CSCI 561: Foundations of Artificial Intelligence
Spring 2016, Homework #3
Due on Apr 18, 2016 at 11:59 PM PST

Introduction

With the help of your logical system, the wise Master Yoda has successfully rooted out the family traitor, Anakin. However, Anakin managed to flee to the UCLA campus before being captured. Once there, he incited the UCLA patriarch Darth Sidious to declare war on the Viterbi Fluffy Hackers family. An LA galaxy war has broken out!

Master Yoda has decided to take an initiative in the war. One idea is to first demolish the enemy's morale before engaging in any fight. To do this, an infiltration team will be sent into the UCLA campus at night, and paint the enemy's family mascot, the Sith Bear, into cardinal (lucky color of the USC squirrels).

However, Anakin has fled with some secret intelligence, which may include this infiltration operation idea. If the enemy knows about this idea, there will be a much higher chance that they will put up night defense near the Sith Bear, which will make it much harder for the operation to be successful, and even risk the safety of the brave infiltration warriors. However, if the enemy put up night defense, but the infiltration is cancelled, the enemy might be demoralized due to wasted effort.

Master Yoda needs to decide whether or not it's worth the effort to approve the operation. As the most successful strategy consultant for the Viterbi family, you were asked to provide analysis on the chance of successfully demoralizing the enemy, and the best decision to make.

The wise Master Yoda can provide all the basic probabilities and utility functions based on his experience for your analysis. You must build a **Bayesian Network** for analysis. If Master Yoda delegates the command to his field office, Luke, whether to send infiltration will be represented as a probability, then your task will be analyzing various **probabilities**. If Master Yoda makes the decision himself, then your task will be suggesting the **best decision** with the **Maximum Expected Utility**. There **might be more than one decision** to make, as it might be worthwhile to spread rumors and let the enemy know the idea, to trick them into putting up wasted defense.

Assignment

You will be given a Bayesian network which may contain several decision nodes and a utility node. There are two types of tasks:

1) Use variable elimination or enumeration to calculate specific joint probabilities, marginal probabilities, or conditional probabilities, when there is either no decision node, or the decision node has a set value (for conditional probabilities). (~50%)
2) Calculate the expected utility or maximum expected utility, when there are one or more unset decision nodes. (~50%)

Pseudocode

1. The enumeration algorithm for answering queries on Bayesian networks: AIMA Figure 14.9
2. The variable elimination algorithm for inference in Bayesian networks: AIMA Figure 14.11
3. A decision-theoretic agent that selects rational actions: AIMA Section 16.5.2

Please note in this homework we don't require any logs or intermediate results, so you may choose to use any algorithm to solve the problems, as long as you get the correct outputs.

Input

You will be given a Bayesian network and several queries in a text file ending with a **.txt** extension. *Sample01.txt* presents the Bayesian network as in Figure 1, in which all nodes only have two values: "+" (event occurred) or "-" (event not occurred).

(sample01.txt)

```
P(NightDefense = +, Infiltration = -)
P(Demoralize = + | LeakIdea = +, Infiltration = +)
******
LeakIdea
0.4
***
NightDefense | LeakIdea
0.8 +
0.3 -
***
Infiltration
0.5
***
Demoralize | NightDefense Infiltration
0.3 + +
0.6 + -
0.95 - +
0.05 - -
```

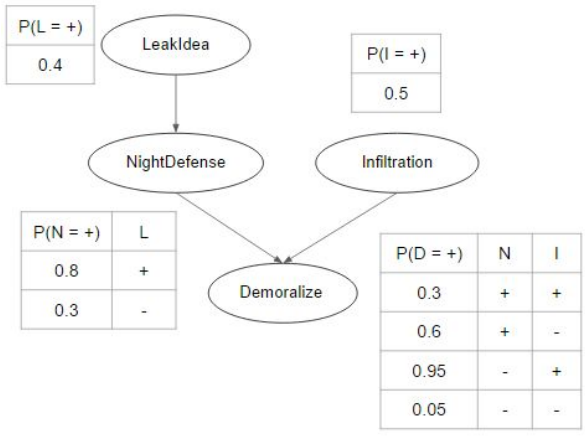

Figure 1

The input format is as follows:
The first few lines are queries (one query per line). The query can have three forms:

