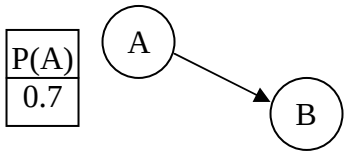
CSE 590ST: Statistical Methods in Computer Science

Homework 2

Due in class on May 3, 2004

1. In the following Bayesian network, what is P(B)?



A	P(B)
T	1.0
F	0.0

2. Write a Gibbs sampler for Bayesian networks. Describe briefly how you implemented the sampler, particularly how nodes are initialized and updated, and how probabilities are computed. We may ask you to demonstrate your code. Submit your code by going to http://abstract.cs.washington.edu/~mattr/590st/.

   The sampler should use rejection sampling to find a good initial node assignment. The sampler must provide a probability, given some set of query and evidence nodes, and their associated values. For example, it should be able to answer things like P(legs=three, color=orange | animal=cat, accident=false). The sampler should be able to perform at least one million samples, and output an answer to the probabilistic query at 10, 100, 1000, 10k, 100k, 200k, 300k, …, 1 million samples. (Note: throughout this homework assignment, the term "sample" means one complete Gibbs pass through all the nodes in the network). You should also seed your random number generator based on time (e.g. srand(time(NULL)) in C), or some other method, so that multiple runs do not produce identical results. See the appendix for suggestions and help in writing your sampler.

3. Using your Gibbs sampler, compute P(B) for the Bayesian network above (you may download the network in BIF format at http://www.cs.washington.edu/education/courses/590st/04sp/ab.bif). Is the answer different from your solution to #1 above? If so, explain why this happens and propose a solution.

4. Now we will use your Gibbs sampler to perform inference on the insurance network, which is used to evaluate car insurance risks in order to appropriately set insurance premiums. Download *insurance.bif* at http://www.cs.washington.edu/education/courses/590st/04sp/insurance.bif.
   Using your Gibbs sampler, find P(Accident=Severe, OtherCarCost=HundredThou | DrivingSkill= Expert). Write down the probability computed by your sampler at 10, 100, 1000, 10k, 100k, 200k, …, 1 million samples. Does it appear to have converged

by 1 million samples? Run your Gibbs sampler again and write down the resulting probabilities again. Did it converge to the same value as the first run? Create a graph of 10 runs of your program, where the x-axis is the number of samples, the probability is the y-axis, and each run is one line on the graph. When does each run appear to have converged. Comparing with the others, when have they all actually converged? Can you explain why this happens?

5. Would you expect P(Accident=Severe | DrivingSkill=Expert) to be greater or less than the probability evaluated in question 4? Create the same graph as you did in question 4, but for P(Accident=Severe | DrivingSkill=Expert).

6. Now we will measure the convergence of your Gibbs sampler when its starting point is chosen with uniform probability. This means, for each unobserved node in the network, its starting value is chosen uniformly at random (and each observed node is simply set to its observed value). Construct the same graph as you did in #5 (plotting P(Accident=Severe | DrivingSkill=Expert)). Does the convergence seem to be affected by the initialization method?

7. It can sometimes be useful to have a quantitative measure of convergence. One useful one compares the variance of the probability estimate within a given run vs. the estimates across multiple runs. Suppose we perform m runs of Gibbs, each of which uses n samples. Let P_{ij} be the probability estimate computed by run i with j samples. Then we define B as the between-run variance, and W as the within-run variance:

$$B = \frac{n}{m-1}\sum_{i=1}^{m}(\overline{P}_i - \overline{P})^2, \text{ where } \overline{P}_i = \frac{1}{n}\sum_{j=1}^{n}P_{ij} \text{ and } \overline{P} = \frac{1}{m}\sum_{j=1}^{m}\overline{P}_i$$

$$W = \frac{1}{m}\sum_{i=1}^{m}s_i^2, \quad \text{where} \quad s_i^2 = \frac{1}{n-1}\sum_{j=1}^{n}(P_{ij} - \overline{P}_i)^2$$

$$R = \frac{1}{n}\cdot\frac{B}{W}$$

When the variance within a run is the same as the variance between runs, we know that the Markov chain has “forgotten” its starting point and is representative of the true distribution. As this happens, R will approach 1.0.

Using the equations above and your Gibbs sampler, plot the evolution of the value of R when using 10 Markov chains to compute P(Accident=Severe | DrivingSkill =Expert). The x-axis of your plot will be the number of samples, the y-axis will be the value of R. There should be two lines – one for each of the initialization methods (rejection sampling and uniformly distributed). Estimate the number of samples required to achieve an R < 1.2 for each.

Appendix

You may write your Gibbs sampler in a standard computer language of your choice. If you wish to use a language other than C, C++, Java, Scheme, Lisp, perl, or python, please ask us first to verify it will be acceptable. The sampler must be able to read a .bif (Bayesian Interchange Format) file and use Gibbs sampling in order to answer a probabilistic query.

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. Note that, the next homework will also involve Bayesian networks, so you may want to consider a small investment of time in understanding VFML in order to save you time on this and your future homework.

VFML

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*. If it is useful to you, you may use *BNGenerateSample*.

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.