Artificial Intelligence: PA 3 (v1.0)

CSPs

Due May 3$^{\text{rd}}$, 2020

Overview

The learning objectives for this assignment are:

- implement direct and rejection sampling methods to approximate probabilities
- utilize Bayesian networks to compute exact probabilities using conditional probability tables (CPT) and marginalization

Introduction

Using Bayes Nets (BNs) to represent events in a compact manner is common in certain AI applications. BN scale because of 2 important characteristics:

- The full joint distribution takes $\mathcal{O}(d^n)$ values (where d is the number of values of the random variable with the largest domain and n is the number of variables/nodes). BNs take $\mathcal{O}(n * d^k)$, where n and d are defined the same and k is the maximum number of parents a single node has in the network.
- a byproduct of having fewer probabilities is that practitioners have fewer probabilities to collect/estimate. This makes collecting these statistics tractable.

Friendly structures, known as polytrees, allow for exact computations to be made in polynomial time. BN in general require exponential time in order to compute entries from the full joint distribution. In this assignment, you will implement two methods to compute the approximate distribution, $\hat{\mathcal{P}}$, using a BN. These two methods are:

- Direct Sampling
- Rejection Sampling

You are provided some template code and some examples of the expected results.

1 Overview of Template Code

This section outlines the supplied code template.

1.1 Incomplete CPT class (requires directSample to be written)

A class for representing condition probability tables (CPT), the location of where probabilities are stored for BN. The CPT stores the following values:

varName	The name of the variable, usually a single character (matching the BN shown in Figure 14.23 in the textbook).
priorVars	A Python list where each element is the variable name (single character) on which this variable is conditioned. The ordering of this list MUST match the ordering of the variables in the probTable variable (the next variable).
probTable	a Python dictionary storing the probabilities as the value. The key is a tuple of booleans, which should be the same length as the list in priorVars. Each tuple element matches a setting of true/false for the respected priorVar (lists are in the same order).

The class method *directSample* needs to be completed. It should reference the CPT's probability table (probTable) for the variable being sampled using the values in the *observed* parameter. *observed* is a Python dictionary, where the key is the variable name (single character) and the value is True or False. *directSample* then needs to generate a random number between 0 and 1, which is then transformed into a True or False value by using the probability in the *probTable* class variable. This observation is then added to a COPY of the dictionary. This new dictionary should be returned to the caller.

1.2 An Example Bayes Network

The CPT class is used to construct the Bayes net that is shown in Figure 14.23 of the textbook. A list named *orderedVars* shows a topological ordering for the nodes in this BN. This list can be traversed in order to compute probabilities. For example, sample a value for the first variable, which is appended to the dictionary by using a call to *directSample*. Then, the next variable in *orderedVars* is sampled (sending the dictionary as the observed variables). Thus, a complete traversal of this list produces a complete sample (a value for each random variable in the BN).

2 Direct Sampling

You need to construct a Python method named *priorSample*, which is called by *directSampling*. The method *directSampling* is provided in the example code. This method accepts the following parameters:

orderedVars	topological ordering of the variables (provided in the code for this BN in the main method)
eventToCalc	a Python dictionary, where each key in the dictionary is a variable and the corresponding value is a boolean (True/False). An example of setting an eventToCalc is shown in Figure 2.
nbrOfSamples	an integer representing the number of samples to perform.

This method must return a single value:

runningEstimate	a python list showing the progression of the probability estimate (see Section 4. The last entry in this list would be the final probability estimate.

3 Rejection Sampling

You need to construct a Python method named *rejectSampling* (as prototyped in the example code). This method will approximate $\mathcal{P}(X|evidenceVars)$. Pseudo-code for this method is shown in the textbook (Figure 14.14) and makes use of priorSample (a method that *directSampling* and *rejectSampling* could share). The method must accept the following parameters:

orderedVars	topological ordering of the variables (provided in the code for this BN in the main method)
X	a Python dictionary, where each key in the dictionary is a variable and the corresponding value is a boolean (True/False). An example of setting X is within the main method (see queryVar).
evidenceVars	a Python dictionary, where each key in the dictionary is a variable and the corresponding value is a boolean (True/False). An example of setting X is within the main method (see queryVar).
nbrOfSamples	an integer representing the number of samples to perform.

