Computer Science 384
St. George Campus

July 20, 2020
University of Toronto

Homework Assignment #4: Probability and Bayesian Inference
Due: August 11, 2020 by 10:00 PM

Silent Policy: *A silent policy will take effect 24 hours before this assignment is due, i.e. no question about this assignment will be answered, whether it is asked on the discussion board, via email or in person.*

Late Policy: 10% per day after the use of 3 grace days.

Total Marks: This part of the assignment represents 13% of the course grade.

Handing in this Assignment

What to hand in on paper: Nothing.

What to hand in electronically: You must submit your assignment electronically. Download `code.zip` which contains `bnetbase.py` and `medicalDiagnosis.py`. Modify `bnetbase.py` so that it solves Question 1, which specified in this document. Once you are done with this, answer Question 2 using this Google form https://forms.gle/c1P6eHmNZcdtp8Jm9. Identify yourself in the Google form using your teach.cs ID. **Submit your modified files** `bnetbase.py` **as well as** `acknowledgment_form.pdf` using MarkUs Your login to MarkUs is your teach.cs username and password. It is your responsibility to include all necessary files in your submission. You can make as many submissions to MarkUs or to Google Forms as you like while you still have grace days; the number of grace days you use will be determined by the time of your final submission. Only your **last** submission (to Google Forms or to MarkUs) will be marked.

Your code, and the answers you provide to the Google Form, will be tested electronically. You will be supplied with a testing script that will run a **subset** of the tests. If your code fails all of the tests performed by the script (using Python version 3.7), you will receive zero marks. It's up to you to figure out further test cases to further test your code – that's part of the assignment!

When your code is submitted, we will run a more extensive set of tests which will include the tests run in the provided testing script and a number of other tests. You have to pass all of these more elaborate tests to obtain full marks on the assignment.

Your code will not be evaluated for partial correctness, it either works or it doesn't. It is your responsibility to hand in something that passes at least some of the tests in the provided testing script.

- *Make certain that your code runs on teach.cs using python3 (version 3.7) using only standard imports.* This version is installed as "python3" on teach.cs. Your code will be tested using this version and you will receive zero marks if it does not run using this version.
- *Do not add any non-standard imports from within the python file you submit (the imports that are already in the template files must remain).* Once again, non-standard imports will cause your code to fail the testing and you will receive zero marks.
- *Do not change the supplied starter code.* Your code will be tested using the original starter code, and if it relies on changes you made to the starter code, you will receive zero marks.

Clarification Page: Important corrections (hopefully few or none) and clarifications to the assignment will be posted on the Assignment 4 Clarification page, linked from the CSC384 A4 web page, also

found at: http://www.teach.cs.toronto.edu/~csc384h/summer/Assignments/A4/a4_faq.html. You are responsible for monitoring the A4 Clarification page.

Questions: Questions about the assignment should be asked on Piazza:

https://piazza.com/utoronto.ca/summer2020/csc384/home.

You may also reach out to the Assignment 4 TAs, Parsa Mirdehghan (p.mirdehghan at gmail.com) and John Chen (johnn.chen at mail.utoronto.ca), or one of the instructors. Please place "A4" and "CSC384" in the subject line of your email.

Introduction

In this assignment you will implement variable elimination for Bayes Nets.

What is supplied: Python code that implements Variable, Factor, and BN 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.

The template file `bnetbase.py` also contains function prototypes for the functions you must implement.

Question 1. Implement Variable Elimination (worth 60/100 marks)

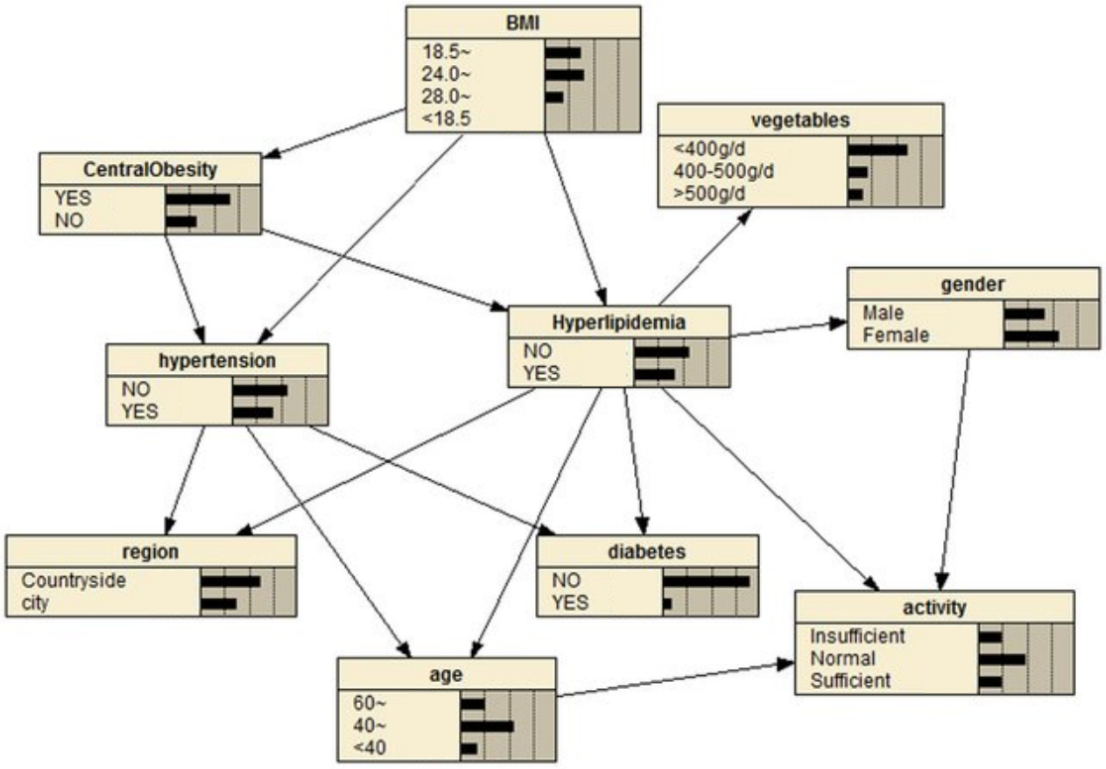
Implement the following functions that operate on Factor objects and then use these functions to implement VE (variable elimination):

