcal{G}})}^{(\ell)}|x_{\mathrm{Pa}(j;\widehat{\mathcal{G}})}^{(\ell)})}{p(x_j^{(\ell)}|x_{\mathrm{Pa}(j;\widehat{\mathcal{G}})}^{(\ell)})p(x_{\overline{\mathrm{Pa}}(j;\widehat{\mathcal{G}})}^{(\ell)}|x_{\mathrm{Pa}(j;\widehat{\mathcal{G}})}^{(\ell)})}, \tag{5.18}$$

the estimated conditional mutual information term between X_j and $X_{\overline{\mathrm{Pa}}(j;\widehat{\mathcal{G}})}$, conditioned on the estimated parent variables $X_{\mathrm{Pa}(j;\widehat{\mathcal{G}})}$.

From Eqs (5.17) and (5.18) the global loss (Eq. (5.16)) can be decomposed into a *structural loss* $\mathcal{L}^S(\widehat{\mathcal{G}}, D)$ and a *parametric loss* $\mathcal{L}^F(\widehat{\mathcal{G}}, \theta^*, D)$:

$$MDL(\widehat{\mathcal{G}}, \theta^*, D) = \mathcal{L}^S(\widehat{\mathcal{G}}, D) + \mathcal{L}^F(\widehat{\mathcal{G}}, \theta^*, D) \tag{5.19}$$

with:

$$\begin{cases} \mathcal{L}^S(\widehat{\mathcal{G}}, D) = \sum_{j=1}^{d} \left[\hat{I}^n(X_j, X_{\overline{\mathrm{Pa}}(j;\widehat{\mathcal{G}})}|X_{\mathrm{Pa}(j;\widehat{\mathcal{G}})})\right] + \frac{\lambda_S}{n}|\widehat{\mathcal{G}}| \\ \mathcal{L}^F(\widehat{\mathcal{G}}, \theta^*, D) = \sum_{j=1}^{d} \Bigg[\frac{1}{n}\sum_{\ell=1}^{n} \log \frac{p(x_j^{(\ell)}|x_{\mathrm{Pa}(j;\widehat{\mathcal{G}})}^{(\ell)})}{q(x_j^{(\ell)}|x_{\mathrm{Pa}(j;\widehat{\mathcal{G}})}^{(\ell)}, \theta_j^*)}] + \frac{\lambda_F}{n}\|\mathbf{z}_j\|_0 \Bigg] \end{cases}$$

The structural loss $\mathcal{L}^S(\widehat{\mathcal{G}}, D)$, akin to local learning approaches (Spirtes et al., 2000; Chickering, 2002) (Section 2.3), aims to identify the Markov equivalence class of the true $\mathcal{G}$. The parametric loss $\mathcal{L}^F(\widehat{\mathcal{G}}, \theta^*, D)$ instead exploits distribution asymmetries, akin cause effect pair methods (Hoyer et al., 2009; Stegle et al., 2010).

5.2.1 Identification of the Markov equivalence class with the structural loss

Within the structural loss $\mathcal{L}^S(\widehat{\mathcal{G}}, D)$, the minimization of $\hat{I}^n(X_j, X_{\overline{\mathrm{Pa}}(j;\widehat{\mathcal{G}})}|X_{\mathrm{Pa}(j;\widehat{\mathcal{G}})})$ exploits the conditional independence relations in the candidate structure. Let us first consider the case when $\hat{I}^n(X_j, X_{\overline{\mathrm{Pa}}(j;\widehat{\mathcal{G}})}|X_{\mathrm{Pa}(j;\widehat{\mathcal{G}})})$ is minimized *independently* for each variable X_j (without considering the acyclicity term on $\widehat{\mathcal{G}}$). In the large sample limit and under classical faithfulness and Markov assumptions, Brown et al. (2012) show that the optimum is obtained for $X_{\mathrm{Pa}(j;\widehat{\mathcal{G}})} = MB(X_j)$, the Markov Blanket of X_j in $\mathcal{G}$. Note that $MB(X_j)$ usually contains spurious edges compared to the true parents $X_{\mathrm{Pa}(j;\mathcal{G})}$; it contains the effect and also the so-called spouses of X_j: if a child of X_j is retained in $X_{\mathrm{Pa}(j;\widehat{\mathcal{G}})}$, then its parents (spouses) are dependent on X_j conditionally to this child, and are retained in $X_{\mathrm{Pa}(j;\widehat{\mathcal{G}})}$.

When enforcing the acyclicity of the candidate graph on $\widehat{G}$ and minimizing the structural fitting loss $\mathcal{L}^S(\widehat{\mathcal{G}}, D)$ with a regularization term on the total number of edges, spurious edges tend to be removed and the Markov equivalence class of the true DAG (CPDAG) can be identified. The intuition is that the acyclicity constraint prevents the children nodes from being selected as parents, hence the spouse nodes do not need be selected either.

In the SAM framework, the CPDAG identification classically relies on the Causal Markov and Faithfulness assumptions (any independence constraint holds in $p(\mathbf{x})$ iff it is present in $\mathcal{G}$); it also relies on a third assumption on the estimated conditional mutual information bounds.

Theorem 5 (DAG identification up to the Markov equivalence class)**.**
Besides CMA and CFA, let us further assume that for any fixed number of samples n:
a) for any pair of variables X_i, X_j and any disjoint subset of variables $V \subset \mathbf{X}$, such that $I(X_j, X_i|X_V) = 0$, one has $\hat{I}^n(X_j, X_i|X_V) < \frac{\lambda_S}{n}$.
b) for any pair of variables X_i, X_j and any disjoint subset of variables $V \subset \mathbf{X}$, such that $I(X_j, X_i|X_V) \neq 0$, one has $\hat{I}^n(X_j, X_i|X_V) > \frac{\lambda_S}{n}$.

Then in the limit of large n:
i) For every $\widehat{\mathcal{G}}$ in the equivalence class of $\mathcal{G}$, $\mathcal{L}^S(\widehat{\mathcal{G}}, D) = \mathcal{L}^S(\mathcal{G}, D)$.
ii) For every $\widehat{\mathcal{G}}$ not in the equivalence class of $\mathcal{G}$, $\mathcal{L}^S(\widehat{\mathcal{G}}, D) > \mathcal{L}^S(\mathcal{G}, D)$.

Proof. in Appendix 5.5.1 □

5.2.2 Identification within Markov equivalence class of DAGs with the parametric loss

The parametric loss $\mathcal{L}^F(\widehat{\mathcal{G}}, \theta^*, D)$ aims to retrieve the true causal model within its Markov equivalence class. Each term

$$\frac{1}{n}\sum_{\ell=1}^{n} \log \frac{p(x_j^{(\ell)}|x^{(\ell)}_{\text{Pa}(j;\widehat{\mathcal{G}})})}{q(x_j^{(\ell)}|x^{(\ell)}_{\text{Pa}(j;\widehat{\mathcal{G}})}, \theta_j^*)}$$

measures the ability of $\hat{f}_j$ to fit the conditional distribution of X_j based on its parents $X_{\text{Pa}}(j;\widehat{\mathcal{G}})$. In the large sample limit, this term converges towards $\mathbb{E}_p\left[\log \frac{p(x_j|x_{\text{Pa}(j;\widehat{\mathcal{G}})})}{q(x_j|x_{\text{Pa}(j;\widehat{\mathcal{G}})}, \theta_j^*)}\right]$.

Note that when considering sufficiently powerful causal mechanisms, this term goes to 0 in the large sample limit even if $\widehat{\mathcal{G}} \neq \mathcal{G}$: as shown by Hyvärinen and Pajunen (1999), it is always possible to find a function $\hat{f}_j$ such that $X_j = \hat{f}_j(X_{\text{Pa}(j;\widehat{\mathcal{G}})}, E_j)$, with $E_j \perp\!\!\!\perp X_{\text{Pa}(j;\widehat{\mathcal{G}})}$, corresponding to a probabilistic conditional model q such that $q(x_j|x_{\text{Pa}(j;\widehat{\mathcal{G}})}, \theta_j^*) = p(x_j|x_{\text{Pa}(j;\widehat{\mathcal{G}})})$ (hence $\mathbb{E}_p\left[\log \frac{p(x_j|x_{\text{Pa}(j;\widehat{\mathcal{G}})})}{q(x_j|x_{\text{Pa}(j;\widehat{\mathcal{G}})}, \theta_j^*)}\right] = 0$).

When restricting the *capacity* of the causal mechanism space however, this parametric fitting term may support model identification within the Markov equivalence class of the DAG. Following (Stegle et al., 2010)' pioneering work, SAM uses a soft constraint (a regularization term) to restrict the capacity of the considered mechanism, specifically the number of active neurons involved in $\hat{f}_j$:

$$\mathcal{L}^F(\widehat{\mathcal{G}}, \theta^*, D) = \frac{1}{n} \sum_j \sum_{\ell=1}^{n} \log \frac{p(x_j^{(\ell)} | x_{\mathrm{Pa}(j;\widehat{\mathcal{G}})}^{(\ell)})}{q(x_j^{(\ell)} | x_{\mathrm{Pa}(j;\widehat{\mathcal{G}})}^{(\ell)}, \theta_j^*)} + \lambda_z \|\mathbf{z}_j\|_0$$

Theorem 6. *For every DAG $\widehat{\mathcal{G}} \neq \mathcal{G}$ in the Markov equivalence class of $\mathcal{G}$, given the Working Hypothesis 1 and the causal Markov and faithfulness assumptions:*

$$\sum_{j=1}^{d} K(p(x_j | x_{Pa(j;\mathcal{G})})) \overset{+}{\leq} \sum_{j=1}^{d} K(p(x_j | x_{Pa(j;\widehat{\mathcal{G}})})), \tag{5.20}$$

Proof. in Appendix 5.5.2 □

Following (Janzing and Scholkopf, 2010; Marx and Vreeken, 2017) and approximating the Kolmogorov complexity with the Minimum Description Length (section 2.2.3), for every DAG $\widehat{\mathcal{G}} \neq \mathcal{G}$ in the Markov equivalence class of $\mathcal{G}$:

$$MDL(\mathcal{G}, \theta^*, D) \leq MDL(\widehat{\mathcal{G}}, \theta^*, D) \tag{5.21}$$

According to equation (5.19):

$$\mathcal{L}^S(\mathcal{G}, D) + \mathcal{L}^F(\mathcal{G}, \theta^*, D) \leq \mathcal{L}^S(\widehat{\mathcal{G}}, D) + \mathcal{L}^F(\widehat{\mathcal{G}}, \theta^*, D) \tag{5.22}$$

Under the conditions given in Theorem 1, for DAGs in the Markov equivalence class of $\mathcal{G}$ in the large sample limit the structural score $\mathcal{L}^S(\widehat{\mathcal{G}}, D)$ is minimal and equal to $\mathcal{L}^S(\mathcal{G}, D)$. It yields:

$$\mathcal{L}^F(\mathcal{G}, \theta^*, D) \leq \mathcal{L}^F(\widehat{\mathcal{G}}, \theta^*, D) \tag{5.23}$$

Within the Markov equivalence class, the parametric loss can disambiguate the different structures and support the identification of the true $\mathcal{G}$. An illustration is presented in Appendix 5.5.2.

5.3 Experimental setting

The goal of the validation is to experimentally answer two questions. The first one concerns SAM's performance compared to the state of the art, depending on whether the underlying joint distribution complies with usual assumptions (Gaussian distributions for the variables and the noise, linear causal mechanisms). The second question concerns the merits and drawbacks of SAM's strategy of learning non-linear causal mechanisms, and relying on adversarial learning.

This section first describes the SAM configurations and hyper-parameter settings used in the experiments, followed by the detail of the synthetic,[3] realistic and real-world datasets involved in the experiments. The baseline algorithms and their hyper-parameter settings, and the performance indicators are last described.

For convenience and reproducibility, all considered algorithms have been integrated in the publicly available CausalDiscovery Toolbox,[4] including the most recent baseline versions at the time of the experiments.

5.3.1 SAM configurations

Each causal mechanism $\hat{f}_j$ is sought as a 1-hidden layer NN with $n_h^g = 200$ neurons, using tanh activation. Note that this activation function enables to represent linear mechanisms when deemed appropriate. The discriminator is a 2-hidden layer NN with $n_h^D = 200$ LeakyReLU units on each layer and batch normalization (Ioffe and Szegedy, 2015). Structural gates $a_{i,j}$ and functional gates $z_{h,j}$ are initialized to 0 with probability 1/2, except for the self-loop terms $a_{i,i}$ set to 0. SAM is trained for $n_{iter} = 10,000$ epochs using Adam (Kingma and Ba, 2014) with initial learning rate 0.01. SAM hyper-parameters are calibrated using 10 synthetic datasets (five of 20 variables and five of 100 variables) of type VI (section 5.3.2). In all experiments, $\lambda_S = 5$, $\lambda_F = 0.005$, and

$$\lambda_D = \begin{cases} 0 & \text{if } t < 5,000 \\ 1 & \text{otherwise} \end{cases}$$

with t the number of epochs: the first half of the run does not take into account the acyclicity constraint and focuses on the identification of the Markov blankets for each variable; the acyclicity constraint intervenes in the second half of the run.

Four variants have been considered: the full SAM (Alg. 6) and three lesioned variants designed to examine the benefits of non-linear mechanisms and adversarial training.

Specifically, **SAM-lin** desactivates the non-linear option and only implements linear causal mechanisms (with no functional gates), replacing Eq (5.4) with:

$$\hat{X}_j = \sum_{i=1}^{d} W_{j,i} a_{j,i} X_i + W_{j,d+1} E_j + W_{j,0} \tag{5.24}$$

A second variant, **SAM-mse**, replaces the adversarial loss with a standard mean-square error loss, replacing the f-gan term in Eq. (5.13) with $\frac{1}{n} \sum_{j=1}^{d} \sum_{\ell=1}^{n} (x_j^{(\ell)} - \tilde{x}_j^{(\ell)})^2$.

A third variant, **SAM-lin-mse**, involves both linear mechanisms and mean square error losses.

[3] The codes for generating the synthetic datasets are available at https://github.com/Diviyan-Kalainathan.

[4] https://github.com/diviyan-kalainathan/causaldiscoverytoolbox.

5.3.2 Benchmarks

The synthetic datasets include 10 DAGs with 20 variables and 10 DAGs with 100 variables.

1. The DAG structure is such that the number of parents for each variable is uniformly drawn in $\{0, \dots, 5\}$;

2. For the i-th DAG, the mean μ_i and variance σ_i of the noise variables are drawn as $\mu_i \sim \mathbb{U}(-2, 2)$ and $\sigma_i \sim \mathbb{U}(0, 0.4)$ and the distribution of the noise variables is set to $\mathcal{N}(\mu_i, \sigma_i)$;

3. For each graph, a 500 sample-dataset is iid generated following the topological order of the graph, with for $\ell = 1$ to 500:

$$x^{(\ell)} = (x_1^{(\ell)}, \dots, x_d^{(\ell)}), \;\; x_i^{(\ell)} \sim f_i(X_{\mathrm{Pa}(i)}, E_i), \text{ with } E_i \sim \mathcal{N}(\mu_i, \sigma_i)$$

All variables are normalized to zero-mean and unit-variance.

