Name:

Assignment # **7** Due: **29 Oct 95** 6.034 Item # **16**

Please indicate the section you attend, or if you do not attend a section, put a tick mark on the last line.

Time	Room	Instructor	TA	Tick
10–11	34-303	Randy Davis	——	
10–11	34-302	Tomás Lozano-Pérez	Gina Levow	
11–12	36-156	Randy Davis	——	
11–12	34-302	Tomás Lozano-Pérez	Gina Levow	
12–1	34-303	Ken Yip	Hyung Chang	
1–2	34-302	Ken Yip	Hyung Chang	
1–2	34-301	Peter Szolovits	Andreas Argiriou	
2–3	34-304	Peter Szolovits	Andreas Argiriou	
2–3	34-301	Polly Pook	Jeff Breidenbach	
3–4	34-303	Polly Pook	Jeff Breidenbach	
—	—	—		

Massachusetts Institute of Technology

6.034 Artificial Intelligence

Assignment # **7** Due: **29 Oct 95** 6.034 Item # **16**

This problem set is handed out on Tuesday, October 22, 1996, and due one week later. It overlaps slightly the due date of problem set 6, which is due on Thursday, October 24.

Goals

This assignment is meant to teach you about methods of reasoning with uncertainty. The main part focuses on *Bayes Networks*, a graphical formalism that represents probabilistic relationships among random variables. These methods can be used in many different kinds of reasoning tasks in AI, but here we focus on diagnostic applications.

Background

Much of what we know about the world is known with various degrees of certainty. Although many different numerical models have been proposed for representing this uncertainty, many researchers today believe that probability theory—the most extensively studied uncertainty calculus—is an appropriate basis for building computerized reasoning systems.[1] We assume that you have read (and understood) the paper "Uncertainty and Decisions in Medical Informatics" by Szolovits, handed out Monday, which reviews much of the required background.

Mathematical Preliminaries

We take the world of interest to consist of a set of random variables, $\mathcal{X} = \{X_1, X_2, \ldots, X_n\}$. Each of the X_i can take on one of a discrete set of values, $\{x_{i1}, x_{i2}, \ldots, x_{ik_i}\}$.[2] Each possible combination of assignments of values to each of the variables represents a possible state of this world. There are $\prod_{i=1}^{n} k_i$ such states. This is clearly exponential in the number of variables; for example, if each variable is binary, all of the $k_i = 2$ and the number of states is

[1] The other major alternatives pursued by some include *fuzzy set theory* and *Dempster-Shafer theory*.

[2] Formulations using continuous variables are also possible, but we will not pursue them here. A continuous variable can, of course, be approximated by a large number of discrete values.

2^n. Each state may be identified with the particular values that it assigns to each variable. We might have, for example, $S_{12} = \{X_1 = a, X_2 = e, \ldots, X_n = b\}$.

A probability function, P, assigns a probability to each of these possible states. The probability for each state, $P(S_i)$, is the *joint probability* of that particular assignment of values to the variables. All states are distinct, and they exhaustively enumerate the possibilities; therefore,

$$\sum_{j=1}^{\prod_{i=1}^{n} k_i} P(S_j) = 1.0$$

We will often be interested in the probability not of individual states, but of certain combinations of particular variables taking on particular values. For instance, we may be interested in the probability that $\{X_3 = a, X_5 = b\}$. In such cases, we wish to treat other variables as "don't cares." We call such a partial description of the world a *circumstance*,[3] C, and the variables that have values assigned the *instantiation-set* of the circumstance, $I(C)$. The probability of a circumstance C can be computed by summing the probabilities of all states that assign the same values to the variables in the instantiation set $I(C)$ as C. If we re-order variables $\mathcal{X}$ so that the first m are the instantiation set of C, then (in a shorthand notation):

$$P(C) = \sum_{X_{m+1}} \ldots \sum_{X_n} P(S).$$

For example, if the world consists of four binary variables, W, X, Y and Z, then the circumstance $\{X = \text{true}, Z = \text{false}\}$ is given by

$$P(\{X = \text{true}, Z = \text{false}\}) = \sum_{u \in \{\text{true,false}\}} \sum_{v \in \{\text{true,false}\}} P(\{W = u, X = \text{true}, Y = v, Z = \text{false}\})$$

The computational difficulty of calculating the probability of a circumstance comes precisely from the need to sum over a possibly vast number of states. The number of such states to sum over is exponential in the number of "don't cares" in a circumstance.