1) Starting with “P”, asking for specific joint probabilities, marginal probabilities, or conditional probabilities, when there is either no decision node, or the decision node has a set value (for conditional probabilities):
   e.g. `P(NightDefense = +, Infiltration = -)`
   `P(Demoralize = + | LeakIdea = +, Infiltration = +)`
2) Starting with “EU”, asking for expected utility, conditioned on other observed evidence, when there is one or more than one unset decision nodes (expected utility will always be based on all unset decision nodes):
   e.g. `EU(Infiltration = +)`
   `EU(Infiltration = + | LeakIdea = +)`
   `EU(LeakIdea = -, Infiltration = +)`
3) Starting with “MEU”, asking for maximum expected utility, conditioned on other observed evidence, when there is one or more than one unset decision nodes:
   e.g. `MEU(Infiltration)`
   `MEU(Infiltration | LeakIdea = +)`
   `MEU(LeakIdea, Infiltration)`

The line after all queries will have six “*” as the separator.
The following lines represent a Bayesian network by showing the tables of probabilities / conditional probabilities for each event / node. The tables are separated by three “*”, and will have the following format:

e.g.
```
Demoralize | NightDefense Infiltration
0.3 + +
0.6 + -
0.9 - +
0.05 - -
```

- The first line contains the node’s name, followed by the names of its parents, separated by a “|” sign (or no “|” sign when there’s no parent).
- All node names begin with an uppercase letter and contain letters only. You can assume that each name has at most 20 letters.
- The following lines show the probabilities for all combinations of parent node values.
- **All nodes can only have two values, “+” (event occurred) or “-” (event did not occurred)**.
- The probability will range from 0 to 1 (but not 0 or 1 exactly), and is always for occurrence (“+”) only. Your program should compute the probability of nonoccurrence (“-”).
  For example, given
  `P(LeakIdea = +) = 0.4`
  in *`sample01.txt`*, your program should compute:
  `P(LeakIdea = -) = 1 - P(LeakIdea = +) = 0.6`
- The parent node values always follow the order in which they appear in the first line.
- With the exception of the first line, there is no specific order between lines of the tables (e.g. “`+ -`” may appear after “`- +`”).

- You may assume each node may have at most 3 parent nodes (8 lines of probabilities).
- When a node has no parent, then there is only a single number (probability of occurrence):
  e.g.
  ```
  LeakIdea
  0.4
  ```
- Every node has a corresponding table in the input file, so you can know the network structure from the first line of each table. For example, your program should figure out the network structure in *sample01.txt* based on the following information:
  ```
  LeakIdea
  NightDefense | LeakIdea
  Infiltration
  Demoralize | NightDefense Infiltration
  ```
- There won't be any directed cycles in the given networks.
- Parent nodes always have their tables appearing before the child node.

There is no decision node in *sample01.txt*. The node "Infiltration" is changed to a decision node in *sample02.txt*, with an extra utility node and its table, and different queries:

(sample02.txt)

```
P(Demoralize = + | LeakIdea = -, Infiltration = +)
EU(Infiltration = +)
EU(Infiltration = + | LeakIdea = +)
MEU(Infiltration)
MEU(Infiltration | LeakIdea = +)
******
LeakIdea
0.4
***
NightDefense | LeakIdea
0.8 +
0.3 -
***
Infiltration
decision
***
Demoralize | NightDefense Infiltration
0.3 + +
0.6 + -
0.95 - +
0.05 - -
******
utility | Demoralize
100 +
-10 -
```

