Project 3 Inference

Bayes net for inference

As requirement, the project should have ability to parser special xml documents and extract probability from them. So the beyesi python file is used to complete the task. First I design the Db class and it will have two properties – variables and nodetree. In variable dictionary I will build database for every variable and their probability with special conditions. In nodetree I will create a DAG tree and every node will contain the probability information, their children node and their parents node information.

Parser function

Import xml.dom.minidom and os modules to connect xml file with program. As observation, it can get variables' name from "NAME" node in xml files. In the same case, exact all the information from special dom node then transform them into string format, finally store in Db class. Due to there will be two different format of xml file and one of two wont have "PROPERTY" dom node. So in basic of that divide the two format document.

CreateDAG function

Build node for every variable in xml file and as information to make sure which are their parents node or children node. In the end arrange the order of variable to complete the DAG.

Note

This part is always same in every modules of inference no matter what algorithm it uses.

Test

Main function

In this function it import sys modules to receive data from terminal view. Then create corresponding bayes net. Finally parser the input sentence and exact the query information.

Parserinput function

When using sys modules, the input sentence will be regarded as a string containing not only the sentence but the file information. So exact particular variable from it and then transform it into other form.

Parserquery function

It is used to handle the input sentence which has been processed and extract the evidence variable and query variable.

emAsk function

It is used for normalize the probability after complicated calculation. Then return a list containing two probability to result.

emAll function

This is aimed to count the probability of some variables when there has been evidence based on bayes net. Normally it can help to get P(x|x(parents)). In enumerate algorithm it should input all the variables. If the variable is not in evidence, it will assign the variable as its domains. If the variable is in evidence, it will only assign as given value. After assignment, it will be added into evidence with given value.

Rest function

It will remove the variable after it get assigned.

Have_value function

It used to check if the variable have been assigned and find the value it has.

Position function

It can find the probability from bayes net with variable’s parents information. Index expresses the place of parents node and value used to specify which probability should be taken.

Findprob function

With given variable and evidence, it will return the probability in bayes net.

Reject_sampling function

When adapting reject sample algorithm it will produce every possible world, then find their probability. Then sampling these world on basic of probability and reject all the samples which are not satisfied.

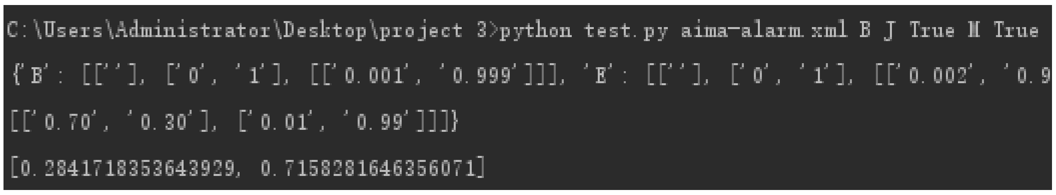
Generate function

It will create the specimens as requirement. In reject algorithm, it will generate all the possible world. In likehood algorithm it just create the desired specimens with evidence, then find their weight and probability.

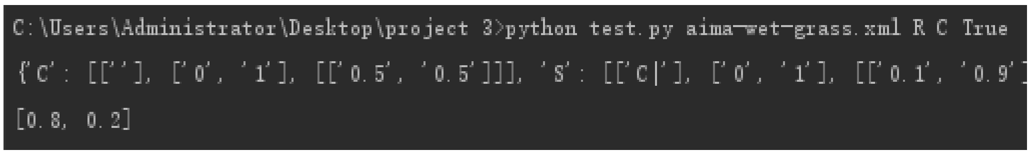
Likehood_sampling function

When adapting reject sample algorithm it will calculate the condition’s likehood based on their weight and probability. When applying gibbs algorithm, it will first find the transition probability then by multiples to get P(x|mb(parents)).

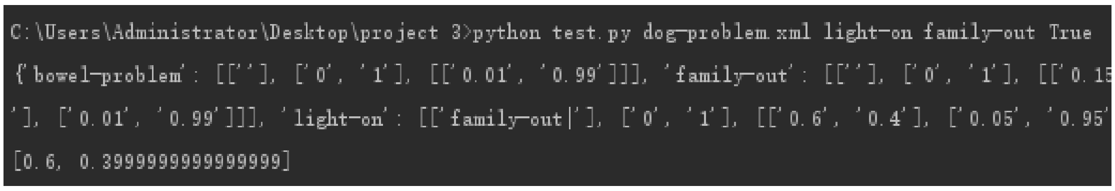
Result



```
C:\Users\Administrator\Desktop\project 3>python test.py aima-alarm.xml B J True M True
{'B': [[''], ['0', '1'], [['0.001', '0.999']]], 'E': [[''], ['0', '1'], [['0.002', '0.9
[['0.70', '0.30'], ['0.01', '0.99']]]}
[0.2841718353643929, 0.7158281646356071]
```



```
C:\Users\Administrator\Desktop\project 3>python test.py aima-wet-grass.xml R C True
{'C': [[''], ['0', '1'], [['0.5', '0.5']]], 'S': [['C|'], ['0', '1'], [['0.1', '0.9']
[0.8, 0.2]
```



```
C:\Users\Administrator\Desktop\project 3>python test.py dog-problem.xml light-on family-out True
{'bowel-problem': [[''], ['0', '1'], [['0.01', '0.99']]], 'family-out': [[''], ['0', '1'], [['0.15
'], ['0.01', '0.99']]], 'light-on': [['family-out|'], ['0', '1'], [['0.6', '0.4'], ['0.05', '0.95'
[0.6, 0.3999999999999999]
```

Note

In this part, the main function, pareserinput, parserquery, emAll, rest, have_value, position and findprob functions are basically the same. Only when adapting different algorithm there will be some differences.