Defining P for each state of the world is a rather tedious and unintuitive way to describe probabilities in the world, although the ability to assign an exponentially large number of independent probabilities would allow any probability distribution to be described. It might happen that the world really *is* so complex that only such an exhaustive enumeration of the probability of each state is adequate, but fortunately many variables appear to be *independent*. For example, in medical diagnosis, the probability that you get a strep infection is essentially independent of the probability that you catch valley fever (a fungal infection of the lungs prevalent in California agricultural areas). Formally, this means that $P(\text{strep}, \text{vf}) = P(\text{strep})P(\text{vf})$. When two variables both depend on the same set of variables but not directly on each other, they are *conditionally independent*. For example, if an

[3] "Circumstance" is not a commonly-used term for partial descriptions of the world. People use terms such as "partially-specified state" and other equally unsatisfying terms.

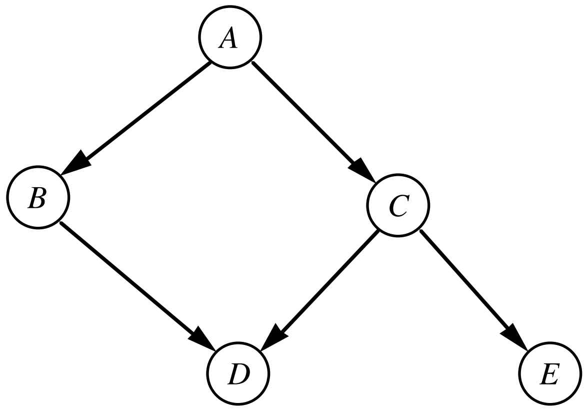

Figure 1: The MCBN1 network shows probabilistic dependencies among the variables A, B, C, D and E. Numerical details are not shown.

infection causes both a fever and diarrhea, but there is no other correlation between these symptoms, then $P(\text{fever}, \text{diarrhea}|\text{infection}) = P(\text{fever}|\text{infection})P(\text{diarrhea}|\text{infection})$.

The independencies among variables in a domain support a convenient graphical notation, called a *Bayes network*. In it, each variable is a node, and each probabilistic dependency is drawn as a directed arc to a dependent node from the node it depends on. This notion of probabilistic dependency does not necessarily correspond to what we think of as causal dependency, but it is often convenient to identify them. When there is no directed arc from one variable to another, then we say that the second does not depend on the first. For example, in the network in Figure 1, E depends only on C. Therefore the influence of any of the other variables on E can impinge only by changing the probability of C. For each node X, we call those nodes from which directed arcs exist in the network to X the *parents* of X, written $\pi(X)$. Some nodes have no parents ($\pi(X) = \emptyset$); they have only an *a priori* probability and do not depend on others.

In this representation, instead of assessing joint probabilities for each state, we define the probability distribution by writing down for each node X

$$P(X|\pi(X))$$

For MCBN1, this requires $P(A)$, $P(B|A)$, $P(C|A)$, $P(D|B,C)$ and $P(E|C)$. Each of these is actually a multidimensional array, one dimension corresponding to each variable in the conditional probability, and the final dimension spanning the possible values of that variable. Because all of the MCBN1 variables are binary, this means $P(A)$ holds just two values, corresponding to the probability that $A = \text{true}$ and $A = \text{false}$, and $P(D|B,C)$ is a $2 \times 2 \times 2$ array. Note that for any particular set of values for parent variables, the probabilities for all possible values of the variable associated with a node must sum to one. Thus, in the case of the $2 \times 2 \times 2$ array $P(D|B,C)$, we need to determine only 2×2 values: the probabilities that D is true under the four possible circumstances defined by the possible values of the parents. In each circumstance, the probability that $D = \text{false}$ is just $1 - P(\{D = \text{true}\}|B,C)$. In general, if a node has p parents, each of which can take on k values, and if the node itself can take on k values, then the total number of parameters to determine is $(k-1)k^p$. One way to (drastically) reduce the number of parameters that need to be estimated is to assume that the parents influence the variable at this node in some simpler, systematic way. For example, if we believe that the influence of the parents on this node may be modeled by a *noisy-or*—i.e., each parent is a (probabilistic) cause of this node and that these causal influences are independent—then we need only p parameters to characterize the node if all variables are binary.

