Middle East Technical University Department of Computer Engineering

CENG 462

Artificial Intelligence

Fall '2022-2023

Homework 4

Due date: 25 December 2021, Sunday, 23:55

1 Objectives

This assignment aims to familiarize you with the Bayes Net at the implementation level and provide hands-on experience with three concept Representation,Inference and Sampling.

2 Problem Definition

So far, we've been dealing with the problems where the actions have had deterministic consequences and running search methods in order to obtain their optimal solutions (i.e. sequence of actions that attains the problem goal with the least incurred cost). Each applied action has resulted in a particular state with a probability of 1 and all the techniques (we have implemented so far) have depended on this fact for execution. For the problems in which this assumption is violated, it is no use employing the previous techniques.

In this assignment, the first thing is representing the given input as Bayes Net. After that, do exact inference and Gibbs sampling.

2.1 Bayes Nets (Representation)

Bayesian networks can represent essentially any full joint probability distribution and in many cases can do so very concisely.It is a directed graph in which each node is annotated with quantitative probability information. The full specification is as follows:

1. Each node corresponds to a random variable, which may be discrete or continuous.
2. A set of directed links or arrows connects pairs of nodes. If there is an arrow from node X to node Y , X is said to be a parent of Y. The graph has no directed cycles (and hence is a directed acyclic graph, or DAG.
3. Each node Xi has a conditional probability distribution $P(X_i|Parents(X_i))$ that quantifies the effect of the parents on the node.

As an example of a Bayes Net, consider a model where we have five binary random variables described below:

- B: Burglary occurs.
- A: Alarm goes off.
- E: Earthquake occurs.
- J: John calls.
- M: Mary calls.

Assume the alarm can go off if either a burglary or an earthquake occurs, and that Mary and John will call if they hear the alarm. We can represent these dependencies with the graph shown below.

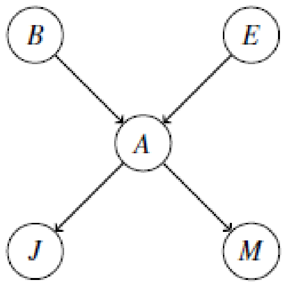


Figure 1: A sample representation of these dependencies.

To define a Bayes Net as consisting of:

- A directed acyclic graph of nodes, one per variable X.
- A conditional distribution for each node $P(X|A_i...A_n)$, where A_i is the ith parent of X, stored as a **conditional probability table** or **CPT**. Each CPT has n+2 columns: one for the values of each of the n parent variables $A_i...A_n$, one for the values of X, and one for the conditional probability of X.

In the Alarm model above,we would store probability tables $P(B)$,$P(E)$,$P(A|B,E)$,$P(J|A)$,$P(M|A)$ Given all of the CPTs for a graph, we can calculate the probability of a given assignment using the chain rule:

$$P(X1, X2, ...Xn) = \prod_{i=1}^{n} P(x_i|parents(X_i)$$

Or in other words, that the probability of a specific value of X_i depends only on the values assigned to X_i's parents.
For the alarm model above, we might calculate the probability of one event as follows: $P(-b, -e, +a, +j, -m) = P(-b).P(-e).P(+a|-b, -e).P(+j|+a).P(-m|+a)$

2.2 Bayes Nets (Inference)

Inference is the process of calculating the Joint Probability Density Function (PDF) for some set of query variables based on some set of observed variables. We can solve this problem naively by forming the joint PDF and using inference by enumeration.

A query $P(X|e)$ can be answered using below equation:

$$\mathbf{P}(X \mid \mathbf{e}) = \alpha\, \mathbf{P}(X, \mathbf{e}) = \alpha \sum_{\mathbf{y}} \mathbf{P}(X, \mathbf{e}, \mathbf{y})\,.$$

Figure 2: General inference equation.

