Causal Inference (MGT-416)
Final Project

- This project can be solved in groups of 1 up to 3 students. You must mention the name of all the participants. Note that all the students in a group will get the same grade.
- All of the programming implementations must be done in either MATLAB, or Python.
- Deadline: Tuesday, JUNE 14, 2022, 09:00 AM (No late submissions will be accepted.)
- Upload a single .zip file on Moodle, including the implementations, necessary output files, and the report as a pdf file.

1 Learning the causal structure of a Bayesian network

Constraint-based methods are one of the main approaches for learning the causal structure of a system [5]. These methods rely on statistical tests, namely *conditional independence* (CI) tests, to recover the structure. Let $\mathbf{V}$ denote the set of variables of a causal system, with joint distribution $P_{\mathbf{V}}$. Let $\mathcal{G}$ be the (unknown) causal DAG that describes the causal relationships among the variables of the system. Each vertex of $\mathcal{G}$ represents a variable in $\mathbf{V}$. We therefore will use the words variable and vertex interchangeably. Let $\mathbf{X}, \mathbf{Y}, \mathbf{Z} \subseteq \mathbf{V}$ be three disjoint sets of variables. Under the assumption that Markov property and faithfulness hold, d-separation relations in $\mathcal{G}$ are equivalent to conditional independence relations in the corresponding data. More precisely, for any three disjoint subsets of variables $\mathbf{X}, \mathbf{Y}, \mathbf{Z} \subseteq \mathbf{V}$,

$$(\mathbf{X} \perp \mathbf{Y} | \mathbf{Z}) \text{ in } \mathcal{G} \quad \iff \quad (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} | \mathbf{Z}) \text{ in } P_{\mathbf{V}}, \tag{1}$$

where $\perp$ and $\perp\!\!\!\perp$ denote d-separation and statistical independence relations, respectively.

The conditional independence relations (right-hand side of Eq. 1) can be statistically estimated using the samples of the dataset (you will find further details in Section 1.3.) As a result, the d-separation relations in $\mathcal{G}$ can be learned from data. This, in turn, can be used to recover the structure $\mathcal{G}$ up to *Markov equivalence class* (MEC). Markov equivalence class of a DAG $\mathcal{G}$ is the set of all DAGs that share the same d-separation relations with $\mathcal{G}$. It was shown that the causal structure could be learned only up to MEC using mere observational data [2, 4].

In this project, you will implement two constraint-based causal discovery algorithms, namely SGS (named after its authors, Spirtes, Glymour, and Scheines) and PC (named after the same authors, Peter and Clark) [4]. These algorithms begin with discovering the skeleton of $\mathcal{G}$, i.e., the undirected graph resulting from removing the edge orientations in $\mathcal{G}$. They then proceed to orient the edges by detecting the v-structures (aka immoralities) and using Meek rules [1]. We begin by describing the skeleton discovery process in these algorithms.

1.1 Recovering the skeleton

SGS. Begin with an empty graph over $\mathbf{V}$. Take two arbitrary variables $X, Y \in \mathbf{V}$. For any subset $\mathbf{Z} \subset \mathbf{V} \setminus \{X, Y\}$, check the statistical test $X \perp\!\!\!\perp Y|\mathbf{Z}$. If it does not hold for any set $\mathbf{Z}$, then draw an undirected edge between X and Y. Repeat this procedure for all X, Y to learn the whole skeleton.

PC (PC1). For a vertex X in $\mathcal{G}$, let $N(X)$ denote the set of its neighbors, i.e., the union of parents and children of X. It can be shown that for any pair of vertices $X, Y \in \mathbf{V}$, if $X \notin N(Y)$, then there exists a subset of vertices $\mathbf{Z} \in N(X) \cup N(Y)$ such that $X \perp Y|\mathbf{Z}$. We will use this fact to skip performing unnecessary CI tests.

1. Begin with a complete undirected graph over $\mathbf{V}$.
2. Let $d = 0$.
3. For any edge (X, Y) in the graph, check the result of CI test $(X \perp\!\!\!\perp Y|\mathbf{Z})$ for all subsets $\mathbf{Z} \subseteq \mathbf{V} \setminus \{X, Y\}$ of size $|\mathbf{Z}| = d$. If any of these CI relations hold, remove the edge between X and Y.
4. Increase d by one. If there exists a vertex with at least d neighbors, repeat from step (3).

