Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Department of Computer Science and Engineering (Data Science)

Subject: Probabilistic Graph Models (DJ19DSC402)

Bhuvi Ghosh
60009210191

AY: 2022-23

Experiment 5

(Implement exact inference in Bayesian Networks)

Aim: Implement exact inference in Bayesian Networks.

Theory:

What is a Bayesian Network?

A Bayesian Network (BN) is a Directed Acyclic Graph (DAG) whose nodes are random variables in a given domain and whose edges correspond intuitively to a direct influence of one node to another. A BN provides the skeleton for representing a joint probability distribution (JPD) in a factorized (simpler) way. BN is a compact representation of a set of conditional independence assumptions about a distribution.

More formally: A BN structure G is a DAG whose nodes represent random variables X1, …, Xn. Let PaG(Xi) denote the parents of Xi in G. The BN structure encodes the following set of conditional independencies assumptions, called local independencies, denoted by Il(G): For every variable Xi:

X : (Xi ⊥ None-Descendants(Xi) | PaG(Xi))

In other words, every variable is independent of all its none-descendants given its parents in the BN structure. This definition will allow us later to present the factorization theorem.

Example: The student's grade depends on his intelligence and the difficulty of the course. The student asks his professor for a recommendation letter; he only looks at his grade to write the letter (Also varies depending on his stress, mood, and/or other unknown stochastic events). According to the description, the following variables are involved:

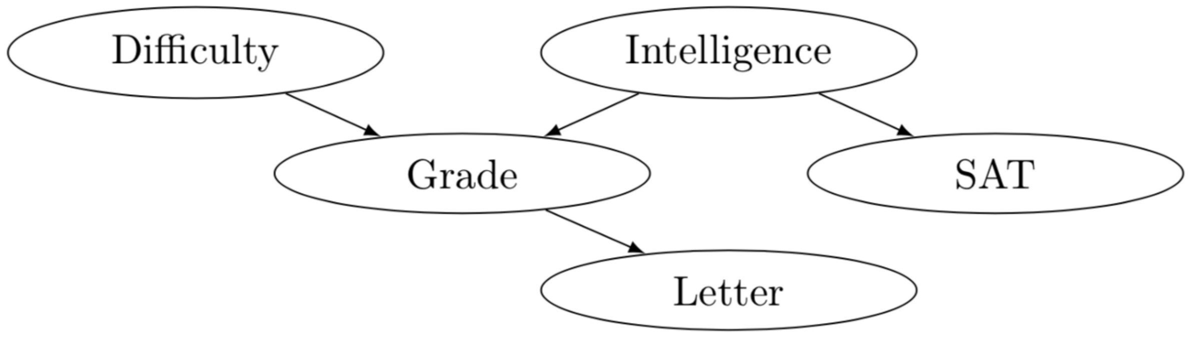
Difficulty: Val(difficulty)={easy, hard}={d0, d1}
Letter: Val(letter) = {week, strong}={l0, l1}

Intelligence: Val(intelligence) = {low, high} = {i0, i1}

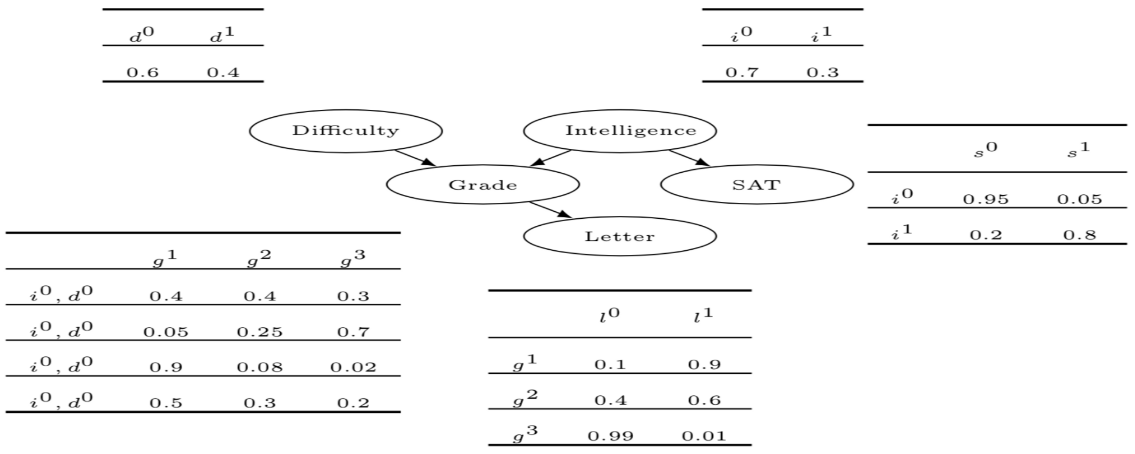
Grade: Val(Grade) = {low, medium, high} = {g1, g2, g3}

SAT: Val(SAT)={low, high}={s0, s1}

Note that the joint distribution P(Difficulty, Letter, Intelligence, Grade, SAT) has 48 possible entries (2 ∗ 2 ∗ 2 ∗ 2 ∗ 3). One possible Bayesian Network for this example is:



