CSCE A405/A605 (Adv) Artificial Intelligence

Patrick Pragman
UAID 30812903
21NOV22

1. Which of the following independencies is guaranteed in the above Bayes net?

   (a) "Steering course failure" $\perp$ "High ship speed"| "Operational system failure"
   **Answer:**Not guaranteed, because "operational system failure" is a child of both.

   (b) "Communication failure" $\perp$ "Presence of iceberg on navigation course" | "High iceberg density"
   **Answer:**Independent

   (c) "Mechanical failure" $\perp$ "Iceberg-ship collision" | "Technical operation failure"
   **Answer:**Not guaranteed, because "Iceberg-Ship collision" is a descendent of both Technical Operation Failure" and "Mechanical Failure."

   (d) "Mechanical failure" $\perp$ "Iceberg-ship collision" | "Navigational failure"
   **Answer:**Not guaranteed for the same reason as above

2. Assuming the domain of each variable to be True, False, and the following CPTs, compute p(H | P=True, W=True). Run iceberg.ipynb and confirm your results.

   **Answer:**

$$P(H|P,W) = \sum_{s \in S} P(H|P,W,s)P(s)$$

$$= P(H|P,W,\neg S)P(\neg S) + P(H|P,W,S)P(S)$$

$$= P(H|P,W,\neg S)(0.7) + P(H|P,W,S)(0.3)$$

$$= (0.4)(0.7) + (0.8)(0.3)$$

$$= 0.28 + 0.24 = 0.52$$

   which matches the findings from the program.

3. Run the last two cells in the iceberg.ipynb. Find p(H—S=0) using the generated samples. Explain your findings and that which algorithms these cells implement.

   **Answer:**

   This bit of code is generating some samples of the Bayes net.

   Between "liklihood weighted sample" and "forward sample" I get slightly different values for $P(H|S=0)$ which makes quite a bit of sense. I get slightly different values even if I set the seed to the same value. This is because they sample in different ways.

A forward sample enters the Bayesnet at the nodes without parents. It randomly generates samples from node to child node as it heads down the Bayes net. The likelihood weighted sampling is similar and calculates the sample corrected for probability of getting that value as you go down the Bayes net.

Regardless, you get two sets of samples, of which the likelihood weighted sample is probably the better sample.

You can really see how this technique works when you set the same random seed and increase the size of the sample.

Still, I wanted to write some code to calculate $P(H|S = 0)$ given the samples, so I modified the cells a little bit to get this (see attached as well):

For the forward sample:

```
inference = BayesianModelSampling(bayesNet)
df = inference.forward_sample(size=20)

# calculate P(H|S=0)
S_eq_0 = df[df['S'] == 0]  # filter dataframe by S = 0

# now count the entries in both and generate a probability
p_h_given_s_eq_0 = len(S_eq_0[S_eq_0['H'] == 1])/len(S_eq_0)
print(p_h_given_s_eq_0)
print(df)
```

This renders the following output:

```
100%|██████████| 5/5 [00:00<00:00, 533.38it/s]
100%|██████████| 20/20 [00:00<00:00, 2078.39it/s]
    P  S  W  H
0   0  0  1  0
1   0  0  1  0
2   0  0  1  0
3   0  0  0  0
4   0  0  0  0
5   0  0  1  0
6   1  0  0  1
7   0  0  1  0
8   0  0  0  0
9   0  0  0  0
10  0  0  1  0
11  1  0  1  0
12  1  0  0  0
13  1  0  1  1
14  0  0  1  1
15  0  0  0  0
16  0  0  1  0
17  0  0  0  0
18  0  0  0  0
19  0  0  0  0
0.15
```

Similarly, we can see the following code for a liklihood weighted sample:

```
inference = BayesianModelSampling(bayesNet)
evidence = [State('S', 0)]

df = inference.likelihood_weighted_sample(evidence=evidence, size=20)
```

```
# calculate P(H|S=0)
S_eq_0 = df[df['S'] == 0]  # get all the cases where S = 0

# divide the cases where that has H = 1 by the total amount
# of entries in the S = 0 data
p_h_given_s_eq_0 = len(S_eq_0[S_eq_0['H'] == 1])/len(S_eq_0)

print(p_h_given_s_eq_0)
print(df)
```