Modified PC (PC2). For a variable $X \in \mathbf{V}$, the Markov boundary of X, denoted by $\mathrm{Mb}(X)$ is a minimal set of variables such that for any variable $Y \notin \mathrm{Mb}(X)$,

$$X \perp\!\!\!\perp Y|\mathrm{Mb}(X).$$

It can be shown that for any pair of variables X and Y,

$$Y \in \mathrm{Mb}(Y) \quad \iff \quad X \not\perp\!\!\!\perp Y|\mathbf{V} \setminus \{X, Y\}. \tag{2}$$

In order to further reduce the number of performed CI tests, we can modify PC algorithm to make use of the Markov boundary information. Note that from what we discussed above, if $Y \notin \mathrm{Mb}(X)$, then there is no edge between X and Y. In fact, under Markov property and faithfulness, $N(X) \subseteq \mathrm{Mb}(X)$. As a result, we can initialize PC algorithm with the moralized graph instead of a complete graph. Note that moralized graph is an undirected graph where each vertex is connected to the vertices in its Markov boundary with an edge. The steps of this algorithm are as follows.

1. Begin with an empty graph over $\mathbf{V}$.
2. For every pair of variables X, Y, draw an undirected edge between X and Y if $X \not\perp\!\!\!\perp Y|\mathbf{V} \setminus \{X, Y\}$.
3. Let $d = 0$.
4. For any edge (X, Y) in the graph, check the result of CI test $(X \perp\!\!\!\perp Y|\mathbf{Z})$ for all subsets $\mathbf{Z} \subseteq \mathbf{V} \setminus \{X, Y\}$ of size $|\mathbf{Z}| = d$. If any of these CI relations hold, remove the edge between X and Y.
5. Increase d by one. If there exists a vertex with at least d neighbors, repeat from step (4).

1.2 Orienting the edges

The edge orientation process is the same among all algorithms mentioned above. There are two steps for orienting the edges. First, we need to identify the v-structures. Any structure $X - Y - W$, that is, a tuple of vertices (X, Y, W) where there is no edge between X and W is a *potential* v-structure. Note that since X and Z have no edge between them, you have found some set $\mathbf{Z} \subseteq \mathbf{V} \setminus \{X, W\}$ in one of the steps of the algorithm such that $X \perp\!\!\!\perp W|\mathbf{Z}$. It can be shown that $X - Y - Z$ is a v-structure, i.e., is oriented as $X \to Y \leftarrow W$ if and only if $Y \notin \mathbf{Z}$. After orienting all the v-structures, We apply Meek rules [1] to orient the rest of the edges up to MEC (i.e., not all edges can necessarily be oriented.)

Meek suggested four rules to orient the edges of the recovered DAG [1]. These four rules are shown to be complete. That is, by applying these four rules, every edge that is invariant among the graphs in the MEC, gets oriented. In this project, you will implement two of these rules, namely the first two rules of Meek:

- If $A \to B - C$, that is the edge between B and C is not oriented, and there is no edge between A and C, orient $B - C$ as $B \to C$. This rule prevents adding new v-structures to the graph.
- If $A \to B \to C$ and $A - C$, that is, the edge between A and C is not oriented, orient it as $A \to C$. This rule is to maintain acyclicity.

1.3 CI tests

Performing statistical CI tests is a challenging task in general, due to both method-specific limitations and lack of data. There is ongoing research on how to perform CI tests for various applications. In this project, we assume that the generative model is linear with Gaussian noises. For this model, it can be shown that conditional independence tests reduce to evaluating the conditional partial correlations among variables. Partial correlation can be computed in several ways using data. One such way is to use correlation and precision matrices. For more information on this topic, refer to `https://en.wikipedia.org/wiki/Partial_correlation#Using_matrix_inversion` and `https://en.wikipedia.org/wiki/Partial_correlation#As_conditional_independence_test`. For this project, you are provided with an implementation of CI test (in both MATLAB and Python) as a function with input D (data matrix), X and Y (two variables), and $\mathbf{Z}$ (a subset of variables). This function outputs a boolean value where 1 and 0 indicate $X \perp\!\!\!\perp Y|\mathbf{Z}$ and $X \not\perp\!\!\!\perp Y|\mathbf{Z}$, respectively. Further details of this function is found in the description of the function in the corresponding file.

Tasks

1. Implement three functions (SGS, PC1, and PC2) for learning the skeleton of the causal graph. These functions have one input, namely a data matrix D, where each row is a sample and each column corresponds to one variable. They return two output: the adjacency matrix of the skeleton; and a set of separating sets for non-adjacent pairs of variables. $\mathbf{Z}$ is called a separating set for X, Y if $X \perp\!\!\!\perp Y|\mathbf{Z}$. As an example, if the learned skeleton is $A - B - C$ with no edges between A and C, the output must be the matrix

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix},$$

along with a separating set for the pair (A, C). Try to implement the algorithms as efficiently as possible.[1]

2. Implement a function that receives the output of the functions in part (1), i.e., an adjacency matrix of an undirected graph along with separating sets for non-adjacent pairs of variables, and orients the graph. Your function should start with discovering the v-structures and then apply the first two Meek rules. The output of this function is a the adjacency matrix of a partially oriented graph (i.e., with both directed and undirected edges).

