CSC242: Intro to AI
Project 3: Uncertainty Modeling

March 18, 2025

1 Overview

Making decisions in spite of uncertainty is one of the classic problems of AI.

In this project, you will build a tool for modeling uncertainty. Specifically, your program will be provided with a series of conditional probability tables defining a *Bayesian network*, and you will be expected to implement an API for querying the distribution. This assignment will be limited in scope to discrete probability distributions.

On a deeper level, the goals of this assignment are to build your intuitions for computational modeling of probabilistic information. Therefore you are required to implement exact inference, rejection sampling, and Gibbs sampling. You are also required to conduct your own experiments to search for relationships between degree of computation and the accuracy and reliability of inference.

You will report your results in a PDF writeup. Your grade will be determined by both automated testing and human TA review.

2 Project Requirements

2.1 Representing Probability Distributions

Each network will be represented in plain text, specified as a variable descriptor segment followed by a conditional probability table (CPT) descriptor segment.

The variable descriptor segment will give the number of variables, then a line-by-line description of each variable. The variables will be given by their name, then the elements of their domains specified. This is somewhat redundant for boolean valued variables, but flexibly handles any domain size. All values are assumed to be string constants.

Representing probability distributions can be challenging. There are typically exponentially many entries, and they must be easily indexed to evaluate during inference. In this assignment we are adopting a simplifying, common assumption: all distributions will be given in *lexicographic* order, with ordering among values matching the order in which they are specified with the variables.

The CPT section will begin with the number of CPTs, then each CPT specified with a blank line between them. The CPTs will always be assumed to be of the form $P(Y|X_1, X_2, \ldots, X_M)$, meaning a distribution over a single discrete variable, conditioned on zero, one, or more additional discrete variables.

Each line of the CPT will give a distribution over possible values of Y, in the order specified in Y's variable definition. There will be one line for each combination of the parent variables, with the combinations evaluated in lexicographic order. Extra whitespace will be ignored. The "#" symbol will be used as a comment symbol, and any remaining text on a line after "#" should be ignored.

2.2 Inference API

Your program should implement a basic REPL (read-evaluate-print-loop) and at least the following commands:

1. `load NETWORK` - load a network from the specified file

2. `xquery Q | X1=x1 .. XN=xn` - print the distribution for $P(Q|X1 = x1, \ldots XN = xn)$, evaluted via exact methods. (I.e., exhaustive summation.)

3. `rquery Q | X1=x1 .. XN=xn` - print the distribution for $P(Q|X1 = x1, \ldots XN = xn)$, evaluated via rejection sampling.

4. `gquery Q | X1=x1 .. XN=xn` - print the distribution for $P(Q|X1 = x1, \ldots XN = xn)$, evaluated via gibbs sampling.

5. `quit` - terminate the program.

You must through experimental work develop reasonable numbers of samples to collect via the `rquery` and `gquery` methods. The output of each of the three queries above should always be exactly one line consisting of the values of the corresponding distribution in the order in which the values are specified in the variable declaration.

2.3 Experiments

How do you know your rejection sampler is working? Experiment with number of samples on different queries for the sample network, (or after validating your exact method), observe the effect of number of samples on accuracy. Observe the effect of kinds of query on number of iterations to achieve usable samples. Compare the total computation necessary to reliably obtain a "close" answer vs exact methods. Research and use the concept of "Kullback-Leibler Divergence" to determine how far apart the sampled distributions are from the true distributions.

How many samples do you need to obtain via gibbs sampling to get "close", like with rejection sampling? How does the amount of computation compare? If you allow a period of "burn-in", can you get more "robust" answers?

What effects do network architecture or extreme numerical values have on the relationships you investigated above? (E.g., chains, vs trees, vs fully connected or disconnected graphs, and similar structural variants).

To be precise, you should:

1. Explore and report the number of samples required to accurately sample with rejection sampling.

2. Explore and report the number of samples required to accurately sample with Gibbs sampling, and whether or not burn-in (or when) is helpful.

3. Identify any special networks or specific queries which have interesting properties in relation to these investigations.

2.4 Writeup

Convey your answers to the experiments portion in a writeup. You should include well-crafted plots and figures as appropriate. You should discuss broadly how effective and reliable you found the sampling algorithms, and give practical advice for using them efficiently.

If you use AI for this assignment, you must disclose your use. You are discouraged from using AI to generate solutions to the core algorithms - implementing them and debugging them yourself will be an important learning opportunity.

3 Submission Instructions

You may use Java, Python, C, or C++ for this assignment. In any case, you should write your solution as a single file named according to your chosen language: `Main.java`, `main.py`, `main.c`, or `main.cpp`. No other languages are supported for this assignment. Do not use any package declarations or modules.

Upload the source code of your solution and a PDF export of your report directly to gradescope. Ensure your name and UR email are clearly indicated at the top of your source file (e.g., as a comment), and as the top of your report. Please also include a short README file describing your code and how to use any additional features you choose to implement. Group submissions should be submitted by one student who is responsible for tagging all group members in the gradescope submission. Group submissions should include the names and emails of ALL members in both the program source code and at the beginning of the written report. The report must also contain a brief section indicating the contributions of each team member and a description of your collaboration style. Maximum of 4 students per group.

Submissions will be accepted until 11:59PM, Sunday April 6th. For maximum preparation for the quiz, you are strongly encouraged to complete this by Sunday March 30th.

3.1 Grading

- 70% Program (accurate inference, well-designed, README)

- 30% Report (good experimental work, results complete, clearly and precisely conveyed)

4 Example Network and Program Output

This is an example network (suppose it is saved as "network.bn"):

```
3 # Number of variables
A t f
B t f
C t f

3 # Number of CPTs
A
0.7 0.3 # p(A)

B
0.5 0.5 # p(B)

C A B    # CPT for P(C | A, B)
0.5 0.5  # P(C | A=t, B=t)
0.7 0.3  # P(C | A=t, B=f)
0.8 0.2  # P(C | A=f, B=t)
0.1 0.9  # p(C | A=f, B=f)
```

Here is an example program run using the above network:

```
load network.bn
xquery A
0.7000 0.3000
rquery A
0.6924 0.3076
gquery A
0.7036 0.2964
xquery C
0.5550 0.4450
rquery C
0.5534 0.4466
gquery C
0.5702 0.4298
quit
```

Note that the non-numerical lines represent user input, and the numerical lines represent program output. You are also free to print *any additional information you like*, however, to facilitate autograding, you must print any extra information to stderr instead of stdout.