Programing Assignment 2020

DECEMBER 26

Course: Machine Intelligence
Name: Ahmed Mohamed Ahmed Hassan
ID:1162057

Informations

Name:

Ahmed Mohamed Ahmed Hassan

Section:

Credit

ID:

1162057

Problem Number:

Problem 1

BayesNet.py

**In this file I construct the Class "BayesNet" to Construct my Bayesian network,
And define some methods that will help me to construct the Network.**

First initialize the Network with an empty array of random Variables, and a set to get each random variable by its name.

Then use the method add to add random variable to the network, given it the random variable name, its parents and its conditional probability table (CPT)

Then, I made another class "Variable" to make for each random variable an object that has its name, parents, CPT and its domain (I need it its domain when I want to know the variables it takes) like in our case the domain for the random variable 'AB' is { tie, win, lose}

And Class "CPTable" I made it to build the CPT table, by passing all the expected events and the probabilities of each random variable upon defining each random variable. The events when its parents only change since each rando variable depends only on its parents.

Exact.py

Containing the function that calculates the exact inference after Variable elimination For the second Case Only (3 teams), but you can enter any results for the first two matches.

I applied this equation but for any Match Results (for the first 2 matches.

$$\alpha' \times \sum_{a \in AQ} \sum_{b \in BQ} P(AB = Win \mid a,\ b) \times \sum_{c \in CQ} P(BC \mid b, c) \times P(AC = Tie \mid a,\ c)$$

MCMC.py

By Using the pseudo code our book, I managed to implement the Gibbs function. In the Gibbs function, first I get the domain of the random variable I will be calculating its probability for example if we want to calculate P(AB | anything), so the domain will be tie, lose and win and initialize each one by zero. Then I got all the remaining random variables that doesn't exist in the Evidence. And give each random variable a random value using random.choice and giving it all the values, it can take. Then using markov blanket sample which return a sample from P(X | mb) where mb denotes that the variables in the Markov blanket of X take their values from event e (which must assign a value to each). The Markov blanket of X is X's parents, children, and children's parents. Then using probability distribution from the counts I got after N iterations.

NeededFunctions.py

This file contains some functions I will be using widely in other files, each function I make a comment above it explaining what it should do.

main.py

Take from the user the inputs (Number of teams, the first matches result but the last, the algorithm Exact or MCMC).

How to run the code:

Run the main, enter the Number of teams then enter the team results (win,tie,lose) then choose the algorithm you want to use (press 0 for Exact, 1 for MCMC). Exact only work for 3 teams as required in questions 3 (I only added that the user can specify the results of the first 2 matches) but MCMC is generic for any number of teams.

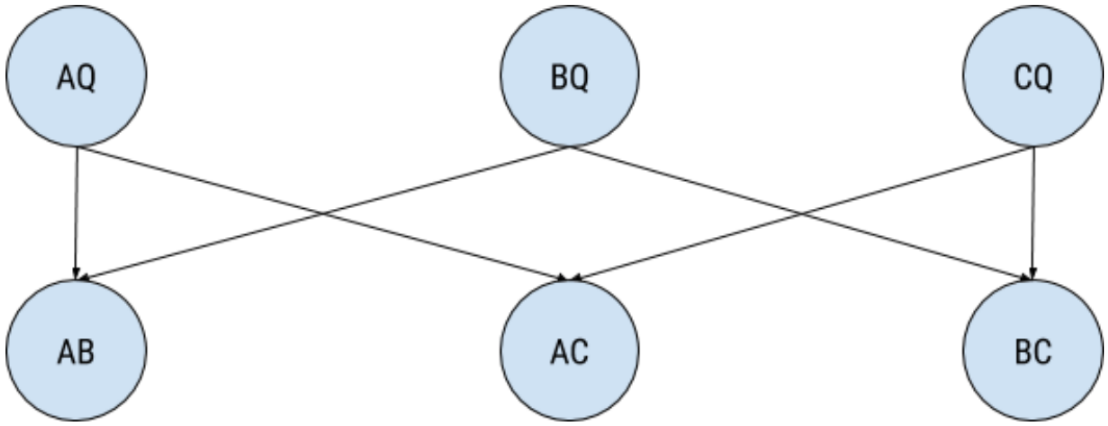
Dependencies required:

- **Python**
- **NumPy**
- **Random**

Textual answers:

a) **The classes are Team, with instances A, B, and C, and Match, with instances AB, BC, and CA.**
   Each team has a quality Q and each match has two teams and the result of the match. The team names for each match are fixed in advance. The prior over quality could be uniform each ¼ and the probability of a win for team 1 should increase with Q(T eam1) − Q(T eam2).

b) I constructed the Bayesian Network in the code and here how it will looks:



While AQ,BQ,CQ is the teams quality, and AB,AC,BC is the outcome or the result of the matches and each is random variable and the arrows represent dependencies, for example AQ doesn't depend on any but AB depends on AQ and BQ

c) I calculated by code (Exact.py) and the answers would be
Win 0.34228987037362896
Lose 0.38694938119537803
Tie 0.270760748430993
Which means that C would probably win.

d) The inference cost will be O(2n) because all the team qualities become coupled.

e) I implemented the MCMC in the code (MCMC.py), and here some results from running the code some times

{'win': 0.416, 'tie': 0.1925, 'lose': 0.3915}
{'win': 0.378875, 'lose': 0.423625, 'tie': 0.1975}

I used the pseudo code of GIBBS in the book. The algorithm converges very quickly but it does not scale very well. The scale problem comes from the Z variable. The algorithm requires to sample values for all variables that do not belong to the evidence plus the query variable

How I wrote the code:

First I read some of chapter 14 in the book to understand the concepts behind Bayesian network, the difference between exact and approx. inference and how to calculate each and which algorithms I need. For example to solve an exact inference we can use variable elimination which I used to write the code after calculating to the last equation then implemented it in Exact.py. then used the pseudo code in the book of Gibbs to implement the MCMC.

Results and Calculations:

Exact:

Win 0.34228987037362896
Lose 0.38694938119537803
Tie 0.270760748430993

MCMC:

{'win': 0.416, 'tie': 0.1925, 'lose': 0.3915} // N=2000
{'win': 0.378875, 'lose': 0.423625, 'tie': 0.1975} // N=2000

And with each run we get different values. Since it's a random process after all and explained why it did so in sub question e

Assumptions:

I assumed the order of the matches, so for example if we have A,B,C. so the order will be as follows. A plays all its matches first then B (AB,AC,BC). And that we always calculate the result of the last match. And the qualities varies only from 0 to 3 as required in the question.