Université de Montréal

Apprentissage de modèles causaux par réseaux de neurones artificiels

par

Philippe Brouillard

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Mémoire présenté en vue de l'obtention du grade de
Maître ès sciences (M.Sc.)
en informatique

10 juillet 2020

Université de Montréal

Faculté des arts et des sciences

Ce mémoire intitulé

Apprentissage de modèles causaux par réseaux de neurones artificiels

présenté par

Philippe Brouillard

a été évalué par un jury composé des personnes suivantes :

Aaron Courville
(président-rapporteur)

Alain Tapp
(directeur de recherche)

Nicolas Le Roux
(membre du jury)

Résumé

Dans ce mémoire par articles, nous nous intéressons à l'apprentissage de modèles causaux à partir de données. L'intérêt de cette entreprise est d'obtenir une meilleure compréhension des données et de pouvoir prédire l'effet qu'aura un changement sur certaines variables d'un système étudié. Comme la découverte de liens causaux est fondamentale en sciences, les méthodes permettant l'apprentissage de modèles causaux peuvent avoir des applications dans une pléthore de domaines scientifiques, dont la génomique, la biologie et l'économie.

Nous présentons deux nouvelles méthodes qui ont la particularité d'être des méthodes non-linéaires d'apprentissage de modèles causaux qui sont posées sous forme d'un problème d'optimisation continue sous contrainte. Auparavant, les méthodes d'apprentissage de modèles causaux abordaient le problème de recherche de graphes en utilisant des stratégies de recherche voraces. Récemment, l'introduction d'une contrainte d'acyclicité a permis d'aborder le problème différemment.

Dans un premier article, nous présentons une de ces méthodes: GraN-DAG. Sous certaines hypothèses, GraN-DAG permet d'apprendre des graphes causaux à partir de données observationnelles. Depuis la publication du premier article, plusieurs méthodes alternatives ont été proposées par la communauté pour apprendre des graphes causaux en posant aussi le problème sous forme d'optimisation continue avec contrainte. Cependant, aucune de ces méthodes ne supportent les données interventionnelles. Pourtant, les interventions réduisent le problème d'identifiabilité et permettent donc l'utilisation d'architectures neuronales plus expressives. Dans le second article, nous présentons une autre méthode, DCDI, qui a la particularité de pouvoir utiliser des données avec différents types d'interventions. Comme le problème d'identifiabilité est moins important, une des deux instanciations de DCDI est un approximateur de densité universel. Pour les deux méthodes proposées, nous montrons que ces méthodes ont de très bonnes performances sur des données synthétiques et réelles comparativement aux méthodes traditionelles.

Mots clés: apprentissage structuré causal, inférence causale, modèles causaux, apprentissage automatique, réseaux neuronaux, apprentissage profond

Abstract

In this thesis by articles, we study the learning of causal models from data. The goal of this entreprise is to gain a better understanding of data and to be able to predict the effect of a change on some variables of a given system. Since discovering causal relationships is fundamental in science, causal structure learning methods have applications in many fields that range from genomics, biology, and economy.

We present two new methods that have the particularity of being non-linear methods learning causal models casted as a continuous optimization problem subject to a constraint. Previously, causal strutural methods addressed this search problem by using greedy search heuristics. Recently, a new continuous acyclity constraint has allowed to address the problem differently.

In the first article, we present one of these non-linear method: GraN-DAG. Under some assumptions, GraN-DAG can learn a causal graph from observational data. Since the publication of this first article, several alternatives methods have been proposed by the community by using the same continuous-constrained optimization formulation. However, none of these methods support interventional data. Nevertheless, interventions reduce the identifiability problem and allow the use of more expressive neural architectures. In the second article, we present another method, DCDI, that has the particularity to leverage data with several kinds of interventions. Since the identifiabiliy issue is less severe, one of the two instantiations of DCDI is a universal density approximator. For both methods, we show that these methods have really good performances on synthetic and real-world tasks comparatively to other classical methods.

Keywords: causal structure learning, causal inference, causal model, machine learning, neural network, deep learning

Table des matières

Résumé i

Abstract iii

Liste des tableaux ix

Liste des figures xiii

Liste des sigles et des abréviations xvii

Remerciements xix

Introduction 1

Chapitre 1. Apprentissage automatique et apprentissage profond 3

1.1. Apprentissage automatique 3
1.1.1. Problèmes d'apprentissage 3
1.1.2. Apprentissage 4
1.1.3. Généralisation 5

1.2. Apprentissage profond 7
1.2.1. Réseaux de neurones 7
1.2.2. Optimisation et descente de gradient 9
1.2.3. Descente de gradient stochastique et variantes 10
1.2.4. Rétropropagation du gradient 11

Chapitre 2. Causalité 13

2.1. Modèles causaux 15
2.1.1. Réseau bayésien 15
2.1.2. Modèle graphique causal 16
2.1.3. Modèle causal structurel 17

2.2. Apprentissage de modèles causaux 18
2.2.1. Problème d'identifiabilité 19

2.2.2. Algorithmes d'apprentissage de graphes causaux 21
2.2.3. Métriques de graphes causaux 23

Chapitre 3. Gradient-Based Neural DAG Learning 25

3.1. Introduction 26

3.2. Background 27
3.2.1. Causal graphical models 28
3.2.2. Structure identifiability 28
3.2.3. NOTEARS: Continuous optimization for structure learning 29

3.3. GraN-DAG: Gradient-based neural DAG learning 30
3.3.1. Neural network connectivity 30
3.3.2. A weighted adjacency matrix 31
3.3.3. A differentiable score and its optimization 31
3.3.4. Thresholding 33
3.3.5. Overfitting 33
3.3.6. Computational Complexity 34

3.4. Experiments 34
3.4.1. Synthetic data 35
3.4.2. Real and pseudo-real data 37

3.5. Related Work 38

3.6. Conclusion 40

Chapitre 4. Differentiable Causal Discovery from Interventional Data 41

4.1. Introduction 42
4.1.1. Contributions 43

4.2. Background and related work 44
4.2.1. Definitions 44
4.2.2. Causal structure learning 45
4.2.3. Continuous constrained optimization for structure learning 46

4.3. DCDI: Differentiable causal discovery from interventional data 47
4.3.1. A score for imperfect interventions 47
4.3.2. A continuous-constrained formulation 49
4.3.3. Unknown interventions 50

4.3.4. DCDI with normalizing flows . . . 50
4.4. Experiments . . . 51
4.4.1. Results for different intervention types . . . 52
4.4.2. Scalability experiments . . . 53
4.5. Conclusion . . . 54
Chapitre 5. Conclusion . . . 55
Références bibliographiques . . . 57
Appendix A. Article 1: Matériel supplémentaire . . . 67
A.1. Appendix . . . 68
A.1.1. Optimization . . . 68
A.1.2. Thresholding to ensure acyclicity . . . 69
A.1.3. Preliminary neighborhood selection and DAG Pruning . . . 70
A.1.4. Large Sample Size Experiment . . . 72
A.1.5. Details on data sets generation . . . 73
A.1.6. Supplementary experiments . . . 74
A.1.7. Metrics . . . 77
A.1.8. Hyperparameters . . . 78
A.1.9. Hyperparameter Selection via Held-out Score . . . 78
Appendix B. Article 2: Matériel supplémentaire . . . 85
B.1. Theory . . . 86
B.1.1. Theoretical Foundations for Causal Discovery with Imperfect Interventions 86
B.1.2. Proof of Theorem 3 . . . 88
B.2. Additional information . . . 93
B.2.1. Synthetic data sets . . . 93
B.2.2. Deep Sigmoidal Flow: Architectural details . . . 94
B.2.3. Optimization . . . 94
B.2.4. Baseline methods . . . 99
B.2.5. Default hyperparameters and hyperparameter search . . . 99
B.3. Additional experiments . . . 100
B.3.1. Real-world data set . . . 100
B.3.2. Learning causal direction from complex distributions . . . 102

B.3.3. Scalability experiments 104
B.3.4. Ablation study 104
B.3.4.1. Perfect interventions 107
B.3.4.2. Imperfect interventions 108
B.3.4.3. Unknown interventions 109
B.3.5. Different kinds of interventions 109
B.3.5.1. Perfect interventions 110
B.3.5.2. Imperfect interventions 111
B.3.6. Comprehensive results of the main experiments 111
B.3.6.1. Perfect interventions 112
B.3.6.2. Imperfect interventions 113
B.3.6.3. Unknown interventions 114

Liste des tableaux

3.1 Results for ER and SF graphs of 10 nodes with Gauss-ANM data 35

3.2 Results for ER and SF graphs of 50 nodes with Gauss-ANM data 36

3.3 Results on real and pseudo-real data 38

A.1 Total number of iterations ($\times 10^3$) before augmented Lagrangian converges on Gauss-ANM data. 69

A.2 PNS and pruning ablation study for GraN-DAG (averaged over 10 datasets from ER1 with 50 nodes) 71

A.3 PNS and pruning ablation study for DAG-GNN and NOTEARS (averaged over 10 datasets from ER1 with 50 nodes) 72

A.4 Effect of sample size - Gauss-ANM 50 nodes ER4 (averaged over 10 datasets) . . . 72

A.5 Results for ER and SF graphs of 20 nodes with Gauss-ANM data 74

A.6 Results for ER and SF graphs of 100 nodes with Gauss-ANM data 74

A.7 SHD for GES and PC (against GraN-DAG for reference) with Gauss-ANM data 75

A.8 Lower and upper bound on the SID for GES and PC (against GraN-DAG for reference) with Gauss-ANM data. See Appendix A.1.7 for details on how to compute SID for CPDAGs. 75

A.9 LIN 76

A.10 ADD-FUNC 77

A.11 Synthetic post nonlinear data sets 77

A.12 Gauss-ANM - 10 nodes with hyperparameter search 80

A.13 Gauss-ANM - 50 nodes with hyperparameter search 80

A.14 Gauss-ANM - 20 nodes with hyperparameter search 80

A.15 Gauss-ANM - 100 nodes with hyperparameter search 81

A.16 PNL-GP with hyperparameter search 81

A.17 PNL-MULT with hyperparameter search 81

A.18 LIN with hyperparameter search 81
A.19 ADD-FUNC with hyperparameter search 82
A.20 Results for real and pseudo real data sets with hyperparameter search 82
A.21 Hyperparameter search spaces for each algorithm 83

B.1 Hyperparameter search spaces for each algorithm 100
B.2 Default Hyperparameter for DCDI-G and DCDI-DSF 101
B.3 Results for the flow cytometry data sets 101
B.4 Results for the linear data set with perfect intervention 107
B.5 Results for the additive noise model data set with perfect intervention 107
B.6 Results for the nonlinear with non-additive noise data set with perfect intervention 107
B.7 Results for the linear data set with imperfect intervention 107
B.8 Results for the additive noise model data set with imperfect intervention 108
B.9 Results for the nonlinear with non-additive noise data set with imperfect intervention 108
B.10 Results for the linear data set with perfect intervention with unknown targets ... 108
B.11 Results for the additive noise model data set with perfect intervention with unknown targets 108
B.12 Results for the nonlinear with non-additive noise data set with perfect intervention with unknown targets 109
B.13 Results for the linear data set with perfect intervention 110
B.14 Results for the additive noise model data set with perfect intervention 110
B.15 Results for the nonlinear with non-additive noise data set with perfect intervention 110
B.16 Results for the linear data set with imperfect intervention 110
B.17 Results for the additive noise model data set with imperfect intervention 111
B.18 Results for the nonlinear with non-additive noise data set with imperfect intervention 111
B.19 Results for linear data set with perfect intervention 112
B.20 Results for the additive noise model data set with perfect intervention 112
B.21 Results for the nonlinear with non-additive noise data set with perfect intervention 112
B.22 Results for the linear data set with imperfect intervention 113

B.23 Results for the additive noise model data set with imperfect intervention 113
B.24 Results for the nonlinear with non-additive noise data set with imperfect intervention 113
B.25 Results for the linear data set with perfect intervention with unknown targets . . . 114
B.26 Results for the additive noise model data set with perfect intervention with unknown targets 114
B.27 Results for the nonlinear with non-additive noise data set with perfect intervention with unknown targets 114

Liste des figures

1.1 Représentation de a) sur-apprentissage, b) sous-apprentissage et c) d'une modélisation adéquate où k correspond au degré du polynôme utilisé pour accomplir la tâche de régression. Dans chaque figure, les cercles représentent les données d'entraînement, tandis que les losanges ombragés représentent les données de validation. Les courbes représentent les fonctions apprises. 5

1.2 Représentation d'un réseau de neurones artificiels à deux couches cachées. La direction de la propagation avant et de la rétropropagation y est illustrée. 8

2.1 Représentation de graphes formés de 3 noeuds. En a), deux chaînes, en b) et en c) une v-structure. 15

2.2 Représentation d'un graphe G associé à un certain modèle causal et d'un graphe G' induit par une intervention parfaite dans le même modèle sur la variable X_2. . 17

2.3 Chaque image représente $\mathcal{G}$ l'ensemble des graphes et $\mathcal{P}$ l'ensemble des distributions sur X. G représente le graphe qui a été utilisé pour générer P_X et les liens orientés illustre que P_X est Markov à certains graphes. La zone ombragée en c) représente le sous-ensemble de $\mathcal{G}$ qui respecte une certaine supposition sur le modèle. 19

2.4 En a), un CPDAG représentant une classe d'équivalence de Markov. En b) et c), les deux DAGs appartenant à cette classe d'équivalence. Exemple adapté de (Chickering, 2002). 20

2.5 Un graphe G et deux graphes estimés $\hat{G}_1$ et $\hat{G}_2$ ne différant que d'un lien de G. Pour $\hat{G}_1$, le lien $X_4 \to X_3$ est superflu par rapport à G. Pour $\hat{G}_2$, le lien entre X_1 et X_2 a été inversé. Exemple adapté de (Peters et Bühlmann, 2015b). 24

4.1 Different intervention types (shown in red). In imperfect interventions, the causal relationships are altered. In perfect interventions, the targeted node is cut out from its parents. 44

4.2 **Perfect interventions.** SHD and SID (lower is better) for 20-node graphs 52

4.3 **Imperfect interventions.** SHD and SID for 20-node graphs 53

4.4 **Unknown interventions.** SHD and SID for 20-node graphs 53

A.1 Entries of the weighted adjacency matrix A_ϕ as training proceeds in GraN-DAG for a synthetic data set ER4 with 10 nodes. Green curves represent edges which appear in the ground truth graph while red ones represent edges which do not. The horizontal dashed line at 10^{-4} is the threshold ϵ introduced in Section 3.3.4. We can see that GraN-DAG successfully recovers most edges correctly while keeping few spurious edges. 76

B.1 Different $\mathcal{I}$-DAGs with a single intervention. The first graph is alone in its $\mathcal{I}$-Markov equivalence class since reversing the $1 \to 2$ edge would break the immorality $1 \to 2 \leftarrow \zeta$. The second graph is also alone in its equivalence class since reversing $1 \to 2$ would create a new immorality $\zeta \to 1 \leftarrow 2$. The third DAG is not alone in its equivalence class since reversing $1 \to 2$ would preserve the skeleton without adding or removing an immorality. It should become apparent that adding more interventions will likely reduce the size of the $\mathcal{I}$-Markov equivalence class by introducing more immoralities. 87

B.2 **Top:** Learning curves during training. *NLL* and *NLL on validation* are respectively the (pseudo) negative log-likelihood (NLL) on training and validation sets. *AL minus NLL* can be thought of as the acyclicity constraint violation plus the edge sparsity regularizer. *AL* and *AL on validation set* are the augmented Lagrangian objectives on training and validation set, respectively. **Middle and bottom:** Entries of the matrix $\sigma(\Lambda)$ w.r.t. to the number of iterations (green edges = edge present in the ground truth DAG, red edges = edge not present). The adjacency matrix to the left correspond to the ground truth DAG. The other matrices correspond to $\sigma(\Lambda)$ at 20 000, 30 000 and 62 000 iterations. 97

B.3 Learned targets $\sigma(\beta_{kj})$ compared to the ground truth targets. 99

B.4 Joint density learned by DCDI-DSF. White dots are data points and the color represents the learned density. The x-axis is cause and the y-axis is the effect. First row is observational while second row is with an imperfect intervention on the effect. 102

B.5 Joint density learned by DCDI-G. White dots are data points and the color represents the learned density. The x-axis is cause and the y-axis is the effect. First row is observational while second row is with an imperfect intervention on the effect. 103

B.6 We report the runtime (in hours), SHD and SID of multiple methods in multiple settings. The horizontal dashed lines at 12 hours represents the time limit imposed. When a curve reaches this dashed line, it means that the method could not finish within 12 hours. We write $\geq$ 16G when the RAM memory needed by the algorithm exceeded 16GB. All data sets have 10 interventional targets containing $0.1d$ targets. We considered perfect interventions. **Left:** Different data set sizes. Ten nodes ANM data with connectivity $e = 1$. **Right:** Different number of variables. NN data set with connectivity $e = 4$ and 10^4 samples. Each curve is an average over 5 different datasets while the error bars are %95 confidence intervals computed via bootstrap. 105

B.7 SHD and SID for DCDI-G and DCD on data sets with a different number of interventional settings. 106

Liste des sigles et des abréviations

ANM	Modèle à bruit additif, de l'anglais *Additive Noise Model*
BN	Réseau bayésien, de l'anglais *Bayesian Network*
CGM	Modèle graphique causal, de l'anglais *Causal Graphical Model*
CPDAG	Graphe complété partiellement orienté acyclique, de l'anglais *Completed Partially Directed Acyclic Graph*
DAG	Graphe orienté acyclique, de l'anglais *Directed Acyclic Graph*
DSF	Flot sigmoidal profond, de l'anglais *Deep Sigmoidal Flow*
KCI-test	Test d'indépendance conditionnel par noyau, de l'anglais *Kernel Conditional Independence test*
NN	Réseau de neurones, de l'anglais *Neural Network*
PNS	Sélection préliminaire de voisins, de l'anglais *Preliminary Neighbors Selection*

ReLU	Unité linéaire rectifiée, de l'anglais *Rectified Linear Unit*
SHD	Distance structurelle de Hamming, de l'anglais *Structural Hamming Distance*
SID	Distance structurelle interventionnelle, de l'anglais *Structural Interventional Distance*

Remerciements

Je souhaite en premier lieu remercier mon directeur de recherche Alain Tapp pour sa supervision et pour sa curiosité contagieuse.

En second lieu, j'aimerais remercier mes collaborateurs qui ont rendu ce travail possible: Sébastien Lachapelle, Tristan Deleu, Alexandre Drouin, Alexandre Lacoste et Simon Lacoste-Julien. Un merci particulier à Sébastien qui m'a introduit au domaine fantastique qu'est la causalité.

Un remerciement à Patrick Roy pour m'avoir incité à l'époque à m'inscrire au cours *séminaire d'introduction aux sciences cognitives*. Un remerciement aussi à Pierre Poirier et Luc Faucher qui ont su rendre ce cours extrêmement intéressant. C'est lors de ce cours que j'ai découvert pour la première fois l'apprentissage profond et que j'ai pris la décision de faire de la recherche dans ce domaine.

J'aimerais aussi remercier mes collègues du Mila (et autres) avec qui j'ai toujours du plaisir: Florian Golemo, Meta Markovic, Dmitry Serdyuk, Jeanne Brière, Tristan Sylvain et Virgile Sylvain. Et j'en oublie plusieurs...

J'aimerais remercier mon père et ma soeur qui me donnent leur soutien inconditionnel. Un énorme merci à Assya, ma compagne de vie, pour son soutien, ses conseils et les bons moments qui ont pu rendre ce mémoire réalisable.

Introduction

Une révolution dans le domaine de l'intelligence artificielle entraînant des impacts majeurs sur notre société a récemment eu lieu. Plusieurs de ces succès les plus saillants ont été grandement médiatisés: conduite automobile autonome (Bojarski et al., 2016), performance surhumaine aux jeux de Go (Silver et al., 2016; Singh et al., 2017), d'échecs et de Shogi (Silver et al., 2018) et à des jeux vidéo de stratégies en temps réel (Vinyals et al., 2019). Ces succès s'inscrivent dans la quête de créer de l'intelligence artificielle. Bien qu'il soit difficile de donner une définition définitive de l'intelligence artificielle (Russell et Norvig, 2002), on accepte souvent la définition générale qui postule que c'est un système, qui sur la base de son environnement, a les capacités de résoudre des problèmes de manière adaptative. Si on établit un lien plus direct avec son homologue qu'est l'intelligence humaine, on dira que l'intelligence artificielle cherche à créer des modèles qui ont des capacités semblables aux capacités cognitives des humains, soit les capacités de catégoriser, apprendre, généraliser, développer et utiliser des modèles conceptuels (Larivée, 2006). Le développement de systèmes intelligents est, d'une part, une opportunité de mieux comprendre l'intelligence en soi et d'autre part, d'améliorer notre qualité de vie par le développement d'applications dans des domaines aussi variés que le diagnostic médical, la découverte de médicaments ou la traduction automatique.

Les succès récents de l'intelligence artificielle proviennent principalement d'un sous-domaine nommé apprentissage profond. Inspiré de réseaux de neurones biologiques, les modèles d'apprentissage profond sont constitués de réseaux de neurones artificiels qui peuvent apprendre automatiquement à partir de données. Contrairement à d'anciennes méthodes qui nécessitaient le travail d'experts pour identifier un ensemble de règles, les modèles d'apprentissage profond sont en mesure d'apprendre automatiquement à partir de données brutes, en autant qu'il y ait un grand nombre d'exemples. Dans une multitude de domaines tels que la reconnaissance d'image, la reconnaissance faciale, la traduction automatique, ce sont les modèles d'apprentissage profond qui obtiennent les résultats à l'état de l'art. Malgré les avancées indéniables de l'apprentissage profond, plusieurs limitations ont été mises de l'avant récemment (Marcus, 2018; Pearl, 2018). Entre autres, les modèles sont très "fragiles", c'est-à-dire que lorsque déployés dans un contexte légèrement différent au contexte

d'entraînement, leur performance peut grandement se détériorer. Ces problèmes limitent les bénéfices d'applications qui peinent à généraliser lorsqu'ils sont déployés dans un environnement différent ou qui varie au cours du temps. Pourtant, cette capacité à généraliser ou à extrapoler est naturelle chez l'humain et est présente déjà chez les enfants en bas âge (Walker et Gopnik, 2013). Dans l'objectif de concevoir des systèmes intelligents, il semble impératif de contrevenir à ces limitations.

Plusieurs chercheurs ont mis de l'avant un ingrédient manquant (Pearl, 2018; Schölkopf, 2019): la causalité. En effet, la grande majorité des modèles d'apprentissage profond sont purement statistiques. En intégrant des principes du domaine de la causalité, il devient possible de modéliser des changements encourus lorsque l'environnement subit des modifications. De plus, à partir de modèles causaux, il peut être possible de répondre à des questions contrefactuelles, c'est-à-dire des questions du type "Que serait-il arrivé si j'avais fait l'action X au lieu de l'action Y?". En d'autres termes, certaines des capacités désirées de l'intelligence artificielle (généraliser, développer et utiliser des modèles conceptuels) qui ne sont pas présentes dans les modèles actuels semblent atteignables en intégrant des concepts du domaine de la causalité.

Ce mémoire présente mes travaux de recherche au cours de ma maîtrise à l'Université de Montréal sous la supervision d'Alain Tapp. Deux études seront présentées où, en se basant sur des techniques d'apprentissage profond, on cherche à apprendre des graphes causaux à partir de données. La première étude présente un modèle qui peut apprendre des modèles causaux à partir de données sans intervention. La deuxième étude présente une approche, qui peut être vue comme une extension de la première étude, qui apprend des modèles causaux en tirant profit de données interventionnelles.

Le chapitre 1 présentera le domaine de l'apprentissage automatique et plus précisément le domaine de l'apprentissage profond. Cette présentation ne se veut aucunement une revue exhaustive, mais sert à contextualiser et à présenter des concepts nécessaires à la compréhension du mémoire. Le chapitre 2 présentera brièvement le domaine de la causalité et de l'apprentissage de modèles causaux. Le chapitre 3 et 4 présenteront les deux études mentionnées précédemment. Finalement, le chapitre 5 conclura en présentant des avenues de recherche qui semblent particulièrement intéressantes.

Chapitre 1

Apprentissage automatique et apprentissage profond

1.1. Apprentissage automatique

En apprentissage automatique, on cherche à apprendre automatiquement à partir de données un modèle qui pourra accomplir une certaine tâche. On veut que ce modèle soit en mesure de généraliser, c'est-à-dire, faire des prédictions sur des données jamais observées auparavant. Cette section constitue un bref survol de l'apprentissage automatique qui est nécessaire pour la compréhension du mémoire. Pour une présentation plus détaillée, consultez Murphy (2012) et Bishop (2006).

1.1.1. Problèmes d'apprentissage

De manière générale, on identifie deux grandes classes de problème d'apprentissage: l'*apprentissage supervisé* et l'*apprentissage non-supervisé*. La distinction provient du type de données utilisées et le genre de questions auxquelles on souhaite répondre. En apprentissage supervisé, les données D sont sous la forme $\{(\boldsymbol{x}^{(i)},y^{(i)})\}_{i=1}^{n}$ où $\boldsymbol{x} \in \mathcal{X}$ représente un vecteur de caractéristiques, $y \in \mathcal{Y}$ représente une *étiquette* (ou *cible*) et n représente le nombre d'échantillons. On suppose que ces données ont été échantillonnées d'une certaine distribution $P_{X,Y}$ de manière indépendante (*indépendantes et identiquement distribuées*). On cherchera à estimer la probabilité d'une étiquette étant donné ses caractéristiques, soit $p(y|\boldsymbol{x})$ [1]. Lorsque l'espace des étiquettes est discret, soit $\mathcal{Y} = \{1, \ldots, k\}$, c'est un problème dit de *classification* tandis que lorsque l'espace des étiquettes est continu, soit $\mathcal{Y} = \mathbb{R}$, c'est un problème de *régression*. Lorsque la prédiction est un vecteur $\boldsymbol{y}$ où les composants y_i sont dépendants entre eux, on réfère à cette tâche comme un problème d'*apprentissage structuré*.

[1] Pour simplifier la notation, nous utilisons la notation où $p(x)$ est équivalent à $p(X = x)$

En apprentissage non-supervisé, les données ne contiennent pas d'étiquettes, soit $D = \{\boldsymbol{x}^{(i)}\}_{i=1}^{n}$. En général, on cherche à estimer la probabilité jointe P_X. À partir de cette estimation de densité, le type de tâche peut grandement varier: partitionnement de données (*clustering*), génération de données, détection d'anomalies, etc.

1.1.2. Apprentissage

Pour simplifier la présentation de l'apprentissage, concentrons-nous sur les problèmes d'apprentissage supervisé. À partir de données, on cherche une certaine fonction $f : \mathcal{X} \to \mathcal{Y}$ parmi une certaine famille qui pourra résoudre une certaine tâche de classification ou de régression. Plus précisément, dans le cas d'un modèle paramétrique $f_{\boldsymbol{\theta}}$, on cherche la valeur des paramètres $\boldsymbol{\theta}$ qui minimiseront une certaine *fonction de coût* $L : \mathcal{Y} \times \mathcal{Y} \to [0, \infty)$ qui indique à quel point $f_{\boldsymbol{\theta}}$ est adapté à résoudre le problème. Une valeur faible de la fonction de coût L indique une erreur faible: les prédictions de $f_{\boldsymbol{\theta}}$ sont proches des vraies valeurs. À partir de la fonction de coût, on peut définir le *risque* suivant sur $\boldsymbol{\theta}$:

$$R(\boldsymbol{\theta}) = \mathbb{E}_{(\boldsymbol{x},y)\sim P_{X,Y}}[L(f_{\boldsymbol{\theta}}(\boldsymbol{x}), y)] \tag{1.1.1}$$

On souhaite trouver le $\boldsymbol{\theta}$ qui minimisera le risque, soit:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} R(\boldsymbol{\theta}) \tag{1.1.2}$$

Bien que certains choix de fonction de coût semblent naturels, en pratique on utilise une fonction de coût substitut (*surrogate loss*). Souvent, la fonction de coût découle de la log-vraisemblance. Par exemple, pour un problème de classification à deux classes, on veut réduire la *perte 0-1*, soit:

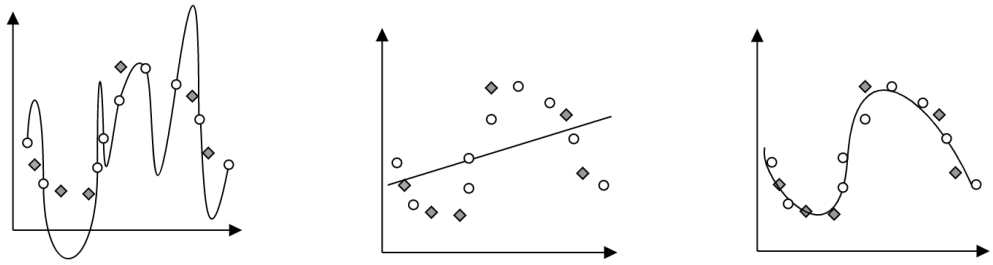
$$L(f_{\boldsymbol{\theta}}(\boldsymbol{x}), y) = \begin{cases} 0 & \text{si } f_{\boldsymbol{\theta}}(\boldsymbol{x}) = y \\ 1 & \text{sinon} \end{cases} \tag{1.1.3}$$

Cependant, comme cette fonction n'est pas continue et qu'un grand nombre de méthodes d'apprentissage automatique se basent sur la dérivée de la fonction de coût, il est souvent avantageux de prendre le *négatif log-vraisemblance* (*negative log-likelihood*) comme risque en utilisant:

$$L(f_{\boldsymbol{\theta}}(\boldsymbol{x}), y) = -\log p_{\boldsymbol{\theta}}(y|\boldsymbol{x}) \tag{1.1.4}$$

Si on définit $p_{\boldsymbol{\theta}}(y|\boldsymbol{x}) = Ber(\hat{y})$ où $\hat{y} = f_{\boldsymbol{\theta}}(\boldsymbol{x})$ (et en supposant qu'on s'assure que $0 \leq \hat{y} \leq 1$), on retrouve la forme classique de l'*entropie croisée binaire* (la fonction de coût la plus couramment utilisé pour des problèmes de classification à deux classes) qui est continue:

$$\begin{aligned} L(y, \hat{y}) &= -\log(\hat{y}^y(1-\hat{y})^{1-y}) \\ &= -y\log\hat{y} - (1-y)\log(1-\hat{y}) \end{aligned} \tag{1.1.5}$$



(a) $k = 8$ (b) $k = 1$ (c) $k = 3$

Fig. 1.1. Représentation de a) sur-apprentissage, b) sous-apprentissage et c) d'une modélisation adéquate où k correspond au degré du polynôme utilisé pour accomplir la tâche de régression. Dans chaque figure, les cercles représentent les données d'entraînement, tandis que les losanges ombragés représentent les données de validation. Les courbes représentent les fonctions apprises.

De la même manière, pour les problèmes de régression, l'*erreur quadratique moyenne*, qui découle aussi de la log-vraisemblance en assumant que $p_{\boldsymbol{\theta}}(y|\boldsymbol{x})$ suit une normale, est couramment utilisée.

1.1.3. Généralisation

En pratique, nous n'avons pas accès à la distribution des données, mais seulement à un nombre fini d'exemples. On estime donc le risque précédent par le *risque empirique* qui est un estimateur non-biaisé du risque $R(\boldsymbol{\theta})$, soit:

$$\hat{R}(\boldsymbol{\theta}, D) = \frac{1}{|D|} \sum_{i=1}^{|D|} L(f_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}), y^{(i)}) \tag{1.1.6}$$

Si f_θ est une méthode très expressive, minimiser le risque empirique pourrait être problématique. C'est-à-dire, qu'on pourrait par exemple avoir une fonction qui mémorise les données et retourne un risque empirique de 0. Cependant, on ne cherche pas tant à prédire les exemples sur lesquelles on s'entraînent, mais bien à généraliser à de nouveaux exemples provenant de la même distribution. Voyons comment s'assurer que notre méthode généralise bien à de nouvelles données.

Pour commencer, illustrons la *sélection de modèle* en utilisant une tâche de régression où $f_{\boldsymbol{\theta}}$ est un polynôme de degré k. Si on peut choisir le degré k, alors pour $k = n - 1$ on obtient un risque de 0 sur les données. Cependant, on obtiendrait probablement un risque très élevé sur de nouvelles données en utilisant ce même polynôme (voir Figure 1.1 a)). En pratique, on partitionne le jeu de données en un ensemble d'entraînement D_{train}, un ensemble de validation D_{val} et un ensemble de test D_{test}. On trouve $\boldsymbol{\theta}$ en minimisant le risque empirique sur D_{train}, mais on choisit les *hyperparamètres* en minimisant le risque empirique sur D_{val} (en utilisant le $\boldsymbol{\theta}$ trouvé sur D_{train}). Dans le cas de la régression polynomiale, on

peut voir en Figure 1.1 c), que c'est avec un $k = 3$ qu'on obtient un risque minimal sur D_{val}. Par *hyperparamètre*, on désigne généralement des paramètres qui peuvent contrôler le comportement d'un algorithme et qui ne peuvent pas être facilement optimisés. Dans le cas de la régression polynomiale, le degré k est clairement un hyperparamètre, car il contrôle directement l'expressivité de l'algorithme. Finalement, après avoir trouvé des paramètres et des hyperparamètres, on peut évaluer la performance de l'algorithme sur D_{test}.

Lorsque l'erreur sur l'ensemble de validation est très élevée par rapport à l'erreur sur l'ensemble d'entraînement, comme dans l'exemple où $k = n-1$, on considère que c'est du *sur-apprentissage*. Le modèle ne parvient pas à bien généraliser à des données en dehors de D_{train}. Lorsque le modèle a une erreur très élevée sur l'ensemble d'entraînement, on peut considérer que c'est du *sous-apprentissage*. L'expressivité de f peut déterminer grandement le résultat de l'apprentissage. Si la fonction f^* qu'on cherche à apprendre ne fait pas partie de la famille des modèles de l'algorithme, on obtient un *biais* et on se trouvera potentiellement dans un contexte de sous-apprentissage. Pour reprendre l'exemple de la régression polynomiale, en utilisant $k = 1$ pour modéliser une fonction hautement non-linéaire, on obtiendra une performance faible sur l'ensemble d'entraînement (voir Figure 1.1 a).

Lorsqu'on utilise des méthodes ayant une grande expressivité, plusieurs techniques sont utilisées pour éviter le sur-apprentissage. Une technique courante est la *régularisation* où il suffit d'ajouter un terme $\Omega(\boldsymbol{\theta})$ à notre objectif:

$$\hat{R}(\boldsymbol{\theta}, D) = \frac{1}{|D|} \sum_{i=1}^{|D|} L(f_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}), y^{(i)}) + \lambda \Omega(\boldsymbol{\theta}) \tag{1.1.7}$$

Ce terme de régularisation, qui dépend des paramètres $\boldsymbol{\theta}$, impose une pénalité sur la complexité de $f_{\boldsymbol{\theta}}$. L'hyperparamètre $\lambda > 0$ pondère l'influence de ce terme de régularisation. Deux types de terme de régularisation couramment utilisés sont la norme L_1 et L_2 sur les paramètres. Dans le premier cas, la régularisation induit une parcimonie (*sparsity*) des paramètres $\boldsymbol{\theta}$, tandis que dans le second cas, la régularisation contraint les valeurs des paramètres à rester faible. D'un point de vue bayésien, ce terme peut être vu comme un *a priori* sur $\boldsymbol{\theta}$. En effet, si on considère le *maximum a posteriori* au lieu du *maximum de vraisemblance*, un terme additionnel apparaît:

$$\begin{aligned} \hat{\boldsymbol{\theta}}_{MAP} &= \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta}|\boldsymbol{x}, y) \\ &= \arg\max_{\boldsymbol{\theta}} \log p(y|\boldsymbol{x}, \boldsymbol{\theta}) p(\boldsymbol{\theta}) \\ &= \arg\max_{\boldsymbol{\theta}} \log p(y|\boldsymbol{x}, \boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) \end{aligned} \tag{1.1.8}$$

Pour un *a priori* Laplacien et Gaussien, le terme additionnel $\log p(\boldsymbol{\theta})$ correspondra respectivement aux normes L_1 et L_2.

1.2. Apprentissage profond

L'apprentissage profond est un sous-domaine de l'apprentissage automatique qui se base sur les *réseaux de neurones artificiels* et la *rétropropagation du gradient*. Tel que mentionné précédemment, l'apprentissage profond a récemment connu un succès considérable et est souvent à l'état de l'art dans un vaste ensemble de domaines. Bien que l'engouement autour de ce domaine peut sembler récent, le développement de ce domaine date.

Déjà en 1943, le premier modèle de neurone artificiel a été proposé par McCulloch et Pitts (McCulloch et Pitts, 1943) . En 1958, le perceptron est inventé: un modèle constitué de neurones artificiels pouvant apprendre automatiquement (Rosenblatt, 1958). Mais sa faible expressivité est jugée comme trop limitante par plusieurs chercheurs (Marvin et Seymour, 1969). C'est finalement dans les années 80 que le domaine prend son envol avec le développement de la *rétropropagation du gradient* qui permet l'apprentissage de modèles beaucoup plus complexes que le perceptron (McClelland et al., 1986). Par la suite, le domaine a subi une certaine stagnation. Il faut attendre autour des années 2010 pour que ces modèles deviennent à nouveau très populaires. Leur récent succès peut être attribué à trois causes: l'augmentation de la puissance de calcul des ordinateurs, l'amélioration de la qualité et de la taille des jeux de données et une amélioration des algorithmes utilisés.

Une des caractéristiques avantageuses de l'apprentissage profond, c'est l'*extraction automatique de caractéristiques*. Auparavant, plusieurs méthodes d'apprentissage automatique requérait un choix méticuleux de caractéristiques. Ce choix était souvent exécuté par des experts du domaine concerné. Par exemple, dans le domaine du traitement automatique des langues naturelles, des linguistes étaient nécessaires pour identifier les caractéristiques les plus importantes d'un problème. Ces techniques sont souvent très coûteuses et généralisent mal. Une autre caractéristique avantageuse de l'apprentissage profond est, lorsqu'on considère des données complexes comme des images ou des sons, la grande expressivité des modèles. Nous présentons ici de manière formelle les réseaux de neurones et la rétropagation du gradient. Pour un traitement plus complet du domaine, voir Goodfellow et al. (2016b).

1.2.1. Réseaux de neurones

Un réseau de neurones artificiels est constitué de plusieurs *couches* qui représentent chacune des opérations mathématiques simples. Globalement, le réseau de neurones est une fonction $f : \mathcal{X} \to \mathcal{Y}$ qui est elle-même une composition de plusieurs fonctions $f^{(k)} \circ \cdots \circ f^{(2)} \circ f^{(1)}(x)$ correspondant aux différentes couches (voir Figure 1.2). Chaque couche i est paramétrée par un biais $\boldsymbol{b}_i \in \mathbb{R}^{h_{i+1}}$ et une matrice de poids $W_i \in \mathbb{R}^{h_{i+1} \times h_i}$ où h_{i+1} correspond à la dimension de la sortie de $f^{(i)}$ et h_1 la dimensionnalité de l'input $\boldsymbol{x}$. Chaque $f^{(i)}$ est une transformation linéaire suivie d'une non-linéarité ϕ_i qui est appliquée sur chaque élément, soit:

$$f^{(i)}(\boldsymbol{x}) = \phi_i(W_i \boldsymbol{x} + \boldsymbol{b}_i) \tag{1.2.1}$$

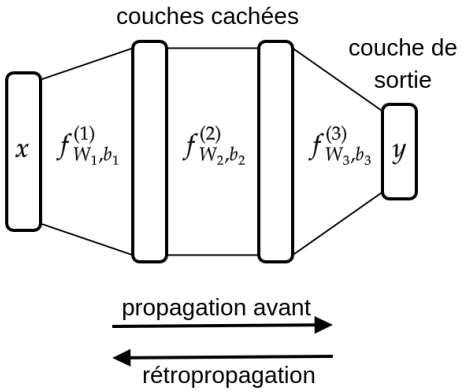


Fig. 1.2. Représentation d'un réseau de neurones artificiels à deux couches cachées. La direction de la propagation avant et de la rétropropagation y est illustrée.

On désigne souvent globalement le réseau de neurones par $f_{\boldsymbol{\theta}}$ où $\boldsymbol{\theta} = \{W_i, b_i\}_{i=1}^{k}$ est la liste de tous les paramètres pour les k couches. La dernière couche est déterminée par le type de problème considéré. Par exemple, pour une régression, la dernière couche peut être seulement une application linéaire (sans la non-linéarité ϕ). Pour un problème de classification à plusieurs classes, la fonction *softmax* σ est utilisée pour retourner des valeurs semblables à des probabilités (soit dans l'intervalle [0,1] et sommant à 1). La fonction softmax est définie comme suit sur un vecteur $\boldsymbol{z}$:

$$\sigma(z_1, \ldots, z_d) = \frac{1}{Z}(e^{z_1}, \ldots, e^{z_d}) \quad \text{où } Z = \sum_{i=1}^{d} e^{z_i} \tag{1.2.2}$$

Finalement, on applique la fonction de coût à cette sortie.

Le réseau de neurones précédemment décrit est nommé *perceptron multicouches*. On nomme *couche cachée*, les couches entre l'entrée et la couche de sortie. Le *perceptron*, discuté en introduction, est un modèle sans couche cachée qui manque d'expressivité: il ne peut même pas modéliser une fonction OU-exclusif. Avec une seule couche cachée, la situation est très différente: le réseau de neurones est un approximateur universel (Hornik, 1991), c'est-à-dire qu'il peut modéliser n'importe quelle fonction en autant que le nombre de neurones de sa couche cachée soit suffisant. Malgré ce résultat, il est souvent intéressant d'utiliser des réseaux de neurones dits profonds, c'est-à-dire ayant un grand nombre de couches cachées. En effet, d'une part, comparativement aux réseaux à une couche cachée le nombre total de neurones requis pour approximer une certaine fonction sera plus petit (Montufar et al., 2014) et, d'autre part, une meilleure généralisation a été observée sur un ensemble considérable de tâches (Goodfellow et al., 2016b).

Il est possible de composer les fonctions différemment et de partager des paramètres entre différentes couches. Plusieurs familles d'architectures spécialisées à différentes applications ont justement été proposées. Par exemple, pour des données spatiales, comme des images,

les *réseaux de neurones convolutionnels*, qui utilisent la convolution comme fonction de base, sont particulièrement intéressants. Afin d'être invariants aux translations, ils utilisent un certain partage de paramètres et une opération de *pooling* où les activations d'une certaine région sont résumées par une statistique (p. ex. le maximum). Les *réseaux de neurones récurrents* sont particulièrement appropriés pour les données séquentielles. L'astuce est de réutiliser les mêmes fonctions pour les différentes unités d'une séquence. Bien qu'avec un très grand nombre d'exemples, un réseau de neurones classique serait en mesure d'apprendre les structures sous-jacentes des données, utiliser un modèle qui a de bons *a priori* permet d'obtenir de bon résultats avec un nombre plus faible d'exemples.

La grande capacité des réseaux de neurones peut être un couteau à double tranchant: le sur-apprentissage survient souvent, et ce, même en utilisant un modèle ayant de bons *a priori*. Il est possible d'utiliser les mêmes techniques que pour l'apprentissage automatique comme une régularisation L_1 ou L_2 sur les paramètres. Plusieurs autres techniques sont particulièrement utilisées dans le domaine de l'apprentissage profond dont l'augmentation du jeu de données (*dataset augmentation*), le *dropout* (Srivastava et al., 2014) et l'arrêt précoce (*early stopping*, dont nous reparlerons dans la Section 1.2.2). Voyons plus en détail l'augmentation du jeu de données. Puisque le sur-apprentissage survient souvent lorsque le nombre de données est insuffisant par rapport à la capacité du modèle, il est possible d'augmenter un jeu de données en créant des exemples transformés. Par exemple, pour des images, nous aimerions que notre modèle soit invariant à de petites transformations qui sont naturelles (rotation, translation, bruit, etc). En créant soi-même de nouvelles données en appliquant ces transformation, on observe souvent une meilleure généralisation (Shorten et Khoshgoftaar, 2019).

1.2.2. Optimisation et descente de gradient

Comme présenté dans la section précédente, l'apprentissage, c'est-à-dire trouver de bonnes valeurs de paramètres, peut être posé comme un problème d'optimisation. En apprentissage profond, la technique d'optimisation la plus utilisée est la *descente de gradient*. Les réseaux de neurones présentent des objectifs non-convexes dû à leurs non-linéarités. Contrairement à des modèles ayant des objectifs convexes, il n'est pas possible de trouver directement un minimum en calculant le gradient. Cependant, en se déplaçant itérativement en direction inverse du plus grand gradient, on peut minimiser l'objectif. Soit:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \nabla_{\boldsymbol{\theta}_t} \hat{R}(\boldsymbol{\theta}_t, D) \tag{1.2.3}$$

où $\eta > 0$ est le *taux d'apprentissage*. Donc, à chaque itération, les valeurs des paramètres $\boldsymbol{\theta}$ sont ajustées en effectuant un "pas" dans la direction opposée au gradient et en pondérant ce pas par le taux d'apprentissage η.

Comme l'objectif est non-convexe, il est probable que cette méthode trouve un minimum local. En pratique, cela ne semble pas être un problème majeur et plusieurs travaux semblent montrer que cela pourrait être expliqué du fait qu'en haute dimension (et sous certaines conditions) la plupart des minimums locaux ont des valeurs proches au minimum global (Sagun et al., 2014; Kawaguchi, 2016). De plus, en pratique, on n'optimisera pas jusqu'à ce que $\boldsymbol{\theta}$ atteigne un minimum, mais on arrêtera lorsque l'erreur de validation ne diminue plus. Cette technique d'arrêt précoce (*early stopping*) est une forme de régularisation, car elle limite les valeurs de $\boldsymbol{\theta}$ qui seront explorés. De plus, le succès des techniques de descente de gradient dépend du point de départ des paramètres. Il est donc important de bien initialiser les paramètres des réseaux de neurones. Plusieurs méthodes d'initialisations ont fait leur preuve et sont couramment utilisées (dont l'initialisation Glorot (Glorot et Bengio, 2010a)).