Six categories of causal mechanisms have been considered: besides those considered for the experimental validation of the CAM algorithm (Peters et al., 2014), a more complex one is considered, leveraging the non-linearity of neural nets:

I *Linear*: $X_i = \sum_{j \in \mathrm{Pa}(i)} a_{i,j} X_j + E_i$, where $a_{i,j} \sim \mathcal{N}(0, 1)$

II *Sigmoid AM*: $X_i = \sum_{j \in \mathrm{Pa}(i)} f_{i,j}(X_j) + E_i$, where $f_{i,j}(x_j) = a \cdot \frac{b \cdot (x_j + c)}{1 + |b \cdot (x_j + c)|}$ with $a \sim \mathrm{Exp}(4) + 1$, $b \sim \mathcal{U}([-2, -0.5] \cup [0.5, 2])$ and $c \sim \mathcal{U}([-2, 2])$.

III *Sigmoid Mix*: $X_i = f_i(\sum_{j \in \mathrm{Pa}(i)} X_j + E_i)$, where f_i is as in the previous bullet-point.

IV *GP AM*: $X_i = \sum_{j \in \mathrm{Pa}(i)} f_{i,j}(X_j) + E_i$ where $f_{i,j}$ is an univariate Gaussian process with a Gaussian kernel of unit bandwidth.

V *GP Mix*: $X_i = f_i([X_{\mathrm{Pa}(i)}, E_i])$, where f_i is a multivariate Gaussian process with a Gaussian kernel of unit bandwidth.

VI *NN*: $X_i = f_i(X_{\mathrm{Pa}(i)}, E_i)$, with f_i a 1-hidden layer neural network with 20 *tanh* units, with all neural weights sampled from $\mathcal{N}(0, 1)$.

5.3.3 Baseline algorithms

The following algorithms have been used, with their default parameters: the score-based methods GES (Chickering, 2002) and GIES (Hauser and Bühlmann, 2012) with Gaussian scores; the hybrid method MMHC (Tsamardinos et al., 2006), the L_1 penalized method for causal discovery CCDr (Aragam and Zhou, 2015), the LiNGAM algorithm (Shimizu et al., 2006) and the causal additive model CAM (Peters et al., 2014). Lastly, the PC algorithm (Spirtes et al., 2000) has been considered with four conditional independence tests in the Gaussian and non-parametric settings:

- PC-Gauss: using a Gaussian conditional independence test on z-scores;
- PC-HSIC: using the HSIC independence test (Zhang et al., 2012) with a Gamma null distribution (Gretton et al., 2005b);
- PC-RCIT: using the Randomized Conditional Independence Test (RCIT) with random Fourier features (Strobl et al., 2017);
- PC-RCOT: the Randomized conditional Correlation Test (RCOT) (Strobl et al., 2017).

PC,[5] GES and LINGAM versions are those of the *pcalg* package (Kalisch et al., 2012). MMHC is implemented with the *bnlearn* package (Scutari, 2009). CCDr is implemented with the *sparsebn* package (Aragam et al., 2017).

The GENIE3 algorithm (Irrthum et al., 2010) is also considered, though it does not focus on DAG discovery *per se* as it achieves feature selection, retains the Markov Blanket of each variable using random forest algorithms. Nevertheless, this method won the DREAM4 In Silico Multifactorial challenge (Marbach et al., 2009), and is therefore included in the baseline algorithms (using the *GENIE3* R package).

5.3.4 Performance indicators

For the sake of robustness, 16 independent runs have been launched for each dataset-algorithm pair. The average causation score $c_{i,j}$ for each edge $X_i \to X_j$ is measured as the fraction of runs where this edge belongs to $\widehat{\mathcal{G}}$. When an edge is left undirected, e.g with PC algorithm, it is counted as appearing with both orientations with weight $1/2$. A t-test is used to assess whether the score difference between any two methods is statistically significant with a p-value below 0.001.

Precision-recall A true positive is an edge $i \to j$ of the true DAG $\mathcal{G}$ which is correctly recovered by the algorithm; T_p is the number of true positive. A false negative is an edge of $\mathcal{G}$ which is missing in $\widehat{\mathcal{G}}$; F_n is the number of false negatives. A false positive is an edge in $\widehat{\mathcal{G}}$ which is not in $\mathcal{G}$ (reversed edges and edges which are not in the skeleton of $\mathcal{G}$); F_p is the number of false positives. The precision-recall curve, showing the tradeoff between precision ($T_p/(T_p + F_p)$) and recall ($T_p/(T_p + F_n)$) for different causation thresholds (Fig. 5.6), is summarized by the **Area under the Precision Recall Curve (AuPR)**, ranging in [0,1], with 1 being the best.[6]

Structural Hamming Distance Another performance indicator used in the causal graph discovery framework is the Structural Hamming Distance (SHD) (Tsamardinos et al., 2006), set to the number of missing edges and redundant edges in the found structure. This SHD

[5] The better-performing, order-independent version of the PC algorithm proposed by Colombo and Maathuis (2014) is used.

[6] Using the *scikit-learn v0.20.1* library (Pedregosa et al., 2011).

score is computed in the following by considering all edges $i \to j$ with $c_{i,j} > .5$. Note that a reversal error (retaining $j \to i$ while $\mathcal{G}$ includes edge $i \to j$) is counted as a single mistake.

$$\mathrm{SHD}(\hat{A}, A) = \sum_{i,j} |\hat{A}_{i,j} - A_{i,j}| - \frac{1}{2} \sum_{i,j} (1 - max(1, \hat{A}_{i,j} + A_{j,i})), \tag{5.25}$$

with A (respectively $\hat{A}$) the adjacency matrix of $\mathcal{G}$ (resp. the found causal graph $\widehat{\mathcal{G}}$).

Structural Intervention Distance refers to an adaptation of the SHD metric for causal graphs (Peters and Bühlmann, 2013). It counts the number of wrong causal paths between connected variables. The lower the value, the better: having a value of 0 corresponds to correct causal relationships between each pairs of connected variables in the true graph.

5.4 Experiments

This section first reports on the experimental results obtained on synthetic datasets. Realistic biological data coming from the SynTREN simulator (Van den Bulcke et al., 2006) on 20- and 100-node graphs, and from GeneNetWeaver (Schaffter et al., 2011) on the DREAM4 challenge are thereafter considered (section 5.4.2), and we last consider the extensively studied flow cytometry dataset (Sachs et al., 2005) (section 5.4.3).

The detail of all results is given in Appendix 5.5.3, reporting the average performance indicators, standard deviation, and computational cost of all considered algorithms.

5.4.1 Synthetic datasets

20 variable-graphs The comparative results (Fig. 5.3) demonstrate SAM's robustness in term of Area under the Precision Recall Curve (AUPR) on all categories of 20-node graphs. Specifically, SAM is dominated by GES and GIES on linear mechanisms and by CAM for Gaussian univariate mechanisms, reminding that GES and GIES (resp. CAM) specifically aim at linear mechanisms (resp. Gaussian univariate mechanisms). Note that, while the whole ranking of the algorithms may depend on the considered performance indicator, the best performing algorithm is most often the same regardless of whether the AUPR or the Structural Hamming distance is considered. For non-linear cases with complex interactions (the Sigmoid Mix and NN cases), SAM significantly outperforms other non-parametric methods such as PC-HSIC, PC-RCOT and PC-RCIT. In the linear Gaussian setting, SAM aims to the Markov equivalence class of the true graph (under causal Markov and faithfulness assumptions) and performs less well than for e.g. the GP mix where SAM can exploit both conditional independence relations and distribution asymmetries. Though seemingly counter-intuitive, a graph with more complex interactions between noise and variables may be actually easier to recover than a graph generated with simple mechanisms (see also Wang and Blei (2018)).

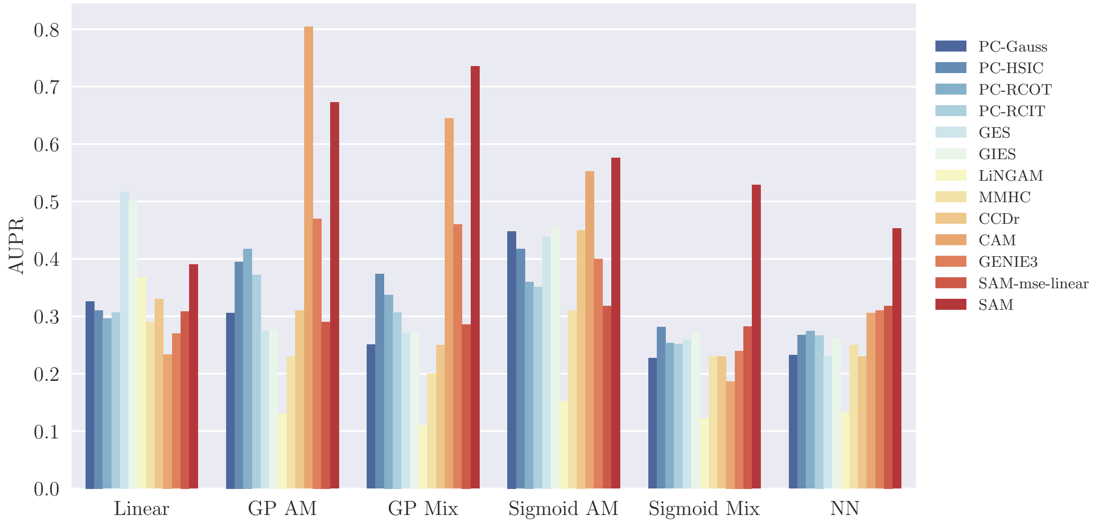


Figure 5.3: Performance of causal graph discovery methods on 20-node synthetic graphs measured by the Area under the Precision Recall Curve (the higher, the better). SAM ranks among the top-three methods, being only dominated by GES and GIES for linear mechanisms and by CAM for univariate mechanisms (better seen in color).

SAM's computational cost is one order of magnitude higher than that of the other methods (all measured on a single CPU core Intel Xeon 2.7Ghz).[7] The lesioned versions, SAM-lin, SAM-mse and SAM-line-mse have significantly worse performances than SAM (except for the linear mechanism and additive Gaussian noise cases), demonstrating the merits of the NN-based and adversarial learning approach in the general case.

100-variable graphs The comparative results on the 100-node graphs (Fig. 5.4) confirm the good overall robustness of SAM. As could have been expected, SAM is dominated by CAM on the GP AM, GP Mix and Sigmoid AM; indeed, focusing on the proper causal mechanism space yields a significant advantage, all the more so as the number of variables increases. Nevertheless, SAM does never face a catastrophic failure, and it even performs quite well on linear datasets. A tentative explanation is based on the fact that the *tanh* activation function enables to capture linear mechanisms; another explanation is based on the adversarial loss, empirically more robust than the MSE loss in high-dimensional problems.

In terms of computational cost, SAM scales well at $d = 100$ variables, particularly when compared to its best competitor CAM, that uses a combinatorial graph search. The PC-HSIC algorithm had to be stopped after 50 hours; more generally, constraint-based methods based

[7] A speed up factor of 25 can be obtained for SAM using a GPU environment with single graphic card GeForce GTX 1080Ti, particularly beneficial for the GAN training.

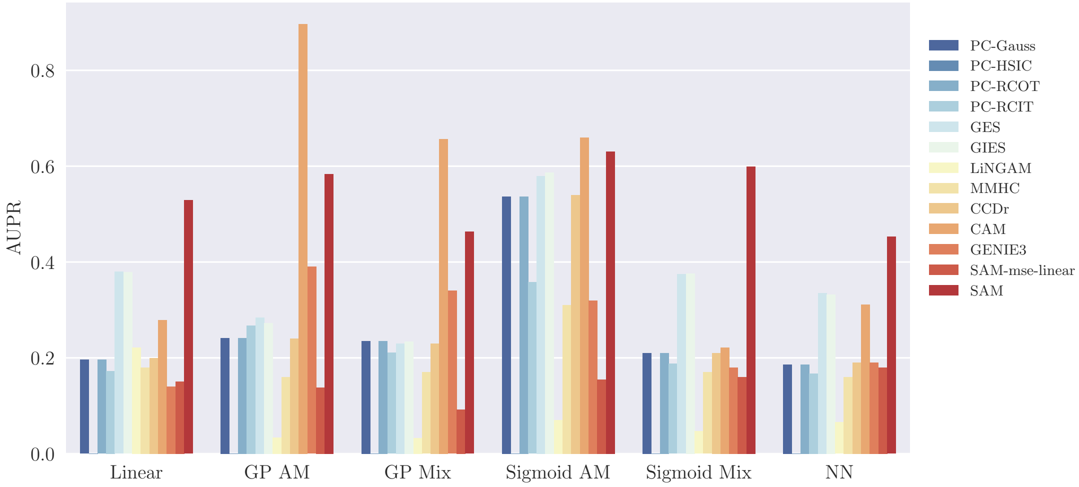


Figure 5.4: Performance of causal graph discovery methods on 100-node synthetic graphs measured by the Area under the Precision Recall Curve (the higher, the better). On datasets relying on Gaussian processes, CAM tops the leaderboard by a significant margin as its search space matches the sought causal mechanisms. SAM demonstrates its robustness with respect to the underlying generative models (better seen in color).

on the PC algorithm do not scale well w.r.t. the number of variables.

5.4.2 Simulated biological datasets

As said, the SynTREN (Van den Bulcke et al., 2006) and GeneNetWeaver (GNW) (Schaffter et al., 2011) simulators of genetic regulatory networks have been used to generate observational data reflecting realistic complex regulatory mechanisms, high-order conditional dependencies between expression patterns and potential feedback cycles, based on an available causal model.

SynTREN simulator Sub-networks of E. coli (Shen-Orr et al., 2002) have been considered, where interaction kinetics are based on Michaelis-Menten and Hill kinetics (Mendes et al., 2003). Overall, ten 10-nodes and ten 100-nodes graphs have been considered.[8] For each graph, 500-sample datasets are generated by SynTREN.

Likewise, the comparative results on all SynTREN graphs (Fig. 5.5) demonstrate the good performances of SAM. Overall, the best performing methods take into account both distribu-

[8] Random seeds set to 1...10 are used for the sake of reproducibility. SynTREN hyper-parameters include a probability of 1.0 (resp. 0.1) for complex 2-regulator interactions (resp. for biological noise, experimental noise and noise on correlated inputs).