A Bayesian network gives a complete representation of the full joint distribution. $P(x, e, y)$ in the joint distribution can be written as products of conditional probabilities from the network.Therefore, **a query can be answered using a Bayesian network by computing sums of products of conditional probabilities from the network.**

Consider the query $P(Burglary|JohnCalls = true, MaryCalls = true)$. The hidden variables for this query are Earthquake and Alarm. From 2equation this query written as:

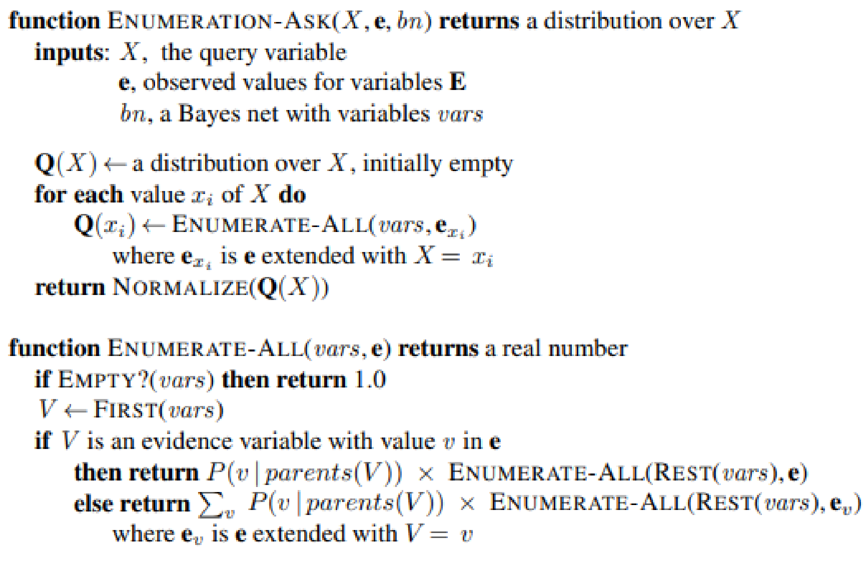
$$P(B|j, m) = \alpha \sum_{e} \sum_{a} P(B, j, m, e, a)$$

In terms of CPT entries: We have

$$P(b|j, m) = \alpha \sum_{e} \sum_{a} P(b)P(e)P(a|b, e)P(j|a)P(m|a)$$

To compute this expression, we have to add four terms, each computed by multiplying five numbers.

The ENUMERATION-ASK algorithm 3 evaluates such trees using depth-first recursion. The algorithm is very similar in structure to the backtracking algorithm for solving CSPs and the DPLL algorithm for satisfiability.



```
function ENUMERATION-ASK($X$, $\mathbf{e}$, $bn$) returns a distribution over $X$
  inputs: $X$, the query variable
          $\mathbf{e}$, observed values for variables $\mathbf{E}$
          $bn$, a Bayes net with variables $vars$

  $\mathbf{Q}(X) \leftarrow$ a distribution over $X$, initially empty
  for each value $x_i$ of $X$ do
      $\mathbf{Q}(x_i) \leftarrow$ ENUMERATE-ALL($vars$, $\mathbf{e}_{x_i}$)
          where $\mathbf{e}_{x_i}$ is $\mathbf{e}$ extended with $X = x_i$
  return NORMALIZE($\mathbf{Q}(X)$)

function ENUMERATE-ALL($vars$, $\mathbf{e}$) returns a real number
  if EMPTY?($vars$) then return 1.0
  $V \leftarrow$ FIRST($vars$)
  if $V$ is an evidence variable with value $v$ in $\mathbf{e}$
      then return $P(v \mid parents(V)) \times$ ENUMERATE-ALL(REST($vars$), $\mathbf{e}$)
      else return $\sum_v P(v \mid parents(V)) \times$ ENUMERATE-ALL(REST($vars$), $\mathbf{e}_v$)
          where $\mathbf{e}_v$ is $\mathbf{e}$ extended with $V = v$
```

Figure 13.11 The enumeration algorithm for exact inference in Bayes nets.