Il arrive qu'il soit nécessaire d'utiliser des méthodes additionnelles pour certains problèmes d'optimisation spécifiques. Dans les deux approches proposées dans les articles de ce mémoire, on souhaite résoudre un problème d'optimisation sous une contrainte d'égalité. En général, on aura:

$$\min_x f(x) \quad s.t.\ g(x) = 0 \tag{1.2.4}$$

Une méthode simple dans ce cas est le multiplicateur de Lagrange (pour un traitement plus général supportant aussi les contraintes d'inégalités, voir l'approche Karush-Kuhn-Tucker). Par cette méthode, en ajoutant $\lambda g(x)$ à l'objectif, il est possible de transformer le problème d'optimisation sous contrainte comme un problème d'optimisation régulier, soit:

$$\min_x f(x) + \lambda g(x) \tag{1.2.5}$$

Dans les deux approches proposées, une méthode similaire est utilisée: la méthode du Lagrangien augmenté. En plus du multiplicateur de Lagrange, l'objectif est augmenté d'un terme additionnel: $\frac{\mu_k}{2} g(x)^2$. et il est nécessaire de résoudre une série de problème d'optimisation en utilisant une valeur croissante de μ_k.

1.2.3. Descente de gradient stochastique et variantes

En pratique, il n'est souvent pas possible d'appliquer directement la descente de gradient tel que présenté. En effet, les jeux de données immenses n'entrent pas en mémoire de l'ordinateur et le calcul du gradient peut être long. Il est possible d'approximer le gradient en utilisant seulement un petit sous-ensemble du jeu de données (une *batch*) et ainsi d'éviter ces problèmes computationnels. À chaque itération, on piochera une batch d'exemples sur laquelle on calculera une version bruitée du gradient.

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \nabla_{\boldsymbol{\theta}_t} \sum_{i \in B} L(f_{\boldsymbol{\theta}_t}(\boldsymbol{x}^{(i)}), y^{(i)}) \tag{1.2.6}$$

où B est un ensemble d'indices piochés uniformément dans l'ensemble des indices des exemples. Cette estimation du gradient n'est pas biaisée. Avec cette méthode, il est cependant nécessaire de réduire graduellement le taux d'apprentissage pour converger. Pour accélérer la convergence, plusieurs techniques ont été proposées. Une idée intuitive est l'inertie (*momentum*). Par analogie, on peut voir le gradient comme la vitesse d'une particule. En accumulant la vitesse, on peut calculer l'inertie de la particule. Cette inertie peut accélérer la convergence, surtout dans les cas où le gradient est petit ou très bruité. Une autre approche intéressante provient de la méthode de Newton qui, en prenant en compte la courbature de l'espace, a un taux de convergence rapide. Cependant, elle n'est pas utilisable en pratique de par la complexité du calcul de la hessienne. Néanmoins, certaines méthodes vont approximer cette hessienne. Finalement, RMSProp (Tieleman et Hinton, 2012b) et Adam (Kingma et Ba, 2014) sont deux méthodes très populaires qui utilisent une approximation (biasiée) du second moment. Comme il arrive souvent que seulement certaines directions dans l'espace de paramètres sont sensibles, ces deux méthodes ont des taux d'apprentissage adaptatifs pour chaque paramètre.

1.2.4. Rétropropagation du gradient

Comment calculer efficacement le gradient de la fonction de coût par rapport aux différents paramètres? En effet, calculer le gradient d'une fonction complexe peut être coûteux. Cependant, en tirant profit de leur structure, il est possible de rendre ce calcul plus simple et moins coûteux. La clé de la solution est la dérivation en chaîne. Comme rappel, la règle de la dérivation en chaîne postule que si z est le résultat d'une composition de fonctions, il est possible de calculer la dérivée en la décomposant par étape. Soit $z = f_2(f_1(x)) = f_2(y)$ alors:

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx} \tag{1.2.7}$$

Précédemment, nous avons vu comment à partir d'une entrée, la sortie d'un réseau de neurone est calculée en appliquant successivement les fonctions $f^{(i)}$ associées à chaque couche. Cette propagation de l'information de l'entrée jusqu'au calcul de la fonction de coût L est nommé propagation avant (*forward propagation*, voir Figure 1.2). Dans le sens inverse, le calcul du gradient $\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$, de la fonction de coût aux différents paramètres $\{W_i, b_i\}_{i=1}^{k}$, la propagation est nommé rétropropagation (*backpropagation*). En décomposant la dérivation de L couche par couche par la règle de la dérivation en chaîne et en gardant en mémoire les différentes dérivées, on peut éviter de recalculer plusieurs fois les mêmes dérivées. En pratique, les différentes librairies d'apprentissage profond (dont Pytorch (Paszke et al., 2017) et TensorFlow (Abadi et al., 2015)) implémentent cette méthode et évitent à l'utilisateur de calculer manuellement les dérivées en utilisant des techniques de différentiation automatique.

Chapitre 2

Causalité

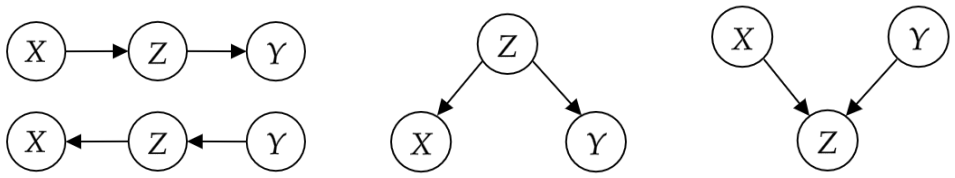
Le concept de causalité est au coeur de la science et des stratégies de prise de décision. On cherche à identifier des associations entre différents concepts, mais aussi à identifier les concepts qui sont des causes. En identifiant les liens causaux, on peut être en mesure d'évaluer l'impact qu'aura un changement sur d'autres variables (ex. l'impact de la prise d'un médicament sur la santé d'un patient). Une approche purement statistique - on pourrait dire qui s'intéresse seulement à déterminer les associations entre les variables - est insuffisante pour répondre à ce genre de question. De même, une approche statistique peut parfois donner des résultats qui, sans interprétation causale, contredisent complètement notre intuition (par exemple, le paradoxe de Simpson (Simpson, 1951)).

De manière surprenante, ce n'est que récemment qu'un effort soutenu a été déployé pour faire le pont entre le domaine de la causalité et de l'apprentissage automatique (Schölkopf, 2019). Pourtant, avoir un raisonnement causal est une capacité fondamentale chez l'humain. Elle se développe en bas âge (Walker et Gopnik, 2013) et est à la base de nos processus d'induction et de planification. Les contrefactuelles, où l'on imagine des mondes possibles sur la base d'évènements qui ne se sont pas produits, sont très communes dans nos raisonnements et sont le résultat d'une modélisation riche de notre environnement qui prend en considération des liens causaux. Il semble évident que ces capacités sont recherchées pour concevoir des systèmes intelligents. La hiérarchie de la causalité de Pearl (Pearl, 2019b) introduit trois niveaux de type de raisonnement: associatif, interventionnel et contrefactuel. Au niveau associatif, on s'intéresse à des questions du genre "comment observer X change ma croyance par rapport à Y ?". Au niveau supérieur dit interventionnel, on répond à des questionnements du genre "quelles sont les conséquences de faire X ?". Finalement, au niveau contrefactuel, on répond à question sur des évènements potentiels, du genre "que serait-il arrivé si j'avais fait X au lieu de Y ?". Les modèles d'apprentissage automatique actuels se limitent principalement au niveau associatif.

Pour illustrer la limitation de ce niveau de raisonnement, voyons un exemple concret de problème qui peut survenir avec des modèles d'apprentissage automatique. Supposons qu'on entraîne un classifieur à catégoriser des images sur la base du type d'animal présenté. Le jeu de données utilisé pour l'entraînement contient plusieurs images de vaches dans un pré. Suite à son entraînement, le classifieur obtient un taux d'erreur faible sur les images de vaches. Lorsque déployé, on soumet au classifieur des images de vaches qui se trouve sur la plage. Soudainement, le classifieur a un taux d'erreur élevé: il ne reconnaît plus les vaches! Cet exemple réel (Beery et al., 2018) semble indiquer que le classifieur n'a pas réellement capturé les propriétés intrinsèques d'une vache, mais a plutôt basé ses prédictions sur le décor (le pré). En se basant sur des associations fortuites, ce classifieur n'est pas robuste aux changements. Or, on souhaite en général qu'un classifieur apprenne des propriétés des objets qui sont invariants à des changements d'environnement. Encore une fois, une des solutions semblent résider dans l'application de concepts causaux. Déjà, l'application de certains de ces concepts a permis d'avoir des modèles d'apprentissage automatique plus robustes aux changements d'environnement (Arjovsky et al., 2019). De plus, les modèles d'apprentissage profond ont souvent le défaut d'être des boîtes noires, c'est-à-dire qu'il est difficile d'interpréter leur décision. Encore une fois, la causalité semble offrir une solution à ce problème (Pearl, 2019b). En résumé, on peut motiver l'intérêt de la causalité en apprentissage automatique (et profond) à un niveau pratique de bien des manières: meilleure compréhension des données étudiées, meilleure interprétabilité et robustesse accrue des modèles.

Dans les articles de ce mémoire, nous nous intéressons plus particulièrement à l'apprentissage de modèles causaux. À partir de données, on veut apprendre un modèle qui permettra de faire des prédictions sur des changements dans un système. Ce type de modèle peut avoir des applications dans un vaste ensemble de domaines scientifiques. Il contient les trois niveaux de raisonnement de Pearl et permet donc de répondre à des questions contrefactuelles.

La causalité est au coeur de recherche remontant au moins depuis Aristote (Falcon, 2006) et a donné lieu à de multiples définitions et débats philosophiques. Dans le cadre de ce mémoire, par causalité nous entendons spécifiquement la causalité tel que définie par Pearl (Pearl, 2009b). Dans la première partie de ce chapitre, nous verrons comment définir plus formellement la causalité à partir des graphes causaux. Par la suite, nous verrons comment apprendre des graphes causaux à partir de données et les problèmes qui y sont rattachés. Pour un traitement plus poussé de la causalité, voir Pearl (2009b) et Peters et al. (2017b).



(a) Chaînes (b) Fourche (c) V-structure

Fig. 2.1. Représentation de graphes formés de 3 noeuds. En a), deux chaînes, en b) et en c) une v-structure.

2.1. Modèles causaux

2.1.1. Réseau bayésien

Avant de considérer les modèles causaux, commençons par une brève présentation de concept fondamentaux provenant des modèles graphiques probabilistes. Un réseau bayésien est constitué d'un graphe orienté acyclique (DAG) $G = (V, E)$ qui va définir une certaine factorisation d'une distribution P_X sur un vecteur $X = (X_1, \ldots, X_d)$. Chaque variable aléatoire X_j est associé à un noeud $j \in V = \{1, 2, \ldots, d\}$ et les liens entre les noeuds représentent une dépendance entre les variables. De plus, la distribution P est *Markov* par rapport au graphe G, c'est-à-dire qu'elle se factorise ainsi:

$$p(x_1, \ldots, x_d) = \prod_{j=1}^{d} p_j(x_j | x_{\pi_j^G}) \tag{2.1.1}$$

où π_j^G représente l'ensemble des noeuds parents à X_j dans le graphe G et lorsque x est indicé par un ensemble, cela dénote les différents entrées x_i où $i \in B$.

À partir de réseau bayésien, il est possible de faire de l'inférence de manière plus efficace en considérant la bonne décomposition de la distribution jointe, c'est-à-dire répondre à des questions du genre quelle est la probabilité que $X_i = x_i$ étant donné que $X_j = x_j$. À partir du graphe, il est aussi possible de lire des indépendances conditionnelles en utilisant le critère de *d-séparation*. Avant d'expliquer ce critère, présentons 3 types de graphes élémentaires qui constituent un réseau bayésien (voir Fig. 2.1): chaîne, fourche et v-structure. On dit que des noeuds forment une immoralité lorsque qu'un noeud a des parents qui ne sont pas liés (comme dans la v-structure de la Fig. 2.1 (c)). Succinctement, le critère de d-séparation indique que si deux ensembles de variables A et B sont d-séparés par un ensemble C, c'est-à-dire si tous les chemins sont *bloqués* par C, alors $A \perp\!\!\!\perp B | C$. Un chemin entre deux noeuds a et b est dit bloqué si un noeud de ce chemin fait partie de l'ensemble sur lequel on conditionne (C) et que ce noeud ne fait pas partie d'une immoralité. Un chemin est aussi bloqué si un certain noeud du chemin est un noeud au milieu d'une immoralité ne faisant pas partie de C et que aucun des descendants de ce noeud ne fait partie de C. En Fig. 2.1 (a) et (b), on peut voir

que les variables X et Y sont d-séparés par Z. On peut donc dire que ces graphes encodent l'indépendance inconditionnelle suivante: $X \perp\!\!\!\perp Y|Z$. Pour la v-structure en Fig. 2.1 (c), le chemin entre X et Y est bloqué, car Z fait partie d'une immoralité. L'indépendance encodée est donc $X \perp\!\!\!\perp Y$. Une dernière notion importante est la *fidélité* (*faithfulness*). Une distribution est dite fidèle à un graphe si toutes indépendances conditionnelles implique une d-séparation équivalente dans le graphe. Pour une présentation plus détaillée des réseaux bayésiens, voir Koller et Friedman (2009).

2.1.2. Modèle graphique causal

Similairement aux réseaux bayésiens, les modèles graphiques causaux comprennent une distribution P_X sur un vecteur $X = (X_1, \ldots, X_d)$ qui est Markov à un DAG G. Chaque variable aléatoire X_j est associée à un noeud, mais dans ce cas-ci, les liens entre les noeuds représentent un lien causal entre les variables. Bien que semblable aux réseaux bayésiens, les modèles graphiques causaux se distinguent lorsque des interventions sont appliquées sur les différentes variables. Une intervention est l'assignation d'une variable à une certaine valeur ou densité comme lors d'une expérience scientifique où l'on force la valeur d'une certaine variable d'un système.

De manière générale, si on intervient sur la variable X_i, cela est équivalent à remplacer la conditionnelle $p_i(x_i|x_{\pi_i^G})$ par une nouvelle conditionnelle $\tilde{p}_i(x_i|x_{\pi_i^G})$. En d'autres termes, la distribution jointe interventionnelle sera donnée par:

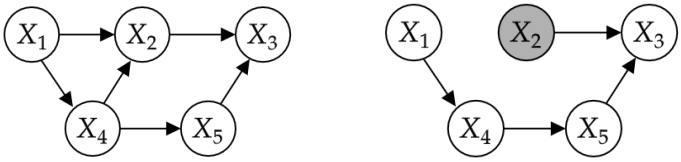
$$p^{do(X_i)}(x) = \prod_{j \neq i} p_j(x_j|x_{\pi_j^G})\tilde{p}_i(x_i|x_{\pi_i^G}) \tag{2.1.2}$$

où $do(X_i)$ indique qu'une intervention a été appliquée sur la variable X_i. Il est aussi possible de faire une intervention sur plusieurs variables à la fois. On désignera par un ensemble I les variables sur lesquels on intervient et par $p^{do(X_I)}$ la distribution induite. On peut constater que le changement de ce genre d'intervention est local: seulement les conditionnelles sur lesquelles l'intervention a eu lieu auront changées.

Un cas particulier d'interventions qui est souvent considéré est l'intervention dite *parfaite*. Pour ce type d'intervention, on remplace la conditionnelle par une marginale. En forçant ainsi une variable à une certaine valeur, celle-ci devient indépendante de ses noeuds parents. La Figure 2.2 représente le graphe causal d'un modèle causal et le graphe causal induit par une intervention parfaite où les liens des parents des noeuds ayant subis des intervention ont été retirés. La distribution jointe interventionnelle induite par une intervention sur I sera:

$$p^{do(X_I)}(x) = \prod_{j \notin I} p_j(x_j|x_{\pi_j^G}) \prod_{j \in I} \tilde{p}_j(x_j) \tag{2.1.3}$$

où $\tilde{p}_j$ désigne les nouvelles densités assignées par intervention. Il est à noter qu'en général $p(x|y) \neq p^{do(Y=y)}(x)$: intervenir est un concept différent de conditionner.



(a) DAG G (b) DAG G' avec intervention

Fig. 2.2. Représentation d'un graphe G associé à un certain modèle causal et d'un graphe G' induit par une intervention parfaite dans le même modèle sur la variable X_2.

2.1.3. Modèle causal structurel

Pour illustrer plusieurs concepts, présentons une autre manière de définir un modèle causal: les modèles causaux structurels. Chaque noeud X_i est une fonction déterministe de ses parents et d'un certain bruit:

$$X_i := f_i(X_{\pi_i^G}, N_i) \tag{2.1.4}$$

où les bruits N_i sont mutuellement indépendants. Les fonctions f_i vont engendrer une factorisation de la distribution comme pour le modèle graphique causal. Il est à noter que ce type de modèle contient strictement plus d'informations (entre autre, il fixe les contrefactuelles que nous présenterons sous peu).

Considérons un exemple simple de modèle causal structurel constitué de seulement deux variables aléatoires, X et Y, qui sont mutuellement dépendantes. Supposons que les fonctions qui lient les variables sont:

$$\begin{aligned} X &:= N_X \\ Y &:= 10X + N_Y \end{aligned} \tag{2.1.5}$$

et donc que le lien causal est de X vers Y, soit $X \to Y$, et que $N_X, N_Y \sim N(0,1)$. Si on utilisait un réseau bayésien pour représenter la décomposition de cette distribution jointe, on pourrait tout aussi bien représenter $p(X,Y)$ par $p(Y|X)p(X)$ que par $p(X|Y)p(Y)$. Cependant, la décomposition causale adéquate est $p(X,Y) = p(Y|X)p(X)$: la direction est importante. L'application d'interventions rend évident cette asymétrie. En effet, en intervenant sur X (en fixant sa valeur à 2), la cause, on aura aussi un impact sur l'effet Y:

$$\begin{aligned} X &:= 2 \\ Y &:= 20 + N_Y = N(20,1) \end{aligned} \tag{2.1.6}$$

Si au contraire l'intervention est faite sur Y, l'effet, il n'y aura pas d'impact sur la cause X:

$$X := N_X = N(0,1) \tag{2.1.7}$$
$$Y := 2$$

On peut constater que les modèles graphiques causaux capturent l'asymétrie intuitive entre la cause et l'effet.

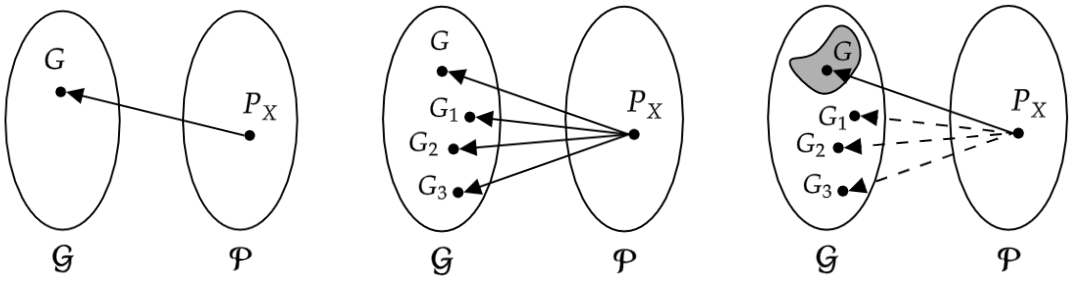
Bien que l'intervention induise une nouvelle distribution, cette distribution a des propriétés semblables à la distribution d'origine que la modélisation par un modèle graphique causal prend en compte. En effet, toutes les conditionnelles n'ayant pas subi d'intervention restent les même dans la distribution interventionnelle. Cette indépendance des différents mécanismes causaux est particulièrement intéressante en apprentissage automatique. Si un certain changement de contexte est équivalent à une intervention dans un graphe causal, alors une méthode qui aura modélisé indépendemment les mécanismes pourra s'adapter plus facilement à ce nouvel environnement, car certains mécanismes auront déjà été appris. De même, en connaissant le graphe causal et l'intervention appliquée, il est possible de connaître la distribution interventionnelle sans l'observer.

Un autre concept intéressant découlant d'une modélisation causale est la contrefactuelle. On peut définir formellement des questions du genre: "Dans le contexte où on observe $X = x$ et $Y = y$, quel aurait été la probabilité d'observer $Y = y$ si X aurait plutôt été égal à x'". Cela équivaut à conditionner les variables aux valeurs observées et à faire une intervention, soit $p^{do(X=x')}(Y = y|X = x, Y = y)$. Dans le contexte de l'intelligence artificielle, le concept de contrefactuelle est très intéressant puisqu'il correspond à évaluer les conséquences d'une action sans avoir à effectuer ladite action.

En pratique, il arrive souvent qu'on observe seulement un sous-ensemble des variables d'un graphe causal. À ce moment, il est tout de même possible d'inférer certaines valeurs en contrôlant les variables non-observées. Un algorithme (Tian et Pearl, 2002) basé sur le *do-calculus* (Pearl, 1995), constitué de trois règles simples, permettent d'inférer ces valeurs lorsque certaines hypothèses sont respectées.

2.2. Apprentissage de modèles causaux

Souvent, un graphe causal sera déterminé par des experts du domaine concerné. Cependant, comme mentionné dans le cas de l'apprentissage profond, il peut être pertinent de l'apprendre à partir de données de manière automatique. Ce problème peut être très difficile. Pour le reste de ce mémoire, nous considérons un cas simplifié par rapport aux applications réelles. Nous considérons que toutes les variables d'intérêt sont observées et que chacune de ces variables correspond exactement à un noeud (il n'y a pas d'apprentissage de



(a) Cas identifiable (b) Cas non-identifiable (c) Cas identifiable avec $\mathcal{A}$

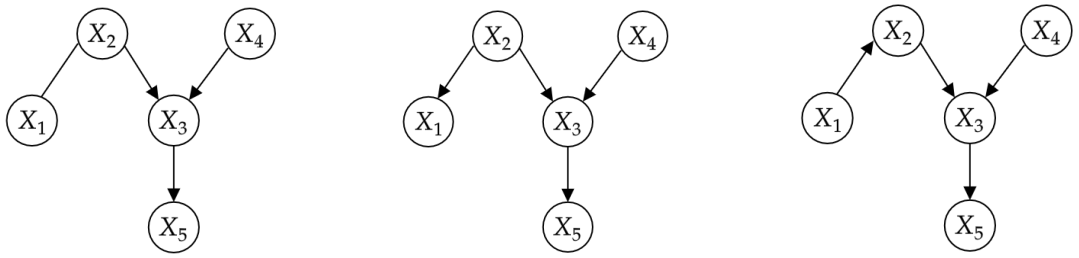
Fig. 2.3. Chaque image représente $\mathcal{G}$ l'ensemble des graphes et $\mathcal{P}$ l'ensemble des distributions sur X. G représente le graphe qui a été utilisé pour générer P_X et les liens orientés illustre que P_X est Markov à certains graphes. La zone ombragée en c) représente le sous-ensemble de $\mathcal{G}$ qui respecte une certaine supposition sur le modèle.

représentation nécessaire). Dans la section suivante, nous verrons tout d'abord les problèmes d'identifiabilité: dans certaines situations, il n'est tout simplement pas possible de trouver le graphe causal. Dans la Section 2.2.2, nous présenterons les deux principaux types de méthodes permettant d'estimer le graphe causal ainsi qu'un algorithme pour chaque type.

2.2.1. Problème d'identifiabilité

Avant même de présenter comment apprendre un graphe causal, il est nécessaire de vérifier si cette tâche est possible. Considérons tout d'abord le cas purement observationnel, c'est-à-dire sans intervention. Supposons que nous ayons encore une fois deux variables aléatoires X et Y qui sont dépendantes. Nous voulons déterminer le bon graphe causal, soit $X \to Y$ ou $X \leftarrow Y$. En général, il ne sera pas possible de le faire. En effet, posons f comme une fonction bijective sur X et Y. À partir de données purement observationnelles, les données observées peuvent autant être expliqué par $f : \mathcal{X} \to \mathcal{Y}$ que par $g = f^{-1} : \mathcal{Y} \to \mathcal{X}$. On peut illustrer le problème d'identifiabilité avec la Figure 2.3. Dans la Figure 2.3 a), on peut voir un graphe identifiable: un seul graphe est Markov à la distribution observée. Dans la partie du milieu, plusieurs graphes sont Markov à la distribution: il est impossible de savoir celui qui est à l'origine des données.

En posant des conditions, par exemple sur le type de fonctions ou de bruit du modèle génératif, il est possible de restreindre l'espace des graphes potentiels et donc d'obtenir un graphe identifiable tel qu'illustré dans la Figure 2.3 c). Pour illustrer intuitivement comment des conditions du modèle peuvent impacter l'identifiabilité, voyons deux exemples simples. Pour commencer, considérons des modèles ayant une fonction non-injective avec du bruit additif unimodal. Par exemple, $X = N_X$ et $Y = f(X) + N_Y$ où f est non-injectif. Alors, il est évident qu'il est seulement possible de modéliser adéquatement dans la direction $X \to Y$ avec un modèle qui suppose l'unimodalité. En effet, il est impossible de trouver un



(a) CPDAG (b) DAG 1 (c) DAG 2

Fig. 2.4. En a), un CPDAG représentant une classe d'équivalence de Markov. En b) et c), les deux DAGs appartenant à cette classe d'équivalence. Exemple adapté de (Chickering, 2002).

modèle respectant les conditions d'unimodalité qui peut modéliser adéquatement $Y \to X$ qui sera multimodal. D'un autre côté, même en considérant seulement certains types de modèles, il peut être impossible de trouver le graphe d'origine si ces conditions ne sont pas suffisamment contraignantes. Par exemple, supposons que nous sachions que le modèle est linéaire avec bruit gaussien. Pour reprendre un exemple similaire à celui de la section précédente: $X = N(0,1)$ et $Y = X + N(0,1) = N(0,2)$. Le modèle inverse est, lui aussi, un modèle linéaire avec bruit gaussien, soit $X = \frac{1}{\sqrt{2}}(\frac{Y}{\sqrt{2}} + N(0,1))$ et $Y = N(0,2)$. Il est impossible de déterminer lequel de ces deux modèles a généré les données.

Plusieurs résultats d'identifiabilité ont été découverts et rapportés dans la littérature. Par exemple, les modèles linéaires ayant un bruit non-gaussien (Shimizu et al., 2006b), les modèles linéaires avec la même variance pour chaque variable (Peters et Bühlmann, 2014) et les modèles non-linéaires à bruit additifs (Peters et al., 2014b) sont tous identifiables. Un modèle non-linéaire à bruit additif est défini comme suit: $X_i := f_i(X_{\pi_i^G}) + N_i$ où f_i sont des fonctions non-linéaires. Cependant, en pratique, on connaît rarement le modèle génératif à la source des données et on ne peut utiliser ces résultats d'identifiabilité. Dans tous les cas, même s'il n'est pas possible d'identifier le graphe d'origine, on peut tout de même identifier un ensemble de graphes. Cet ensemble de graphes est représenté par une classe d'équivalence dite de Markov où deux graphes sont équivalents ssi ils ont le même *squelette* et les mêmes immoralités (Verma et Pearl, 1991). Le squelette d'un DAG est le graphe engendré par ce DAG en retirant les directions des liens. Cette classe d'équivalence est identifiable en autant que la distribution soit fidèle au DAG et correspond à trouver les indépendances conditionnelles. Pour représenter graphiquement une classe d'équivalence, un graphe complété partiellement orienté acyclique (CPDAG) est utilisé. La Figure 2.4 présente une classe d'équivalence représenté par un CPDAG et les deux DAGs appartenant à cette classe. On peut constater que le CPDAG a un lien non-orienté et que le lien entre X_3 et X_5 est orienté, car sinon une nouvelle immoralité serait introduite.

Lorsqu'on a accès à des données provenant des distributions interventionnelles, la situation est différente. On nomme classe d'équivalence interventionnelle de Markov (Hauser et Bühlmann, 2012; Solus et al., 2017), l'ensemble des graphes respectant les indépendances conditionnelles des différentes distributions interventionnelles. L'ensemble des graphes de cette classe est un sous-ensemble de ceux de la classe d'équivalence de Markov. En d'autres termes, en général, le fait d'avoir des données interventionnelles aide à l'identifiabilité. Si des interventions sont faites sur chaque noeud individuellement, le graphe devient identifiable. En pratique, le nombre d'interventions nécessaires est souvent inférieur au nombre total de noeuds (Eberhardt, 2012; Eberhardt et al., 2012).

Pour donner un exemple simple (tiré de Hauser et Bühlmann (2012)), supposons que l'on observe la classe d'équivalence représentée par le CPDAG suivant $X_1 - X_2 - \cdots - X_d$ où les tirets représentent des liens non-dirigés. L'ensemble des graphes causaux possibles, qui encode le même ensemble d'indépendances conditionnelles, peut être résumé par:

$$X_1 \leftarrow X_2 \leftarrow \cdots \leftarrow X_{i-1} \leftarrow X_i \rightarrow X_{i+1} \rightarrow \cdots \rightarrow X_d$$

où X_i est la source d'une fourche. En effet, les autres graphes qui inclurait des immoralités ne font pas partie de la classe d'équivalence observée. Comme la source peut se trouver sur n'importe quel des d noeuds, il y a un total de d graphes équivalents. À partir de seulement une seule intervention, le graphe peut devenir identifiable. En effet, si on intervient sur la source X_i (par chance), on n'observera pas de changement dans les dépendances de X_i avec les différents X_j et on pourra conclure que X_i est la source. En général, en intervenant sur le noeud du milieu, on peut déterminer la moitié de l'orientation des liens. Il est clair qu'il est avantageux d'utiliser des données interventionnelles en plus de données observationnelles. Cependant, en pratique, il est peut être coûteux ou même impossible d'effectuer certaines interventions.

2.2.2. Algorithmes d'apprentissage de graphes causaux

Même si un DAG est identifiable, il peut être difficile de l'apprendre à partir d'un échantillon fini. En effet, le nombre de DAGs possibles croît de manière super-exponentielle par rapport au nombre de noeuds d, soit $O(d!2^{d^2})$. Pour d = 1,2,3,4,5 il y a respectivement $1, 3, 25, 543, 29281$ DAGs possibles [1].

On peut diviser en deux grandes catégories les méthodes de découverte de graphes causaux: les méthodes basées sur les contraintes (*constraint-based*) et celles basées sur le score (*score-based*). Les méthodes basées sur les contraintes utilisent des tests d'indépendance conditionnelle. En assumant que la condition de fidélité est respectée, il est ainsi possible de trouver la classe d'équivalence de Markov. Voyons un exemple de méthode basée sur les

[1]Pour davantage d'information sur cette suite, voir https://oeis.org/A003024

contraintes nommé PC (le nom de la méthode provenant des prénoms des auteurs Peter et Clark) (Spirtes et al., 2000b). Afin d'expliquer cette méthode, présentons d'abord un lemme important de Verma et Pearl (1991): deux noeuds X et Y du graphe G sont i) adjacents ssi ils ne peuvent pas être d-séparé par aucun sous-ensemble $S \subseteq V \setminus \{X, Y\}$ et ii) si au contraire ils sont non-adjacents, alors ils sont d-séparés soit par les parents de X, soit par ceux de Y. En utilisant ce lemme, il est théoriquement facile de trouver le squelette du graphe: pour chaque paire (X, Y), on peut déterminer si les noeuds sont adjacents en testant si $X \perp\!\!\!\perp Y|S$ pour tous $S \subseteq V \setminus \{X, Y\}$. Naturellement, cette recherche est coûteuse et implique des tests d'indépendance conditionnelle qui peuvent être problématique avec un nombre d'échantillon faible surtout lorsque le nombre de variables est élevé (Shah et Peters, 2018; Zhang et al., 2011). L'ingéniosité de PC est de d'abord tester si $X \perp\!\!\!\perp Y$ et ensuite d'augmenter graduellement la taille de l'ensemble de conditionnement S et de seulement y inclure des noeuds parents de X ou de Y. En pratique, cette idée améliore souvent le coût computationnel, surtout lorsque le graphe est parcimonieux. Finalement, à partir de l'ensemble des indépendances conditionnelles, l'algorithme oriente les liens qui font partie d'immoralités.

Les méthodes basées sur le score utilisent un certain score $S(G, D)$. Le principe est de simplement trouver le graphe qui maximise ce score, soit:

$$G^* = \arg\max_{G \in \mathcal{G}} S(G, D) \tag{2.2.1}$$

Tel que mentionné auparavant, comme l'espace des DAGs $\mathcal{G}$ est immense, il n'est pas possible de calculer le score pour chaque DAG de cet espace et de choisir celui avec le score le plus élevé. Plusieurs méthodes utilisent donc des heuristiques de recherche voraces. Pour illustrer les méthodes basées sur le score, voyons le cas classique de la méthode *Greedy Equivalence Search* (GES) (Chickering, 2002). GES utilise un modèle linéaire avec bruit gaussien et utilise par défaut le *critère d'information bayésien* comme score (basé sur la log-vraisemblance pénalisé par le nombre de paramètres). La méthode assume la fidélité et tente de retourner la classe d'équivalence de Markov. Le principe de base est d'ajouter ou retirer des liens un à un au lieu de tester toutes les combinaisons possibles. Même qu'au lieu de se promener d'un DAG à l'autre, la méthode est plus efficace en cherchant l'espace des classes d'équivalences. Plus en détail: GES commence avec un CPDAG sans aucun lien et calcule le score de ce graphe. Ensuite, on considère toutes les classes d'équivalence possibles d'atteindre en ajoutant un nouveau lien et on calcule leur score. On conserve le DAG ayant le plus haut score et on tente d'ajouter un autre lien. Recalculer les nouveaux scores n'est pas si coûteux puisqu'on utilise habituellement un score qui est décomposable, c'est-à-dire

qu'on peut calculer séparément une partie du score à partir d'un noeud et de ses parents:

$$S(G, D) = \sum_{i=1}^{d} s(X_i, X_{\pi_i^G}) \tag{2.2.2}$$

où s correspond au score d'une conditionnelle. En d'autres termes, il est seulement nécessaire de recalculer le score des conditionnelles où l'ajout d'un lien change l'ensemble de parents. L'ajout est répété jusqu'à ce qu'il ne soit plus possible de trouver un CPDAG avec un score plus élevé que le précédent. Après la phase d'ajout, on procède à une phase de retrait. On teste tous les classes d'équivalence atteignables en retirant un lien et on conserve celle ayant le score le plus élevé. On répète l'opération jusqu'à convergence et, finalement, on retourne le CPDAG trouvé. Naturellement, puisque la méthode utilise une heuristique vorace, il est possible que le CPDAG retourné corresponde à un maximum local.

Une variante de GES qui permet de gérer le cas des données interventionnelles est *Greedy Interventional Equivalence Search* (GIES) (Hauser et Bühlmann, 2012). Pour chaque exemple, on fournit à l'algorithme la cible des interventions. Le principe reste similaire à GES: on ajoute ou retire des liens qui permettent d'augmenter le score, mais cette fois-ci en cherchant l'espace des classes d'équivalence interventionnelle de Markov. Les auteurs de la méthode ont également ajouté une troisième phase: une phase de retournement où on teste l'inversion de liens existants.

2.2.3. Métriques de graphes causaux

Souvent, on ne parvient pas à estimer exactement le graphe d'origine. Comment évaluer le graphe estimé par rapport au vrai graphe? Il existe un ensemble de méthodes (dont Constantinou (2019) fait une bonne revue), mais nous n'en présenterons que deux. Une première métrique simple est la *distance structurelle de Hamming* (SHD) où $SHD : \mathcal{G} \times \mathcal{G} \to \mathbb{N}$ qui retourne le nombre de liens différents entre deux graphes partiellement orientés (dont les DAGs sont un sous-ensemble) :

$$SHD(G_1, G_2) = |\{(i,j)|\text{le lien } i \to j \text{ n'est pas le même pour } G_1 \text{ et } G_2\}| \tag{2.2.3}$$

Parfois, il peut être aussi intéressant d'observer les différentes composantes de ce score: faux positifs, faux négatifs et liens inversés. Tandis que cette mesure est purement structurelle, la *distance structurelle interventionnelle*[2] (SID) (Peters et Bühlmann, 2015b) considère les implications causales. SID est défini sur l'espace des DAGs, soit $SID : \mathcal{G} \times \mathcal{G} \to \mathbb{N}$. $SID(G_1, G_2)$ correspond au nombre de liens de distribution interventionnelle $i \to j$ de G_1 qui ne peut être correctement estimé par G_2. En pratique, SID est calculé en déterminant si l'*ensemble d'ajustement* de G_2 est valide pour une intervention sur un noeud dans le graphe

[2]Contrairement à ce que son nom pourrait laisser croire, la distance structurelle interventionnelle n'est pas une distance puisqu'elle n'est pas symétrique.

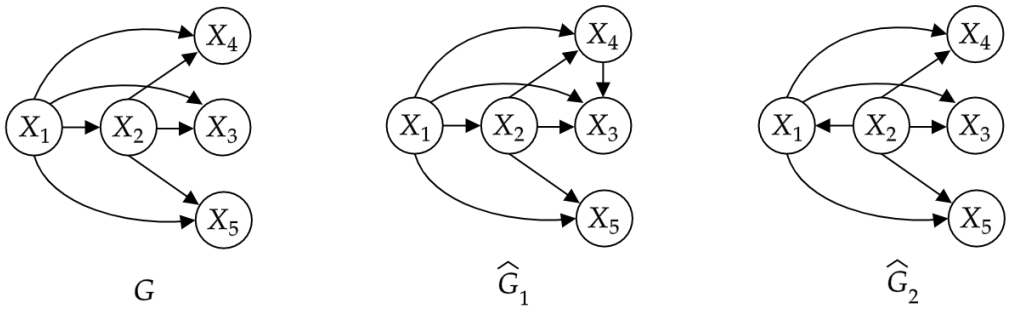


Fig. 2.5. Un graphe G et deux graphes estimés $\hat{G}_1$ et $\hat{G}_2$ ne différant que d'un lien de G. Pour $\hat{G}_1$, le lien $X_4 \to X_3$ est superflu par rapport à G. Pour $\hat{G}_2$, le lien entre X_1 et X_2 a été inversé. Exemple adapté de (Peters et Bühlmann, 2015b).

G_1. Un ensemble d'ajustement valide pour une certaine variable est un ensemble de variable qui permet de faire une estimation non-biaisé d'une distribution interventionelle à partir de données observationnelles. Une extension de cette métrique a été proposée pour traiter les CPDAGs. Dans ce cas, la métrique retourne des bornes inférieures et supérieures sur l'ensemble des DAGs de la classe d'équivalence.

Comme ces deux métriques seront utilisées dans les deux études de ce mémoire, prenons le temps de développer une intuition de leur comportement. Notons tout d'abord que si on obtient le même graphe, soit $G_1 = G_2$, alors $SHD(G_1, G_2)$ et $SID(G_1, G_2)$ seront tous deux égaux à 0. Cependant, on peut avoir $SID(G_1, G_2) = 0$, bien que $G_1 \neq G_2$. Cela provient du fait que $SID(G_1, G_2) = 0$ ssi G_1 est un sous-graphe de G_2, soit $G_1 \subseteq G_2$. En d'autres termes, même si SID est de 0, on peut avoir un SHD élevé. Afin d'illustrer la complémentarité de ces méthodes examinons l'exemple donnée à la Figure 2.5. Supposons que G est le vrai graphe que l'on cherche à découvrir. Nous l'avons estimé par $\hat{G}_1$ et $\hat{G}_2$ qui ne diffèrent que d'un lien de G. Lorsqu'on calcule le SHD par rapport au vrai graphe, on obtient donc 1 pour les deux graphes. Cependant les valeurs de SID sont très différentes. Puisque $G \subseteq \hat{G}_1$, alors $SID(G, \hat{G}_1) = 0$, tandis que $SID(G, \hat{G}_2) = 8$. En d'autres termes, pour un même SHD, l'impact au niveau causal de la différence d'un lien peut être très différent.

Chapitre 3

Gradient-Based Neural DAG Learning

par

Sébastien Lachapelle[1], Philippe Brouillard[1], Tristan Deleu[1] et Simon Lacoste-Julien[1]

([1]) Mila & DIRO, Université de Montréal

Cet article a été publié dans International Conference on Learning Representations (ICLR) 2020.

Contributions: *Sébastien Lachapelle* a mené le projet et a écrit l'article. Il a eu l'idée originale d'appliquer la contrainte d'acyclicté continue avec des réseaux de neurones. Il a implémenté cette méthode et réalisé la plupart des expériences.

J'ai intégré les différentes méthodes de bases (sauf DAG-GNN) et modifié ces méthodes pour qu'elles puissent supporter la validation croisée. J'ai généré les données synthétiques et conçu l'expérience de grands échantillons.

Tristan Deleu a intégré la méthode DAG-GNN et a fait les expériences avec DAG-GNN et NOTEARS.

Résumé. Nous proposons une nouvelle approche basée sur le score pour apprendre un graphe orienté acyclique (DAG) à partir de données observationnelles. Nous adaptons une formulation récemment proposée du problème en optimisation continue sous contrainte, afin de pouvoir supporter des relations non-linéaires entre les variable en utilisant des réseaux de neurones. Cette extension permet de modéliser des interactions complexes tout en évitant la nature combinatoire du problème. En plus de comparer notre méthode aux méthodes d'optimisation continue existantes, nous fournissons des comparaisons empiriques manquantes aux méthodes de recherche voraces non-linéaires. Sur des ensembles de données synthétiques et réels, cette nouvelle méthode surpasse les méthodes continues actuelles sur la plupart des tâches, tout en étant compétitive avec les méthodes de recherche voraces existantes sur des métriques importantes pour l'inférence causale.
Mots clés : apprentissage de la structure causale, apprentissage structuré, inférence causale, réseau de neurones

Abstract. We propose a novel score-based approach to learning a directed acyclic graph (DAG) from observational data. We adapt a recently proposed continuous constrained optimization formulation to allow for nonlinear relationships between variables using neural networks. This extension allows to model complex interactions while avoiding the combinatorial nature of the problem. In addition to comparing our method to existing continuous optimization methods, we provide missing empirical comparisons to nonlinear greedy search methods. On both synthetic and real-world data sets, this new method outperforms current continuous methods on most tasks, while being competitive with existing greedy search methods on important metrics for causal inference.
Keywords: causal structure learning, structure learning, causal inference, neural network

3.1. Introduction

Structure learning and causal inference have many important applications in different areas of science such as genetics (Koller et Friedman, 2009; Peters et al., 2017a), biology (Sachs et al., 2005a) and economics (Pearl, 2009a). *Bayesian networks* (BN), which encode conditional independencies using *directed acyclic graphs* (DAG), are powerful models which are both interpretable and computationally tractable. *Causal graphical models* (CGM) (Peters et al., 2017a) are BNs which support *interventional* queries like: *What will happen if someone external to the system intervenes on variable X?* Recent work suggests that causality could partially solve challenges faced by current machine learning systems such as robustness to out-of-distribution samples, adaptability and explainability (Pearl, 2019a; Magliacane et al., 2018). However, structure and causal learning are daunting tasks due to both the combinatorial nature of the space of structures (the number of DAGs grows *super exponentially* with the number of nodes) and the question of *structure identifiability* (see Section 3.2.2). Nevertheless, these graphical models known qualities and promises of improvement for machine intelligence renders the quest for structure/causal learning appealing.

The typical motivation for learning a causal graphical model is to predict the effect of various interventions. A CGM can be best estimated when given interventional data, but interventions are often costly or impossible to obtained. As an alternative, one can use exclusively observational data and rely on different assumptions which make the graph identifiable from the distribution (see Section 3.2.2). This is the approach employed in this paper.

We propose a score-based method (Koller et Friedman, 2009) for structure learning named GraN-DAG which makes use of a recent reformulation of the original *combinatorial problem* of finding an optimal DAG into a *continuous constrained optimization problem*. In the original method named NOTEARS (Zheng et al., 2018a), the directed graph is encoded as a *weighted adjacency matrix* which represents coefficients in a linear *structural equation model* (SEM) (Pearl, 2009a) (see Section 3.2.3) and enforces acyclicity using a constraint which is both efficiently computable and easily differentiable, thus allowing the use of numerical solvers. This continuous approach improved upon popular methods while avoiding the design of greedy algorithms based on heuristics.

Our first contribution is to extend the framework of Zheng et al. (2018a) to deal with nonlinear relationships between variables using neural networks (NN) (Goodfellow et al., 2016a). To adapt the acyclicity constraint to our nonlinear model, we use an argument similar to what is used in Zheng et al. (2018a) and apply it first at the level of *neural network paths* and then at the level of *graph paths*. Although GraN-DAG is general enough to deal with a large variety of parametric families of conditional probability distributions, our experiments focus on the special case of nonlinear Gaussian *additive noise models* since, under specific assumptions, it provides appealing theoretical guarantees easing the comparison to other graph search procedures (see Section 3.2.2 & 3.3.3). On both synthetic and real-world tasks, we show GraN-DAG often outperforms other approaches which leverage the continuous paradigm, including DAG-GNN (Yu et al., 2019b), a recent nonlinear extension of Zheng et al. (2018a) which uses an evidence lower bound as score.

Our second contribution is to provide a missing empirical comparison to existing methods that support nonlinear relationships but tackle the optimization problem in its discrete form using greedy search procedures, namely CAM (Bühlmann et al., 2014) and GSF (Huang et al., 2018a). We show that GraN-DAG is competitive on the wide range of tasks we considered, while using pre- and post-processing steps similar to CAM.

We provide an implementation of GraN-DAG here.

3.2. Background

Before presenting GraN-DAG, we review concepts relevant to structure and causal learning.

3.2.1. Causal graphical models

We suppose the natural phenomenon of interest can be described by a random vector $X \in \mathbb{R}^d$ entailed by an underlying CGM $(P_X, \mathcal{G})$ where P_X is a probability distribution over X and $\mathcal{G} = (V, E)$ is a DAG (Peters et al., 2017a). Each node $j \in V$ corresponds to exactly one variable in the system. Let $\pi_j^{\mathcal{G}}$ denote the set of parents of node j in $\mathcal{G}$ and let $X_{\pi_j^{\mathcal{G}}}$ denote the random vector containing the variables corresponding to the parents of j in $\mathcal{G}$. Throughout the paper, we assume there are no hidden variables. In a CGM, the distribution P_X is said to be *Markov* to $\mathcal{G}$, i.e. we can write the probability density function (pdf) of P_X as $p(x) = \prod_{j=1}^{d} p_j(x_j|x_{\pi_j^{\mathcal{G}}})$ where $p_j(x_j|x_{\pi_j^{\mathcal{G}}})$ is the conditional pdf of variable X_j given $X_{\pi_j^{\mathcal{G}}}$. A CGM can be thought of as a BN in which directed edges are given a causal meaning, allowing it to answer queries regarding *interventional distributions* (Koller et Friedman, 2009).