- A decision node will never have a parent node, and instead of a probability table, there is only the word “decision” (non-capitalized) to mark it as a decision node.
- There are possibly more than one decision nodes. For example, “`LeakIdea`” might be made a decision node as well (if Master Yoda wants to spread some rumors). You may assume there are at most 3 decision nodes.
- When asking for probability in a query, the values of all decision nodes will be given as conditions.
- With at least one decision node in the network, there will always be a utility node as well, and its utility table will be given at the end of the input file, separated from others by six “*”.
- The format of a utility table is similar to that of a normal node with parent. The first line begins with the word “utility” (non-capitalized), separated with a “|” sign, then lists all dependency nodes.
- The following lines show the utility value for all combinations of dependency node values.
- The utility value is always an integer, possibly negative.
- There will be no more than one utility node, with no more than 3 dependency nodes.

Additional Notes:

- All event names, numeric values and signs are separated by whitespace, except for query function (e.g. P, EU, MEU), parentheses and the comma sign (“,”) in a query. Please follow the format of the sample files carefully when parsing the input file.
- The input file given to your program will not contain any formatting errors, so it is not necessary to check for those.

Output

The result should be printed to a file called **output.txt**. Given the sample input above, the output content should be as follows:

(sample01.output.txt)

```
0.25
0.43
```

(sample02.output.txt)

```
0.76
59
37
+ 59
- 44
```

For each query in the input file, your program should generate a corresponding result (one result per line) as the output. The result may have three forms:

1) “P” query - A decimal value between 0 and 1, **rounded to two decimals**:

e.g. `0.40`

2) "EU" query - An integer value:
   e.g. `50`
3) "MEU" query - One sign("+" or "-") for each decision node, followed by an integer value representing the maximum expected utility, all separated with **a single whitespace**:
   e.g. `+ 50`

- When there are more than one decisions, the order of decisions should be the same as in the query.
  e.g. Input: `MEU(Infiltration, LeakIdea)`
  Output: `+ - 50`
- The test cases will be designed so that there will always be one unique solution with MEU.
- **For EU and MEU queries, all calculations should be done in decimal number accuracy, but the output expected utility value should be rounded to nearest integer**.
- **Don't** print additional whitespace after the value, or extra line break in the end.

Additional Notes:

- **You will get zero points if you don't follow the output format exactly.** Any regrading requests about this will be ignored.
- With the given description, we don't believe that multiple outputs are possible for any test case. If you are able to think of any such case, please let us know and we will make the necessary changes in the grading guidelines.
- The final test cases will be different from the sample test cases provided. Your assignment will be graded based on the performance on the final test cases only.

Grading Notice

- **Please follow the instructions carefully. Any deviations from the instructions will lead your grade to be zero for the assignment**. If you have any questions, please use the discussion board. Do not assume anything that is not explicitly stated.
- You must use **PYTHON** (Python 2.7) to implement your code. You are allowed to use standard libraries only. You have to implement any other functions or methods by yourself.
- You need to create a file named "hw3cs561s16.py". The command to run your program would be as follows: (When you submit the homework on labs.vocareum.com, the following commands will be executed.)
  python hw3cs561s16.py –i *inputFile*

- You will use labs.vocareum.com to submit your code. Please refer to http://help.vocareum.com/article/30-getting-started-students to get started with the system. Please only upload your code to the "/work" directory. Don't create any subfolder or upload any other files.
- If we are unable to execute your code successfully, you will not receive any credit.
- **For your final submission, please do not print any logs on the console.**

- **When you press "Submit" on Vocareum, your code will be run against the grading script and the sample input/output files. You will receive feedback on where your code is making errors, if any. You can click "Submit" to test new versions of your code as many times as you like up until the deadline. Your last submission will be graded. The sample input/output files are designed to cover many basic situations of the problem and rules of the output log, so please utilize them as much as you can.**
- Your program should handle all test cases within a reasonable time (not more than a few seconds for each sample test case). The complexity of test cases is similar to, but not necessarily the same as, the ones provided in the homework.
- The deadline for this assignment is Apr 18, 2016 at 11:59 PM PST. **No late homework will be accepted.** Any late submissions will not be graded. Any emails regarding late submissions will be ignored.