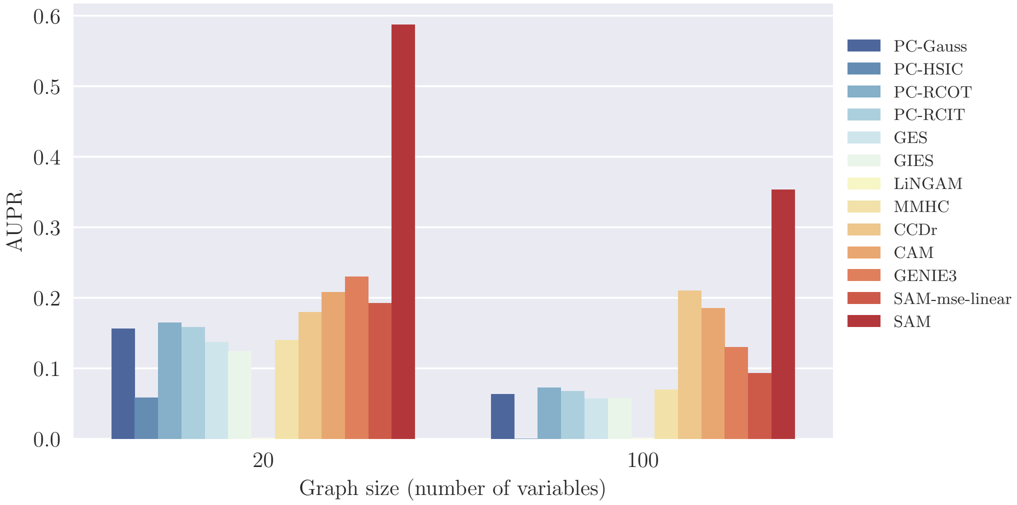


Figure 5.5: Performance of causal graph discovery methods on SynTREN graphs measured by the Area under the Precision Recall Curve (the higher, the better). Left: 20 nodes. Right: 100 nodes (better seen in color).

.

tion asymmetry and multivariate interactions. Constraint-based methods are hampered by the lack of v-structures, preventing the orientation of many edges to be based on CI tests only (PC-HSIC algorithm was stopped after 50 hours and LiNGAM did not converge on one of the datasets). The benefits of using non-linear mechanisms on such problems are evidenced by the difference between SAM-lin-mse and SAM-mse (Appendix 5.5.3). The Precision-Recall curve is displayed on Fig. 5.6 for representative 20-node and 100-node graphs, confirming that SAM can be used to infer networks having complex distributions, complex causal mechanisms and interactions.

GeneNetWeaver simulator - DREAM4 Five 100-nodes graphs generated using the GeneNetWeaver simulator define the *In Silico Size 100 Multifactorial* challenge track of the *Dialogue for Reverse Engineering Assessments and Methods* (DREAM) initiative. These graphs are subnetworks of transcriptional regulatory networks of E. coli and S. cerevisiae and their dynamics are simulated using a kinetic gene regulation model, where noise is added both in the dynamics of the networks and on the measurement of expression data. Multifactorial perturbations are simulated by slightly increasing or decreasing the basal activation of all genes of the network simultaneously by different random amounts. In total, the number of expression conditions for each network is set to 100.

The comparative results on these five graphs (Fig. 5.7) show that GENIE3 outperforms all other methods, with SAM ranking second. A tentative explanation for GENIE3 excellent

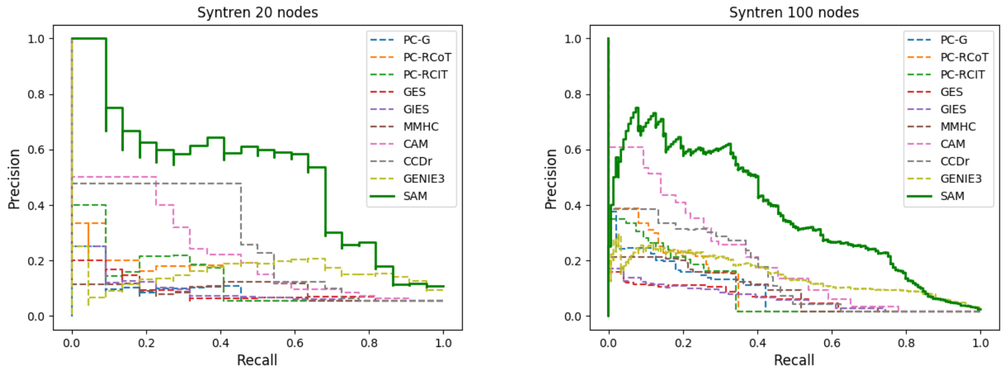


Figure 5.6: Precision/Recall curve for two SynTREN graphs: Left, 20 nodes; Right, 100 nodes (better seen in color).

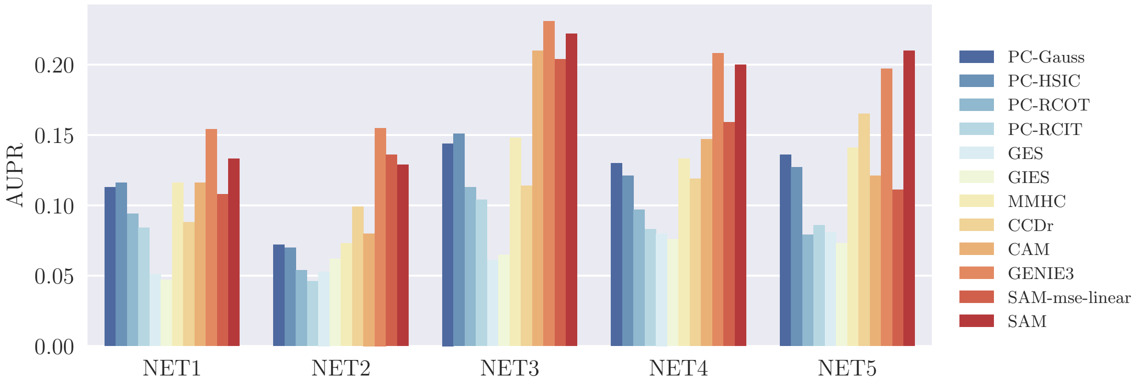


Figure 5.7: Performance of causal graph discovery methods on 5 artificial datasets of the Dream4 In Silico Multifactorial Challenge measured by the Area under the Precision Recall Curve (the higher, the better). GENIE3 achieves the best performance on 4 datasets, with SAM close second (better seen in color).

performance is that it does not enforce the discovery of acyclic graphs, which is appropriate as regulatory networks involve feedback loops. The Precision/Recall curves (Fig. 5.8) demonstrate that SAM matches GENIE3 performances in the low recall region. Overall, on such complex problem domains, it appears relevant to make few assumptions on the underlying generative model (like GENIE3 and SAM), while being able to capture high-order conditional dependencies between variables. As said, LiNGAM did not converge on one of these datasets.

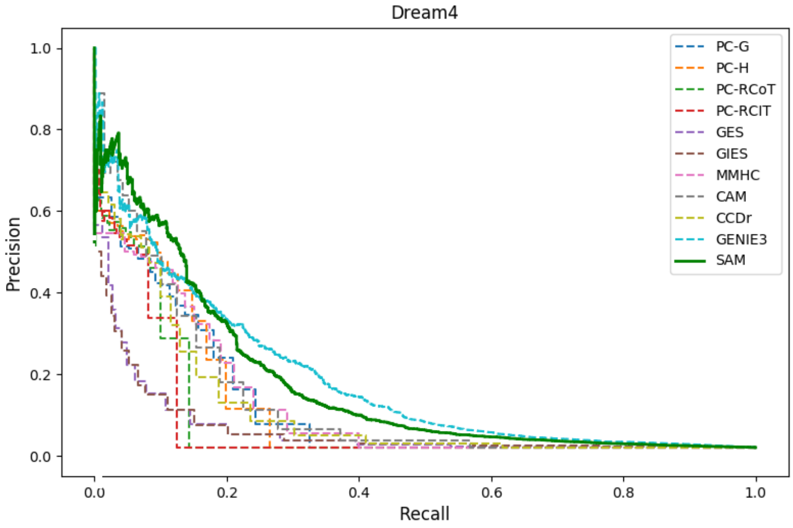


Figure 5.8: Precision/Recall curve for the Dream4 *In Silico Multifactorial Challenge* (better seen in color).

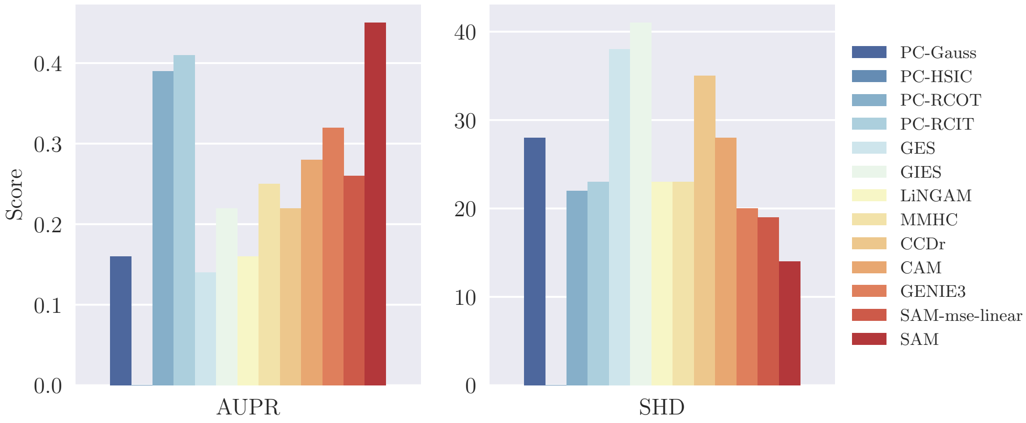


Figure 5.9: Performance of causal graph discovery methods on the protein network problem (Sachs et al., 2005). Left, Area under the Precision Recall curve (the higher the better); Right, Structural Hamming distance (the lower, the better). SAM significantly outperforms all other methods on this dataset (better seen in color).

5.4.3 Real-world biological data

The well-studied protein network problem (Sachs et al., 2005) is associated with observational data including 7,466 observational samples. Same experimental setting is used as for the other problem, with a bootstrap ratio of 0.8. According to both performance indicators (Fig. 5.9), SAM significantly outperforms the other methods. The precision/recall curve (Fig. 5.10) shows that SAM is particularly accurate when its confidence score is high, showing that for critical applications where false negatives are to be avoided, using SAM with a threshold is a viable option. Notably, SAM recovers the transduction pathway *raf*→*mek*→*erk* corresponding to direct enzyme-substrate causal effect (Sachs et al., 2005).

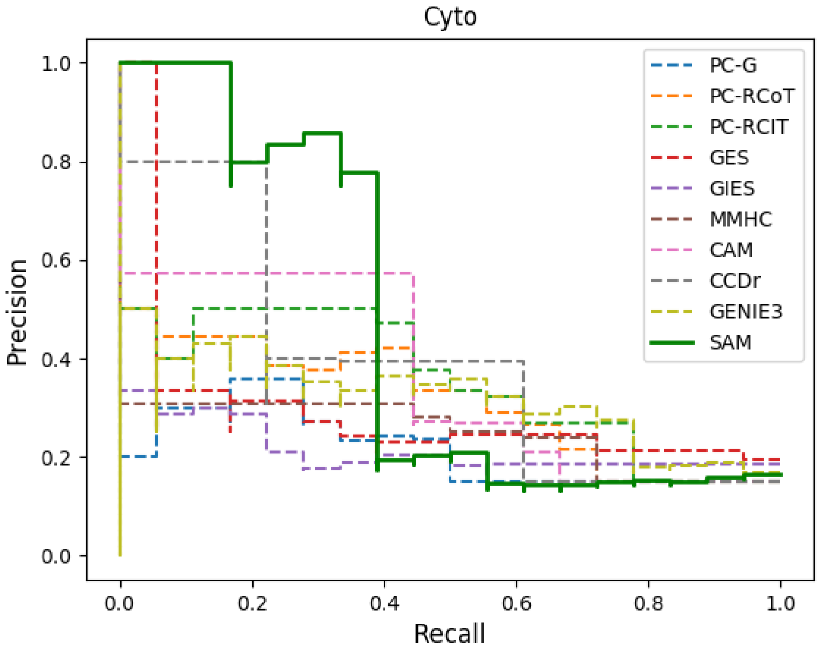


Figure 5.10: Precision/Recall curve for the curve protein network (better seen in color).

Partial conclusion

The main contribution of the paper is a unifying causal discovery framework, exploiting both structural independence and distributional asymmetries through optimizing well-founded structural and functional criteria. This framework is implemented in the SAM algorithm[9], leveraging the non-parametric power of Generative Adversarial Neural networks (GANs) to capture a faithful generative model and enforce the discovery of acyclic causal graphs through sparsity and algebraic regularizations, using stochastic gradient descent.

Extensive empirical evidence is gathered to show SAM robustness across diverse synthetic, realistic and real-world problems. Lesion studies are conducted to assess whether and when it is beneficial to learn non-linear mechanisms and to rely on adversarial learning as opposed

[9] Available at https://github.com/Diviyan-Kalainathan/SAM.

to MSE minimization.

As could have been expected, in particular settings SAM is dominated by algorithms specifically designed for this setting, such as CAM (Bühlmann et al., 2014) in the case of additive noise model and Gaussian process mechanisms, and GENIE3 when facing causal graphs with feedback loops. Nevertheless, SAM most often ranks first and always avoids catastrophic failures. The main limitation of SAM is its computational cost, higher by an order of magnitude than other approaches on 20-variable problems. On 100-variable problems however, SAM catches up with the other approaches as it avoids the combinatorial exploration of the graph space.

5.5 Appendix

5.5.1 Structural loss: Proof of Theorem 5

Theorem 5. *DAG identification up to the Markov equivalence class*
It is assumed, besides causal Markov ad faithfulness assumptions, that there exists some integer n_0 such that for any $n > n_0$,
a) for any pair of variables X_i, X_j and any disjoint set of variables $V \subset \mathbf{X}$ such that

$$I(X_j, X_i | X_V) = 0$$

its empirical estimate satisfies:

$$\hat{I}^n(X_j, X_i | X_V) < \frac{\lambda_S}{n}$$

b) likewise, for any pair of variables X_i, X_j and any disjoint set of variables $V \subset \mathbf{X}$,

$$I(X_j, X_i | X_V) > 0 \Rightarrow \hat{I}^n(X_j, X_i | X_V) > \frac{\lambda_S}{n}$$

Then as n goes to ∞, the minimum of the structural loss (Eq. (5.19)) is reached on the equivalence class of $\mathcal{G}$:
i) For every $\widehat{\mathcal{G}}$ in the equivalence class of $\mathcal{G}$, $\mathcal{L}^S(\widehat{\mathcal{G}}, D) = \mathcal{L}^S(\mathcal{G}, D)$.
ii) For every $\widehat{\mathcal{G}}$ not in the equivalence class of $\mathcal{G}$, $\mathcal{L}^S(\widehat{\mathcal{G}}, D) > \mathcal{L}^S(\mathcal{G}, D)$.