3.2.2. Structure identifiability

In general, it is impossible to recover $\mathcal{G}$ given only samples from P_X, i.e. without *interventional data*. It is, however, customary to rely on a set of assumptions to render the structure fully or partially *identifiable*.

Definition 1. Given a set of assumptions A on a CGM $\mathcal{M} = (P_X, \mathcal{G})$, its graph $\mathcal{G}$ is said to be *identifiable* from P_X if there exists no other CGM $\tilde{\mathcal{M}} = (\tilde{P}_X, \tilde{\mathcal{G}})$ satisfying all assumptions in A such that $\tilde{\mathcal{G}} \neq \mathcal{G}$ and $\tilde{P}_X = P_X$.

There are many examples of graph identifiability results for continuous variables (Peters et al., 2014a; Peters et Bühlman, 2014; Shimizu et al., 2006a; Zhang et Hyvärinen, 2009) as well as for discrete variables (Peters et al., 2011). These results are obtained by assuming that the conditional densities belong to a specific parametric family. For example, if one assumes that the distribution P_X is entailed by a structural equation model of the form

$$X_j := f_j(X_{\pi_j^{\mathcal{G}}}) + N_j \quad \text{with } N_j \sim \mathcal{N}(0, \sigma_j^2) \;\; \forall j \in V \tag{3.2.1}$$

where f_j is a nonlinear function satisfying some mild regularity conditions and the noises N_j are mutually independent, then $\mathcal{G}$ is identifiable from P_X (see Peters et al. (2014a) for the complete theorem and its proof). This is a particular instance of *additive noise models* (ANM). We will make use of this result in our experiments in Section 3.4.

One can consider weaker assumptions such as *faithfulness* (Peters et al., 2017a). This assumption allows one to identify, not $\mathcal{G}$ itself, but the *Markov equivalence class* to which it belongs (Spirtes et al., 2000a). A Markov equivalence class is a set of DAGs which encode exactly the same set of conditional independence statements and can be characterized by a graphical object named a *completed partially directed acyclic graph* (CPDAG) (Koller et Friedman, 2009; Peters et al., 2017a). Some algorithms we use as baselines in Section 3.4 output only a CPDAG.

3.2.3. NOTEARS: Continuous optimization for structure learning

Structure learning is the problem of learning $\mathcal{G}$ using a data set of n samples $\{x^{(1)}, ..., x^{(n)}\}$ from P_X. Score-based approaches cast this problem as an optimization problem, i.e. $\hat{\mathcal{G}} = \arg\max_{\mathcal{G} \in \text{DAG}} \mathcal{S}(\mathcal{G})$ where $\mathcal{S}(\mathcal{G})$ is a regularized maximum likelihood under graph $\mathcal{G}$. Since the number of DAGs is super exponential in the number of nodes, most methods rely on various heuristic greedy search procedures to approximately solve the problem (see Section 3.5 for a review). We now present the work of Zheng et al. (2018a) which proposes to cast this combinatorial optimization problem into a continuous constrained one.

To do so, the authors propose to encode the graph $\mathcal{G}$ on d nodes as a weighted adjacency matrix $U = [u_1|\dots|u_d] \in \mathbb{R}^{d \times d}$ which represents (possibly negative) coefficients in a linear SEM of the form $X_j := u_j^\top X + N_i \quad \forall j$ where N_j is a noise variable. Let $\mathcal{G}_U$ be the directed graph associated with the SEM and let A_U be the (binary) adjacency matrix associated with $\mathcal{G}_U$. One can see that the following equivalence holds:

$$(A_U)_{ij} = 0 \iff U_{ij} = 0 \tag{3.2.2}$$

To make sure $\mathcal{G}_U$ is acyclic, the authors propose the following constraint on U:

$$\operatorname{Tr} e^{U \odot U} - d = 0 \tag{3.2.3}$$

where $e^M \triangleq \sum_{k=0}^{\infty} \frac{M^k}{k!}$ is the *matrix exponential* and $\odot$ is the Hadamard product.

To see why this constraint characterizes acyclicity, first note that $({A_U}^k)_{jj}$ is the number of cycles of length k passing through node j in graph $\mathcal{G}_U$. Clearly, for $\mathcal{G}_U$ to be acyclic, we must have $\operatorname{Tr} {A_U}^k = 0$ for $k = 1,2, ..., \infty$. By equivalence (3.2.2), this is true when $\operatorname{Tr}(U \odot U)^k = 0$ for $k = 1,2, ..., \infty$. From there, one can simply apply the definition of the matrix exponential to see why constraint (3.2.3) characterizes acyclicity (see Zheng et al. (2018a) for the full development).

The authors propose to use a regularized negative least square score (maximum likelihood for a Gaussian noise model). The resulting continuous constrained problem is

$$\max_U \mathcal{S}(U, \mathbf{X}) \triangleq -\frac{1}{2n} \|\mathbf{X} - \mathbf{X}U\|_F^2 - \lambda \|U\|_1 \quad \text{s.t.} \quad \operatorname{Tr} e^{U \odot U} - d = 0 \tag{3.2.4}$$

where $\mathbf{X} \in \mathbb{R}^{n \times d}$ is the design matrix containing all n samples. The nature of the problem has been drastically changed: we went from a combinatorial to a continuous problem. The difficulties of combinatorial optimization have been replaced by those of non-convex optimization, since the feasible set is non-convex. Nevertheless, a standard numerical solver for constrained optimization such has an *augmented Lagrangian method* (Bertsekas, 1999) can be applied to get an approximate solution, hence there is no need to design a greedy search procedure. Moreover, this approach is more global than greedy methods in the sense that the whole matrix U is updated at each iteration. Continuous approaches to combinatorial

optimization have sometimes demonstrated improved performance over discrete approaches in the literature (see for example Alayrac et al. (2018, §5.2) where they solve the multiple sequence alignment problem with a continuous optimization method).

3.3. GraN-DAG: Gradient-based neural DAG learning

We propose a new nonlinear extension to the framework presented in Section 3.2.3. For each variable X_j, we learn a fully connected neural network with L hidden layers parametrized by $\phi_{(j)} \triangleq \{W_{(j)}^{(1)}, \dots, W_{(j)}^{(L+1)}\}$ where $W_{(j)}^{(\ell)}$ is the ℓth weight matrix of the jth NN (biases are omitted for clarity). Each NN takes as input $X_{-j} \in \mathbb{R}^d$, i.e. the vector X with the jth component masked to zero, and outputs $\theta_{(j)} \in \mathbb{R}^m$, the m-dimensional parameter vector of the desired distribution family for variable X_j.[1] The fully connected NNs have the following form

$$\theta_{(j)} \triangleq W_{(j)}^{(L+1)} g(\dots g(W_{(j)}^{(2)} g(W_{(j)}^{(1)} X_{-j})) \dots) \quad \forall j \tag{3.3.1}$$

where g is a nonlinearity applied element-wise. Note that the evaluation of all NNs can be parallelized on GPU. Distribution families need not be the same for each variable. Let $\phi \triangleq \{\phi_{(1)}, \dots, \phi_{(d)}\}$ represents all parameters of all d NNs. Without any constraint on its parameter $\phi_{(j)}$, neural network j models the conditional pdf $p_j(x_j|x_{-j}; \phi_{(j)})$. Note that the product $\prod_{j=1}^d p_j(x_j|x_{-j}; \phi_{(j)})$ does not integrate to one (i.e. it is not a joint pdf), since it does not decompose according to a DAG. We now show how one can constrain ϕ to make sure the product of all conditionals outputted by the NNs is a joint pdf. The idea is to define a new weighted adjacency matrix A_ϕ similar to the one encountered in Section 3.2.3, which can be directly used inside the constraint of Equation 3.2.3 to enforce acyclicity.

3.3.1. Neural network connectivity

Before defining the weighted adjacency matrix A_ϕ, we need to focus on how one can make some NN outputs unaffected by some inputs. Since we will discuss properties of a single NN, we drop the NN subscript (j) to improve readability.

We will use the term *neural network path* to refer to a computation path in a NN. For example, in a NN with two hidden layers, the sequence of weights $(W_{h_1 i}^{(1)}, W_{h_2 h_1}^{(2)}, W_{k h_2}^{(3)})$ is a NN path from input i to output k. We say that a NN path is *inactive* if at least one weight along the path is zero. We can loosely interpret the *path product* $|W_{h_1 i}^{(1)}||W_{h_2 h_1}^{(2)}||W_{k h_2}^{(3)}| \geq 0$ as the strength of the NN path, where a path product is equal to zero if and only if the path is inactive. Note that if all NN paths from input i to output k are inactive (i.e. the sum of their path products is zero), then output k does not depend on input i anymore since the information in input i will never reach output k. The sum of all path products from input

[1] Not all parameter vectors need to have the same dimensionality, but to simplify the notation, we suppose $m_j = m \ \ \forall j$

i to output k for all input i and output k can be easily computed by taking the following matrix product.

$$C \triangleq |W^{(L+1)}| \dots |W^{(2)}||W^{(1)}| \in \mathbb{R}_{\geq 0}^{m \times d} \tag{3.3.2}$$

where $|W|$ is the element-wise absolute value of W. Let us name C the *neural network connectivity matrix*. It can be verified that C_{ki} is the sum of all NN path products from input i to output k. This means it is sufficient to have $C_{ki} = 0$ to render output k independent of input i.

Remember that each NN in our model outputs a parameter vector θ for a conditional distribution and that we want the product of all conditionals to be a valid joint pdf, i.e. we want its corresponding directed graph to be acyclic. With this in mind, we see that it could be useful to make a certain parameter θ not dependent on certain inputs of the NN. To have θ independent of variable X_i, it is sufficient to have $\sum_{k=1}^{m} C_{ki} = 0$.

3.3.2. A weighted adjacency matrix

We now define a weighted adjacency matrix A_ϕ that can be used in constraint of Equation 3.2.3.

$$(A_\phi)_{ij} \triangleq \begin{cases} \sum_{k=1}^{m} \left(C_{(j)}\right)_{ki}, & \text{if } j \neq i \\ 0, & \text{otherwise} \end{cases} \tag{3.3.3}$$

where $C_{(j)}$ denotes the connectivity matrix of the NN associated with variable X_j.

As the notation suggests, $A_\phi \in \mathbb{R}_{\geq 0}^{d \times d}$ depends on all weights of all NNs. Moreover, it can effectively be interpreted as a weighted adjacency matrix similarly to what we presented in Section 3.2.3, since we have that

$$(A_\phi)_{ij} = 0 \implies \theta_{(j)} \text{ does not depend on variable } X_i \tag{3.3.4}$$

We note $\mathcal{G}_\phi$ to be the directed graph entailed by parameter ϕ. We can now write our adapted acyclicity constraint:

$$h(\phi) \triangleq \operatorname{Tr} e^{A_\phi} - d = 0 \tag{3.3.5}$$

Note that we can compute the gradient of $h(\phi)$ w.r.t. ϕ (except at points of non-differentiability arising from the absolute value function, similar to standard neural networks with ReLU activations (Glorot et al., 2011); these points did not appear problematic in our experiments using SGD).

3.3.3. A differentiable score and its optimization

We propose solving the maximum likelihood optimization problem

$$\max_{\phi} \mathbb{E}_{X \sim P_X} \sum_{j=1}^{d} \log p_j(X_j | X_{\pi_j^\phi}; \phi_{(j)}) \quad \text{s.t.} \quad \operatorname{Tr} e^{A_\phi} - d = 0 \tag{3.3.6}$$

where π_j^ϕ denotes the set of parents of node j in graph $\mathcal{G}_\phi$. Note that $\sum_{j=1}^d \log p_j(X_j|X_{\pi_j^\phi};\phi_{(j)})$ is a valid log-likelihood function when constraint (3.3.5) is satisfied.

As suggested in Zheng et al. (2018a), we apply an augmented Lagrangian approach to get an approximate solution to program (3.3.6). Augmented Lagrangian methods consist of optimizing a sequence of subproblems for which the exact solutions are known to converge to a stationary point of the constrained problem under some regularity conditions (Bertsekas, 1999). In our case, each subproblem is

$$\max_\phi \mathcal{L}(\phi, \lambda_t, \mu_t) \triangleq \mathbb{E}_{X\sim P_X} \sum_{j=1}^d \log p_j(X_j|X_{\pi_j^\phi};\phi_{(j)}) - \lambda_t h(\phi) - \frac{\mu_t}{2} h(\phi)^2 \tag{3.3.7}$$

where λ_t and μ_t are the Lagrangian and penalty coefficients of the tth subproblem, respectively. These coefficients are updated after each subproblem is solved. Since GraN-DAG rests on neural networks, we propose to approximately solve each subproblem using a well-known stochastic gradient algorithm popular for NN in part for its implicit regularizing effect (Poggio et al., 2018). See Appendix A.1.1 for details regarding the optimization procedure.

In the current section, we presented GraN-DAG in a general manner without specifying explicitly which distribution family is parameterized by $\theta_{(j)}$. In principle, any distribution family could be employed as long as its log-likelihood can be computed and differentiated with respect to its parameter θ. However, it is not always clear whether the exact solution of problem (3.3.6) recovers the ground truth graph $\mathcal{G}$. It will depend on both the modelling choice of GraN-DAG and the underlying CGM $(P_X, \mathcal{G})$.

Proposition 2. Let ϕ^* and $\mathcal{G}_{\phi^*}$ be the optimal solution to (3.3.6) and its corresponding graph, respectively. Let $\mathcal{M}(A)$ be the set of CGM $(P', \mathcal{G}')$ for which the assumptions in A are satisfied and let $\mathcal{C}$ be the set of CGM $(P', \mathcal{G}')$ which can be represented by the model (e.g. NN outputting a Gaussian distribution). If the underlying CGM $(P_X, \mathcal{G}) \in \mathcal{C}$ and $\mathcal{C} = \mathcal{M}(A)$ for a specific set of assumptions A such that $\mathcal{G}$ is identifiable from P_X, then $\mathcal{G}_{\phi^*} = \mathcal{G}$.

Proof: Let P_ϕ be the joint distribution entailed by parameter ϕ. Note that the population log-likelihood $\mathbb{E}_{X\sim P_X} \log p_\phi(X)$ is maximal iff $P_\phi = P_X$. We know this maximum can be achieved by a specific parameter ϕ^* since by hypothesis $(P_X, \mathcal{G}) \in \mathcal{C}$. Since $\mathcal{G}$ is identifiable from P_X, we know there exists no other CGM $(\tilde{P}_X, \tilde{\mathcal{G}}) \in \mathcal{C}$ such that $\tilde{\mathcal{G}} \neq \mathcal{G}$ and $\tilde{P}_X = P_X$. Hence $\mathcal{G}_{\phi^*}$ has to be equal to $\mathcal{G}$.■

In Section 3.4.1, we empirically explore the identifiable setting of nonlinear Gaussian ANMs introduced in Section 3.2.2. In practice, one should keep in mind that solving (3.3.6) exactly is hard since the problem is non-convex (the augmented Lagrangian converges only to a stationary point) and moreover we only have access to the empirical log-likelihood (Proposition 2 holds for the population case).

3.3.4. Thresholding

The solution outputted by the augmented Lagrangian will satisfy the constraint only up to numerical precision, thus several entries of A_ϕ might not be exactly zero and require thresholding. To do so, we mask the inputs of each NN j using a binary matrix $M_{(j)} \in \{0,1\}^{d\times d}$ initialized to have $(M_{(j)})_{ii} = 1 \ \ \forall i \neq j$ and zeros everywhere else. Having $(M_{(j)})_{ii} = 0$ means the input i of NN j has been thresholded. This mask is integrated in the product of Equation 3.3.2 by doing $C_{(j)} \triangleq |W_{(j)}^{(L+1)}| \dots |W_{(j)}^{(1)}| M_{(j)}$ without changing the interpretation of $C_{(j)}$ ($M_{(j)}$ can be seen simply as an extra layer in the NN). During optimization, if the entry $(A_\phi)_{ij}$ is smaller than the threshold $\epsilon = 10^{-4}$, the corresponding mask entry $(M_{(j)})_{ii}$ is set to zero, permanently. The masks $M_{(j)}$ are never updated via gradient descent. We also add an iterative thresholding step at the end to ensure the estimated graph $\mathcal{G}_\phi$ is acyclic (described in Appendix A.1.2).

3.3.5. Overfitting

In practice, we maximize an empirical estimate of the objective of problem (3.3.6). It is well known that this maximum likelihood score is prone to overfitting in the sense that adding edges can never reduce the maximal likelihood (Koller et Friedman, 2009). GraN-DAG gets around this issue in four ways. First, as we optimize a subproblem, we evaluate its objective on a held-out data set and declare convergence once it has stopped improving. This approach is known as *early stopping* (Prechelt, 1997). Second, to optimize (3.3.7), we use a stochastic gradient algorithm variant which is now known to have an implicit regularizing effect (Poggio et al., 2018). Third, once we have thresholded our graph estimate to be a DAG, we apply a final pruning step identical to what is done in CAM (Bühlmann et al., 2014) to remove spurious edges. This step performs a regression of each node against its parents and uses a significance test to decide which parents should be kept or not. Fourth, for graphs of 50 nodes or more, we apply a *preliminary neighbors selection* (PNS) before running the optimization procedure as was also recommended in Bühlmann et al. (2014). This procedure selects a set of potential parents for each variables. See Appendix A.1.3 for details on PNS and pruning. Many score-based approaches control overfitting by penalizing the number of edges in their score. For example, NOTEARS includes the L1 norm of its weighted adjacency matrix U in its objective. GraN-DAG regularizes using PNS and pruning for ease of comparision to CAM, the most competitive approach in our experiments. The importance of PNS and pruning and their ability to reduce overfitting is illustrated in an ablation study presented in Appendix A.1.3. The study shows that PNS and pruning are both very important for the performance of GraN-DAG in terms of SHD, but do not have a significant effect in terms of SID. In these experiments, we also present NOTEARS and DAG-GNN with PNS and pruning, without noting a significant improvement.

3.3.6. Computational Complexity

To learn a graph, GraN-DAG relies on the proper training of neural networks on which it is built. We thus propose using a stochastic gradient method which is a standard choice when it comes to NN training because it scales well with both the sample size and the number of parameters and it implicitly regularizes learning. Similarly to NOTEARS, GraN-DAG requires the evaluation of the matrix exponential of A_ϕ at each iteration costing $\mathcal{O}(d^3)$. NOTEARS justifies the use of a batch proximal quasi-Newton algorithm by the low number of $\mathcal{O}(d^3)$ iterations required to converge. Since GraN-DAG uses a stochastic gradient method, one would expect it will require more iterations to converge. However, in practice we observe that GraN-DAG performs fewer iterations than NOTEARS before the augmented Lagrangian converges (see Table A.1 of Appendix A.1.1). We hypothesize this is due to early stopping which avoids having to wait until the full convergence of the subproblems hence limiting the total number of iterations. Moreover, for the graph sizes considered in this paper ($d \leq 100$), the evaluation of $h(\phi)$ in GraN-DAG, which includes the matrix exponentiation, does not dominate the cost of each iteration ($\approx 4\%$ for 20 nodes and $\approx 13\%$ for 100 nodes graphs). Evaluating the approximate gradient of the log-likelihood (costing $\mathcal{O}(d^2)$ assuming a fixed minibatch size, NN depth and width) appears to be of greater importance for $d \leq 100$.

3.4. Experiments

In this section, we compare GraN-DAG to various baselines in the continuous paradigm, namely DAG-GNN (Yu et al., 2019b) and NOTEARS (Zheng et al., 2018a), and also in the combinatorial paradigm, namely CAM (Bühlmann et al., 2014), GSF (Huang et al., 2018a), GES (Chickering, 2003a) and PC (Spirtes et al., 2000a). These methods are discussed in Section 3.5. In all experiments, each NN learned by GraN-DAG outputs the mean of a Gaussian distribution $\hat{\mu}_{(j)}$, i.e. $\theta_{(j)} := \hat{\mu}_{(j)}$ and $X_j | X_{\pi_j^\mathcal{G}} \sim \mathcal{N}(\hat{\mu}_{(j)}, \hat{\sigma}^2_{(j)}) \;\; \forall j$. The parameters $\hat{\sigma}^2_{(j)}$ are learned as well, but do not depend on the parent variables $X_{\pi_j^\mathcal{G}}$ (unless otherwise stated). Note that this modelling choice matches the nonlinear Gaussian ANM introduced in Section 3.2.2.

We report the performance of random graphs sampled using the *Erdős-Rényi* (ER) scheme described in Appendix A.1.5 (denoted by RANDOM). For each approach, we evaluate the estimated graph on two metrics: the *structural hamming distance* (SHD) and the *structural interventional distance* (SID) (Peters et Bühlmann, 2013). The former simply counts the number of missing, falsely detected or reversed edges. The latter is especially well suited for causal inference since it counts the number of couples (i, j) such that the interventional distribution $p(x_j | do(X_i = \bar{x}))$ would be miscalculated if we were to use the estimated graph to form the parent adjustement set. Note that GSF, GES and PC output only a CPDAG, hence the need to report a lower and an upper bound on the SID. See

Appendix A.1.7 for more details on SHD and SID. All experiments were ran with publicly available code from the authors website. See Appendix A.1.8 for the details of their hyperparameters. In Appendix A.1.9, we explain how one could use a held-out data set to select the hyperparameters of score-based approaches and report the results of such a procedure on almost every settings discussed in the present section.

3.4.1. Synthetic data

We have generated different *data set types* which vary along four dimensions: data generating process, number of nodes, level of edge sparsity and graph type. We consider two graph sampling schemes: *Erdős-Rényi* (ER) and *scale-free* (SF) (see Appendix A.1.5 for details). For each data set type, we sampled 10 data sets of 1000 examples as follows: First, a ground truth DAG $\mathcal{G}$ is randomly sampled following either the ER or the SF scheme. Then, the data is generated according to a specific sampling scheme.

The first data generating process we consider is the nonlinear Gaussian ANM (Gauss-ANM) introduced in Section 3.2.2 in which data is sampled following $X_j := f_j(X_{\pi_j^{\mathcal{G}}}) + N_j$ with mutually independent noises $N_j \sim \mathcal{N}(0, \sigma_j^2) \;\; \forall j$ where the functions f_j are independently sampled from a Gaussian process with a unit bandwidth RBF kernel and with σ_j^2 sampled uniformly. As mentioned in Section 3.2.2, we know $\mathcal{G}$ to be identifiable from the distribution. Proposition 2 indicates that the modelling choice of GraN-DAG together with this synthetic data ensure that solving (3.3.6) to optimality would recover the correct graph. Note that NOTEARS and CAM also make the correct Gaussian noise assumption, but do not have enough capacity to represent the f_j functions properly.

We considered graphs of 10, 20, 50 and 100 nodes. Tables 3.1 & 3.2 present results only for 10 and 50 nodes since the conclusions do not change with graphs of 20 or 100 nodes (see Appendix A.1.6 for these additional experiments). We consider graphs of d and $4d$ edges (respectively denoted by ER1 and ER4 in the case of ER graphs). We report the performance of the popular GES and PC in Appendix A.1.6 since they are almost never on par with the best methods presented in this section.

Table 3.1. Results for ER and SF graphs of 10 nodes with Gauss-ANM data

	ER1		ER4		SF1		SF4	
	SHD	SID	SHD	SID	SHD	SID	SHD	SID
GraN-DAG	**1.7±2.5**	**1.7±3.1**	**8.3±2.8**	**21.8±8.9**	**1.2±1.1**	**4.1±6.1**	**9.9±4.0**	**16.4±6.0**
DAG-GNN	11.4±3.1	37.6±14.4	35.1±1.5	81.9±4.7	9.9±1.1	29.7±15.8	20.8±1.9	48.4±15.6
NOTEARS	12.2±2.9	36.6±13.1	32.6±3.2	79.0±4.1	10.7±2.2	32.0±15.3	20.8±2.7	49.8±15.6
CAM	**1.1±1.1**	**1.1±2.4**	**12.2±2.7**	**30.9±13.2**	**1.4±1.6**	**5.4±6.1**	**9.8±4.3**	**19.3±7.5**
GSF	6.5±2.6	[6.2±10.8 17.7±12.3]	21.7±8.4	[37.2±19.2 62.7±14.9]	**1.8±1.7**	**[2.0±5.1 6.9±6.2]**	**8.5±4.2**	**[13.2±6.8 20.6±12.1]**
RANDOM	26.3±9.8	25.8±10.4	31.8±5.0	76.6±7.0	25.1±10.2	24.5±10.5	28.5±4.0	47.2±12.2

Table 3.2. Results for ER and SF graphs of 50 nodes with Gauss-ANM data

	ER1		ER4		SF1		SF4	
	SHD	SID	SHD	SID	SHD	SID	SHD	SID
GraN-DAG	**5.1±2.8**	**22.4±17.8**	**102.6±21.2**	**1060.1±109.4**	**25.5±6.2**	**90.0±18.9**	**111.3±12.3**	**271.2±65.4**
DAG-GNN	49.2±7.9	304.4±105.1	191.9±15.2	2146.2±64	49.8±1.3	182.8±42.9	144.9±13.3	540.8±151.1
NOTEARS	62.8±9.2	327.3±119.9	202.3±14.3	2149.1±76.3	57.7±3.5	195.7±54.9	153.7±11.8	558.4±153.5
CAM	**4.3±1.9**	**22.0±17.9**	**98.8±20.7**	**1197.2±125.9**	**24.1±6.2**	**85.7±31.9**	**111.2±13.3**	**320.7±152.6**
GSF	25.6±5.1	[21.1±23.1 79.2±33.5]	**81.8±18.8**	[**906.6±214.7 1030.2±172.6**]	**31.6±6.7**	[85.8±29.9 147.3±49.9]	**120.2±10.9**	[284.7±80.2 379.9±98.3]
RANDOM	535.7±401.2	272.3±125.5	708.4±234.4	1921.3±203.5	514.0±360.0	381.3±190.3	660.6±194.9	1198.9±304.6

We now examine Tables 3.1 & 3.2 (the errors bars represent the standard deviation across datasets per task). We can see that, across all settings, GraN-DAG and CAM are the best performing methods, both in terms of SHD and SID, while GSF is not too far behind. The poor performance of NOTEARS can be explained by its inability to model nonlinear functions. In terms of SHD, DAG-GNN performs rarely better than NOTEARS while in terms of SID, it performs similarly to RANDOM in almost all cases except in scale-free networks of 50 nodes or more. Its poor performance might be due to its incorrect modelling assumptions and because its architecture uses a strong form of parameter sharing between the f_j functions, which is not justified in a setup like ours. GSF performs always better than DAG-GNN and NOTEARS but performs as good as CAM and GraN-DAG only about half the time. Among the continuous approaches considered, GraN-DAG is the best performing on these synthetic tasks.

Even though CAM (wrongly) assumes that the functions f_j are additive, i.e. $f_j(x_{\pi_j^\mathcal{G}}) = \sum_{i \in \pi_j^\mathcal{G}} f_{ij}(x_j) \;\; \forall j$, it manages to compete with GraN-DAG which does not make this incorrect modelling assumption[2]. This might partly be explained by a bias-variance trade-off. CAM is biased but has a lower variance than GraN-DAG due to its restricted capacity, resulting in both methods performing similarly. In Appendix A.1.4, we present an experiment showing that GraN-DAG can outperform CAM in higher sample size settings, suggesting this explanation is reasonable.

Having confirmed that GraN-DAG is competitive on the ideal Gauss-ANM data, we experimented with settings better adjusted to other models to see whether GraN-DAG remains competitive. We considered linear Gaussian data (better adjusted to NOTEARS) and nonlinear Gaussian data with additive functions (better adjusted to CAM) named LIN and ADD-FUNC, respectively. See Appendix A.1.5 for the details of their generation. We report the results of GraN-DAG and other baselines in Table A.9 & A.10 of the appendix. On linear Gaussian data, most methods score poorly in terms of SID which is probably due to the unidentifiability of the linear Gaussian model (when the noise variances are unequal). GraN-DAG and CAM perform similarly to NOTEARS in terms of SHD. On ADD-FUNC,

[2]Although it is true that GraN-DAG does not wrongly assume that the functions f_j are additive, it is not clear whether its neural networks can exactly represent functions sampled from the Gaussian process.

CAM dominates all methods on most graph types considered (GraN-DAG is on par only for the 10 nodes ER1 graph). However, GraN-DAG outperforms all other methods which can be explained by the fact that the conditions of Proposition 2 are satisfied (supposing the functions $\sum_{i\in\pi_j^{\mathcal{G}}} f_{ij}(X_i)$ can be represented by the NNs).

We also considered synthetic data sets which do not satisfy the additive Gaussian noise assumption present in GraN-DAG, NOTEARS and CAM. We considered two kinds of *post nonlinear causal models* (Zhang et Hyvärinen, 2009), PNL-GP and PNL-MULT (see Appendix A.1.5 for details about their generation). A post nonlinear model has the form $X_j := g_j(f_j(X_{\pi_j^{\mathcal{G}}}) + N_j)$ where N_j is a noise variable. Note that GraN-DAG (with the current modelling choice) and CAM do not have the representational power to express these conditional distributions, hence violating an assumption of Proposition 2. However, these data sets differ from the previous additive noise setup only by the nonlinearity g_j, hence offering a case of mild model misspecification. The results are reported in Table A.11 of the appendix. GraN-DAG and CAM are outperforming DAG-GNN and NOTEARS except in SID for certain data sets where all methods score similarly to RANDOM. GraN-DAG and CAM have similar performance on all data sets except one where CAM is better. GSF performs worst than GraN-DAG (in both SHD and SID) on PNL-GP but not on PNL-MULT where it performs better in SID.

3.4.2. Real and pseudo-real data

We have tested all methods considered so far on a well known data set which measures the expression level of different proteins and phospholipids in human cells (Sachs et al., 2005a). We trained only on the $n = 853$ observational samples. This dataset and its ground truth graph proposed in Sachs et al. (2005a) (11 nodes and 17 edges) are often used in the probabilistic graphical model literature (Koller et Friedman, 2009). We also consider pseudo-real data sets sampled from the SynTReN generator (Van den Bulcke, 2006). This generator was designed to create synthetic transcriptional regulatory networks and produces simulated gene expression data that approximates experimental data. See Appendix A.1.5 for details of the generation.

In applications, it is not clear whether the conditions of Proposition 2 hold since we do not know whether $(P_X, \mathcal{G}) \in \mathcal{C}$. This departure from identifiable settings is an occasion to explore a different modelling choice for GraN-DAG. In addition to the model presented at the beginning of this section, we consider an alternative, denoted GraN-DAG++, which allows the variance parameters $\hat{\sigma}^2_{(i)}$ to depend on the parent variables $X_{\pi_i^{\mathcal{G}}}$ through the NN, i.e. $\theta_{(i)} := (\hat{\mu}_{(i)}, \log \hat{\sigma}^2_{(i)})$. Note that this is violating the additive noise assumption (in ANMs, the noise is independent of the parent variables).

In addition to metrics used in Section 3.4.1, we also report SHD-C. To compute the SHD-C between two DAGs, we first map each of them to their corresponding CPDAG and measure the SHD between the two. This metric is useful to compare algorithms which only outputs a CPDAG like GSF, GES and PC to other methods which outputs a DAG. Results are reported in Table 3.3.

Table 3.3. Results on real and pseudo-real data

	Protein signaling data set			SynTReN (20 nodes)		
	SHD	SHD-C	SID	SHD	SHD-C	SID
GraN-DAG	13	11	47	**34.0±8.5**	**36.4±8.3**	161.7±53.4
GraN-DAG++	13	10	48	**33.7±3.7**	**39.4±4.9**	127.5±52.8
DAG-GNN	16	21	44	93.6±9.2	97.6±10.3	157.5±74.6
NOTEARS	21	21	44	151.8±28.2	156.1±28.7	**110.7±66.7**
CAM	**12**	**9**	55	**40.5±6.8**	**41.4±7.1**	152.3±48
GSF	18	10	[44, 61]	61.8±9.6	63.3±11.4	[**76.7±51.1, 109.9±39.9**]
GES	26	28	[**34, 45**]	82.6±9.3	85.6±10	[157.2±48.3, 168.8±47.8]
PC	17	11	[47, 62]	**41.2±5.1**	**42.4±4.6**	[154.8±47.6, 179.3±55.6]
RANDOM	21	20	60	84.7±53.8	86.7±55.8	175.8±64.7

First, all methods perform worse than what was reported for graphs of similar size in Section 3.4.1, both in terms of SID and SHD. This might be due to the lack of identifiability guarantees we face in applications. On the protein data set, GraN-DAG outperforms CAM in terms of SID (which differs from the general trend of Section 3.4.1) and arrive almost on par in terms of SHD and SHD-C. On this data set, DAG-GNN has a reasonable performance, beating GraN-DAG in SID, but not in SHD. On SynTReN, GraN-DAG obtains the best SHD but not the best SID. Overall, GraN-DAG is always competitive with the best methods of each task.

3.5. Related Work

Most methods for structure learning from observational data make use of some identifiability results similar to the ones raised in Section 3.2.2. Roughly speaking, there are two classes of methods: *independence-based* and *score-based* methods. GraN-DAG falls into the second class.

Score-based methods (Koller et Friedman, 2009; Peters et al., 2017a) cast the problem of structure learning as an optimization problem over the space of structures (DAGs or CPDAGs). Many popular algorithms tackle the combinatorial nature of the problem by performing a form of greedy search. GES (Chickering, 2003a) is a popular example. It usually assumes a linear parametric model with Gaussian noise and greedily search the space of CPDAGs in order to optimize the Bayesian information criterion. GSF (Huang et al., 2018a), is based on the same search algorithm as GES, but uses a generalized score function which can model nonlinear relationships. Other greedy approaches rely on parametric assumptions which render $\mathcal{G}$ fully identifiable. For example, Peters et Bühlman (2014) relies on a linear

Gaussian model with equal variances to render the DAG identifiable. RESIT (Peters et al., 2014a), assumes nonlinear relationships with additive Gaussian noise and greedily maximizes an independence-based score. However, RESIT does not scale well to graph of more than 20 nodes. CAM (Bühlmann et al., 2014) decouples the search for the optimal node ordering from the parents selection for each node and assumes an additive noise model (ANM) (Peters et al., 2017a) in which the nonlinear functions are additive. As mentioned in Section 3.2.3, NOTEARS, proposed in Zheng et al. (2018a), tackles the problem of finding an optimal DAG as a continuous constrained optimization program. This is a drastic departure from previous combinatorial approaches which enables the application of well studied numerical solvers for continuous optimizations. Recently, Yu et al. (2019b) proposed DAG-GNN, a graph neural network architecture (GNN) which can be used to learn DAGs via the maximization of an evidence lower bound. By design, a GNN makes use of parameter sharing which we hypothesize is not well suited for most DAG learning tasks. To the best of our knowledge, DAG-GNN is the first approach extending the NOTEARS algorithm for structure learning to support nonlinear relationships. Although Yu et al. (2019b) provides empirical comparisons to linear approaches, namely NOTEARS and FGS (a faster extension of GES) (Ramsey et al., 2017), comparisons to greedy approaches supporting nonlinear relationships such as CAM and GSF are missing. Moreover, GraN-DAG significantly outperforms DAG-GNN on our benchmarks. There exists certain score-based approaches which uses integer linear programming (ILP) (Jaakkola et al., 2010; Cussens, 2011) which internally solve continuous linear relaxations. Connections between such methods and the continuous constrained approaches are yet to be explored.

When used with the additive Gaussian noise assumption, the theoretical guarantee of GraN-DAG rests on the identifiability of nonlinear Gaussian ANMs. Analogously to CAM and NOTEARS, this guarantee holds only if the correct identifiability assumptions hold in the data and if the score maximization problem is solved exactly (which is not the case in all three algorithms). DAG-GNN provides no theoretical justification for its approach. NOTEARS and CAM are designed to handle what is sometimes called the *high-dimensional setting* in which the number of samples is significantly smaller than the number of nodes. Bühlmann et al. (2014) provides consistency results for CAM in this setting. GraN-DAG and DAG-GNN were not designed with this setting in mind and would most likely fail if confronted to it. Solutions for fitting a neural network on less data points than input dimensions are not common in the NN literature.

Methods for causal discovery using NNs have already been proposed. SAM (Kalainathan et al., 2018a) learns conditional NN generators using adversarial losses but does not enforce acyclicity. CGNN (Goudet et al., 2018), when used for multivariate data, requires an initial skeleton to learn the different functional relationships.

GraN-DAG has strong connections with MADE (Germain et al., 2015), a method used to learn distributions using a masked NN which enforces the so-called *autoregressive property*. The autoregressive property and acyclicity are in fact equivalent. MADE does not learn the weight masking, it fixes it at the beginning of the procedure. GraN-DAG could be used with a unique NN taking as input all variables and outputting parameters for all conditional distributions. In this case, it would be similar to MADE, except the variable ordering would be learned from data instead of fixed *a priori*.

3.6. Conclusion

The continuous constrained approach to structure learning has the advantage of being more global than other approximate greedy methods (since it updates all edges at each step based on the gradient of the score but also the acyclicity constraint) and allows to replace task-specific greedy algorithms by appropriate off-the-shelf numerical solvers. In this work, we have introduced GraN-DAG, a novel score-based approach for structure learning supporting nonlinear relationships while leveraging a continuous optimization paradigm. The method rests on a novel characterization of acyclicity for NNs based on the work of Zheng et al. (2018a). We showed GraN-DAG outperforms other gradient-based approaches, namely NOTEARS and its recent nonlinear extension DAG-GNN, on the synthetic data sets considered in Section 3.4.1 while being competitive on real and pseudo-real data sets of Section 3.4.2. Compared to greedy approaches, GraN-DAG is competitive across all datasets considered. To the best of our knowledge, GraN-DAG is the first approach leveraging the continuous paradigm introduced in Zheng et al. (2018a) which has been shown to be competitive with state of the art methods supporting nonlinear relationships.

Chapitre 4

Differentiable Causal Discovery from Interventional Data

by

Philippe Brouillard *[1], Sébastien Lachapelle *[1], Alexandre Lacoste[2], Simon Lacoste-Julien[3], and Alexandre Drouin[4]

([1]) Mila & DIRO, Université de Montréal
([2]) Element AI
([3]) Mila & DIRO, Université de Montréal
([4]) Element AI

Cet article a été soumis dans *Neural Information Processing Systems* (NeurIPS) 2020 (présentement en review).

Contribution: J'ai mené ce projet et j'ai donc participé à toutes les étapes de manière significative. Je me suis plus particulièrement concentré sur l'implémentation des différentes méthodes proposées et l'intégration des méthodes de bases (baseline), la génération des données synthétiques, la plupart des expériences et la revue de littérature.

* Contribution équivalente

Sébastien Lachapelle a réalisé la partie théorique dont le théorème et sa preuve. Il a également réalisé les expériences d'adaptabilité à différentes tailles de graphes et d'échantillons. Il a aussi suggéré l'idée originale d'adapter GraN-DAG aux données interventionnelles.

Alexandre Drouin, en plus de contribuer en tant que superviseur, a participé aux expérimentations et a effectué une recherche d'hyperparamètre extensive. Il a aussi intégré DSF.

Résumé. La découverte de relations causales dans les données est une tâche difficile qui implique de résoudre un problème combinatoire pour lequel la solution n'est pas toujours identifiable. Une nouvelle ligne de recherche reformule le problème combinatoire en un problème d'optimisation continu sous contrainte, permettant l'utilisation de différentes techniques d'optimisation puissantes. Cependant, les méthodes basées sur cette idée n'utilisent pas de données interventionnelles, ce qui peut pourtant atténuer considérablement les problèmes d'identification. Dans ce travail, nous proposons une méthode basée sur les réseaux de neurones qui peut tirer parti des données interventionnelles. Nous illustrons la flexibilité du cadre continu contraint en tirant parti d'architectures neuronales expressives telles que des *flots normalisés* (normalizing flows). Nous montrons que notre approche se compare favorablement à l'état de l'art dans une variété de contextes dont des données avec interventions parfaites et imparfaites pour lesquelles les noeuds ciblés peuvent même être inconnus.
Mots clés : apprentissage de la structure causale, apprentissage structuré, inférence causale, réseau de neurones

Abstract. Discovering causal relationships in data is a challenging task that involves solving a combinatorial problem for which the solution is not always identifiable. A new line of work reformulates the combinatorial problem as a continuous constrained optimization one, enabling the use of different powerful optimization techniques. However, methods based on this idea do not yet make use of interventional data, which can significantly alleviate identifiability issues. In this work, we propose a neural network-based method for this task that can leverage interventional data. We illustrate the flexibility of the continuous-constrained framework by taking advantage of expressive neural architectures such as normalizing flows. We show that our approach compares favorably to the state of the art in a variety of settings, including perfect and imperfect interventions for which the targeted nodes may even be unknown.
Keywords: causal structure learning, structure learning, causal inference, neural network

4.1. Introduction

The inference of causal relationships is a problem of fundamental interest in science. In all fields of research, experiments are systematically performed with the goal of elucidating the underlying causal dynamics of systems. This quest for causality is motivated by the desire to take actions that induce a controlled change in a system. Achieving this requires to answer questions, such as "what would be the impact on the system if this variable were

changed from value x to y?", which cannot be answered without causal knowledge (Pearl, 2009b).

In this work, we address the problem of data-driven causal discovery (Heinze-Deml et al., 2018a). Our goal is to design an algorithm that can automatically discover causal relationships from data. More formally, we aim to learn a *causal graphical model* (CGM) (Peters et al., 2017b), which consists of a joint distribution coupled with a directed acyclic graph (DAG), where edges indicate direct causal relationships. Achieving this based on observational data alone is challenging since, under the faithfulness assumption, the true DAG is only identifiable up to a *Markov equivalence class* (Verma et Pearl, 1991). Fortunately, identifiability can be improved by considering interventional data, i.e., the outcome of some experiments. In this case, the DAG is identifiable up to an *interventional Markov equivalence class*, which is a subset of the Markov equivalence class (Yang et al., 2018; Hauser et Bühlmann, 2012), and, when observing enough interventions (Eberhardt, 2008; Eberhardt et al., 2005), the DAG is exactly identifiable. In practice, it may be possible for domain experts to collect such interventional data, resulting in clear gains in identifiability. For instance, in genomics, recent advances in gene editing technologies have given rise to high-throughput methods for interventional gene expression data (Dixit et al., 2016).

Nevertheless, even with interventional data at hand, finding the right DAG is challenging. The solution space is immense and grows super-exponentially with respect to the number of variables of interest. Recently, Zheng et al. (2018b) proposed to cast this search problem as a constrained continuous optimization problem, avoiding the computationally intensive search performed by score-based and constrained-based methods (Peters et al., 2017b). The work of Zheng et al. (2018b) was limited to linear causal relationships, but was quickly extended to nonlinear ones via neural networks (Lachapelle et al., 2020; Yu et al., 2019a; Zheng et al., 2020; Ng et al., 2019; Kalainathan et al., 2018b; Zhu et Chen, 2020). However, such approaches have only been applied to *observational* data under some assumptions (e.g., additive noise models) and cannot make use of interventional data. In this work, we propose a differentiable approach to causal discovery that can make use of *interventional* data to model complex nonlinear causal relationships without making such assumptions.

4.1.1. Contributions

- We propose the approach Differentiable Causal Discovery with Interventions (DCDI): a general differentiable causal structure learning method that can leverage perfect, imperfect and unknown interventions (Section 4.3). We propose two instantiations, one of which is a universal density approximator that uses normalizing flows (Section 4.3.4).

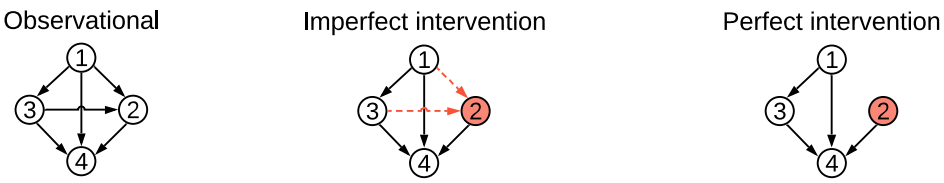


Fig. 4.1. Different intervention types (shown in red). In imperfect interventions, the causal relationships are altered. In perfect interventions, the targeted node is cut out from its parents.

- We prove an identifiability result to support our proposed approach (Theorem 3, Section 4.3.1).
- We provide an extensive comparison of DCDI to state-of-the-art methods in a wide variety of conditions, including multiple functional forms and types of interventions (Section 4.4).

4.2. Background and related work

4.2.1. Definitions

Causal graphical models. A CGM is defined by a distribution P_X over a random vector $X = (X_1, \cdots, X_d)$ and a DAG $\mathcal{G} = (V, E)$. Each node $i \in V = \{1, \cdots d\}$ is associated with a random variable X_i and each edge $(i,j) \in E$ represents a direct causal relation from variable X_i to X_j. The distribution P_X is Markov to the graph $\mathcal{G}$, which means that the joint distribution can be factorized as such:

$$p(x_1, \cdots, x_d) = \prod_{j=1}^{d} p_j(x_j | x_{\pi_j^{\mathcal{G}}}), \tag{4.2.1}$$

where $\pi_j^{\mathcal{G}}$ is the set of parents of the node j in the graph $\mathcal{G}$, and x_B, for a subset $B \subseteq V$, denotes the entries of the vector x with indices in B. In this work, we assume *causal sufficiency*, i.e., there is no hidden common cause that is causing more than one variable in X (Peters et al., 2017b).

