Assignment 2

Due: Saturday, 11/26/2022 at 11:59 pm -- Total 5 points

Causal AI and Machine Learning (IS 698/800)
Information Systems
University of Maryland, Baltimore County

Instructions:

The completed assignment file must be submitted via Blackboard submission folder by 11:59pm on **11/26/2022**.

Submission checklist:

You have both coding and non-coding analyses in this assignment. Please follow the guideline provided below for submission.

1. For question 1, submit a document that shows your work. You can show your work in this document. You can also do your work in papers, take pictures after you are done, and insert pictures of your work in the document.
2. For question 2, you need to write code in a Jupyter Notebook. Google Colab provides free execution environment for Jupyter Notebook. You should either share the Notebook via a GitHub link or Google drive folder link so I can see the date of the file. Then provide a clickable link. You should also save your code as an ipynb/Rmd file and upload to BlackBoard. The Notebook should have your name at the beginning.

Please prepare your answers using this document and submit it as a single file (preferably as a *.doc or *.docx file or *.pdf). You can do your work in papers and insert clear picture of your work in the document. List the names of the group members.

Important Note on Academic Integrity:

This is an **individual assignment**. You may NOT receive help of any kind from ANY person other than the instructure, who is available to answer clarification questions via slack messages or office hours. Please review the academic conduct requirements from the course syllabus. Any evidence of plagiarism or cheating or other academic integrity violations will lead to a 0 mark for the assignment, and a possible F grade for the course. We do employ safe-assign on Blackboard and other observational ways to identify such incidents. **We might also randomly select students to explain their work**. You should be able to explain how you did the assignment.

Questions

Q1 – 1.5 points

Markov Equivalence Class – The Marko equivalence class of DAGs can be represented using a Completed Partially Acyclic Graph or CPDAG. Find the Markov equivalence class of the DAGs (directed acyclic graphs) given in Fig 1.

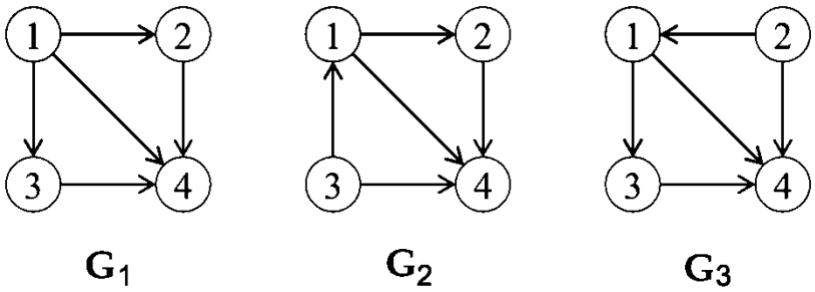

Fig. 1. Directed acyclic graphs.

Q2 – 3.5 points

[Code] Structure Learning

In this problem, you need to write code in a Jupyter notebook to estimate the graphical structure of a model from the given dataset. The data is given in the attached file named "dataset.csv". The causal model is Markovian, that is, it does not have any unobserved confounders.

You are allowed to use existing libraries to write the solution to this problem. Also, you can code your solution without the help of any packages. You are allowed to use either Python or R to write code. You can follow the causal discovery coding example that we discussed in class.

1. Load and display the given dataset from the file. [0.5 point]
2. Use the PC algorithm to estimate an equivalence class for the given dataset. Please use $\alpha = 0.05$ in your conditional independence tests. Plot the graph. [1 point]
3. Use the GES algorithm to estimate an equivalence class for the given dataset. Plot the graph. [1 point]
4. The PC and GES algorithms estimate CPDAGs. There are some edges in CPDAGs associated with the answer to question (1) and (2) that are not oriented. Let's say we assume that the data was generated by an additive noise model. In that case, we can test each undirected edge under the assumption that the function is either non-linear or that the unobserved parent is non-Gaussian. With these assumptions, let's say we found that the edge between F and B nodes can be oriented as F→B. Now, given this information draw all possible causal graphs. [1 point]

[Note: For question (4), you can draw these causal graphs using code or on paper. Please attach the hand-drawn graphs in your submitted solution.]

For questions (2), (3) and (4), if you use Python to write code and require code to perform conditional independence test then you can use the following code.

```
from sklearn.linear_model import LinearRegression

# numpy data of x independent of data y conditioned on array of z vectors
def independent(x,y,z=[]):
      x = (x-x.mean())/x.std()
      y = (y - y.mean())/y.std()
      if len(z)>0:
      z = np.stack(z).T
      regy = LinearRegression().fit(z, y)
      y = y - regy.predict(z)
      regx = LinearRegression().fit(z, x)
      x = x - regx.predict(z)
      x = (x-x.mean())/x.std()
      y = (y - y.mean())/y.std()

return abs((x*y).mean() - x.mean()*y.mean())*100 < 1
```