CS7290 Causal Modeling in Machine Learning: Homework 3

Submission guidelines

Use a Jupyter notebook and/or R Markdown file to combine code and text answers. Compile your solution to a static PDF document(s). Submit both the compiled PDF and source files. The TA's will recompile your solutions, and a failing grade will be assigned if the document fails to recompile due to bugs in the code. If you use Google Collab, send the link as well as downloaded PDF and source files.

Background

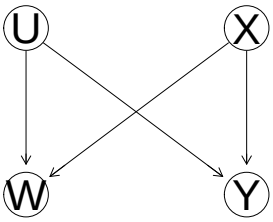
This assignment is going to cover several topics, including some that haven't been taught at the time this was assigned. We will cover those topics in subsequent classes.

- Recognizing valid adjustment sets
- Covariate adjustment with parent and back-door criterion
- Front-door criterion
- Propensity matching and inverse probability weighting
- Intro to structural causal models

1. Recognizing valid adjustment sets

1.1

The following DAG represents a causal model of user behavior in an app.



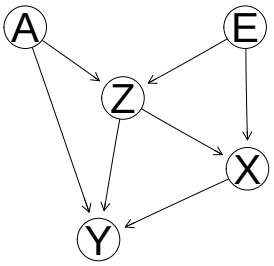
U represents the user specific preferences. X represents the introduction of a feature designed to make users make certain in-app purchases, Y was whether or not the user made the purchase, W represents app usage after the feature is introduced.

1. (3 points) You are interested in estimating the causal effect of X on Y. What is the valid adjustment set?
2. (2 points) What would happen if you adjusted for W? Be specific.
3. (4 points) Suppose you want to assess the effect of X on Y for users who go on to have a high amount of app usage. You wanted to compute the causal effect for each level of W. Fill in the blanks on the right-hand-side and left-hand-side for the adjustment formula of interest:

$$P(Y = y|?) = \sum_{?} P(Y = y|?)P(?|?) \tag{1}$$

1.2

Consider the following DAG.

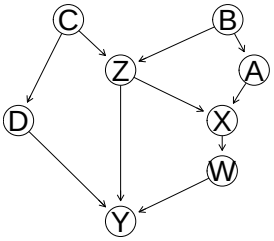


You are interest in estimating the causal effect of X on Y.

1. (2 points) Is the set containing only Z a valid adjustment set? Why or why not?
2. (3 points) List all of the valid adjustment sets (there are three) and write the adjustment formula for each adjustment set.
3. (1 point) Suppose that E and A are both observable, but observing E costs $10 per data point and observing A costs $5 per data point. Which conditioning set do you go with?

1.3

Consider the following DAG:



(3 points) 1. List all of the sets of variables that satisfy the backdoor criterion to determine the causal effect of X on Y. (3 points) 2. List all of the minimal sets of variables that satisfy the backdoor criterion to determine the causal effect of X on Y (i.e., any set of variables such that, if you removed any one of the variables from the set, it would no longer meet the criterion). (3 points) 3. List all the minimal sets of variables that need to be measured in order to identify the effect of D on Y. (3 points) 4. Now suppose we want to know the causal effect of intervening on 2 variables. List all the minimal sets of variables that need to be measured in order to identify the effect of set {D, W} on Y, i.e., $P(Y = y|do(D = d), do(W = w))$.

2. Covariate adjustment

2.1

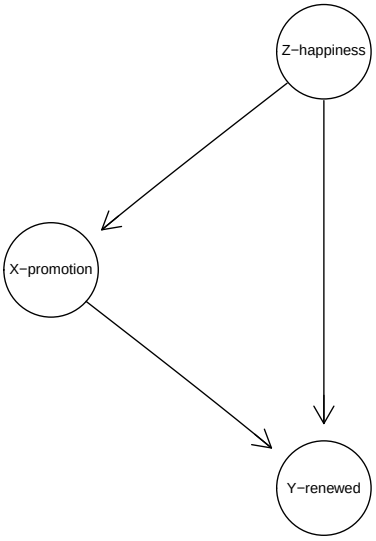
You are a data scientist at a prominent tech company with paid subscription entertainment media streaming service. You come across some data on a promotional campaign. The campaign targeted 70K subscibers users who were coming to a subscription renewal time and were at high risk of not renewing. They were targeted with two types of promotions, call them promotion 0 and promotion 1.

	Overall
Promotion 0	77.9% (27272/35000)
Promotion 1	82.6% (28902/35000)

You do some digging and find out the promotions the users were offered dependended on how happy the users were (quantified from user behavior and customer service interactions).