Interventions. In contrast with standard Bayesian Networks, CGMs support interventions. Formally, an intervention on a variable x_j corresponds to replacing its conditional $p_j(x_j|x_{\pi_j^{\mathcal{G}}})$ by a new conditional $\tilde{p}_j(x_j|x_{\pi_j^{\mathcal{G}}})$ in Equation (4.2.1), thus modifying the distribution only locally. Interventions can be performed on multiple variables simultaneously and we call *interventional target* the set $I \subseteq V$ of such variables. When considering more than one intervention, we denote the interventional target of the kth intervention by I_k. Throughout this paper, we assume that the observational distribution (the original distribution without interventions) is observed, and denote it by $I_1 := \emptyset$. We define the *interventional family* by $\mathcal{I} := (I_1, \cdots, I_K)$, where K is the number of interventions (including the observational

setting). Finally, the kth interventional joint density is

$$p^{(k)}(x_1, \cdots, x_d) := \prod_{j \notin I_k} p_j^{(1)}(x_j | x_{\pi_j^{\mathcal{G}}}) \prod_{j \in I_k} p_j^{(k)}(x_j | x_{\pi_j^{\mathcal{G}}}) \,, \tag{4.2.2}$$

where the assumption of causal sufficiency is implicit to this definition of interventions.
Type of interventions. The general type of interventions described in (4.2.2) are called imperfect (or soft, parametric) (Peters et al., 2017b; Eaton et Murphy, 2007; Eberhardt, 2007). A specific case that is often considered is (stochastic) perfect interventions (or hard, structural) (Eberhardt et Scheines, 2007; Yang et al., 2018; Korb et al., 2004) where $p_j^{(k)}(x_j | x_{\pi_j^{\mathcal{G}}}) = p_j^{(k)}(x_j)$ for all $j \in I_k$, thus removing the dependencies with their parents (see Figure 4.1). Real-world examples of these types of interventions include gene knockout/knockdown in biology. Analogous to a perfect intervention, a gene knockout completely suppresses the expression of one gene and removes dependencies to regulators of gene expression. In contrast, a gene knockdown hinders the expression of one gene without removing dependencies with regulators (Zimmer et al., 2019), and is thus an imperfect intervention.

4.2.2. Causal structure learning

In causal structure learning, the goal is to recover the causal DAG $\mathcal{G}$ using samples from P_X and, when available, from interventional distributions. This problem presents two main challenges: 1) the size of the search space is super-exponential in the number of nodes (Chickering, 2003b) and 2) the true DAG is not always identifiable (more severe without interventional data). Methods for this task are often divided into three groups: constraint-based, score-based, and hybrid methods. We briefly review these below.

Constraint-based methods typically rely on conditional independence testing to identify edges in $\mathcal{G}$. The PC algorithm (Spirtes et al., 2000b) is a classical example that works with observational data. It performs conditional independence tests with a conditioning set that increases at each step of the algorithm and finds an equivalence class that satisfies all independencies. Methods that support interventional data include COmbINE (Triantafillou et Tsamardinos, 2015) and HEJ (Hyttinen et al., 2014), which rely on Boolean satisfiability solvers to find a graph that satisfies all constraints. In contrast with our method, these two can account for latent confounders. Another type of constraint-based method exploits the invariance of causal mechanisms across interventional distributions, e.g., ICP (Peters et al., 2016; Heinze-Deml et al., 2018b). As will later be presented in Section 4.3, our loss function also accounts for such invariances.

Score-based methods formulate the problem of estimating the ground truth DAG $\mathcal{G}^*$ by optimizing a score function $\mathcal{S}$ over the space of DAGs. The estimated DAG $\hat{\mathcal{G}}$ is given by

$$\hat{\mathcal{G}} \in \underset{\mathcal{G} \in \text{DAG}}{\arg\max}\, \mathcal{S}(\mathcal{G}) \,. \tag{4.2.3}$$

A typical choice of score in the purely observational setting is the regularized maximum likelihood score:

$$\mathcal{S}(\mathcal{G}) := \max_{\theta} \mathbb{E}_{X \sim P_X} \log f_\theta(X) - \lambda |\mathcal{G}| \,, \tag{4.2.4}$$

where f_θ is a density function parameterized by θ, $|\mathcal{G}|$ is the number of edges in $\mathcal{G}$ and λ is a positive scalar. Since the space of DAGs is enormous, these methods often rely on greedy combinatorial search algorithms. A typical example is GIES (Hauser et Bühlmann, 2012), an adaptation of GES (Chickering, 2003b) to perfect interventions. In contrast with our method, GIES assumes a *linear* gaussian model and optimizes the Bayesian information criterion (BIC) over the space of $\mathcal{I}$-Markov equivalence classes (see Definition 5 in Appendix B.1.1). CAM (Bühlmann et al., 2014) is also a score-based method using greedy search, but it is nonlinear: it assumes an additive noise model where the nonlinear functions are additive. In the original paper, CAM only addresses the observational case where additive noise models are identifiable, however code is available to support perfect interventions.

Hybrid methods combine constraint and score-based approaches. Among these, IGSP (Wang et al., 2017; Yang et al., 2018) is a method that optimizes a score based on conditional independence tests. Contrary to GIES, this method has been shown to be consistent under the faithfulness assumption. Furthermore, this method has recently been extended to support interventions with unknown targets (UT-IGSP) (Squires et al., 2020), which are also supported by our method.

4.2.3. Continuous constrained optimization for structure learning

A new line of research initiated by Zheng et al. (2018b), which serves as basis for our work, reformulates the combinatorial problem of finding the optimal DAG as a continuous constrained-optimization problem, effectively avoiding the combinatorial search. Analogous to standard score-based approaches, these methods rely on a model f_θ parametrized by θ, though θ also encodes the graph $\mathcal{G}$. Central to this class of methods are both the use a *weighted adjacency matrix* $A_\theta \in \mathbb{R}_{\geq 0}^{d \times d}$ (which depends on the parameters of the model) and the acyclicity constraint introduced by Zheng et al. (2018b) in the context of linear models:

$$\operatorname{Tr} e^{A_\theta} - d = 0 \,. \tag{4.2.5}$$

The weighted adjacency matrix encodes the DAG estimator $\hat{\mathcal{G}}$ as $(A_\theta)_{ij} > 0 \iff i \to j \in \hat{\mathcal{G}}$. Zheng et al. (2018b) showed, in the context of linear models, that $\hat{\mathcal{G}}$ is acyclic if and only if the constraint $\operatorname{Tr} e^{A_\theta} - d = 0$ is satisfied. The general optimization problem is then

$$\max_{\theta} \mathbb{E}_{X \sim P_X} \log f_\theta(X) - \lambda \Omega(\theta) \quad \text{s.t.} \quad \operatorname{Tr} e^{A_\theta} - d = 0 \,, \tag{4.2.6}$$

This turns into the BIC score when the expectation is estimated with n samples, the model has one parameter per edge (like in linear models) and $\lambda = \frac{\log n}{2n}$ (Peters et al., 2017b, Section 7.2.2).

where $\Omega(\theta)$ is a regularizing term penalizing the number of edges in $\hat{\mathcal{G}}$. This problem is then approximately solved using an augmented Lagrangian procedure, as proposed by Zheng et al. (2018b). Note that the problem in Equation (4.2.6) is very similar to the one resulting from Equations (4.2.3) and (4.2.4).

Continuous-constrained methods differ in their choice of model, weighted adjacency matrix, and the specifics of their optimization procedures. For instance, NOTEARS (Zheng et al., 2018b) assumes a Gaussian linear model with equal variances where $\theta := W \in \mathbb{R}^{d\times d}$ is the matrix of regression coefficients, $\Omega(\theta) := ||W||_1$ and $A_\theta := W \odot W$ is the weighted adjacency matrix. Several other methods use neural networks to model nonlinear relations via f_θ and have been shown to be competitive with classical methods (Lachapelle et al., 2020; Zheng et al., 2020). Some define the adjacency matrix A as a function of the weights θ of the neural networks (Lachapelle et al., 2020; Zheng et al., 2020), while others decouple θ and A by using a Gumbel-Softmax approach to model the presence/absence of edges (Kalainathan et al., 2018b; Ng et al., 2019). In terms of scoring, most methods rely on maximum likelihood or variants like implicit maximum likelihood (Kalainathan et al., 2018b) and evidence lower bound (Yu et al., 2019a). Zhu et Chen (2020) also rely on the acyclicity constraint, but use reinforcement learning as a search strategy to estimate the DAG. Ke et al. (2019) learn a DAG from data with unknown interventions using a meta-learning approach with a similar form of acyclicity constraint. However, their work covers only discrete distribution and single node interventions. To the best of our knowledge, no work has investigated, in a general manner, the use of continuous-constrained approaches in the context of interventions as we present in the next section.

4.3. DCDI: Differentiable causal discovery from interventional data

In this section, we present a score for imperfect interventions, provide a theorem showing its validity, and show how it can be maximized using the continuous-constrained approach to structure learning. We also provide an extension to unknown interventions without theoretical justification.

4.3.1. A score for imperfect interventions

The model we consider uses neural networks to model conditional densities. Moreover, we encode the DAG $\mathcal{G}$ with a binary adjacency matrix $M^{\mathcal{G}} \in \{0,1\}^{d\times d}$ which acts as a mask on the neural networks inputs. In line with the definition of interventions in Equation (4.2.2),

we model the joint density of the kth intervention by

$$f^{(k)}(x; M^{\mathcal{G}}, \phi) := \prod_{j \notin I_k} \tilde{f}(x_j; \mathrm{NN}(M_j^{\mathcal{G}} \odot x; \phi_j^{(1)})) \prod_{j \in I_k} \tilde{f}(x_j; \mathrm{NN}(M_j^{\mathcal{G}} \odot x; \phi_j^{(k)}))\,, \tag{4.3.1}$$

where $\phi := \{\phi^{(1)}, \cdots, \phi^{(K)}\}$, the NN's are neural networks parameterized by $\phi_j^{(1)}$ or $\phi_j^{(k)}$ (depending on whether j is in the interventional target I_k or not), the operator $\odot$ denotes the Hadamard product (element-wise) and $M_j^{\mathcal{G}}$ denotes the jth column of $M^{\mathcal{G}}$, which enables selecting the parents of node j in the graph $\mathcal{G}$. The neural networks output the parameters of a density function $\tilde{f}$, which in principle, could be any density. We experiment with Gaussian distributions and more expressive normalizing flows (see Section 4.3.4).

We propose maximizing the following regularized maximum log-likelihood score

$$\mathcal{S}_{\mathrm{int}}(\mathcal{G}) := \max_{\phi} \sum_{k=1}^{K} \mathbb{E}_{X \sim p^{(k)}} \log f^{(k)}(X; M^{\mathcal{G}}, \phi) - \lambda |\mathcal{G}|\,, \tag{4.3.2}$$

where $p^{(k)}$ stands for the kth ground truth interventional distribution from which the data is sampled. Intuitively, this score favors graphs in which a conditional $p(x_j | x_{\pi_j^{\mathcal{G}}})$ is invariant across all interventional distributions in which x_j is not a target, i.e., $j \notin I_k$.

We now present our theoretical result (see Appendix B.1.2 for the proof). This theorem states that, under appropriate assumptions, maximizing $\mathcal{S}_{\mathrm{int}}(\mathcal{G})$ yields an estimated DAG $\hat{\mathcal{G}}$ that is $\mathcal{I}$-Markov equivalent to the true DAG $\mathcal{G}^*$ (see Definition 5 in Appendix B.1.1).

Theorem 3. Let $\mathcal{G}^*$ be the ground truth DAG and $\hat{\mathcal{G}} \in \arg\max_{\mathcal{G} \in \mathrm{DAG}} \mathcal{S}_{\mathrm{int}}(\mathcal{G})$. Under Assumptions 1 & 2 (Appendix B.1.2) and for $\lambda > 0$ small enough, $\hat{\mathcal{G}}$ is $\mathcal{I}$-Markov equivalent to $\mathcal{G}^*$.

Assumption 1 requires that the model is expressive enough while Assumption 2 requires that the ground truth distributions are $\mathcal{I}$-faithful to the ground truth graph (generalization of the standard faithfulness assumption to interventions).

To interpret this result, note that the $\mathcal{I}$-Markov equivalence class of $\mathcal{G}^*$ tends to get smaller as we add interventional targets to the interventional family $\mathcal{I}$. As an example, when $\mathcal{I} = (\emptyset, \{1\}, \cdots, \{d\})$, i.e., when each node is individually targeted by an intervention, $\mathcal{G}^*$ is alone in its class and, consequently, $\hat{\mathcal{G}} = \mathcal{G}^*$. See Corollary 1 in Appendix B.1.1 for more details.

Perfect interventions. The score $\mathcal{S}_{\mathrm{int}}(\mathcal{G})$ can be modified to work with perfect interventions, i.e., where the targeted nodes are completely disconnected from their parents. The idea is simple and relies on the fact that the conditionals targeted by the intervention in Equation (4.3.1) do not depend on the graph $\mathcal{G}$ anymore. This means that these terms can be removed without affecting the maximization w.r.t. $\mathcal{G}$. We use this version of the score when experimenting with perfect interventions.

4.3.2. A continuous-constrained formulation

To allow for gradient-based stochastic optimization, we follow Kalainathan et al. (2018b); Ng et al. (2019) and treat the adjacency matrix $M^{\mathcal{G}}$ as *random*, where the entries $M_{ij}^{\mathcal{G}}$ are independent Bernoulli variables with success probability $\sigma(\alpha_{ij})$ (σ is the sigmoid function) and α_{ij} is a scalar parameter. We group these α_{ij}'s into a matrix $\Lambda \in \mathbb{R}^{d\times d}$. We then replace the score $\mathcal{S}_{\text{int}}(\mathcal{G})$ (4.3.2) with the following relaxation:

$$\hat{\mathcal{S}}_{\text{int}}(\Lambda) := \max_{\phi} \mathop{\mathbb{E}}_{M\sim\sigma(\Lambda)} \left[\sum_{k=1}^{K} \mathop{\mathbb{E}}_{X\sim p^{(k)}} \log f^{(k)}(X; M, \phi) - \lambda ||M||_0 \right] , \tag{4.3.3}$$

where we dropped the $\mathcal{G}$ superscript in M to lighten notation. This score tends asymptotically to $\mathcal{S}_{\text{int}}(\mathcal{G})$ as $\sigma(\Lambda)$ concentrates more and more its mass on $\mathcal{G}$. While the expectation of the log-likelihood term is intractable, the expectation of the regularizing term simply evaluates to $\lambda||\sigma(\Lambda)||_1$. This score can then be maximized under the acyclicity constraint presented in Section 4.2.3:

$$\max_{\Lambda} \hat{\mathcal{S}}_{\text{int}}(\Lambda) \quad \text{s.t.} \quad \operatorname{Tr} e^{\sigma(\Lambda)} - d = 0 \,. \tag{4.3.4}$$

This problem presents two main challenges: it is a constrained problem and it contains intractable expectations. As proposed by Zheng et al. (2018b), we rely on the *augmented Lagrangian* procedure to optimize ϕ and Λ jointly under the acyclicity constraint. This procedure transforms the constrained problem into a sequence of unconstrained subproblems which can themselves be optimized via a standard stochastic gradient descent algorithm for neural networks such as RMSprop. The procedure should converge to a stationary point of the original constrained problem (which is not necessarily the global optimum due to the non-convexity of the problem). In Appendix B.2.3, we give details on the augmented Lagrangian procedure and show the learning process in details with a concrete example.

The gradient of the likelihood part of $\hat{\mathcal{S}}_{\text{int}}(\Lambda)$ w.r.t. Λ is estimated using the Straight-Through Gumbel estimator which amounts to using Bernoulli samples in the forward pass and Gumbel-Softmax samples in the backward pass which can be differentiated w.r.t. Λ via the reparametrization trick (Jang et al., 2017; Maddison et al., 2017). This approach was already shown to give good results in the context of continuous optimization for causal discovery in the purely observational case (Ng et al., 2019; Kalainathan et al., 2018b). We emphasize that our approach belongs to the general framework presented in Section 4.2.3 where the global parameter θ is $\{\phi, \Lambda\}$, the weighted adjacency matrix A_θ is $\sigma(\Lambda)$ and the regularizing term $\Omega(\theta)$ is $||\sigma(\Lambda)||_1$.

In practice, we observe that $\sigma(\Lambda)$ tends to become deterministic as we optimize.

4.3.3. Unknown interventions

Until now we have assumed that the interventional targets I_k are known. For the case where they are unknown, we propose a simple modification to our score by adding a random binary matrix $R \in \{0,1\}^{K\times d}$, where $R_{kj} = 1$ means that X_j is a target in I_k. Similarly to the matrix M, each entry R_{kj} follows independent Bernoulli distribution with probability $\sigma(\beta_{kj})$ where β_{kj} are parameters that are learned. The likelihood is then:

$$f^{(k)}(x; M, R, \phi) := \prod_{j=1}^{d} \tilde{f}(x_j; \mathrm{NN}(M_j \odot x; \phi_j^{(1)}))^{1-R_{kj}} \tilde{f}(x_j; \mathrm{NN}(M_j \odot x; \phi_j^{(k)}))^{R_{kj}} . \tag{4.3.5}$$

The resulting score is close to (4.3.3), but the expectation is taken w.r.t. to M and R. Also, a regularization term $-\lambda_R||R||_0$ is added to encourage the sparsity of the learned interventional targets. Similarly to Λ, the Straight-Through Gumbel estimator is used to estimate the gradient of the score w.r.t. the parameters β_{kj}. For perfect interventions, we adapt this score by completely masking the input of the neural networks under interventions. In related work, Ke et al. (2019) also use neural networks, but they support only *single* unknown target interventions and they estimate the gradient w.r.t. Λ using the log-trick which is known to have high variance (Rezende et al., 2014) compared to reparameterized gradient (Maddison et al., 2017).

4.3.4. DCDI with normalizing flows

In this section, we describe how the scores presented in Sections 4.3.2 & 4.3.3 can accommodate powerful density approximators. In the purely observational setting, very expressive models usually hinder identifiability, but given enough interventions, this is not a problem anymore. There are many possibilities when it comes to the choice of the density function $\tilde{f}$. In this paper, we experimented with simple Gaussian distributions as well as *normalizing flows* (Rezende et Mohamed, 2015) which can represent complex causal relationships, e.g., multi-modal distributions that can occur in the presence of latent variables that are parent of only one variable.

A normalizing flow $\tau(\cdot; \omega)$ is an invertible function (e.g., a neural network) parameterized by ω with a tractable Jacobian, which can be used to model complex densities by transforming a simple random variable via the change of variable formula:

$$\tilde{f}(z; \omega) := \left|\det \left(\frac{\partial \tau(z; \omega)}{\partial z}\right)\right| p(\tau(z; \omega)) , \tag{4.3.6}$$

where $\frac{\partial \tau(z;\omega)}{\partial z}$ is the Jacobian matrix of $\tau(\cdot; \omega)$ and $p(\cdot)$ is a simple density function, e.g., a Gaussian. The function $\tilde{f}(\cdot; \omega)$ can be plugged directly into the scores presented earlier by letting the neural networks $\mathrm{NN}(\cdot; \phi_j^{(k)})$ output the parameter ω_j of the normalizing flow τ_j for each variable x_j. In our implementation, we use *deep sigmoidal flows* (DSF), a specific

instantiation of normalizing flows which is a universal density approximator (Huang et al., 2018b). Details about DSF are relayed to Appendix B.2.2.

4.4. Experiments

We tested DCDI with Gaussian densities (DCDI-G) and with normalizing flows (DCDI-DSF) on a real-world data set and several synthetic data sets. The real-world task is a flow cytometry data set from Sachs et al. (2005b). Our results, reported in Appendix B.3.1, show that our approach performs comparably to state-of-the-art methods. In this section, we focus on synthetic data sets, since these allow for a more systematic comparison of methods against various factors of variation (type of interventions, graph size, density, type of mechanisms).

We consider synthetic data sets with three interventional settings: perfect/known, imperfect/known, and perfect/unknown. Each data set has one of the three different types of causal mechanisms: i) linear Squires et al. (2020), ii) nonlinear additive noise model (ANM) Bühlmann et al. (2014), and iii) nonlinear with non-additive noise using neural networks (NN) Kalainathan et al. (2018b). For each data set type, graphs vary in size ($d = 10$ or 20) and density ($e = 1$ or 4 where $e \cdot d$ is the average number of edges). For conciseness, we present results for 20-node graphs in the main text and report results on 10-node graphs in Appendix B.3.6; conclusions are similar for all sizes. For each condition, ten graphs are sampled with their causal mechanisms and then observational and interventional data are generated. Each data set has 10000 samples uniformly distributed in the different interventional settings. A total of d interventions were performed, each by sampling up to $0.1d$ target nodes. For more details on the generation process, see Appendix B.2.1.

Most methods have an hyperparameter controlling DAG sparsity. Although performance is sensible to this hyperparameter, many papers do not specify how it was selected. For score-based methods (GIES, CAM and DCDI), we select it by maximizing the held-out likelihood as explained in Appendix B.2.5 (without using the ground truth DAG). In contrast, for constraint-based methods (IGSP and UT-IGSP), we use a fixed cutoff parameter ($\alpha = 1e^{-3}$) that yielded overall good results since they do not have a likelihood model to evaluate on held-out data. We report additional results with different cutoff values in Appendix B.3.6. For these methods, we always pick the most advantageous independence test: partial correlation test for Gaussian linear data and KCI-test (Zhang et al., 2011) for nonlinear data.

The performance of each method is assessed by two metrics on the estimated graph compared to the ground truth graph: i) the *structural Hamming distance* (SHD) which is simply the number of edges that differ between two DAGs (either reversed, missing or superfluous) and ii) the *structural interventional distance* (SID) which assesses how two DAGs differ with respect to their causal inference statements (Peters et Bühlmann, 2015b). To further demonstrate the benefits of using interventional data and the usefulness of our

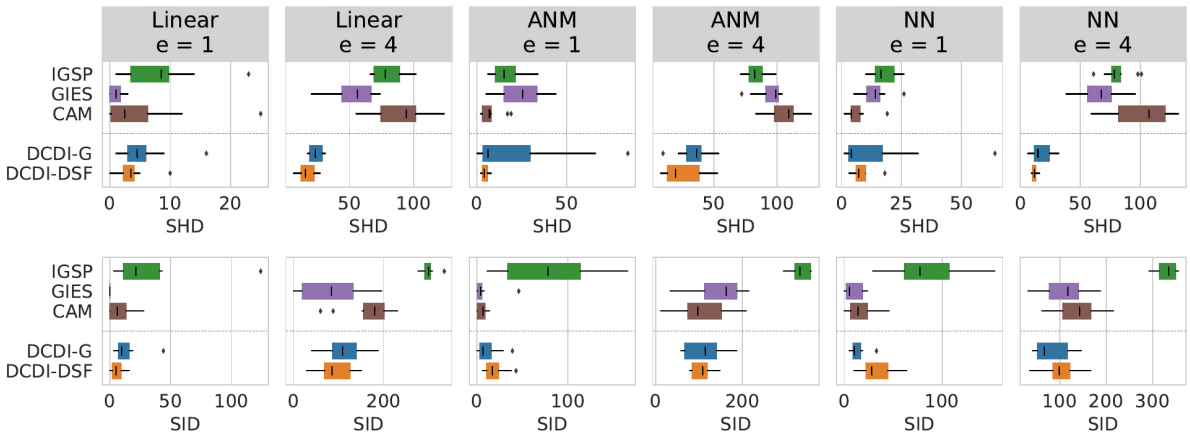


Fig. 4.2. Perfect interventions. SHD and SID (lower is better) for 20-node graphs

new objective, we report an ablation study in B.3.4. Our implementation is available here and additional information about the baseline methods is provided in Appendix B.2.4.

4.4.1. Results for different intervention types

Perfect interventions. We compare our methods to GIES (Hauser et Bühlmann, 2012), a modified version of CAM (Bühlmann et al., 2014) that support interventions and IGSP (Wang et al., 2017). The conditionals of targeted nodes were replaced by the marginal $\mathcal{N}(2,1)$ similarly to Hauser et Bühlmann (2012); Squires et al. (2020). Boxplots for SHD and SID over 10 graphs are shown in Figure 4.2. For all conditions, DCDI-G and DCDI-DSF shows competitive results in term of SHD and SID. For graphs with a higher number of average edges, DCDI-G and DCDI-DSF outperform all methods. GIES often shows the best performance for the linear data set, which is not surprising given that it makes the right assumptions, i.e., linear functions with Gaussian noise.

Imperfect interventions. Our conclusions are similar to the perfect intervention setting. As shown in Figure 4.3, DCDI-G and DCDI-DSF show competitive results and outperform other methods for graphs with a higher connectivity. The nature of the imperfect interventions are explained in Appendix B.2.1.

Perfect unknown interventions. We compare to UT-IGSP (Squires et al., 2020), an extension of IGSP that deal with unknown interventions. The data used are the same as in the perfect intervention setting, but the intervention targets are hidden. Results are shown in Figure 4.4. Except for linear data sets with sparse graphs, DCDI-G and DCDI-DSF show an overall better performance than UT-IGSP.

Summary. For all intervention settings, DCDI has overall the best performance. In Appendix B.3.5, we show similar results for different types of perfect/imperfect interventions. While the advantage of DCDI-DSF over DCDI-G is marginal, it might be explained by the fact that the densities can be sufficiently well modeled by DCDI-G. In Appendix B.3.2, we

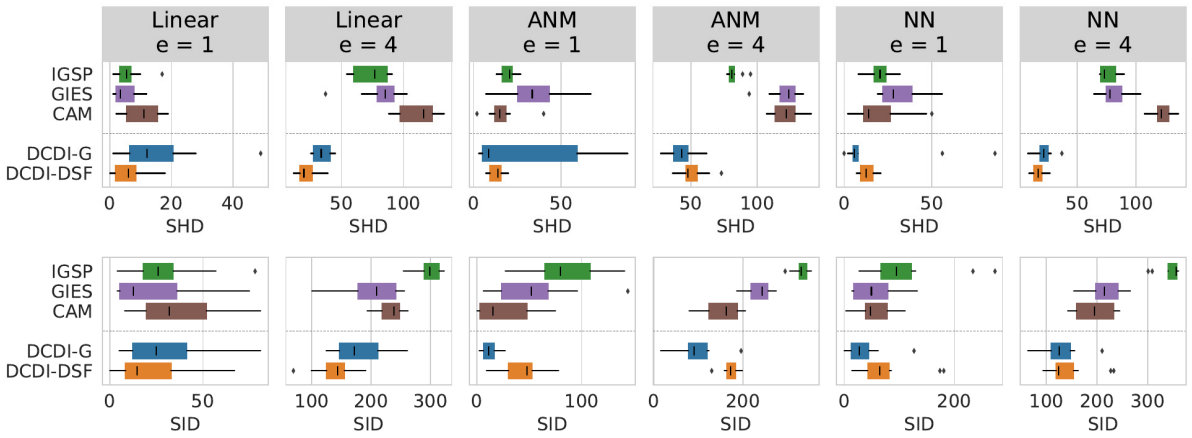


Fig. 4.3. Imperfect interventions. SHD and SID for 20-node graphs

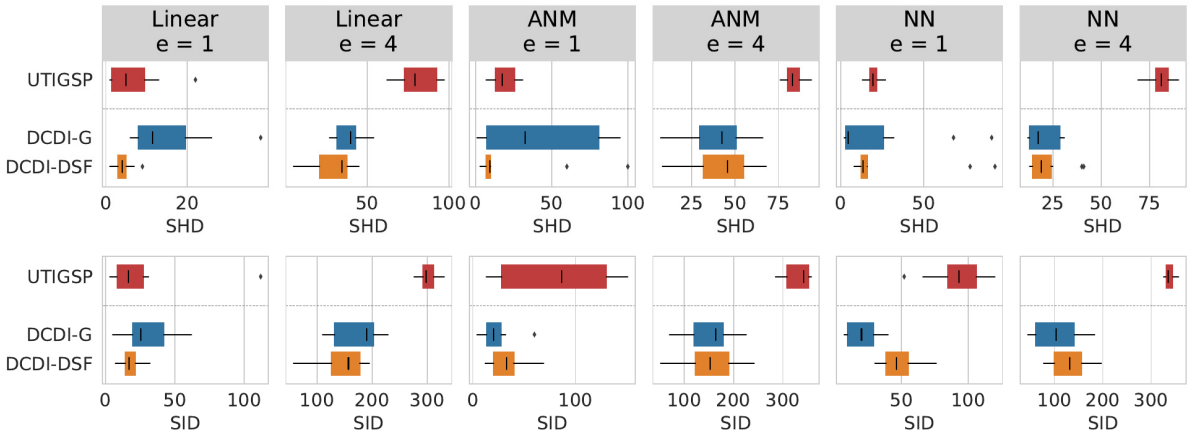


Fig. 4.4. Unknown interventions. SHD and SID for 20-node graphs

illustrate the benefits of using high capacity estimators in this context. We show cases where methods without sufficient capacity, such as DCDI-G, fail to detect the right causal direction, whereas DCDI-DSF systematically succeeds.

4.4.2. Scalability experiments

So far the experiments focused on moderate size data sets, both in terms of number of variables (10 or 20) and number of examples ($\approx 10^4$). In Appendix B.3.3, we compare the running times of DCDI to those of other methods on graphs of up to 100 nodes and on data sets of up to 1 million examples.

The augmented Lagrangian procedure on which DCDI relies requires the computation of the matrix exponential at each gradient step, which costs $O(d^3)$. We found this does not prevent DCDI from being applied to 100 nodes graphs. Several constraint-based methods use kernel-based conditional independence tests (Zhang et al., 2011; Fukumizu et al., 2008), which scale poorly with the number of examples. For example, KCI-test scales in $O(n^3)$ (Strobl et al., 2019) and HSIC in $O(n^2)$ (Zhang et al., 2018). On the other hand,

DCDI is not greatly affected by the sample size since it relies on stochastic gradient descent which is known to scale well with the data set size (Bottou, 2010). Our comparison shows that, among all considered methods, DCDI is the only one supporting nonlinear relationships that can scale to as much as one million examples. We believe that this can open the way to new applications of causal discovery where data is abundant.

4.5. Conclusion

We proposed a general continuous-constrained method for causal discovery which can leverage various types of interventional data as well as expressive neural architectures, such as normalizing flows. This approach is rooted in a sound theoretical framework and is competitive with other state-of-the-art algorithms on real and simulated data sets, both in terms of graph recovery and scalability. This work opens interesting opportunities for future research. One direction is to extend DCDI to time-series data, where non-stationarities can be modeled as unknown interventions (Pfister et al., 2019). Another exciting direction is to learn representations of variables across multiple systems that could serve as prior knowledge for causal discovery in low data settings.

Chapitre 5

Conclusion

Les deux travaux présentent des méthodes pour apprendre des modèles causaux à partir de données observationnelles ou interventionnelles. Ces méthodes présentent des résultats très compétitifs par rapport aux méthodes traditionnelles sur un ensemble varié de tâches, tant au niveau de leur performance que de leur adaptabilité à des graphes et échantillons de grandes tailles. L'utilisation de la contrainte continue d'acyclicité avec les réseaux de neurones artificiels ouvre une nouvelle approche au problème de l'apprentissage de modèle causal. Cette nouvelle approche recèle de possibilités pour aborder des problèmes variés.

Par exemple, récemment, Pamfil et al. (2020) ont proposé une extension de NOTEARS de Zheng et al. (2018b), qui utilise la même contrainte d'acyclicité, pour le problème d'apprentissage de modèles causaux à partir de données séquentielles. Il semble naturel d'étendre cette méthode aux relations non-linéaires en utilisant des réseaux de neurones artificiels comme proposé dans GraN-DAG, surtout que dans l'absence de variables confondantes, le problème d'identifiabilité est moins important (Peters et al., 2017b). Mais, plus intéressant que cela est le problème de données séquentielles contenant des interventions cachées (récemment considéré par Pfister et al. (2019)). Contrairement aux données non-séquentielles avec cibles connues, ce type de données est courant et facile à obtenir.

Une autre avenue de recherche intéressante, qui implique aussi des réseaux de neurones, a été introduite par Lopez-Paz et al. (2015). Lopez-Paz et al. (2015) pose le problème d'apprentissage de modèles causaux comme un problème classique d'apprentissage supervisé. Ils supposent qu'ils ont accès à certaines données, disons des paires de cause et effet, dont la vraie direction causale est connue. Ils utilisent ces données comme données d'entraînement et ils peuvent ainsi généraliser à de nouvelles paires. L'intérêt de cette méthode est qu'il n'est pas nécessaire d'avoir de suppositions sur la forme fonctionnel des relations causales. Cette forme sera apprise par un réseau de neurones. Lopez-Paz et al. (2015), Lopez-Paz et al. (2017) et Hill et al. (2019) ont obtenus des résultats prometteurs sur des paires causales provenant de jeux de données réels (les paires de Tubingen, photographies et données en génomique).

Cependant, leur approche ne tire pas profit d'informations extérieures sur les différentes variables d'intérêt. Pour donner un exemple concret, en génomique, on cherche parfois à apprendre des réseaux de régulation génétique à partir de mesures d'expression des gènes. Or, dans ce cas, l'information extérieure pourrait simplement être la séquence des gènes. À partir de réseaux de neurones artificiels, il pourrait être possible d'apprendre une représentation des variables pour apprendre en quelque sorte une "disposition causale" (Lopez-Paz et al., 2017) des variables. Pour reprendre l'exemple des gènes, certaines séquences sont connues pour principalement en réguler d'autres. En apprenant une représentation des différentes variables, il serait possible d'inférer des graphes causaux en utilisant des jeux de données moins volumineux. La génomique est encore une fois un candidat idéal puisqu'il existe beaucoup d'études sur l'interaction de certaines paires de gènes, mais peu sur des réseaux impliquant un grand nombre de gènes.

Il est indéniable que le mariage du domaine de l'apprentissage automatique et de la causalité est prometteur à bien des niveaux et est une avenue qu'il vaut la peine de considérer dans la quête d'une intelligence artificielle forte.

Références bibliographiques

M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, et X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

J. Alayrac, P. Bojanowski, N. Agrawal, J. Sivic, I. Laptev, et S. Lacoste-Julien. Learning from narrated instruction videos. *TPAMI*, 40(9):2194–2208, Sep. 2018.

M. Arjovsky, L. Bottou, I. Gulrajani, et D. Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

A.-L. Barabási. Scale-free networks: a decade and beyond. *Science*, 2009.

A.-L. Barabási et R. Albert. Emergence of scaling in random networks. *Science*, 1999.

S. Beery, G. Van Horn, et P. Perona. Recognition in terra incognita. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 456–473, 2018.

J. Bergstra et Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 2012.

D. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

C. M. Bishop. *Pattern recognition and machine learning*. springer, 2006.

M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*. 2010.

P. Bühlmann, J. Peters, et J. Ernest. CAM: Causal additive models, high-dimensional order search and penalized regression. *Annals of Statistics*, 2014.

P. Bühlmann, J. Peters, et J. Ernest. Cam: Causal additive models, high-dimensional order search and penalized regression. *The Annals of Statistics*, 2014.

D. Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 2003a.

D. M. Chickering. Optimal structure identification with greedy search. *Journal of machine learning research*, 3(Nov):507–554, 2002.

D. M. Chickering. Optimal structure identification with greedy search. In *Journal of Machine Learning Research*, 2003b.

A. Clauset, C. Shalizi, et M. Newman. Power-law distributions in empirical data. *SIAM review*, 2009.

A. Constantinou. Evaluating structure learning algorithms with a balanced scoring function. *arXiv preprint arXiv:1905.12666*, 2019.

J. Cussens. Bayesian network learning with cutting planes. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence*, 2011.

A. Dixit, O. Parnas, B. Li, J. Chen, C. P. Fulco, L. Jerby-Arnon, N. D. Marjanovic, D. Dionne, T. Burks, R. Raychndhury, T. M. Adamson, B. Norman, E. S. Lander, J. S. Weissman, N. Friedman, et A. Regev. Perturb-seq: dissecting molecular circuits with scalable single-cell rna profiling of pooled genetic screens. *Cell*, 2016.

D. Eaton et K. Murphy. Exact bayesian structure learning from uncertain interventions. In *Artificial intelligence and statistics*, 2007.

F. Eberhardt. Causation and intervention. *Unpublished doctoral dissertation, Carnegie Mellon University*, 2007.

F. Eberhardt. Almost Optimal Intervention Sets for Causal Discovery. In *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence*, 2008.

F. Eberhardt. Almost optimal intervention sets for causal discovery. *arXiv preprint arXiv:1206.3250*, 2012.

F. Eberhardt et R. Scheines. Interventions and causal inference. *Philosophy of Science*, 2007.

F. Eberhardt, C. Glymour, et R. Scheines. On the Number of Experiments Sufficient and in the Worst Case Necessary to Identify all Causal Relations among N Variables. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, 2005.

F. Eberhardt, C. Glymour, et R. Scheines. On the number of experiments sufficient and in the worst case necessary to identify all causal relations among n variables. *arXiv preprint arXiv:1207.1389*, 2012.

A. Falcon. Aristotle on causality. 2006.

K. Fukumizu, A. Gretton, X. Sun, et B. Schölkopf. Kernel measures of conditional dependence. In *Advances in neural information processing systems*, 2008.

M. Germain, K. Gregor, I. Murray, et H. Larochelle. Made: Masked autoencoder for distribution estimation. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

P. Geurts, D. Ernst, et L. Wehenkel. Extremely randomized trees. *Machine learning*, 2006.

X. Glorot et Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010a.

X. Glorot et Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, 2010b.

X. Glorot, A. Bordes, et Y. Bengio. Deep sparse rectifier neural networks. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, 2011.

I. Goodfellow, Y. Bengio, et A. Courville. *Deep Learning*. MIT Press, 2016a.

I. Goodfellow, Y. Bengio, et A. Courville. *Deep learning*. MIT press, 2016b.

O. Goudet, D. Kalainathan, P. Caillou, D. Lopez-Paz, I. Guyon, et M. Sebag. Learning Functional Causal Models with Generative Neural Networks. In *Explainable and Interpretable Models in Computer Vision and Machine Learning*. Springer International Publishing, 2018.

A. Hauser et P. Bühlmann. Characterization and greedy learning of interventional markov equivalence classes of directed acyclic graphs. *Journal of Machine Learning Research*, 13 (Aug):2409–2464, 2012.

C. Heinze-Deml, M. H. Maathuis, et N. Meinshausen. Causal structure learning. *Annual Review of Statistics and Its Application*, 2018a.

C. Heinze-Deml, J. Peters, et N. Meinshausen. Invariant causal prediction for nonlinear models. *Journal of Causal Inference*, 2018b.

S. Hill, C. J. Oates, D. A. Blythe, et S. Mukherjee. Causal learning via manifold regularization. 2019.

K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

B. Huang, K. Zhang, Y. Lin, B. Schölkopf, et C. Glymour. Generalized score functions for causal discovery. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018a.

C.-W. Huang, D. Krueger, A. Lacoste, et A. Courville. Neural autoregressive flows. In *Proceedings of the 35th International Conference on Machine Learning*, 2018b.

A. Hyttinen, F. Eberhardt, et M. Järvisalo. Constraint-based causal discovery: Conflict resolution with answer set programming. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, 2014.

T. Jaakkola, D. Sontag, A. Globerson, et M. Meila. Learning Bayesian Network Structure using LP Relaxations. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, 2010.

E. Jang, S. Gu, et B. Poole. Categorical reparameterization with gumbel-softmax. *Proceedings of the 34th International Conference on Machine Learning*, 2017.

D. Kalainathan, O. Goudet, I. Guyon, D. Lopez-Paz, et M. Sebag. Sam: Structural agnostic model, causal discovery and penalized adversarial learning. *arXiv preprint arXiv:1803.04929*, 2018a.

D. Kalainathan, O. Goudet, I. Guyon, D. Lopez-Paz, et M. Sebag. Sam: Structural agnostic model, causal discovery and penalized adversarial learning. *arXiv preprint arXiv:1803.04929*, 2018b.

K. Kawaguchi. Deep learning without poor local minima. In *Advances in neural information processing systems*, pages 586–594, 2016.

N. R. Ke, O. Bilaniuk, A. Goyal, S. Bauer, H. Larochelle, C. Pal, et Y. Bengio. Learning neural causal models from unknown interventions. *arXiv preprint arXiv:1910.01075*, 2019.

D. P. Kingma et J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

D. Koller et N. Friedman. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. MIT Press, 2009.

K. B. Korb, L. R. Hope, A. E. Nicholson, et K. Axnick. Varieties of causal intervention. In *Pacific Rim International Conference on Artificial Intelligence*, 2004.

S. Lachapelle, P. Brouillard, T. Deleu, et S. Lacoste-Julien. Gradient-based neural DAG learning. In *Proceedings of the 8th International Conference on Learning Representations*, 2020.

S. Larivée. *L'intelligence: Approches biocognitives, développementales et contemporaines*. ERPI, 2006.

D. Lopez-Paz, K. Muandet, B. Schölkopf, et I. Tolstikhin. Towards a learning theory of cause-effect inference. In *International Conference on Machine Learning*, pages 1452–1461, 2015.

D. Lopez-Paz, R. Nishihara, S. Chintala, B. Scholkopf, et L. Bottou. Discovering causal signals in images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6979–6987, 2017.

C. J. Maddison, A. Mnih, et Y. W. Teh. The concrete distribution: A continuous relaxation of discrete random variables. *Proceedings of the 34th International Conference on Machine Learning*, 2017.

S. Magliacane, T. van Ommen, T. Claassen, S. Bongers, P. Versteeg, et J. Mooij. Domain adaptation by using causal inference to predict invariant conditional distributions. In *Advances in Neural Information Processing Systems 31*, 2018.

G. Marcus. Deep learning: A critical appraisal. *arXiv preprint arXiv:1801.00631*, 2018.

M. Marvin et A. P. Seymour. Perceptrons, 1969.

J. L. McClelland, D. E. Rumelhart, P. R. Group, et al. Parallel distributed processing. *Explorations in the Microstructure of Cognition*, 2:216–271, 1986.

W. S. McCulloch et W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

G. F. Montufar, R. Pascanu, K. Cho, et Y. Bengio. On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, pages 2924–2932, 2014.

J. M. Mooij, S. Magliacane, et T. Claassen. Joint causal inference from multiple contexts. *arXiv preprint arXiv:1611.10351*, 2016.

K. P. Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

I. Ng, Z. Fang, S. Zhu, Z. Chen, et J. Wang. Masked gradient-based causal structure learning. *arXiv preprint arXiv:1910.08527*, 2019.

R. Pamfil, N. Sriwattanaworachai, S. Desai, P. Pilgerstorfer, K. Georgatzis, P. Beaumont, et B. Aragam. Dynotears: Structure learning from time-series data. In *International Conference on Artificial Intelligence and Statistics*, pages 1595–1605, 2020.

A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, et A. Lerer. Automatic differentiation in pytorch. 2017.

J. Pearl. Causal diagrams for empirical research. *Biometrika*, 82(4):669–688, 1995.

J. Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, 2nd edition, 2009a.

J. Pearl. *Causality*. Cambridge university press, 2009b.

J. Pearl. Theoretical impediments to machine learning with seven sparks from the causal revolution. *arXiv preprint arXiv:1801.04016*, 2018.

J. Pearl. The seven tools of causal inference, with reflections on machine learning. *Commun. ACM*, 2019a.

J. Pearl. The seven tools of causal inference, with reflections on machine learning. *Communications of the ACM*, 62(3):54–60, 2019b.

J. Peters et P. Bühlman. Identifiability of Gaussian structural equation models with equal error variances. *Biometrika*, 2014.

J. Peters et P. Bühlmann. Structural intervention distance (sid) for evaluating causal graphs. *Neural Computation*, 2013.

J. Peters et P. Bühlmann. Identifiability of gaussian structural equation models with equal error variances. *Biometrika*, 101(1):219–228, 2014.

J. Peters et P. Bühlmann. Structural intervention distance (SID) for evaluating causal graphs. *Neural Computation*, 2015a.

J. Peters et P. Bühlmann. Structural intervention distance for evaluating causal graphs. *Neural computation*, 27(3):771–799, 2015b.

J. Peters, D. Janzing, et B. Schölkopf. Causal inference on discrete data using additive noise models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011.

J. Peters, J. M. Mooij, D. Janzing, et B. Schölkopf. Causal discovery with continuous additive noise models. *Journal of Machine Learning Research*, 2014a.

J. Peters, J. M. Mooij, D. Janzing, et B. Schölkopf. Causal discovery with continuous additive noise models. *The Journal of Machine Learning Research*, 15(1):2009–2053, 2014b.

J. Peters, P. Bühlmann, et N. Meinshausen. Causal inference by using invariant prediction: identification and confidence intervals. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 2016.

J. Peters, D. Janzing, et B. Schölkopf. *Elements of Causal Inference - Foundations and Learning Algorithms*. MIT Press, 2017a.

J. Peters, D. Janzing, et B. Schölkopf. *Elements of causal inference: foundations and learning algorithms*. MIT press, 2017b.

N. Pfister, P. Bühlmann, et J. Peters. Invariant causal prediction for sequential data. *Journal of the American Statistical Association*, 114(527):1264–1276, 2019.

T. A. Poggio, K. Kawaguchi, Q. Liao, B. Miranda, L. Rosasco, X. Boix, J. Hidary, et H. Mhaskar. Theory of deep learning III: explaining the non-overfitting puzzle. *arXiv preprint arXiv:1801.00173*, 2018.

L. Prechelt. Early stopping - but when? In *Neural Networks: Tricks of the Trade, volume 1524 of LNCS, chapter 2*, 1997.

J. Ramsey, M. Glymour, R. Sanchez-Romero, et C. Glymour. A million variables and more: the fast greedy equivalence search algorithm for learning high-dimensional graphical causal models, with an application to functional magnetic resonance images. *I. J. Data Science and Analytics*, 2017.

D. J. Rezende et S. Mohamed. Variational inference with normalizing flows. *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

D. J. Rezende, S. Mohamed, et D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning*, 2014.

F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

S. Russell et P. Norvig. Artificial intelligence: a modern approach. 2002.

K. Sachs, O. Perez, D. Pe'er, D. Lauffenburger, et G. Nolan. Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, 2005a.

K. Sachs, O. Perez, D. Pe'er, D. A. Lauffenburger, et G. P. Nolan. Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, 2005b.

L. Sagun, V. U. Guney, G. B. Arous, et Y. LeCun. Explorations on high dimensional landscapes. *arXiv preprint arXiv:1412.6615*, 2014.

B. Schölkopf. Causality for machine learning. *arXiv preprint arXiv:1911.10500*, 2019.

R. D. Shah et J. Peters. The hardness of conditional independence testing and the generalised covariance measure. *arXiv preprint arXiv:1804.07203*, 2018.

S. Shimizu, P. Hoyer, A. Hyvärinen, et A. Kerminen. A linear non-gaussian acyclic model for causal discovery. *Journal of Machine Learning Research*, 2006a.

S. Shimizu, P. O. Hoyer, A. Hyvärinen, et A. Kerminen. A linear non-gaussian acyclic model for causal discovery. *Journal of Machine Learning Research*, 7(Oct):2003–2030, 2006b.

