B551 Fall 2011 Homework 4

Due date: 10/20/2011

1) Build a one-node Bayes network to represent the probability of drawing candy flavors out of a bag manufactured by some candy company. The node should be named "Flavor" and have three possible values: "Lime", "Lemon", and "Cherry", with probabilities 0.3, 0.3, and 0.4 respectively.
   a) What does the DiscreteBayesNet.ml_result() method return? (since there are no other nodes, your evidence variable is an empty dict).
   b) Set up a loop and draw N samples of the "Flavor" node using the DiscreteBayesNet.rand_result() method. Count the fraction of samples of each flavor. Repeat this with N=100, N=1,000, and N=10,000. What do you observe?
   c) For each value of N in part b, perform the same experiments 100 times using different random seeds. For a fixed value of N, compute the average squared error between the estimated probabilities and the true probabilities. What do you observe?

2) Build a 2-node Bayes network to represent the probability of drawing candy flavors from one of 10 bins in a store. The parent node should be named "Bin" and have possible values "1" through "10". These values should have uniform probabilities (i.e., 10%). The child node should be named "Flavor" and have the following probability table:

Bin	Lemon Lime Cherry
1	0.0, 0.0, 1.0
2	0.0, 1.0, 0.0
3	1.0, 0.0, 0.0
4	0.0, 0.33, 0.67
5	0.0, 0.67, 0.33
6	0.33, 0.0, 0.67
7	0.67, 0.0, 0.33
8	0.33, 0.67, 0.0
9	0.67, 0.33, 0.0
10	0.33, 0.33, 0.33

Suppose a child comes to us with a lime-flavored candy that he or she got out of one of the bins. For each bin, what is the probability that the candy came from that bin? Use the network's enumerate_ask() method to determine your answer.

3) Suppose we know that the probability that the grass is wet in the morning depends on two conditions: Whether or not it rained, and whether or not the sprinkler system came on during the night. There is, overall, a 30% chance of rain. The sprinkler system has a moisture sensor and is supposed to stay off if there is rain, or come on otherwise, but sometimes it malfunctions. If it rains, there is a 10% chance the sprinkler comes on; if it does not, there is a 95% chance the sprinkler comes on. If the sprinkler does not come on and there is no rain, the grass will not be wet in the morning; if one or the other happens, there is a 90% chance the grass will be wet. If both occur, there is a 95% chance the grass will be wet.

   a) Build a Bayesian network to represent this system.

b) What is the probability, overall, that the grass is wet?
c) If the grass is wet, what is the probability that it rained the night before?
d) If we hear the sprinkler come on during the night, will the probability that it also rained be independent of whether the grass is wet the next day? Justify your answer.

4) In the files "poll_train.txt" and "poll_test.txt" there are samples of people's voting habits. Each file contains one polling record per line. Each record consists of 6 values representing, in order, the voter's gender (male/female, notated M and F), race or ethnicity (white/black/hispanic/asian/other, represented by first letter of value), age ('0' for 18-29, '1' for 30-45, '2' for 45-59, '3' for 60 and up), political ideology (progressive/moderate/conservative, by first letter), population of local area ('LC' for large city, 'SC' for small city, 'SB' for suburbs, 'ST' for small town, and 'RU' for rural), and vote (democrat/republican/independent, by first letter).

In the file politics.py is a dict named politicsVars. It defines variable names for each of the parts in a voting record, and maps those to the possible values for those variables. In the same file are functions named load_samples_training() and load_samples_testing(). Each takes a filename as an argument and returns a list of samples.

Each item in the list returned by load_samples_training() is a dict mapping all the variable names to the values of one polling record. This list can be passed directly into DiscreteBayesNet.train_ml(). Each item in the list returned by load_samples_testing() is a tuple. The first entry in the tuple is a dict like the entries from load_samples_training(), EXCEPT that the dict does not contain an entry for the Vote variable. You may use this dict as evidence in DiscreteBayesNet.enumerate_ask(). The second entry in the tuple is the value of the Vote variable corresponding to that entry, which can be held in reserve and used to test an estimate of the Vote value.

To test a network designed to model this sytem, we would use load_samples_testing() to load some test cases from a file, then step through those testing samples. On each sample, we would use the evidence in sample[0] to enumerate_ask() a probability distribution for the Vote variable, and derive a guess for the Vote value based on that distribution. Then we would test our guess against the outcome given in sample[1]. After trying all the samples, we can divide the number of correct guesses by the number of samples and obtain our accuracy.

To test your network for questions below, use testing samples obtained from poll_test.txt.

a) Construct a Bayesian network representing the system. Some questions to consider in your design: What variables does the Vote outcome depend on? How do other variables depend on each other?
b) What independence relations exist in your network?
c) Try to hand-tune your network's probability tables. What is the best accuracy you can achieve? What are the greatest difficulties you encounter? (Note: Superhuman efforts are not required here. Don't spend more time than you need to in order to get a decent grasp of the situation and answer the questions.)
d) Explain the theoretical and practical differences between DiscreteBayesNet.ml_result(), DiscreteBayesNet.enumerate_ask(), and the Monte-Carlo methods in DiscreteBayesNet.monte_carlo_estimate()?
e) Load training samples from the file poll_train.txt. Use DiscreteBayesNet.train_ml() to train your network. What accuracy do you achieve now? How does this process compare to hand-tuning?

Submission Instructions

Place the code you use to answer each question in hw4.py, in functions p1() through p4(). Run all the functions when hw4.py is executed by placing calls to them in an "if __name__=='__main__':" block. Turn in hw4.py on OnCourse.

Write up your answers to the written parts of the questions and turn them in hard-copy in class.