Pruning Lab

In this lab, you will implement pruning based on d-separability

Pruning

Expand on the idea of d-separation to build code that will take as input:

- a Bayesian network
- a set of query variables
- a set of evidence variables

And return a sub-network which does not contain any variables not needed for the query.

Is is not necessary for you to build a parser to read in the network. Just encode the structure in your code. If you really insist on parsing a file format you might look at any of the many Bayesian network file formats. You may also just use the dot format (see [1]). You may use code from the internet for your input and output routines, as long as you abide by the license that covers that code and that the license does not affect my ability to grade the code. Of course getting d-separation code from the net is not allowed, since that is the point of the exercise.

Please test your code on a variety of networks of your own making, or ones you find on the web. Be sure that you test your code on networks that include the following:

- Influence that flows forward (i.e. evidence->node a->node b->Query node)
- Influence that flows backward (i.e. evidence<-node a<-node b<-Query node)
- Influence that flows through "v" structures (i.e. node a->evidence<-Query node)
- Influence that flows through "y" structures (i.e. node a->node b<-Query node and node b->evidence)
- Influence that flows through "inverted-v" structures (i.e. evidence<-node a->Query node)
- Mixed sequences of these structures
- Multiple Evidence nodes
- Multiple Query nodes
- Multiple paths from the Evidence to the Query, some active, some not.

You are **encouraged** to share interesting structures on the discussion page, but don't share the answers. You are likewise encouraged to use these shared structures when you turn in your assignment. Please identify which structures you contributed to the class and which you obtained from others.

Please turn in the following:

- A short (less than one page) description of how your algorithm works.
- Sample runs some of which should have significantly more than 10 nodes. Note that you have the obligation to show me that your approach works correctly, you will need some small runs to convince me.
- Give a summary of a run with an 'impressive' number of nodes. How big a network can your code effectively work on? What is the big-O of your algorithm?
- A summary of your thoughts on your algorithm and why it is important
- Source code

Please submit this by e-mail.

Grading Rubric for the Report

Your report will be graded out of 100 points, which will be assigned as follows:

- 20 points for your description of your algorithm
- 10 points for your demonstration of the algorithm on small graphs (the test cases listed above are particularly important in demonstrating correctness)
- 10 points for demonstrating correctness on medium sized graphs (the "more than 10 nodes" part mentioned above)
- 10 points for running your code on big nodes (the "impressive" category above)
- 10 points for your Big-O analysis
- 5 points for your summary
- 30 points for the correctness of the algorithm (as determined by your demonstrations in your paper - be sure you are convincing!)

Please turn in a professional-looking report. A sloppy paper or failure to follow directions in other ways will lose you points on your report.