Figure 3: Enumaration-ask algorithm.

2.3 Bayes Nets (Sampling)

Alongside exact inference,there exist approximate inference in Bayes Nets. These algorithm called as randomized sampling algorithms.There are four sampling algorithms but in this assigment, you are expected implement the Gibbs Sampling algorithm.In this approach, the algorithm for Bayesian networks starts with an arbitrary state (with the evidence variables fixed at their observed values) and generates a next state by randomly sampling a value for one of the nonevidence variables X_i.The sampling for X_i is done conditioned on the current values of the variables in the Markov blanket of X_i.(Markov blanket of a variable consists of its parents, children, and children's parents.) The algorithm therefore wanders randomly around the state space—the space of possible complete assignments—flipping one variable at a time, but keeping the evidence variables fixed.

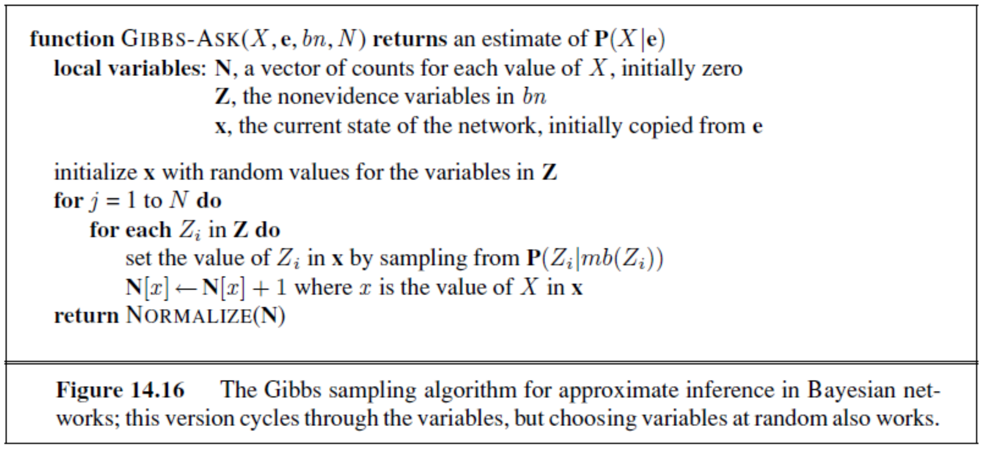
Consider the query $P(Burglary|JohnCalls = true, MaryCalls = true)$. The evidence variables JohnCalls and MarryCalls are fixed to their observed values and non-evidence variables Burglary,Earthquake and Alarm are initialized randomly let us say to true, true and false respectively. Thus, the initial state is [Burglary = false, Alarm = false, Earthquake = true, JohnCalls = true, MaryCalls =true] Now the nonevidence variables are sampled repeatedly in an arbitrary order. For example:

1. Burglary is sampled, given the current values of its Markov blanket variables: in this case,, we sample from $P(Burglary|Alarm = true, Earthquake = true)$. (Shortly, we will show how to calculate this distribution.) Suppose the result is Burglary =false. Then the new current state is [Burglary = false, Alarm = true, Earthquake = true, JohnCalls = true, MaryCalls =true].

2. Earthquake is sampled, given the current values of its Markov blanket variables: in this case,, we sample from $P(Earthquake|Alarm = true, Burglary = false)$. (Shortly, we will show how to calculate this distribution.) Suppose the result is Earthquake =false. Then the new current state is [Burglary = false, Alarm = true, Earthquake = false, JohnCalls = true, MaryCalls =true].

3. Alarm is sampled, given the current values of its Markov blanket variables: in this case,, we sample from $P(Alarm|JohnCalls = true, MaryCalls = true, Earthquake = true, Burglary = false)$. (Shortly, we will show how to calculate this distribution.) Suppose the result is Alarm =false. Then the new current state is [Burglary = false, Alarm = false, Earthquake = true, JohnCalls = true, MaryCalls =true].

