CSC442 Intro to AI
Project 3: Uncertain Inference

In this project you will get some experience with uncertain inference by implementing some of the algorithms for it from the textbook and evaluating your results. We will focus on Bayesian networks, since they are popular, well-understood, and well-explained in the textbook. They are also the basis for many other formalisms used in AI for dealing with uncertain knowledge.

Background

Recall that a Bayesian network is directed acyclic graph whose vertices are the random variables $\{X\} \cup \mathbf{E} \cup \mathbf{Y}$, where

- X is the query variable
- $\mathbf{E}$ are the evidence variables
- $\mathbf{e}$ are the observed values for the evidence variables
- $\mathbf{Y}$ are the unobserved (or hidden) variables

The inference problem for Bayesian Networks is to calculate $\mathbf{P}(X \mid \mathbf{e})$, that is, the conditional distribution of the query variable given the evidence (observed values of the evidence variables). In other words, compute the probability of each possible value of the query variable, given the evidence.

In general, we have that:

$$\mathbf{P}(X \mid \mathbf{e}) = \alpha \, \mathbf{P}(X, \mathbf{e}) = \alpha \sum_{\mathbf{y}} \mathbf{P}(X, \mathbf{e}, \mathbf{y}) \qquad \text{(AIMA Eq. 13.9)}$$

And for a Bayesian network you can factor that full joint distribution into a product of the conditional probabilities stored at the nodes of the network:

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i \mid \textit{parents}(X_i)) \qquad \text{(AIMA Eq. 14.2)}$$

Therefore:

$$\mathbf{P}(X \mid \mathbf{e}) = \alpha \sum_{\mathbf{y}} \prod_{i=1}^{n} P(x_i \mid \textit{parents}(X_i))$$

Or in words: "a query can be answered from a Bayesian Network by computing sums of products of conditional probabilities from the network" (AIMA, page 523). These equations are the basis for the various inference algorithms for Bayesian networks.

Part I: Exact Inference

For the first part of the project, you must implement the "inference by enumeration" algorithm described in AIMA Section 14.4. Pseudo-code for the algorithm is given in Figure 14.9. Section 14.4.2 suggests some speedups for you to consider.

Your implementation must be able to handle different problems and queries. For exact inference, your program must accept the following arguments on the command-line:

- The filename of the XMLBIF encoding of the Bayesian network. You may assume that these filenames will end in "`.xml`".
- The name of the query variable, matching one of the variables defined in the file.
- The names and values of evidence variables, again using names and domain values as defined in the file.

So for example, if this was a Python program, you might have the following to invoke it on the alarm example:

```
python mybninferencer.py aima-alarm.xml B J true M true
```

That is, load the network from the XMLBIF file `aima-alarm.xml`, the query variable is B, and the evidence variables are J with value $true$ and M also with value $true$.

Similarly, if you choose to use Java, then the "wet grass" example from the book (also included with my code) might be:

```
java MyBNInferencer aima-wet-grass.xml R S true
```

The network is in XMLBIF file `aima-wet-grass.xml`, the query variable is R (for $Rain$) and the evidence is that S ($Sprinkler$) has value $true$.

The output of your program should be the posterior distribution of the query variable given the evidence. That is, print out the probability of each possible value of the query variable given the evidence.

Your writeup and/or README must make it very clear how to run your program and specify these parameters. If you cannot make them work as described above, you should check with the TAs before the deadline and explain the situation if for some reason it can't be resolved.

Part II: Approximate Inference

For the second part of the project, you need to implement at least one of the *approximate inference* techniques for Bayesian networks as discussed in class (and in the textbook) The algorithms you may choose are:

1. Rejection sampling
2. Likelihood weighting
3. Gibbs sampling

For running these approximate inferencers, you need to specify the number of samples to be used for the approximation. This should be the first parameter to your program. For example:

```
% java MyBNApproxInferencer 1000 aima-alarm.xml B J true M true
```

This specifies 1000 samples be used for the run. The distribution of the random variable B will be printed at the end of the run. If you need additional parameters, document their use carefully.

Part III: Evaluation

You will be provided with three XML format Bayesian networks:

1. aima-alarm.xml
2. aima-wet-grass.xml
3. dog-problem.xml

You should use these sample networks to test your code. The Alarm and WetGrass networks may be particularly helpful, as there are worked examples in the textbook and in the slides. In your writeup, you must quantify how many samples are necessary for your sampling algorithm to be within 1% of the exact value. Note that the number of samples will depend on the specific topology of the network; choose your own queries, evidence, and network and record your choices in your writeup.

Writeup

For the algorithms that you implement, explain your design and implementation briefly, and evaluate the results as previously mentioned.

Additional general requirements for writeups are the same as for previous projects, as described below.

Rubric

- 25% – XMLBIF parsing and network representation.
- 25% – Exact Inference
- 25% – Sampling
- 25% – Writeup and Experimental Work

Submission

Upload a zipfile containing your source code and your writeup to blackboard by November 24th at 1159PM for 5% bonus credit. The ultimate deadline is December 1st at 1159pm. You may work in a group of up to three students (all currently taking CSC 442). If you do so, only one student should submit the project but you must also include a contributors section in the writeup which states your team members' names and contributions to the project.