- **multiply_factors** (worth 10 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 10 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 10 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 5 points). This function takes as input a list of numbers and returns a new list of numbers where the numbers sum to 1, i.e., the function normalizes the input numbers.
- **VE** (worth 25 points). This function 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). Compute the probability of every possible assignment to Q given the evidence specified by the evidence settings of the evidence variables. Return these 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.

Question 2: Problem Solving with your VE Implementation (worth 30/100 marks)

For the following questions, you will submit your answers using the Google form that is located at https://forms.gle/c1P6eHmNZcdtp8Jm9.

1. Examine the file `medicalDiagnosis.py`. This specifies a Bayes Net for diagnosing various reasons why a a person might have Hyperlipidemia. The layout of this Bayes Net is shown below, and the various CPTs for the Net are specified in `medicalDiagnosis.py` as Factors:



Each variable of the Net is shown in a square box along with the values that the variable can take. For example, The variable *CentralObesity* (which is the variable *co* in the file `medicalDiagnosis.py`) can take on one of two different values: "YES" and "NO".

The numbers and bars show the unconditional probabilities of the variables taking on their different values. For the various CPTs for the Net, see `medicalDiagnosis.py`.

Using your Variable Elimination implementation (or based on inspection!), answer the following questions and post your answers to the Google Form:

(a) (worth 5 points) Show a case of conditional independence in the Net where knowing some evidence item $V1 = d1$ makes another evidence item $V2 = d2$ irrelevant to the probability of some third variable $V3$. (Note that conditional independence requires that the independence holds for all values of $V3$).

(b) (worth 5 points) Show a case of conditional independence in the Net where two variables are independent given one set of evidence, yet they become dependent given evidence at an additional variable.

(c) (worth 10 points) Show a sequence of accumulated evidence items $V1 = d1, ..., Vk = dk$ (i.e., each evidence item in the sequence is added to the previous evidence items) such that each additional evidence item increases the probability that some variable V has the value d. (That is, the probability of V = d increases monotonically as we add evidence items). What is P(V = d—V1 = d1,...,Vk = dk)?

(d) (worth 10 points) Show a sequence of accumulated evidence items $V1 = d1, ..., Vk = dk$ (i.e., each evidence item in the sequence is added to the previous evidence items) such that each additional evidence item decreases the probability that some variable V has the value d. (That is, the probability of V = d decreases monotonically as we add evidence items). What is P(V = d—V1 = d1,...,Vk = dk)?

Question 3: Is your Bayes Network Fair? **(worth 10/100 marks)**

Let's say that we want to use the network specified in `medicalDiagnosis.py` to predict if someone has Hyperlipidemia. To do this, we'll predict a person has Hyperlipidemia whenever $P(Hyperlipidemia = YES|Evidence) > 0.5$.

Before we act on any of our predictions, though, we'd like to know that they are not subject to gender bias! More specifically, we'd like to know that they are equally fair to both men and women (i.e. not less accurate for one group than another). But this could mean different things!

1. It could mean that our predictions are well **'Separated'** from gender, meaning

   $P(Prediction = YES|Hyperlipidemia = YES, Gender = Female) =$
   $P(Prediction = YES|Hyperlipidemia = YES)$

2. Alternately, it could mean that our predictions are **'Sufficient'** (and gender tells us nothing more than our label about the presence of disease), meaning

   $P(Hyperlipidemia = YES|Prediction = YES, Gender = Female) =$
   $P(Hyperlipidemia = YES|Prediction = YES)$

Show that, in general, it's impossible to build a classifier that is both Sufficient and Separated at the same time. More specifically, assume C is a classification, Y is a label representing 'ground truth', A is some 'protected attribute' (e.g. gender), and that all events in the joint distribution of (A,C,Y) have positive probability. Define **Separation** as meaning A is independent of C given Y, and **Sufficiency** as meaning A is independent of Y given C. Show that enforcing both **Separation and Sufficiency** at the same time implies A is independent of (Y, C).

HAVE FUN and GOOD LUCK!