CS3243: Introduction to Artificial Intelligence **Fall 2020**

Lecture 11: November 4

Lecturer: Prof. Kuldeep S. Meel *Scribe: Ang Zheng Yong*

11.1 Bayesian Networks

Recall from last week, we discussed a problem which gave us the following Bayesian network:

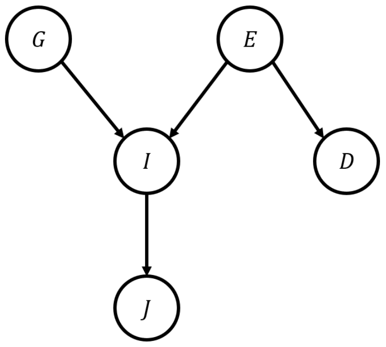

Figure 11.1: Bayesian network model for example

We also assumed that we have the following table of data:

G	E	I	freq.	Pr
T	T	T	160	$P(G, E, I) = 160/600$
T	T	F	60	$P(G, E, \bar{I}) = 60/600$
T	F	T	240	$P(G, \bar{E}, I) = 240/600$
T	F	F	40	$P(G, \bar{E}, \bar{I}) = 40/600$
F	T	T	10	$P(\bar{G}, E, I) = 10/600$
F	T	F	60	$P(\bar{G}, E, \bar{I}) = 60/600$
F	F	T	10	$P(\bar{G}, \bar{E}, I) = 10/600$
F	F	F	20	$P(\bar{G}, \bar{E}, \bar{I}) = 20/600$

Today, we will be discussing how are we going to fill up the entries of the conditional probability tables (CPTs) for each node from our data.

11.1.1 Filling up CPTs

Let us focus on a subset of nodes from Figure 11.1:

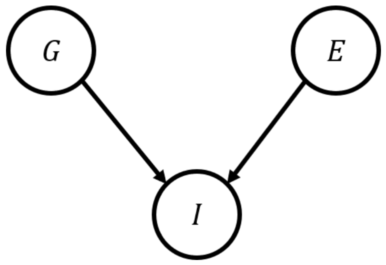

Figure 11.2: Subset of nodes from Figure 11.1

By processing the table from the previous page in a *row-by-row manner*, we will be able to fill up the CPTs for the nodes G, E, and I. The tables for each of the nodes are given below:

$\#G$	#total	$P(G)$
$160+60+240+40=500$	$\sum \text{freq} = 600$	$\frac{\#G}{\#\text{total}} = \frac{500}{600}$

$\#E$	#total	$P(E)$
$160+60+10+60=290$	$\sum \text{freq} = 600$	$\frac{\#E}{\#\text{total}} = \frac{290}{600}$

G	E	$\#I$	#total	$P(I\|G,E)$
T	T	160	$160+60=220$	$\frac{\#I}{\#\text{total}} = \frac{160}{220}$
T	F	240	$240+40=280$	$\frac{\#I}{\#\text{total}} = \frac{240}{280}$
F	T	10	$10+60=70$	$\frac{\#I}{\#\text{total}} = \frac{10}{70}$
F	F	10	$10+20=30$	$\frac{\#I}{\#\text{total}} = \frac{10}{30}$

Sometimes, we may be given data in factored form. In other words, we may receive the data separately for E and B. In this case, for each data point, we can go to each entry in our CPTs, update the total count, and calculate the probabilities accordingly.

On the other hand, if we encounter missing data, we may choose to use our background knowledge or prior understanding of the underlying distributions to assign approximate values to the entries.