	Overall	Unhappy	Happy
Promotion 0	77.9% (27272/35000)	93.2% (8173/8769)	73.3% (19228/26231)
Promotion 1	82.6% (28902/35000)	86.9% (23339 / 26872)	68.7% (5582/8128)

You assume the following causal DAG:



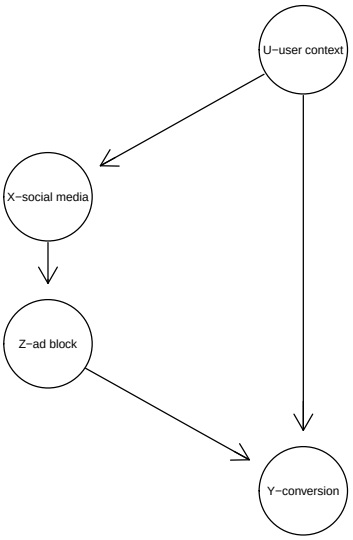
You are interested in the average causal effect $P(Y = 1|\text{do}(X = 0)) - P(Y = 1|\text{do}(X = 1))$

1. (5 points) Build the model with Pyro using the values in the table. Use `pyro.condition` to calculate the causal effect by adjusting for happiness.
2. (5 points) Suppose you could not observe happiness. Use `pyro.do` to calculate the causal effect with do-calculus.

unit = 1K	No adblock (50%)		Adblock (50%)		All subjects (800)	
	social	no social	social	no social	social	no social
Total	380	20	20	380	400	400
No conversion	323 (85%)	1 (5%)	18 (90%)	38 (10%)	341 (85.25%)	39 (9.75%)
Conversion	57 (15%)	19 (95%)	2 (10%)	342 (90%)	59 (14.75%)	361 (90.25%)
	No adblock	Adblock	No adblock	Adblock	No adblock	Adblock
No conversion	323 (85%)	18 (90%)	1 (5%)	38 (10%)	324 (81%)	56 (14%)
Conversion	57 (15%)	2 (10%)	19 (95%)	342 (90%)	76 (19%)	344 (86%)

2.2

Consider the table and the corresponding causal model.



1. (5 points) User context is unobserved. Use `pyro.condition` to calculate the causal effect of social media on conversions using front-door adjustment.
2. (5 points) Verify your result from number 1 using do-calculus with `pyro.do`.

3. Inverse probability weighting with a propensity score.

Probabilistic programming generally works by executing the program many times, and then reasoning on the ensemble of *program executions*, which vary because the program is probabilsitic. A program execution is typically called an *execution trace*, or just *trace*. The data structure representing a trace stores the values of the variables in the program, the log-probability of the trace, as well as other useful items. Pyro has a class called `Trace` that serves as a trace data structure. Given the following model:

```
def model():
  x = sample('x', Normal(0, 1))
  y = sample('y', Normal(x, 1))
  return x, y
```

Suppose you wanted to generate 3 samples from the model as well as the probability of each sample. You can use the following approach to handle and generate traces.

```
import numpy as np
trace_handler = pyro.poutine.trace(model)
for i in range(3):
  trace = trace_handler.get_trace()
  x = trace.nodes['x']['value']
  y = trace.nodes['y']['value']
  log_prob = trace.log_prob_sum()
  p = np.exp(log_prob)
  print(x, y, p)
```

Questions:

1. (3 points) Use the data in problem 2.1 to create the following propensity score function:

   ```
   def propensity(x, z):
   # returns P(X = x | Z = z)
   ...
   ```

2. (3 points) Use the model from problem 2.1 to generate 1000 samples, along with the sample probabilities. Print the first 10 samples.

3. (1 point) Using your `propensity` function, create a list of weights by, for each sample i, multiplying the sample probability by $1/P(X = x_i|Z = z_i)$.

4. (3 points) Sample with replacement 1000 samples from the original list using the new weights.

5. (3 points) Call this new set of samples Ω. Let $p^{\Omega}(X = x)$ be the proportion of times $X == x$ in Ω and $p^{\Omega}(X = x|Y = y)$ be the proportion of the Ω samples where $X == x$ after filtering for samples where $Y == y$. If you performed the above inverse probability weighting procedure correctly, then $P^{\text{model}}(Y = y|\text{do}(X = x)) \approx p^{\Omega}(Y = y|X = x)$ (the LHS and RHS are equal as the sample size goes to infinity). Confirm this by recalculating the causal effect from problem 2 using this method.

4. Structural causal models

1 (3 points)

Consider the SCM $\mathbb{M}$:

$$\begin{aligned} X &:= N_X \\ Y &:= X^2 + N_Y \\ N_X, N_Y &\overset{\text{i.i.d}}{\sim} N(0,1) \end{aligned}$$

Write this model in Pyro and generate 10 samples of X and Y.