Each state visited during this process is a sample that contributes to the estimate for the query variable Burglary. If the process visits 20 states where Burglary is true and 60 states where Burglary is false, then the answer to the query is NORMALIZE(20, 60) = 0.25, 0.75. The complete algorithm is 4



```
function GIBBS-ASK($X$, $\mathbf{e}$, $bn$, $N$) returns an estimate of $\mathbf{P}(X|\mathbf{e})$
  local variables: $\mathbf{N}$, a vector of counts for each value of $X$, initially zero
                   $\mathbf{Z}$, the nonevidence variables in $bn$
                   $\mathbf{x}$, the current state of the network, initially copied from $\mathbf{e}$

  initialize $\mathbf{x}$ with random values for the variables in $\mathbf{Z}$
  for $j$ = 1 to $N$ do
      for each $Z_i$ in $\mathbf{Z}$ do
          set the value of $Z_i$ in $\mathbf{x}$ by sampling from $\mathbf{P}(Z_i|mb(Z_i))$
          $\mathbf{N}[x] \leftarrow \mathbf{N}[x] + 1$ where $x$ is the value of $X$ in $\mathbf{x}$
  return NORMALIZE($\mathbf{N}$)
```

Figure 14.16 The Gibbs sampling algorithm for approximate inference in Bayesian networks; this version cycles through the variables, but choosing variables at random also works.

Figure 4: Gibbs-ask algorithm.

3 Specifications

- You are going to implement ENUMERATION-ASK and GIBBS-ASK in Python 3.
- Each query is represented with a file and this file will be fed to your implementation along with one of the methods.
- To be able to run both methods from a single place, you are expected to write the following function:

```python
def DoInference(method_name, problem_file_name,num_iteration):
```

  The parameter specifications are as follows:

  - `method_name`: specifies which method to be run for the given Query problem. It can be assigned to one of the values from `["ENUMERATION", "GIBBS"]`.
  - `problem_file_name`: specifies the path of the query file to be solved. File names take values such as "query1.txt", "query2.txt", "query3.txt".
  - `num_iteration`: specifies the number of iteration in Gibbs algorithm in Enumaration algorithm the value is 0.

  During the evaluation process, this function will be called and returned information will be inspected. The returned information should be as follows for both methods:

```python
Value_True,Value_False = DonInference(method_name, problem_file_name,
    num_iteration ))
```

  - Value_True,Value_False: are the values of the probability distribution for the possible outcomes.

- The query is described in ".txt" files and have the following structure:

```
[BayesNetNodes]
Node1
Node2
...
[Paths]
([Parents_of_Node1],'Node1')
([Parents_of_Node2],'Node2')
...
[ProbabilityTable]
('Node1',[Parents_ofNode1],{0.001}) if Node1 doesn not any parent
('Node2',[Parents_ofNode2],{True: 0.90, False: 0.05}) if Node2 has one
    parent
('Node3',[Parents_ofNode3],{(True, True): 0.001,(True, False): 0.002, (
    False, True): 0.003, (False, False): 0.004}) if Node3 has two parents
...
[Query]
('Burglary', {'JohnCalls': True, 'MaryCalls': True})
```

  Query information and algorithm parameters are provided in separate sections and each section is formed with '[]'. They are as follows:

  - `[BayesNetNodes]` section defines Nodes of Bayes net line by line.
  - `[Paths]` section specifies the parent of nodes line by line. If a node does not have parent than,there is no line for this node.
  - `[ProbabilityTable]` section provides each paths probability values line by line.

    – `[Query]` section specifies the query according to given Bayes Net.

- No `import` statement is allowed except for **copy** module with which you may perform a deep copy operation on a list/dictionary. All methods should be implemented in a single solution file. You are expected to implement all the necessary operations by yourself.