11.1.2 Deciding on a model

Suppose that we believe that the mood of the driver, D, is not affected by the cost of ERP, E. Instead, we believe that the mood of the driver depends on the students' grades, G. In this case, our Bayesian network model will look as such:

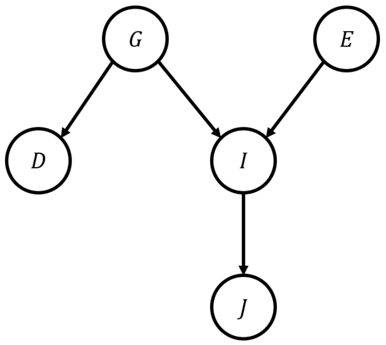

Figure 11.3: Bayesian network model for our new perception of the world

How do we determine which model is able to provide a more accurate description of the world? We first denote Figure 11.1 as M_1 and Figure 11.3 as M_2, and attempt to fill up the CPT entries for M_2. Then, we will collect a number of observations/evidence, and then compare between the models to determine which model provides a better description of the evidence collected.

For instance, we might have collected the following data:

$$e_1 : G = T,\ E = T,\ D = F,\ I = T,\ J = F$$
$$e_2 : G = T,\ E = F,\ D = T,\ I = T,\ J = F$$

To compare the between the models based on our evidence, we will be comparing between $P(M_1|\{e_1, e_2\})$ and $P(M_2|\{e_1, e_2\})$, where by Bayes' rule,

$$P(M_1|\{e_1, e_2\}) = \frac{P(\{e_1, e_2\}|M_1)P(M_1)}{P(\{e_1, e_2\})} \text{ and } P(M_2|\{e_1, e_2\}) = \frac{P(\{e_1, e_2\}|M_2)P(M_2)}{P(\{e_1, e_2\})}$$

Effectively, we are using *prior distributions* (i.e. $P(M_i)$) to update *posterior distributions* (i.e. $P(M_i|\text{evidence})$). We will now proceed to analyze how we could obtain each of the components on the R.H.S. of the equations.

- As discussed in the previous lecture, we can ignore the computation of the denominator, $P(\{e_1, e_2\})$, since both $P(M_1|\{e_1, e_2\})$ and $P(M_2|\{e_1, e_2\})$ share the same denominator.
- We may also assume that the data collected are independent of each other. Thus, we have

  $$\forall i,\ P(\{e_1, e_2\}|M_i) = P(e_1|M_i)P(e_2|M_i)$$

  Since $\forall i, j,\ P(e_j|M_i)$ can be computed easily using topological sort as taught in last week's lecture, where

  $$P(e_j|M_i) = P(G, E, D, I, J|M)$$

  thus the computation for $P(\{e_1, e_2\}|M_i)$ is trivial.
- The final component to consider is $P(M_1)$ and $P(M_2)$. How do we calculate these values?

11.1.3 Dealing with $P(M)$

$P(M_1)$ and $P(M_2)$ reflect the probabilities of each of the models occurring in real life. In fact, it is often difficult to obtain these probabilities, and there are several possible ways to do so:

1. We may choose to start with prior distributions for $P(M_1)$ and $P(M_2)$, based on our own past experiences.
2. We may also base the values of $P(M_1)$ and $P(M_2)$ on historical information from the usage of either of these models.

However, note that it is only necessary for us to figure out the ratio $P(M_1)/P(M_2)$, since we are only trying to compare the magnitude of $P(M_1|\{e_1, e_2\})$ with $P(M_2|\{e_1, e_2\})$.

11.1.4 Modelling

How do we come up with models for comparison? The idea is to start with some arbitrary model M, and move on to a new model M' by making some minor edits to our original model M. For instance, we may have an initial model M_a, given by

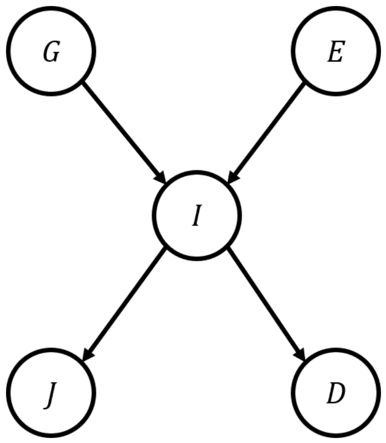

Figure 11.4: Model M_a

We can then make some minor edits to M_a, and arrive at a new model M_b.