Proof. Let $\widehat{\mathcal{G}}$ be a DAG, and let $\widehat{\mathcal{G}}'$ be defined from $\widehat{\mathcal{G}}$ by adding a single edge $X_k \to X_j$ such that $\widehat{\mathcal{G}}'$ is still a DAG. Let us compare the structural losses of $\widehat{\mathcal{G}}$ and $\widehat{\mathcal{G}}'$:

$$\begin{aligned}\Delta\mathcal{L}^S &= \mathcal{L}^S(\widehat{\mathcal{G}}', D) - \mathcal{L}^S(\widehat{\mathcal{G}}, D)\\ &= \hat{I}^n(X_{\overline{\mathrm{Pa}}(j;\widehat{\mathcal{G}}')}, X_j | X_{\mathrm{Pa}(j;\widehat{\mathcal{G}}')}) - \hat{I}^n(X_{\overline{\mathrm{Pa}}(j;\widehat{\mathcal{G}})}, X_j | X_{\mathrm{Pa}(j;\widehat{\mathcal{G}})}) + \frac{\lambda_S}{n}\end{aligned}$$

From

$$\hat{I}^n(X_j, X_{-j}) = \hat{I}^n(X_j, X_{\mathrm{Pa}(j;\widehat{\mathcal{G}}')}) + \hat{I}^n(X_{\overline{\mathrm{Pa}}(j;\widehat{\mathcal{G}}')}, X_j | X_{\mathrm{Pa}(j;\widehat{\mathcal{G}}')}) \tag{5.26}$$

and

$$\hat{I}^n(X_j, X_{-j}) = \hat{I}^n(X_j, X_{\mathrm{Pa}(j;\widehat{\mathcal{G}})}) + \hat{I}^n(X_{\overline{\mathrm{Pa}}(j;\widehat{\mathcal{G}})}, X_j|X_{\mathrm{Pa}(j;\widehat{\mathcal{G}})}) \tag{5.27}$$

it follows:

$$\begin{aligned}\Delta\mathcal{L}_S &= -\hat{I}^n(X_j, X_{\mathrm{Pa}(j;\widehat{\mathcal{G}}')}) + \hat{I}^n(X_j, X_{\mathrm{Pa}(j;\widehat{\mathcal{G}})}) + \frac{\lambda_S}{n} \\ &= -\hat{I}^n(X_j, X_{\mathrm{Pa}(j;\widehat{\mathcal{G}})} \cup X_k) + \hat{I}^n(X_j, X_{\mathrm{Pa}(j;\widehat{\mathcal{G}})}) + \frac{\lambda_S}{n} \\ &= -\hat{I}^n(X_j, X_k|X_{\mathrm{Pa}(j;\widehat{\mathcal{G}})}) + \frac{\lambda_S}{n}\end{aligned}$$

- If $X_j \perp\!\!\!\perp X_k|X_{\mathrm{Pa}(j;\widehat{\mathcal{G}})}$, then $I(X_j, X_k|X_{\mathrm{Pa}(j;\widehat{\mathcal{G}})}) = 0$ and according to assumption a), for $n > n_0$ $\hat{I}^n(X_j, X_k|X_{\mathrm{Pa}(j;\widehat{\mathcal{G}})}) < \frac{\lambda_S}{n}$ and $\Delta\mathcal{L}^S > 0$.
  In other words, for $n > n_0$ the loss increases when adding any irrelevant edge.

- If $X_j \not\perp\!\!\!\perp X_k|X_{\mathrm{Pa}(j;\widehat{\mathcal{G}})}$, then $I(X_j, X_k|X_{\mathrm{Pa}(j;\widehat{\mathcal{G}})}) \neq 0$. It follows from assumption b) that $\hat{I}^n(X_j, X_k|X_{\mathrm{Pa}(j;\widehat{\mathcal{G}})}) > \frac{\lambda_S}{n}$ and therefore $\Delta\mathcal{L}^S < 0$.
  Likewise, the loss decreases for large n when adding any edge that removes an irrelevant conditional independence.

Both results establish the consistency of the structural loss $\mathcal{L}^S$ (Chickering (2002), Prop 8). □

5.5.2 Parametric loss : Proof of Theorem 6

Theorem 6. *For every DAG $\widehat{\mathcal{G}} \neq \mathcal{G}$ in the Markov equivalence class of $\mathcal{G}$, under the causal Markov and faithfulness assumptions and Working Hypothesis 1:*

$$\sum_{j=1}^{d} K(p(x_j|x_{Pa(j;\mathcal{G})})) \overset{+}{\leq} \sum_{j=1}^{d} K(p(x_j|x_{Pa(j;\widehat{\mathcal{G}})})), \tag{5.28}$$

Proof. Under the Working Hypothesis, the shortest description of p is given by the sum of descriptions of the conditional probability distributions :

$$K(p(\mathbf{x})) \overset{+}{=} \sum_{j=1}^{d} K(p(x_j|x_{\mathrm{Pa}(j;\mathcal{G})})), \tag{5.29}$$

where the equality holds if conditionals $p(x_j|x_{\mathrm{Pa}(j;\mathcal{G})})$ are algorithmically independent (Janzing and Scholkopf, 2010).

Thus for any DAG $\widehat{\mathcal{G}}$ in the Markov equivalence class of the $\mathcal{G}$ s.t. $\widehat{\mathcal{G}} \neq \mathcal{G}$:

$$p(\mathbf{x}) = \prod_{j=1}^{d} p(x_j|x_{\mathrm{Pa}(j;\widehat{\mathcal{G}})}) \tag{5.30}$$

After Lemeire and Steenhaut (2010), the sum of the description of the conditionals $p(x_j|x_{\mathrm{Pa}(j;\widehat{\mathcal{G}})})$ is always greater than the description of their product, therefore,

$$K(p(\mathbf{x})) \overset{+}{\leq} \sum_{j=1}^{d} K(p(x_j|x_{\mathrm{Pa}(j;\widehat{\mathcal{G}})})) \tag{5.31}$$

Thus:

$$\sum_{j=1}^{d} K(p(x_j|x_{\mathrm{Pa}(j;\mathcal{G})})) \overset{+}{\leq} \sum_{j=1}^{d} K(p(x_j|x_{\mathrm{Pa}(j;\widehat{\mathcal{G}})})), \tag{5.32}$$

which concludes the proof.

□

5.5.3 Details of SAM experiments results

This appendix reports the detail of the experimental results summarized in section 5.4. Computational time are measured on a 48-core Intel(R) Xeon(R) CPU E5-2650 CPU (between parentheses, on a Nvidia GTX 1080 GPU).

20-variable artificial graphs Tables 5.1 and 5.2 show the robustness of SAM w.r.t. diverse types of mechanisms. In terms of average precision (Table 5.1) SAM is respectively dominated by GES on linear (resp. CAM on GP AM) mechanisms, which is explained as GES (resp. CAM) is specifically designed to identify linear (resp. Gaussian) causal mechanisms. In terms of Average Structural Hamming distance (Table 5.2), SAM is likewise dominated by algorithms specifically tailored to the considered type of mechanisms (LiNGAM, CAM and CCDr), while yielding globally good performances. SAM main weakness is its computational cost (second higher cost over all considered algorithms).

100-variable artificial graphs Tables 5.3 and 5.4 show the scalability of SAM w.r.t. the number of variables. In terms of average precision (Table 5.3), SAM is only dominated by CAM on the GP AM, GP Mix and Sigmoid AM causal mechanisms (noting that CAM is tailored to Gaussian Processes). Most interestingly, its computational time favorably compares to that of CAM on 100-variable problems. Note that PC-HSIC had to be stopped after 50 hours.

Realistic problems (SynTReN, GENIE3, and Cyto) Tables 5.5 and 5.6 show the robustness of SAM on realistic problems generated with the SynTReN simulator (20 graphs of 20 nodes and 100 nodes) and on the so-called Sachs problem (Sachs et al., 2005) (Cyto) in terms of average precision (the higher the better) and structural Hamming distance (the lower the better). SAM yields significantly better results in all cases except on the SynTReN 100 nodes, where it is dominated by GENIE3 in terms of structural Hamming distance.

Table 5.1: Artificial graphs with 20 variables: Average Precision (std. dev.) of all compared algorithms over all six types of distributions (the higher the better). Significantly better results (t-test with p-value 0.001) are underlined. The computational time is per graph.

AP	Linear	GP AM	GP Mix	Sigmoid AM	Sigmoid Mix	NN	Global	Time in s.
PC-Gauss	0.33 (0.06)	0.31 (0.07)	0.25 (0.10)	0.45 (0.10)	0.23 (0.06)	0.23 (0.05)	0.30 (0.07)	1
PC-HSIC	0.31 (0.07)	0.40 (0.06)	0.37 (0.07)	0.42 (0.08)	0.28 (0.08)	0.27 (0.03)	0.34 (0.06)	46 523
PC-RCOT	0.30 (0.04)	0.42 (0.06)	0.34 (0.06)	0.36 (0.05)	0.25 (0.05)	0.27 (0.03)	0.32 (0.05)	356
PC-RCIT	0.31(0.05)	0.37 (0.04)	0.31 (0.05)	0.35 (0.05)	0.25 (0.06)	0.27 (0.04)	0.30 (0.04)	181
GES	**0.52** (0.07)	0.27 (0.06)	0.27 (0.07)	0.44 (0.14)	0.26 (0.11)	0.23 (0.07)	0.33 (0.10)	1
GIES	0.50 (0.09)	0.28 (0.07)	0.27 (0.10)	0.46 (0.14)	0.27 (0.10)	0.26 (0.09)	0.34 (0.12)	1
MMHC	0.29 (0.05)	0.23 (0.03)	0.20 (0.04)	0.31 (0.03)	0.23 (0.03)	0.25 (0.03)	0.25 (0.03)	1
LiNGAM	0.37 (0.05)	0.13 (0.03)	0.11 (0.02)	0.15 (0.06)	0.12 (0.02)	0.13 (0.03)	0.17 (0.03)	2
CAM	0.23 (0.07)	**0.80** (0.07)	0.64 (0.12)	0.55 (0.11)	0.19 (0.04)	0.31 (0.10)	0.45 (0.08)	2 880
CCDr	0.33 (0.06)	0.31 (0.07)	0.25 (0.09)	0.45 (0.10)	0.23 (0.06)	0.23 (0.05)	0.30 (0.07)	2
GENIE3	0.27 (0.05)	0.47 (0.04)	0.46 (0.08)	0.40 (0.05)	0.24 (0.02)	0.31 (0.04)	0.36 (0.05)	54
SAM-lin-mse	0.31 (0.04)	0.29 (0.05)	0.29 (0.05)	0.32 (0.06)	0.28 (0.04)	0.32 (0.08)	0.30 (0.07)	332 (70)
SAM-mse	0.29 (0.04)	0.43 (0.04)	0.46 (0.10)	0.40 (0.08)	0.26 (0.05)	0.33 (0.07)	0.36 (0.05)	2 984 (91)
SAM-lin	0.49 (0.10)	0.28 (0.04)	0.29 (0.03)	0.41 (0.09)	0.35 (0.08)	0.34 (0.08)	0.30 (0.07)	14 812 (645)
SAM	0.39 (0.08)	0.67 (0.08)	**0.74** (0.12)	**0.58** (0.13)	**0.53** (0.06)	**0.45** (0.09)	**0.56** (0.09)	17 388 (676)

Table 5.2: Artificial graphs with 20 variables: Average Structural Hamming Distance (std. dev.) of all compared algorithms over all six types of distributions (the lower the better). Significantly better results (t-test with p-value 0.001) are underlined.

SHD	Linear	GP AM	GP Mix	Sigmoid AM	Sigmoid Mix	NN
PC-Gauss	42.80 (6.74)	46.65 (4.68)	45.60 (5.45)	38.95 (9.93)	52.15 (6.46)	48.35 (7.37)
PC-HSIC	43.15 (5.04)	42.85 (7.05)	40.65 (5.16)	41.05 (9.23)	47.35 (9.32)	44.85 (6.83)
PC-RCOT	42.40 (4.42)	40.65 (6.16)	40.40 (6.38)	42.90 (8.52)	46.35 (7.49)	43.30 (6.68)
PC-RCIT	42.35 (5.09)	44.05 (5.85)	41.00 (6.24)	42.70 (9.41)	46.45 (6.37)	42.80 (7.05)
GES	43.05 (18.5)	72.20 (9.60)	57.45 (8.21)	46.55 (15.9)	75.60 (16.8)	78.05 (17.5)
GIES	42.70 (17.7)	70.45 (8.64)	57.65 (10.1)	47.55 (15.0)	57.65 (10.1)	75.25 (15.0)
MMHC	45.5 (5.25)	62.3 (4.67)	64.0 (6.85)	54.80 (9.59)	56.3 (7.16)	50.30 (7.36)
LiNGAM	**36.50** (4.99)	46.70 (5.23)	43.20 (6.80)	45.80 (8.72)	52.10 (5.82)	54.80 (10.2)
CAM	71.15 (6.47)	**26.80** (6.68)	42.65 (10.2)	50.90 (9.63)	75.45 (11.5)	70.50 (10.1)
CCDr	42.80 (6.40)	46.65 (4.44)	45.60 (5.17)	**38.90** (9.42)	52.15 (6.12)	48.35 (6.99)
GENIE3	40.3 (6.96)	43.7 (5.81)	38.9 (7.14)	44.5 (8.41)	42.4 (5.80)	40.9 (7.23)
SAM-lin-mse	43,00 (7.29)	47.56 (6.70)	41.56 (6.31)	48.22 (9.61)	45.44 (5.56)	42.89 (7.68)
SAM-mse	46.78 (6.03)	41.00 (5.42)	36.11 (3.93)	44.33 (11.6)	49.56 (4.69)	44.89 (7.43)
SAM-lin	39.00 (6.46)	54.33 (6.29)	46.11 (4.25)	45.33 (8.86)	47.11 (6.37)	44.56 (8.69)
SAM	45.40 (5.32)	31.90 (8.53)	**25.20** (4.54)	40.10 (11.7)	**39.00** (4.40)	**40.80** (6.05)

The Dream4 In Silico Multifactorial Challenge. Tables 5.7 and 5.8 show the robustness of SAM on 5 artificial graphs of the Dream4 In Silico Multifactorial Challenge, respectively in terms of average precision and structural Hamming distance. GENIE3 achieves the best performance overall, and SAM is second.

Table 5.3: Artificial graphs with 100 variables: Average Precision (std. dev.) of all compared algorithms over all six types of distributions (the higher the better). Significantly better results (t-test with p-value 0.001) are underlined. The computational time is per graph.