This BN structure allows us to factorize the joint distribution P(D,I,G,S,L) as P(D)P(I)P(G|D,I)P(S|I)P(L|G). Because each node is independent of its non-descendants given its parents, so you just need to go node by node and condition each distribution on the parents of the given node. Having fewer parents per node allows us to compute the full joint distribution as the product of simpler factors. Having the BN structure is not all. Besides that, we need to specify the distribution of each of those factors (parameters). The following figure shows an example:



d^0	d^1
0.6	0.4

i^0	i^1
0.7	0.3

	s^0	s^1
i^0	0.95	0.05
i^1	0.2	0.8

	g^1	g^2	g^3
i^0, d^0	0.4	0.4	0.3
i^0, d^0	0.05	0.25	0.7
i^0, d^0	0.9	0.08	0.02
i^0, d^0	0.5	0.3	0.2

	l^0	l^1
g^1	0.1	0.9
g^2	0.4	0.6
g^3	0.99	0.01

Note that there is one conditional probability table (CPT) for each of the nodes. Having all the necessary CPT's, we can calculate the probability of any joint assignment of the variables, for example:

$$\begin{aligned} P(i^1, d^0, g^2, s^1, l^0) &= P(i^1) * P(d^0) * P(g^2|i^1, d^0) * P(s^1|i^1) * P(l^0|g^2) \\ &= 0.3 * 0.6 * 0.08 * 0.8 * 0.4 = 0.004608 \end{aligned}$$

Reasoning in Bayesian Networks:

Showing an example to see how to calculate a conditional probability from the network when not all the variables are observed. The main idea is that we know the probabilities of every variable conditioned on its parents, so in case we do not have all the parents observed, we have to sum them up. The Bayes rule has to be used. For example, let's calculate the probability of i equals 1 given l equals 0:

$$P(i^1|l^0) = \frac{P(l^0|i^1)P(i^1)}{P(l^0)}$$

$$= \frac{\sum_G P(l^0|i^1, g^G)P(g^G|i^1)P(i^1)}{\sum_G P(l^0|g^G)P(g^G)}$$

Note that in the second step we add the respective parent variables (and sum them up), to be able to use the probability values provided in the CPTs. Next, we show how to calculate the numerator of the previous expression:

$$\sum_G P(l^0|i^1, g^G)P(g^G|i^1)P(i^1) = P(l^0|g^1)P(g^1|i^1)P(i^1) + P(l^0|g^2)P(g^2|i^1)P(i^1) + P(l^0|g^3)P(g^3|i^1)P(i^1)$$

Using the probability values mentioned in the above diagram we get:

$P(i^1) = 0.3$

$P(l^0|g^1) = 0.1$
$P(l^0|g^2) = 0.4$
$P(l^0|g^3) = 0.99$
$P(g^1|i^1) = \sum_D P(g^1|i^1, d^D)P(d^D|i^1)$
$P(g^1|i^1) = P(g^1|i^1, d^0)P(d^0|i^1) + P(g^1|i^1, d^1)P(d^1|i^1)$
$P(g^1|i^1) = 0.9 * 0.6 + 0.5 * 0.4 = 0.74$
$P(g^2|i^1) = P(g^2|i^1, d^0)P(d^0|i^1) + P(g^2|i^1, d^1)P(d^1|i^1)$
$P(g^2|i^1) = 0.08 * 0.6 + 0.3 * 0.4 = 0.168$
$P(g^3|i^1) = P(g^3|i^1, d^0)P(d^0|i^1) + P(g^3|i^1, d^1)P(d^1|i^1)$
$P(g^3|i^1) = 0.02 * 0.6 + 0.2 * 0.4 = 0.092$

Then the numerator becomes:

$$\sum_G P(l^0|i^1, g^G)P(g^G|i^1)P(i^1) = 0.1*0.74*0.3+0.4*0.168*0.3+0.99*0.092*0.3 = 0.069 \approx 0.07$$

NOTE: The denominator is not calculated. Usually the negation of the case is found out and the values are normalized. But the denominator can also be found out using the same method. As shown below:

$\sum_G P(l^0|g^G)P(g^G) = P(l^0|g^1)P(g^1) + P(l^0|g^2)P(g^2) + P(l^0|g^3)P(g^3)$

$P(l^0|g^1) = 0.1$
$P(l^0|g^2) = 0.4$
$P(l^0|g^3) = 0.99$

$P(g^G) = \sum_{D,I} P(g^G, d^D, i^I)$
$P(g^G) = \sum_D \sum_I P(g^G, d^D|i^I)P(i^I)$
$P(g^G) = \sum_D \sum_I P(g^G|d^D, i^I)P(d^D|i^I)P(i^I)$

$$P(g^1) = P(g^1|d^0, i^0)P(d^0|i^0)P(i^0) + P(g^1|d^0, i^1)P(d^0|i^1)P(i^1) + P(g^1|d^1, i^0)P(d^1|i^0)P(i^0) + P(g^1|d^1, i^1)P(d^1|i^1)P(i^1)$$