The probability of a state is a product over all variables of the probability that the variable takes on its particular value (in that state) given that its parents take on their particular values:

$$P(X_1 = v_1, \ldots, X_n = v_n) = \prod_{i=1,\ldots,n} P(X_i | \pi(X_i))$$

The right hand side is an abbreviation for

$$\prod_{i=1,\ldots,n} P(X_i | \pi(X_i)) = \prod_{i=1,\ldots,n} P(X_i = v_i | X_j = v_j, \ldots, X_l = v_l)$$

where $\pi(X) = \{X_j, \ldots, X_l\}$. As described above, to find the probability of a circumstance, we must still sum over all of the states that are consistent with the circumstance—i.e., a number of states exponential in the number of "don't cares."

Much of what is complicated about performing inference in Bayes networks is that we cannot afford to do the computations just as they are described above. Instead, we seek to find efficient decompositions of the networks so that computation can be done more efficiently. As usual, even the problem of finding such optimal decompositions is *NP*-hard, so in practice all we can hope for is a reasonably efficient strategy for finding reasonably good decompositions. In the code used in this assignment, we employ an implementation of an algorithm developed by Greg Cooper, who was at Stanford at the time,[4] and implemented by Peter Szolovits, originally in Common Lisp.

Code Preliminaries

The programs for this assignment are contained in the following files:

1. `load-ps7.sc` simply loads up the other files.

2. `cooper.sc` implements most of the programs, including convenient ways to define Bayes networks, to decompose them for efficient computation, and to perform the computations.

3. `utils.sc` contains (almost exactly) the utilities also used in Problem Set 6.

[4] Cooper, G. F. (1990). Bayesian Belief-network Inference Using Recursive Decomposition (Technical Report No. KSL 90-05). Stanford University.

4. `structs.sc` defines the data structures for Bayes networks, nodes in such a network, and the record structures that hold the decomposition. Note that these data structures are circular (e.g., nodes include lists of their parents and children, which must of course similarly include the original node, etc.). Therefore, it is not possible to print these structures with the Scheme printer.

5. `debug.sc` holds a few functions that allow one to print out parts of the circular structures.

6. `examples.sc` defines two Bayes networks, `mcbn1` and `mcbn2`, which have the structure shown in Figure 1 but with different numerical models.

7. `sort.sc` is, again, the same as it was for the previous problem set.

We do not include a listing of the above programs with the problem set. You are not expected to understand the details of the implementation in `cooper.sc`,

`Define-bnet` is a function that makes it convenient to define a net. Here is an example of its use, to define `MCBN1`.

```
(define mcbn1
  (define-bnet 'mcbn1
    '((a () (0.8 0.2))                       ;; P(A) = 0.2
      (b (a) ((0.9 0.1) (0.3 0.7)))          ;; P(B|~A) = 0.1, P(B|A) = 0.7
      (c (a) ((0.7 0.3) (0.4 0.6)))          ;; ...
      (d (b c) (((0.9 0.1) (0.7 0.3)) ((0.6 0.4) (0.2 0.8))))
      ;; P(D|~B,~C) = 0.1, P(D|~B,C) = 0.3, P(D|B,~C) = 0.4, P(D|B,C) = 0.8
      (e (c) ((0.9 0.1) (0.6 0.4))))))
```

See the comments in `cooper.sc` for the exact syntax and options. Defining the network also calls the procedure `dissect`, which computes the decomposition that makes computation more efficient.

Our code provides just a single program that can pose a query to the Bayes net: `infer`. It takes as its arguments a Bayes net and a list that specifies which variables should be set to which values. For example,

```
(infer mcbn1 '((d #t)))
```

asks for the probability of the circumstance in which the variable D has the value *true*. The probability of a state can be computed by making sure that every variable appears in this list.

```
(infer mcbn1 '((a #t) (b #f) (c #t) (d #t) (e #t)))  ==> 0.00432
```

Finally, the situation in which no variables are set is consistent with all states, and must have probability 1.0.

```
(infer mcbn1 '())  ==> 1.0
```

Problem 1 Computing conditional probabilities (5 pts)

`Infer` computes the probability of a circumstance. Often, we will be interested instead in a conditional probability. Recalling the definition of the conditional probability, write a function, `cp`, that takes a Bayes net and two lists of variable/value specifications (like the second argument to `infer`), and computes the conditional probability that the first circumstance holds given that the second holds. For example, you should be able to compute

```
(cp mcbn1 '((e #f)) '((a #t)))
```