The following output is rendered:

```
Generating for node: H: 100%|| 4/4 [00:00<00:00, 236.27it/s]
0.25
    P  S  W  H  _weight
0   1  0  1  0     0.7
1   0  0  1  0     0.7
2   1  0  1  0     0.7
3   0  0  1  0     0.7
4   0  0  1  0     0.7
5   1  0  1  0     0.7
6   1  0  0  0     0.7
7   1  0  1  0     0.7
8   1  0  0  1     0.7
9   0  0  0  1     0.7
10  0  0  1  1     0.7
11  0  0  1  0     0.7
12  0  0  1  1     0.7
13  0  0  0  0     0.7
14  1  0  0  0     0.7
15  1  0  0  0     0.7
16  0  0  0  0     0.7
17  1  0  0  0     0.7
18  1  0  0  0     0.7
19  1  0  1  1     0.7
```

Note how it's close to the same value, remember that this is generating a random sample based on the likelihood of the nodes in the graph. If you want to get the same output from it every time you run it, set the seed to be the same, but you'll notice that the forward sample and the *liklihood_weighted_sample* will most likely still be different.

4. Modify the program iceberg.ipynb to include two other variables L and V as follows.

   (a) Submit your modified notebook or a screenshot of it.

   **Answer:**

   I have attached a Python file to this submission. It turned out to be cleaner to separate the code into a separate python file from the notebook.

   (b) Find $P(V|L, \neg S)$.

   **Answer:**

   I ran the script mentioned above and received the following output:

   ```
   P(V | L, ~S) =  0.6432325295053906$
   ```

   To calculate this, I used a likelihood weighted sample with a size of 1 million. If you run it several times you get around 64 percent. Intuitively, this makes sense, because the probability of L is fairly low, and the probability of not S is fairly high, so you would expect it

(c) Write the joint probability distribution p(P, S, W, L, H, V) in terms of conditional probabilities. (Reminder: Similar to what we did in the lectures, Example 1: $p(T, J, R, S) = p(T|R, S)p(J|R)p(R)p(S)$)

**Answer:**

We can use the following identity to solve this:

$$P(x_1, x_2, x_3, ...x_n) = \prod_{i \in X} p(x_i|Parents(x_i)$$

So:

$P(P, S, W, L, H, V) = P(P|Parents(P)) \cdot P(S|Parents(S)) \cdot P(W|Parents(W)) \cdot P(L|Parents(L)) \cdot P(H|Parents(H))P(V|Parents(V))$

This is a lot, but we can simplify it some, for one, P, S, W, and L have no parents, so:

$$P(P, S, W, L, H, V) = P(P) \cdot P(S) \cdot P(W) \cdot P(L) \cdot P(H|Parents(H)) \cdot P(V|Parents(V))$$

We can simplify this some too

$$P(P, S, W, L, H, V) = P(P) \cdot P(S) \cdot P(W) \cdot P(L) \cdot P(H|P, S, W) \cdot P(V|L, H)$$

From the table we can actually calculate this

$$= (0.4)(0.3)(0.5)(0.8)(0.8) = 0.00384$$

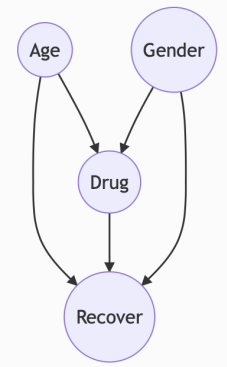
5. A doctor gives a patient a (D)rug dependent on their (A)ge and (G)ender. Whether or not the patient (R)ecovers depends on all D,A,G. In addition A$\perp$ G—$\emptyset$.

   dom(D) = drug,no drug dom(A) = old, young dom(G) = female,male dom(R) = recovers, doesn't recover

   (a) Draw a Bayesian network for the above situation.

   **Answer:**

   Age and gender are independent of each other and feed into drug, but also you can recover without the drug at all, so age and gender directly feed into the node "Recover."



(b) Explain how to compute p(recover|drug)

**Answer:**

I reckon the best way to do this would be "inference by enumeration."

$$P(R|D) = \alpha P(R, D) = \alpha \sum_{y \in Y} P(R, y, D)$$

where $Y$ is all of the possible combinations age and gender.

Or

$$P(R|D) = \alpha P(R, D) = \alpha \sum_{a \in Age} \sum_{g \in Gender} P(R, a, g, D)$$

$$= \alpha \Big[ P(R, young, woman, D) + P(R, old, woman, D) + P(R, young, man, D) + P(R, old, man, D) \Big]$$

This would give us all the possible combinations and matches our intuition.

(c) Explain how to compute p(recover|drug,young)

**Answer:**

This should follow the same thought process as the previous problem, but we don't have as much to iterate over:

$$P(R|D, young) = \alpha P(R, D, young) = \alpha \sum_{g \in Gender} P(R, Age = young, g, D)$$