1 Bayesian Networks

Bayesian Networks are directed acyclic graphs (DAG) where the nodes represent random variables and directed edges capture their dependence. Consider the simplest graph

Figure 1: Simplest $A \longrightarrow B$ graph. A causes B or B is a consequence of A.

We would say that A is a parent of B, B is a child of A, that A influences, or causes B, B depends on A. Also, $P(A, B) = P(A)P(B|A)$.

The independence of two nodes in a DAG depends on their relative position in the graph as well as on the knowledge of other nodes in the graph. The following simple example illustrates the influence of conditioning on the independence.

Example 1. Let X_1 and X_2 be results of flips of two fair coins, and X_3 an indicator if the values X_1 and X_2 coincide. Thus, $P(X_1 = H) = P(X_1 = T) = 0.5$ and $P(X_2 = H) = P(X_2 = T) = 0.5$.

$$\boxed{X_1} \longrightarrow \boxed{X_3} \longleftarrow \boxed{X_2}$$

We are interested in $P(X_3 = \text{'yes'}|X_1, X_2)$. The nodes X_1 and X_2 are marginally independent (when we do not have evidence on X_3) but become dependent if the value of X_3 is known,

$$\frac{1}{2} = P(X_1 = T, X_2 = T|X_3 = 1) \neq P(X_1 = T|X_3 = 1)P(X_2 = T|X_3 = 1) = \frac{1}{2} \cdot \frac{1}{2}.$$

Hard evidence for a node X is evidence that the state of X is takes a particular value.

The notion of d-separation

1. In a serial connection from X_1 to X_3 via X_2, evidence from X_1 to X_3 is blocked only when we have hard evidence about X_2.

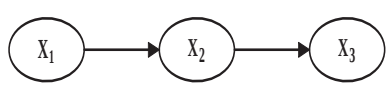

Figure 2: Case 1.

2. In a diverging connection where X_1 and X_3 have the common parent X_2 evidence from X_1 to X_3 is blocked only when we have hard evidence about X_2.

3. In a converging connection where X_3 has parents X_1 and X_2 any evidence about X_3 results in evidence transmitted between X_1 and X_2.

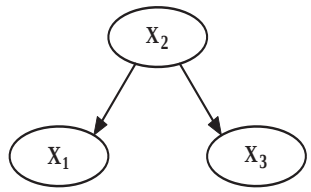

Figure 3: Case 2.

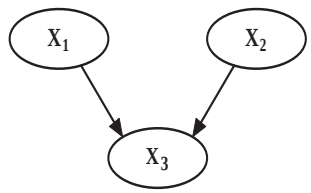

Figure 4: Case 3.

In Cases **1** and **2** we say that the nodes X_1 and X_3 are d-separated when there is hard evidence about X_2. In Case **3**, X_1 and X_2 are only d-separated when there is no evidence about X_3. In general two nodes which are not d-separated are said to be d-connected.

These three cases enable us to determine in general whether any two nodes in a given BN are dependent (d-connected) given the evidence entered in the BN. Formally:

Definition of d-separation: Two nodes X and Y in a BN are d-separated if, for all paths between X and Y, there is an intermediate node A for which either:

1. the connection is serial or diverging and the state of A is known for certain; or

2. the connection is diverging and neither A (nor any of its descendants) have received any evidence at all.

The problem of exact probabilistic inference in an arbitrary Bayes network is NP-Hard.[Cooper 1988] NP-Hard problems are at least as computational complex as NP-complete problems No algorithms has ever been found which can solve a NP-complete problem in polynomial time Although it has never been proved whether P = NP or not, many believe that it indeed is not possible. Accordingly, it is unlikely that we could develop an general-purpose efficient exact method for propagating probabilities in an arbitrary network

L-S algorithm is an efficient exact probability inference algorithm in an arbitrary Bayes Network, [2]

To find approximate solution one may sample from the BN and approximate needed probabilities by the corresponding empirical relative frequencies.

Here is an approach to sample from a Bayes net, called forward sampling.

1. Insert any evidence we have, by instantiating the relevant nodes.

2. Find the nodes that have no parents, and generate instantiations according to the (unconditonal) CPT for those nodes.

3. Generate values for their children conditional on whatever values we have just created for the parents.

4. Keep going “down” the net until we have generated a complete sample by generating values for children given the current values of their parents.