AP	Linear	GP AM	GP Mix	Sigmoid AM	Sigmoid Mix	NN	Global	Time in s.
PC-Gauss	0.20 (0.03)	0.24 (0.03)	0.23 (0.03)	0.54 (0.04)	0.21 (0.04)	0.19 (0.03)	0.27 (0.03)	13
PC-HSIC	-	-	-	-	-	-	-	-
PC-RCOT	0.20 (0.03)	0.24 (0.03)	0.23 (0.02)	0.54 (0.04)	0.21 (0.04)	0.19 (0.03)	0.27 (0.03)	31 320
PC-RCIT	0.17 (0.03)	0.27 (0.03)	0.21 (0.02)	0.36 (0.03)	0.19 (0.02)	0.17 (0.01)	0.23 (0.02)	46 440
GES	0.38 (0.08)	0.28 (0.05)	0.23 (0.02)	0.58 (0.06)	0.37 (0.06)	0.34 (0.06)	0.36 (0.05)	1
GIES	0.38 (0.08)	0.27 (0.05)	0.23 (0.03)	0.59 (0.04)	0.38 (0.07)	0.33 (0.06)	0.36 (0.05)	5
MMHC	0.18 (0.02)	0.16 (0.01)	0.17 (0.01)	0.31 (0.02)	0.17 (0.02)	0.16 (0.01)	0.19 (0.01)	5
LiNGAM	0.22 (0.05)	0.03 (0.01)	0.03 (0.01)	0.07 (0.02)	0.05 (0.01)	0.07 (0.01)	0.08 (0.02)	5
CAM	0.28 (0.05)	**0.90** (0.03)	**0.66** (0.03)	**0.66** (0.03)	0.22 (0.03)	0.31 (0.04)	0.50 (0.03)	45 899
CCDr	0.20 (0.03)	0.24 (0.03)	0.23 (0.02)	0.54 (0.04)	0.21 (0.04)	0.19 (0.03)	0.27 (0.03)	3
GENIE3	0.14 (0.02)	0.39 (0.02)	0.34 (0.02)	0.32 (0.02)	0.18 (0.02)	0.19 (0.01)	0.26 (0.02)	511
SAM-lin-mse	0.15 (0.02)	0.14 (0.01)	0.09 (0.01)	0.16 (0.03)	0.16 (0.02)	0.18 (0.02)	0.15 (0.02)	3 076 (74)
SAM-mse	0.15 (0.02)	0.25 (0.02)	0.11 (0.02)	0.18 (0.02)	0.18 (0.02)	0.19 (0.01)	0.18 (0.02)	18 180 (118)
SAM-lin	0.51 (0.09)	0.29 (0.04)	0.18 (0.01)	0.51 (0.04)	0.50 (0.04)	0.44 (0.07)	0.41 (0.02)	24 844 (1 980)
SAM	**0.53** (0.08)	0.58 (0.04)	0.46 (0.05)	0.63 (0.04)	**0.60** (0.07)	**0.45** (0.09)	**0.54** (0.06)	24 844 (2 041)

Table 5.4: Artificial graphs with 100 variables: Average Structural Hamming Distance (std. dev.) of all compared algorithms over all six types of distributions (the lower the better). Significantly better results (t-test with p-value 0.001) are underlined.

SHD	Linear	GP AM	GP Mix	Sigmoid AM	Sigmoid Mix	NN
PC-Gauss	262.65 (19.87)	255.35 (12.99)	250.00 (10.85)	170.55 (12.05)	258.30 (16.49)	260.80 (15.79)
PC-HSIC	-	-	-	-	-	-
PC-RCOT	262.65 (19.87)	255.35 (12.99)	250.00 (10.85)	170.55 (12.05)	258.30 (16.49)	260.80 (15.79)
PC-RCIT	253.05 (18.87)	246.30 (17.58)	246.95 (9.950)	208.75 (16.11)	244.80 (17.30)	246.05 (10.00)
GES	292.10 (38.00)	412.40 (31.04)	326.15 (17.91)	206.30 (21.39)	365.85 (32.54)	391.95 (43.10)
GIES	288.40 (34.29)	417.00 (30.76)	322.10 (18.24)	202.95 (15.75)	371.45 (29.28)	385.75 (42.37)
MMHC	275.12 (13.54)	372.41 (18.6)	345.15 (15.2)	296.51 (15.3)	315.01 (12.7)	284.93 (14.05)
LiNGAM	230.00 (12.11)	251.00 (21.76)	252.00 (10.85)	241.10 (16.78)	251.44 (17.42)	250.60 (15.69)
CAM	309.25 (26.91)	**94.60** (11.20)	**170.70** (11.99)	159.85 (12.39)	354.25 (18.32)	333.20 (28.84)
CCDr	262.65 (19.87)	255.35 (12.99)	250.00 (10.85)	170.55 (12.05)	258.30 (16.49)	260.80 (15.79)
GENIE3	240.2 (17.62)	252.4 (18.33)	247.0 (10.66)	238.5 (19.46)	238.3 (16.66)	237.3 (13.16)
SAM-lin-mse	238.56 (16.84)	256.78 (12.02)	247.89 (10.28)	239.67 (19.10)	238.44 (16.65)	234.11 (8.45)
SAM-mse	269.89 (20.82)	238.89 (13.08)	249.67 (11.01)	238.33 (17.57)	256.89 (19.83)	243.67 (11.02)
SAM-lin	193.89 (24.94)	251.89 (13.05)	265.67 (11.41)	196.78 (12.53)	195.67 (14.26)	199.78 (20.71)
SAM	**182.30** (26.38)	186.10 (13.05)	211.60 (19.22)	**158.00** (17.74)	**167.60** (17.40)	**186.89** (18.96)

Table 5.5: Realistic problems: Average precision (std dev.) over 20 graphs (the higher the better). Left: 20 nodes. Middle: 100 nodes. Right: real protein network.. Significantly better results (t-test with p-value 0.001) are underlined.

AP	SynTREN 20 nodes	SynTREN 100 nodes	Cyto
PC-Gauss	0.16 (0.06)	0.06 (0.01)	0.16
PC-HSIC	0.06 (0.01)	-	-
PC-RCOT	0.16 (0.05)	0.07 (0.02)	0.39
PC-RCIT	0.16 (0.05)	0.07 (0.01)	0.41
GES	0.14 (0.06)	0.06 (0.01)	0.14
GIES	0.12 (0.04)	0.06 (0.01)	0.22
MMHC	0.14 (0.05)	0.07 (0.01)	0.25
LiNGAM	-	-	0.16
CAM	0.21 (0.08)	0.19 (0.04)	0.28
CCDr	0.18 (0.12)	0.21 (0.05)	0.22
GENIE3	0.23 (0.07)	0.13 (0.02)	0.32
SAM-lin-mse	0.19 (0.08)	0.09 (0.02)	0.26
SAM-mse	0.40 (0.14)	0.17 (0.02)	0.28
SAM-lin	0.24 (0.23)	0.13 (0.03)	0.23
SAM	**0.59** (0.15)	**0.35** (0.06)	**0.45**

Table 5.6: Realistic problems: Structural Hamming distance (std. dev.) over 20 graphs (the higher the better). Left: 20 nodes. Middle: 100 nodes. Right: real protein network.. Significantly better results (t-test with p-value 0.001) are underlined.

SHD	SynTREN 20 nodes	SynTREN 100 nodes	Cyto
PC-Gauss	53.42 (6.13)	262.65 (19.87)	28
PC-HSIC	24.13 (4.08)	-	-
PC-RCOT	34.21 (7.99)	213.51 (8.60)	22
PC-RCIT	33.20 (7.54)	204.95 (8.77)	23
GES	67.26 (12.26)	436.02 (18.99)	38
GIES	69.31 (12.55)	430.55 (22.80)	41
MMHC	67.2 (8.42)	346 (14.44)	38
LiNGAM	-	-	23
CAM	57.85 (9.10)	222.9 (12.38)	28
CCDr	54.97 (16.68)	228.8 (21.15)	35
GENIE3	23.6 (4.14)	153.2 (4.59)	20
SAM-lin-mse	25.44 (4.97)	240.1 (3.92)	19
SAM-mse	25.67 (6.96)	173.78 (6.36)	22
SAM-lin	30.45 (8.09)	168.89 (5.63)	20
SAM	**19.02** (5.83)	**160.21** (13.03)	**14**

Table 5.7: Precision on 5 artificial graphs of the Dream4 In Silico Multifactorial Challenge (the higher, the better). The best results are in bold..

AP	NET1	NET2	NET3	NET4	NET5
PC-Gauss	0.113	0.072	0.144	0.130	0.136
PC-HSIC	0.116	0.070	0.151	0.121	0.127
PC-RCOT	0.094	0.054	0.113	0.097	0.079
PC-RCIT	0.084	0.046	0.104	0.083	0.086
GES	0.051	0.053	0.061	0.080	0.081
GIES	0.047	0.062	0.065	0.076	0.073
MMHC	0.116	0.073	0.148	0.133	0.141
LiNGAM	-	-	-	-	-
CAM	0.116	0.080	0.210	0.147	0.121
CCDr	0.088	0.099	0.114	0.119	0.165
GENIE3	**0.154**	**0.155**	**0.231**	**0.208**	0.197
SAM-lin-mse	0.108	0.136	0.204	0.159	0.111
SAM-mse	0.095	0.066	0.188	0.145	0.136
SAM-lin	0.080	0.077	0.190	0.170	0.134
SAM	0.133	0.129	0.222	0.200	**0.210**

Table 5.8: Structural Hamming distance on 5 artificial graphs of the Dream4 In Silico Multifactorial Challenge (the lower, the better). The best results are in bold.

AP	NET1	NET2	NET3	NET4	NET
PC-Gauss	183	261	200	223	203
PC-HSIC	170	249	193	210	192
PC-RCOT	174	248	193	211	191
PC-RCIT	172	248	193	211	191
GES	252	333	279	286	266
GIES	261	314	281	304	274
MMHC	188	263	206	223	203
LiNGAM	-	-	-	-	-
CAM	178	250	182	213	196
CCDr	187	248	209	227	189
GENIE3	**172**	**245**	**190**	208	193
SAM-lin-mse	176	249	195	211	193
SAM-mse	171	253	197	211	192
SAM-lin	175	249	190	**204**	191
SAM	176	251	191	209	**192**

Chapter 6

Conclusions

Causal discovery represents a crucial domain of machine learning to understand models and to be able to predict effects of interventions, of modifications in a given system, which is needed for decision makers and practitioners. Traditionnally, causal discovery in the graph setting is performed leveraging structural information in the data, namely conditional dependencies and independencies (Spirtes et al., 2000; Chickering, 2002). In the pairwise setting, as the structural information is limited, methods leverage distributional asymmetries to indentify causality (Hoyer et al., 2009; Zhang and Hyvärinen, 2010). Our approach extends the works of Stegle et al. (2010); Janzing et al. (2012); Bühlmann et al. (2014), using to use neural networks to leverage both structural and distributional information.

6.1 Discussion

The main contribution of this thesis is threefold: i) a principled framework for causal discovery rooted on information theory is presented together with a theoretical analysis, establishing the optimality of the proposed approach under mild assumptions; ii) two algorithms, implementing the proposed approach have been presented; both non linear and leveraging the representative power of neural networks; iii) a throughout experimental validation of these algorithms, along with open-source tools used to evaluate algorithms.

The first algorithm, **Causal Generative Neural Networks (CGNN)** (Goudet et al., 2018), assuming the existence of an causal graph skeleton shows how to find the optimal solution, expressed as a constructive description of the underlying data distribution: A hill-climbing search in performed in the 2^L possibilities of graphs in the graph skeleton such as score based-methods, L representing the number of edges. The originality is to propose an evaluation score that can be optimized within the neural network, based on the Maximum Mean Discrepancy (MMD) distance (Gretton et al., 2007) between the original distribution and the generated one. This score inherits a good property of MMD: the unique optimum in the large sample limit. The value of this metric at convergence of the neural network also represents the score of the candidate graph. Once learned, the CGNN model can be used to generate data and simulate interventions in the causal graph.

Limitations The main limitation of CGNN resides in its computational complexity: score is quadratic in the number of samples because of the MMD metric, and requires to retrain a neural network for each new candidate graph. Such complex score is not well suited to score-based methods, that heuristically explore the graph space and test numerous graph candidates. Considering the cost of evaluating a candidate graph, only the greediest graph search heuristics are applicable to CGNN, thus preventing an efficient exploration of the graph space to provide optimal solutions. The proposed CGNN algorithm does also possess a significant amount of hyperparameters, such as the learning rate of the neural network, or the number of epochs of training. However, the parameter controlling the number of hidden units of the generators is the most crucial parameter: it controls the class of admissible functions as causal mechanisms. Set too high, all kinds of functions are accepted, which allows the generators to generate variables from noise variables. On the opposite, the predictions can also be biased if the this parameter is set too low, as the neural network saturate. In CGNN, this parameter corresponds directly to the number of hidden units in the neural network. While setting these hyperparameters, an equilibrium must be found. In practice, the use of artificial datasets allows to set properly their values by performing a grid-search.

Addressing these limitations, the **Structural Agnostic Model (SAM)** (Kalainathan et al., 2019) represents the second algorithmic contribution of this thesis. In contrast, SAM is a global learning method, leveraging gradient descent to learn its graph; the candidate graph graph is broken down into a set of generators, one for each variable.

A generator's task is to find the parents of the target variable.

Since the computational graph is differenciable up to the set of parents of each variable, the variable selection is learned through automatic differentiation.

The metric for variable generation is a adversarial neural network (Goodfellow et al., 2014), comparing between the true observational data and the data generated by the set of networks. This GAN setting leads to for a linear computational cost with respect to data size, compared to CGNN, which is quadratic because of the MMD. SAM allows us to infer a causal graph in a single learning phase of the neural network, thus reducing drastically the computational cost compared to CGNN. The optimization of the structure to obtain a causal graph, made of binary values translating the presence or absence of edges in the graph, is made directly using the Gumbel softmax trick (Jang et al., 2016; Maddison et al., 2016), allowing backpropagation through discrete variables (Section 3.3.5). This allows us to formulate the super-exponential graph-search as a optimization problem that can be resolved through backpropagation, allowing SAM to be run on hundreds of variables. In SAM, the issue of fixing the number of hidden units of the generators is remediated with a soft constraint: the number of hidden units is learned using the Gumbel Softmax binary optimization, similar to the one used for learning the structure. This yields a smoother regulation of the number of hidden units, thus making the structure less sensitive to this hyperparameter.

Limitations The consequence of using a adversarial neural network in the SAM architecture is the relative instability of training. Although a mode collapse of the GAN was never noticed during all experiments, there is no indication of the progress of the training and the quality of the current model.