This method must return a tuple consisting of the following values:

rejectRatio	the proportion (express as a number between 0.0 and 1.0) of generated samples that were rejected because they contradicted the evidence variables.
runningEstimate	a python list showing the progression of the probability estimate (see Section 4. The last entry in this list would be the final probability estimate.

4 Plotting the Running Estimate

The methods of *directSampling* and *rejectionSampling* only provide an estimate of the exact probabilities. As the number of samples is increased, the estimate improves. The methods you will be implementing (*directSampling* and *rejectionSampling*) must keep a running estimate of the probability. This can be accomplished by appending to a Python list the estimate based on the samples generated (so far). For example, imagine that 7 samples were generated for a single binary variable (True/False) as shown in Figure 1. The bottom row in the table shows what would be the corresponding entries in the python list (as it shows the probability estimate at that point if the method were to stop).

Samples	F	T	T	T	F	T	T
Running Estimate of P(x = True)	$\frac{0}{1}$	$\frac{1}{2}$	$\frac{2}{3}$	$\frac{3}{4}$	$\frac{3}{5}$	$\frac{4}{6}$	$\frac{5}{7}$

Figure 1: An example of a running estimate of estimating the probability of a simple binary random variable.

A method named *plotRunningEstimate* has been included in the supplied code. This code accepts the running estimate as a Python list, a filename to generate the plot, the variable(s) that are being queried (as a dictionary) and an optional evidence variable (also encoded as a dictionary). Below is an example of calling it and the graph this produced.

```
eventToCalc = {'B': True, 'M': False, 'I': True, 'G': True, 'J': True}

runningEstimate_B_notM_I_G_J = directSampling(orderedVars, eventToCalc,1000)

plotRunningEstimate(runningEstimate_B_notM_I_G_J, 'runningEstimate_B_notM_I_G_J.pdf',
                    eventToCalc)
```

Figure 2: An example of calling directSampling and the provided plotting code.

Using the BN in the book, we can see that the exact probability is: $0.9 \times 0.9 \times 0.5 \times 0.8 \times 0.9 = 0.2916$. The plot shows how the direct sampling method continues to refine its estimate as the number of samples increases.
This code segment also shows how to create a dictionary representing an event for which you want to estimate the probability. An evidence dictionary (used for rejection sampling) can be created in the same way (an example is in the supplied template code).

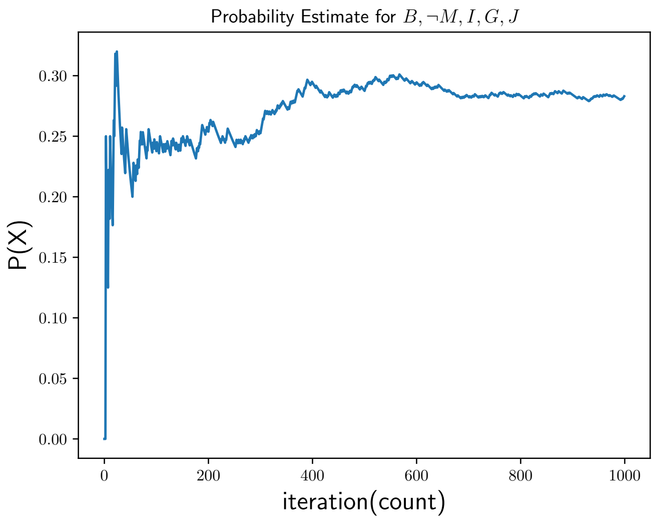

Figure 3: A running estimate of the event $B, \neg M, I, G, J$. The exact computation for this probability is 0.2916.

Tasks

You must augment the provided source code by written/completing the following methods:

- complete the directSample method within the CPT class
- complete the directSampling method
- complete the rejectionSampling method
- Estimate the probabilities below using the specified method and produce a PDF with the results of the runningEstimate plot

Run your code to compute an estimate of the following probabilities and **include the running estimate plot** for each in your PDF. Your PDF must ALSO show the computation of the exact probabilities (I provide an example for both direct sampling and one with conditional probabilities requiring the use of rejection sampling).

