Purpose

We need to implement a Bayesian network that can output exact and approximate inferences of a query variable given the evidence variables. Using the textbook, the inference by enumeration algorithm was implemented for exact inference. The likelihood weighting method was implemented for the approximate inferences.

Compile and Run

Compile: chmod +x main.py
Run in Terminal
Exact: python3 main.py aima-alarm.xml B J true M true
Approximate: python3 main.py 1000 aima-alarm.xml B J true M true

Files

parser.py: Contains functions to parse the XMLBIF file and construct the Bayesian network:
bn.py: Defines classes for the Bayesian network
exact_inference.py: Implements the exact inference algorithm using enumeration.
approximate_inference.py: Computes approximate inference by likelihood weighting algorithm
main.py: main file that handles command-line arguments and outputs the inferences.
read.py: given code that outputs(variables, domains, parents, tables)

References/Files

- Pseudocode from AIMA
- 3x Xmlbif files
- main.py
- parser.py
- bn.py
- exact_inference.py
- Approximate_inference.py
- read.py

Writeup and Experimental Work

- Part 0: Parsing Algo.
- Part 1: Exact Inference
- Part 2: Approximate Inference
- Part 3: Example Analysis
- Part 4: Evaluation of Sample Size

Part 0: XMLBIF Parsing and BN

parser.py:

- changed given code read.py from the professor with two files (parser.py & bn.py)
- Extracted variables, their domains, parent, and CPTs from the XMLBIF file.
- Included helper functions like extract_probabilities to process CPT, and list_product to compute Cartesian products of parent domains

bn.py:

- Represents node in the Bayesian Network, with name, domain, parents, children, and CPT

Part 1: Exact Inference

exact_inference.py:

- BNs are DAGs: parents' values must be found before other variables are (toplogical_sort)
- topological_sort: Implement marking algo. to store variable in list
- enumeration_ask, enumeration_all from textbook.

Part 2: Approximate Inference (Likelihood Weighting)

approximate_inference.py: Computes approximate inference by likelihood weighting algorithm
X_name (W) = query
e: evidence variables
bn: bayesian network from file.xml
N: sample number
Algorithm:
Initialize weight(w) = 1.0 and empty sample x = {}.

For each variable in e:

assign observed value, and update weight by multiplying variables by observed value given parents.

For variable not in e:

Sample variable based on P(variable| parents)
Add sample value to x

Random sampling to get sample value
Update samples' weight
Call normalize function to compute P

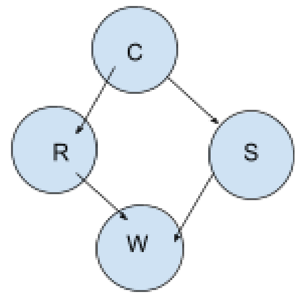
Example Analysis

WetGrass

Variables

C (Cloudy)
S (Sprinkler)
R (Rain)
W(WetGrass)

BN



Ex. P(W | S=true R=true) probability of grass is wet given sprinkler is on and is raining
(python3 main.py 1000 aima-wet-grass.xml W S true R true)

XML File

```
<!-- P(C) -->
<DEFINITION>
        <FOR>C</FOR>
        <TABLE>0.5 0.5</TABLE>
</DEFINITION>

<!-- P(S|C) -->
<DEFINITION>
        <FOR>S</FOR>
        <GIVEN>C</GIVEN>
        <TABLE>
          <!--          S    !S -->
          <!-- C -->   0.1   0.9
          <!-- !C -->  0.5   0.5
        </TABLE>
</DEFINITION>

<!-- P(R|C) -->
<DEFINITION>
        <FOR>R</FOR>
        <GIVEN>C</GIVEN>
        <TABLE>
          <!--          R    !R -->
          <!-- C -->   0.8   0.2
          <!-- !C -->  0.2   0.8
        </TABLE>
</DEFINITION>

<!-- P(W|S,R) -->
<DEFINITION>
        <FOR>W</FOR>
        <GIVEN>S</GIVEN>
        <GIVEN>R</GIVEN>
        <TABLE>
          <!--            W     !W -->
          <!-- S R -->   0.99  0.01
          <!-- S !R -->  0.90  0.1
          <!-- !S R -->  0.90  0.1
          <!-- !S !R --> 0.0   1.0
        </TABLE>
</DEFINITION>
```

Likelihood Weighting

Repeat N times{
W = {'true': 0.0, 'false': 0.0}
get sample value of C:
- Not in evidence
- x={'C': 'false'}

Get sample value of S
- In evidence
- Get value P(S = 'true' | C = 'false')
- w *= P(S = 'true' | C = 'false')

R:
- P(R = 'true' | C = 'false')
- w *= P(R = 'true' | C = 'false')

W:
- P(W | S = 'true', R = 'true')
- x['W'] = 'true'

W['true'] += w }

return normalize (W)

Evaluation of Sample Size
In aima wet grass

Query: W
Evidence: S = true, R = true
Exact Inference:

$$P(W{=}true \mid S{=}true, R{=}true)=0.99$$
$$P(W{=}false \mid S{=}true,R{=}true)=0.00999999998$$

Approximate Inference Table

N	P(W=true)	Absolute Error %
50	1.0	1.01%
100	1.0	1.01%
200	0.9944196428571428	.5%
10300	0.990626818074082	.06%