2

Consider the SCM $\mathbb{M}$:

$$\begin{aligned} X &:= N_X \\ Y &:= 4X + N_Y \\ N_X, N_Y &\overset{\text{i.i.d}}{\sim} N(0,1) \end{aligned}$$

1. (1 point) Draw a picture of the model's DAG.
2. (2 points) $P_Y^{\mathbb{M}}$ is a normal distribution with what mean and variance?
3. (2 points) $P_Y^{\mathbb{M}:do(X=2)}$ is a normal distribution with what mean and variance?
4. (2 points) How and why does $P_Y^{\mathbb{M}:X=2}$ differ or ot differ from $P_Y^{\mathbb{M}:do(X=2)}$?
5. (2 points) $P_X^{\mathbb{M}:Y=2}$ is a normal distribution with what mean and variance?
6. (2 points) $P_X^{\mathbb{M}:do(Y=2)}$ is a normal distribution with what mean and variance?
7. (3 points) Write this model in code and generate 10 samples from $P_{X,Y}^{\mathbb{M}}$.
8. (2 points) Use the `do` operator to generate 100 samples from $P_Y^{\mathbb{M}:do(X=2)}$ and visualize the results in a histogram.
9. (3 points) Use the `condition` operator and a Pyro inference algorithm to generate 10 samples from $P_X^{\mathbb{M}:Y=2}$. Use one of the Bayesian inference procedures described in the lecture notes.

4.3 Counterfactual inference algorithm

X and Y are causes of Z. The causal mechanism is either an AND gate or and OR gate depending on initial conditions.

AND Gate		
X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

OR Gate		
X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

There is a 50% probability it is an AND gate and a 50% probability it is an OR gate. X and Y both have a 50% chance of being equal to 1 in both of the gates.

The following code represents the structural assignments in a structural causal model of this system.

```
def fx(N):
  X = N
  return X

def fy(N):
  Y = N
  return Y

def fz(X, Y, N):
  # Mixture of AND gate and OR gate
  Z = N * min((X + Y), tensor(1.)) + (tensor(1.) - N) * (X * Y)
  return Z
```

Problem solving hint: Pyro has a distribution called `Delta`. Its constructor takes only one parameter (e.g. `Delta(a)`), and when you sample from it, you always get a value equal to that parameter. In other words all of the probability in the distribution is concentrated on the parameter. For example, if you write `A = sample("A", Dirac(a))`, then the value you sample for A will always be `a`. Why would you want `A = sample("A", Dirac(a))` instead of just `A = a`? The reason the `sample` function has you name a variable (e.g. `"A"` in `sample("A", ...)`) is so you can store it by name in the trace object (see Problem 3 for a discussion of traces), and refer to that item later with expressions like `condition(model, {"A": a})`. When you have a deterministically set variable and you want to apply `condition` or `do` to it, you can sample it from a `Delta` distribution.

1. (Calculate by hand, 1 point) Suppose we observe that X is 1 and Z is 1. What is the probability it is an OR gate?
2. (1 point) What is $P(Y = 1|X = 1, Z = 1)$?
3. (Calculate by hand, 1 point) Suppose we observe that X is 1 and Z is 1. What would Z have been if X were 0? Express this as a probability distribution (assign a probabilities to Z == 1 and Z == 0).
4. (2 points) Fill in the "..." in the following SCM.

   ```
   def model():
     Nx = sample('Nx', Bernoulli(tensor(.5)))
     Ny = sample('Ny', Bernoulli(tensor(.5)))
     Nz = sample('Nz', Bernoulli(tensor(.5)))
     ...
     return X, Y, Z
   ```

5. (4 points) Condition the model on X = 1 and Z = 1. Infer the posterior on the noise distribution conditional on X = 1 and Z = 1 using importance sampling. Do this by passing the conditioned model to `pyro.infer.Importance`, and naming the resulting object `posterior`. You know it worked if `type(posterior)` returns an object of the class `pyro.infer.importance.Importance`, and

`type(posterior())` returns and object of the class `pyro.poutine.trace_struct.Trace`. Calculate $P(Y = 1|X = 1, Z = 1)$.

6. (5 points) Compute the counterfactual probability $P^{\text{model};X=1,Z=1,do(X=0)}(Z = 1)$ using the counterfactual algorithm described in class, and compare the result with your math:

    1. Create an intervention model using the intervention $do(X = 0)$.
    2. As in problem 3, iteratively generate samples of Z by sampling a trace from the posterior in each iteration.
    3. In each iteration, pull the values Nx, Ny, and Nz from the trace and condition the intervention model on these values. Then generate a sample value of Z. Each time you do this, you simulate the counterfactual model by using noise values conditional on real evidence, and combining it with an intervention that conflicts with that evidence.
    4. Calculate $P^{\text{model};X=1,Z=1,do(X=0)}(Z = 1)$ as the average value of Z in the samples.