5. If, at any time, we generate a value for an evidence node that does not agree with the evidence at that node then abort and start from scratch.

The problem with this approach is that, if the evidence you inserted was unlikely, you will have to throw

away vast numbers of samples. If you have no evidence, then forward sampling is entirely sensible.

2 Myrphy's BNT Kit, Alarm Example and CS 8803B Project at Tech

This section is adapted from Project 3 in course CS 8803B Artificial Intelligence taught in Fall 2002 and 2003 here at Georgia Tech. For details, consult:
`http://www.cc.gatech.edu/classes/AY2003/cs8803b_fall/`

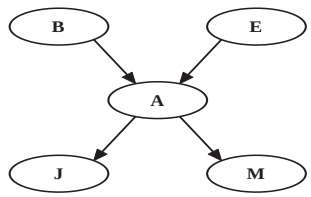

Figure 5: Alarm Example

BNT for Bayesian reasoning Here we describe how to use BNT and Matlab to perform Bayesian reasoning on a simple belief network (this example is taken from: Artificial Intelligence: A Modern Apprroach; S. Russell and P. Norvig, Prentice Hall, 1995., chapter 15–a diagram of the network appears in figure 15.2 on page 439). First, start Matlab, and load up the BNT toolkit.

```
>> addpath 'your path to BNT'
>> add_BNT_to_path
```

Next, we must construct an adjacency matrix for the directed acyclic graph corresponding to the belief net. We must number the nodes in topological order (parents before children, grandparents before grandchildren, etc.). Here, we have chosen to use the variables B, E, A, J, and M to refer to the nodes Burglary, Earthquake, Alarm, JohnCalls, and MaryCalls, respectively. In Matlab, arrays and matrices are indexed from 1, so the first node must be numbered 1.

```
>> N = 5;
>> dag = zeros(N,N);
>> B = 1; E = 2; A = 3; J = 4; M = 5;
>> dag([B,E],A) = 1;
>> dag(A,[J,M]) = 1;
>> dag
dag =
     0     0     1     0     0
     0     0     1     0     0
     0     0     0     1     1
     0     0     0     0     0
     0     0     0     0     0
```

Next, we construct the actual belief network object. The node_sizes variable refers to the size of the domains of each variable. This example uses boolean nodes, so the size is only 2, but it is possible to have, say, 4 or 6 values in the domain of a variable.

```
>> node_sizes = 2*ones(1,N);
>> bnet = mk_bnet(dag, node_sizes, 'names', {'B','E','A','J','M'});
```

You can visualize the graph structure of a belief net using the following command:

```
>> draw_graph(bnet.dag);
```

This will pop up a window with a drawing of the graph structure of your network. The nodes will be numbered according to your scheme (not labeled), so you must know the numbering scheme to interpret this drawing. You must number the values of the domains much like you number the nodes in the network, and once again Matlab needs 1-based indexing (more on this below). In our case, we use index 1 to represent false and index 2 to represent true. This is arbitrary, but it must be consistent. The reason for these indices is when we go to specify the conditional probability tables (or CPTs) for each node, our scheme will be used to index the values in the table.

```
>> false = 1; true = 2;
```

Here is our first two CPTs, for nodes B and E. We only have a prior probability, here, so we need to specify CPT(false) and CPT(true), yielding a 2-element vector of the form $[P(F)\ \ P(T)]$:

```
>> bnet.CPD{B} = tabular_CPD(bnet, B, [.999, .001]);
>> bnet.CPD{E} = tabular_CPD(bnet, E, [.998, .002]);
```

Here, the prior $P(B) = .001$, and the prior $P(E) = .002$. The CPT for A is more complicated, since it has two parents (B and E). The CPT will take three indices, one for each parent node, and one for the value of the node itself. The order of the indices is the numerical order of the nodes. In this case, B, E, and A are 1, 2, and 3, so the first index corresponds to the value of B, the second the value of E, and the third the value of A. So, for instance CPT(2,2,2) is the conditional probability $P(A|B, E)$, which is .95 in our example. We do not need to specify the 3-dimensional vector, however; we can use a 1-dimensional vector and BNT will "reshape" it for us. Matlab increments the left-most indices first, so the order will be as follows:

```
B E | A | P(A|B,E)
------------------
F F | F | .999
T F | F | .06
F T | F | .71
T T | F | .05
F F | T | .001
T F | T | .94
F T | T | .29
T T | T | .95
```