At the cost of computational complexity, a more stable alternative would be to replace the discriminator with a MMD (Gretton et al., 2007) metric.

Applications of SAM and CGNN

The algorithms proposed in this thesis are already being used in real-world applications in various domains such as social sciences or bio-informatics. A first application is based of two datasets on quality of life at work (QWL) collected by the French Ministry of work (Kalainathan et al. (2018), in french). The datasets are composed of a 500 feature survey on quality of life at work of 30000 workers and financial results of hundreds of thousands of French companies. The goal of the study is to evaluate the impact of QWL on the productivity of a company. An initial study of the data was made on the survey dataset, published in (Kalainathan et al. (2018), in french); it consisted of analyzing the various types of profiles identified in the dataset with a principal component analysis and a K-means clustering. Afterwards, a causal discovery application has been done on the financial data of the French companies, as it is all numerical data, well suited for neural network based algorithms. After selecting groups of companies from a same activity sector, the causal discovery experiments highlighted some interesting causal links, such as the access to training programs for employees tending to reduce accidents at work and occupational diseases.

A second application of SAM is on the detection of genetic regulation effect (Bothorel et al. (2019), in french): the study focuses on the interactions of genes and the underlying mechanisms, to provide a better understanding of problems in health, agronomy and much more. The objective of this application is to represent the various interactions of a given cellular system to understand the various regulation mechanisms and their related illnesses through the deregulation of some phenomena due to external/environmental factors. This study complements Zaag et al. (2014), based on the clustering and analysis of the 26374 genes of interest of the plant *Arabidopsis thaliana*; causal discovery is applied only on a single cluster of interest. The use of SAM allowed to highlight some possible causal links between genes, to be validated by performing real-world experiments while reporting consistency with already known causal links. Further work consist in extending SAM to apply it on the whole database of 26374 genes.

Finally, a third application of SAM and CGNN focuses on the generation of data: as the model is a generative neural network, it also can be used to generate data after training. The main objective of the MediChal (Yale et al., 2018) project is to provide means to generate realistic data given private data, while keeping the privacy of the data. The study concerns the MIMIC dataset (Johnson et al., 2016) which contains private medical data, which is unpractical for scientific studies and teaching purposes. Therefore, SAM provides a causal graph on data along with open-source generated data. The causal graph allows the practitioners to have insight on the generation of the data and on its structure and virtually perform experiments through the generators.

Finally, SAM and CGNN did impact the machine learning research community, which produced either extensions (Yu et al., 2019; Lachapelle et al., 2019) or models for time-series (Nauta, 2018) or for feature selection (Doquet and Sebag, 2019) by leveraging SAM and CGNN's unique structures.

Limitations of the proposed approaches

The proposed SAM and CGNN algorithms both possess a significant amount of hyperparameters, such as the learning rate of the neural network, or the number of epochs of training. While setting these hyperparameters, an equilibrium must be found. In practice, the use of artificial datasets allows to set properly their values by performing a grid-search.

CGNN's main limitation is its inability to scale up to hundreds of variables due to its scoring function. SAM tackles this issue by leveraging gradient descent and fully using the parallel computation in GPUs. A downside is its memory consumption as SAM stores for each variable a neural network; during the forward phase of SAM, the dataset has to be fed into each generator thus making the memory consumption grow quadratically with the number of variables. This refrains from using SAM on datasets of thousands of variables on modern GPUs.

Generative neural networks are known to perform well with numerical data, but not with mixed data. CGNN and SAM do possess the same flaws: they are not supporting categorical data directly, as the evaluation has to be adapted depending on which kind of data is given: the marginal distribution of a categorical value is composed of spikes at each category, which can be considered as highly complex if it is considered as a numerical value. This limitation is preventing CGNN and SAM from being used on mixed-type datasets, which represent the majority of datasets for causal discovery.

Finally, the causal sufficiency assumption is made in SAM, such as in many other causal discovery methods. Making this assumption is rather tricky when dealing with real data, where the presence of hidden confounding variables is not known, which might ultimately lead to wrong conclusions. Indeed, hidden confounding variables introduce dependencies between variables in the data without any direct causal relationship. CGNN does provide an extension for dealing with hidden confounders: by introducing shared noise between each pair of variables, it allows for explaining the dependencies with the presence of a common noise variable.

Unlike other approaches, methods involving neural networks are exposed to instability: the stochasticity of the learning procedure or the initialization of the neural networks affect their final performance and predictions. This phenomenon is also present in SAM and CGNN, making the predictions vary in independent runs while considering the same data and parameters. This instability is alleviated by computing independently multiple runs on the same setting, and then averaging the results. The averaging procedure is usually done with 16 runs, as the experiments highlighted that running more does not improve predictive performance.

6.2 Research perspectives

In this section, we will discuss the improvements and extensions on the Structural Agnostic Model (SAM) algorithm, as it represents a more computationally efficient and better performing alternative to CGNN. The potential developments of SAM will allow it to extend its usability in the most general cases, thus freeing the practitioner from checking all the assumptions on the data before performing causal discovery.

6.2.1 Reducing the computational cost

SAM can scale up to few hundreds of variables with the modern GPU architectures, which is enough for many applications; in order for it to accept more variables efficiently, revising the architecture of the neural network is needed. The main memory consumption of SAM comes from the generative neural networks in parallel for each variable. A way to refrain from initializing one generator for each variable is to merge the generators in an auto-encoder fashion, while adding rules to prevent a variable to generate itself (Figure 6.1).

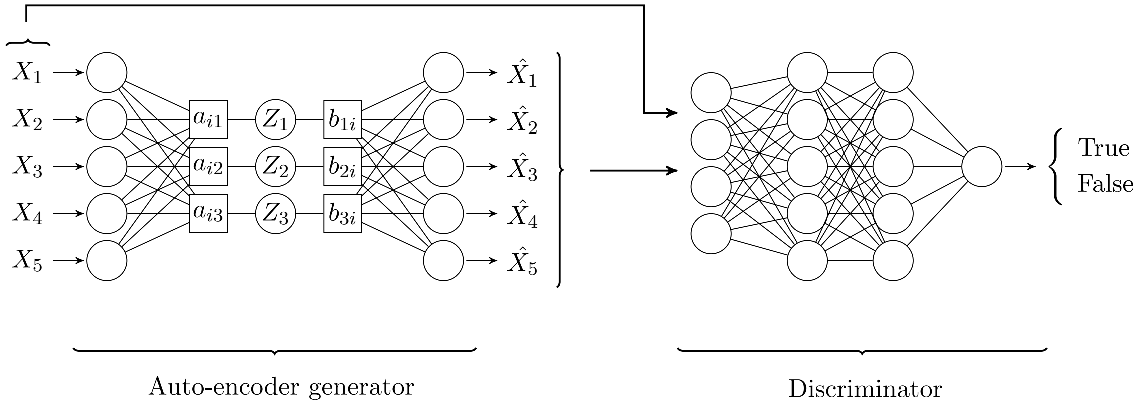


Figure 6.1: Proposed architecture of the auto-encoder SAM on an example of 5 variables, the binary coefficients a_{ik} and b_{ik} define the causal graph, and possess a constraint such that a variable cannot generate itself

Let $\{Z_i\}_{i=1}^{K}$ be sets of k neurons of the only hidden layer of the structure, and $\mathbf{A} = a_{ik}, \mathbf{B} = b_{ik}$ represent binary structural gates, as in SAM. a_{ik} represents the connection of the inputs with the sets of hidden units, and b_{ki} represents the connection of the sets to the output. In order to refrain a variable from generating itself, a set taking a variable X_i as input cannot be used to generate X_i, which corresponds to : $a_{ik} * b_{ki} = 0$. On a global scale, this constraint becomes:

$$\mathbf{A} \circ \mathbf{B}^t = \mathbf{0} \tag{6.1}$$

with $\circ$ corresponding to the Hadamard product. Finally, the graph structure would be obtained by matrix product between the two gate matrices[1]:

$$\mathcal{A} = \mathbf{A} \cdot \mathbf{B} \tag{6.2}$$

where $\mathcal{A}$ corresponds to the adjacency matrix of the graph. This new formulation of SAM allows to compress its structure and have more efficient generators that share data and weights.

6.2.2 Confounding variables

As mentioned previously, providing a solution to the hidden confounding variables issue is of significant importance when SAM is to be used in real-world applications. However, unlike linear methods where analytic resolutions of the linear FCMs are possible (Hoyer et al., 2006, 2008; Salehkaleybar et al., 2019), it is nearly impossible to solve analytically the FCMs for complex causal mechanisms in the case of hidden confounding variables.

A possible alternative to alleviate causal sufficiency is to introduce and learn correlated noise between variables in the model, such as in CGNN. Instead of introducing a noise variable between each pair of variables, a handy way to formulate this correlated noise is through a correlation matrix: Let Σ be a correlation between d noise variables E_i, d being the number of variables in the dataset. At each epoch, d independent noise variables ε_i are drawn from the $\mathcal{N}(0, 1)$ distribution; these noises are fed into the correlation matrix, producing the correlated noises E_i given as input to the generators:

$$\{E_1, \dots, E_d\}_{j=1}^{n} = \{\varepsilon_1, \dots, \varepsilon_d\}_{j=1}^{n} \cdot \Sigma \tag{6.3}$$

The introduction of differenciable parameters in the correlation matrix allows the SAM structure to automatically leverage correlation between variables in an efficient manner, and thus provides SAM with means to explain dependencies in the data originating from hidden variables.

6.2.3 Time series

Causal discovery and time series are often closely related: in many domains such as econometrics, bio-informatics, time-series datasets allow practitioners to detect causalities of phenomena more accurately than cross-sectional observational data. Indeed, the delays between the

[1] cropping values to 1 is necessary to obtain a binary adjacency matrix

causes and the effects highlights the causal mechanisms, and can be exploited by algorithms to infer causal relationships. Traditionally, practitioners leverage Granger causality to reveal causal relationships; Let X_t and Y_t two temporal series, Granger causality states that:

$$X \overset{g}{\rightarrow} Y \quad \text{if} \quad [X_0, \dots, X_t] \;\; \text{allows to predict} \;\; Y_{t+1}$$

An adaptation of the SAM architecture to support time series would be to introduce recurrent neural networks (RNNs) in the generators, such as in the C-RNN-GAN (Mogren, 2016). This new structure would allow SAM to detect dependencies in time because of the memory in the RNN structure.

6.2.4 Adaptation for discrete and mixed-type data

For neural network architectures to seamlessly support the usage of discrete or mixed type data, an adaptation of the structure of the neural network is often needed. Usually, this adaptation is made using the one-hot encoding of variables and adding a softmax or sigmoid function at the output. However, applying those changes directly to SAM is not possible because of the generators and the GAN architecture: the output of the generators should be discrete variables and not logit values nor probabilities (output of the sigmoid/softmax function). This is due to the adversarial setting that will easily distinguish generated numerical data from true discrete data.

One solution to circumvent this issue is to use both a one-hot encoding along with the Gumbel-Softmax trick (Maddison et al., 2016; Jang et al., 2016) at the output of the generators: it will allow for seamless generation of discrete variables. However, adding a one hot encoding to the already memory-heavy SAM does penalize its computational complexity; other approaches for encoding discrete data such as auto-encoding might bring more efficient solutions to this problem.

6.2.5 Cyclic graphs

Most of the causal discovery algorithms seek directed acyclic graphs (DAG), and few attempts have been made to recover cyclic graphs without assuming linear mechanisms (Forré and Mooij, 2018). Making this acyclicity assumption proves itself to be non-realistic in some real cases such as protein regulation, where feedback loops intervene to regulate the production or inhibition of a protein. This problem proves itself to be tricky, as the removal of the DAG constraint in SAM (Zheng et al., 2018b), does shift the parent-children recovery objective to a Markov blanket recovery problem (c.f. Section 5.2.1). Indeed, the spouses of the variables will be taken into account as they are not independent conditionally to the children variables. An alternative to this DAG constraint would be another constraint, specifically targeting V-structures and spouses. This constraint should be a soft constraint, as structures such as the one depicted on Fig. 6.2 should also be allowed.

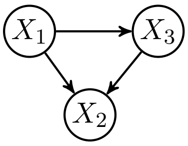


Figure 6.2: Causal structure where the spouse is directly causally related

Another alternative to the DAG constraint would be to add another constraint such as Forré and Mooij (2018) that checks the global and local consistency of the graph at each epoch, at the cost of computational complexity.

6.3 Long-term perspectives

Modeling and leveraging interventions and experiments Interventions and experiments represent the gold standard to infer causal relationships: they provide reliable guarantees on the causal directions. However, these are often hard to obtain or very costly, making this data quite sparse. Implementing and adapting algorithms to take account for these new kinds of data allow for more robust causal discovery.

On the opposite, having a model that could predict the causal effect of a variable given another could provide decision makers crucial information: being able to predict accurately counterfactual effects with certainty and error margins in order to make optimal decisions.

Domain adaptation for the cause-effect problem The algorithms developed for the cause-effect pair challenges (Guyon, 2013, 2014) have introduced a new paradigm of considering the pairwise causal discovery problem as a pattern recognition problem. These algorithms managed to achieve strong performance on the challenges, attaining AUROC scores of 0.8. However, practitioners soon noticed that these algorithm had their performance quite dependent of the pairs given as training and the test pairs, more specifically dependent on the difference between those sets of data. This issue can be associated with a domain adaptation problem, where the source domain consists in artificial pairs, and the target domain corresponds to real data, or test data. Introducing domain adaptation through a GAN setting (Ganin et al., 2016) for causal discovery would allow algorithms such as RCC or NCC (Lopez-Paz et al., 2015, 2016) to cope with the difference between the sets of data during the training phase of the neural network architecture.

Appendices

Appendix A

Causal discovery toolbox

Causal modeling is key to understand physical or artificial phenomena and to guide interventions. Most softwares for causal discovery have been developed in the R programming language (Kalisch et al., 2018; Scutari, 2018), and a few causal discovery algorithms are available in Python e.g. RCC (Lopez-Paz et al., 2015), CGNN (Goudet et al., 2018) and SAM (Kalainathan et al., 2019), while Python supports many current machine learning frameworks such as PyTorch (Paszke et al., 2017).

The **Causal Discovery Toolbox** (CDT) is an open-source Python package concerned with observational causal discovery, aimed at learning both the causal graph and the associated causal mechanisms from samples of the joint probability distribution of the data. CDT includes many state-of-the-art causal modeling algorithms (some of which are imported from R), that supports GPU hardware acceleration and automatic hardware detection. A main goal of CDTis to provide the users with guidance towards end-to-end experiments, by including scoring metrics, and standard benchmark datasets such as the ”Sachs” dataset (Sachs et al., 2005).

Compared to other causal discovery packages, CDT unifies pairwise and score-based multivariate approaches within a single package, implementing an step-by-step pipeline approach (Fig. A.1).

