Causal Structure Learning

Amandine Sandri, Yvan Sraka

February 4, 2019

Introduction

Determining causal structure among variables based on observational data is of great interest in many areas of science. Causal models are much more informative than probability models. A joint distribution already tells us how probable events are and how probabilities would change with subsequent observations, however causal models also tell us how these probabilities would change as a result of external interventions. For instance, an intervention might be a policy decision or action performed in everyday activity. Therefore, the effect of any intervention may be predicted by a causal network and this ability for causal network to predict such them requires assumptions to build the network.

In this presentation, we will see how to learn causal structure from randomly generated data, taking into account the effect of intervention or distribution changes and identifying the additional assumptions needed for the purpose of causal learning.

I) Notation and basic concepts

1) Structural Equation Model

The model is a Structural Causal Model (SCM) may be referred to as structural equation model (1). In SCM, all dependences are represented by functions that compute variables from other variables, or in other words, that model various types of mechanisms.

$$Xj \leftarrow \sum_{k=1}^{p}(\beta_{j,k}.X_k + \epsilon_j) \tag{1}$$

for $j = 1, ..., p$, or in vector notation:

$$X \leftarrow BX + \epsilon \tag{2}$$

where X represents a p-dimensional random variable, B a $p \times p$ matrix with entries $B_{j,k} = \beta_{j,k}$ and noise contributions $\epsilon = (\epsilon_1, ..., \epsilon_p)^t$

It is possible to use several different Structural Causal Models to generate one single distribution. Those models are assumed to be invariant between each other and this refers to the autonomy assumption where one structural equation does not vary following any change in other structural equation.

In this project, the models will be represented under Directed Acyclic Graph (DAG) which represents the most popular approach to make the causal inference problem more malleable by describing conditional dependence information and causal structure. The basic decomposition scheme displayed by a DAG is as described in equation (2). When we assume having a distribution P defined on n discrete variables $x_1, x_2, ..., x_n$, therefore, the chain rule of probability calculus allows us decompose P as a product of n conditional distributions.

$$P(x_i, ..., x_n) = \prod_j P(xj|x_0, ..., x_{j-1}) \tag{3}$$

Then, assuming that xj is not dependent of all its predecessor and knowing the value of a group of predecessors PAj, then, the equation (3) may be hugely simplified by :

$$P(x_i, ..., x_n) = \prod_j P(x_j|pa_j) \tag{4}$$

which considers only the possible realization of the set PA_j.

2) Conditional dependencies

This previous factorization (4) of the joint probability distribution shows conditional independence relationships between variables that can also be shown by DAG causal network. For example, the figure (3) shows an example of a Bayesian Network which is actually a DAG.

Conditional independencies are implied by the Markov properties. There exists a relation between two variables if they are linked by a path. If there is only a single edge between the variables, they cannot be conditionally independent. Every node is independent from its non-descendants given its parents[3]. When looking at conditional independencies locally by applying local Markov property on several different nodes, it generates a list of conditional independencies which may be difficult to be retrieve exhaustively. To overcome this, the concept of d-separation criterion has been introduced as a graphical concept to read off those independencies in a DAG.

d-separation criterion

Two nodes X and Y are d-separated by a set S if on every path between X and Y at least (i) one non-collider is in S or (ii) one collider is not in S nor has a descendant in S[2].

3) Interventions

For any given DAG representing a system behavior, we may be interested in its changes of behavior under intervention. Causal effects under intervention can be estimated by using intervention calculus. Two kind of intervention may be taken into account: the *do*-intervention and the additive intervention. The first one is represented under the below modified structural vector annotation:

$$X_j \leftarrow Z_j \tag{5}$$

where Z_j is the value taken by variable X_j following the intervention.

The second intervention is defined by below structural equation:

$$X_j \leftarrow \sum_{k=1}^{p} (\beta_{j,k}.X_k + \epsilon_j + Z_j) \tag{6}$$

where Z_j is the additional noise added to variable X_j.

Those interventions motivate the fact that after an intervention, some parts of the data generating process changes and hence, the data sets simulated differ between each other according to the selected intervention parameters.

4) Graphical representation

As we saw, different DAG may have the same graphical d-separation. Therefore, both graphical CPDAG and PDAG represent Markov equivalence classes of DAG or MAG, respectively, and can be inferred directly from observational data only (no need to take interventional data into account). Note that d stands for "Directed", so one calls this criterion as m-separation for MAG.

