Bayesian Networks

Jordan Turley

For this project, we were tasked with implementing a Bayesian network for a burglary, earthquake, alarm, John calls, and Mary calls.

For my Bayesian network, I had a Node, Probability, NodeValue, and BNet class. The NodeValue class stores the name of a node and a value (true or false) for that node. The Probability class stores a list of NodeValue objects and a probability as a number. For example, for the Alarm node, when B is true and E is true, the probability is 0.97. The Probability class would store that when B is true and E is true, the probability is 0.97. The Node class stored the name of a node and the various probabilities for it, as a list of Probability objects. For a node that has no parents, the list of probabilities only has one probability, and that probability does not have any nodes that it depends on. For example, for the Burglary node, it would store the name, “B”, and the list of probabilities, containing only one Probability object, which has an empty list of NodeValues and the probability. For the Alarm node, it stores the name, “A”, and a list of four probability objects, for each combination of B and E true or false. The node class also stores this node’s children as a list of nodes. The BNet class brings everything together, generating the Bayesian network, which is hard coded, getting the user’s input from the command line arguments, and calculating the probability.

For my implementation, I used rejection sampling. I generated a prior sample, then compared this to the evidence and query variables. I followed the guide you gave us very closely, and implemented the rejectionSample method in the BNet class straight from the pseudo code we were given. I ran rejection sampling for one million trials before calculating the estimated probability.

For my implementation, I used Java. I did not use any third-party packages or libraries.

In this project, I learned a lot. Before I read through the prompt closely, I didn't know what I was doing or how I was going to do it, but after I read through it and the directions for rejection sampling, it was not too bad. The rejection sample method itself was much simpler than I expected. In this project, I learned how using a Bayesian network can be very helpful in calculating probabilities for various events. While it does not give an exact probability for an event, to test, I tried asking for one of the probabilities that we already knew, like a burglary occurring, and the answer I got was always extremely close. It could be even more accurate using more trials; initially I used 100,000, but this wasn't as accurate as I would like, so I increased it to 1,000,000, and it made it much more accurate. This implementation is not as expandable as it should be or as I would like, but I don't think it would be too hard to switch from hard coding the network to reading it in from a text file or something. If you had a good way to represent it in text, or if there is a standard way to represent a Bayesian network in text, it might not be too hard to read this in instead of hard coding the network.

I attached my output with the code in the .zip file.

For my Bayesian network on paper, I tried to model it so that if one of them didn't put in time, they were more likely to be hacked, but because they are on the same network, the other was also more likely to be hacked, but not as much more likely. Same with having their files show up in WikiLeaks. I modeled it so that if one was hacked, both of their files were at risk, the one that was hacked was more at risk than the other.