Recall that true (T) is 2 and false (F) is 1. This gives us the CPT for A:

```
>> bnet.CPD{A} = tabular_CPD(bnet, A, [.999, .06, .71, .05, .001, .94, .29, .95]);
```

The tables for the last two nodes, J and M, can be computed similarly. Since there is only one parent for each of them, the vector will be of length four. For instance, for J it will contain $[P(J|A)\ \ P(J|A)\ \ P(J|A)\ \ P(J|A)]$.

```
>> bnet.CPD{J} = tabular_CPD(bnet, J, [.95, .10, .05, .90]);
>> bnet.CPD{M} = tabular_CPD(bnet, M, [.99, .30, .01, .70]);
```

Now for the inference. First we set up the inference engine. Here we are using the junction tree inference method, which is an exact method of inference for any belief net (not just polytrees).

```
>> engine = jtree_inf_engine(bnet);
```

Here we set up the evidence for a diagnostic query. A diagnostic inference is a bottom-up inference from effects to causes. Here, we are interested in the probability of a burglary given that John has called, $P(B|J)$.

```
>> evidence = cell(1,N);
>> evidence{J} = true;
```

The following commands actually compute the answer:

```
>> [engine, loglik] = enter_evidence(engine, evidence);
>> marg = marginal_nodes(engine, B);
>> marg.T(true)
ans =
    0.0163
```

The last command returns the answer (1.6%). Actually, `marg.T` is a table, indexed by the possible domain values for the node B. It is the posterior distribution for the node B given our evidence:

```
 >> marg.T
ans =
    0.9837
    0.0163
```

Next, we do a causal inference. Causal inferences reason top-down from causes to effects. Here, we ask for the likelihood that Mary calls given that there is a burglary, $P(M|B)$. The evidence:

```
>> evidence = cell(1,N);
>> evidence{B} = true;
```

And the inference:

```
>> [engine, loglik] = enter_evidence(engine, evidence);
>> marg = marginal_nodes(engine, M);
>> marg.T
ans =
    0.3414
    0.6586
```

Here, $P(M|B)$ is 65.9%. A simple inter-causal inference is to ask what the likelihood of one causal node having a particular value is, given the value of another. Here, we ask for the likelihood of a burglary given an earthquake (and this simple network):

```
>> evidence = cell(1,N);
>> evidence{E} = true;
>> [engine, loglik] = enter_evidence(engine, evidence);
>> marg = marginal_nodes(engine, B);
>> marg.T(true)
ans =
   1.0000e-03
```

The answer returned is 0.1% (low, as we would expect). This is not a terribly interesting query. Often times, inter-causal inference is known as “explaining away”, and this next example illustrates this pattern more fully. First, we do a simple diagnostic query asking for the probability of a burglary given that the alarm went off. Next, we revise it with evidence that an earthquake occurred. This lowers the answer considerably, hence “explaining away” the alarm.

```
>> evidence = cell(1,N);
>> evidence{A} = true;
>> [engine, loglik] = enter_evidence(engine, evidence);
>> marg = marginal_nodes(engine, B);
>> marg.T(true)
ans =
    0.3736
```

We see that the answer is 37.4%. Next, we add the evidence of an earthquake, and recompute:

```
>> evidence{E} = true;
>> [engine, loglik] = enter_evidence(engine, evidence);
>> marg = marginal_nodes(engine, B);
>> marg.T(true) % returns P(B|A,E)
ans =
    0.0033
```

The probability drops to 3.3%, hence "explaining away" the alarm.

Approximate Inference in Bayes Nets

The Markov Blanket for node X_i is a set of all parents of X_i, children of X_i, and spouses of X_i (other parents of X_is children).

MCMC works well in Bayes Nets. The key step is sampling

$$P(X_i|X_{\neq i})$$

in a particular enumeration. In Bayes nets any variable X_i is independent of variables outside of Markov Blanket of X_i,

$$P(X_i|X_{\neq i}) = P(X_i|\text{Markov Blanket}(X_i)).$$

Exercise. Show that

$$P(X_i|\text{Markov Blanket}(X_i)) \propto P(X_i|\text{Parents}(X_i)) \times \prod_{i=1}^{k} P(Y_i|\text{Parents}(Y_i)),$$

where $Y_i,\ i = 1, 2, \ldots, k$ are children of X_i.