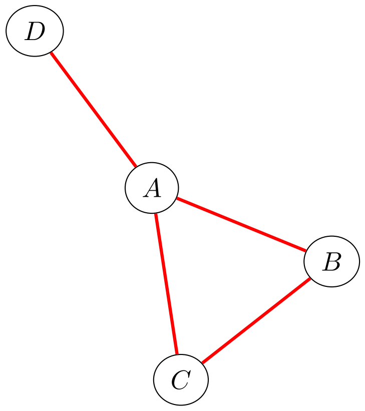

Figure 1: e.g. of Graph (undirected)

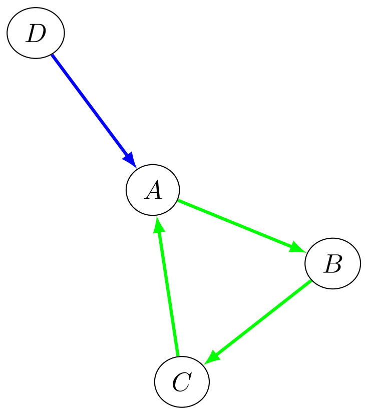

Figure 2: e.g. of **Directed** Graph (DG) which contain a cycle (in green)

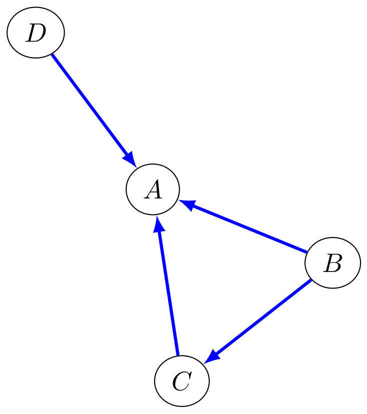

Figure 3: e.g. of **Directed Acyclic** Graph (DAG)

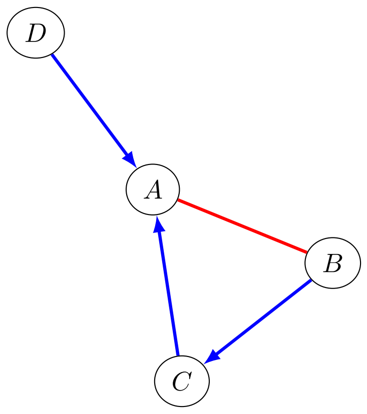

Figure 4: e.g. of **Partially Directed Acyclic** Graph (PDAG)

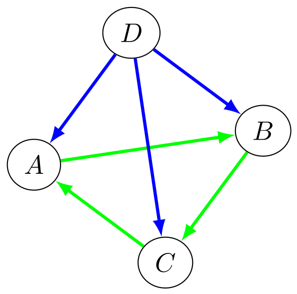

Figure 5: e.g. of **Completed Directed** Graph (this is not a DAG!)

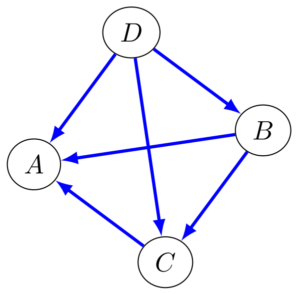

Figure 6: e.g. of **Completed Directed Acyclic** Graph

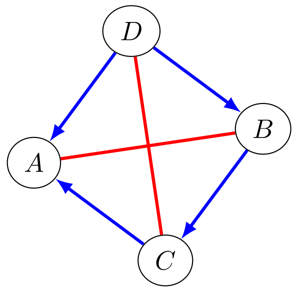

Figure 7: e.g. of **Completed Partially Directed Acyclic** Graph (CPDAG)

5) Latent and hidden variables

This introduction will be closed with a last concept of latent and hidden variables which are important to take into account in this context of model building. A latent variable can neither be observed nor directly measured and it is rather inferred from directly observed and measure data. When latent variable corresponds to physical reality but are difficult to measure in practice, then we call them hidden variable. Those variables can me modelled by two different ways, either identifiable as nodes in structural equations or as an underlying dependence between the noise terms. In the case we assume there is no latent variables, we assume the noise terms to be independent.

II) Material and methods

In this section are presented the various packages and algorithms that have been tested during this project.

1) Packages and functions used

Our study of assessing causal graphs estimation has been led through the use of a R Package named `CompareCausalNetwork` as suggested in Christina Heinze-Deml and al[1]. This package provides a unified interface to various causal graph estimation methods and implementations of constraint-based methods and score-based methods that are the algorithms we will use to build causal graphs assuming certain hypothesis. In next section, those algorithmic methods and their possible assumptions are presented.