C. Shorten et T. M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, 2019.

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

E. H. Simpson. The interpretation of interaction in contingency tables. *Journal of the Royal Statistical Society: Series B (Methodological)*, 13(2):238–241, 1951.

S. Singh, A. Okun, et A. Jackson. Artificial intelligence: Learning to play go from scratch. *Nature*, 550(7676):336–337, 2017.

L. Solus, Y. Wang, L. Matejovicova, et C. Uhler. Consistency guarantees for permutation-based causal inference algorithms. *arXiv preprint arXiv:1702.03530*, 2017.

P. Spirtes, C. Glymour, et R. Scheines. *Causation, Prediction, and Search*. MIT press, 2000a.

P. Spirtes, C. N. Glymour, R. Scheines, et D. Heckerman. *Causation, prediction, and search*. MIT press, 2000b.

C. Squires, Y. Wang, et C. Uhler. Permutation-based causal structure learning with unknown intervention targets. *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence*, 2020.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, et R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

E. V. Strobl, K. Zhang, et S. Visweswaran. Approximate kernel-based conditional independence tests for fast non-parametric causal discovery. *Journal of Causal Inference*, 2019.

J. Tian et J. Pearl. *Studies in Causal Reasoning and Learning*. PhD thesis, 2002. AAI3070088.

T. Tieleman et G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012a.

T. Tieleman et G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012b.

S. Triantafillou et I. Tsamardinos. Constraint-based causal discovery from multiple interventions over overlapping variable sets. *Journal of Machine Learning Research*, 2015.

T. Van den Bulcke, et al. SynTReN: a generator of synthetic gene expression data for design and analysis of structure learning algorithms. 2006.

T. Verma et J. Pearl. *Equivalence and synthesis of causal models*. UCLA, Computer Science Department, 1991.

O. Vinyals, I. Babuschkin, J. Chung, M. Mathieu, M. Jaderberg, W. M. Czarnecki, A. Dudzik, A. Huang, P. Georgiev, R. Powell, et al. Alphastar: Mastering the real-time strategy game starcraft ii. *DeepMind blog*, page 2, 2019.

C. M. Walker et A. Gopnik. Causality and imagination. *The Oxford handbook of the development of imagination*, page 342, 2013.

Y. Wang, L. Solus, K. Yang, et C. Uhler. Permutation-based causal inference algorithms with interventions. In *Advances in Neural Information Processing Systems*, 2017.

K. D. Yang, A. Katcoff, et C. Uhler. Characterizing and learning equivalence classes of causal DAGs under interventions. *Proceedings of the 35th International Conference on Machine Learning*, 2018.

Y. Yu, J. Chen, T. Gao, et M. Yu. DAG-GNN: DAG structure learning with graph neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, 2019a.

Y. Yu, J. Chen, T. Gao, et M. Yu. DAG-GNN: DAG structure learning with graph neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, 2019b.

K. Zhang et A. Hyvärinen. On the identifiability of the post-nonlinear causal model. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, 2009.

K. Zhang, J. Peters, D. Janzing, et B. Schölkopf. Kernel-based conditional independence test and application in causal discovery. *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, 2011.

K. Zhang, Z. Wang, J. Zhang, et B. Schölkopf. On estimation of functional causal models: General results and application to the post-nonlinear causal model. *ACM Trans. Intell. Syst. Technol.*, 2015.

Q. Zhang, S. Filippi, A. Gretton, et D. Sejdinovic. Large-scale kernel methods for independence testing. *Statistics and Computing*, 2018.

X. Zheng, B. Aragam, P. Ravikumar, et E. Xing. Dags with no tears: Continuous optimization for structure learning. In *Advances in Neural Information Processing Systems 31*, 2018a.

X. Zheng, B. Aragam, P. Ravikumar, et E. Xing. Dags with no tears: Continuous optimization for structure learning. In *Advances in Neural Information Processing Systems 31*, 2018b.

X. Zheng, C. Dan, B. Aragam, P. Ravikumar, et E. Xing. Learning sparse nonparametric dags. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, 2020.

S. Zhu et Z. Chen. Causal discovery with reinforcement learning. *Proceedings of the 8th International Conference on Learning Representations*, 2020.

A. M. Zimmer, Y. K. Pan, T. Chandrapalan, R. W. Kwong, et S. F. Perry. Loss-of-function approaches in comparative physiology: is there a future for knockdown experiments in the era of genome editing? *Journal of Experimental Biology*, 2019.

Appendix A

Article 1: Matériel supplémentaire

A.1. Appendix

A.1.1. Optimization

Let us recall the augmented Lagrangian:

$$\max_{\phi} \mathcal{L}(\phi, \lambda_t, \mu_t) \triangleq \mathbb{E}_{X \sim P_X} \sum_{i=1}^{d} \log p_i(X_i | X_{\pi_i^\phi}; \phi_{(i)}) - \lambda_t h(\phi) - \frac{\mu_t}{2} h(\phi)^2 \tag{A.1.1}$$

where λ_t and μ_t are the Lagrangian and penalty coefficients of the tth subproblem, respectively. In all our experiments, we initialize those coefficients using $\lambda_0 = 0$ and $\mu_0 = 10^{-3}$. We approximately solve each non-convex subproblem using RMSprop ((Tieleman et Hinton, 2012a)), a stochastic gradient descent variant popular for NNs. We use the following gradient estimate:

$$\begin{aligned} &\nabla_\phi \mathcal{L}(\phi, \lambda_t, \mu_t) \approx \nabla_\phi \hat{\mathcal{L}}_B(\phi, \lambda_t, \mu_t) \\ &\text{with} \quad \hat{\mathcal{L}}_B(\phi, \lambda_t, \mu_t) \triangleq \frac{1}{|B|} \sum_{x \in B} \sum_{i=1}^{d} \log p_i(x_i | x_{\pi_i^\phi}; \phi_{(i)}) - \lambda_t h(\phi) - \frac{\mu_t}{2} h(\phi)^2 \end{aligned} \tag{A.1.2}$$

where B is a minibatch sampled from the data set and $|B|$ is the minibatch size. The gradient estimate $\nabla_\phi \hat{\mathcal{L}}_B(\phi, \lambda_t, \mu_t)$ can be computed using standard deep learning libraries. We consider a subproblem has converged when $\hat{\mathcal{L}}_H(\phi, \lambda_t, \mu_t)$ evaluated on a held-out data set H stops increasing. Let ϕ_t^* be the approximate solution to subproblem t. Then, λ_t and μ_t are updated according to the following rule:

$$\begin{aligned} \lambda_{t+1} &\leftarrow \lambda_t + \mu_t h\left(\phi_t^*\right) \\ \mu_{t+1} &\leftarrow \begin{cases} \eta \mu_t, & \text{if } h\left(\phi_t^*\right) > \gamma h\left(\phi_{t-1}^*\right) \\ \mu_t, & \text{otherwise} \end{cases} \end{aligned} \tag{A.1.3}$$

with $\eta = 10$ and $\gamma = 0.9$. Each subproblem t is initialized using the previous subproblem solution ϕ_{t-1}^*. The augmented Lagrangian method stops when $h(\phi) \leq 10^{-8}$.

Total number of iterations before augmented Lagrangian converges: In GraN-DAG and NOTEARS, every subproblem is approximately solved using an iterative algorithm. Let T be the number of subproblems solved before the convergence of the augmented Lagrangian. For a given subproblem t, let K_t be the number of iterations executed to approximately solve it. Let $I = \sum_{t=1}^{T} K_t$ be the *total number of iterations* before the augmented Lagrangian converges. Table A.1 reports the total number of iterations I for GraN-DAG and NOTEARS, averaged over ten data sets. Note that the matrix exponential is evaluated once per iteration. Even though GraN-DAG uses a stochastic gradient algorithm, it requires less iterations than NOTEARS which uses a batch proximal quasi-Newton method. We hypothesize early stopping avoids having to wait until full convergence before moving to the next subproblem, hence reducing the total number of iterations. Note that GraN-DAG total

run time is still larger than NOTEARS due to its gradient requiring more computation to evaluate (total runtime $\approx$ 10 minutes against $\approx$ 1 minute for 20 nodes graphs and $\approx$ 4 hours against $\approx$ 1 hour for 100 nodes graphs). GraN-DAG runtime on 100 nodes graphs can be roughly halved when executed on GPU.

Table A.1. Total number of iterations ($\times 10^3$) before augmented Lagrangian converges on Gauss-ANM data.

	20 nodes ER1	20 nodes ER4	100 nodes ER1	100 nodes ER4
GraN-DAG	27.3 ± 3.6	30.4 ± 4.2	23.1 ± 0.7	23.1 ± 0.8
NOTEARS	67.1 ± 35.3	72.3 ± 24.3	243.6 ± 12.3	232.4 ± 12.9

A.1.2. Thresholding to ensure acyclicity

The augmented Lagrangian outputs ϕ_T^* where T is the number of subproblems solved before declaring convergence. Note that the weighted adjacency matrix $A_{\phi_T^*}$ will most likely not represent an acyclic graph, even if we threshold as we learn, as explained in Section 3.3.4. We need to remove additional edges to obtain a DAG (edges are removed using the mask presented in Section 3.3.4). One option would be to remove edges one by one until a DAG is obtained, starting from the edge (i,j) with the lowest $(A_{\phi_T^*})_{ij}$ up to the edge with the highest $(A_{\phi_T^*})_{ij}$. This amounts to gradually increasing the threshold ϵ until $A_{\phi_T^*}$ is acyclic. However, this approach has the following flaw: It is possible to have $(A_{\phi_T^*})_{ij}$ significantly higher than zero while having $\theta_{(j)}$ almost completely independent of variable X_i. This can happen for at least two reasons. First, the NN paths from input i to output k might end up cancelling each others, rendering the input i inactive. Second, some neurons of the NNs might always be saturated for the observed range of inputs, rendering some NN paths *effectively inactive* without being inactive in the sense described in Section 3.3.1. Those two observations illustrate the fact that having $(A_{\phi_T^*})_{ij} = 0$ is only a sufficient condition to have $\theta_{(j)}$ independent of variable X_i and not a necessary one.

To avoid this issue, we consider the following alternative. Consider the function $\mathcal{L} : \mathbb{R}^d \mapsto \mathbb{R}^d$ which maps all d variables to their respective conditional likelihoods, i.e. $\mathcal{L}_i(X) \triangleq p_i(X_i \mid X_{\pi_i^{\phi_T^*}})\ \ \forall i$. We consider the following expected Jacobian matrix

$$\mathcal{J} \triangleq \mathbb{E}_{X \sim P_X} \left| \frac{\partial \mathcal{L}}{\partial X} \right|^\top \tag{A.1.4}$$

where $\left|\frac{\partial \mathcal{L}}{\partial X}\right|$ is the Jacobian matrix of $\mathcal{L}$ evaluated at X, in absolute value (element-wise). Similarly to $(A_{\phi_T^*})_{ij}$, the entry $\mathcal{J}_{ij}$ can be loosely interpreted as the strength of edge (i,j). We propose removing edges starting from the lowest $\mathcal{J}_{ij}$ to the highest, stopping as soon as acyclicity is achieved. We believe $\mathcal{J}$ is better than $A_{\phi_T^*}$ at capturing which NN inputs are

effectively inactive since it takes into account NN paths cancelling each others and saturated neurons. Empirically, we found that using $\mathcal{J}$ instead of $A_{\phi_T^*}$ yields better results, and thus we report the results with $\mathcal{J}$ in this paper.

A.1.3. Preliminary neighborhood selection and DAG Pruning

PNS: For graphs of 50 nodes or more, GraN-DAG performs a *preliminary neighborhood selection* (PNS) similar to what has been proposed in Bühlmann et al. ((2014)). This procedure applies a variable selection method to get a set of possible parents for each node. This is done by fitting an *extremely randomized trees* ((Geurts et al., 2006)) (using `ExtraTreesRegressor` from `scikit-learn`) for each variable against all the other variables. For each node a feature importance score based on the gain of purity is calculated. Only nodes that have a feature importance score higher than $0.75 \cdot$ `mean` are kept as potential parent, where `mean` is the mean of the feature importance scores of all nodes. Although the use of PNS in CAM was motivated by gains in computation time, GraN-DAG uses it to avoid overfitting, without reducing the computation time.

Pruning: Once the thresholding is performed and a DAG is obtained as described in A.1.2, GraN-DAG performs a pruning step identical to CAM ((Bühlmann et al., 2014)) in order to remove spurious edges. We use the implementation of Bühlmann et al. ((2014)) based on the R function `gamboost` from the `mboost` package. For each variable X_i, a generalized additive model is fitted against the current parents of X_i and a significance test of covariates is applied. Parents with a p-value higher than 0.001 are removed from the parent set. Similarly to what Bühlmann et al. ((2014)) observed, this pruning phase generally has the effect of greatly reducing the SHD without considerably changing the SID.

Ablation study: In Table A.2, we present an ablation study which shows the effect of adding PNS and pruning to GraN-DAG on different performance metrics and on the negative log-likelihood (NLL) of the training and validation set. Note that, before computing both NLL, we reset all parameters of GraN-DAG except the mask and retrained the model on the training set without any acyclicity constraint (acyclicity is already ensure by the masks at this point). This retraining procedure is important since the pruning removes edges (i.e. some additional NN inputs are masked) and it affects the likelihood of the model (hence the need to retrain).

Table A.2. PNS and pruning ablation study for GraN-DAG (averaged over 10 datasets from ER1 with 50 nodes)

PNS	Pruning	SHD	SID	NLL (train)	NLL (validation)
False	False	1086.8±48.8	31.6±23.6	0.36±0.07	1.44±0.21
True	False	540.4±70.3	17.4±16.7	0.52±0.08	1.16±0.17
False	True	11.8±5.0	39.7±25.5	0.78±0.12	0.84±0.12
True	True	6.1±3.3	29.3±19.5	0.78±0.13	0.83±0.12

A first observation is that adding PNS and pruning improve the NLL on the validation set while deteriorating the NLL on the training set, showing that those two steps are indeed reducing overfitting. Secondly, the effect on SHD is really important while the effect on SID is almost nonexistent. This can be explained by the fact that SID has more to do with the ordering of the nodes than with false positive edges. For instance, if we have a complete DAG with a node ordering coherent with the ground truth graph, the SID is zero, but the SHD is not due to all the false positive edges. Without the regularizing effect of PNS and pruning, GraN-DAG manages to find a DAG with a good ordering but with many spurious edges (explaining the poor SHD, the good SID and the big gap between the NLL of the training set and validation set). PNS and pruning helps reducing the number of spurious edges, hence improving SHD.

We also implemented PNS and pruning for NOTEARS and DAG-GNN to see whether their performance could also be improved. Table A.3 reports an ablation study for DAG-GNN and NOTEARS. First, the SHD improvement is not as important as for GraN-DAG and is almost not statistically significant. The improved SHD does not come close to performance of GraN-DAG. Second, PNS and pruning do not have a significant effect of SID, as was the case for GraN-DAG. The lack of improvement for those methods is probably due to the fact that they are not overfitting like GraN-DAG, as the training and validation (unregularized) scores shows. NOTEARS captures only linear relationships, thus it will have a hard time overfitting nonlinear data and DAG-GNN uses a strong form of parameter sharing between its conditional densities which possibly cause underfitting in a setup where all the parameters of the conditionals are sampled independently.

Moreover, DAG-GNN and NOTEARS threshold aggressively their respective weighted adjacency matrix at the end of training (with the default parameters used in the code), which also acts as a form of heavy regularization, and allow them to remove many spurious edges. GraN-DAG without PNS and pruning does not threshold as strongly by default which explains the high SHD of Table A.2. To test this explanation, we removed all edges (i, j) for which $(A_\phi)_{ij} < 0.3$ for GraN-DAG and obtained an SHD of 29.4±15.9 and an SID of

This was the default value of thresholding used in NOTEARS and DAG-GNN.

85.6±45.7, showing a significant improvement over NOTEARS and DAG-GNN, even without PNS and pruning.

Table A.3. PNS and pruning ablation study for DAG-GNN and NOTEARS (averaged over 10 datasets from ER1 with 50 nodes)

Algorithm	PNS	Pruning	SHD	SID	Score (train)	Score (validation)
DAG-GNN	False	False	56.8±11.1	322.9±103.8	-2.8±1.5	-2.2±1.6
	True	False	55.5±10.2	314.5±107.6	-2.1±1.6	-2.1±1.7
	False	True	49.4±7.8	325.1±103.7	-1.8±1.1	-1.8±1.2
	True	True	47.7±7.3	316.5±105.6	-1.9±1.6	-1.9±1.6
NOTEARS	False	False	64.2±9.5	327.1±110.9	-23.1±1.8	-23.2±2.1
	True	False	54.1±10.9	321.5±104.5	-25.2±2.7	-25.4±2.8
	False	True	49.5±8.8	327.7±111.3	-26.7±2.0	-26.8±2.1
	True	True	49.0±7.6	326.4±106.9	-26.23±2.2	-26.4±2.4

A.1.4. Large Sample Size Experiment

In this section, we test the bias-variance hypothesis which attempts to explain why CAM is on par with GraN-DAG on Gauss-ANM data even if its model wrongly assumes that the f_j functions are additive. Table A.4 reports the performance of GraN-DAG and CAM for different sample sizes. We can see that, as the sample size grows, GraN-DAG ends up outperforming CAM in terms of SID while staying on par in terms of SHD. We explain this observation by the fact that a larger sample size reduces variance for GraN-DAG thus allowing it to leverage its greater capacity against CAM which is stuck with its modelling bias. Both algorithms were run with their respective default hyperparameter combination.

This experiment suggests GraN-DAG could be an appealing option in settings where the sample size is substantial. The present paper focuses on sample sizes typically encountered in the structure/causal learning litterature and leave this question for future work.

Table A.4. Effect of sample size - Gauss-ANM 50 nodes ER4 (averaged over 10 datasets)

Sample size	Method	SHD	SID
500	CAM	123.5 ± 13.9	1181.2 ± 160.8
	GraN-DAG	130.2 ± 14.4	1246.4 ± 126.1
1000	CAM	103.7 ± 15.2	1074.7 ± 125.8
	GraN-DAG	104.4 ± 15.3	942.1 ± 69.8
5000	CAM	74.1 ± 13.2	845.0 ± 159.8
	GraN-DAG	71.9 ± 15.9	554.1 ± 117.9
10000	CAM	66.3 ± 16.0	808.1 ± 142.9
	GraN-DAG	65.9 ± 19.8	453.4 ± 171.7

A.1.5. Details on data sets generation

Synthetic data sets: For each data set type, 10 data sets are sampled with 1000 examples each. As the synthetic data introduced in Section 3.4.1, for each data set, a ground truth DAG $\mathcal{G}$ is randomly sampled following the ER scheme and then the data is generated. Unless otherwise stated, all root variables are sampled from $\mathcal{U}[-1,1]$.

- *Gauss-ANM* is generated following $X_j := f_j(X_{\pi_j^{\mathcal{G}}}) + N_j \ \forall j$ with mutually independent noises $N_j \sim \mathcal{N}(0, \sigma_j^2) \ \ \forall j$ where the functions f_j are independently sampled from a Gaussian process with a unit bandwidth RBF kernel and $\sigma_j^2 \sim \mathcal{U}[0.4,0.8]$. Source nodes are Gaussian with zero mean and variance sampled from $\mathcal{U}[1,2]$
- *LIN* is generated following $X_j|X_{\pi_j^{\mathcal{G}}} \sim w_j^T X_{\pi_j^{\mathcal{G}}} + 0.2 \cdot \mathcal{N}(0, \sigma_j^2) \ \ \forall j$ where $\sigma_j^2 \sim \mathcal{U}[1,2]$ and w_j is a vector of $|\pi_j^{\mathcal{G}}|$ coefficients each sampled from $\mathcal{U}[0,1]$.
- *ADD-FUNC* is generated following $X_j|X_{\pi_j^{\mathcal{G}}} \sim \sum_{i \in \pi_j^{\mathcal{G}}} f_{j,i}(X_i) + 0.2 \cdot \mathcal{N}(0, \sigma_j^2) \ \ \forall j$ where $\sigma_j^2 \sim \mathcal{U}[1,2]$ and the functions $f_{j,i}$ are independently sampled from a Gaussian process with bandwidth one. This model is adapted from Bühlmann et al. ((2014)).
- *PNL-GP* is generated following $X_j|X_{\pi_j^{\mathcal{G}}} \sim \sigma(f_j(X_{\pi_j^{\mathcal{G}}}) + \mathit{Laplace}(0, l_j)) \ \ \forall j$ with the functions f_j independently sampled from a Gaussian process with bandwidth one and $l_j \sim \mathcal{U}[0,1]$. In the two-variable case, this model is identifiable following the Corollary 9 from Zhang et Hyvärinen ((2009)). To get identifiability according to this corollary, it is important to use non-Gaussian noise, explaining our design choices.
- *PNL-MULT* is generated following $X_j|X_{\pi_j^{\mathcal{G}}} \sim \exp(\log(\sum_{i \in \pi_j^{\mathcal{G}}} X_i) + |\mathcal{N}(0, \sigma_j^2)|) \ \ \forall j$ where $\sigma_j^2 \sim \mathcal{U}[0,1]$. Root variables are sampled from $\mathcal{U}[0,2]$. This model is adapted from Zhang et al. ((2015)).

SynTReN: Ten datasets have been generated using the SynTReN generator (`http://bioinformatics.intec.ugent.be/kmarchal/SynTReN/index.html`) using the software default parameters except for the *probability for complex 2-regulator interactions* that was set to 1 and the random seeds used were 0 to 9. Each dataset contains 500 samples and comes from a 20 nodes graph.

Graph types: *Erdős-Rényi* (ER) graphs are generated by randomly sampling a topological order and by adding directed edges were it is allowed independently with probability $p = \frac{2e}{d^2-d}$ were e is the expected number of edges in the resulting DAG. *Scale-free* (SF) graphs were generated using the Barabási-Albert model ((Barabási et Albert, 1999)) which is based on preferential attachment. Nodes are added one by one. Between the new node and the existing nodes, m edges (where m is equal to d or $4d$) will be added. An existing node i have the probability $p(k_i) = \frac{k_i}{\sum_j k_j}$ to be chosen, where k_i represents the degree of the node i. While ER graphs have a degree distribution following a Poisson distribution, SF graphs have a degree distribution following a power law: few nodes, often called *hubs*, have a high

degree. Barabási ((2009)) have stated that these types of graphs have similar properties to real-world networks which can be found in many different fields, although these claims remain controversial ((Clauset et al., 2009)).

A.1.6. Supplementary experiments

Gauss-ANM: The results for 20 and 100 nodes are presented in Table A.5 and A.6 using the same Gauss-ANM data set types introduced in Section 3.4.1. The conclusions drawn remains similar to the 10 and 50 nodes experiments. For GES and PC, the SHD and SID are respectively presented in Table A.7 and A.8. Their performances do not compare favorably to the GraN-DAG nor CAM. Figure A.1 shows the entries of the weighted adjacency matrix A_ϕ as training proceeds in a typical run for 10 nodes.

LIN & ADD-FUNC: Experiments with LIN and ADD-FUNC data is reported in Table A.9 & A.10. The details of their generation are given in Appendix A.1.5.

PNL-GP & PNL-MULT: Table A.11 contains the performance of GraN-DAG and other baselines on post nonlinear data discussed in Section 3.4.1.

Table A.5. Results for ER and SF graphs of 20 nodes with Gauss-ANM data

	ER1		ER4		SF1		SF4	
	SHD	SID	SHD	SID	SHD	SID	SHD	SID
GraN-DAG	**4.0 ±3.4**	**17.9±19.5**	**45.2±10.7**	**165.1±21.0**	**7.6±2.5**	28.8±10.4	**36.8±5.1**	**62.5±18.8**
DAG-GNN	25.6±7.5	109.1±53.1	75.0±7.7	344.8±17.0	19.5±1.8	60.1±12.8	49.5±5.4	115.2±33.3
NOTEARS	30.3±7.8	107.3±47.6	79.0±8.0	346.6±13.2	23.9±3.5	69.4±19.7	52.0±4.5	120.5±32.5
CAM	**2.7±1.8**	**10.6±8.6**	**41.0±11.9**	**157.9±41.2**	**5.7±2.6**	**23.3±18.0**	**35.6±4.5**	**59.1±18.8**
GSF	12.3±4.6	[15.0±19.9 45.6±22.9]	**41.8±13.8**	[**153.7±49.4 201.6±37.9**]	**7.4±3.5**	[**5.7±7.1 27.3±13.2**]	**38.6±3.6**	[**54.9±14.4 86.7±24.2**]
RANDOM	103.0±39.6	94.3±53.0	117.5±25.9	298.5±28.7	105.2±48.8	81.1±54.4	121.5±28.5	204.8±38.5

Table A.6. Results for ER and SF graphs of 100 nodes with Gauss-ANM data

	ER1		ER4		SF1		SF4	
	SHD	SID	SHD	SID	SHD	SID	SHD	SID
GraN-DAG	**15.1±6.0**	**83.9±46.0**	**206.6±31.5**	**4207.3±419.7**	**59.2±7.7**	**265.4±64.2**	**262.7±19.6**	**872.0±130.4**
DAG-GNN	110.2±10.5	883.0±320.9	379.5±24.7	8036.1±656.2	97.6±1.5	438.6±112.7	316.0±14.3	1394.6±165.9
NOTEARS	125.6±12.1	913.1±343.8	387.8±25.3	8124.7±577.4	111.7±5.4	484.3±138.4	327.2±15.8	1442.8±210.1
CAM	**17.3±4.5**	**124.9±65.0**	**186.4±28.8**	**4601.9±482.7**	**52.7±9.3**	**230.3±36.9**	**255.6±21.7**	**845.8±161.3**
GSF	66.8±7.3	[104.7±59.5 238.6±59.3]	> 12 hours	— —	**71.4±11.2**	[**212.7±71.1 325.3±105.2**]	**275.9±21.0**	[**793.9±152.5 993.4±149.2**]
RANDOM	1561.6±1133.4	1175.3±547.9	2380.9±1458.0	7729.7±1056.0	2222.2±1141.2	1164.2±593.3	2485.0±1403.9	4206.4±1642.1

Note that GSF results are missing for two data set types in Tables A.6 and A.11. This is because the search algorithm could not finish within 12 hours, even when the maximal in-degree was limited to 5. All other methods could run in less than 6 hours.

Table A.7. SHD for GES and PC (against GraN-DAG for reference) with Gauss-ANM data

	10 nodes		20 nodes		50 nodes		100 nodes	
	ER1	ER4	ER1	ER4	ER1	ER4	ER1	ER4
GraN-DAG	1.7±2.5	8.3±2.8	4.0 ±3.4	45.2±10.7	5.1±2.8	102.6±21.2	15.1±6.0	206.6±31.5
GES	13.8±4.8	32.3±4.3	43.3±12.4	94.6±9.8	106.6±24.7	254.4±39.3	292.9±33.6	542.6±51.2
PC	8.4±3	34±2.6	20.136.4±6.5	73.1±5.8	44.0±11.6	183.6±20	95.2±9.1	358.8±20.6
	SF1	SF4	SF1	SF4	SF1	SF4	SF1	SF4
GraN-DAG	1.2±1.1	9.9±4.0	7.6±2.5	36.8±5.1	25.5±6.2	111.3±12.3	59.2±7.7	262.7±19.6
GES	8.1±2.4	17.4±4.5	26.2±7.5	50.7±6.2	73.9±7.4	178.8±16.5	190.3±22	408.7±24.9
PC	4.8±2.4	16.4±2.8	13.6±2.1	44.4±4.6	43.1±5.7	135.4±10.7	97.6±6.6	314.2±17.5

Table A.8. Lower and upper bound on the SID for GES and PC (against GraN-DAG for reference) with Gauss-ANM data. See Appendix A.1.7 for details on how to compute SID for CPDAGs.

	10 nodes		20 nodes		50 nodes		100 nodes	
	ER1	ER4	ER1	ER4	ER1	ER4	ER1	ER4
GraN-DAG	1.7±3.1	21.8±8.9	17.9±19.5	165.1±21.0	22.4±17.8	1060.1±109.4	83.9±46.0	4207.3±419.7
GES	[24.1±17.3 27.2±17.5]	[68.5±10.5 75±7]	[62.1±44 65.7±44.5]	[301.9±19.4 319.3±13.3]	[150.9±72.7 155.1±74]	[1996.6±73.1 2032.9±88.7]	[582.5±391.1 598.9±408.6]	[8054±524.8 8124.2±470.2]
PC	[22.6±15.5 27.3±13.1]	[78.1±7.4 79.2±5.7]	[80.9±51.1 94.9±46.1]	[316.7±25.7 328.7±25.6]	[222.7±138 256.7±127.3]	[2167.9±88.4 2178.8±80.8]	[620.7±240.9 702.5±255.8]	[8236.9±478.5 8265.4±470.2]
	SF1	SF4	SF1	SF4	SF1	SF4	SF1	SF4
GraN-DAG	4.1±6.1	16.4±6.0	28.8±10.4	62.5±18.8	90.0±18.9	271.2±65.4	265.4±64.2	872.0±130.4
GES	[11.6±9.2 16.4±11.7]	[39.3±11.2 43.9±14.9]	[54.9±23.1 57.9±24.6]	[89.5±38.4 105.1±44.3]	[171.6±70.1 182.7±77]	[496.3±154.1 529.7±184.5]	[511.5±257.6 524±252.2]	[1421.7±247.4 1485.4±233.6]
PC	[8.3±4.6 16.8±12.3]	[36.5±6.2 41.7±6.9]	[42.2±14 59.7±14.9]	[95.6±37 118.5±30]	[124.2±38.3 167.1±41.4]	[453.2±115.9 538±143.7]	[414.5±124.4 486.5±120.9]	[1369.2±259.9 1513.7±296.2]

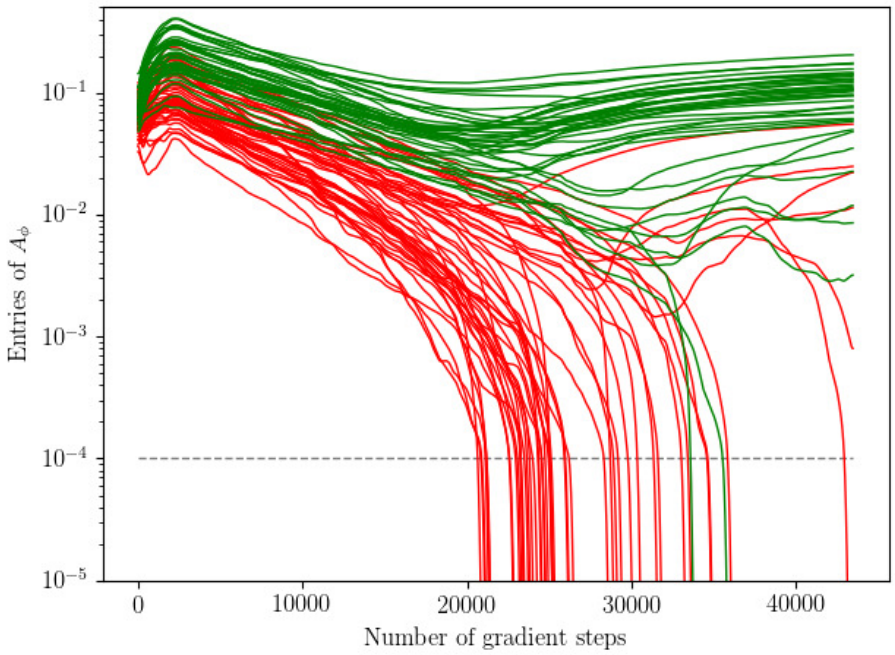


Fig. A.1. Entries of the weighted adjacency matrix A_ϕ as training proceeds in GraN-DAG for a synthetic data set ER4 with 10 nodes. Green curves represent edges which appear in the ground truth graph while red ones represent edges which do not. The horizontal dashed line at 10^{-4} is the threshold ϵ introduced in Section 3.3.4. We can see that GraN-DAG successfully recovers most edges correctly while keeping few spurious edges.

Table A.9. LIN

#Nodes	10				50			
Graph Type	ER1		ER4		ER1		ER4	
Metrics	SHD	SID	SHD	SID	SHD	SID	SHD	SID
Method								
GraN-DAG	$\mathbf{7.2 \pm 2.0}$	27.3 ± 8.1	30.7 ± 3.3	75.8 ± 6.9	$\mathbf{33.9 \pm 8.6}$	$\mathbf{255.8 \pm 158.4}$	181.9 ± 24.0	2035.8 ± 137.2
DAG-GNN	10.3 ± 3.5	39.6 ± 14.7	$\mathbf{18.9 \pm 4.8}$	$\mathbf{63.7 \pm 8.9}$	54.1 ± 9.2	330.4 ± 117.1	$\mathbf{130.3 \pm 17.3}$	$\mathbf{1937.5 \pm 89.8}$
NOTEARS	$\mathbf{9.0 \pm 3.0}$	35.3 ± 13.4	27.9 ± 4.3	$\mathbf{72.1 \pm 7.9}$	45.5 ± 7.8	310.7 ± 125.9	$\mathbf{126.1 \pm 13.0}$	$\mathbf{1971.1 \pm 134.3}$
CAM	10.2 ± 6.3	31.2 ± 10.9	33.6 ± 3.3	77.5 ± 2.3	$\mathbf{36.2 \pm 5.8}$	$\mathbf{234.8 \pm 105.1}$	182.5 ± 17.6	$\mathbf{1948.7 \pm 113.5}$
GSF	9.2 ± 2.9	$[19.5 \pm 14.6$ $31.6 \pm 17.3]$	38.6 ± 3.7	$[73.8 \pm 7.6$ $85.2 \pm 8.3]$	46.7 ± 4.1	$[\mathbf{176.4 \pm 98.8}$ $\mathbf{215.0 \pm 108.9}]$	> 12 hours	
RANDOM	22.0 ± 2.9	30.0 ± 13.8	34.4 ± 2.4	78.8 ± 5.5	692.6 ± 7.5	360.3 ± 141.4	715.9 ± 16.0	$\mathbf{1932.7 \pm 40.2}$

Table A.10. ADD-FUNC

#Nodes	10				50			
Graph Type	ER1		ER4		ER1		ER4	
Metrics	SHD	SID	SHD	SID	SHD	SID	SHD	SID
Method								
GraN-DAG	$\mathbf{2.8 \pm 2.5}$	$\mathbf{7.5 \pm 7.7}$	14.5 ± 5.2	52.6 ± 10.8	16.6 ± 5.3	103.6 ± 52.9	86.4 ± 21.6	1320.6 ± 145.8
DAG-GNN	10.1 ± 3.4	23.3 ± 11.5	18.3 ± 3.6	56.4 ± 6.1	45.5 ± 7.9	261.1 ± 88.8	224.3 ± 31.6	1741.0 ± 138.3
NOTEARS	11.1 ± 5.0	16.9 ± 11.3	20.3 ± 4.9	53.5 ± 10.5	53.7 ± 9.5	276.1 ± 96.8	201.8 ± 22.1	1813.6 ± 148.4
CAM	$\mathbf{2.5 \pm 2.0}$	$\mathbf{7.9 \pm 6.4}$	$\mathbf{6.0 \pm 5.6}$	$\mathbf{29.3 \pm 19.3}$	$\mathbf{9.6 \pm 5.1}$	$\mathbf{39.0 \pm 34.1}$	$\mathbf{42.9 \pm 6.6}$	$\mathbf{857.0 \pm 184.5}$
GSF	9.3 ± 3.9	$[13.9 \pm 8.3$ $24.1 \pm 12.5]$	29.5 ± 4.3	$[60.3 \pm 11.6$ $75.0 \pm 4.5]$	49.5 ± 5.1	$[151.5 \pm 73.8$ $213.9 \pm 82.5]$	> 12 hours	
RANDOM	23.0 ± 2.2	26.9 ± 18.1	33.5 ± 2.3	76.0 ± 6.2	689.7 ± 6.1	340.0 ± 113.6	711.5 ± 9.0	1916.2 ± 65.8

Table A.11. Synthetic post nonlinear data sets

		PNL-GP		PNL-MULT	
		SHD	SID	SHD	SID
10 nodes ER1	GraN-DAG	**1.6±3.0**	**3.9±8.0**	**13.1±3.8**	35.7±12.3
	DAG-GNN	11.5±6.8	32.4±19.3	17.900±6.2	40.700±14.743
	NOTEARS	10.7±5.5	34.4±19.1	**14.0±4.0**	38.6±11.9
	CAM	**1.5±2.6**	**6.8±12.1**	**12.0±6.4**	36.3±17.7
	GSF	**6.2±3.3**	[7.7±8.7, 18.9±12.4]	**10.7±3.0**	**[9.8±11.9, 25.3±11.5]**
	RANDOM	23.8±2.9	36.8±19.1	23.7±2.9	37.7±20.7
10 nodes ER4	GraN-DAG	**10.9±6.8**	**39.8±21.1**	32.1±4.5	77.7±5.9
	DAG-GNN	32.3±4.3	75.8±9.3	37.0±3.1	82.7±6.4
	NOTEARS	34.1±3.2	80.8±5.5	37.7±3.0	81.700±7.258
	CAM	**8.4±4.8**	**30.5±20.0**	34.4±3.9	79.6±3.8
	GSF	25.0±6.0	[44.3±14.5, 66.1±10.1]	31.3±5.4	**[58.6±8.1, 76.4±9.9]**
	RANDOM	35.0±3.3	80.0±5.1	33.6±3.5	76.2±7.3
50 nodes ER1	GraN-DAG	16.5±7.0	64.1±35.4	**38.2±11.4**	213.8±114.4
	DAG-GNN	56.5±11.1	334.3±80.3	83.9±23.8	507.7±253.4
	NOTEARS	50.1±9.9	319.1±76.9	78.5±21.5	425.7±197.0
	CAM	**5.1±2.6**	**10.7±12.4**	**44.9±9.9**	284.3±124.9
	GSF	31.2±6.0	[59.5±34.1, 122.4±32.0]	**46.3±12.1**	**[65.8±62.2, 141.6±72.6]**
	RANDOM	688.4±4.9	307.0±98.5	691.3±7.3	488.0±247.8
50 nodes ER4	GraN-DAG	**68.7±17.0**	**1127.0±188.5**	**211.7±12.6**	2047.7±77.7
	DAG-GNN	203.8±18.9	2173.1±87.7	246.7±16.1	2239.1±42.3
	NOTEARS	189.5±16.0	2134.2±125.6	**220.0±9.9**	2175.2±58.3
	CAM	**48.2±10.3**	**899.5±195.6**	**208.1±14.8**	2029.7±55.4
	GSF	105.2±15.5	[1573.7±121.2, 1620±102.8]	> 12 hours	—
	RANDOM	722.3±9.0	1897.4±83.7	710.2±9.5	1935.8±56.9

A.1.7. Metrics

SHD takes two partially directed acyclic graphs (PDAG) and counts the number of edge for which the edge type differs in both PDAGs. There are four edge types: $i \leftarrow j$, $i \rightarrow j$, $i - j$ and $i \quad j$. Since this distance is defined over the space of PDAGs, we can use it to compare DAGs with DAGs, DAGs with CPDAGs and CPDAGs with CPDAGs. When comparing a DAG with a CPDAG, having $i \leftarrow j$ instead of $i - j$ counts as a mistake.

SHD-C is very similar to SHD. The only difference is that both DAGs are first mapped to their respective CPDAGs before measuring the SHD.

Introduced by Peters et Bühlmann ((2015a)), SID counts the number of interventional distribution of the form $p(x_i|\ do(x_j = \hat{x}_j))$ that would be miscalculated using the *parent adjustment formula* ((Pearl, 2009a)) if we were to use the predicted DAG instead of the ground truth DAG to form the parent adjustment set. Some care needs to be taken to evaluate the SID for methods outputting a CPDAG such as GES and PC. Peters et Bühlmann ((2015a)) proposes to report the SID of the DAGs which have approximately the minimal and the maximal SID in the Markov equivalence class given by the CPDAG. See Peters et Bühlmann ((2015a)) for more details.

A.1.8. Hyperparameters

All GraN-DAG runs up to this point were performed using the following set of hyperparameters. We used RMSprop as optimizer with learning rate of 10^{-2} for the first subproblem and 10^{-4} for all subsequent suproblems. Each NN has two hidden layers with 10 units (except for the real and pseudo-real data experiments of Table 3.3 which uses only 1 hidden layer). Leaky-ReLU is used as activation functions. The NN are initialized using the initialization scheme proposed in Glorot et Bengio ((2010b)) also known as *Xavier initialization*. We used minibatches of 64 samples. This hyperparameter combination have been selected via a small scale experiment in which many hyperparameter combinations have been tried manually on a single data set of type ER1 with 10 nodes until one yielding a satisfactory SHD was obtained. Of course in practice one cannot select hyperparameters in this way since we do not have access to the ground truth DAG. In Appendix A.1.9, we explain how one could use a held-out data set to select the hyperparameters of score-based approaches and report the results of such a procedure on almost settings presented in this paper.

For NOTEARS, DAG-GNN, and GSF, we used the default hyperparameters found in the authors code. It (rarely) happens that NOTEARS and DAG-GNN returns a cyclic graph. In those cases, we removed edges starting from the weaker ones to the strongest (according to their respective weighted adjacency matrices), stopping as soon as acyclicity is achieved (similarly to what was explained in Appendix A.1.2 for GraN-DAG). For GES and PC, we used default hyperparameters of the `pcalg` R package. For CAM, we used the the default hyperparameters found in the `CAM` R package, with default PNS and DAG pruning.

A.1.9. Hyperparameter Selection via Held-out Score

Most structure/causal learning algorithms have hyperparameters which must be selected prior to learning. For instance, NOTEARS and GES have a regularizing term in their score controlling the sparsity level of the resulting graph while CAM has a thresholding level for its pruning phase (also controlling the sparsity of the DAG). GraN-DAG and DAG-GNN have many hyperparameters such as the learning rate and the architecture choice for the neural

networks (i.e. number of hidden layers and hidden units per layer). One approach to selecting hyperparameters in practice consists in trying multiple hyperparameter combinations and keeping the one yielding the best score evaluated on a held-out set ((Koller et Friedman, 2009, p. 960)). By doing so, one can hopefully avoid finding a DAG which is too dense or too sparse since if the estimated graph contains many spurious edges, the score on the held-out data set should be penalized. In the section, we experiment with this approach on almost all settings and all methods covered in the present paper.

Experiments: We explored multiple hyperparameter combinations using random search ((Bergstra et Bengio, 2012)). Table A.12 to Table A.20 report results for each dataset types. Each table reports the SHD and SID averaged over 10 data sets and for each data set, we tried 50 hyperparameter combinations sampled randomly (see Table A.21 for sampling schemes). The hyperparameter combination yielding the best held-out score among all 50 runs is selected *per data set* (i.e. the average of SHD and SID scores correspond to potentially different hyperparameter combinations on different data sets). 80% of the data was used for training and 20% was held out (GraN-DAG uses the same data for early stopping and hyperparameter selection). Note that the held-out score is always evaluated without the regularizing term (e.g. the held-out score of NOTEARS was evaluated without its L1 regularizer).

The symbols $^{++}$ and $^{+}$ indicate the hyperparameter search improved performance against default hyperparameter runs above one standard deviation and within one standard deviation, respectively. Analogously for $^{--}$ and $^{-}$ which indicate a performance reduction. The flag $_{***}$ indicate that, on average, less than 10 hyperparameter combinations among the 50 tried allowed the method to converge in less than 12 hours. Analogously, $_{**}$ indicates between 10 and 25 runs converged and $_{*}$ indicates between 25 and 45 runs converged.

Discussion: GraN-DAG and DAG-GNN are the methods benefiting the most from the hyperparameter selection procedure (although rarely significantly). This might be explained by the fact that neural networks are in general very sensitive to the choice of hyperparameters. However, not all methods improved their performance and no method improves its performance in all settings. GES and GSF for instance, often have significantly worse results. This might be due to some degree of model misspecification which renders the held-out score a poor proxy for graph quality. Moreover, for some methods the gain from the hyperparameter tuning might be outweighed by the loss due to the 20% reduction in training samples.

Additional implementation details for held-out score evaluation: GraN-DAG makes use of a final pruning step to remove spurious edges. One could simply mask the inputs of the NN corresponding to removed edges and evaluate the held-out score. However, doing so yields an unrepresentative score since some masked inputs have an important role in the learned function and once these inputs are masked, the quality of the fit might greatly

suffer. To avoid this, we retrained the whole model from scratch on the training set with the masking fixed to the one recovered after pruning. Then, we evaluate the held-out score with this retrained architecture. During this retraining phase, the estimated graph is fixed, only the conditional densities are relearned. Since NOTEARS and DAG-GNN are not always guaranteed to return a DAG (although they almost always do), some extra thresholding might be needed as mentioned in Appendix A.1.8. Similarly to GraN-DAG's pruning phase, this step can seriously reduce the quality of the fit. To avoid this, we also perform a retraining phase for NOTEARS and DAG-GNN. The model of CAM is also retrained after its pruning phase prior to evaluating its held-out score.