1. $\mathcal{P}(b, \neg m, i, g, j)$. This is the example shown in Figure 3.
2. $\mathcal{P}(b)$. Make sure to include work showing the exact computation in the PDF.
3. $\mathcal{P}(b|i)$. This is the example that is setup within *main* in the source code. Include your rejection percentage in your analysis. The computation of the exact probability for this problem are shown below (so, your approximation should "approach" this value):

$\mathcal{P}(b\|i) = \alpha\mathcal{P}(B,i) = \alpha\sum_y \mathcal{P}(B,i,y)$	marginalization, or summing out (see eq 13.9 and section 14.4.1 in textbook). The set y could represent all the other variables in the network (m, g, j). Because of the conditional independence in the network, we only require m.
$\alpha\sum_m \mathcal{P}(B,i,m)$	
$\alpha\langle\mathcal{P}(b,i,m) + \mathcal{P}(b,i,\neg m), \mathcal{P}(\neg b,i,m) + \mathcal{P}(\neg b,i,\neg m)\rangle$	This is the expansion for this summuation. Since we are going to need α to normalize the distribution, we will need to compute b and $\neg b$ (thus, the capital B that appears in the equations).
$\mathcal{P}(b,i,m) = \mathcal{P}(b)\mathcal{P}(m)\mathcal{P}(i\|b,m)$	The BN allows us to rewrite this smaller subset of the joint distribution as the product of these conditional distributions using the CPTs. Similar equations are used when to compute the other terms.
$\alpha\langle\mathcal{P}(b,i,m) + \mathcal{P}(b,i,\neg m), \mathcal{P}(\neg b,i,m) + \mathcal{P}(\neg b,i,\neg m)\rangle$	
$\alpha\langle 0.9 \times 0.1 \times 0.9 + 0.9 \times 0.9 \times 0.5,$	
$0.1 \times 0.1 \times 0.5 + 0.1 \times 0.9 \times 0.1\rangle$	Plugging in the probability from the BN
$\alpha\langle 0.486, 0.014\rangle$	
$\langle 0.972, 0.028\rangle$	Normalize distribution

4. $\mathcal{P}(m|j, \neg b)$ Include your rejection percentage in your analysis. Make sure to include work showing the exact computation in the PDF (this requires a lot of steps, but will best prepare you for the exam, use the steps above as a guide).

Collaboration Policy and Package Usage

For this project, you are only allow to use the following python packages contained in the template file plus time, numpy and matplotlib if you choose. If you need more than these packages, please see me prior to using them in your assignment.

If you need more than these packages, please **request** them from me (as maybe I omitted a few here).

For this assignment, you may work in a group of 2 (and with prior approval, a group of 3). Helping colleagues (other groups) to find a bug or to help them understand some concepts is OK, sharing of code or obtaining code that you did not write is prohibited. You should be able to explain all of your code to me, as failure to do so will be an indicating that you over-collaborated. I will be using plagiarism detection tools in this class.

If you have questions about this policy, please see me.

Deliverable

The following deliverables must be placed in a single zip file named **PA3Canvas.zip** and uploaded to canvas. This zip file must include the following:

1. Your python code in a file named *cs444_PA3.py*. Make sure that your name (or names if working in a group) appears at the top of the file.
2. Sample output from your program (you can simply run your program and redirect the output to a file). Make sure that somewhere in your submission (the PDF report is a fine place to put this) you show how to invoke your program to generate this sample output.
3. A PDF showing the results of estimating the probabilities outlined in the task section and the computations showing the exact probabilities and the difference between your estimate and the exact value. You MUST include the plot of the running estimate for each of the 4 computations outlined in the task section.

Rubrics

The following is the breakdown how this assignment will be graded:

Item	Description	Points
Readability/Style	Program is commented, following naming style for either Python/Java. Points will be deducted for poor variable names or unreadable code	5
directSample method	implement	20
directSampling method	implement	25
rejectionSampling method	implement	25
PDF report	probability estimates and exact calculations in PDF. Running your python script should generate the PDFs for the plots required and outlined in the Tasks section of this assignment	30