Released: DATE HERE

Table of Contents

Introduction

In this assignment, we will be working with a python representation of a Directed Acyclic Graph, or DAG. You will then encode an algorithm required to make exact inferences given a DAG; this called Variable Elimination.

Then, we will explore our ability to represent **causal structures** using a DAG. As we've discussed in class, Causal DAGs are DAGs that make some assertions above and beyond a regular DAG. With a Causal DAG, each arrow represents causal flow. Moreover, in a Causal DAG, all arrows have purpose. It is therefore not permitted to place an arrow between variables when they are independent or conditionally independent. The arrows in our Causal DAG will therefore represent a "minimal" set.

With a Causal DAG, we can ask Causal questions, i.e. questions of the form "If I change the value assigned to this variable, what will be the effect on the other variables?". These kinds of questions are routinely asked in disciplines like medicine, as scientists may want to determine the causal effect of treatments on diseases, or the effect of lifestyle choices on health outcomes. Causal questions are equally important to practitioners of Artificial Intelligence and Machine Learning (AI/ML), as we can use these questions to assess the causal effects of an AI/ML system's inputs on its outputs. Say, for example, a green pixel in the upper right hand corner of an image is found to cause the self-driving car you have built to accelerate. This could be a problem, should you deploy your car in Greenland (and, of course, assuming Greenland is very green)!

In this assignment, we'll do some investigation of causal effects as they relate to COVID mortality, and using some real data.

You can download the starter files below.

- **What is supplied:**
  - Python code that implements Variable, Factor, and BN (or DAG) objects. The file bnetbase.py contains the class definitions for these objects. The code supports representing factors as tables of values indexed by various settings of the variables in the factor's scope.
  - COVID-19 data (description here)

The template file **solution.py** contains prototypes for the functions you must implement.

Implement Variable Elimination (worth 50/100 marks)

To begin, we will develop some machinery to make exact inferences based on an arbitrary BN (or DAG). We will be using the function called “VE” to make the inferences, but this function will depend on several subroutines. As we’ve discussed in class, to make exact inferences we will sometimes need to multiply factors together, and other times we will need to sum, restrict or normalize them. We will also need to be careful as to which factors we process before others! We will therefore encode a heuristic (called “Min Fill Ordering”) to identify variables that are associated with small factors. Here is a list of functions you need to implement. Add your code under each function in the **solution.py** file.

- **VE (worth 25 points).** This is the function that performs exact inference. It takes as input a Bayes Net object (object of class BN), a variable that is the query variable Q, and a list of variables E that are the evidence variables (all of which have had some value set as evidence using the variable’s set evidence interface). It will then compute the probability of every possible assignment to Q given the evidence specified by the evidence settings of the evidence variables. Return probabilities as a list, where every number corresponds the probability of one of Q’s possible values. Do not modify any factor of the input Bayes net.

To complete VE you will need implementations of the following functions:

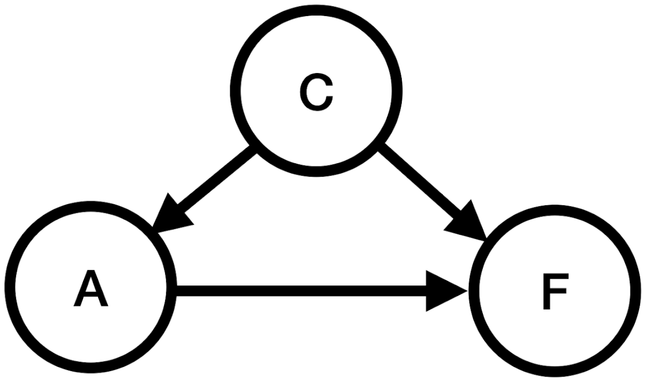
- **multiply_factors (worth 7 points).** This function takes as input a list of Factor objects; it creates and returns a new factor that is equal to the product of the factors in the list. Do not modify any of the input factors.
- **restrict_factor (worth 7 points).** This function takes as input a single factor, a variable V and a value d from the domain of that variable. It creates and returns a new factor that is the restriction of the input factor to the assignment V = d. Do not modify the input factor.
- **sum_out_variable (worth 7 points).** This function takes as input a single factor, and a variable V; it creates and returns a new factor that is the result of summing V out of the input factor. Do not modify the input factor.
- **normalize (worth 7 points).** This function takes as input a list of numbers. It will then calculate the sum of these numbers and use this sum as a normalizing constant. It will then return a new list of numbers wherein the numbers sum to 1, i.e., it will apply the normalizing constant to normalize the input numbers.
- **min_fill_ordering (worth 7 points).** This function takes as input a list of factors and a query variable and returns a list of variables that are ordered according to the min fill heuristic. The list will be ordered such that the first variable in the list generates the smallest factor upon elimination. The query variable should NOT be part of the returned ordered list.