Sprinkler Example

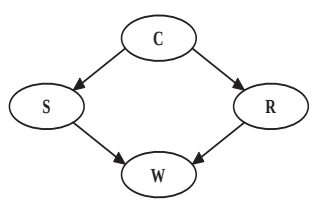

Figure 6: Sprinkler

To be added!

Alarm Example Again

We have seen `alarm` example in Murphy's BNT. Here we discuss some exact calculations and use of BUGS software for approximate calculations.

Assume your house has an alarm system against burglary. You live in the seismically active area and the alarm system can get occasionally set off by an earthquake. You have two neighbors, Mary and John, who do not know each other. If they hear the alarm they call you, but this is not guaranteed. They also call you from time to time just to chat.

Asia Example

Lauritzen and Spiegelhalter (1988) introduce a fictitious *expert system* representing the diagnosis of a patient presenting to a chest clinic, having just come back from a trip to Asia and showing *dyspnoea* (shortness-of-breath). A graphical model for the underlying process is shown in the Figure 7,

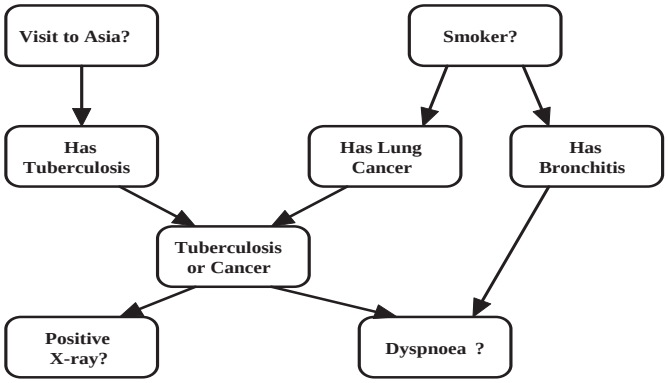

Figure 7: Lauritzen and Spiegelhalter (1988) Asia Bayes Net: A fictitious *expert system* representing the diagnosis of a patient, having just come back from a trip to Asia and showing *dyspnoea*.

where each variable is binary. The BUGS code is shown below and the conditional probabilities used are given in Lauritzen and Spiegelhalter (1988).

```
model Asia;
{
   asia         ~ dcat(p.asia);
   smoking      ~ dcat(p.smoking[]);
   tuberculosis ~ dcat(p.tuberculosis[asia,]);
   lung.cancer  ~ dcat(p.lung.cancer[smoking,]);
   bronchitis   ~ dcat(p.bronchitis[smoking,]);
   either      <- max(tuberculosis,lung.cancer);
   xray         ~ dcat(p.xray[either,]);
   dyspnoea     ~ dcat(p.dyspnoea[either,bronchitis,])
}
```

list(p.asia = c(0.99, 0.01), p.tuberculosis = structure(.Data = c(0.99,0.01,0.95,0.05), .Dim = c(2,2)), p.bronchitis = structure(.Data = c(0.70,0.30,0.40,0.60), .Dim = c(2,2)), p.smoking = c(0.50,0.50), p.lung.cancer = structure(.Data = c(0.99,0.01,0.90,0.10), .Dim = c(2,2)), p.xray = structure(.Data = c(0.95,0.05,0.02,0.98), .Dim = c(2,2)), p.dyspnoea = structure(.Data = c(0.9,0.1, 0.2,0.8, 0.3,0.7, 0.1,0.9), .Dim = c(2,2,2)))

list(asia = 2, dyspnoea = 2)

References

[1] Howard, Ronald A. (1983). The used car buyer, in *The Principles and Applications of Decision Analysis*: Vol. II, Ronald A. Howard and J. E. Matheson (eds.), Strategic Decisions Group, Menlo Park, CA. Originally copyright 1962.

[2] Lauritzen, Steffen L. and David J. Spiegelhalter (1988). Local computations with probabilities on graphical structures and their application to expert systems. *J. Royal Statistics Society B*, **50**, 157–194.

[3] Pearl, Judea (1988) Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann, San Mateo, CA. 2nd edition 1991.

[4] Spiegelhalter, David J. (1986). Probabilistic reasoning in predictive expert systems, in *Uncertainty in Artificial Intelligence*, L. N. Kanal and J. F. Lemmer (Eds.), North-Holland, Amsterdam, pp. 47-67.