Table A.12. Gauss-ANM - 10 nodes with hyperparameter search

Graph Type	ER1		ER4		SF1		SF4	
Metrics	SHD	SID	SHD	SID	SHD	SID	SHD	SID
Method								
GraN-DAG	$\mathbf{1.0 \pm 1.6}^{+}$	$\mathbf{0.4 \pm 1.3}^{++}$	$\mathbf{5.5 \pm 2.8}^{+}$	$\mathbf{9.7 \pm 8.0}^{++}$	$\mathbf{1.3 \pm 1.8}^{-}$	$\mathbf{3.0 \pm 3.4}^{+}$	$\mathbf{9.6 \pm 4.5}^{+}$	$\mathbf{15.1 \pm 6.1}^{+}$
DAG-GNN	$10.9 \pm 2.6^{+}$	$35.5 \pm 13.6^{+}$	$38.3 \pm 2.9^{--}$	$84.4 \pm 3.5^{-}$	$9.9 \pm 1.7^{+}$	$30.3 \pm 18.8^{-}$	$21.4 \pm 2.1^{-}$	$44.0 \pm 15.5^{+}$
NOTEARS	$26.7 \pm 6.9^{--}$	$35.2 \pm 10.6^{+}$	$20.9 \pm 6.6^{++}$	$62.0 \pm 6.7^{++}$	$20.4 \pm 9.6^{--}$	$38.8 \pm 16.7^{-}$	$26.9 \pm 7.4^{-}$	$61.1 \pm 13.8^{-}$
CAM	$3.0 \pm 4.2^{-}$	$2.2 \pm 5.7^{-}$	$\mathbf{7.7 \pm 3.1}^{++}$	$23.2 \pm 14.7^{+}$	$\mathbf{2.4 \pm 2.5}^{-}$	$\mathbf{5.2 \pm 5.5}^{+}$	$\mathbf{9.6 \pm 3.1}^{+}$	$\mathbf{20.1 \pm 6.8}^{-}$
GSF	$5.3 \pm 3.3^{+}$	$[8.3 \pm 13.2^{+}$ $15.4 \pm 13.5]$	$23.1 \pm 7.9^{-}$	$[56.1 \pm 20.4^{-}$ $65.1 \pm 19.3]$	$3.3 \pm 2.5^{-}$	$[7.0 \pm 11.6^{-}$ $12.2 \pm 11.0]$	$14.2 \pm 5.6^{--}$	$[26.2 \pm 11.1^{-}$ $36.9 \pm 21.6]$
GES	$38.6 \pm 2.1^{--}$	$[20.3 \pm 15.4^{+}$ $28.3 \pm 18.4]$	$33.0 \pm 3.4^{-}$	$[66.2 \pm 7.0^{+}$ $76.6 \pm 4.3]$	$38.3 \pm 2.4^{--}$	$[8.8 \pm 5.2^{-}$ $25.5 \pm 18.2]$	$33.6 \pm 4.8^{--}$	$[32.7 \pm 12.7^{-}$ $52.0 \pm 14.0]$

Table A.13. Gauss-ANM - 50 nodes with hyperparameter search

Graph Type	ER1		ER4		SF1		SF4	
Metrics	SHD	SID	SHD	SID	SHD	SID	SHD	SID
Method								
GraN-DAG	$\mathbf{3.8 \pm 3.3}^{+}$	$\mathbf{15.0 \pm 14.0}^{+}$	$\mathbf{105.6 \pm 16.5}^{-}$	$\mathbf{1131.7 \pm 91.0}^{-}$	$\mathbf{24.7 \pm 6.4}^{+}$	$\mathbf{86.5 \pm 34.6}^{+}$	$\mathbf{112.7 \pm 15.5}^{-}$	$\mathbf{268.3 \pm 85.8}^{+}$
DAG-GNN	$47.0 \pm 7.8^{+}$	$268.1 \pm 118.0^{+}$	$196.2 \pm 14.4^{-}$	$1972.8 \pm 110.6^{++}$	$51.8 \pm 5.6^{-}$	$166.5 \pm 48.9^{+}$	$144.2 \pm 11.6^{+}$	$473.4 \pm 105.4^{+}$
NOTEARS	$193.5 \pm 77.3^{--}$	$326.0 \pm 99.1^{+}$	$369.5 \pm 81.9^{--}$	$2062.0 \pm 107.7^{+}$	$104.8 \pm 22.4^{--}$	$290.3 \pm 136.8^{-}$	$213.0 \pm 35.1^{--}$	$722.7 \pm 177.3^{-}$
CAM	$\mathbf{4.0 \pm 2.7}^{+}$	$\mathbf{21.1 \pm 22.1}^{+}$	$\mathbf{105.6 \pm 20.9}^{-}$	$1225.9 \pm 205.7^{-}$	$\mathbf{23.8 \pm 6.0}^{+}$	$\mathbf{81.5 \pm 15.3}^{+}$	$\mathbf{112.2 \pm 14.0}^{-}$	$\mathbf{333.8 \pm 156.0}^{-}$
GSF	$24.9 \pm 7.4^{+}_{*}$	$[40.0 \pm 26.3^{-}_{*}$ $77.5 \pm 45.3]$	$129.3 \pm 20.4^{--}_{*}$	$[1280.8 \pm 202.3^{--}_{*}$ $1364.1 \pm 186.7]$	$35.3 \pm 6.9^{-}_{*}$	$[99.7 \pm 41.7^{-}_{*}$ $151.9 \pm 59.7]$	$\mathbf{121.6 \pm 11.7}^{-}_{***}$	$[\mathbf{310.8 \pm 108.1}^{-}_{***}$ $\mathbf{391.9 \pm 93.3}]$
GES	$1150.1 \pm 9.8^{--}$	$[112.7 \pm 71.1^{+}$ $132.0 \pm 89.0]$	$1066.1 \pm 11.7^{--}$	$[1394.3 \pm 81.8^{++}$ $1464.8 \pm 63.8]$	$1161.7 \pm 7.0^{--}$	$[322.8 \pm 211.1^{-}$ $336.0 \pm 215.4]$	$1116.1 \pm 14.2^{--}$	$[1002.7 \pm 310.9^{--}$ $1094.0 \pm 345.1]$

Table A.14. Gauss-ANM - 20 nodes with hyperparameter search

Graph Type	ER1		ER4		SF1		SF4	
Metrics	SHD	SID	SHD	SID	SHD	SID	SHD	SID
Method								
GraN-DAG	$\mathbf{2.7 \pm 2.3}^{+}$	$\mathbf{9.6 \pm 10.3}^{+}$	$\mathbf{35.9 \pm 11.8}^{+}$	$\mathbf{120.4 \pm 37.0}^{++}$	$\mathbf{6.5 \pm 2.4}^{+}$	$\mathbf{17.5 \pm 6.3}^{++}$	$\mathbf{35.6 \pm 4.1}^{+}$	$\mathbf{54.8 \pm 14.3}^{+}$
DAG-GNN	$21.0 \pm 6.1^{+}$	$98.8 \pm 42.2^{+}$	$77.2 \pm 6.5^{-}$	$345.6 \pm 18.6^{-}$	$19.1 \pm 0.7^{+}$	$55.0 \pm 20.1^{+}$	$50.2 \pm 5.4^{-}$	$118.7 \pm 33.2^{-}$
NOTEARS	$101.5 \pm 39.6^{--}$	$100.4 \pm 47.0^{+}$	$124.0 \pm 16.3^{--}$	$267.0 \pm 46.5^{++}$	$55.0 \pm 28.2^{--}$	$87.6 \pm 26.9^{-}$	$66.7 \pm 8.3^{--}$	$154.6 \pm 43.0^{-}$
CAM	$\mathbf{2.8 \pm 2.2}^{-}$	$\mathbf{11.5 \pm 10.2}^{-}$	$64.3 \pm 29.3^{-}$	$\mathbf{121.7 \pm 73.1}^{+}$	$\mathbf{5.5 \pm 1.6}^{+}$	$\mathbf{19.3 \pm 7.8}^{+}$	$\mathbf{36.0 \pm 5.1}^{-}$	$\mathbf{66.3 \pm 28.6}^{-}$
GSF	$11.6 \pm 3.0^{+}$	$[26.4 \pm 13.3^{-}$ $49.8 \pm 26.5]$	$\mathbf{46.2 \pm 12.6}^{-}$	$[172.7 \pm 40.8^{-}$ $213.5 \pm 38.6]$	$12.8 \pm 2.1^{--}$	$[32.1 \pm 14.0^{--}$ $56.2 \pm 13.8]$	$42.3 \pm 5.1^{-}$	$[68.9 \pm 27.7^{-}$ $95.1 \pm 33.8]$
GES	$169.9 \pm 5.0^{--}$	$[45.4 \pm 29.2^{+}$ $57.2 \pm 36.6]$	$142.8 \pm 7.7^{--}_{*}$	$[223.3 \pm 33.6^{++}_{*}$ $254.7 \pm 22.0]$	$168.1 \pm 3.3^{--}$	$[46.7 \pm 21.7^{+}$ $53.3 \pm 20.0]$	$162.2 \pm 10.4^{--}$	$[151.1 \pm 57.4^{--}$ $195.8 \pm 57.4]$

Table A.15. Gauss-ANM - 100 nodes with hyperparameter search

Graph Type	ER1		ER4		SF1		SF4	
Metrics	SHD	SID	SHD	SID	SHD	SID	SHD	SID
Method								
GraN-DAG	$\mathbf{15.1 \pm 7.5}^{+}$	$\mathbf{65.1 \pm 33.2}^{+}$	$\mathbf{191.6 \pm 17.8}^{+}$	$\mathbf{4090.7 \pm 418.0}^{+}$	$\mathbf{51.6 \pm 10.2}^{+}$	$\mathbf{210.6 \pm 51.9}^{++}$	$\mathbf{255.7 \pm 21.1}^{+}$	$\mathbf{790.5 \pm 159.7}^{+}$
DAG-GNN	$103.9 \pm 9.1^{+}$	$757.6 \pm 215.0^{+}$	$387.1 \pm 25.3^{-}$	$7741.9 \pm 522.5^{+}$	$103.5 \pm 8.2^{-}$	$391.7 \pm 60.0^{+}$	$314.8 \pm 16.3^{+}$	$1257.3 \pm 185.2^{+}$
NOTEARS	$421.3 \pm 207.0^{--}$	$945.7 \pm 339.7^{-}$	$631.1 \pm 136.6^{--}$	$8272.4 \pm 444.2^{-}$	$244.3 \pm 63.8^{--}$	$815.6 \pm 346.5^{-}$	$482.3 \pm 114.1^{--}$	$1929.7 \pm 363.1^{--}$
CAM	$\mathbf{12.3 \pm 4.9}^{++}$	$128.0 \pm 66.3^{-}$	$\mathbf{198.8 \pm 22.2}^{-}$	$4602.2 \pm 523.7^{-}$	$\mathbf{51.1 \pm 9.4}^{+}$	$\mathbf{233.6 \pm 62.3}^{-}$	$\mathbf{255.7 \pm 22.2}^{-}$	$\mathbf{851.4 \pm 206.0}^{-}$
GSF	$100.2 \pm 9.9_{**}^{--}$	$[719.8 \pm 242.1_{**}^{--}$ $721.1 \pm 242.9]$	$387.6 \pm 23.9_{***}$	$[7535.1 \pm 595.2_{***}$ $7535.1 \pm 595.2]$	$67.3 \pm 14.0_{***}^{+}$	$[254.5 \pm 35.4_{***}^{--}$ $340.4 \pm 70.4]$	$315.1 \pm 16.7_{***}^{--}$	$[1214.0 \pm 156.4_{***}^{--}$ $1214.0 \pm 156.4]$
GES	$4782.5 \pm 22.9^{--}$	$[362.3 \pm 267.7^{+}$ $384.1 \pm 293.6]$	$4570.1 \pm 27.9^{--}$	$[5400.7 \pm 299.2^{++}$ $5511.5 \pm 308.5]$	$4769.1 \pm 26.7^{--}$	$[1311.1 \pm 616.6^{--}$ $1386.2 \pm 713.9]$	$4691.3 \pm 47.3^{--}$	$[3882.7 \pm 1010.6^{--}$ $3996.7 \pm 1075.7]$

Table A.16. PNL-GP with hyperparameter search

#Nodes	10				50			
Graph Type	ER1		ER4		ER1		ER4	
Metrics	SHD	SID	SHD	SID	SHD	SID	SHD	SID
Method								
GraN-DAG	$\mathbf{1.2 \pm 2.2}^{+}$	$\mathbf{1.9 \pm 4.2}^{+}$	$\mathbf{9.8 \pm 4.9}^{+}$	$\mathbf{29.0 \pm 17.6}^{+}$	$12.8 \pm 4.9^{+}$	$55.3 \pm 24.2^{+}$	$73.9 \pm 16.8^{-}$	$\mathbf{1107.2 \pm 144.7}^{+}$
DAG-GNN	$10.6 \pm 4.9^{+}$	$35.8 \pm 19.6^{-}$	$38.6 \pm 2.0^{--}$	$82.2 \pm 5.7^{--}$	$48.1 \pm 8.4^{+}$	$330.4 \pm 69.9^{+}$	$192.5 \pm 19.2^{+}$	$2079.5 \pm 120.9^{+}$
NOTEARS	$20.6 \pm 11.4^{-}$	$30.5 \pm 18.8^{+}$	$24.2 \pm 6.5^{++}$	$66.4 \pm 6.9^{++}$	$102.1 \pm 27.3^{--}$	$299.8 \pm 85.8^{+}$	$660.0 \pm 258.2^{--}$	$1744.0 \pm 232.9^{++}$
CAM	$\mathbf{2.7 \pm 4.0}^{-}$	$6.4 \pm 11.8^{+}$	$\mathbf{8.7 \pm 4.5}^{-}$	$\mathbf{30.9 \pm 20.4}^{-}$	$\mathbf{4.0 \pm 2.4}^{+}$	$\mathbf{10.7 \pm 12.4}^{+}$	$\mathbf{52.3 \pm 8.5}^{-}$	$\mathbf{913.9 \pm 209.3}^{-}$
GSF	$12.9 \pm 3.9_{***}^{--}$	$[10.5 \pm 8.7_{***}^{--}$ $53.6 \pm 23.8]$	$40.7 \pm 1.3_{**}^{--}$	$[79.2 \pm 3.8_{**}^{--}$ $79.2 \pm 3.8]$	$48.8 \pm 3.9_{**}^{--}$	$[281.6 \pm 70.7_{**}^{--}$ $281.6 \pm 70.7]$	$199.9 \pm 15.2_{***}^{--}$	$[1878.0 \pm 122.4_{***}^{--}$ $1948.4 \pm 139.6]$

Table A.17. PNL-MULT with hyperparameter search

#Nodes	10				50			
Graph Type	ER1		ER4		ER1		ER4	
Metrics	SHD	SID	SHD	SID	SHD	SID	SHD	SID
Method								
GraN-DAG	$\mathbf{10.0 \pm 4.5}^{+}$	$\mathbf{29.1 \pm 9.7}^{+}$	$\mathbf{32.9 \pm 3.3}^{-}$	$76.7 \pm 4.1^{+}$	$\mathbf{59.8 \pm 28.2}^{-}$	$\mathbf{213.6 \pm 97.3}^{+}$	$272.1 \pm 69.4^{-}$	$2021.6 \pm 185.8^{+}$
DAG-GNN	$14.6 \pm 3.1^{++}$	$\mathbf{36.9 \pm 10.6}^{+}$	$38.9 \pm 2.0^{-}$	$85.8 \pm 1.2^{--}$	$\mathbf{64.3 \pm 27.8}^{+}$	$508.8 \pm 317.2^{-}$	$\mathbf{212.5 \pm 12.3}^{++}$	$2216.9 \pm 95.6^{+}$
NOTEARS	$28.8 \pm 9.1^{--}$	$\mathbf{30.3 \pm 11.8}^{+}$	$\mathbf{35.4 \pm 3.8}^{+}$	$78.4 \pm 7.5^{+}$	$160.2 \pm 67.5^{--}$	$443.5 \pm 205.1^{-}$	$229.2 \pm 25.4^{-}$	$2158.8 \pm 70.3^{+}$
CAM	$17.2 \pm 8.0^{-}$	$\mathbf{33.7 \pm 14.4}^{+}$	$\mathbf{32.3 \pm 6.5}^{+}$	$76.6 \pm 8.2^{+}$	$97.5 \pm 71.1^{-}$	$\mathbf{282.3 \pm 123.8}^{+}$	$251.0 \pm 25.9^{--}$	$2026.2 \pm 58.2^{+}$
GSF	$15.6 \pm 4.4_{**}^{--}$	$[\mathbf{10.0 \pm 6.3}_{**}^{--}$ $\mathbf{60.1 \pm 17.2}]$	$39.3 \pm 2.2_{**}^{--}$	$[76.0 \pm 9.6_{**}^{--}$ $79.9 \pm 5.3]$	$\mathbf{66.4 \pm 14.4}_{***}^{--}$	$[145.1 \pm 96.1_{***}^{--}$ $618.8 \pm 257.0]$	> 12 hours	

Table A.18. LIN with hyperparameter search

#Nodes	10				50			
Graph Type	ER1		ER4		ER1		ER4	
Metrics	SHD	SID	SHD	SID	SHD	SID	SHD	SID
Method								
GraN-DAG	$\mathbf{10.1 \pm 3.9}^{-}$	$\mathbf{28.7 \pm 14.7}^{-}$	$34.7 \pm 2.9^{--}$	$79.5 \pm 4.4^{-}$	$\mathbf{40.8 \pm 10.3}^{-}$	$\mathbf{236.3 \pm 101.7}^{+}$	$256.9 \pm 55.7^{--}$	$2151.4 \pm 144.3^{-}$
DAG-GNN	$\mathbf{9.0 \pm 2.7}^{++}$	$\mathbf{35.6 \pm 11.4}^{-}$	$\mathbf{19.6 \pm 4.6}^{+}$	$\mathbf{63.9 \pm 7.5}^{-}$	$48.3 \pm 6.8^{+}$	$381.7 \pm 145.4^{-}$	$\mathbf{149.7 \pm 17.2}^{++}$	$2070.7 \pm 51.9^{--}$
NOTEARS	$\mathbf{14.0 \pm 4.1}_{*}^{--}$	$\mathbf{32.2 \pm 7.9}_{*}^{+}$	$\mathbf{20.7 \pm 5.1}^{++}$	$\mathbf{63.1 \pm 8.0}^{++}$	$87.7 \pm 44.3_{*}^{-}$	$\mathbf{294.3 \pm 99.3}_{*}^{+}$	$200.3 \pm 67.1^{--}$	$\mathbf{1772.7 \pm 143.7}^{++}$
CAM	$\mathbf{8.8 \pm 6.0}^{+}$	$\mathbf{25.8 \pm 13.5}^{+}$	$33.9 \pm 2.8^{-}$	$77.1 \pm 4.5^{+}$	$\mathbf{34.8 \pm 7.0}^{+}$	$\mathbf{221.2 \pm 98.3}^{+}$	$202.2 \pm 14.3^{--}$	$1990.8 \pm 97.5^{-}$
GSF	$\mathbf{10.7 \pm 3.5}^{-}$	$[\mathbf{15.8 \pm 8.4}^{-}$ $\mathbf{45.2 \pm 20.2}]$	$33.4 \pm 3.3^{++}$	$[71.7 \pm 11.5^{+}$ $77.3 \pm 6.1]$	$54.4 \pm 6.5_{*}^{--}$	$[158.1 \pm 115.9_{*}^{-}$ $560.9 \pm 220.7]$	$195.6 \pm 9.9_{**}$	$[2004.9 \pm 85.2_{**}$ $2004.9 \pm 85.2]$

Table A.19. ADD-FUNC with hyperparameter search

#Nodes	10				50			
Graph Type	ER1		ER4		ER1		ER4	
Metrics	SHD	SID	SHD	SID	SHD	SID	SHD	SID
Method								
GraN-DAG	$\mathbf{2.6 \pm 2.4}^{+}$	$\mathbf{4.3 \pm 4.3}^{+}$	$\mathbf{7.0 \pm 3.1}^{++}$	$\mathbf{37.1 \pm 12.4}^{++}$	$\mathbf{13.2 \pm 6.7}^{+}$	$\mathbf{72.1 \pm 55.2}^{+}$	$90.1 \pm 25.6^{-}$	$1241.7 \pm 289.8^{+}$
DAG-GNN	$8.7 \pm 2.8^{++}$	$22.3 \pm 9.4^{+}$	$25.3 \pm 3.8^{++}$	$63.6 \pm 8.6^{++}$	$44.7 \pm 9.7^{++}$	$306.9 \pm 114.7^{+}$	$194.0 \pm 20.4^{+}$	$1949.3 \pm 107.1^{+}$
NOTEARS	$21.2 \pm 11.5^{-}_{*}$	$15.5 \pm 9.9^{+}_{*}$	$13.3 \pm 4.3^{++}$	$\mathbf{41.3 \pm 11.5}^{++}$	$186.8 \pm 83.0^{--}$	$276.9 \pm 92.1^{-}$	$718.4 \pm 170.4^{--}$	$1105.9 \pm 250.1^{++}$
CAM	$\mathbf{3.0 \pm 2.2}^{-}$	$\mathbf{8.1 \pm 6.3}^{-}$	$\mathbf{6.2 \pm 5.5}^{-}$	$\mathbf{28.5 \pm 21.5}^{+}$	$\mathbf{10.0 \pm 4.6}^{-}$	$\mathbf{44.2 \pm 32.1}^{-}$	$\mathbf{46.6 \pm 9.5}^{-}$	$\mathbf{882.5 \pm 186.5}^{-}$
GSF	$5.5 \pm 4.1^{+}$	[$7.5 \pm 12.3^{+}$ 16.3 ± 12.9]	$19.1 \pm 7.0^{++}$	[$44.5 \pm 19.7^{+}$ 60.4 ± 16.5]	$29.8 \pm 7.6^{++}_{*}$	[$\mathbf{44.6 \pm 42.6}^{++}_{*}$ $\mathbf{96.8 \pm 46.7}$]	$140.4 \pm 31.7_{***}$	[$1674.4 \pm 133.9_{***}$ 1727.6 ± 145.2]

Table A.20. Results for real and pseudo real data sets with hyperparameter search

Data Type	Protein signaling data set			SynTReN - 20 nodes		
Metrics	SHD	SHD-C	SID	SHD	SHD-C	SID
Method						
GraN-DAG	12.0^{+}	$\mathbf{9.0}^{+}$	48.0^{-}	$41.2 \pm 9.6^{-}$	$43.7 \pm 8.3^{-}$	$144.3 \pm 61.3^{+}$
GraN-DAG++	14.0^{-}	11.0^{-}	57.0^{-}	$46.9 \pm 14.9^{-}$	$49.5 \pm 14.7^{-}$	$158.4 \pm 61.5^{-}$
DAG-GNN	16.0	14.0^{+}	59.0^{-}	$\mathbf{32.2 \pm 5.0}^{++}$	$\mathbf{32.3 \pm 5.6}^{++}$	$194.2 \pm 50.2^{-}$
NOTEARS	15.0^{+}	14.0^{+}	58.0^{-}	$44.2 \pm 27.5^{++}$	$45.8 \pm 27.7^{++}$	$183.1 \pm 48.4^{--}$
CAM	$\mathbf{11.0}^{+}$	**9.0**	51.0^{+}	$101.7 \pm 37.2^{--}$	$105.6 \pm 36.6^{--}$	$\mathbf{111.5 \pm 25.3}^{++}$
GSF	20.0^{-}	14.0^{-}	[37.0^{+} 60.0]	$\mathbf{27.8 \pm 5.4}^{++}_{*}$	$\mathbf{27.8 \pm 5.4}^{++}_{*}$	[$207.6 \pm 55.4^{--}_{*}$ 209.6 ± 59.1]
GES	47.0^{-}	50.0^{-}	[$\mathbf{37.0}^{+}$ **47.0**]	$167.5 \pm 5.6^{--}$	$172.2 \pm 7.0^{--}$	[$\mathbf{75.3 \pm 24.4}^{++}$ $\mathbf{97.6 \pm 30.8}$]

Table A.21. Hyperparameter search spaces for each algorithm

	Hyperparameter space
GraN-DAG	Log(learning rate) $\sim U[-2,-3]$ (first subproblem) Log(learning rate) $\sim U[-3,-4]$ (other subproblems) $\epsilon \sim U\{10^{-3}, 10^{-4}, 10^{-5}\}$ Log(pruning cutoff) $\sim U\{-5,-4,-3,-2,-1\}$ # hidden units $\sim U\{4,8,16,32\}$ # hidden layers $\sim U\{1,2,3\}$ Constraint convergence tolerance $\sim U\{10^{-6}, 10^{-8}, 10^{-10}\}$ PNS threshold $\sim U[0.5, 0.75, 1, 2]$
DAG-GNN	Log(learning rate) $\sim U[-4,-2]$ # hidden units in encoder $\sim U\{16,32,64,128,256\}$ # hidden units in decoder $\sim U\{16,32,64,128,256\}$ Bottleneck dimension (dimension of Z) $\sim U\{1,5,10,50,100\}$ Constraint convergence tolerance $\sim U\{10^{-6}, 10^{-8}, 10^{-10}\}$
NOTEARS	L1 regularizer coefficient $\sim U\{0.001, 0.005, 0.01, 0.05, 0.1, 0.5\}$ Final threshold $\sim U\{0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1\}$ Constraint convergence tolerance $\sim U\{10^{-6}, 10^{-8}, 10^{-10}\}$
CAM	Log(Pruning cutoff) $\sim U[-6, 0]$
GSF	Log(RKHS regression regularizer) $\sim U[-4, 4]$
GES	Log(Regularizer coefficient) $\sim U[-4, 4]$

Appendix B

Article 2: Matériel supplémentaire

B.1. Theory

B.1.1. Theoretical Foundations for Causal Discovery with Imperfect Interventions

Before showing results about our regularized maximum likelihood score from Section 4.3.1, we start by briefly presenting useful definitions and results from Yang et al. ((2018)). We refer the reader to the original paper for a more comprehensive introduction to these notions, examples, and proofs. Throughout the appendix, we assume that the reader is comfortable with the concept of d-separation and immorality in directed graphs. Recall that we always assume $\emptyset \in \mathcal{I}$ and $I_1 := \emptyset$. We use the notation $i \to j \in \mathcal{G}$ to indicate that the edge (i,j) is in the edge set of $\mathcal{G}$. Given disjoint $A,B,C \subset V$, when C d-separates A from B in graph $\mathcal{G}$ we write $A \perp\!\!\!\perp_{\mathcal{G}} B \mid C$ and when random variables X_A and X_B are independent given X_C in distribution f, we write $X_A \perp\!\!\!\perp_f X_B \mid X_C$.

Definition 4. For a DAG $\mathcal{G}$ and an interventional family $\mathcal{I}$, let

$$\mathcal{M}_{\mathcal{I}}(\mathcal{G}) := \{(f^{(k)})_{k\in[K]} \mid f^{(k)}(x_1, \cdots, x_d) = \prod_{j\notin I_k} f_j^{(1)}(x_j \mid x_{\pi_j^{\mathcal{G}}}) \prod_{j\in I_k} f_j^{(k)}(x_j \mid x_{\pi_j^{\mathcal{G}}}) \;\; \forall k \in [K]\}$$

Definition 4 defines a set $\mathcal{M}_{\mathcal{I}}(\mathcal{G})$ which contains all the sets of distributions $(f^{(k)})_{k\in[K]}$ which are coherent with the definition of interventions provided at Equation (4.2.2). Note that the assumption of causal sufficiency is implicit to this definition of interventions. Analogously to the observational case, two different DAGs $\mathcal{G}_1$ and $\mathcal{G}_2$ can induce the same interventional distributions.

Definition 5. ($\mathcal{I}$-Markov Equivalence Class) Two DAGs $\mathcal{G}_1$ and $\mathcal{G}_2$ are *$\mathcal{I}$-Markov equivalent* iff $\mathcal{M}_{\mathcal{I}}(\mathcal{G}_1) = \mathcal{M}_{\mathcal{I}}(\mathcal{G}_2)$. We denote by $\mathcal{I}$-MEC($\mathcal{G}_1$) the set of all DAGs which are $\mathcal{I}$-Markov equivalent to $\mathcal{G}_1$, this is the *$\mathcal{I}$-Markov equivalence class* of $\mathcal{G}_1$.

We now define an augmented graph containing exactly one node for each intervention k.

Definition 6. Given a DAG $\mathcal{G}$ and an interventional family $\mathcal{I}$, the associated *$\mathcal{I}$-DAG*, denoted by $\mathcal{G}^{\mathcal{I}}$, is the graph $\mathcal{G}$ augmented with nodes ζ_k and edges $\zeta_k \to i$ for all $k \in [K]\setminus\{1\}$ and all $i \in I_k$.

In the observational case, we say that a distribution f has the Markov property w.r.t. a graph $\mathcal{G}$ if whenever some d-separation holds in the graph, the corresponding conditional independence holds in f. The next definition generalizes this idea to interventions.

Definition 7. ($\mathcal{I}$-Markov property) Let $\mathcal{I}$ be interventional family such that $\emptyset \in \mathcal{I}$ and $(f^{(k)})_{k\in[K]}$ be a set of strictly positive densities over X. We say that $(f^{(k)})_{k\in[K]}$ satisfies the *$\mathcal{I}$-Markov property* w.r.t. the $\mathcal{I}$-DAG $\mathcal{G}^{\mathcal{I}}$ iff

1. For any disjoint $A, B, C \subset V$, $A \perp\!\!\!\perp_{\mathcal{G}} B|C$ implies $X_A \perp\!\!\!\perp_{f^{(k)}} X_B|X_C$ for all $k \in [K]$.

Yang et al. ((2018)) defines $\mathcal{M}_{\mathcal{I}}(\mathcal{G})$ slightly differently, but show their definition to be equivalent to the one used here. See Lemma A.1 in Yang et al. ((2018))

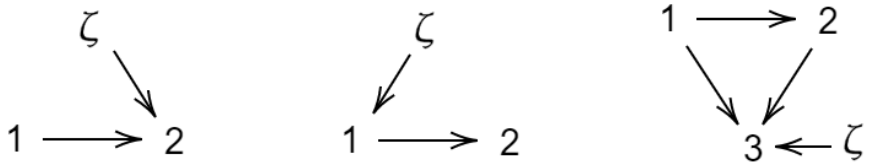


Fig. B.1. Different $\mathcal{I}$-DAGs with a single intervention. The first graph is alone in its $\mathcal{I}$-Markov equivalence class since reversing the $1 \to 2$ edge would break the immorality $1 \to 2 \leftarrow \zeta$. The second graph is also alone in its equivalence class since reversing $1 \to 2$ would create a new immorality $\zeta \to 1 \leftarrow 2$. The third DAG is not alone in its equivalence class since reversing $1 \to 2$ would preserve the skeleton without adding or removing an immorality. It should become apparent that adding more interventions will likely reduce the size of the $\mathcal{I}$-Markov equivalence class by introducing more immoralities.

2. For any disjoint $A, C \subset V$ and $k \in [K] \setminus \{1\}$,
$A \perp\!\!\!\perp_{\mathcal{G}^{\mathcal{I}}} \zeta_k \mid C \cup \zeta_{-k}$ implies $f^{(k)}(X_A|X_C) = f^{(1)}(X_A|X_C)$, where $\zeta_{-k} := \zeta_{[K]\setminus\{1,k\}}$.

The next proposition relates the definition of interventions with the $\mathcal{I}$-Markov property that we just defined.

Proposition 8. (Yang et al. ((2018))) Suppose $\emptyset \in \mathcal{I}$. Then $(f^{(k)})_{k\in[K]} \in \mathcal{M}_{\mathcal{I}}(\mathcal{G})$ iff $(f^{(k)})_{k\in[K]}$ is $\mathcal{I}$-Markov to $\mathcal{G}^{\mathcal{I}}$.

The next theorem gives a graphical characterization of $\mathcal{I}$-Markov equivalence classes.

Theorem 9. (Yang et al. ((2018))) Suppose $\emptyset \in \mathcal{I}$. Two DAGs $\mathcal{G}_1$ and $\mathcal{G}_2$ are $\mathcal{I}$-Markov equivalent iff their $\mathcal{I}$-DAGs $\mathcal{G}_1^{\mathcal{I}}$ and $\mathcal{G}_2^{\mathcal{I}}$ share the same skeleton and immoralities.

See Figure B.1 for a simple illustration of this concept. We now present a very simple corollary which gives a situation where the $\mathcal{I}$-Markov equivalence class contains a unique graph.

Corollary 1. Let $\mathcal{G}$ be a DAG and let $\mathcal{I} = (\emptyset, \{1\}, \cdots, \{d\})$. Then $\mathcal{G}$ is alone in its $\mathcal{I}$-Markov equivalence class.

Proof. By Theorem 9, all $\mathcal{I}$-Markov equivalent graphs will share its skeleton with $\mathcal{G}$, so we consider only graphs obtained by reversing edges in $\mathcal{G}$.

Consider any edge $i \to j$ in $\mathcal{G}$. We note that $i \to j \leftarrow \zeta_{j+1}$ forms an immorality in the $\mathcal{I}$-DAG $\mathcal{G}^{\mathcal{I}}$. Reversing $i \to j$ would break this immorality which would imply that the resulting DAG is not $\mathcal{I}$-Markov equivalent to $\mathcal{G}$, by Theorem 9. Hence, $\mathcal{G}$ is alone in its equivalence class.■

B.1.2. Proof of Theorem 3

We are now ready to present the main result of this section. We recall the score function introduced in Section 4.3.1:

$$\mathcal{S}_{\text{int}}(\mathcal{G}) := \max_{\phi} \sum_{k=1}^{K} \mathbb{E}_{X\sim p^{(k)}} \log f^{(k)}(X; M^{\mathcal{G}}, \phi) - \lambda|\mathcal{G}|\,, \tag{B.1.1}$$

$$\text{where} \quad f^{(k)}(x; M^{\mathcal{G}}, \phi) := \prod_{j\notin I_k} \tilde{f}(x_j; \text{NN}(M_j^{\mathcal{G}} \odot x; \phi_j)) \prod_{j\in I_k} \tilde{f}(x_j; \text{NN}(M_j^{\mathcal{G}} \odot x; \phi_j^{(k)}))\,. \tag{B.1.2}$$

Recall that $(p^{(k)})_{k\in[K]}$ are the ground truth interventional distributions with ground truth graph $\mathcal{G}^*$. We define $\mathcal{F}_{\mathcal{I}}(\mathcal{G})$ to be the set of all $(f^{(k)})_{k\in[K]}$ which are expressible by the model specified in Equation (B.1.2). More precisely,

$$\mathcal{F}_{\mathcal{I}}(\mathcal{G}) := \{(f^{(k)})_{k\in[K]} \mid \exists\ \phi \text{ s.t. } \forall\ k \in [K]\ f^{(k)}(x) = f^{(k)}(x; M^{\mathcal{G}}, \phi)\}\,. \tag{B.1.3}$$

It should be clear from the definitions that $\mathcal{F}_{\mathcal{I}}(\mathcal{G}) \subset \mathcal{M}_{\mathcal{I}}(\mathcal{G})$. Thus, from Proposition 8 we see that the $\mathcal{I}$-Markov property holds for all $(f^{(k)})_{k\in[K]} \in \mathcal{F}_{\mathcal{I}}(\mathcal{G})$. This fact will be useful in the proof of Theorem 3.

Theorem 3 relies on two assumptions. The first one requires that model is expressive enough to represent the ground truth distributions exactly and the second one is a faithfulness assumption similar in spirit to those encountered in standard structure learning.

Assumption 1. (Sufficient capacity) The model specified in Equation (B.1.2) is expressive enough to represent the ground truth distributions, i.e. $(p^{(k)})_{k\in[K]} \in \mathcal{F}_{\mathcal{I}}(\mathcal{G}^*)$. Moreover, we assume each $p^{(k)}$ has finite entropy.

Assumption 2. ($\mathcal{I}$-Faithfulness)

(1) For any disjoint $A, B, C \subset V$,

$$A \not\perp\!\!\!\perp_{\mathcal{G}^*} B|C \ \text{ implies } \ \exists\ k \in [K] \text{ s.t. } X_A \not\perp\!\!\!\perp_{p^{(k)}} X_B|X_C\,.$$

(2) For any disjoint $A, C \subset V$ and $k \in [K]$,

$$A \not\perp\!\!\!\perp_{\mathcal{G}^*\mathcal{I}} \zeta_k \mid C \cup \zeta_{-k} \text{ implies } p^{(k)}\left(X_A|X_C\right) \neq p^{(1)}\left(X_A|X_C\right).$$

The first condition resembles the standard faithfulness assumption, except that we only require that the corresponding conditional independence does not hold in a given intervention distribution. The second condition is simply the converse of the second condition in the $\mathcal{I}$-Markov property (Definition 7).

The next lemma shows that the difference $\mathcal{S}_{\text{int}}(\mathcal{G}^*) - \mathcal{S}_{\text{int}}(\mathcal{G})$ can be rewritten as a minimization of a sum of KL divergences plus the difference in regularizing terms. We slightly compress the notation by dropping the $_{\text{int}}$ subscript and writing $f^{(k)}_{\mathcal{G}\phi}$ to refer the model joint $f^{(k)}(x; \mathcal{G}, \phi)$.

Lemma 10. Under Assumption 1, we have

$$\mathcal{S}(\mathcal{G}^*) - \mathcal{S}(\mathcal{G}) = \min_{\phi} \sum_{k\in[K]} D_{KL}(p^{(k)}||f^{(k)}_{\mathcal{G}\phi}) + \lambda(|\mathcal{G}| - |\mathcal{G}^*|)\,. \tag{B.1.4}$$

Proof.

$$\begin{aligned}
\mathcal{S}(\mathcal{G}^*) - \mathcal{S}(\mathcal{G}) =& \mathcal{S}(\mathcal{G}^*) - \sum_{k\in[K]} \mathbb{E}_{p^{(k)}} \log p^{(k)}(X) - \mathcal{S}(\mathcal{G}) + \sum_{k\in[K]} \mathbb{E}_{p^{(k)}} \log p^{(k)}(X) && \text{(B.1.5)}\\
=& \max_{\phi} \sum_{k\in[K]} \mathbb{E}_{p^{(k)}} \log f^{(k)}_{\mathcal{G}^*\phi}(X) - \sum_{k\in[K]} \mathbb{E}_{p^{(k)}} \log p^{(k)}(X)\\
& - \max_{\phi} \sum_{k\in[K]} \mathbb{E}_{p^{(k)}} \log f^{(k)}_{\mathcal{G}\phi}(X) + \sum_{k\in[K]} \mathbb{E}_{p^{(k)}} \log p^{(k)}(X)\\
& + \lambda(|\mathcal{G}| - |\mathcal{G}^*|) && \text{(B.1.6)}\\
=& \min_{\phi} - \sum_{k\in[K]} \mathbb{E}_{p^{(k)}} \log f^{(k)}_{\mathcal{G}\phi}(X) + \sum_{k\in[K]} \mathbb{E}_{p^{(k)}} \log p^{(k)}(X)\\
& - \min_{\phi} - \sum_{k\in[K]} \mathbb{E}_{p^{(k)}} \log f^{(k)}_{\mathcal{G}^*\phi}(X) - \sum_{k\in[K]} \mathbb{E}_{p^{(k)}} \log p^{(k)}(X)\\
& + \lambda(|\mathcal{G}| - |\mathcal{G}^*|) && \text{(B.1.7)}\\
=& \min_{\phi} \sum_{k\in[K]} D_{KL}(p^{(k)}||f^{(k)}_{\mathcal{G}\phi}) - \min_{\phi} \sum_{k\in[K]} D_{KL}(p^{(k)}||f^{(k)}_{\mathcal{G}^*\phi})\\
& + \lambda(|\mathcal{G}| - |\mathcal{G}^*|) && \text{(B.1.8)}
\end{aligned}$$

By Assumption 1 $(p^{(k)})_{k\in[K]} \in \mathcal{F}_{\mathcal{I}}(\mathcal{G}^*)$ which implies that $\min_{\phi} \sum_{k\in[K]} D_{KL}(p^{(k)}||f^{(k)}_{\mathcal{G}^*\phi}) = 0$.■

The following definition will be useful for the next lemma.

Definition 11. Given a DAG $\mathcal{G}$ with node set V and two nodes i,$j \in V$, we define the following sets:

$$T^{\mathcal{G}}_{ij} := \{\ell \in V \mid \text{the immorality } i \to \ell \leftarrow j \text{ is in } \mathcal{G}\} \tag{B.1.9}$$

$$L^{\mathcal{G}}_{ij} := \mathbf{DE}_{\mathcal{G}}(T^{\mathcal{G}}_{ij}) \cup \{i,j\}\,, \tag{B.1.10}$$

where $\mathbf{DE}_{\mathcal{G}}(S)$ is the set of descendants of S in $\mathcal{G}$, including S itself.

Lemma 12. Let $\mathcal{G}$ be a DAG with node set V. When $i \to j \notin \mathcal{G}$ and $i \leftarrow j \notin \mathcal{G}$ we have

$$i \perp\!\!\!\perp_{\mathcal{G}} j \mid V \setminus L^{\mathcal{G}}_{ij}\,. \tag{B.1.11}$$

Proof: By contradiction. Suppose there is a path from $(i = a_0, a_1, ..., a_p = j)$ with $p > 1$ which is not d-blocked by $V \setminus L^{\mathcal{G}}_{ij}$ in $\mathcal{G}$. We first consider the case where the path contains no colliders.

If the path contains no colliders, then $a_0 \leftarrow a_1$ or $a_{p-1} \to a_p$. Moreover, since the path is not d-blocked and both a_1 and a_{p-1} are not colliders, $a_1, a_{p-1} \in L^{\mathcal{G}}_{ij}$. But this implies that there is a directed path from $i = a_0$ to a_1 and a directed path from $j = a_p$ to a_{p-1}. This

creates a directed cycle: either $a_0 \to \cdots \to a_1 \to a_0$ or $a_p \to \cdots \to a_{p-1} \to a_p$. This is a contradiction since $\mathcal{G}$ is acyclic.

Suppose there is a collider a_k, i.e. $a_{k-1} \to a_k \leftarrow a_{k+1}$. Since the path is not d-blocked, there must exists a node $z \in \mathbf{DE}_{\mathcal{G}}(a_k) \cup \{a_k\}$ such that $z \notin L_{ij}^{\mathcal{G}}$. If $i = a_{k-1}$ and $j = a_{k+1}$, then clearly $z \in L_{ij}^{\mathcal{G}}$, which is a contradiction. Otherwise, $i \neq a_{k-1}$ or $j \neq a_{k+1}$. Without loss of generality, assume $i \neq a_{k-1}$. Clearly, a_{k-1} is not a collider and since the path is not d-blocked, $a_{k-1} \in L_{ij}^{\mathcal{G}}$. But by definition, $L_{ij}^{\mathcal{G}}$ also contains all the descendants of a_{k-1} including z. Again, this is a contradiction with $z \notin L_{ij}^{\mathcal{G}}$.■

We recall Theorem 1 from Section 4.3.1 and present its proof.

Theorem 3. Let $\mathcal{G}^*$ be the ground truth DAG and $\hat{\mathcal{G}} \in \arg\max_{\mathcal{G} \in \mathrm{DAG}} \mathcal{S}_{\mathrm{int}}(\mathcal{G})$. Under Assumptions 1 & 2 and for $\lambda > 0$ small enough, $\hat{\mathcal{G}}$ is $\mathcal{I}$-Markov equivalent to $\mathcal{G}^*$.

Proof. It is sufficient to prove that, for all $\mathcal{G} \notin \mathcal{I}\text{-MEC}(\mathcal{G}^*)$, $\mathcal{S}(\mathcal{G}^*) > \mathcal{S}(\mathcal{G})$. We use Theorem 9 which states that $\hat{\mathcal{G}}$ is not $\mathcal{I}$-Markov equivalent to $\mathcal{G}^*$ if and only if $\hat{\mathcal{G}}^{\mathcal{I}}$ does not share its skeleton or its immoralities with $\mathcal{G}^{*\mathcal{I}}$. The proof is organized in six cases. Cases 1-2 treat when $\mathcal{G}$ and $\mathcal{G}^*$ do not share the same skeleton, cases 3 & 4 when their immoralities differ and cases 5 & 6 when their immoralities implying interventional nodes ζ_k differ. In almost every cases, the idea is the same:

(1) Use Lemma 12 to find a d-separation which holds in $\mathcal{G}^{\mathcal{I}}$ and show it does not hold in $\mathcal{G}^{*\mathcal{I}}$;
(2) Use the fact that $\mathcal{F}_{\mathcal{I}}(\mathcal{G}) \subset \mathcal{M}_{\mathcal{I}}(\mathcal{G})$, Proposition 8 and the $\mathcal{I}$-faithfulness assumption to obtain an invariance which holds for all $(f^{(k)})_{k\in[K]} \in \mathcal{F}_{\mathcal{I}}(\mathcal{G})$ but not in $(p^{(k)})_{k\in[K]}$;
(3) Use the fact that the invariance forces $\min_\phi \sum_{k\in[K]} D_{KL}(p^{(k)}||f_{\mathcal{G}\phi}^{(k)})$ to be greater than zero and;
(4) Conclude that $\mathcal{S}(\mathcal{G}^*) > \mathcal{S}(\mathcal{G})$ via Lemma 10.

Case 1: We consider the graphs $\mathcal{G}$ such that there exists $i \to j \in \mathcal{G}^*$ but $i \to j \notin \mathcal{G}$ and $i \leftarrow j \notin \mathcal{G}$. Let $\mathbb{G}$ be the set of all such $\mathcal{G}$. By Lemma 12, $i \perp\!\!\!\perp_{\mathcal{G}} j \mid V \setminus L_{ij}^{\mathcal{G}}$ but clearly $i \not\perp\!\!\!\perp_{\mathcal{G}^*} j \mid V \setminus L_{ij}^{\mathcal{G}}$. Hence, i) by $\mathcal{I}$-faithfulness (Assumption 2) we have $X_i \not\perp\!\!\!\perp_{p^{(k_0)}} X_j | X_{V\setminus L_{ij}^{\mathcal{G}}}$ for some $k_0 \in [K]$ and ii) by the $\mathcal{I}$-Markov property (Proposition 8) we have $X_i \perp\!\!\!\perp_{f^{(k_0)}} X_j | X_{V\setminus L_{ij}^{\mathcal{G}}}$ for all $(f^{(k)})_{k\in[K]} \in \mathcal{F}_{\mathcal{I}}(\mathcal{G})$. This means that $\min_\phi D_{KL}(p^{(k_0)}||f_{\mathcal{G}\phi}^{(k_0)}) > 0$. For notation convenience, let us define

$$\eta(\mathcal{G}) := \min_\phi \sum_{k\in[K]} D_{KL}(p^{(k)}||f_{\mathcal{G}\phi}^{(k)})\,. \tag{B.1.12}$$

Note that

$$\eta(\mathcal{G}) \geq \min_\phi D_{KL}(p^{(k_0)}||f_{\mathcal{G}\phi}^{(k_0)}) > 0\,, \tag{B.1.13}$$

where the first inequality holds by non-negativity of the KL divergence. Using Lemma 10, we can write

$$\mathcal{S}(\mathcal{G}^*) - \mathcal{S}(\mathcal{G}) = \eta(\mathcal{G}) + \lambda(|\mathcal{G}| - |\mathcal{G}^*|)\,. \tag{B.1.14}$$

If $|\mathcal{G}| \geq |\mathcal{G}^*|$ then clearly $\mathcal{S}(\mathcal{G}^*) - \mathcal{S}(\mathcal{G}) > 0$. Let $\mathbb{G}^+ := \{\mathcal{G} \in \mathbb{G} \mid |\mathcal{G}| < |\mathcal{G}^*|\}$. To make sure we have $\mathcal{S}(\mathcal{G}^*) - \mathcal{S}(\mathcal{G}) > 0$ for all $\mathcal{G} \in \mathbb{G}^+$, we need to pick λ sufficiently small. Choosing $0 < \lambda < \min_{\mathcal{G}\in\mathbb{G}^+} \frac{\eta(\mathcal{G})}{|\mathcal{G}^*|-|\mathcal{G}|}$ is sufficient since

$$\lambda < \min_{\mathcal{G}\in\mathbb{G}^+} \frac{\eta(\mathcal{G})}{|\mathcal{G}^*| - |\mathcal{G}|} \tag{B.1.15}$$

$$\iff \lambda < \frac{\eta(\mathcal{G})}{|\mathcal{G}^*| - |\mathcal{G}|} \quad \forall \mathcal{G} \in \mathbb{G}^+ \tag{B.1.16}$$

$$\iff \lambda(|\mathcal{G}^*| - |\mathcal{G}|) < \eta(\mathcal{G}) \quad \forall \mathcal{G} \in \mathbb{G}^+ \tag{B.1.17}$$

$$\iff 0 < \eta(\mathcal{G}) + \lambda(|\mathcal{G}| - |\mathcal{G}^*|) = \mathcal{S}(\mathcal{G}^*) - \mathcal{S}(\mathcal{G}) \quad \forall \mathcal{G} \in \mathbb{G}^+\,. \tag{B.1.18}$$

Case 2: We consider the graphs $\mathcal{G}$ such that there exists $i \to j \in \mathcal{G}$ but $i \to j \notin \mathcal{G}^*$ and $i \leftarrow j \notin \mathcal{G}^*$. We can assume $i \to j \in \mathcal{G}^*$ implies $i \to j \in \mathcal{G}$ or $i \leftarrow j \in \mathcal{G}$, since otherwise we are in Case 1. Hence, it means $|\mathcal{G}| > |\mathcal{G}^*|$ which in turn implies that $\mathcal{S}(\mathcal{G}^*) > \mathcal{S}(\mathcal{G})$.