2) Algorithms implemented and their assumptions

a) Assumptions

Many algorithm inferring causality rely on several assumptions:

Causal sufficiency

A set of variables is said to be causally sufficient if there is no hidden common cause that is causing more than one variable in X[5].

Faithfulness

We have seen that using d-separation criterion allowed us to check whether a causal structure implies conditional independencies. When assuming that Markov property holds, all conditional independencies having been found thanks to d-separation criterion also apply to every structural equation model with this given causal structure. However, some specific structural equation models might imply more conditional independence statements. This is why, we overcome this limit by assuming that the conditional independences statements found after applying *d*-separation criterion are exactly the same as the ones described through the structural equation model. This assumption is known as faithfulness.

Acyclicity

This notion assumes there is no cycle within the graph that the algorithm will learn from the data.

Gaussianity of the noise distribution

We assume the noise distribution follows a normal law.

b) Causal discovery algorithms

Constraint-based methods: PC algorithms

This algorithm is the most famous in constraint-based approach for learning causal structure from data. Its implementation consists in three main steps, that first aim to learn the skeleton graph, which is composed entirely of undirected edges, then it searches for separation sets and finally, it orients the undirected edges to generate a new class of graph but still equivalent to DAG.

During the first step, the algorithm starts from the fully connected network and removes or retains edges according to the results of conditional independences test.

It assumes sufficiency, faithfulness and acyclicity.

Score-based methods: Greedy Equivalence Search (GES) algorithms

This algorithm consists in optimizing some score like the likelihood or the BIC to look for the best candidate network that would fit the best the datasets.

It has two steps named the forward step and the backward step. The first one consists in completing an empty graph with one single edge at a time under the condition that the edge improves the score. The backward phase uses the output of the forward step as an input and consists in removing the single edge that allow an improvement of the score, and this is done until the score can no longer be improved.

3) Datasets generation

Several steps are needed to generate simulated data on which learn the causal network.

There is first the generation of an adjacency matrix B in a manner that it does not contain zero value at all. Thus, the entries of B are sampled iteratively until we obtain, at least, one non-zero entry. Then, we sample the corresponding coefficients of the entries.

The second step consists in simulating the intervention. For this purpose, we use the function simulate intervention from the package comparecausalnetwork. An indicator matrix Ie, k is used as an information whether an intervention takes place on a specific setting (in which case, the entry equals to 1) or not (in which case, the entry equals to 0).

Then, one generates the noise terms thanks to a covariance matrix which allow to appreciate how much the presence and the strength of hidden variables may influence the structural equation. We also decide whether or not to form a cycle between variables.

Finally, after having generate the data, we introduce non-linearity i the distribution of the observation by transforming all variables X_j by $X_j \leftarrow log(X_j)$.

4) Methods of evaluation

To evaluate the performance of the method put in place we assess the relationships established with the parents and the ancestors. For this, we use the three interrogations below that allow us to assess:

- Whether it exists or not a relation thanks to the queries `isParent`, `isAncestor` ;
- Whether there is absence of a relation thanks to the queries `isNoParent`, `isNoAncestor` ;
- Whether there is a potential existence of a relation thanks to the queries `isPossibleParent` and `isPossibleAncestor`.

III) Results

Implementation details: our R notebook containing the different experiments performed could be found on our GitHub repository:

https://yvan-sraka.github.io/causal-structure-learning/

We think it is a good idea to compare different methods of reconstruction. Most make assumptions about the nature of the relationships between variables, the type of network or the presence of latent variables, disturbances, etc.

It may be interesting to add those options to our tool to test the robustness of the methods to these assumptions.

a) Size of the data

We have generated data with different parameters such as:

- the number of variables describing the data, aka the number of edges (function of graph topoly choosed) ;
- the number of observations, aka the number of graph nodes (in figure 8 we fixed it to 4 for readability purpose).

We can observe that the execution time of all methods increase (as expected) with the number of edges and nodes of the graph.

b) Nonlinear relations

If we consider that the distribution of the observations for a node X is generated with a function of its parents: $d(x) = f(P_a(X)) + \epsilon$, we can choose f among functions not linear (of the type $log(Y)$, $tanh(Y)$ etc ...) or not monotonic, but informative (like Y^2 for Y in $[-1, 1]$). When there are several parents we can use (non) linear combinations of Y or use polynomials for example.