Calculate this value by hand and check it against your program's output.[5] Turn in your program and show its results on the following problems, all using `mcbn1`:

$$P(e = \texttt{\#t} | b = \texttt{\#f})$$

$$P(d = \texttt{\#t}, e = \texttt{\#t} | a = \texttt{\#f}, b = \texttt{\#f})$$

$$P(a = \texttt{\#t} | d = \texttt{\#f}, e = \texttt{\#f})$$

Problem 2 Modeling complex probabilistic dependence

Here is a summary of an extended example used by Judea Pearl to introduce Bayes networks in his classic book, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, 1986:

> Holmes is worried that his house will be burgled. For the time period of interest, there is a 10^{-4} *a priori* chance of this happening, and Holmes has installed a burglar alarm to try to forestall this event. The alarm is 95% reliable in sounding when a burglary happens, but also has a false positive rate of 1%. Holmes' neighbor, Watson, is sure (well, say 90% sure) to call Holmes at his office if the alarm sounds, but he is also a bit of a practical joker, and knowing Holmes' concern, might (30%)

[5] If you ever get a probability from a program that is not in the range between 0 and 1 (inclusive), you have a good hint that you've made a mistake.

call even if the alarm is silent. Holmes' other neighbor, Mrs. Gibbons, is a well-known lush and often befuddled, but Holmes believes that she is four times more likely to call him if there is an alarm than if not.

Part A Building the model (4 pts)

Draw a Bayes network that describes the above dependencies. Now define a Bayes net, `Holmes`, that models this. Think carefully about the probabilistic dependencies and exactly what they mean. Turn in a listing of the call to `define-bnet` that defines your network.

You will have noticed that the story above does not tell the actual conditional probability that Gibbons will call if the alarm sounds or that she will call if the alarm is silent. Indeed, these probabilities would be quite difficult for Holmes to estimate. Fortunately, it is only the likelihood ratio between these conditions that matters. You can verify that your choice or particular numbers to use for the conditional probabilities leading to Gibbons' call are immaterial so long as their ratio is 4.

Part B Calculations (2 pts)

Within a few minutes of each other, Holmes is phoned by both Watson and Gibbons. Use your network to calculate the probability that Holmes has suffered a burglary.

Part C Explaining away (3 pts)

Realizing that Holmes lives in Los Angeles, it occurs to us that there might be another explanation for the (probable) alarm at Holmes' house. If a sufficiently large earthquake occurs, it may set off the burglar alarm even without there being a burglary. Define a new Bayes net, `quake`, based on the same facts that you used for `holmes`, but with the following additional story:

> Holmes, always the anxious type, listens to KNX, "News Radio," while at work, which will report, with 95% reliability, the occurrence of a sufficiently large quake. Holmes thinks that a burglary and an earthquake can each set off his burglar alarm in independent ways that make *noisy-or* a good model. From his original belief about the probabilistic connection between burglary and alarm, Holmes can compute a causal probability from burglary to alarm and a leak probability for an alarm occurring in the absence of a burglary. He estimates the probability of a quake causing an alarm as 60%. The incidence of quakes in Los Angeles is equal to the incidence of burglaries.

Use the `quake` network to calculate the probability of Holmes' house having been burgled if he receives calls from Watson and Gibbons, but also hears the report of a quake on KNX.

Recall that if C depends on A and B, whose influences combine by a noisy-or, then

$$1 - P(C|A, B) = (1 - P(C|A))(1 - P(C|B))(1 - \text{leak})$$

Part D Multiple connections (3 pts)

> Holmes realizes that his daughter was planning to visit his house this day, and would surely (98%) call if she heard the alarm, unless she was convinced that it was really the earthquake that had set it off.

Build another variation of the `quake` network, `daughter`, in which you model the daughter's calling or not calling. Describe and justify the modeling assumptions that you make, and the numbers you choose. (*Hint:* There is no single right answer here; choose a reasonable model of this part of the story and justify it.)

If she does *not* call, what is the probability that Holmes' house has in fact been burgled. Explain how this relates to the answer from the `quake` network.

Problem 3 Probabilistic diagnosis

Consider the diagnostic domain `dom3` of Problem Set 6, reproduced and shown graphically in Figure 2.