Cases 1 and 2 completely cover the situations where $\mathcal{G}^{\mathcal{I}}$ and $\mathcal{G}^{*\mathcal{I}}$ do not share the same skeleton. Next, we assume that $\mathcal{G}^{\mathcal{I}}$ and $\mathcal{G}^{*\mathcal{I}}$ do have the same skeleton (which implies that $|\mathcal{G}| = |\mathcal{G}^*|$). The remaining cases treat the differences in immoralities.

Case 3: Suppose $\mathcal{G}^*$ contains an immorality $i \to \ell \leftarrow j$ which is not present in $\mathcal{G}$. We first show that $\ell \notin L_{ij}^{\mathcal{G}}$. Suppose the opposite. This means ℓ is a descendant of both i and j in $\mathcal{G}$. Since $\mathcal{G}$ and $\mathcal{G}^*$ share skeleton and because $i \to \ell \leftarrow j$ is not an immorality in $\mathcal{G}$, we have that $i \leftarrow \ell \in \mathcal{G}$ or $\ell \to j \in \mathcal{G}$, which in both cases creates a cycle. This is a contradiction.

The path (i,ℓ,j) is not d-blocked by $V \setminus L_{ij}^{\mathcal{G}}$ in $\mathcal{G}^*$ since $\ell \in V \setminus L_{ij}^{\mathcal{G}}$. By $\mathcal{I}$-faithfulness (Assumption 2), this means that $X_i \not\perp\!\!\!\perp_{p^{(k_0)}} X_j \mid X_{V\setminus L_{ij}^{\mathcal{G}}}$ for some $k_0 \in [K]$. Since $\mathcal{G}^*$ and $\mathcal{G}$ share the same skeleton, we know $i \to j$ and $i \leftarrow j$ are not in $\mathcal{G}$. Using Lemma 12 and the $\mathcal{I}$-Markov property (Proposition 8), we have that $X_i \perp\!\!\!\perp_{f^{(k_0)}} X_j \mid X_{V\setminus L_{ij}^{\mathcal{G}}}$ for all $(f^{(k)})_{k\in[K]} \in \mathcal{F}_{\mathcal{I}}(\mathcal{G})$. Similarly to Case 1, this implies that $\eta(\mathcal{G}) > 0$ which in turn implies that $\mathcal{S}(\mathcal{G}^*) - \mathcal{S}(\mathcal{G}) > 0$ (using the fact $|\mathcal{G}^*| = |\mathcal{G}|$).

Case 4: Suppose $\mathcal{G}$ contains an immorality $i \to \ell \leftarrow j$ which is not present in $\mathcal{G}^*$. Since $\mathcal{G}$ and $\mathcal{G}^*$ share the same skeleton and $\ell \notin V \setminus L_{ij}^{\mathcal{G}}$, we know there is a (potentially undirected) path (i,ℓ,j) which is not d-blocked by $V \setminus L_{ij}^{\mathcal{G}}$ in $\mathcal{G}^*$. By $\mathcal{I}$-faithfulness (Assumption 2), we know that $X_i \not\perp\!\!\!\perp_{p^{(k_0)}} X_j \mid X_{V\setminus L_{ij}^{\mathcal{G}}}$ for some $k_0 \in [K]$. However, by Lemma 12 and the $\mathcal{I}$-Markov property (Proposition 8), we have that $X_i \perp\!\!\!\perp_{f^{(k_0)}} X_j \mid X_{V\setminus L_{ij}^{\mathcal{G}}}$ for all $(f^{(k)})_{k\in[K]} \in \mathcal{F}_{\mathcal{I}}(\mathcal{G})$. Thus, again, $\mathcal{S}(\mathcal{G}^*) - \mathcal{S}(\mathcal{G}) > 0$.

So far, all cases did not require interventional nodes ζ_k. Cases 5 and 6 treat the difference in immoralities implying interventional nodes ζ_k. Note that the arguments are analog to cases 5 and 6.

Case 5: Suppose that there is an immorality $i \to \ell \leftarrow \zeta_j$ in $\mathcal{G}^{*\mathcal{I}}$ which does not appear in $\mathcal{G}^{\mathcal{I}}$. The path (i,ℓ,ζ_j) is not d-blocked by $\zeta_{-j} \cup V \setminus L^{\mathcal{G}^{\mathcal{I}}}_{i\zeta_j}$ in $\mathcal{G}^{*\mathcal{I}}$ since $\ell \in \zeta_{-j} \cup V \setminus L^{\mathcal{G}^{\mathcal{I}}}_{i\zeta_j}$ (by same argument as presented in Case 3). By $\mathcal{I}$-faithfulness (Assumption 2), this means that

$$p^{(1)}(x_i \mid x_{V \setminus L^{\mathcal{G}^{\mathcal{I}}}_{i\zeta_j}}) \neq p^{(j)}(x_i \mid x_{V \setminus L^{\mathcal{G}^{\mathcal{I}}}_{i\zeta_j}}) \,. \tag{B.1.19}$$

On the other hand, Lemma 12 implies that $i \perp\!\!\!\perp_{\mathcal{G}^{\mathcal{I}}} \zeta_j \mid \zeta_{-j} \cup V \setminus L^{\mathcal{G}^{\mathcal{I}}}_{i\zeta_j}$. By the $\mathcal{I}$-Markov property (Proposition 8), we have that

$$f^{(1)}(x_i \mid x_{V \setminus L^{\mathcal{G}^{\mathcal{I}}}_{i\zeta_j}}) = f^{(j)}(x_i \mid x_{V \setminus L^{\mathcal{G}^{\mathcal{I}}}_{i\zeta_j}}) \quad \text{for all } (f^{(k)})_{k\in[K]} \in \mathcal{F}_{\mathcal{I}}(\mathcal{G}) \,. \tag{B.1.20}$$

This means that $\mathcal{S}(\mathcal{G}^*) > \mathcal{S}(\mathcal{G})$ since

$$\begin{aligned} \mathcal{S}(\mathcal{G}^*) - \mathcal{S}(\mathcal{G}) &= \min_{\phi} \sum_{k\in[K]} D_{KL}(p^{(k)} || f^{(k)}_{\mathcal{G}\phi}) && \text{(B.1.21)} \\ &\geq \min_{\phi} D_{KL}(p^{(1)} || f^{(1)}_{\mathcal{G}\phi}) + D_{KL}(p^{(j)} || f^{(j)}_{\mathcal{G}\phi}) && \text{(B.1.22)} \\ &> 0 \,, && \text{(B.1.23)} \end{aligned}$$

where the last equality holds because both divergences cannot be put to zero simultaneously.

Case 6: Suppose that there is an immorality $i \to \ell \leftarrow \zeta_j$ in $\mathcal{G}^{\mathcal{I}}$ which does not appear in $\mathcal{G}^{*\mathcal{I}}$. The path (i, ℓ, ζ_j) is not d-blocked by $\zeta_{-j} \cup V \setminus L^{\mathcal{G}^{\mathcal{I}}}_{i\zeta_j}$ in $\mathcal{G}^{*\mathcal{I}}$, since $\ell \notin \zeta_{-j} \cup V \setminus L^{\mathcal{G}^{\mathcal{I}}}_{i\zeta_j}$ and both $\mathcal{I}$-DAGs share the same skeleton. It follows by $\mathcal{I}$-faithfulness (Assumption 2) that

$$p^{(1)}(x_i \mid x_{V \setminus L^{\mathcal{G}^{\mathcal{I}}}_{i\zeta_j}}) \neq p^{(j)}(x_i \mid x_{V \setminus L^{\mathcal{G}^{\mathcal{I}}}_{i\zeta_j}}) \,. \tag{B.1.24}$$

On the other hand, Lemma 12 implies that $i \perp\!\!\!\perp_{\mathcal{G}^{\mathcal{I}}} \zeta_j \mid \zeta_{-j} \cup V \setminus L^{\mathcal{G}^{\mathcal{I}}}_{i\zeta_j}$. Again by the $\mathcal{I}$-Markov property (Proposition 8), it means that

$$f^{(1)}(x_i \mid x_{V \setminus L^{\mathcal{G}^{\mathcal{I}}}_{i\zeta_j}}) = f^{(j)}(x_i \mid x_{V \setminus L^{\mathcal{G}^{\mathcal{I}}}_{i\zeta_j}}) \quad \text{for all } (f^{(k)})_{k\in[K]} \in \mathcal{F}_{\mathcal{I}}(\mathcal{G}) \,. \tag{B.1.25}$$

By an argument identical to that of Case 5, it follows that $\mathcal{S}(\mathcal{G}^*) > \mathcal{S}(\mathcal{G})$.

The proof is complete since there is no other way in which $\mathcal{G}^{\mathcal{I}}$ and $\mathcal{G}^{*\mathcal{I}}$ can differ in terms of skeleton and immoralities.■

B.2. Additional information

B.2.1. Synthetic data sets

In this section, we describe how the different synthetic data sets were generated. For each type of data set, we first sample a DAG following the *Erdős-Rényi* scheme and then we sample the parameters of the different causal mechanisms as stated below (in the bulleted list). For 10-node graphs, single node interventions are performed on every node. For 20-node graphs, interventions target 1 to 2 nodes chosen uniformely at random. Then, $n/(d+1)$ examples are sampled for each interventional setting (if n is not divisible by $d+1$, some intervention setting may have one extra sample in order to have a total of n samples). The data are then normalized: we subtract the mean and divide by the standard deviation. For all data sets, the source nodes are Gaussian with zero mean and variance sampled from $\mathcal{U}[1,2]$. The noise variables N_j are mutually independent and sampled from $\mathcal{N}(0, \sigma_j^2) \ \ \forall j$, where $\sigma_j^2 \sim \mathcal{U}[1,2]$.

For perfect intervention, the distribution of intervened nodes is replaced by a marginal $\mathcal{N}(2,1)$. This type of intervention, that produce a mean-shift, is similar to those used in Hauser et Bühlmann ((2012)); Squires et al. ((2020)). For imperfect interventions, besides the initial parameters, an extra set of parameters were sampled by perturbing the initial parameters as described below. For nodes without parents, the distribution of intervened nodes is replaced by a marginal $\mathcal{N}(2,1)$. Both for the perfect and imperfect cases, we explore other types of interventions and report the results in Appendix B.3.5. We now describe the causal mechanisms and the nature of the imperfect intervention for the three different types of data set:

- The *linear* data sets are generated following $X_j := w_j^T X_{\pi_j^{\mathcal{G}}} + 0.4 \cdot N_j \ \ \forall j$, where w_j is a vector of $|\pi_j^{\mathcal{G}}|$ coefficients each sampled uniformly from $[-1,-0.25] \cup [0.25,1]$ (to make sure there are no w close to 0). Imperfect interventions are obtained by adding a random vector of $\mathcal{U}([-5,-2] \cup [2,5])$ to w_j.
- The *additive noise model* (ANM) data sets are generated following $X_j := f_j(X_{\pi_j^{\mathcal{G}}}) + 0.4 \cdot N_j \ \forall j$, where the functions f_j are fully connected neural networks with one hidden layer of 10 units and *leaky ReLU* with a negative slope of 0.25 as nonlinearities. The weights of each neural network are randomly initialized from $\mathcal{N}(0,1)$. Imperfect interventions are obtained by adding a random vector of $\mathcal{N}(0,1)$ to the last layer.
- The *nonlinear with non-additive noise* (NN) data sets are generated following $X_j := f_j(X_{\pi_j^{\mathcal{G}}}, N_j) \ \forall j$, where the functions f_j are fully connected neural networks with one hidden layer of 20 units and *tanh* as nonlinearities. The weights of each neural network are randomly initialized from $\mathcal{N}(0,1)$. Similarly to the additive noise model, imperfect intervention are obtained by adding a random vector of $\mathcal{N}(0,1)$ to the last layer.

B.2.2. Deep Sigmoidal Flow: Architectural details

A layer of a Deep Sigmoidal Flow is similar to a fully-connected network with one hidden layer, a single input, and a single output, but is defined slightly differently to ensure that the mapping is invertible and that the Jacobian is tractable. Each layer l is defined as follows:

$$h^{(l)}(x) = \sigma^{-1}(w^T \sigma(a \cdot x + b)) \,, \tag{B.2.1}$$

where $0 < w_i < 1$, $\sum_i w_i = 1$ and $a_i > 0$. In our method, the neural networks $\mathrm{NN}(\cdot; \phi_j^{(k)})$ output the parameters (w_j, a_j, b_j) for each DSF τ_j. To ensure that the determinant of the Jacobian is calculated in a numerically-stable way, we follow the recommendations of Huang et al. ((2018b)). While other flows like the Deep Dense Sigmoidal Flow have more capacity, DSF was sufficient for our use.

B.2.3. Optimization

In this section, we show how the augmented Lagrangian is applied, how the gradient is estimated and, finally, we illustrate the learning dynamics by analyzing an example.

Let us recall the score and the optimization problem from Section 4.3.2:

$$\hat{\mathcal{S}}_{\text{int}}(\Lambda) := \max_{\phi} \mathop{\mathbb{E}}_{M \sim \sigma(\Lambda)} \left[\sum_{k=1}^{K} \mathop{\mathbb{E}}_{X \sim p^{(k)}} \log f^{(k)}(X; M, \phi) - \lambda ||M||_0 \right] , \tag{B.2.2}$$

$$\max_{\Lambda} \hat{\mathcal{S}}_{\text{int}}(\Lambda) \quad \text{s.t.} \quad \operatorname{Tr} e^{\sigma(\Lambda)} - d = 0 \,. \tag{B.2.3}$$

We optimize for ϕ and Λ jointly, which yields the following optimization problem:

$$\max_{\phi, \Lambda} \mathop{\mathbb{E}}_{M \sim \sigma(\Lambda)} \left[\sum_{k=1}^{K} \mathop{\mathbb{E}}_{X \sim p^{(k)}} \log f^{(k)}(X; M, \phi) \right] - \lambda ||\sigma(\Lambda)||_1 \quad \text{s.t.} \quad \operatorname{Tr} e^{\sigma(\Lambda)} - d = 0 \,, \tag{B.2.4}$$

where we used the fact that $\mathbb{E}_{M \sim \sigma(\Lambda)} ||M||_0 = ||\sigma(\Lambda)||_1$. Let us use the notation:

$$h(\Lambda) := \operatorname{Tr} e^{\sigma(\Lambda)} - d. \tag{B.2.5}$$

The augmented Lagrangian transforms the constrained problem into a sequence of unconstrained problems of the form

$$\max_{\phi, \Lambda} \mathop{\mathbb{E}}_{M \sim \sigma(\Lambda)} \left[\sum_{k=1}^{K} \mathop{\mathbb{E}}_{X \sim p^{(k)}} \log f^{(k)}(X; M, \phi) \right] - \lambda ||\sigma(\Lambda)||_1 - \gamma_t h(\Lambda) - \frac{\mu_t}{2} h(\Lambda)^2 \,, \tag{B.2.6}$$

where γ_t and μ_t are the Lagrangian multiplier and the penalty coefficient of the tth unconstrained problem, respectively. In all our experiments, we initialize $\gamma_0 = 0$ and $\mu_0 = 10^{-8}$. Each such problem is approximately solved using a stochastic gradient descent algorithm (RMSprop ((Tieleman et Hinton, 2012b)) in our experiments). We consider that a subproblem has converged when (B.2.6) evaluated on a held-out data set stops increasing. Let (ϕ_t^*, Λ_t^*) be the approximate solution to subproblem t. Then, γ_t and μ_t are updated according

to the following rule:

$$\begin{aligned}\gamma_{t+1} &\leftarrow \gamma_t + \mu_t \cdot h\left(\Lambda_t^*\right) \\ \mu_{t+1} &\leftarrow \begin{cases} \eta \cdot \mu_t, & \text{if } h\left(\Lambda_t^*\right) > \delta \cdot h\left(\Lambda_{t-1}^*\right) \\ \mu_t, & \text{otherwise} \end{cases}\end{aligned} \tag{B.2.7}$$

with $\eta = 2$ and $\delta = 0.9$. Each subproblem t is initialized using the previous subproblem's solution $(\phi_{t-1}^*, \Lambda_{t-1}^*)$. The augmented Lagrangian method stops when $h(\Lambda) \leq 10^{-8}$ and the graph formed by adding an edge whenever $\sigma(\Lambda) > 0.5$ is acyclic.
Gradient estimation. The gradient of (B.2.6) w.r.t. ϕ and Λ is estimated by

$$\nabla_{\phi,\Lambda}\left[\frac{1}{|B|}\sum_{i\in B}\log f^{(k_i)}(x^{(i)}; M^{(i)}, \phi) - \lambda_t h(\Lambda) - \frac{\mu_t}{2}h(\Lambda)^2\right], \tag{B.2.8}$$

where B is an index set sampled without replacement, $x^{(i)}$ is an example from the training set and k_i is the index of its corresponding intervention. To compute the gradient of the likelihood part w.r.t. Λ, we use the Straight-Through Gumbel-Softmax estimator, adapted to sigmoids ((Maddison et al., 2017; Jang et al., 2017)). This approach was already used in the context of causal discovery without interventional data ((Ng et al., 2019; Kalainathan et al., 2018b)). The matrix $M^{(i)}$ is given by

$$M^{(i)} := \mathbb{I}(\sigma(\Lambda + L^{(i)}) > 0.5) + \sigma(\Lambda + L^{(i)}) - \text{grad-block}(\sigma(\Lambda + L^{(i)}))\,, \tag{B.2.9}$$

where $L^{(i)}$ is a $d \times d$ matrix filled with independent Logistic samples, $\mathbb{I}$ is the indicator function applied element-wise and the function *grad-block* is such that grad-block$(z) = z$ and ∇_zgrad-block$(z) = 0$. This implies that each entry of $M^{(i)}$ evaluates to a discrete Bernoulli sample with probability given by $\sigma(\Lambda)$ while the gradient w.r.t. Λ is computed using the soft Gumbel-Softmax sample. This yields a biased estimation of the actual gradient of objective (B.2.6), but its variance is low compared to the popular unbiased REINFORCE estimator (a Monte Carlo estimator relying on the log-trick) ((Rezende et al., 2014; Maddison et al., 2017)). A temperature term can be added inside the sigmoid, but we found that a temperature of one gave good results.

We considered a different relaxation for the discrete variable M. We tried treating M directly as a learnable parameter constrained in [0,1] via gradient projection. But this approach yielded significantly worse results. We believe the fact M is continuous is problematic, because as an entry of M gets closer and closer to zero, the weights of the first neural network layer can compensate, without affecting the likelihood whatsoever. This cannot happen when using the Straight-Through Gumbel-Softmax estimator because the neural network weights are only exposed to discrete M.

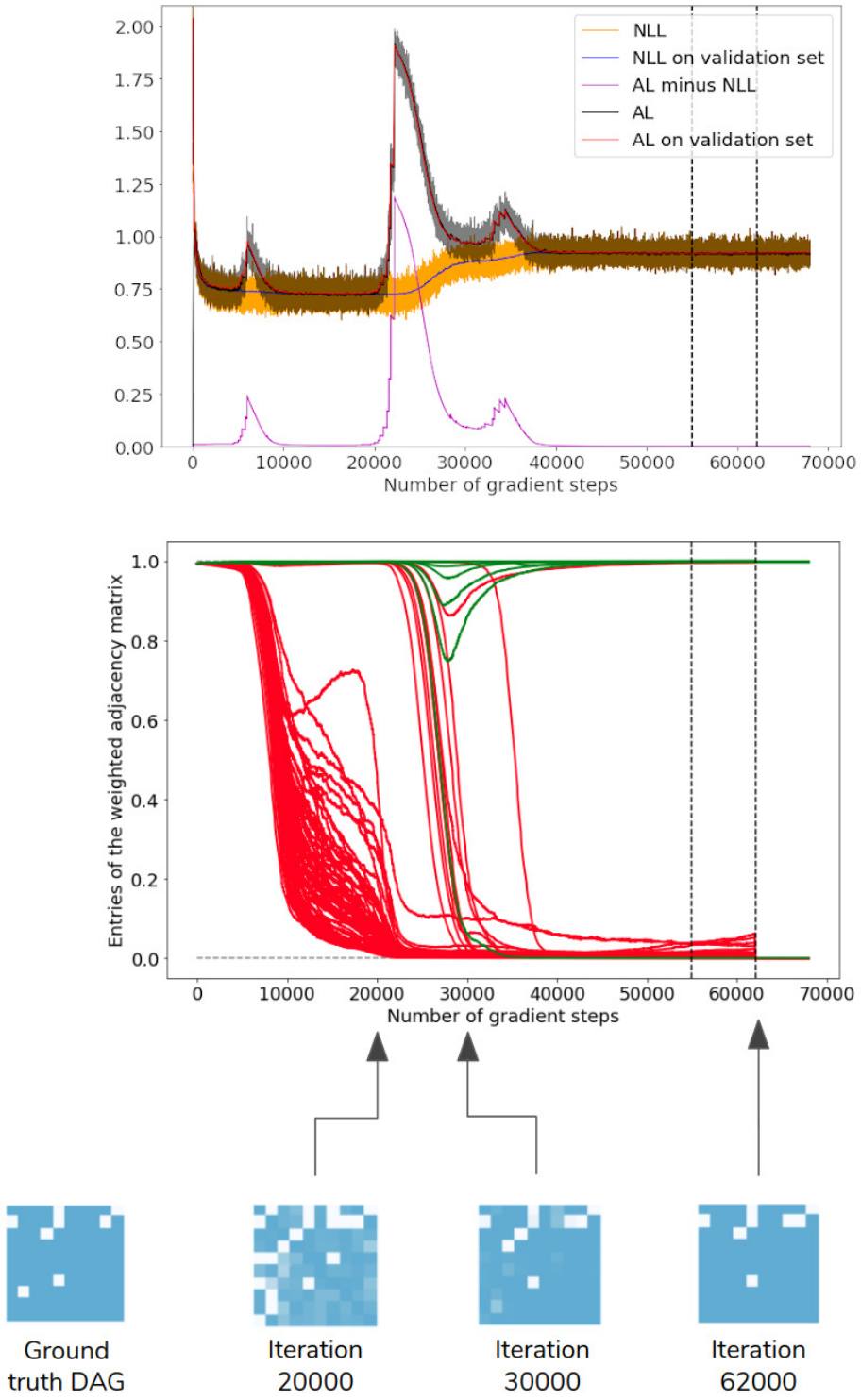


Fig. B.2. Top: Learning curves during training. *NLL* and *NLL on validation* are respectively the (pseudo) negative log-likelihood (NLL) on training and validation sets. *AL minus NLL* can be thought of as the acyclicity constraint violation plus the edge sparsity regularizer. *AL* and *AL on validation set* are the augmented Lagrangian objectives on training and validation set, respectively. **Middle and bottom:** Entries of the matrix $\sigma(\Lambda)$ w.r.t. to the number of iterations (green edges = edge present in the ground truth DAG, red edges = edge not present). The adjacency matrix to the left correspond to the ground truth DAG. The other matrices correspond to $\sigma(\Lambda)$ at 20 000, 30 000 and 62 000 iterations.

Learning dynamics. We present in Figure B.2 the learning curves (top) and the matrix $\sigma(\Lambda)$ (middle and bottom) as DCDI-DSF is trained on a linear data set with perfect intervention sampled from a sparse 10-node graph (the same phenomenon was observed in a wide range of settings). In the graph at the top, we show the augmented Lagrangian and the (pseudo) negative log-likelihood (NLL) on train and validation set. To be exact, the NLL corresponds to a negative log-likelihood only once acyclicity is achieved. In the graph representing $\sigma(\Lambda)$ (middle), each curve represent a $\sigma(\alpha_{ij})$: green edges are edges present in the ground truth DAG and red edges are edges not present. The same information is presented in matrix form for a few specific iterations and can be easily compared to the adjacency matrix of the ground truth DAG (white = presence of an edge, blue = absence). Recall that when a $\sigma(\alpha_{ij})$ is equal (or close to) 0, it means that the entry ij of the mask M will also be 0. This is equivalent to say that the edge is not present in the learned DAG.

In this section we review some important steps of the learning dynamics. At first, the NLL on the training and validation sets decrease sharply as the model fits the data. Around the iteration 5000, the decrease slows down and the weights of the constraint (namely γ and μ) are increased. This puts pressure on the entries $\sigma(\alpha_{ij})$ to decrease. At iteration 20 000, many $\sigma(\alpha_{ij})$ that correspond to red edges have diminished close to 0, meaning that edges are correctly removed. It is noteworthy to mention that the matrix at this stage is close to being symmetric: the algorithm did not yet choose an orientation for the different edges. While this learned graph still has false positive edges, the skeleton is reminiscent of a Markov Equivalence Class. As the training progresses, the weights of the constraint are greatly increased passed the 20 000th iteration leading to the removal of additional edges (leading also to an NLL increase). Around iteration 62 000 (the second vertical line), the stopping criterion is met: the acyclicity constraint is below the threshold (i.e. $h(\Lambda) \leq 10^{-8}$), the learned DAG is acyclic and the augmented Lagrangian on the validation set is not improving anymore. Edges with a $\sigma(\alpha_{ij})$ higher than 0.5 are set to 1 and others set to 0. The learned DAG has a SHD of 1 since it has a reversed edge compared to the ground truth DAG.

Finally, we illustrate the learning of interventional targets in the (perfect) unknown intervention setting by comparing an example of $\sigma(\beta_{kj})$, the learned targets, with the ground truth targets in Figure B.3. Results are from DCDI-G on 10-node graph with higher connectivity. Each column correspond to an interventional target I_k and each row correspond to a node. In the right matrix, a dark grey square in position ij means that the node i was intervened on in the interventional setting I_j. Each entry of the left matrix corresponds to the value of $\sigma(\beta_{kj})$. The binary matrix R (from Equation 4.3.5) is sampled following these entries.

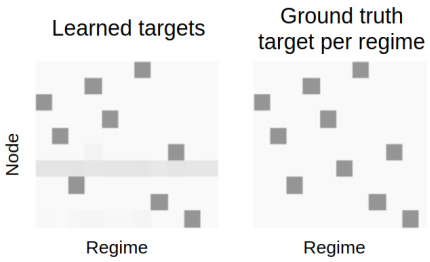


Fig. B.3. Learned targets $\sigma(\beta_{kj})$ compared to the ground truth targets.

B.2.4. Baseline methods

In this section, we provide additional details on the baseline methods and cite the implementations that were used. GIES has been designed for the perfect interventions setting. It assumes linear relations with Gaussian noise and outputs an $\mathcal{I}$-Markov equivalence classes. In order to obtain the SHD and SID, we compare a DAG randomly sampled from the returned $\mathcal{I}$-Markov equivalence classes to the ground truth DAG. CAM has been modified to support perfect interventions. In particular, we modified the loss similarly to the loss proposed for DCDI in the perfect intervention setting. Also, the preliminary neighbor search (PNS) and pruning processes were modified to not take into account data where variables are intervened on. Note that, while these two methods yield competitive results in the imperfect intervention setting, they were designed for perfect interventions: the targeted conditional are not fitted by an additional model (in contrast to our proposed score), they are simply removed from the score.

For GIES, we used the implementation from the R package `pcalg`. For CAM, we modified the implementation from the R package `pcalg`. For IGSP and UT-IGSP, we used the implementation from `https://github.com/uhlerlab/causaldag`. The cutoff values used for `alpha-inv` was always the same as `alpha`. The normalizing flows that we used for DCDI-DSF were adapted from the DSF implementation provided by its author ((Huang et al., 2018b)). We also used several tools from `https://github.com/FenTechSolutions/CausalDiscoveryToolbox` to interface R with Python and to compute the SHD and SID metrics.

B.2.5. Default hyperparameters and hyperparameter search

For all score-based methods, we performed a hyperparameter search. The models were trained on 80% examples and evaluated on the 20% remaining examples. The hyperparameter combination chosen was the one that induced the lowest negative log-likelihood on the held-out examples. For DCDI, a grid search was performed over 10 values of the regularization coefficient (see Table B.1) for known interventions (10 hyperparameter combinations in total) and, in the unknown intervention case, 3 values for the regularization coefficient of

the learned targets λ_R were also explored (30 hyperparameter combinations in total). For GIES and CAM, 50 hyperparameter combinations were considered using a random search following the sampling scheme of Table B.1.

For IGSP and UT-IGSP, we could not do a similar hyperparameter search since there is no score available to rank hyperparameter combinations. Thus, all examples were used to fit the model. Despite this, we explored a range of cutoff values around 10^{-5} (the value used for all the experiments in Squires et al. ((2020))): $\log_{10}\alpha = \{-2, -3, -5, -7, -9\}$. We chose 10^{-3}, which yielded low SHD and SID. Note that in a realistic setting, we do not have access to the ground truth graphs to make that decision.

Table B.1. Hyperparameter search spaces for each algorithm

	Hyperparameter space
DCDI	$\log_{10}(\lambda) \sim \mathcal{U}\{-7, -6, -5, -4, -3, -2, -1, 0, 1, 2\}$ $\log_{10}(\lambda_R) \sim \mathcal{U}\{-4, -3, -2\}$ (only for unknown interventions)
CAM	$\log_{10}(\text{pruning cutoff}) \sim \mathcal{U}[-7, 0]$
GIES	$\log_{10}(\text{regularizer coefficient}) \sim \mathcal{U}[-4, 4]$

Except for the normalizing flows of DCDI-DSF, DCDI-G and DCDI-DSF used exaclty the same default hyperparameters that are summarized in Table B.2. Some of these hyperparameters (μ_0, γ_0), which are related to the optimization process are presented in Appendix B.2.3. These hyperparameters were used for almost all experiments, except for the real-world data set and the two-node graphs with complex densities, where overfitting was observed. Smaller architectures were tested until no major overfitting was observed. The default hyperparameters were chosen using small-scale experiments on perfect-known interventions data sets in order to have a small SHD. Since we observed that DCDI is not highly sensible to changes in hyperparameter values, only the regularization factors were part of a more thorough hyperparameter search. The neural networks were initialized following the Xavier initialization ((Glorot et Bengio, 2010a)). The neural network activation functions were leaky-ReLU. RMSprop was used as the optimizer ((Tieleman et Hinton, 2012b)) with minibatches of size 64.

B.3. Additional experiments

B.3.1. Real-world data set

We tested the methods that support perfect intervention on the flow cytometry data set of Sachs et al. ((2005b)). The measurements are the level of expression of phosphoproteins and phospholipids in human cells. Interventions were performed by using reagents to activate or inhibit the measured proteins. As in Wang et al. ((2017)), we use a subset of the data

Table B.2. Default Hyperparameter for DCDI-G and DCDI-DSF

DCDI hyperparameters
μ_0: 10^{-8}, γ_0: 0, η: 2, δ: 0.9
Augmented Lagrangian constraint threshold: 10^{-8}
learning rate: 10^{-3}
hidden units: 16
hidden layers: 2
flow hidden units: 16 (only for DCDI-DSF)
flow hidden layers: 2 (only for DCDI-DSF)

set, excluding experimental conditions where the perturbations were not directly done on a measured protein. This subset comprises 5 846 measurements: 1 755 measurements are considered observational, while the other 4 091 measurements are from five different single node interventions (with the following proteins as targets: Akt, PKC, PIP2, Mek, PIP3). The concensus graph from Sachs et al. ((2005b)) that we use as the ground truth DAG contains 11 nodes and 17 edges. While the flow cytometry data sets is standard in the causal structure learning literature, some concerns have been raised. The "consensus" network proposed by Sachs et al. ((2005b)) has been challenged by some experts ((Mooij et al., 2016)). Also, several assumptions of the different models may not be respected in this real-world data set (for more details, see Mooij et al. ((2016))): i) the causal sufficiency assumption may not hold, ii) the interventions may not be as specific as stated, and iii) the ground truth network is possibly not a DAG since feedback loops are common in cellular signaling networks.

Table B.3. Results for the flow cytometry data sets

Method	SHD	SID	tp	fn	fp	rev	F_1 score
IGSP	18	54	4	6	5	7	0.42
GIES	38	34	10	0	41	7	0.33
CAM	35	20	12	1	30	4	0.51
DCDI-G	36	43	6	2	25	9	0.31
DCDI-DSF	33	47	6	2	22	9	0.33

In Table B.3 we report SHD and SID for all methods, along with the number of true positive (tp), false negative (fn), false positive (fp), reversed (rev) edges, and the F_1 score. There are no measures of central tendencies, since there is only one graph. The modified version of CAM has overall the best performance: the highest F_1 score and a low SID. IGSP has a low SHD, but a high SID, which can be explained by the relatively high number of false negative. DCDI-G and DCDI-DSF have SHDs comparable to GIES and CAM, but higher than IGSP. In terms of SID, they outperform IGSP, but not GIES and CAM. Finally, the DCDI models have F_1 scores similar to that of GIES. Hence, we conclude that DCDI

performs comparably to the state of the art on this data set, while none of the methods show great performance across the board.

Hyperparameters. We report the hyperparameters used for Table B.3. IGSP used the KCI-test with a cutoff value of 10^{-3}. Hyperparameters for CAM and GIES were chosen following the hyperparameter search described in Appendix B.2.5. For DCDI, since overfitting was observed, we included some hyperparameters related to the the architecture in the hyper-parameter grid search (number of hidden units: $\{4, 8\}$, number of hidden layers: {1,2} and only for DSF, number of flow hidden units: $\{4, 8\}$, number of flow layers: {1,2}), and used the scheme described in Appendix B.2.5 for choosing the regularization coefficient.

B.3.2. Learning causal direction from complex distributions

To show that insufficient capacity can hinder learning the right causal direction, we used toy data sets with simple 2-node graphs under perfect and imperfect interventions. We show, in Figure B.4 and B.5, the joint densities respectively learned by DCDI-DSF and DCDI-G. We tested two different data sets: X and DNA, which corresponds to the left and right column, respectively. In both data sets, we experimented with perfect and imperfect interventions, on both the cause and the effect, i.e. $\mathcal{I} = (\emptyset, \{1\}, \{2\})$. In both article2/figures, the top row corresponds to the learned densities when no intervention are performed. The bottom row corresponds to the learned densities under an imperfect intervention on the effect variable (changing the conditional).

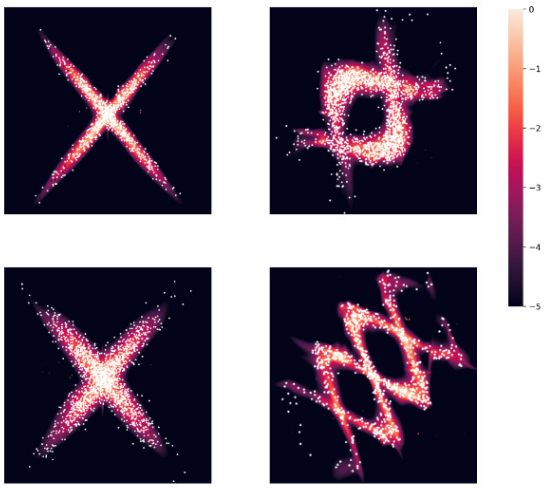


Fig. B.4. Joint density learned by DCDI-DSF. White dots are data points and the color represents the learned density. The x-axis is cause and the y-axis is the effect. First row is observational while second row is with an imperfect intervention on the effect.

For the X data set, both under perfect and imperfect interventions, the incapacity of DCDI-G to model this complex distribution properly makes it conclude (falsely) that there

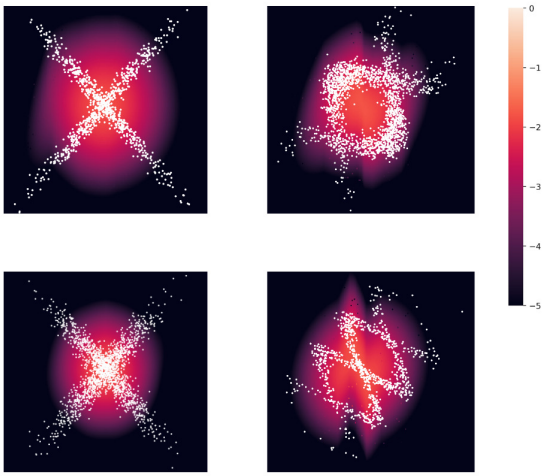


Fig. B.5. Joint density learned by DCDI-G. White dots are data points and the color represents the learned density. The x-axis is cause and the y-axis is the effect. First row is observational while second row is with an imperfect intervention on the effect.

is no dependency between the two variables (the μ outputted by DCDI-G is constant). Conversely, for the DNA data set with perfect interventions, it does infer the dependencies between the two variables and learn the correct causal direction, although the distribution is modeled poorly. Notice that, for the DNA data set with imperfect interventions, the lack of capacity of DCDI-G has pushed it to learn the same density with and without interventions (compare the two densities in the second column of Figure B.5; the learned density functions remain mostly unchanged from top to bottom). This prevented DCDI-G from learning the correct causal direction, while DCDI-DSF had no problem. We believe that if the imperfect interventions were more radical, DCDI-G could have recovered the correct direction even though it lacks capacity. In all cases, DCDI-DSF can easily model these functions and systematically infers the right causal direction.

While the proposed data sets are synthetic, similar multimodal distributions could be observed in real-world data sets due to latent variables that are parent of only one node (i.e., that are not confounders). A hidden variable that act as a selector between two different mechanisms could induce distributions similar to those in Figures B.4 and B.5. In fact, this idea was used to produce the synthetic data sets, i.e., a latent variable $z \in \{0, 1\}$ was sampled and, according to its value, example were generated following one of two mechanisms. The for the X dataset (second column in the article2/figures) was generated by two linear mechanisms in the following way:

$$y := \begin{cases} wx + N & z = 0 \\ -wx + N & z = 1, \end{cases}$$

where N is a Gaussian noise and w was randomly sampled from $[-1, -0.25] \cup [0.25, 1]$.

B.3.3. Scalability experiments

Figure B.6 presents two experiments which study the scalability of various methods in terms of number of examples (left) and number of variables (right). In these experiments, the runtime was restricted to 12 hours while the RAM memory was restricted to 16GB. All experiments considered perfect interventions. Experiments from Figure B.6 were run with fixed hyperparameters. **DCDI.** Same as Table B.2 except $\mu_0 = 10^{-2}$, # hidden units = 8 and $\lambda = 10^{-1}$. **CAM.** Pruning cutoff = 10^{-3}. Preliminary neighborhood selection was performed in the large graph experiments (otherwise CAM cannot run on 50 nodes in less than 12 hours). **GIES.** Regularizing parameter = 1. **IGSP.** The suffixes -G and -K refers to the partial correlation test and the KCI-test, respectively. The α parameter is set to 10^{-3}.
Number of examples. DCDI was the only algorithm supporting nonlinear relationships which could run on as much as 1 million examples without running out of time or memory. We believe different trade-offs between SHD and SID could be achieved with different hyperparameters, especially for GIES and CAM which achieved very good SID but poor SHD.
Number of variables. We see that using a GPU starts to pay off for graphs of 50 nodes or more. For 10-50 nodes data sets, DCDI-GPU outperforms the other methods in terms of SHD and SID, while maintaining a runtime similar to CAM. For the hundred-node data sets, the runtime of DCDI increases significantly with a SHD/SID performance comparable to the much faster GIES. We believe the weaker performance of DCDI in the hundred-node setting is due to the fact that the conditionals are high dimensional functions which are prone to overfitting. Also, we believe this runtime could be significantly reduced by limiting the number of parents via preliminary neighborhood selection similar to CAM ((Bühlmann et al., 2014)). This would have the effect of reducing the cost of computing the gradient of w.r.t. to the neural networks parameters. These adaptions to higher dimensionality are left as future work.

B.3.4. Ablation study

In this section, we show that the proposed losses are relevant by doing an ablation study. We also show that interventions are beneficial to recover the DAG compared to the use of observational data alone. First, in a small scale experiment, we show in Figure B.7 the effect of the number of interventions on the performance of DCDI-G. The SHD and SID of DCDI-G and DCD are shown over ten linear data sets (20-node graph with sparse connectivity) with $\{0, 5, 10, 15, 20\}$ perfect interventions. The baseline DCD is equivalent to DCDI-G, but it uses a loss that doesn't take into account the interventions. It can first be noticed that, as the number of interventions increase, the performance of DCDI-G increases. This increase is particularly noticeable from the purely interventional data to data with 5 interventions. While DCD's performance also increases in term of SHD, it seems to have no clear gain in

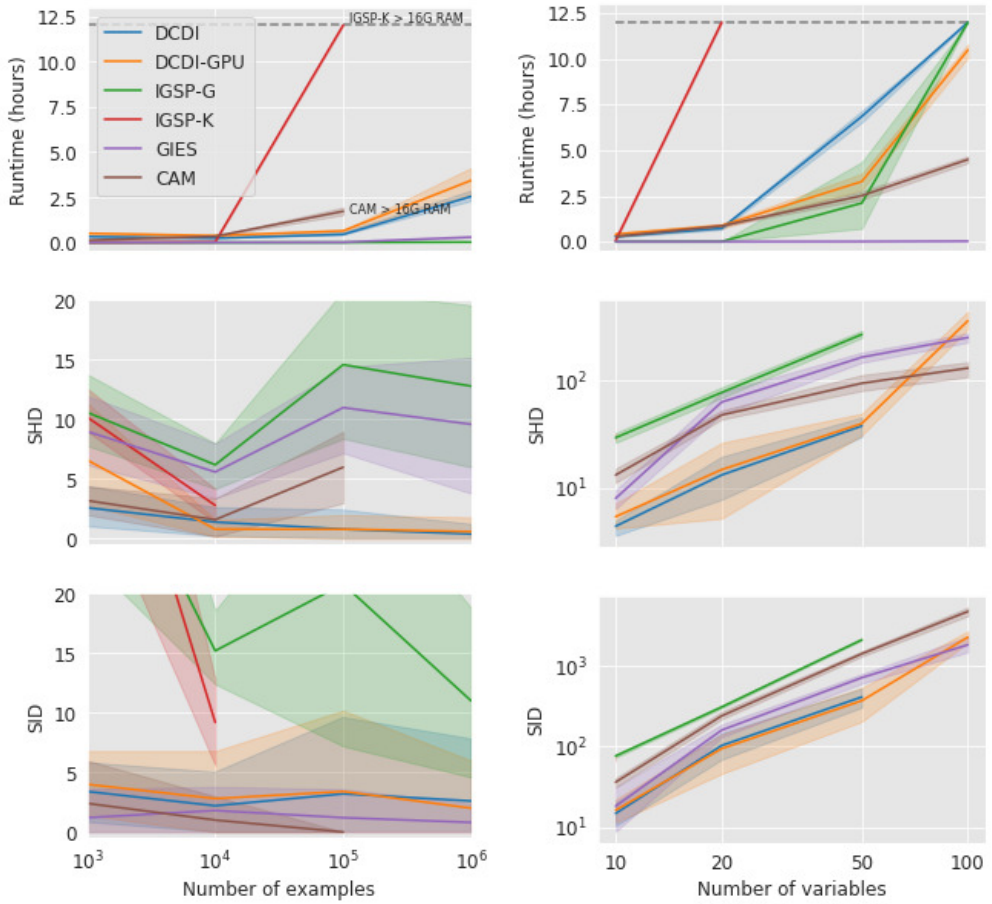


Fig. B.6. We report the runtime (in hours), SHD and SID of multiple methods in multiple settings. The horizontal dashed lines at 12 hours represents the time limit imposed. When a curve reaches this dashed line, it means that the method could not finish within 12 hours. We write $\geq$ 16G when the RAM memory needed by the algorithm exceeded 16GB. All data sets have 10 interventional targets containing $0.1d$ targets. We considered perfect interventions. **Left:** Different data set sizes. Ten nodes ANM data with connectivity $e = 1$. **Right:** Different number of variables. NN data set with connectivity $e = 4$ and 10^4 samples. Each curve is an average over 5 different datasets while the error bars are %95 confidence intervals computed via bootstrap.

term of SID. Also, DCDI-G with interventional data is always better than DCD showing that the proposed loss for perfect interventions is pertinent. Note that the first two boxes are the same since DCDI-G on observational data is equivalent to DCD (the experiment was done only once).