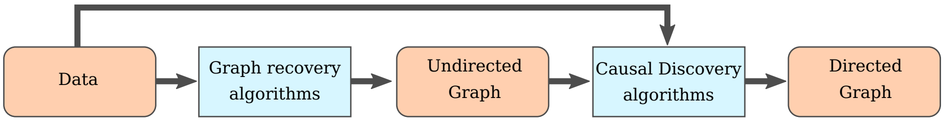


Figure A.1: The CDT causal modeling package: General pipeline

CDT also provides an intuitive approach for including R-based algorithms, facilitating the task of extending the toolkit with additional R packages. The package revolves around the usage of *networkx.Graph* classes, mainly for recovering (un)directed graphs from observational data. CDT currently includes 17 algorithms for graph skeleton identification: 7 methods based on independence tests, and 10 methods aimed at directly recovering the skeleton graph.

It further includes 20 algorithms aimed at causal directed graph prediction, including 11 graphical and 9 pairwise approaches.

A.1 Original contributions of the package

The causal pairwise setting considers a pair of variables and aims to determine the causal relationship between both variables. This setting implicitly assumes that both variables are already conditioned on other covariates, or readjusted with a propensity score (Rosenbaum and Rubin, 1983), and that the remaining latent covariates have little or no influence and can be considered as "noise". The pairwise setting is also relevant to complete a partially directed graph resulting from other causal discovery methods. In the 2010s, the pairwise setting was investigated by Hoyer et al. (2009) among others, who proposed the Additive Noise Model (ANM). Later on, Guyon (2013) on Cause-Effect pair (CEP) problems; CEP formulates bivariate causal identification as a supervised machine learning task, where a classifier is trained from examples (A_i, B_i, ℓ_i), where the variable pair (A_i, B_i) is represented by samples of their joint distribution and label ℓ_i indicates the type of causal relationship between both variables (independent, $A_i \to B_i$, $B_i \to A_i$). CDT is the only package in any language to include causal pairwise discovery algorithms. These algorithms, mostly implemented using Python or Matlab are often left unmaintained. Therefore, many algorithms that are known to be quite efficient (such as Jarfo (Fonollosa, 2016), first and first in the cause-effect pairs challenges, coded in Python 2.7) are outdated and require a substantial amount of work to fix and update. CDT implements 9 pairwise algorithms, all coded in Python, 5 of them being new implementations (NCC, GNN, CDS, RECI and a baseline method based on regression error).

The graph setting, extensively studied in the literature, is supported by many packages. Bayesian approaches rely either on conditional independence tests named **constraint-based methods**, such as PC or FCI (Spirtes et al., 2000; Strobl et al., 2017), or on **score-based methods**, involving finding the graph that maximizes a likelihood score through graph search heuristics, like GES (Chickering, 2002) or CAM (Bühlmann et al., 2014). Other approaches leverage the Generative Network setting, such as CGNN or SAM (Goudet et al., 2018; Kalainathan et al., 2019). Graph setting methods output either a directed acyclic graph or a partially directed acyclic graph. Most approaches in the graph setting are imported from R packages, with the exception of CGNN and SAM.

A.2 Comparison with other packages

To our best knowledge, **Causality** and **Py-Causal** are the only alternatives to CDT for causal discovery in Python. However, the only overlap with CDT concerns the PC-algorithm, common to Py-Causal and CDT. Akin to CDT, Py-Causal is a wrapper package but around the Tetrad Java package. Fig. A.2 compares the runtimes of the two PC implementations on synthetic graphs with of varying size, connectivity, and number of data points, showing a constant gap in with respect to the number of data points and connectivity of the graph. This gap is due to the creation of the subprocess and the data transfer, that are not taken into account in the PyCausal execution runtime. The gap with respect to the number of nodes is due to different implementations and computational complexity. Further effort will be devoted to imposing the efficiency of our Python-Numba implementation of PC.

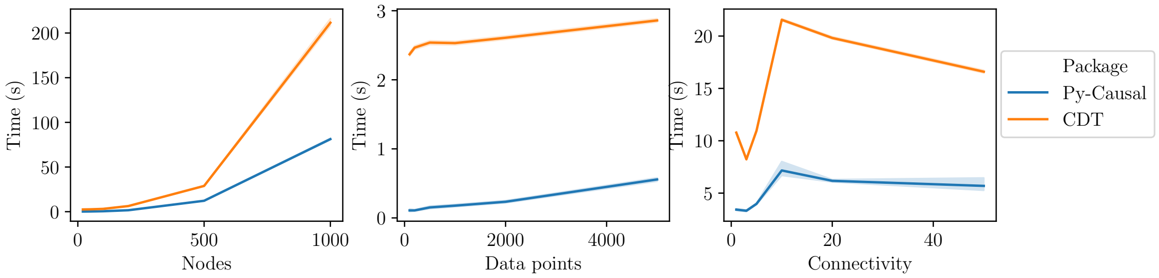


Figure A.2: Runtimes of implementations of PC on various graphs

A.3 Implementation and utilities

R integration. As said, the CDT package integrate 10 algorithms coded in R and 17 coded in Python. The CDT package integrates all of them, using Wrapper functions in Python to enable the user to launch any R script and to control its arguments; the R scripts are executed in a temporary folder with a *subprocess* to avoid the limitations of the Python GIL. The results are retrieved through output files back into the main Python process. The whole procedure is modular and allows contributors to easily add new R functions to the package.

Sustainability and deployment. In order for the package to be easily extended, fostering the integration of further community contributions, special care is given to the quality of tests. Specifically, a Continuous Integration tool added to the git repository, allows to sequentially execute tests on new commits and pull request: i) Test all functionalities of the new version on the package on toy data sets; ii) Build docker images and push them to hub.docker.com ; iii) Push the new version on *pypi*; iv) Update the documentation website. This procedure also allows to test the proper functioning of the package with its dependencies.

A.4 Conclusion and future developments

The Causal Discovery Toolbox (CDT) package allows Python users to apply many causal discovery or graph modeling algorithms on observational data. It is already used in research projects, such as (Yale et al., 2018; Kalainathan et al., 2019). As the output graphs are `networkx.Graph` classes, these are easily exportable into various formats for visualization softwares, using e.g. Graphviz or Gephi. At the package import, tests are realized to pinpoint the configuration of the user: availability of GPUs and R packages and number of CPUs on the host machine.

The package promotes an end-to-end, step-by-step approach: the undirected graph (bivariate dependencies) is first identified, before applying causal discovery algorithms; the latter are constrained from the undirected graph, with significant computational gains.

Future extensions of the package include: i) reimplementing the R algorithms in Python - Numba and reimplement the Pytorch algorithms in Chainer to drop all heavy dependencies and to integrate Cdt in the Python community with a Numpy-API ; ii) developing GPU-compliant implementation of new algorithms; iii) handling interventional data and time-series data (e.g. for neuroimaging and weather forecast). In the longer term, our priority is to provide the user with tests to whether the standard assumptions (e.g. causal sufficiency assumption) hold and assess the risk of applying methods out of their intended scope.

Bibliography

Anandkumar, A., Hsu, D., Javanmard, A., and Kakade, S. (2013). Learning linear bayesian networks with latent variables. In *International Conference on Machine Learning*, pages 249–257.

Aragam, B., Gu, J., and Zhou, Q. (2017). Learning large-scale bayesian networks with the sparsebn package. *arXiv preprint arXiv:1703.04025*.

Aragam, B. and Zhou, Q. (2015). Concave penalized estimation of sparse gaussian bayesian networks. *Journal of Machine Learning Research*, 16:2273–2328.

Banerjee, O., Ghaoui, L. E., and d'Aspremont, A. (2008). Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data. *Journal of Machine learning research*, 9(Mar):485–516.

Barron, A. R. and Cover, T. M. (1991). Minimum complexity density estimation. *IEEE transactions on information theory*, 37(4):1034–1054.

Bartlett, M. and Cussens, J. (2017). Integer linear programming for the bayesian network structure learning problem. *Artificial Intelligence*, 244:258–271.

Better, M., Glover, F., and Laguna, M. (2007). Advances in analytics: Integrating dynamic data mining with simulation optimization. *IBM journal of research and development*, 51(3.4):477–487.

Bothorel, B., Goudet, O., and Duval, B. (2019). Inférence de graphes de régulation génétique. application au génome de la plante arabidopsis thaliana. *Université de Bordeaux*.

Brown, G., Pocock, A., Zhao, M.-J., and Luján, M. (2012). Conditional likelihood maximisation: a unifying framework for information theoretic feature selection. *Journal of machine learning research*, 13(Jan):27–66.

Budhathoki, K. and Vreeken, J. (2017). Causal inference by stochastic complexity. *arXiv preprint arXiv:1702.06776*.

Bühlmann, P., Peters, J., Ernest, J., et al. (2014). Cam: Causal additive models, high-dimensional order search and penalized regression. *The Annals of Statistics*, 42(6):2526–2556.

Chickering, D. M. (2002). Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3(Nov):507–554.

Chickering, D. M., Heckerman, D., and Meek, C. (2004). Large-sample learning of bayesian networks is np-hard. *Journal of Machine Learning Research*, 5(Oct):1287–1330.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv*.

Colombo, D. and Maathuis, M. H. (2014). Order-independent constraint-based causal structure learning. *Journal of Machine Learning Research*, 15(1):3741–3782.

Colombo, D., Maathuis, M. H., Kalisch, M., and Richardson, T. S. (2012). Learning high-dimensional directed acyclic graphs with latent and selection variables. *The Annals of Statistics*, pages 294–321.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314.

Daniusis, P., Janzing, D., Mooij, J., Zscheischler, J., Steudel, B., Zhang, K., and Schölkopf, B. (2012). Inferring deterministic causal relations. *arXiv preprint arXiv:1203.3475.*

Dietterich, T. G. (2000). Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer.

Doquet, G. and Sebag, M. (2019). Agnostic feature selection. *ECML PKDD 2019.*

Drton, M. and Maathuis, M. H. (2016). Structure learning in graphical modeling. *Annual Review of Statistics and Its Application*, (0).

Edwards, R. (1964). Fourier analysis on groups.

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., and Hutter, F. (2015). Efficient and robust automated machine learning. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28*, pages 2962–2970. Curran Associates, Inc.

Fonollosa, J. A. (2016). Conditional distribution variability measures for causality detection. *arXiv preprint arXiv:1601.06680.*

Forré, P. and Mooij, J. M. (2018). Constraint-based causal discovery for non-linear structural causal models with cycles and latent confounders. *arXiv preprint arXiv:1807.03024.*

Friedman, J., Hastie, T., and Tibshirani, R. (2008). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. (2016). Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

Glover, F. and Taillard, E. (1993). A user's guide to tabu search. *Annals of operations research*, 41(1):1–28.

Goldberger, A. S. (1984). Reverse regression and salary discrimination. *Journal of Human Resources*.

Golovin, D., Solnik, B., Moitra, S., Kochanski, G., Karro, J., and Sculley, D. (2017). Google vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1487–1495. ACM.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Neural Information Processing Systems (NIPS)*, pages 2672–2680.

Gordon, A., Eban, E., Nachum, O., Chen, B., Wu, H., Yang, T.-J., and Choi, E. (2018). Morphnet: Fast & simple resource-constrained structure learning of deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1586–1595.

Goudet, O., Kalainathan, D., Caillou, P., Guyon, I., Lopez-Paz, D., and Sebag, M. (2018). Learning functional causal models with generative neural networks. In *Explainable and Interpretable Models in Computer Vision and Machine Learning*, pages 39–80. Springer.

Gretton, A., Borgwardt, K. M., Rasch, M., Schölkopf, B., Smola, A. J., et al. (2007). A kernel method for the two-sample-problem. 19:513.

Gretton, A., Bousquet, O., Smola, A., and Schölkopf, B. (2005a). Measuring statistical dependence with hilbert-schmidt norms. In *International conference on algorithmic learning theory*, pages 63–77. Springer.

Gretton, A., Herbrich, R., Smola, A., Bousquet, O., and Schölkopf, B. (2005b). Kernel methods for measuring independence. *Journal of Machine Learning Research*, 6(Dec):2075–2129.

Grünwald, P. D., Vitányi, P. M., et al. (2008). Algorithmic information theory. *Handbook of the Philosophy of Information*, pages 281–320.

Gumbel, E. J. (1954). Statistical theory of extreme values and some practical applications. *NBS Applied Mathematics Series*, 33.

Guyon, I. (2013). Chalearn cause effect pairs challenge.

Guyon, I. (2014). Chalearn fast causation coefficient challenge.

Guyon, I., Statnikov, A., and Bakir Batu, B. (2019). *Cause Effect Pairs in Machine Learning.* Springer International Publishing.

Hara, K., Saitoh, D., and Shouno, H. (2016). Analysis of dropout learning regarded as ensemble learning. In *International Conference on Artificial Neural Networks*, pages 72–79. Springer.

Hauser, A. and Bühlmann, P. (2012). Characterization and greedy learning of interventional markov equivalence classes of directed acyclic graphs. *Journal of Machine Learning Research*, 13(Aug):2409–2464.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., et al. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine.*

Hoyer, P. O., Janzing, D., Mooij, J. M., Peters, J., and Schölkopf, B. (2009). Nonlinear causal discovery with additive noise models. In *Neural Information Processing Systems (NIPS)*, pages 689–696.

Hoyer, P. O., Shimizu, S., and Kerminen, A. J. (2006). Estimation of linear, non-gaussian causal models in the presence of confounding latent variables. *arXiv preprint cs/0603038.*

Hoyer, P. O., Shimizu, S., Kerminen, A. J., and Palviainen, M. (2008). Estimation of causal effects using linear non-gaussian causal models with hidden variables. *International Journal of Approximate Reasoning*, 49(2):362–378.

Hyvärinen, A. and Pajunen, P. (1999). Nonlinear independent component analysis: Existence and uniqueness results. *Neural Networks*, 12(3):429–439.

Imbens, G. W. and Rubin, D. B. (2015). *Causal inference in statistics, social, and biomedical sciences.* Cambridge University Press.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456.

Irrthum, A., Wehenkel, L., Geurts, P., et al. (2010). Inferring regulatory networks from expression data using tree-based methods. *PloS one*, 5(9):e12776.

Jang, E., Gu, S., and Poole, B. (2016). Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Janzing, D., Mooij, J., Zhang, K., Lemeire, J., Zscheischler, J., Daniušis, P., Steudel, B., and Schölkopf, B. (2012). Information-geometric approach to inferring causal directions. *Artificial Intelligence*, 182:1–31.

Janzing, D. and Scholkopf, B. (2010). Causal inference using the algorithmic markov condition. *IEEE Transactions on Information Theory*, 56(10):5168–5194.