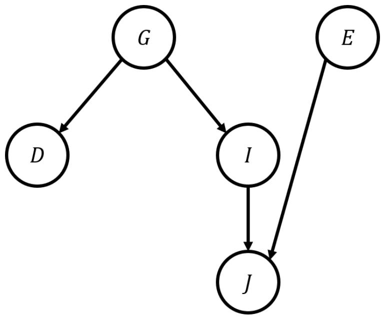

Figure 11.5: Model M_b

After coming up with a new model, we can compute the probability ratio of the two models, and keep the model which gives the higher probability ratio (i.e. the model that more accurately describes the evidence). In order to obtain a model that best describes our evidence, we can repeat this process of formulating a new model and comparing the probability ratio of the 2 models, until we reach some model which always gives a higher probabiliy ratio than other models.

Notice that this concept is exactly similar to local search. In local search, we also begin with an arbitrary state, and gradually transition to another state depending on the value assigned to each state.

Thus, the algorithm for obtaining an optimal model based on some evidence is as follows:

1. Initialize some random model.
2. At every step, look for a neighbour $N(M)$ of the current model.
3. Compute $\frac{P(\text{evidence}|N(M))}{P(\text{evidence}|M)}$, where $N(M)$ is a neighbour of model M.
4. Transition to a new model (or remain on the current model) based on the probability ratio that is computed.

Notice that we are computing $\frac{P(\text{evidence}|N(M))}{P(\text{evidence}|M)}$ instead of $\frac{P(\text{evidence}|N(M))P(N(M))}{P(\text{evidence}|M)P(M)}$. This is because we do not know that value of $\frac{P(N(M))}{P(M)}$, and thus we can assume that $P(N(M)) = P(M)$. Alternatively, we can assign a value based on our prior knowledge of the probability ratio of $\frac{P(N(M))}{P(M)}$.

11.2 Tutorial Discussion

Q: In the following Bayesian network, are F and G independent, given knowledge of the value of C?

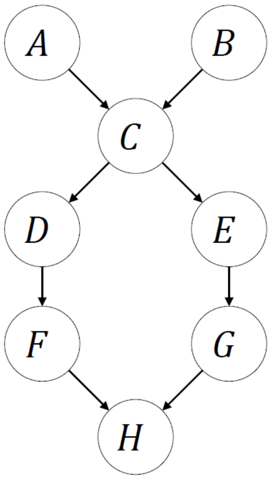

Figure 11.6: Bayesian network from Tutorial 10

In order to show independence, we would want to show that for some arbitrary values f, g, c_i,

$$P(F = f|G = g, C = c_i) = P(F = f|C = c_i)$$

$$\begin{aligned}
P(F = f|G = g, C = c_i) &= \sum_{d_i \in Dom(D)} P(F = f \cap D = d_i|G = g, C = c_i) && \text{axiom of probability}^1 \\
&= \sum_{d_i \in Dom(D)} P(D = d_i|G = g, C = c_i)P(F = f|D = d_i, G = g, C = c_i) && \text{defn of cond. prob.} \\
&= \sum_{d_i \in Dom(D)} P(D = d_i|C = c_i)P(F = f|D = d_i) && \text{axiom of BN} \\
&= \sum_{d_i \in Dom(D)} P(F = f \cap D = d_i|C = c_i) && \text{defn of cond. prob.} \\
&= P(F = f|C = c_i) && \text{axiom of probability}
\end{aligned}$$

$\square$

[1] $P(A = a) = P\left(\bigcup_{b \in Dom(B)} (A = a, B = b)\right) = \sum_{b \in Dom(B)} P(A = a, B = b)$

11.3 Knowledge Representation

Recall from week 2, we have discussed about a mopbot agent, which is able to move in the following state space:

