CS360 – Project #3

Expert Systems (50 Points)

In this part of the project, you will be using a tool to develop and query Bayesian networks. Download the AIspace Belief and Decision Networks Java applet from `http://aispace.org/bayes/`. This Java applet should run on all platforms (we have confirmed that it runs on Windows 7). You might need to add "http://www.aispace.org/" to the 'Exception Site List' available in the 'Security' tab of the 'Java Control Panel' in order to get the applet to run. Read the documentation available at the download site and try it out. Do this well before the deadline to ensure that it works on your computer.

Using the Java applet, develop a Bayesian network that can be used for diagnosis or prediction in an area of your expertise, e.g. to diagnose a disease, car trouble or computer problem or to predict the next presidential election. We expect your Bayesian network to have about 15 nodes and a non-trivial structure. Save your Bayesian network (using 'File' → 'Save program') and print it (using 'File' → 'Print'). Then, demonstrate the power of your Bayesian network. Create three interesting nontrivial scenarios that differ in the variables with known values, show the diagnoses/predictions of your Bayesian network for these scenarios and explain why they make sense.

Game Playing (50 Points)

Consider the following game played with one (fair, six-sided) die.

Initially, you have 5 lives, a total score of zero and a scratch score of zero. You repeatedly roll the die. After each die roll, you follow the following rules:

- If you rolled a 6, then you lose a life and your scratch score is set to zero. If you lose all lives, then the game is over and you lose the game.
- If you rolled a 1 to 5, then the rolled number is added to your scratch score and you have the option to yell "hold." If you yell "hold," your total score is increased by your scratch score and your scratch score is set to zero. If your total score becomes 40 points or higher after you yell "hold," then the game is over and you win the game. Otherwise, you lose a life. If you lose all lives, then the game is over and you lose the game.

Here is the beginning of a game trace:

1. lives = 5; total score = 0; scratch score = 0.

   You roll a 2 and do not yell "hold."

2. lives = 5; total score = 0; scratch score = 2.

   You roll a 3 and do not yell "hold."

3. lives = 5; total score = 0; scratch score = 5.

   You roll a 4 and yell "hold."

4. lives = 4; total score = 9; scratch score = 0.

   You roll a 2 and do not yell "hold."

5. lives = 4; total score = 9; scratch score = 2.

   You roll a 6.

6. lives = 3; total score = 9; scratch score = 0

   ...

A state in this game can be characterized by the remaining number of lives and current scratch and total scores. If both the scratch score and the number of lives are above zero, there is a choice between yelling "hold" and rolling the die or rolling the die without yelling "hold". Otherwise, the only option is rolling the die. Depending on the die roll, you can end up in six different states, after adding the die-roll to the scratch score (or setting it to 0 and losing a life if you rolled a 6).

Write a program in C++ or Java to determine a policy that maximizes the probability of winning (called the "optimal policy"), that is, assign to each state the action to execute in it. It should print out the optimal policy for each state where you have the option of yelling "hold", and the probability of winning the game by following the optimal policy from that state. Exclude states where the sum of the total and scratch scores is at least 40 (since it is always optimal to yell "hold" and win the game if that is the case). Your program's output should match the following format (**the values are made up**):

```
5 0 1 No 0.1890
5 0 2 No 0.1938
4 21 18 Yes 0.9268
...
```

The first three integers on each line denote the remaining number of lives, total score and scratch score, in this order. 'Yes' means that the optimal policy in that state is to yell "hold". The last value denotes the probability of winning the game by following the optimal policy from that state. The order in which you list the states is not important. Make sure your program compiles and runs on 'aludra.usc.edu'.

Using your program, answer the following questions:

a) Assume that you are halfway through the game and have $X > 0$ lives, a total score of 20 and a scratch score of $Y < 20$. For which values of X and Y should you yell "hold" to maximize your probability of winning?

b) How much are you willing to pay for participating in this game at most if you want to maximize your expected profit, winning the game pays \$500 and losing it pays \$0?

Extra Credit (30 Points)

Write a program for finding the optimal policy and answer the same questions for the following variant of the game:

Initially, you have 5 lives, a total score of zero and a scratch score of zero. You repeatedly roll the die. After each die role, you follow the following rules:

- If you rolled a 6, then you lose a life and your scratch score is set to zero. If you lose all lives, then the game is over and you lose the game.
- If you rolled a 5, then you win a life (provided that you had fewer than five lives) and your scratch score is set to zero.
- If you rolled a 1 to 4, then the rolled number is added to your scratch score and you have the option to yell "hold." If you yell "hold," your total score is increased by your scratch score and your scratch score is set to zero. If your total score becomes 40 points or higher after you yell "hold," then the game is over and you win the game. Otherwise, you lose a life. If you lose all lives, then the game is over and you lose the game.

Hints

- You should model the game as an MDP and perform value iteration to find the optimal policies.
- For the first variant of the game, you can program a version of value iteration that processes every state only once, but you need to be careful about the order in which you process the states.

Submission

You can submit your solutions either in PDF via Blackboard or in hard copy to Tansel or Sven (on time). You should include your name and student ID in your submission.

For the first question, also submit the .xml file containing your Bayesian network via Blackboard. For the second question (and the bonus question), submit an archive containing your source code, a makefile to compile it and your program's output named "output.txt" (you should submit a different archive for the bonus question). You should also put your name and student ID as comments at the beginning of your source files.