- Commenting is crucial for understanding your implementation and decisions made during the process. Your implementation can be manually inspected at random or when it does not satisfy the expected outputs.

- use randomṡeed(10), round(Value_True, 3) and round(Value_False, 3)

4 Sample I/O

Here, in addition to textual outputs, visual outputs for the solution of the query are provided. The actions are depicted via their symbol. The utility values are shown at the top left of the states.

query1.txt:

```
[BayesNetNodes]
Burglary
Earthquake
Alarm
JohnCalls
MaryCalls
[Paths]
(['Burglary', 'Earthquake'],'Alarm')
(['Alarm'],'JohnCalls')
(['Alarm'],'MaryCalls')
[ProbabilityTable]
('Burglary',[],{0.001,})
('Earthquake',[],{0.002,})
('JohnCalls',[Alarm],{True: 0.90, False: 0.05})
('MaryCalls',[Alarm],{True: 0.70, False: 0.01})
('Alarm', ['Burglary', 'Earthquake'], {(True, True): 0.95,(True, False): 0.94,
    (False, True): 0.29, (False, False): 0.001})
[Query]
('Burglary', {'JohnCalls': True, 'MaryCalls': True})
```

Expected outputs of the DoInference function with both methods applied on query1.txt:

```
import hw4solutioneXXXXXXX as hw4
>>> hw4.DoInference("ENUMERATION", "query1.txt",0)
(0.284,0.716)
>>> hw4.DoInference("GIBBS", "query1.txt",200)
(0.295,0.705)
```

query2.txt:

```
[BayesNetNodes]
Cloudy
Rain
WetGrass
Sprinkler
[Paths]
(['Cloudy'],'Rain')
(['Cloudy'],'Sprinkler')
(['Sprinkler','Rain'],'WetGrass')
[ProbabilityTable]
('Cloudy',[],{0.5,})
('Rain',[Cloudy],{True: 0.80, False: 0.20})
('Sprinkler',[Cloudy],{True: 0.1, False: 0.5})
('WetGrass', ['Sprinkler', 'Rain'], {(True, True): 0.99,(True, False): 0.90, (
    False, True): 0.90, (False, False): 0.00})
[Query]
('Cloudy', {'Rain': True})
```

Expected outputs of DoInference function with both methods applied on query2.txt:

```
import hw4solutioneXXXXXXX as hw4
>>> hw4.DoInference("ENUMERATION", "query2.txt",0)
(0.8,0.2)
>>> hw4.DoInference("GIBBS", "query2.txt",200)
(0.825,0.175)
```

5 Regulations

1. **Programming Language:** You must code your program in Python 3. Since there are multiple versions and each new version adds a new feature to the language, in order to concur on a specific version, please make sure that your implementation runs on İnek machines.

2. **Implementation:** You have to code your program by only using the functions in the standard module of python. Namely, you **cannot** import any module in your program (except the **copy** module).

3. **Late Submission:** No late submission is allowed. Since we have a strict policy on submissions of homework in order to be able to attend the final exam, please pay close attention to the deadlines.

4. **Cheating: We have zero-tolerance policy for cheating**. People involved in cheating (any kind of code sharing and codes taken from the internet included) will be punished according to the university regulations.

5. **Discussion:** You must follow ODTUClass for discussions and possible updates on a daily basis. If think that your question concerns everyone, please ask them on ODTUClass.

6. **Evaluation:** Your program will be evaluated automatically using the "black-box" technique so make sure to obey the specifications. A reasonable timeout will be applied according to the complexity of test cases. This is not about the code efficiency, its only purpose is avoiding infinite loops due to an erroneous code. In case your implementation does not conform to expected outputs for solutions, it will be inspected manually so commenting will be crucial for grading.

6 Submission

Submission will be done via OdtuClass system. You should upload a **single** python file named in the format **hw4_e<your-student-id>.py** (i.e. hw4_e1234567.py).

7 References

- Announcements Page
- Discussions Page