3. You are given three datasets (D1, D2, D3, D4) corresponding to four different graphs. For instance, D1 is a 500×5 matrix containing 500 samples of 5 variables. Prepare a script to parse the input data from these files. Then use your functions of part (1) to learn the skeletons corresponding to each dataset. Since SGS and PC1 might take a long time to run on D4, you are only asked to report the output of PC2 on this particular dataset. Then use your function of part (2) to orient the edges maximally. Report the final graphs (you should report 10 graphs in total). You can include the graphs corresponding to $D1, D2, D3$ in your text report and upload the adjacency matrix corresponding to $D4$ as a separate file.

4. As a measure of computational complexity, analyze the maximum number of CI tests each algorithm above performs in the worst case. Report an upper bound in terms of the number of variables and the maximum degree of the graph. Compare the worst-case complexity of the three algorithms to each other. How do they compare in practice, i.e., in your simulations? Report the number of CI tests performed by each algorithm.

[1] For instance, if you find out that a CI test $X \perp\!\!\!\perp Y | \mathbf{Z}$ holds, this is to say that there is no edge between X and Y. Therefore, you need not perform any further CI tests for X and Y.

2 Analyzing the stock market data

The goal of this task is to learn the causal structure among 12 technology companies by observing their stock price over a period of time. The list of these companies appear in Table 1. The prices of these companies were sampled every 2 minutes (190 samples per day) for seven market days (03/03/2008 - 03/10/2008), i.e., T = 7.

The log-price of the technology companies are given in a form of a Matlab file. In this file, you will find a $7 \times 190 \times 12$ array M, where $M(i,j,k)$ denotes the j-th sample of the log-price of k-th company at day i.

We can assume that the underlying dynamic which describes the relationships between the log-prices of these companies is jointly Gaussian. This is due to the Black-Scholes model of the market. Therefore, if x and y denote two time-series corresponding to two different companies and $\underline{z}$ denotes the time series of the rest of the companies, directed information $I(x \to y \mid\mid \underline{z})$ can be estimated using the following equation.

$$I(x \to y \mid\mid \underline{z}) = \frac{1}{2}\sum_{t=1}^{T} \log \frac{|\Sigma_{y_1^t \underline{z}_1^{t-1}}||\Sigma_{x_1^{t-1} y_1^{t-1} \underline{z}_1^{t-1}}|}{|\Sigma_{y_1^{t-1} \underline{z}_1^{t-1}}||\Sigma_{x_1^{t-1} y_1^t \underline{z}_1^{t-1}}|},$$

where $|\Sigma_{y_1^t \underline{z}_1^{t-1}}|$ denotes the determinant of the Covariance matrix of $(y(1)), \cdots, y(t), \underline{z}(1), \cdots, \underline{z}(t-1)$. To decide whether your estimated DI is zero or positive use 0.5 as the threshold.

Name	Code
Apple Inc.	APPL(1)
Cisco Systems Inc.	CSCO(2)
Dell Inc. Inc.	DEL(3)
EMC Corporation	EMC(4)
Google Inc.	GOG(5)
Hewlett-Packard	HP(6)
Intel	INT(7)
Microsoft	MSFT(8)
Oracle	ORC(9)
International Business Machines	IBM(10)
Texas Instruments	TXN(11)
Xerox	XRX(12)

Table 1: List of Companies in the analysis

Learn the directed information graph among these companies and plot it. For your submission, you need to submit your codes for (i) computing DIs and (ii) learning DI graph. Also, report a 12×12 DI matrix and your learned DI graph.

We recommend the interested readers to refer to [3] for more information on directed information graphs.

References

[1] Christopher Meek. Causal inference and causal explanation with background knowledge. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pages 403–410, 1995.

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

[3] Christopher J Quinn, Negar Kiyavash, and Todd P Coleman. Directed information graphs. *IEEE Transactions on information theory*, 61(12):6887–6909, 2015.

[4] Peter Spirtes, Clark N Glymour, Richard Scheines, and David Heckerman. *Causation, prediction, and search*. MIT press, 2000.

[5] Kun Zhang, Bernhard Schölkopf, Peter Spirtes, and Clark Glymour. Learning causality and causality-related learning: Some recent progress. *National Science Review*, 5(1):26–29, 2017.