Causal Graphs (worth 40/100 marks)

For this section, we will look at COVID-19 data. This data consists of a list of individuals that contracted COVID in both Italy and China, and it is derived from data that is published and has been analyzed by von Kugelgen et al (see their publication at this URL). This data can be found in "covid.csv"; every row represents an individual that was reported as having COVID in either February or March of 2020. The columns contain the following categorical variables:

- Country of Residence (Italy or China)
- Age (‘0-9’, ‘10-19’, ‘20-29’, ‘30-39’, ‘40-49’, ‘50-59’, ‘60-69’, ‘70-79’, ‘80+’)
- Fatality (“Yes” or “No”)

First, use the BN code that has been supplied to create a Causal Bayesian Network using the data in "covid.csv" and that respects the architecture posited by Kugelgen et al, and which can be illustrated as follows:


$$\begin{tikzcd} & C \arrow[dl] \arrow[dr] & \\ A \arrow[rr] & & F \end{tikzcd}$$

Here, “C” is the “Country of Residence”, “A” is “Age” and “F” is whether or not an individual dies of COVID upon having contracted it. Note that this model posits that “Age” is a **mediator** of the effect of the Country of Residence on Fatality.

Next, use the BN code that has been supplied to create a Causal Bayesian Network using the data in "covid.csv" but which reflects a different architecture, as follows:

$$\begin{tikzcd}
 & C \arrow[dr] & \\
A \arrow[ur] \arrow[rr] & & F
\end{tikzcd}$$

Once again, “C” is the “Country of Residence”, “A” is “Age” and “F” is whether or not an individual dies of COVID upon having contracted it. Note however that this model posits that “Age” is a **confounder** of the effect of the Country of Residence on Fatality.

Let’s now use your VE implementation and the BNs you’ve constructed, above, to estimate the causal effect of Country (Italy v. China) on Fatality.

1. Your first estimate should take into account that some of the effect of “Country” will be mediated by age. Use the formula for *Average Causal Effect (ACE)* that was presented in class to make this calculation.

2. Your second estimate should take into account that the effect of “Country” on fatality will be CONFOUNDED by age. Use the *adjustment formula* for Causal Effect calculations that was presented in class in order to make this calculation.

3. You should notice that your estimates of effect depend quite a bit on your model assumptions!! Given that this is the case and if you were asked for an estimate of the *true* effect of country on fatality, which calculation above might you prefer, and why?

Creating a Causal Graph (worth 10/100):

Diabetes is widespread disease that effects many people all over the world. Scientists have identified variables that are associated with the onset and progression of diabetes. Assume you are a data scientist tasked with studying the causal relationships between different known variables that are related to diabetes. You are given a dataset that includes measures of known variables that are related to diabetes and the severity of each individual's diabetes. These variables are as follows:

- Genetic Predisposition: This is a measure of the genetic susceptibility of an individual to developing diabetes (and is informed by information including the number of direct relatives that have diabetes, and the existence of genetic markers linked to the disease)
- Body Mass Index (BMI): This is a measure of body fat based on an individual's weight and height.
- Dietary Patterns: This is a measure of the type and quality of an individual's diet that is sensitive to an individual's consumption of sugary foods, processed foods, and fruits and vegetables.
- Physical Activity: This measures level of physical activity and exercise undertaken by an individual.
- Blood Glucose: Levels of glucose in the blood, which can indicate diabetes control and management.
- Medication: The use of specific medications or treatments targeted at managing diabetes.
- Comorbidities: Other existing medical conditions or diseases that a patient may have in addition to diabetes.
- Diabetes Severity: A measure of the severity of an individual's disease.

Create two different, yet plausible, causal graphs that illustrate causal relationships between the variables above and diabetes severity. In each graph that you create, identify and justify one mediating variable, one confounding variable, and one collider variable. Limit your justifications to 100 words per variable.

Self-Assessment (worth 2.5 bonus marks)

In 100-250 words, respond to the question "***What is the most significant thing you have learned from the Bayesian and Causal Network module, overall?"***. *This will complete our final "self-assessment exercise" for the term (and will be worth 0.25% of the course mark).* Place your response in a file called "bns.txt"

Starter Code

The starter code contains one Python starter file and a number of training and test files. You can download these at this link.

In that archive you will find the following files:

Project File (the file you will edit and submit on Markus):

`solution.py`

Data Files:

`covid.csv`

Grader Files (the files used to sanity test your solution):

`autograder.py`

`bnetbase.py`

Check Your Solution

We provide an autograder file for basic validation your solution, which you can run by typing:

```
$ python3 autograder.py
```

This code is only meant to give you some basic intuition as to how your VE methods perform; getting 100% on the grader should give you some confidence that your Bayes Net follows the specifications. But it does not mean you will get 100% as your final mark! You will want to create tests of your own that vary training/testing files to see how your code generalizes. *Different files and examples than those used in the autograder.py file will be used to grade your solution.*

What to Submit

You will be using MarkUs to submit your assignment. You will submit your modified `solution.py` and your answer to questions in `bns.txt`.

Make sure to document the code to help our TAs understand the approach you used.

The assignment is due on **XXX**

.