In a larger scale experiment, with the same data sets used in the main text (Section 4.4), we compare DCDI-G and DCDI-DSF to DCD and DCD-no-interv for perfect/known, imperfect/known and perfect/unknown interventions (shown respectively in Appendix B.3.4.1, B.3.4.2, and B.3.4.3). The values reported are the mean and the standard deviation of SHD and SID over ten data sets of each condition. DCD-no-interv is DCDI-G applied to purely

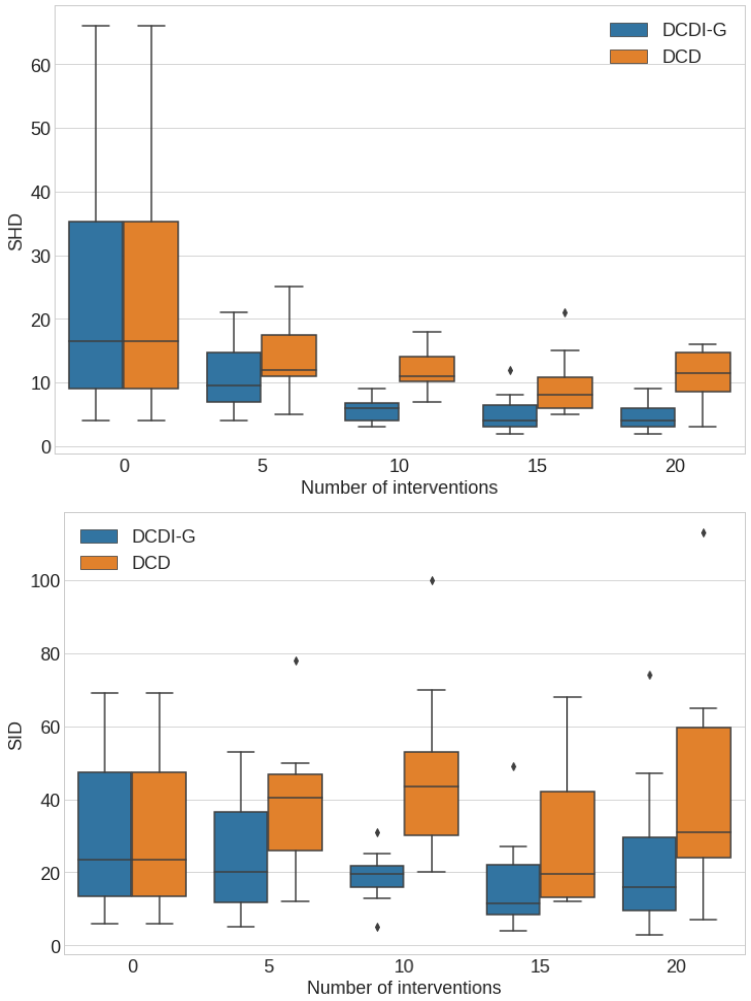


Fig. B.7. SHD and SID for DCDI-G and DCD on data sets with a different number of interventional settings.

observational data. These purely observational data sets were generated from the same CGM as the other data set containing interventions and had the same total sample size. For SHD, the advantage of DCDI over DCD and DCD-no-interv is clear over all conditions. For SID, DCDI has no advantage for sparse graphs, but is usually better for graphs with higher connectivity. As in the first small scale experiment, the beneficial effect of intervention is clear. Also, these results show that the proposed losses for the different type of interventions are pertinent.

Table B.4. Results for the linear data set with perfect intervention

	10 nodes, $e = 1$		10 nodes, $e = 4$		20 nodes, $e = 1$		20 nodes, $e = 4$	
Method	SHD	SID	SHD	SID	SHD	SID	SHD	SID
DCD	6.6 ± 3.6	14.1 ± 11.5	24.4 ± 6.0	67.0 ± 9.2	18.2 ± 15.8	30.9 ± 21.7	56.7 ± 10.2	227.0 ± 38.6
DCD-no-interv	8.9 ± 2.8	19.5 ± 10.9	26.7 ± 5.9	69.0 ± 11.2	24.6 ± 20.5	31.2 ± 22.8	64.4 ± 11.4	292.9 ± 28.9
DCDI-G	1.3 ± 1.9	0.8 ± 1.8	3.3 ± 2.1	10.7 ± 12.0	5.4 ± 4.5	13.4 ± 12.0	23.7 ± 5.6	112.8 ± 41.8
DCDI-DSF	0.9 ± 1.3	0.6 ± 1.9	3.7 ± 2.3	18.9 ± 14.1	3.6 ± 2.7	6.0 ± 5.4	16.6 ± 6.4	92.5 ± 40.1

Table B.5. Results for the additive noise model data set with perfect intervention

	10 nodes, $e = 1$		10 nodes, $e = 4$		20 nodes, $e = 1$		20 nodes, $e = 4$	
Method	SHD	SID	SHD	SID	SHD	SID	SHD	SID
DCD	11.5 ± 6.6	18.2 ± 11.8	30.4 ± 3.8	75.5 ± 4.6	39.3 ± 28.4	39.8 ± 33.3	62.7 ± 14.2	241.0 ± 44.8
DCD-no-interv	11.6 ± 8.8	15.8 ± 12.1	21.3 ± 5.2	63.5 ± 12.3	41.7 ± 44.1	36.2 ± 27.1	43.7 ± 9.2	226.1 ± 42.8
DCDI-G	5.2 ± 7.5	2.4 ± 4.9	4.3 ± 2.4	16.0 ± 11.9	21.8 ± 30.1	11.6 ± 13.1	35.2 ± 13.2	109.8 ± 44.6
DCDI-DSF	4.2 ± 5.6	5.6 ± 5.5	5.5 ± 2.4	23.9 ± 14.3	4.3 ± 1.9	19.7 ± 12.6	26.7 ± 16.9	105.3 ± 22.7

Table B.6. Results for the nonlinear with non-additive noise data set with perfect intervention

	10 nodes, $e = 1$		10 nodes, $e = 4$		20 nodes, $e = 1$		20 nodes, $e = 4$	
Method	SHD	SID	SHD	SID	SHD	SID	SHD	SID
DCD	5.9 ± 6.9	10.9 ± 10.4	15.7 ± 4.9	53.0 ± 9.9	28.7 ± 13.0	29.7 ± 9.3	29.3 ± 8.9	163.1 ± 48.4
DCD-no-interv	11.0 ± 9.3	9.9 ± 11.0	18.4 ± 6.4	56.4 ± 11.0	16.5 ± 22.8	31.9 ± 17.5	31.6 ± 11.3	160.3 ± 46.3
DCDI-G	2.3 ± 3.6	2.7 ± 3.3	2.4 ± 1.6	13.9 ± 8.5	13.9 ± 20.3	13.7 ± 8.1	16.8 ± 8.7	82.5 ± 38.1
DCDI-DSF	7.0 ± 10.7	7.8 ± 5.8	1.6 ± 1.6	7.7 ± 13.8	8.3 ± 4.1	32.4 ± 17.3	11.8 ± 2.1	102.3 ± 34.5

B.3.4.1. Perfect interventions.

Table B.7. Results for the linear data set with imperfect intervention

	10 nodes, $e = 1$		10 nodes, $e = 4$		20 nodes, $e = 1$		20 nodes, $e = 4$	
Method	SHD	SID	SHD	SID	SHD	SID	SHD	SID
DCD	10.6 ± 5.4	24.6 ± 18.2	24.0 ± 4.1	67.2 ± 7.6	21.2 ± 11.5	56.0 ± 31.5	56.7 ± 9.0	268.0 ± 25.4
DCD-no-interv	6.8 ± 4.4	19.5 ± 13.2	27.4 ± 4.4	74.0 ± 7.2	19.8 ± 9.2	48.2 ± 30.6	58.2 ± 9.9	288.6 ± 31.6
DCDI-G	2.7 ± 2.8	8.2 ± 8.8	5.2 ± 3.5	25.1 ± 12.9	15.6 ± 14.5	29.1 ± 23.4	34.0 ± 7.7	180.9 ± 44.5
DCDI-DSF	1.3 ± 1.3	4.2 ± 4.0	1.7 ± 2.4	10.2 ± 14.9	6.9 ± 6.3	22.7 ± 21.9	21.7 ± 8.1	137.4 ± 34.3

Table B.8. Results for the additive noise model data set with imperfect intervention

	10 nodes, $e = 1$		10 nodes, $e = 4$		20 nodes, $e = 1$		20 nodes, $e = 4$	
Method	SHD	SID	SHD	SID	SHD	SID	SHD	SID
DCD	12.0 ± 9.2	14.8 ± 10.4	24.3 ± 3.8	64.5 ± 11.1	51.7 ± 41.7	44.5 ± 20.0	54.1 ± 12.0	196.6 ± 37.2
DCD-no-interv	14.6 ± 4.3	12.1 ± 11.8	24.8 ± 4.8	69.3 ± 8.3	49.5 ± 36.0	32.7 ± 22.7	41.2 ± 8.1	197.7 ± 50.1
DCDI-G	6.2 ± 5.4	7.6 ± 11.0	13.1 ± 2.9	48.1 ± 9.1	30.5 ± 33.0	12.5 ± 8.8	43.1 ± 10.2	96.6 ± 47.1
DCDI-DSF	13.4 ± 8.4	17.9 ± 10.5	14.4 ± 2.4	53.2 ± 8.2	13.1 ± 4.5	43.5 ± 19.2	50.5 ± 11.4	172.1 ± 19.6

Table B.9. Results for the nonlinear with non-additive noise data set with imperfect intervention

	10 nodes, $e = 1$		10 nodes, $e = 4$		20 nodes, $e = 1$		20 nodes, $e = 4$	
Method	SHD	SID	SHD	SID	SHD	SID	SHD	SID
DCD	12.7 ± 8.4	11.8 ± 7.3	15.2 ± 3.7	52.2 ± 9.1	40.4 ± 54.7	45.2 ± 43.9	30.5 ± 8.0	151.2 ± 41.7
DCD-no-interv	13.6 ± 9.7	13.0 ± 8.1	14.8 ± 3.5	51.7 ± 12.5	37.1 ± 40.7	57.1 ± 56.2	31.3 ± 5.5	162.3 ± 40.5
DCDI-G	3.9 ± 3.9	7.5 ± 6.5	7.3 ± 2.2	28.0 ± 10.5	18.2 ± 28.8	36.9 ± 37.0	21.7 ± 8.0	127.3 ± 40.1
DCDI-DSF	5.3 ± 4.2	16.3 ± 10.0	5.9 ± 3.2	35.1 ± 12.3	13.2 ± 5.1	76.5 ± 57.8	16.8 ± 5.3	143.6 ± 48.8

B.3.4.2. Imperfect interventions.

Table B.10. Results for the linear data set with perfect intervention with unknown targets

	10 nodes, $e = 1$		10 nodes, $e = 4$		20 nodes, $e = 1$		20 nodes, $e = 4$	
Method	SHD	SID	SHD	SID	SHD	SID	SHD	SID
DCD	6.6 ± 3.6	14.1 ± 11.5	24.4 ± 6.0	67.0 ± 9.2	18.2 ± 15.8	30.9 ± 21.7	56.7 ± 10.2	227.0 ± 38.6
DCD-no-interv	8.9 ± 2.8	19.5 ± 10.9	26.7 ± 5.9	69.0 ± 11.2	24.6 ± 20.5	31.2 ± 22.8	64.4 ± 11.4	292.9 ± 28.9
DCDI-G	5.3 ± 3.7	12.9 ± 11.5	5.2 ± 3.0	24.3 ± 15.3	15.4 ± 10.3	30.8 ± 18.6	39.2 ± 8.7	173.7 ± 45.6
DCDI-DSF	3.9 ± 4.3	7.1 ± 7.1	7.1 ± 3.6	35.8 ± 12.5	4.3 ± 2.4	18.4 ± 7.3	29.7 ± 12.6	147.8 ± 42.7

Table B.11. Results for the additive noise model data set with perfect intervention with unknown targets

	10 nodes, $e = 1$		10 nodes, $e = 4$		20 nodes, $e = 1$		20 nodes, $e = 4$	
Method	SHD	SID	SHD	SID	SHD	SID	SHD	SID
DCD	11.5 ± 6.6	18.2 ± 11.8	30.4 ± 3.8	75.5 ± 4.6	39.3 ± 28.4	39.8 ± 33.3	62.7 ± 14.2	241.0 ± 44.8
DCD-no-interv	11.6 ± 8.8	15.8 ± 12.1	21.3 ± 5.2	63.5 ± 12.3	41.7 ± 44.1	36.2 ± 27.1	43.7 ± 9.2	226.1 ± 42.8
DCDI-G	7.6 ± 10.3	5.0 ± 5.4	9.1 ± 3.8	37.5 ± 14.1	41.3 ± 39.2	22.9 ± 15.5	39.9 ± 18.8	153.7 ± 50.3
DCDI-DSF	11.9 ± 8.8	13.8 ± 7.9	6.6 ± 2.6	32.6 ± 14.1	22.3 ± 31.9	33.1 ± 17.5	42.5 ± 18.7	152.9 ± 53.4

Table B.12. Results for the nonlinear with non-additive noise data set with perfect intervention with unknown targets

	10 nodes, $e = 1$		10 nodes, $e = 4$		20 nodes, $e = 1$		20 nodes, $e = 4$	
Method	SHD	SID	SHD	SID	SHD	SID	SHD	SID
DCD	5.9 ± 6.9	10.9 ± 10.4	15.7 ± 4.9	53.0 ± 9.9	28.7 ± 13.0	29.7 ± 9.3	29.3 ± 8.9	163.1 ± 48.4
DCD-no-interv	11.0 ± 9.3	9.9 ± 11.0	18.4 ± 6.4	56.4 ± 11.0	16.5 ± 22.8	31.9 ± 17.5	31.6 ± 11.3	160.3 ± 46.3
DCDI-G	3.4 ± 4.2	6.9 ± 7.5	3.3 ± 1.3	20.4 ± 10.4	21.8 ± 32.1	20.9 ± 12.3	20.1 ± 8.1	104.6 ± 47.1
DCDI-DSF	7.8 ± 7.9	11.8 ± 5.7	3.3 ± 1.2	23.2 ± 9.1	27.4 ± 30.9	49.3 ± 15.7	22.2 ± 10.4	131.0 ± 41.0

B.3.4.3. Unknown interventions.

B.3.5. Different kinds of interventions

In this section, we compare DCDI to IGSP using data sets under different kinds of interventions. We report results in tabular form for 10-node and 20-node graphs. For the perfect interventions, instead of replacing the target conditional distribution by the marginal $\mathcal{N}(2,1)$ (as in the main results), we used a marginal that doesn't involve a mean-shift: $\mathcal{U}[-1,1]$. The results reported in Tables B.13, B.14, B.15 of Section B.3.5.1 are the mean and the standard deviation of SHD and SID over ten data sets of each condition. From these results, we can conclude that DCDI-G still outperforms IGSP and, by comparing to DCD (DCDI-G with a loss that doesn't take into account interventions), that the proposed loss is still beneficial for this kind of interventions. It has competitive results compared to GIES and CAM on the linear data set and it outperforms them on the other data sets.

For imperfect intervention, we tried more modest changes in the parameters. For the linear data set, an imperfect intervention consisted of adding $\mathcal{U}[0.5, 1]$ to w_j if $w_j > 0$ and subtracting if $w_j <= 0$. It was done this way to ensure that the intervention would not remove dependencies between variables. For the additive noise model and the nonlinear with non-additive noise data sets, $\mathcal{N}(0, 0.1)$ was added to each weight of the neural networks. Results are reported in Tables B.16, B.17, B.18 of Section B.3.5.2. These smaller changes made the difference between DCD and DCDI imperceptible. For sparse graphs, IGSP has a better or comparable performance to DCDI. For graphs with higher connectivity, DCDI often has a better performance than IGSP.

Table B.13. Results for the linear data set with perfect intervention

	10 nodes, $e = 1$		10 nodes, $e = 4$		20 nodes, $e = 1$		20 nodes, $e = 4$	
Method	SHD	SID	SHD	SID	SHD	SID	SHD	SID
IGSP	4.0 ± 4.8	15.7 ± 15.4	28.8 ± 2.0	72.2 ± 5.1	9.7 ± 8.7	45.1 ± 45.4	68.1 ± 13.6	295.4 ± 27.6
GIES	0.3 ± 0.5	0.0 ± 0.0	4.0 ± 6.5	6.7 ± 17.7	1.5 ± 1.2	0.3 ± 0.9	49.4 ± 22.2	111.9 ± 51.4
CAM	0.6 ± 1.0	0.0 ± 0.0	11.8 ± 4.3	32.2 ± 17.2	6.3 ± 7.4	7.6 ± 9.8	91.4 ± 21.3	181.7 ± 60.5
DCD	6.3 ± 3.4	14.8 ± 10.6	26.1 ± 3.3	66.4 ± 11.4	11.1 ± 4.7	45.8 ± 22.8	49.0 ± 12.0	258.6 ± 41.6
DCDI-G	0.4 ± 0.7	1.3 ± 2.1	7.5 ± 1.4	29.7 ± 8.2	3.2 ± 3.2	12.1 ± 11.2	21.0 ± 4.9	147.6 ± 49.5

Table B.14. Results for the additive noise model data set with perfect intervention

	10 nodes, $e = 1$		10 nodes, $e = 4$		20 nodes, $e = 1$		20 nodes, $e = 4$	
Method	SHD	SID	SHD	SID	SHD	SID	SHD	SID
IGSP	5.7 ± 2.3	23.4 ± 13.6	32.8 ± 2.4	79.3 ± 3.2	14.9 ± 8.1	78.8 ± 64.6	80.5 ± 6.4	337.6 ± 27.3
GIES	7.5 ± 5.1	2.3 ± 2.5	9.2 ± 2.9	27.1 ± 11.5	23.8 ± 18.4	3.1 ± 4.4	89.6 ± 14.7	143.9 ± 53.1
CAM	6.3 ± 6.9	0.0 ± 0.0	6.3 ± 3.8	14.6 ± 20.1	9.2 ± 14.3	13.5 ± 25.1	106.2 ± 14.6	96.2 ± 57.9
DCD	6.4 ± 4.6	22.0 ± 14.7	31.1 ± 3.4	77.4 ± 3.1	18.1 ± 8.0	51.5 ± 41.5	55.7 ± 8.3	261.3 ± 22.5
DCDI-G	0.9 ± 1.2	3.9 ± 6.4	5.2 ± 1.9	24.0 ± 9.3	6.5 ± 5.6	17.9 ± 19.1	26.8 ± 7.0	94.4 ± 41.5

Table B.15. Results for the nonlinear with non-additive noise data set with perfect intervention

	10 nodes, $e = 1$		10 nodes, $e = 4$		20 nodes, $e = 1$		20 nodes, $e = 4$	
Method	SHD	SID	SHD	SID	SHD	SID	SHD	SID
IGSP	6.6 ± 3.9	25.8 ± 17.9	31.1 ± 3.3	77.1 ± 5.7	14.4 ± 4.8	63.8 ± 26.5	79.7 ± 8.1	341.4 ± 18.1
GIES	6.2 ± 3.5	0.9 ± 1.5	9.5 ± 3.6	29.0 ± 17.7	12.2 ± 2.1	3.4 ± 3.2	63.8 ± 11.1	124.9 ± 36.9
CAM	4.1 ± 3.8	2.3 ± 3.4	11.3 ± 4.2	35.4 ± 20.8	4.2 ± 2.3	10.9 ± 10.3	106.6 ± 15.7	144.2 ± 51.8
DCD	6.6 ± 3.5	18.1 ± 8.1	20.6 ± 3.9	65.8 ± 9.9	9.4 ± 4.9	25.6 ± 16.2	28.6 ± 6.8	188.0 ± 28.7
DCDI-G	2.1 ± 1.5	4.6 ± 5.4	5.0 ± 4.3	28.8 ± 17.6	6.4 ± 3.8	15.1 ± 8.0	12.2 ± 2.7	96.1 ± 18.9

B.3.5.1. Perfect interventions.

Table B.16. Results for the linear data set with imperfect intervention

	10 nodes, $e = 1$		10 nodes, $e = 4$		20 nodes, $e = 1$		20 nodes, $e = 4$	
Method	SHD	SID	SHD	SID	SHD	SID	SHD	SID
IGSP	1.1 ± 1.1	5.4 ± 5.4	28.7 ± 3.2	72.4 ± 6.7	4.2 ± 3.9	17.7 ± 12.3	86.1 ± 12.3	289.8 ± 26.3
DCD	3.8 ± 3.6	9.4 ± 6.4	27.7 ± 3.4	74.6 ± 3.5	27.2 ± 22.3	39.3 ± 20.5	65.0 ± 8.0	306.8 ± 26.3
DCDI-G	4.7 ± 4.5	11.5 ± 9.5	27.4 ± 4.9	73.8 ± 5.4	29.6 ± 16.5	37.7 ± 14.5	62.8 ± 6.5	303.2 ± 27.6
DCDI-DSF	4.1 ± 2.3	10.3 ± 7.5	24.3 ± 5.3	69.1 ± 8.7	12.2 ± 2.9	42.6 ± 18.3	56.1 ± 9.2	291.4 ± 35.7

Table B.17. Results for the additive noise model data set with imperfect intervention

	10 nodes, $e = 1$		10 nodes, $e = 4$		20 nodes, $e = 1$		20 nodes, $e = 4$	
Method	SHD	SID	SHD	SID	SHD	SID	SHD	SID
IGSP	5.7 ± 4.0	17.4 ± 13.4	30.3 ± 4.0	73.9 ± 11.3	12.5 ± 6.6	44.9 ± 26.7	85.8 ± 4.4	344.0 ± 9.8
DCD	12.0 ± 10.3	11.3 ± 8.4	23.5 ± 2.1	69.7 ± 2.5	39.5 ± 42.3	28.2 ± 13.9	50.9 ± 7.1	247.8 ± 36.6
DCDI-G	12.7 ± 9.1	11.8 ± 6.5	21.7 ± 4.3	65.2 ± 9.2	16.2 ± 18.0	27.8 ± 13.1	46.2 ± 5.9	240.1 ± 26.3
DCDI-DSF	8.1 ± 8.2	15.8 ± 9.3	23.3 ± 6.3	68.7 ± 8.2	12.3 ± 4.1	39.9 ± 19.5	51.0 ± 7.1	257.7 ± 31.6

Table B.18. Results for the nonlinear with non-additive noise data set with imperfect intervention

	10 nodes, $e = 1$		10 nodes, $e = 4$		20 nodes, $e = 1$		20 nodes, $e = 4$	
Method	SHD	SID	SHD	SID	SHD	SID	SHD	SID
IGSP	7.0 ± 5.7	22.7 ± 19.5	29.4 ± 5.0	74.2 ± 7.3	18.7 ± 7.1	86.3 ± 37.1	81.6 ± 6.9	344.4 ± 20.5
DCD	9.4 ± 8.9	13.3 ± 11.0	15.1 ± 3.7	54.2 ± 9.8	28.5 ± 25.0	25.5 ± 16.8	32.7 ± 9.8	177.1 ± 37.5
DCDI-G	6.7 ± 5.1	13.0 ± 9.7	14.6 ± 3.3	53.9 ± 9.1	28.9 ± 33.7	25.2 ± 15.2	32.3 ± 7.9	177.0 ± 55.8
DCDI-DSF	12.8 ± 9.6	22.9 ± 14.8	14.4 ± 4.8	54.2 ± 10.3	13.3 ± 5.3	54.2 ± 20.9	28.6 ± 8.9	199.5 ± 32.7

B.3.5.2. Imperfect interventions.

B.3.6. Comprehensive results of the main experiments

In this section, we report the main results presented in Section 4.4 in tabular form for 10-node and 20-node graphs. We also include additional results for different cutoff values for IGSP and UT-IGSP, namely $\log_{10} \alpha = \{-2, -3, -5, -7, -9\}$. This range was chosen to be around the cutoff value of 10^{-5} used in Squires et al. ((2020)). The reported values in the following tables are the mean and the standard deviation of SHD and SID over ten data sets of each condition. As stated in the main discussion, our conclusions are similar for 10-node graphs: DCDI has competitive performance in all conditions and outperforms the other methods for graphs with higher connectivity.

Table B.19. Results for linear data set with perfect intervention

	10 nodes, $e=1$		10 nodes, $e=4$		20 nodes, $e=1$		20 nodes, $e=4$	
Method	SHD	SID	SHD	SID	SHD	SID	SHD	SID
IGSP(α=1e-2)	2.4 ± 2.0	9.9 ± 9.5	27.4 ± 4.2	71.1 ± 6.1	6.1 ± 5.1	18.2 ± 11.8	74.7 ± 15.9	272.4 ± 33.7
IGSP(α=1e-3)	2.4 ± 2.2	10.3 ± 10.4	29.4 ± 3.7	73.8 ± 8.1	8.3 ± 6.7	32.1 ± 35.4	79.6 ± 12.4	298.1 ± 16.7
IGSP(α=1e-5)	2.4 ± 2.2	10.9 ± 10.5	31.6 ± 3.8	75.7 ± 5.8	9.4 ± 5.1	39.3 ± 33.6	84.3 ± 10.5	324.7 ± 12.3
IGSP(α=1e-7)	2.6 ± 2.5	13.4 ± 14.6	33.2 ± 3.1	79.6 ± 3.4	9.0 ± 5.5	46.0 ± 39.1	81.7 ± 7.6	336.5 ± 16.6
IGSP(α=1e-9)	2.5 ± 2.3	12.6 ± 12.9	29.6 ± 2.9	73.1 ± 6.7	12.2 ± 5.1	64.9 ± 44.1	82.8 ± 6.4	341.5 ± 14.2
GIES	0.7 ± 1.1	0.0 ± 0.0	4.7 ± 3.7	6.5 ± 13.9	1.1 ± 1.2	0.0 ± 0.0	52.8 ± 18.9	79.8 ± 68.3
CAM	1.9 ± 2.6	1.7 ± 3.1	10.6 ± 3.1	34.5 ± 11.0	5.4 ± 7.9	8.2 ± 9.6	91.1 ± 21.7	167.8 ± 55.4
DCDI-G	1.3 ± 1.9	0.8 ± 1.8	3.3 ± 2.1	10.7 ± 12.0	5.4 ± 4.5	13.4 ± 12.0	23.7 ± 5.6	112.8 ± 41.8
DCDI-DSF	0.9 ± 1.3	0.6 ± 1.9	3.7 ± 2.3	18.9 ± 14.1	3.6 ± 2.7	6.0 ± 5.4	16.6 ± 6.4	92.5 ± 40.1

Table B.20. Results for the additive noise model data set with perfect intervention

	10 nodes, $e=1$		10 nodes, $e=4$		20 nodes, $e=1$		20 nodes, $e=4$	
Method	SHD	SID	SHD	SID	SHD	SID	SHD	SID
IGSP(α=1e-2)	7.7 ± 3.5	21.6 ± 11.3	31.3 ± 2.5	79.7 ± 3.0	18.4 ± 6.5	77.6 ± 41.6	83.9 ± 9.6	323.0 ± 13.6
IGSP(α=1e-3)	9.0 ± 3.9	27.9 ± 15.2	30.1 ± 3.9	77.6 ± 4.9	16.9 ± 8.8	79.5 ± 54.2	83.8 ± 8.9	335.0 ± 21.7
IGSP(α=1e-5)	8.0 ± 3.8	30.1 ± 14.9	31.4 ± 4.2	79.3 ± 4.2	16.6 ± 6.5	80.0 ± 50.8	80.6 ± 8.3	325.5 ± 24.3
IGSP(α=1e-7)	8.3 ± 4.3	33.1 ± 15.0	33.2 ± 2.5	78.4 ± 6.3	16.3 ± 6.9	81.3 ± 48.9	81.3 ± 6.1	330.0 ± 25.5
IGSP(α=1e-9)	9.5 ± 5.2	33.5 ± 13.0	31.3 ± 5.1	72.7 ± 11.9	15.5 ± 6.7	78.8 ± 51.4	80.6 ± 10.2	335.4 ± 17.1
GIES	7.6 ± 4.6	1.9 ± 2.9	10.5 ± 2.5	24.6 ± 13.8	24.2 ± 12.4	7.5 ± 13.8	94.4 ± 10.9	144.4 ± 62.0
CAM	5.2 ± 3.0	1.0 ± 1.9	8.5 ± 3.7	11.5 ± 13.4	7.5 ± 6.0	5.6 ± 4.9	105.7 ± 13.2	108.7 ± 61.0
DCDI-G	5.2 ± 7.5	2.4 ± 4.9	4.3 ± 2.4	16.0 ± 11.9	21.8 ± 30.1	11.6 ± 13.1	35.2 ± 13.2	109.8 ± 44.6
DCDI-DSF	4.2 ± 5.6	5.6 ± 5.5	5.5 ± 2.4	23.9 ± 14.3	4.3 ± 1.9	19.7 ± 12.6	26.7 ± 16.9	105.3 ± 22.7

Table B.21. Results for the nonlinear with non-additive noise data set with perfect intervention

	10 nodes, $e=1$		10 nodes, $e=4$		20 nodes, $e=1$		20 nodes, $e=4$	
Method	SHD	SID	SHD	SID	SHD	SID	SHD	SID
IGSP(α=1e-2)	5.4 ± 3.1	13.7 ± 6.4	29.8 ± 4.0	73.0 ± 8.4	19.6 ± 3.9	80.5 ± 22.7	81.8 ± 8.7	336.2 ± 18.0
IGSP(α=1e-3)	5.7 ± 2.9	19.4 ± 13.0	30.4 ± 3.3	73.5 ± 10.8	17.9 ± 5.6	85.6 ± 37.1	80.6 ± 11.9	330.7 ± 23.5
IGSP(α=1e-5)	6.1 ± 3.0	19.5 ± 12.1	33.3 ± 3.7	78.6 ± 5.2	19.6 ± 4.5	94.4 ± 30.2	77.0 ± 9.5	345.2 ± 9.8
IGSP(α=1e-7)	6.4 ± 3.0	22.6 ± 13.1	33.8 ± 3.4	77.4 ± 10.1	18.5 ± 4.0	85.9 ± 29.1	76.1 ± 11.3	347.5 ± 15.9
IGSP(α=1e-9)	6.7 ± 3.7	24.8 ± 15.9	34.9 ± 2.7	78.8 ± 9.1	19.5 ± 4.6	100.6 ± 33.5	77.9 ± 9.2	341.9 ± 24.6
GIES	4.0 ± 2.4	0.8 ± 1.3	8.3 ± 3.1	26.5 ± 12.2	13.9 ± 5.7	9.4 ± 9.4	65.9 ± 16.5	110.6 ± 48.9
CAM	1.8 ± 1.5	2.8 ± 4.4	7.9 ± 3.6	26.7 ± 19.0	6.1 ± 5.2	18.1 ± 16.3	101.8 ± 24.5	142.5 ± 49.1
DCDI-G	2.3 ± 3.6	2.7 ± 3.3	2.4 ± 1.6	13.9 ± 8.5	13.9 ± 20.3	13.7 ± 8.1	16.8 ± 8.7	82.5 ± 38.1
DCDI-DSF	7.0 ± 10.7	7.8 ± 5.8	1.6 ± 1.6	7.7 ± 13.8	8.3 ± 4.1	32.4 ± 17.3	11.8 ± 2.1	102.3 ± 34.5

B.3.6.1. Perfect interventions.

Table B.22. Results for the linear data set with imperfect intervention

	10 nodes, $e=1$		10 nodes, $e=4$		20 nodes, $e=1$		20 nodes, $e=4$	
Method	SHD	SID	SHD	SID	SHD	SID	SHD	SID
IGSP(α=1e-2)	$2.1_{\pm 1.0}$	$10.8_{\pm 8.1}$	$27.2_{\pm 5.5}$	$71.7_{\pm 6.8}$	$6.6_{\pm 4.0}$	$20.8_{\pm 14.2}$	$64.0_{\pm 9.3}$	$271.0_{\pm 24.7}$
IGSP(α=1e-3)	$3.4_{\pm 2.8}$	$15.9_{\pm 9.5}$	$28.3_{\pm 6.1}$	$74.6_{\pm 7.2}$	$6.1_{\pm 4.6}$	$30.1_{\pm 22.8}$	$74.1_{\pm 14.4}$	$298.5_{\pm 21.0}$
IGSP(α=1e-5)	$3.4_{\pm 2.4}$	$17.0_{\pm 13.1}$	$30.3_{\pm 3.2}$	$76.1_{\pm 5.9}$	$7.9_{\pm 5.2}$	$44.5_{\pm 37.8}$	$77.8_{\pm 11.2}$	$324.2_{\pm 17.8}$
IGSP(α=1e-7)	$3.8_{\pm 1.5}$	$18.4_{\pm 8.8}$	$29.4_{\pm 3.4}$	$76.5_{\pm 5.1}$	$10.9_{\pm 5.5}$	$56.7_{\pm 30.5}$	$78.0_{\pm 8.6}$	$333.1_{\pm 17.1}$
IGSP(α=1e-9)	$5.1_{\pm 3.2}$	$24.5_{\pm 18.8}$	$31.8_{\pm 3.1}$	$79.7_{\pm 4.9}$	$11.2_{\pm 5.6}$	$62.4_{\pm 34.1}$	$75.2_{\pm 9.2}$	$341.6_{\pm 24.2}$
GIES	$5.8_{\pm 5.4}$	$20.3_{\pm 19.2}$	$10.8_{\pm 6.6}$	$38.7_{\pm 23.0}$	$4.9_{\pm 3.8}$	$22.2_{\pm 23.9}$	$81.5_{\pm 18.8}$	$200.4_{\pm 50.3}$
CAM	$8.1_{\pm 6.2}$	$22.6_{\pm 18.8}$	$19.4_{\pm 4.7}$	$56.0_{\pm 10.1}$	$10.5_{\pm 5.8}$	$36.3_{\pm 23.6}$	$111.7_{\pm 16.5}$	$232.5_{\pm 23.4}$
DCDI-G	$2.7_{\pm 2.8}$	$8.2_{\pm 8.8}$	$5.2_{\pm 3.5}$	$25.1_{\pm 12.9}$	$15.6_{\pm 14.5}$	$29.1_{\pm 23.4}$	$34.0_{\pm 7.7}$	$180.9_{\pm 44.5}$
DCDI-DSF	$1.3_{\pm 1.3}$	$4.2_{\pm 4.0}$	$1.7_{\pm 2.4}$	$10.2_{\pm 14.9}$	$6.9_{\pm 6.3}$	$22.7_{\pm 21.9}$	$21.7_{\pm 8.1}$	$137.4_{\pm 34.3}$

Table B.23. Results for the additive noise model data set with imperfect intervention

	10 nodes, $e=1$		10 nodes, $e=4$		20 nodes, $e=1$		20 nodes, $e=4$	
Method	SHD	SID	SHD	SID	SHD	SID	SHD	SID
IGSP(α=1e-2)	$9.2_{\pm 5.2}$	$22.4_{\pm 11.3}$	$34.0_{\pm 4.0}$	$77.4_{\pm 4.5}$	$20.9_{\pm 6.8}$	$83.8_{\pm 38.6}$	$84.7_{\pm 6.4}$	$328.2_{\pm 16.2}$
IGSP(α=1e-3)	$9.0_{\pm 4.8}$	$27.5_{\pm 12.4}$	$33.0_{\pm 2.5}$	$80.5_{\pm 2.4}$	$19.7_{\pm 4.7}$	$85.4_{\pm 34.8}$	$82.6_{\pm 5.4}$	$330.9_{\pm 19.0}$
IGSP(α=1e-5)	$7.9_{\pm 3.5}$	$26.9_{\pm 15.1}$	$34.8_{\pm 3.0}$	$81.2_{\pm 3.7}$	$20.6_{\pm 4.6}$	$92.8_{\pm 46.7}$	$84.7_{\pm 8.2}$	$338.4_{\pm 16.1}$
IGSP(α=1e-7)	$7.8_{\pm 3.5}$	$25.0_{\pm 13.6}$	$34.2_{\pm 2.3}$	$82.0_{\pm 1.9}$	$19.2_{\pm 4.7}$	$82.8_{\pm 30.1}$	$84.8_{\pm 6.7}$	$340.2_{\pm 25.8}$
IGSP(α=1e-9)	$7.9_{\pm 3.7}$	$24.7_{\pm 13.9}$	$34.5_{\pm 3.1}$	$81.2_{\pm 3.7}$	$19.4_{\pm 2.9}$	$91.2_{\pm 27.1}$	$80.3_{\pm 6.4}$	$335.7_{\pm 23.5}$
GIES	$17.6_{\pm 6.7}$	$24.6_{\pm 12.9}$	$18.1_{\pm 5.1}$	$58.0_{\pm 9.0}$	$35.3_{\pm 17.1}$	$55.4_{\pm 41.3}$	$121.2_{\pm 12.4}$	$236.5_{\pm 29.1}$
CAM	$11.2_{\pm 9.3}$	$7.8_{\pm 8.7}$	$9.6_{\pm 3.0}$	$25.2_{\pm 10.8}$	$16.3_{\pm 9.9}$	$26.7_{\pm 27.2}$	$121.9_{\pm 11.6}$	$155.4_{\pm 41.5}$
DCDI-G	$6.2_{\pm 5.4}$	$7.6_{\pm 11.0}$	$13.1_{\pm 2.9}$	$48.1_{\pm 9.1}$	$30.5_{\pm 33.0}$	$12.5_{\pm 8.8}$	$43.1_{\pm 10.2}$	$96.6_{\pm 47.1}$
DCDI-DSF	$13.4_{\pm 8.4}$	$17.9_{\pm 10.5}$	$14.4_{\pm 2.4}$	$53.2_{\pm 8.2}$	$13.1_{\pm 4.5}$	$43.5_{\pm 19.2}$	$50.5_{\pm 11.4}$	$172.1_{\pm 19.6}$

Table B.24. Results for the nonlinear with non-additive noise data set with imperfect intervention

	10 nodes, $e=1$		10 nodes, $e=4$		20 nodes, $e=1$		20 nodes, $e=4$	
Method	SHD	SID	SHD	SID	SHD	SID	SHD	SID
IGSP(α=1e-2)	$7.0_{\pm 4.5}$	$24.3_{\pm 19.0}$	$30.0_{\pm 4.1}$	$76.1_{\pm 5.5}$	$21.9_{\pm 8.9}$	$103.2_{\pm 77.7}$	$79.0_{\pm 8.6}$	$335.5_{\pm 20.5}$
IGSP(α=1e-3)	$6.4_{\pm 5.0}$	$27.0_{\pm 20.9}$	$30.1_{\pm 4.8}$	$74.4_{\pm 7.0}$	$20.9_{\pm 7.0}$	$115.9_{\pm 78.6}$	$76.5_{\pm 7.8}$	$344.1_{\pm 21.8}$
IGSP(α=1e-5)	$5.8_{\pm 5.0}$	$22.6_{\pm 20.6}$	$30.6_{\pm 3.2}$	$77.1_{\pm 3.7}$	$20.1_{\pm 8.5}$	$125.9_{\pm 95.9}$	$74.1_{\pm 5.5}$	$335.8_{\pm 24.3}$
IGSP(α=1e-7)	$6.6_{\pm 4.9}$	$22.4_{\pm 17.6}$	$32.7_{\pm 3.8}$	$78.1_{\pm 5.3}$	$20.7_{\pm 8.7}$	$124.5_{\pm 82.3}$	$76.3_{\pm 5.7}$	$340.8_{\pm 25.8}$
IGSP(α=1e-9)	$6.4_{\pm 4.2}$	$24.5_{\pm 19.1}$	$34.1_{\pm 2.4}$	$77.2_{\pm 6.6}$	$20.9_{\pm 8.6}$	$131.1_{\pm 93.0}$	$76.0_{\pm 5.1}$	$348.3_{\pm 18.4}$
GIES	$9.4_{\pm 5.8}$	$16.7_{\pm 11.9}$	$18.3_{\pm 6.4}$	$52.1_{\pm 15.5}$	$32.8_{\pm 14.1}$	$58.5_{\pm 45.5}$	$81.5_{\pm 13.9}$	$217.3_{\pm 33.9}$
CAM	$4.3_{\pm 3.3}$	$9.3_{\pm 6.8}$	$14.7_{\pm 5.1}$	$45.7_{\pm 14.9}$	$20.7_{\pm 16.2}$	$53.9_{\pm 32.9}$	$121.5_{\pm 9.3}$	$194.1_{\pm 40.3}$
DCDI-G	$3.9_{\pm 3.9}$	$7.5_{\pm 6.5}$	$7.3_{\pm 2.2}$	$28.0_{\pm 10.5}$	$18.2_{\pm 28.8}$	$36.9_{\pm 37.0}$	$21.7_{\pm 8.0}$	$127.3_{\pm 40.1}$
DCDI-DSF	$5.3_{\pm 4.2}$	$16.3_{\pm 10.0}$	$5.9_{\pm 3.2}$	$35.1_{\pm 12.3}$	$13.2_{\pm 5.1}$	$76.5_{\pm 57.8}$	$16.8_{\pm 5.3}$	$143.6_{\pm 48.8}$

B.3.6.2. Imperfect interventions.

Table B.25. Results for the linear data set with perfect intervention with unknown targets

	10 nodes, $e = 1$		10 nodes, $e = 4$		20 nodes, $e = 1$		20 nodes, $e = 4$	
Method	SHD	SID	SHD	SID	SHD	SID	SHD	SID
UTIGSP(α=1e-2)	1.7 ± 2.1	7.0 ± 9.3	27.2 ± 5.8	70.1 ± 9.8	4.7 ± 3.9	14.4 ± 10.8	69.8 ± 12.4	271.8 ± 20.8
UTIGSP(α=1e-3)	1.6 ± 2.2	7.2 ± 10.1	29.6 ± 5.5	73.1 ± 9.4	6.9 ± 6.7	25.2 ± 32.0	81.3 ± 12.4	300.7 ± 17.5
UTIGSP(α=1e-5)	1.2 ± 1.9	5.1 ± 8.7	29.4 ± 4.2	73.2 ± 7.1	8.6 ± 6.0	36.4 ± 29.9	81.5 ± 11.7	323.9 ± 14.1
UTIGSP(α=1e-7)	1.8 ± 2.6	7.6 ± 13.4	29.4 ± 3.4	72.3 ± 9.6	8.6 ± 5.6	42.5 ± 40.2	84.8 ± 9.7	339.7 ± 11.7
UTIGSP(α=1e-9)	1.8 ± 2.4	7.8 ± 13.5	29.2 ± 3.8	70.2 ± 7.5	11.6 ± 7.3	56.3 ± 48.6	81.0 ± 5.4	336.0 ± 13.6
DCDI-G	5.3 ± 3.7	12.9 ± 11.5	5.2 ± 3.0	24.3 ± 15.3	15.4 ± 10.3	30.8 ± 18.6	39.2 ± 8.7	173.7 ± 45.6
DCDI-DSF	3.9 ± 4.3	7.1 ± 7.1	7.1 ± 3.6	35.8 ± 12.5	4.3 ± 2.4	18.4 ± 7.3	29.7 ± 12.6	147.8 ± 42.7

Table B.26. Results for the additive noise model data set with perfect intervention with unknown targets

	10 nodes, $e = 1$		10 nodes, $e = 4$		20 nodes, $e = 1$		20 nodes, $e = 4$	
Method	SHD	SID	SHD	SID	SHD	SID	SHD	SID
UTIGSP(α=1e-2)	9.1 ± 4.2	25.3 ± 10.3	29.0 ± 2.6	73.1 ± 3.1	19.2 ± 7.8	77.4 ± 50.9	84.9 ± 10.1	332.3 ± 14.2
UTIGSP(α=1e-3)	10.4 ± 4.1	28.1 ± 12.9	30.5 ± 4.7	77.8 ± 5.4	18.6 ± 8.5	81.4 ± 53.6	83.8 ± 5.5	331.9 ± 27.3
UTIGSP(α=1e-5)	9.9 ± 4.3	33.6 ± 12.0	32.1 ± 3.9	77.4 ± 6.7	18.7 ± 4.9	86.4 ± 41.8	83.5 ± 6.8	341.7 ± 12.4
UTIGSP(α=1e-7)	9.4 ± 4.9	33.3 ± 14.4	33.7 ± 3.9	76.8 ± 9.4	18.3 ± 6.8	84.3 ± 47.0	83.3 ± 8.1	336.8 ± 21.3
UTIGSP(α=1e-9)	9.4 ± 5.2	32.1 ± 15.2	33.0 ± 4.2	77.7 ± 8.7	18.8 ± 7.0	97.1 ± 55.9	82.9 ± 7.0	329.4 ± 28.2
DCDI-G	7.6 ± 10.3	5.0 ± 5.4	9.1 ± 3.8	37.5 ± 14.1	41.3 ± 39.2	22.9 ± 15.5	39.9 ± 18.8	153.7 ± 50.3
DCDI-DSF	11.9 ± 8.8	13.8 ± 7.9	6.6 ± 2.6	32.6 ± 14.1	22.3 ± 31.9	33.1 ± 17.5	42.5 ± 18.7	152.9 ± 53.4

Table B.27. Results for the nonlinear with non-additive noise data set with perfect intervention with unknown targets

	10 nodes, $e = 1$		10 nodes, $e = 4$		20 nodes, $e = 1$		20 nodes, $e = 4$	
Method	SHD	SID	SHD	SID	SHD	SID	SHD	SID
UTIGSP(α=1e-2)	6.1 ± 4.1	16.6 ± 12.5	28.1 ± 4.8	68.4 ± 14.3	22.1 ± 5.5	100.6 ± 30.1	85.4 ± 9.1	330.0 ± 19.6
UTIGSP(α=1e-3)	6.4 ± 3.6	19.5 ± 14.5	31.0 ± 3.1	76.8 ± 4.3	20.0 ± 4.3	92.2 ± 21.6	81.1 ± 6.2	338.5 ± 10.8
UTIGSP(α=1e-5)	6.8 ± 3.5	21.1 ± 12.9	35.0 ± 2.2	80.6 ± 4.8	20.2 ± 6.1	94.4 ± 33.3	80.2 ± 9.3	339.4 ± 15.2
UTIGSP(α=1e-7)	6.2 ± 3.5	20.0 ± 11.5	32.5 ± 2.1	75.2 ± 9.9	21.2 ± 4.5	103.0 ± 22.1	78.9 ± 9.2	348.7 ± 12.6
UTIGSP(α=1e-9)	7.6 ± 3.8	22.3 ± 13.4	33.9 ± 2.0	78.6 ± 6.9	19.5 ± 4.1	94.9 ± 31.9	77.2 ± 7.4	341.8 ± 19.3
DCDI-G	3.4 ± 4.2	6.9 ± 7.5	3.3 ± 1.3	20.4 ± 10.4	21.8 ± 32.1	20.9 ± 12.3	20.1 ± 8.1	104.6 ± 47.1
DCDI-DSF	7.8 ± 7.9	11.8 ± 5.7	3.3 ± 1.2	23.2 ± 9.1	27.4 ± 30.9	49.3 ± 15.7	22.2 ± 10.4	131.0 ± 41.0

B.3.6.3. Unknown interventions.