[5] Spiegelhalter, David J., A. Philip Dawid, Steffen L. Lauritzen and Robert G. Cowell (1993) "Bayesian analysis in expert systems" in Statistical Science, 8(3), 219-283.

Exercises

1. d**-Separation.** Consider the network as in Figure 8

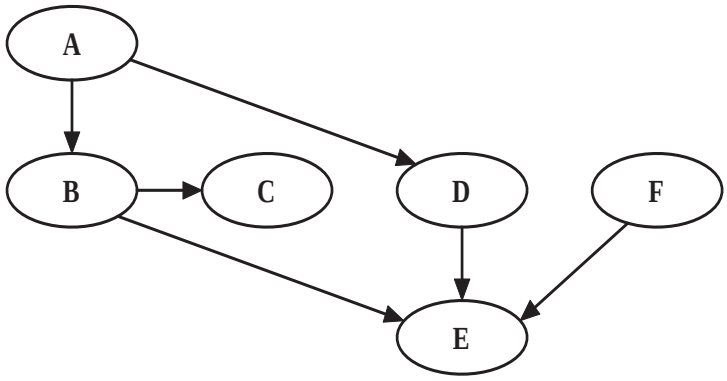

Figure 8: Exercise for d-separation.

(i) Does D d-separate C and F? There are two undirected paths from C to F: (i) $C - B - E - F$. This blocked given D by the node E, since this is not one of the given nodes (i.e., is not D) and has both arrows on the path going into it. (ii) $C - B - A - D - E - F$. This path is also blocked by E (and D as well). So, D does d-separate C and F.

(ii) Do D and E d-separate C and F? The path $C - B - A - D - E - F$ is blocked by the node D given D, E. However, the path $C - B - E - F$ is not blocked: E no longer blocks this path since it is now one of the "given" nodes. So, D and E do not d-separate C and F.

(iii) Write down all pairs of nodes which are independent of each other. Nodes which are independent are those that are d-separated by the empty set of nodes. This means every path between them must contain at least one node with both path arrows going into it. Clearly E is the only candidate for this.

We find that F is independent of A, of B, of C and of D. All other pairs of nodes are dependent on each other.

(iv) Which pairs of nodes are independent of each other given B? We need to find which nodes are d-separated by B. A, C and D are all d-separated from F because of the node E. In fact, C is d-separated from all the other nodes (except B) given B. That is the lot. (A and E are not d-separated by B, because of the path through D). The independent pairs given B are hence: AF, AC, CD, CE, CF, DF.

(v) Do we have that: $P(A, F|E) = P(A|E)P(F|E)$? (i.e., are A and F independent given E?) A and F are NOT independent given E, since E does not d-separate A and F. (Both paths from A to F do not contain a blocking node given E).

2. Example Accident Proneness. Feller [1968]. Imagine a population with two types of individuals: N normal, and N^c, accident prone. And suppose that 5/6 of these people are normal, so that if we randomly select a person from this population the probability that the chosen person is normal is $P(N) = 5/6$. Let A_i be the event that an individual has an accident in year i. For each individual A_i is independent of A_j whenever $i \neq j$. Thus for each individual, whether or not that person has an accident follows a Bernoulli process. The accident probability, however, is different for the two classes of individuals. $P(A_i|N) = .01, P(A_i|N^c) = .1$ The chance of a randomly chosen individual having an accident in a given year follows from the Law of Total Probability

$$P(A_i) = P(A_i|N)P(N) + P(A_i|N^c)P(N^c) = .05/6 + .1/6 = 1.5/6 = .025.$$

The probability that a randomly chosen individual has an accident in both the first and second year follows from the Law of Total Probability and the fact that A_1 and A_2 are independent for a given individual

$$\begin{aligned} P(A_1 \cap A_2) &= P(A_1 \cap A_2|N)P(N) + P(A_1 \cap A_2|N^c)P(N^c) \\ &= P(A_1|N)P(A_2|N)P(N) + P(A_1|N^c)P(A_2|N^c)P(N^c) \\ = .01 \times .01 \times 5/6 + .1 \times .1 \times 1/6 &= .0005/6 + .01/6 = .0105/6 = .00175. \end{aligned}$$

Note that:

$$P(A_2|A_1) = P(A_1 \cap A_2)P(A_2) = .00175/.025 = .07.$$

Therefore A_1 and A_2 are not (unconditionally) independent!