CSC442 Artificial Intelligence
Project 3: Probability

November 9, 2021

In this project you will get some experience with uncertain inference by implementing some of the algorithms discussed in class and evaluating your results. You will also design your own Bayesian Network for causal and diagnostic reasoning. We focus on Bayesian networks because 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. Your writeup will be graded more heavily for this project than your solution code, so please read the complete instructions carefully before beginning.

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:

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

Therefore:

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

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: Designing a Network

Design a Bayesian network for a task which is relevant to your interests. You can design a network by first choosing a set of random variables, then identifying the direct causal relations, and finally breaking down the conditionals and estimating probabilities by hand.

I recommend sticking with low-arity discrete random variables (booleans are easiest.) The algorithms we've discussed also work with continuous variables, but they require some subtle changes, so I advise avoiding anything with infinite or continuous outcomes. If you'd like to model these kinds of values, you can try simple discretizations – e.g., low vs medium vs high, normal vs abnormal, and so on.

Your network should involve at least five random variables if you are working alone, and at least eight random variables if you are working in a group. Please put some thought into this as you will be building a system to answer arbitrary queries over this data – what would be interesting to you?

It will probably be easiest to begin by drawing your network out by hand. Ultimately, I want to see your network as well, so when you go to include it in

the writeup you may want to layout your network using word processing or design software. Once you have designed your network, choose three queries to explore for the rest of the project:

- A causal query – where the query is an "effect" and the evidence are "causes".
- A diagnostic query – where the query is a "cause" and the evidence are "effects".
- A sanity check – any single conditional directly expressed in a table.

Part II: Exact Inference

For the first part of the project, you must implement the "inference by enumeration" algorithm described in AIMA Section 13.3. Pseudo-code for the algorithm is given in the textbook and we have discussed this concept in class. This algorithm has the advantage that it is very simple – at least it is, once you decide how to represent your conditional distributions and design a mechanism to enumerate variable assignments.

In order to streamline this project and focus on the algorithms, you are NOT required to implement any file input or output operations; however, you should know that a common representation for Bayesian networks is called XMLBIF. More information can be found here: `http://www.cs.cmu.edu/afs/cs/user/fgcozman/www/Research/InterchangeFormat/`. Instead, you may HARDCODE the necessary networks for this project. As an example of such a format, you are supplied with the AIMA-ALARM example from the textbook in XMLBIF format.

Sanity Checks

You should ensure that your inference algorithm is correct by performing a few tests. An advantage of working with a simple network (such as the AIMA Alarm) is that you can find exact solutions by hand and verify them, so go ahead and code up that network as well.

You should query your program to determine Pr(John Calls — Earthquake), Pr(Burglary — John Calls), and Pr(Mary calls — Alarm). These three queries are examples of causal reasoning (reasoning along the lines of

direct influence (i.e., from causes to effects)), diagnostic reasoning (reasoning from effects to causes), and a pure sanity check – you should be able to obtain the original conditional probability values by using the "summing out" method. Once your programs outputs match the expected outputs, you can continue to the second part.

Part III: Approximate Inference

A key objective of this project is to understand the nature of approximate inference mechanisms – i.e., sampling algorithms. Therefore, you should implement both **rejection sampling** and **Gibbs sampling**. We have discussed both algorithms in class and they are presented in the textbook as well. The nature of sampling algorithms always leads to the same question: *How many samples do I need?* You're going to answer that in this project!

Estimating Convergence and Accuracy

If you implement them correctly, both algorithms should converge to the true probabilities "in the long run". A general rule of thumb is that an extra digit of accuracy may require an extra factor of ten in the number of samples, but of course, this depends heavily on the network itself as well as which sampling algorithm you use. With low numbers of samples, the output of your algorithm will likely jump around wildly; however, as the number of samples increases, it should stabilize to a given value – the true probability. This leads to a few natural questions:

- How could you define stability?
- How many samples are needed for rejection sampling to become stable?
- How many samples are needed for Gibbs sampling to become stable?
- Do these values depend on the structure of the network or the query itself?
- Is one algorithm "better" than the other?

For this problem, you should construct one graph for each of your three queries using the network you designed. The x-axis should be the number of

samples and the y-axis should have two lines – one for the estimate obtained by rejection sampling, and another for the estimate obtained by Gibbs sampling. Once you have explored the data, return to the questions above and answer them.

Estimating Accuracy

Finally, it is time to compare the accuracy of your sampling algorithms to the exact values computed by brute-force enumeration. Again using the three queries you chose and the network you designed, generate a new set of plots showing the difference between the sampler estimate and the exact value for the queries. Create one more set of three plots, again with the x-axis being the number of samples computed, but where the y-axis is the **relative error** between the sampler and the exact value. The relative error can be calculated as:

$$|sampledValue - exactValue|/exactValue$$

Run your sampling algorithm until both lines approach zero relative error.

Submission Requirements

You should collect your results into a PDF writeup that includes the following:

1. Your custom network, and a description of the random variables.
2. Your implementation must support ARBITRARY queries and evidence values. You have to decide how to represent this and explain it in your writeup.
3. Your custom queries (labeled as causal, diagnostic, and sanity check)
4. Your three graphs (one per query) estimating convergence time for the sampling algorithms.
5. Your three graphs (one per query) estimating relative error for the sampling algorithms.
6. Your answers to each of these questions:

(a) How did you define stability?

(b) How many samples are needed for rejection sampling to become stable? What about to become accurate?

(c) How many samples are needed for Gibbs sampling to become stable? What about to become accurate?

(d) Do these values seem to depend on the structure of the network or the query itself or both?

(e) Is one algorithm "better" than the other?

7. If you worked as part of a group, you must also include a section detailing the group members and how you collaborated (i.e., who did what.)

You should submit a zip file containing your source code and your writeup by Sunday night November 28th for full credit. Late work will be accepted at a penalty of 3% per day.