We do not have enought time to hack with modified distributions of generated data.

c) Network topologies

Regarding the type of networks you can generate random graphs as in the article "Causal Structure Learning", use another unstructured random method (see `random.graph` of the `bnlearn` package: https://cran.r-project.org/web/packages/bnlearn/bnlearn.pdf) or try to generate scale-free networks with more or less of hubs. Again, the interest is to see if there are significant differences in the performance of each method for different tasks.

We have choosen to generate different topologies of networks by generating random graphs with several models:

- The Erdös-Rényi model (random network)
- The Watts-Strogatz model (small-world network)
- The Albert-Barabsi model (scale-free network)

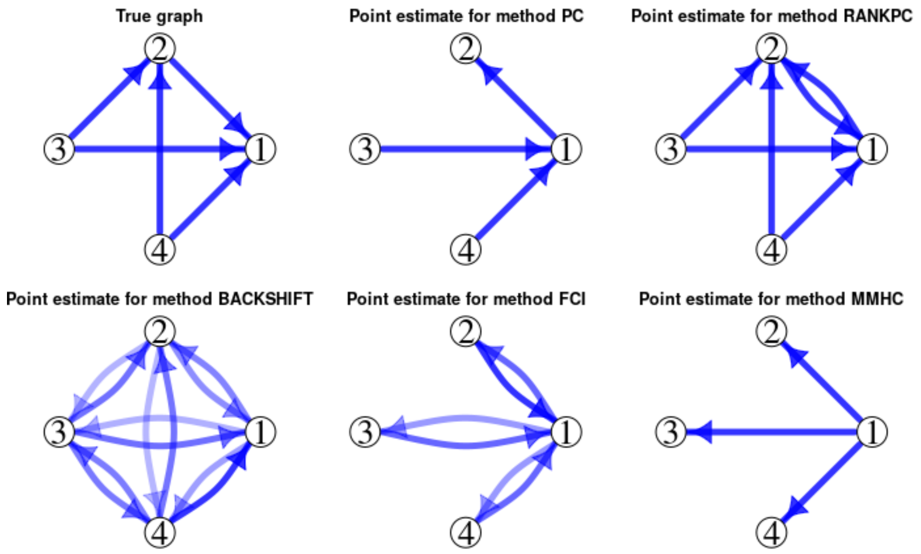
True graph
Point estimate for method PC
Point estimate for method RANKPC
Point estimate for method BACKSHIFT
Point estimate for method FCI
Point estimate for method MMHC
1
2
3
4

Figure 8: Causal structures learned by 5 different simulated methods on a randomly generated 4-nodes scale-free network

IV) Discussion

Perspective: Hybrid methods

Apart from both methods compared during this project which are the constraint-based and score-based algorithms, we could implement another class of algorithms named hybrid methods. This method allows to learn CPDAG by combining constraint-based and score-based methods. They consist first in estimating a supergraph of the skeleton using conditional independences tests and then they use search and score method only taking into account the set of allowed edges to the estimated skeleton. A typical example of this technique is the Max-Min Hill-Climbing (MMHC) algorithm[6]. In the empirical study, authors showed MMHC outperform PC and GES with structural hamming distance (SHD) as a measure[4].

Bibliography

[1] Christina Heinze-Deml, Marloes H. Maathuis, and Nicolai Meinshausen. Causal Structure Learning. *arXiv:1706.09141 [stat]*, June 2017. arXiv: 1706.09141.

[2] Markus Kalisch and Peter Bühlmann. Causal Structure Learning and Inference: A Selective Review. *Quality Technology & Quantitative Management*, 11(1):3–21, January 2014.

[3] Steffen L. Lauritzen. *Graphical models.* Number 17 in Oxford statistical science series. Clarendon Press, Oxford, reprinted 2004 with corrections edition, 2004. OCLC: 253867796.

[4] Karamjit Singh, Garima Gupta, Vartika Tewari, and Gautam Shroff. Comparative Benchmarking of Causal Discovery Techniques. *arXiv:1708.06246 [cs, stat]*, August 2017. arXiv: 1708.06246.

[5] Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, Prediction, and Search*, volume 81 of *Lecture Notes in Statistics*. Springer New York, New York, NY, 1993.

[6] Ioannis Tsamardinos, Laura E. Brown, and Constantin F. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 65(1):31–78, October 2006.