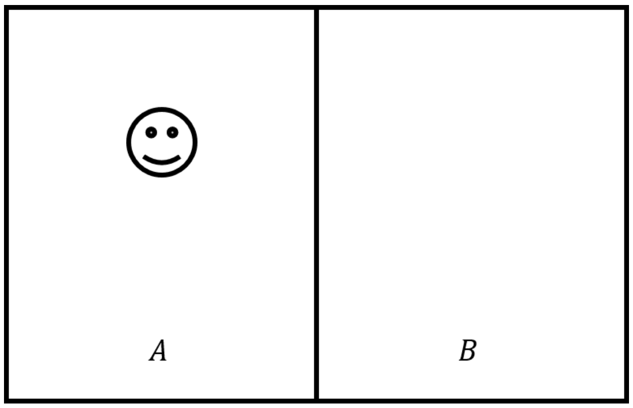

Figure 11.7: State space for mopbot, where the paggro face denotes the current location of the mopbot :)

Our mopbot has:

- Sensors, which enables it to detect whether there is dirt in the room.
- Actuators, which enables it to clean the room or move to another room.

Suppose that the mopbot is given the goal: "go to room B and leave after the room is clean". It might develop the following simple plan p_1 to achieve the goal:

1. Move to B.
2. Clean B.
3. Leave B when no dirt is detected.

Alternatively, the mopbot may also adopt a more complex plan p_2 to achieve the goal:

1. Move to B.
2. If dirt is detected, clean B. Otherwise, there is no need to clean.
3. Leave B when no dirt is detected.

However, how does the mopbot know that it has achieved its goal? The mopbot's plan consists of "no dirt $\implies$ leave room", and we know that "no dirt $\implies$ room is clean", but how does would the mopbot be able to deduce that when it has left the room, the room will be clean? In other words, how does the mopbot know that is has achieved the goal of "room is clean $\implies$ leave room"?

11.3.1 Propositional logic

Knowledge is a concept that is very hard to define, but one definition of knowledge that has been generally agreed upon is:

An agent A knows that statement S is true if and only if:

1. S is true.
2. A believes S is true.
3. A is justified in believing that S is true.

In Bayesian networks, we have to first come up with some model, before we decide whether something is true or false. Since the model is abstract, we have to rely on our beliefs to determine whether something is true or false, and thus we are not going to differentiate between points 1 and 2.

When given the statements "all Greeks are human" and "all humans are mortal", we can deduce that "all Greeks are mortal". However, when given "not all Greeks are human" and "all humans are mortal", we are not able to deduce that "not all Greeks are mortal". Why is this so?

We will now introduce the idea of *propositions*, which are variables that take on values *True* or *False*. According to George Boole (1854), we can decide whether or not we can make such deductions by replacing these statements with propositions. For instance, we can let

- g represent Grades,
- h represent Humans,
- m represent Mortals.

Thus, the first statement is equivalent to:

$$g \to h \wedge h \to m \implies g \to m$$

After introducing propositions, we need a way of combining them to construct meaningful sentences. This led to the introduction of operators, including:

- unary operators: $\neg$
- binary operations: $\wedge$ and $\vee$

Finally, we would want to be able to express our knowledge base. We thus express:

- PROP: the set of all propositions, and
- FORM: the set of all formulas that can be expressed by propositional logic.

Using these expressions, we will be able to classify which statements are allowed in propositional logic, and which are not allowed. For instance, $(p \vee q)$ is allowed under PROP and FORM, whereas $()p \wedge q$ is not allowed.

11.3.2 Recursive definition of FORM

FORM can be recursively defined as such:

$$\begin{aligned}
\mathrm{FORM}_0 &= \mathrm{PROP} \\
\mathrm{FORM}_{i+1} &= \mathrm{FORM}_i \cup \{(\alpha \circ \beta) | \alpha, \beta \in \mathrm{FORM}_i\} \cup \{(\neg\alpha) | \alpha \in \mathrm{FORM}_i\} \\
\mathrm{FORM} &= \bigcup_{i=0}^{\infty} \mathrm{FORM}_i
\end{aligned}$$

where $\alpha \circ \beta \equiv \{\alpha \vee \beta\} \cup \{\alpha \wedge \beta\}$.