Johnson, A. E., Pollard, T. J., Shen, L., Li-wei, H. L., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Celi, L. A., and Mark, R. G. (2016). Mimic-iii, a freely accessible critical care database. *Scientific data*, 3:160035.

Kalainathan, D., Goudet, O., Caillou, P., and Sebag, M. (2018). *Portraits de travailleurs: Comprendre la qualité de vie au travail*. Notes de la Fabrique. Presses de l'Ecole des mines.

Kalainathan, D., Goudet, O., Guyon, I., Lopez-Paz, D., and Sebag, M. (2019). Structural agnostic modelling: Adversarial learning of causal graphs. *arXiv preprint arXiv:1803.04929*.

Kalisch, M., Hauser, A., et al. (2018). Package 'pcalg'.

Kalisch, M., Mächler, M., Colombo, D., Maathuis, M. H., Bühlmann, P., et al. (2012). Causal inference using graphical models with the r package pcalg. *Journal of Statistical Software*, 47(11):1–26.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kingma, D. P. and Ba, J. (2014). Adam: A Method for Stochastic Optimization. *ICLR*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *NIPS*.

Lachapelle, S., Brouillard, P., Deleu, T., and Lacoste-Julien, S. (2019). Gradient-based neural dag learning. *arXiv preprint arXiv:1906.02226*.

Lauritzen, S. L. (1996). *Graphical models*, volume 17. Clarendon Press.

Lemeire, J. and Steenhaut, K. (2010). Inference of graphical causal models: Representing the meaningful information of probability distributions. In *Causality: Objectives and Assessment*, pages 107–120.

Li, C.-L., Chang, W.-C., Cheng, Y., Yang, Y., and Póczos, B. (2017). Mmd gan: Towards deeper understanding of moment matching network. In *Advances in Neural Information Processing Systems*, pages 2203–2213.

Li, M. and Vitányi, P. (2013). *An introduction to Kolmogorov complexity and its applications*. Springer Science & Business Media.

Li, Y., Swersky, K., and Zemel, R. S. (2015). Generative moment matching networks. In *ICML*, pages 1718–1727.

Lopez-Paz, D. (2016). *From dependence to causation*. PhD thesis, University of Cambridge.

Lopez-Paz, D., Muandet, K., Schölkopf, B., and Tolstikhin, I. O. (2015). Towards a learning theory of cause-effect inference. In *ICML*, pages 1452–1461.

Lopez-Paz, D., Nishihara, R., Chintala, S., Schölkopf, B., and Bottou, L. (2016). Discovering causal signals in images. *arXiv preprint arXiv:1605.08179*.

Lopez-Paz, D. and Oquab, M. (2016). Revisiting classifier two-sample tests. *arXiv preprint arXiv:1610.06545*.

Louizos, C., Welling, M., and Kingma, D. P. (2017). Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312*.

Maddison, C. J., Mnih, A., and Teh, Y. W. (2016). The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*.

Marbach, D., Schaffter, T., Floreano, D., Prill, R. J., and Stolovitzky, G. (2009). The dream4 in-silico network challenge. *Draft, version 0.3*.

Marx, A. and Vreeken, J. (2017). Telling cause from effect using mdl-based local and global regression. In *2017 IEEE international conference on data mining (ICDM)*, pages 307–316. IEEE.

Mendes, P., Sha, W., and Ye, K. (2003). Artificial gene networks for objective comparison of analysis algorithms. *Bioinformatics*, 19(suppl_2):ii122–ii129.

Meyer, P. and Bontempi, G. (2013). Information-theoretic gene selection in expression data. *Biological Knowledge Discovery Handbook: Preprocessing, Mining, and Postprocessing of Biological Data*, pages 399–420.

Mitrovic, J., Sejdinovic, D., and Teh, Y. W. (2018). Causal inference via kernel deviance measures. *arXiv preprint arXiv:1804.04622*.

Mogren, O. (2016). C-rnn-gan: Continuous recurrent neural networks with adversarial training. *arXiv preprint arXiv:1611.09904*.

Mooij, J. M., Peters, J., Janzing, D., Zscheischler, J., and Schölkopf, B. (2016). Distinguishing cause from effect using observational data: methods and benchmarks. *Journal of Machine Learning Research*, 17(32):1–102.

Nandy, P., Hauser, A., and Maathuis, M. H. (2015). High-dimensional consistency in score-based and hybrid structure learning. *arXiv preprint arXiv:1507.02608*.

Nauta, M. (2018). Temporal causal discovery and structure learning with attention-based convolutional neural networks. Master's thesis, University of Twente.

Nguyen, X., Wainwright, M. J., and Jordan, M. I. (2010). Estimating divergence functionals and the likelihood ratio by convex risk minimization. *IEEE Transactions on Information Theory*, 56(11):5847–5861.

Nowozin, S., Cseke, B., and Tomioka, R. (2016). f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems*, pages 271–279.

Ogarrio, J. M., Spirtes, P., and Ramsey, J. (2016). A hybrid causal search algorithm for latent variable models. In *Conference on Probabilistic Graphical Models*, pages 368–379.

Paszke, A., Gross, S., Chintala, S., et al. (2017). Automatic differentiation in pytorch.

Pearl, J. (2003). Causality: models, reasoning and inference. *Econometric Theory*, 19(675-685):46.

Pearl, J. (2009). *Causality*. Cambridge university press.

Pearl, J. and Verma, T. (1991). *A formal theory of inductive causation*. University of California (Los Angeles). Computer Science Department.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Peters, J. and Bühlmann, P. (2013). Structural intervention distance (sid) for evaluating causal graphs. *arXiv preprint arXiv:1306.1043*.

Peters, J., Janzing, D., and Schölkopf, B. (2017). *Elements of Causal Inference - Foundations and Learning Algorithms*. MIT Press.

Peters, J., Mooij, J. M., Janzing, D., and Schölkopf, B. (2014). Causal discovery with continuous additive noise models. *The Journal of Machine Learning Research*, 15(1):2009–2053.

Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

Raina, R., Madhavan, A., and Ng, A. Y. (2009). Large-scale deep unsupervised learning using graphics processors. In *Proceedings of the 26th annual international conference on machine learning*, pages 873–880. ACM.

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

Ramsey, J. D. (2015). Scaling up greedy causal search for continuous variables. *arXiv preprint arXiv:1507.07749*.

Rosenbaum, P. R. and Rubin, D. B. (1983). The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70(1):41–55.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

Runge, J. (2017). Conditional independence testing based on a nearest-neighbor estimator of conditional mutual information. *arXiv preprint arXiv:1709.01447*.

Sachs, K., Perez, O., Pe'er, D., Lauffenburger, D. A., and Nolan, G. P. (2005). Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, 308(5721):523–529.

Salehkaleybar, S., Ghassami, A., Kiyavash, N., and Zhang, K. (2019). Learning linear non-gaussian causal models in the presence of latent variables. *arXiv preprint arXiv:1908.03932*.

Scanagatta, M., de Campos, C. P., Corani, G., and Zaffalon, M. (2015). Learning bayesian networks with thousands of variables. In *Advances in neural information processing systems*, pages 1864–1872.

Schaffter, T., Marbach, D., and Floreano, D. (2011). Genenetweaver: in silico benchmark generation and performance profiling of network inference methods. *Bioinformatics*, 27(16):2263–2270.

Scutari, M. (2009). Learning bayesian networks with the bnlearn r package. *arXiv preprint arXiv:0908.3817*.

Scutari, M. (2018). Package ‘bnlearn’.

Sgouritsa, E., Janzing, D., Hennig, P., and Schölkopf, B. (2015). Inference of cause and effect with unsupervised inverse regression. In *AISTATS*.

Shen-Orr, S. S., Milo, R., Mangan, S., and Alon, U. (2002). Network motifs in the transcriptional regulation network of escherichia coli. *Nature genetics*, 31(1):64.

Shimizu, S., Hoyer, P. O., Hyvärinen, A., and Kerminen, A. (2006). A linear non-gaussian acyclic model for causal discovery. *Journal of Machine Learning Research*, 7(Oct):2003–2030.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*.

Spirtes, P., Glymour, C., and Scheines, R. (1993). Causation, prediction and search. 1993. *Lecture Notes in Statistics*.

Spirtes, P., Glymour, C. N., and Scheines, R. (2000). *Causation, prediction, and search*. MIT press.

Spirtes, P., Meek, C., Richardson, T., and Meek, C. (1999). An algorithm for causal inference in the presence of latent variables and selection bias.

Spirtes, P. and Zhang, K. (2016). Causal discovery and inference: concepts and recent methodological advances. In *Applied informatics*, volume 3, page 3. Springer Berlin Heidelberg.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Statnikov, A., Henaff, M., Lytkin, N. I., and Aliferis, C. F. (2012). New methods for separating causes from effects in genomics data. *BMC genomics*, 13(8):S22.

Stegle, O., Janzing, D., Zhang, K., Mooij, J. M., and Schölkopf, B. (2010). Probabilistic latent variable models for distinguishing between cause and effect. In *Neural Information Processing Systems (NIPS)*, pages 1687–1695.

Strobl, E. V., Zhang, K., and Visweswaran, S. (2017). Approximate kernel-based conditional independence tests for fast non-parametric causal discovery. *arXiv preprint arXiv:1702.03877*.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288.

Tsamardinos, I., Aliferis, C. F., Statnikov, A. R., and Statnikov, E. (2003). Algorithms for large scale markov blanket discovery. In *FLAIRS conference*, volume 2, pages 376–380.

Tsamardinos, I., Brown, L. E., and Aliferis, C. F. (2006). The max-min hill-climbing bayesian network structure learning algorithm. *Machine learning*, 65(1):31–78.

Van de Geer, S., Bühlmann, P., et al. (2013). l0-penalized maximum likelihood for sparse directed acyclic graphs. *The Annals of Statistics*, 41(2):536–567.

Van den Bulcke, T., Van Leemput, K., Naudts, B., van Remortel, P., Ma, H., Verschoren, A., De Moor, B., and Marchal, K. (2006). Syntren: a generator of synthetic gene expression data for design and analysis of structure learning algorithms. *BMC bioinformatics*, 7(1):43.

Verma, T. and Pearl, J. (1991). Equivalence and synthesis of causal models. In *Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intelligence*, UAI '90, pages 255–270, New York, NY, USA. Elsevier Science Inc.

Wang, Y. and Blei, D. M. (2018). The blessings of multiple causes. *CoRR*, abs/1805.06826.

Welling, M. (2018). Intelligence per kilowatt-hour. ICML 2018.

Yale, A., Dash, S., Dutta, R., Pavao, A., Kalainathan, D., Guyon, I., and Bennett, K. (2018). Privacy preserving synthetic health data.

Yamada, M., Jitkrittum, W., Sigal, L., Xing, E. P., and Sugiyama, M. (2014). High-dimensional feature selection by feature-wise kernelized lasso. *Neural computation*, 26(1):185–207.

Yu, K., Liu, L., and Li, J. (2018). A unified view of causal and non-causal feature selection. *arXiv preprint arXiv:1802.05844*.

Yu, Y., Chen, J., Gao, T., and Yu, M. (2019). Dag-gnn: Dag structure learning with graph neural networks. *arXiv preprint arXiv:1904.10098*.

Yuan, M. and Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67.

Zaag, R., Tamby, J. P., Guichard, C., Tariq, Z., Rigaill, G., Delannoy, E., Renou, J.-P., Balzergue, S., Mary-Huard, T., Aubourg, S., et al. (2014). Gem2net: from gene expression modeling to-omics networks, a new catdb module to investigate arabidopsis thaliana genes involved in stress response. *Nucleic acids research*, 43(D1):D1010–D1017.

Zhang, J. (2008). On the completeness of orientation rules for causal discovery in the presence of latent confounders and selection bias. *Artificial Intelligence*, 172(16-17):1873–1896.

Zhang, K. and Hyvärinen, A. (2009). On the identifiability of the post-nonlinear causal model. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 647–655. AUAI Press.

Zhang, K. and Hyvärinen, A. (2010). Distinguishing causes from effects using nonlinear acyclic causal models. In *Causality: Objectives and Assessment*, pages 157–164.

Zhang, K., Peters, J., Janzing, D., and Schölkopf, B. (2012). Kernel-based conditional independence test and application in causal discovery. *arXiv preprint arXiv:1202.3775*.

Zheng, X., Aragam, B., Ravikumar, P., and Xing, E. P. (2018a). Dags with NO TEARS: continuous optimization for structure learning. In *NeurIPS*, pages 9492–9503.

Zheng, X., Aragam, B., Ravikumar, P., and Xing, E. P. (2018b). Dags with no tears: Smooth optimization for structure learning. *arXiv preprint arXiv:1803.01422*.

Logo de l'école doctorale

Titre : Réseaux de Neurones Génératifs pour la Découverte de Méchanismes Causaux: Algorithmes et Applications

Mots clés : Causalité, Réseaux de Neurones Génératifs, Réseaux de Neurones Adversariaux, Apprentissage Profond

Résumé : La découverte de relations causales est primordiale pour la planification, le raisonnement et la decision basée sur des données d'observations; confondre correlation et causalité ici peut mener à des conséquences indésirables. La référence pour la découverte de relations causales est d'effectuer des expériences contrôlées. Mais dans la majorité des cas, ces expériences sont coûteuses, immorales ou même impossibles à réaliser. Dans ces cas, il est nécessaire d'effectuer la découverte causale seulement sur des données d'observations. Dans ce contexte de causalité observationnelle, retrouver des relations causales introduit traditionellement des hypothèses considérables sur les données et sur le modèle causal sous-jacent.
Cette thèse vise à relaxer certaines de ces hypothèses en exploitant la modularité et l'expressivité des réseaux de neurones pour la causalité, en exploitant à la fois et indépendences conditionnelles et la simplicité des méchanismes causaux, à travers deux algorithmes. Des expériences extensives sur des données simulées et sur des données réelles ainsi qu'une analyse théorique approfondie prouvent la cohérence et bonne performance des approches proposées.

Title : Generative Neural networks to infer Causal Mechanisms: Algorithms and applications

Keywords : Causal Discovery, Generative Neural Networks, Adversarial Neural Networks, Deep Learning

Abstract : Causal discovery is of utmost importance for agents who must plan, reason and decide based on observations; where mistaking correlation with causation might lead to unwanted consequences. The gold standard to discover causal relations is to perform experiments. However, experiments are in many cases expensive, unethical, or impossible to realize. In these situations, there is a need for observational causal discovery, that is, the estimation of causal relations from observations alone. Causal discovery in the observational data setting traditionally involves making significant assumptions on the data and on the underlying causal model. This thesis aims to alleviate some of the assumptions made on the causal models by exploiting the modularity and expressiveness of neural networks for causal discovery, leveraging both conditional independences and simplicity of the causal mechanisms through two algorithms. Extensive experiments on both simulated and real-world data and a throughout theoretical anaylsis prove the good performance and the soundness of the proposed approaches.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France