CSE 515: Statistical Methods in Computer Science

Homework #3

Due **in class** on May 13, 2009 (source code emailed to lowd at cs)

Guidelines: You can brainstorm with others, but please solve the problems and write up the answers by yourself. You may use textbooks (Koller & Friedman, Russell & Norvig, Wikipedia, etc.), and lecture notes. Please do NOT use any other resources or references (e.g., example code, online problem solutions, etc.) without asking.

Submission instructions: Submit a **hard copy** of your answers in class to Daniel Lowd. Submit your source code for belief propagation by email to `lowd at cs`.

1. K&F 10.17
2. K&F 13.12
3. Apply the variable elimination algorithm to the network in Figure 1 (from Assignment 2) to compute the probability of Cold given Sneeze and TakeMedicine (i.e., $P(C = \text{true}|S = \text{true}, T = \text{true})$). (Please show the order in which you eliminate variables and some sample intermediate calculations, but you don't need to include every single number from every step.)
4. Use your implementation of belief propagation as you answer the following questions.
   (a) *Briefly* describe how you implemented belief propagation (data structures, code organization, etc.).
   (b) Run your algorithm on the Sprinkler network (sprinkler.bif). What is the marginal probability of WetGrass according to belief propagation? What is the true marginal probability of WetGrass (you can easily compute this by hand)? Explain why these numbers are different.
   (c) Suppose X is a variable with k states and f neighboring factors. Explain how to compute all outgoing belief propagation messages with complexity $O(kf)$. Does your method work even when the incoming messages have zeroes? (NOTE: You are free to use a less efficient method in your actual implementation.)

Programming Project: Belief Propagation

The bulk of this assignment is implementing belief propagation. You may write your implementation of belief propagation in a standard computer language of your choice. If you wish to use a language other than C, C++, Java, Perl, OCaml, Matlab, or Python, please ask us first to verify it will be acceptable.

Your implementation must meet the following input/output specifications in order to keep grading tractable:

- You must include two scripts along with your code: one to compile the code on tricycle (if necessary), and one to run the code on tricycle once it has been compiled.
- The compile script should run in a directory consisting of just your submitted files (including any additional libraries your code needs to compile/run).
- The run script must accept one command-line argument, the name of the Bayesian network in .bif (Bayesian Interchange Format) file, and must write all marginal probabilities to the file "result.txt" in the current directory. (You are free to print any diagnostic output you wish to standard output or other files.)
- The "result.txt" file produced must contain a list of variables and their marginal distributions. Variables must appear in the order in which they were introduced in the .BIF file. Example:

```
A 0.9 0.1
B 0.72 0.28
C 0.001 0.999
D 1.0 0
```

The belief propagation algorithm you implement should follow the description given in class, and also described here: `http://www.comm.utoronto.ca/~frank/papers/KFL01.pdf` NOTE: In order to have comparable answers, please use a simple message schedule in which all variable messages are sent and then all factor messages are sent in each iteration. (Mixing up the order of everything can lead to interesting, but different, results.)

In order to make this assignment less onerous, we provide a skeleton in C++ that handles some of the input and output. You are free to use any portions of that skeleton that you find helpful, or discard it entirely and come up with something else. We also provide several example input/output pairs to help you check your implementation.

You must submit an archive (.zip or .tar.gz) containing your complete implementation in an email to `lowd at cs`.

Appendix

We recommend you use the VFML library, which provides routines for reading .bif files, and provides data structures and functions for traversing the graph, retrieving values from conditional probability tables, etc. The documentation for the library may be found at `http://www.cs.washington.edu/dm/vfml/`. The "Getting Started" link describes how to download the library. It is also included in the hw3-dist.zip download with the skeleton code.

Here are some suggestions if you do end up using VFML. Documentation for the Bayesian network routines is found by following Modules → "Belief Net Section" → BeliefNet.h. The **ExamplePtr** class is used for holding values of the nodes. Spend a little time understanding Examples. You can create an **ExamplePtr** by using *BNGetExampleSpec* to get an **ExampleSpecPtr**, which you give as a parameter to **ExampleNew**. Probably the best routine for retrieving CPT values from the nodes will be *BNNodeGetCP*, which takes an **ExamplePtr** that defines the value of the parents of the node and the state of the node that you want to ask the probability of. Note that the ID of a node (from *BNNodeGetID*) is the same as the attribute number in the Example (for instance, when using ExampleSetDiscreteAttributeValue). Similarly, to find out the number of nodes in the network and names of the nodes, etc, you use the **ExampleSpecPtr** associated with the network, and routines like *ExampleSpecGetNumAttributes* and *ExampleSpecLookupAttributeName*.

It could also be useful to look at the code for *BNLikelihoodSampleNTimes* (in BeliefNet.c) to get an idea about traversing the graph, retrieving CPT values, etc.