$$P(g^1) = 0.3 * 0.6 * 0.7 + 0.05 * 0.6 * 0.3 + 0.9 * 0.6 * 0.7 + 0.5 * 0.4 * 0.3 = 0.573$$
$$P(g^2) = 0.4 * 0.6 * 0.7 + 0.25 * 0.6 * 0.3 + 0.08 * 0.6 * 0.7 + 0.3 * 0.4 * 0.3 = 0.2826$$
$$P(g^3) = 0.3 * 0.6 * 0.7 + 0.7 * 0.6 * 0.3 + 0.02 * 0.6 * 0.7 + 0.2 * 0.4 * 0.3 = 0.2844$$

$$\sum_G P(l^0|g^G)P(g^G) = 0.1 * 0.573 + 0.4 * 0.2826 + 0.99 * 0.2844 = 0.4518$$

$$\therefore \quad P(i^1|l^0) = \frac{\sum_G P(l^0|i^1, g^G)P(g^G|i^1)P(i^1)}{\sum_G P(l^0|g^G)P(g^G)} = \frac{0.07}{0.4518} = 0.15$$

Inference over a Bayesian network:

The first is simply evaluating the joint probability of a particular assignment of values for each variable (or a subset) in the network. For this, we already have a factored form of the joint distribution, so we simply evaluate that product using the provided conditional probabilities. If we only care about a subset of variables, we will need to marginalize out the ones we are not interested in. In many cases, this may result in underflow, so it is common to take the logarithm of that product, which is equivalent to adding up the individual logarithms of each term in the product.

The second, more interesting inference task, is to find P(x|e), or, to find the probability of some assignment of a subset of the variables (x) given assignments of other variables (our evidence, e). In the above example, an example of this could be to find P(Sprinkler, WetGrass | Cloudy), where {Sprinkler, WetGrass} is our x, and {Cloudy} is our e. In order to calculate this, we use the fact that P(x|e) = P(x, e) / P(e) = αP(x, e), where α is a normalization constant that we will calculate at the end such that P(x|e) + P(¬x | e) = 1. In order to calculate P(x, e), we must marginalize the joint probability distribution over the variables that do not appear in x or e, which we will denote as Y.

$$P(x|e) = \alpha \sum_{\forall y \in Y} P(x, e, Y)$$

We would calculate P(¬x | e) in the same fashion, just setting the value of the variables in x to false instead of true. Once both P(x | e) and P(¬x | e) are calculated, we can solve for α, which equals 1 / (P(x | e) + P(¬x | e)).

Exact inference algorithms calculate the exact value of probability P(X|Y). Note that in larger networks, Y will most likely be quite large, since most inference tasks will only directly use a small subset of the variables. In cases like these, exact inference as shown above is very computationally intensive, so methods must be used to reduce the amount of computation. One more efficient method of exact inference is through variable elimination, which takes advantage of the fact that each factor only involves a small number of variables.

Lab Assignments to complete in this session

1) Using pgmpy feed the above cpds into a bayesian belief network.
2) Enter the condition that letter = 0, and see how intelligence is affected.
3) Cross-check the intelligence exact inference value with the one given above.

```
from pgmpy.models import BayesianNetwork
from pgmpy.factors.discrete import TabularCPD
# Defining the network structure
model = BayesianNetwork([("D", "G"), ("I", "G"), ("G", "L"), ("I", "S")])
# Defining the CPDs:
cpd_d = TabularCPD("D", 2, [[0.6], [0.4]])
cpd_i = TabularCPD("I", 2, [[0.7], [0.3]])
cpd_g = TabularCPD("G",3, [[0.4,0.05,0.9,0.5], [0.4,0.25,0.08,0.3], [0.2,0.7,0.02,0.2],], evidence=["I", "D"], evidence_card=[2, 2],)
cpd_l = TabularCPD("L",2, [[0.1,0.4,0.99], [0.9,0.6,0.01],], evidence=["G"], evidence_card=[3],)
cpd_s = TabularCPD("S",2, [[0.95,0.2], [0.05,0.8],], evidence=["I"], evidence_card=[2],)
# Associating the CPDs with the network structure.
model.add_cpds(cpd_d, cpd_i, cpd_g, cpd_l, cpd_s)
# Some other methods
model.get_cpds()
```

```
[<TabularCPD representing P(D:2) at 0x7f7e170c99d0>,
 <TabularCPD representing P(I:2) at 0x7f7e1bb64dd0>,
 <TabularCPD representing P(G:3 | I:2, D:2) at 0x7f7d8bb34ed0>,
 <TabularCPD representing P(L:2 | G:3) at 0x7f7e1b6c6f90>,
 <TabularCPD representing P(S:2 | I:2) at 0x7f7e1d0b6590>]
```

```
[ ] # Initializing the VariableElimination class
from pgmpy.inference import VariableElimination
model1 = VariableElimination(model)
```

```
[ ] # Computing the probability of intelligence given letter=0.
q = model1.query(variables=["I"], evidence={"L": 0})
print(q)
```

```
+------+----------+
| I    |   phi(I) |
+======+==========+
| I(0) |   0.8486 |
+------+----------+
| I(1) |   0.1514 |
+------+----------+
```