Problem Set 2

6.S091: Causality

IAP 2023

Due: Thursday, January 26th at 1pm EST

- Problem sets **must** be done in LaTeX.
- Printed problem sets must be turned in at the beginning of lecture. If you cannot attend, please find a classmate to turn the problem set in for you.
- You may use any programming language for your solutions, and you are not required to turn in your code.

Problem 1: Identifying direction by non-Gaussianity [2 points]

You observe data from either the SCM M^a:

$$X_1 = \varepsilon_1$$
$$X_2 = \beta_{12} X_1 + \varepsilon_2$$

where $\varepsilon_1 \perp\!\!\!\perp \varepsilon_2$, or the SCM M^b:

$$X_2 = \varepsilon_2$$
$$X_1 = \beta_{21} X_2 + \varepsilon_1$$

where $\varepsilon_1 \perp\!\!\!\perp \varepsilon_2$. Samples of (X_1, X_2) are in the file `nongaussian.csv` at

`https://github.com/csquires/6.S091-causality/tree/main/psets/pset2`,

where the first column contains samples of X_1 and the second column contains samples of X_2.

Plotting Regression Residuals [1 point]

(a) Let $\widehat{\beta}_{12}$ be the linear regression coefficient when regressing X_2 onto X_1 (without an intercept term). Report $\widehat{\beta}_{12}$ and plot $(X_1, X_2 - \widehat{\beta}_{12} X_1)$.

(b) Let $\widehat{\beta}_{21}$ be the linear regression coefficient when regressing X_1 onto X_2 (without an intercept term). Report $\widehat{\beta}_{21}$ and plot $(X_2, X_1 - \widehat{\beta}_{21} X_2)$.

Causal Direction Inference [1 point]

(c) Which SCM is the data more likely to generated from? Explain.

Problem 2: The PC Algorithm [8 points]

Samples of $(X_1, X_2, X_3, X_4, X_5, X_6, X_7)$ are in the file `pcalg_samples.csv` at

`https://github.com/csquires/6.S091-causality/tree/main/psets/pset2`,

Each row of this matrix corresponds to one sample, with the i-th column of the matrix corresponds to the variable X_i. Your goal in this problem is to learn the causal structure over these variables using the PC algorithm. In the first three subsections, you will develop a function which performs conditional independence testing from data You will then use this conditional independence test to implement the skeleton phase of the PC algorithm.

You are welcome to use packages for helper functions. For example, in Python, you might find the following packages helpful:

- `numpy` and `sklearn` for performing linear regression, computing correlations, and using special functions such as arctanh and Φ, the cumulative distribution function of the standard normal,
- `networkx` for manipulating graphs, e.g. the `graph.neighbors(node)` method to find the neighbors of a node, and
- `itertools` for list all subsets $\mathbf{S} \subseteq \mathcal{X}$ of size k.

Partial correlation [2 points]

The sample partial correlation $\widehat{\rho}(X_i, X_j \mid \mathbf{S})$ can be computed as follows. Let $\widehat{\beta}_{i,\mathbf{S}}$ be the vector of linear regression coefficients when regressing samples of X_i onto samples of $\mathbf{S}$, and let $\widehat{\beta}_{j,\mathbf{S}}$ be the vector of linear regression coefficients when regressing samples of X_j onto samples of $\mathbf{S}$. Here, both regressions should include an intercept term. Let $R_i = X_i - \widehat{\beta}_{i,\mathbf{S}}^\top \mathbf{S}$ and $R_j = X_j - \widehat{\beta}_{i,\mathbf{S}}^\top \mathbf{S}$. Then $\widehat{\rho}(X_i, X_j \mid \mathbf{S})$ is the sample correlation between R_i and R_j. When $\mathbf{S} = \varnothing$, the sample partial correlation is defined as the sample correlation between X_i and X_j, i.e., no regression is used.

Write a function `compute_partial_correlation(samples, i, j, S)` which computes the sample partial correlation between the values in the i-th column of `samples` and the values in the j-th column of `samples` with respect to the columns `S`. You can check the correctness of your function by checking that

$$\widehat{\rho}(X_1, X_7, \{X_3, X_4\}) = \texttt{compute\_partial\_correlation(pcalg\_samples, 1, 7, [3, 4])} \approx 0.01324,$$

and

$$\widehat{\rho}(X_1, X_7, \varnothing) = \texttt{compute\_partial\_correlation(pcalg\_samples, 1, 7, [])} \approx -0.2917.$$

Pay attention to indexing: in Python, `(1, 7, [3, 4])` will become `(0, 6, [2, 3])`.

(a) Report the value of

$$\widehat{\rho}(X_1, X_4, \varnothing) = \texttt{compute\_partial\_correlation(pcalg\_samples, 1, 4, [])}$$

(b) Report the value of

$$\widehat{\rho}(X_1, X_4, \{X_2, X_3\}) = \texttt{compute\_partial\_correlation(pcalg\_samples, 1, 4, [2, 3])}$$

Fisher's z-transformation [1 point]

The Fisher z-transformation or r is defined as

$$\texttt{fisherz}(r) = \frac{1}{2} \ln \left(\frac{1+r}{1-r} \right) = \operatorname{arctanh}(r)$$

The Fisher z-transformation is used to turn the sample partial correlation (which ranges from -1 to 1) into a value ranging from $-\infty$ to ∞, as follows:

$$\widehat{z}(X_i, X_j, \mathbf{S}) := \sqrt{n - |\mathbf{S}| - 3} \cdot \texttt{fisherz}(\widehat{\rho}(X_i, X_j, \mathbf{S}))$$

where n denotes the number of samples.

