Probabilistic Artificial Intelligence
Problem Set 2

Oct 11, 2019

1. Bayesian Networks: d-separation

As discussed in class, conditional independence properties entailed by a Bayesian Network can be read directly from the graph, using the notion of *d-separation*. Given the Bayesian network in Figure 1, which of the following conditional independence statements hold:

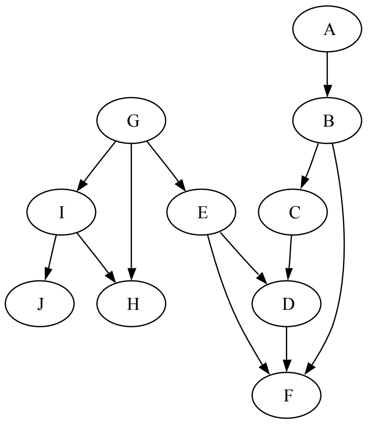

Figure 1: Bayes' net for Problem 1

1. $A \perp F$
2. $A \perp G$
3. $B \perp I \mid F$
4. $D \perp J \mid G, H$
5. $I \perp B \mid H$
6. $J \perp D$
7. $I \perp C \mid H, F$

2. Bayesian Networks: Variable elimination

Suppose you wish to do variable elimination on the Bayesian Network in Figure 1. Consider the variable ordering $A, B, C, D, E, F, G, H, I, J$. For each iteration of the algorithm (i.e., for each of the variables), determine which factors are removed and introduced. As an example, in the first iteration (variable A), the factor $P(A)$ and $P(B \mid A)$ are removed, and replaced by a new factor $g_1(B)$.

3. An algorithm for d-separation

In this exercise, you will implement an algorithm for computing d-separation of variables in Bayesian networks. In fact, the algorithm works in the opposite way by finding all variables that are *not* d-separated from a variable of interest. More formally, given a query variable X and a set of observed variables $\mathcal{Y} = \{Y_1, \ldots, Y_n\}$, the algorithm decides which variables could be dependent on X given $\mathcal{Y}$, that is, it returns the set of reachable variables

$$\mathcal{R}(X \mid \mathcal{Y}) = \{Z \mid Z \notin \text{d-sep}(X; Z \mid \mathcal{Y})\} .$$

You are provided some skeleton Python code in folder `hw2_code_problem` accompanying this document. Take the following steps for this exercise.

1. Install the Python dependencies listed in `README.txt`, if your system does not already satisfy them. After that, you should be able to run `demo.py` and produce some plots, albeit wrong ones for now.
2. Implement the missing code in `core.py` marked with `TODO`.
3. To check whether your implementation is correct, you can run the respective unittests in `tests.py`. Furthermore you can get some more intuition for your implemented code by running it on the v-structure in the demo file. Also, you can try out more example networks from `examples_dsep.py` to test your implementation.
4. Now, create the network shown in Figure 2 and answer the following questions:
   (a) Which variables are reachable from Radio, if nothing is observed?
   (b) Which variables are reachable from Radio, if Phone is observed?
   (c) Which variables are reachable from Radio, if both Phone and Earthquake are observed?

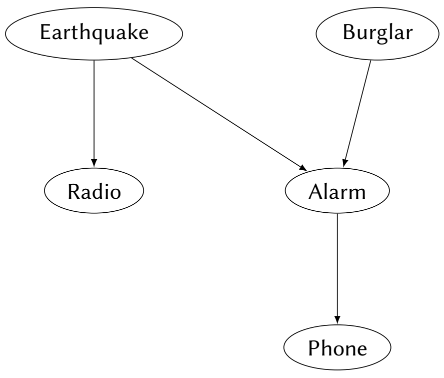

Figure 2: The earthquake network to be implemented.

4. Variable elimination

In this exercise you will use variable elimination to perform inference on a bayesian network. Consider the network in figure 3 and its corresponding conditional probability tables (CPTs).

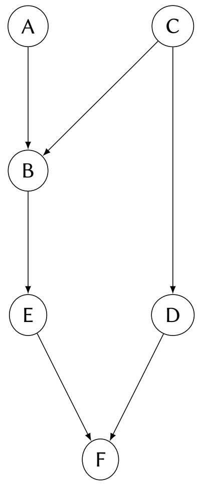

Figure 3: Bayesian network for problem 4.

$$P(A = t) = 0.3 \tag{1}$$
$$P(C = t) = 0.6 \tag{2}$$

Table 1: CPTs for problem 4.

(a) $P(B|A, C)$

A	C	$P(B = t)$
f	f	0.2
f	t	0.8
t	f	0.3
t	t	0.5

(b) $P(D|C)$

C	$P(D = t)$
f	0.9
t	0.75

(c) $P(E|B)$

B	$P(E = t)$
f	0.2
t	0.4

(d) $P(F|D, E)$

D	E	$P(F = t)$
f	f	0.95
f	t	1
t	f	0
t	t	0.25

Assuming a query on A with evidence for B and D, i.e. $P(A|B, D)$, use the variable elimination algorithm to answer the following queries. Make explicit the selected ordering for the variables and compute the probability tables of the intermediate factors.

1. $P(A = t|B = t, D = f)$
2. $P(A = f|B = f, D = f)$
3. $P(A = t|B = t, D = t)$

Consider now the ordering, C, E, F, D, B, A, use again the variable elimination algorithm and write down the intermediate factors, this time without computing their probability tables. Is this ordering better or worse than the one you used before? Why?