Part A Defining a bipartite graph Bayesian problem (3 pts)

Use `define-bnet` (much as it is used in defining `mcbn2` in `examples.sc`) to define the network corresponding to the diagnostic domain `dom3` with the additional probabilistic information shown in Figure 2. Turn in a listing of the expression you use to define this net.

Part B Exploring absent symptoms (2 pts)

Use the program you wrote for computing conditional probabilities to compute the probability of disease `a` given that symptoms `s1`, `s2` and `s3` are present.

```
(define dom3
  (make-domain
   '((a s1 s2 s5)
     (b s1 s2 s4)
     (c s1 s3 s4 s5)
     (d s1 s5)
     (e s2 s3 s4)
     (f s5)
     (g s3 s4)
     (h s3 s5)
     )))
```

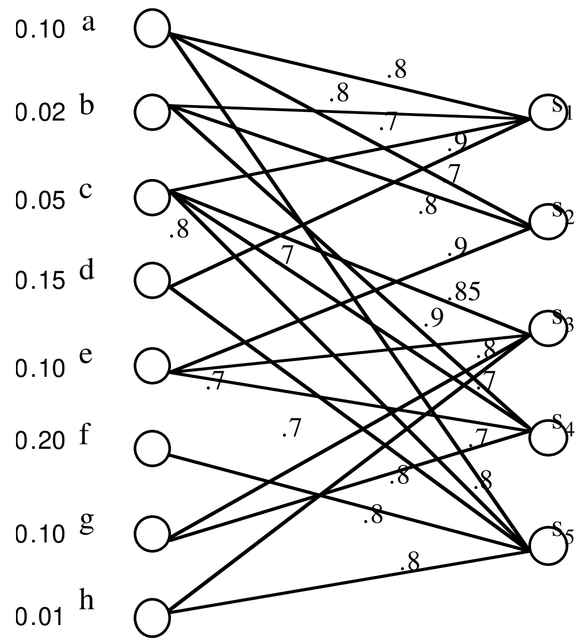

Figure 2: A bipartite graph for a diagnostic problem corresponding to domain `dom3` of Problem Set 6. Disorders are letters, symptoms are s_i, and numbers on arcs between diseases and symptoms are the causal probabilities. We assume independent causation (i.e., noisy-or) combination of causal influences at each symptom, with a uniform leak probability of 0.05.

Now compute the probability of `a` given that those symptoms are present but that `s4` and `s5` are absent. If the answer differs from the previous one, why? Contrast this behavior with the treatment that set-cover diagnosis gives to *absent* symptoms.

Part C Joint probabilities of multiple diseases (2 pts)

Calculate, as above, the probability of disease `e` assuming only that symptoms `s1`, `s2` and `s3` are present.

Calculate, similarly, the probability of disease `h` under the same condition.

Ben Bitdiddle (remember *him*?) suggests that the probability that `a`, `e` and `h` are present together is just the product of the probabilities that each is present by itself (as you've just computed). Check his suggestion by computing the joint probability of these three diseases given that symptoms `s1`, `s2` and `s3` are present. Comment on the sagacity of Ben's advice.

Part D Use for diagnosis (3 pts)

Assuming that `s1`, `s2` and `s3` are present and `s4` and `s5` are absent, find the most likely set of diseases. You should not need to explore the 2^8 possibilities, if you think about which sets seem like promising candidates, and if you are willing to take a slight risk of missing the very best. *Hint:* We do not expect you to write a program to do this, but just to explore "by hand" the most likely sets of diseases.

Problem 4 Grand Synthesis

Part A Design (3 pts)

Describe a general solution to the previous subproblem (finding the most likely set of diseases to explain the presence/absence of a set of symptoms). You should find the set-cover algorithm and our code here to be very useful.

Part B Implementation (Extra Credit of 5 pts)

Implement the design you sketched in the preceding subproblem. You probably need to build both a domain (in the lingo of ps6) and a Bayes net for the same disease-symptom relationships, unless you redefine `causes` and `consequences` to work off the Bayes net data structures directly. Debugging your code will be difficult because of the circular data structures and the consequent problems of printing out. Note the useful procedures in `debug.sc` to help ameliorate, but not solve, this problem.

Show your program and demonstrate it on the above example. Does it get the same solution that you did? Why, or why not?