Write a function `compute_test_statistic(samples, i, j, S)` which computes $\widehat{z}(X_i, X_j, \mathbf{S})$ from an $n \times p$ matrix `samples`. You can check the correctness of your function by checking that

$$\widehat{z}(X_1, X_7, \{X_3, X_4\}) = \texttt{compute\_test\_statistic(pcalg\_samples, 1, 7, [3, 4])} \approx 1.3244,$$

and

$$\widehat{z}(X_1, X_7, \varnothing) = \texttt{compute\_test\_statistic(pcalg\_samples, 1, 7, [])} \approx -30.0345.$$

(c) Report the value of

$$\widehat{z}(X_1, X_4, \{X_2, X_3\}) = \texttt{compute\_test\_statistic(pcalg\_samples, 1, 4, [2, 3])},$$

p-values [1 points]

Under the null hypothesis that $\rho(X_i, X_j, \mathbf{S}) = 0$, the test statistic $\widehat{z}(X_i, X_j, \mathbf{S})$ is distributed according to a standard normal distribution. Let Φ denote the cumulative distribution function for the standard normal distribution. Then the p-value for $\widehat{z}(X_i, X_j, \mathbf{S})$ is the value

$$1 - 2 \cdot \Phi\left(|\widehat{z}(X_i, X_j, \mathbf{S})|\right)$$

Write a function `compute_pvalue(samples, i, j, S)` which computes the p-value for $\widehat{z}(X_i, X_j, \mathbf{S})$ from an $n \times p$ matrix `samples`. You can check the correctness of your function by checking that

$$\texttt{compute\_pvalue(pcalg\_samples, 1, 7, [3, 4])} \approx 0.1854,$$

and

$$\texttt{compute\_pvalue(pcalg\_samples, 1, 7, [])} \approx 0.0.$$

(d) Report the value of

$$\texttt{compute\_pvalue(pcalg\_samples, 1, 4, [2, 3])},$$

Skeleton phase [2 points]

We will now use these computed p-values to perform a conditional independence test. In particular, suppose we want to test the conditional independence statement $X_i \perp\!\!\!\perp_{\mathbb{P}_{\mathcal{X}}} X_j \mid \mathbf{S}$. Given a significance level α, we say that we reject the null hypothesis of conditional independence if

$$\texttt{compute\_pvalue(samples, i, j, S)} \leq \alpha$$

Otherwise, we say that the test passes.

Write a function `pcalg_skeleton(samples, alpha)` which perform the skeleton phase of the PC algorithm using the conditional independence test described, i.e., we remove the edge $X_i - X_j$ if some conditional independence test between X_i and X_j passes.

You can check the correctness of your function by checking that `pcalg_skeleton(pcalg_samples, 0.05)` outputs the graph in Figure 1.

(e) Report the number of edges in the estimated skeleton when $\alpha = 0.2$ and using only the first 500 rows of `samples`, i.e., in Python, the number of edges in the skeleton output by

```
pcalg_skeleton(samples[:500], 0.2
```

(f) Report the number of edges in the estimated skeleton when $\alpha = 0.001$ and using only the first 500 rows of `samples`, i.e., in Python, the number of edges in the skeleton output by

```
pcalg_skeleton(samples[:500], 0.001
```

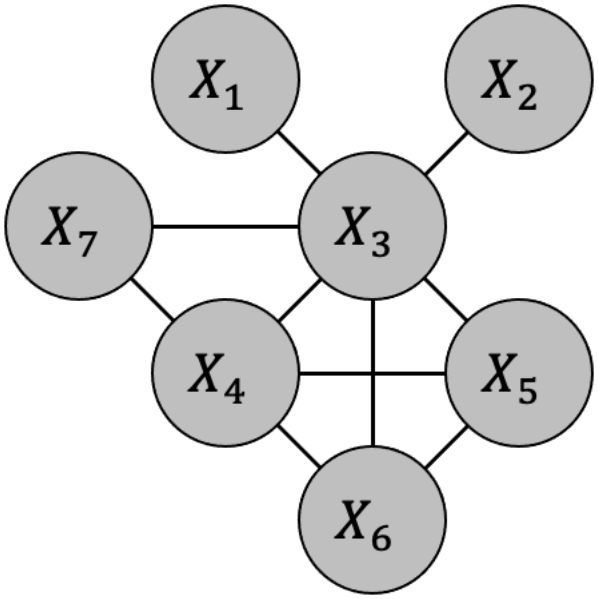

Figure 1: The skeleton that should be output by (e).

Orientation phase [2 points]

Write a function `pcalg_orient(skeleton, separator_function)` which outputs any unshielded colliders according to the orientation phase of the PC algorithm. *You do **not** need to implement the final step of the orientation phase which applies the Meek rules.*

(g) Let `estimated_skeleton`, `estimated_separator_function` be the output of `pcalg_skeleton(pcalg_samples, 0.05)`, i.e., the output that you used to check your implementation of `pcalg_skeleton`. What are the unshielded colliders output by `pcalg_orient(estimated_skeleton, estimated_separator_function)`?

(h) Show that, by the Meek rules, you can orient $X_3 \to X_4$, $X_3 \to X_5$, $X_3 \to X_6$, $X_3 \to X_7$. Can you orient any more edges? If not, explain why.