Final Examination

CS 540: Introduction to Artificial Intelligence

May 9, 2004

LAST (FAMILY) NAME: ________SOLUTION______________________

FIRST NAME: ____

Problem	Score	Max Score
1	———	15
2	———	25
3	———	10
4	———	10
5	———	15
6	———	15
Total	———	90

1. [15] **Probabilistic Reasoning**

You are preparing to go overseas for a study abroad program and the university asks you to take an HIV virus test. You've tested positive, meaning the test indicates HIV. In the U.S. data shows that one in 17,000 people have HIV. Let random variable H stand for "has HIV" and TP stand for "tested positive." For your information, 1/17,000 = 0.00005882. HIV tests are known to be 99.95% accurate in the sense that the probability of testing positive when you *have* the disease is 0.9995, and the probability of testing negative when you do *not have* the disease is also 0.9995.

(a) [10] What is the probability that you actually have the disease? (Keep intermediate quantities to 8 decimal places.)

Given: $P(TP \mid H) = P(\neg TP \mid \neg H) = 0.9995$

and $P(H) = 1/17,000 = 0.00005882$

$$P(H \mid TP) = \frac{P(TP \mid H)\, P(H)}{P(TP)}$$

$$= \frac{(0.9995)(0.00005882)}{P(TP)}$$

$$= \frac{0.00005879}{P(TP)}$$

$$P(TP) = P(TP \mid H)\, P(H) + P(TP \mid \neg H)\, P(\neg H)$$

$$= (0.9995)(0.00005882) + (0.0005)(0.99994118)$$

$$= 0.00055876$$

So, $P(H \mid TP) = \dfrac{0.00005879}{0.00055876} = 0.1052 = 10.5\%$

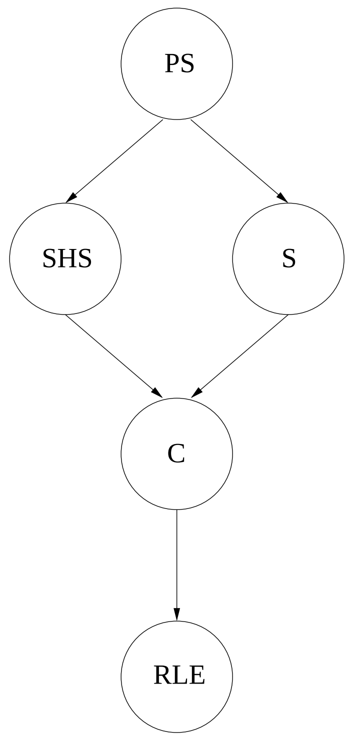
(b) [5] For every 17,000 people who take an HIV test, how many people will get a false positive result (i.e., they test positive when they in fact do not have HIV)?

$P(TP \mid \neg H) = 1 - 0.9995 = 0.0005,$ so $(17000)(0.0005) = 8.5$ people!

2. [25] **Bayesian Networks**

(a) [11] Consider a situation in which we want to reason about the relationship between smoking and lung cancer. We'll use 5 Boolean random variables representing "has lung cancer" (C), "smokes" (S), "has a reduced life expectancy" (RLE), "exposed to second-hand smoke" (SHS), and "at least one parent smokes" (PS). Intuitively, we know that whether or not a person has cancer is directly influenced by whether she is exposed to second-hand smoke and whether she smokes. Both of these things are affected by whether her parents smoke. Cancer reduces a person's life expectancy.

(i) [5] Draw the Bayesian Network (nodes and arcs only; no CPTs) for this domain.



(ii) [3] How many independent values are required to specify all the conditional probability tables (CPTs) for your network?

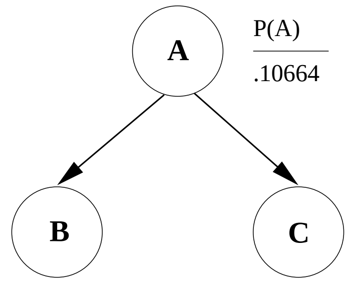
`1 + 2 + 2 + 4 + 2 = 11`

(iii) [3] How many independent values are in the full joint probability distribution for this problem domain?

$2^5 - 1 = 31$

2. (continued)

(b) [14] Consider the following Bayesian Network containing 3 Boolean random variables:



A	P(B)
t	0.85
f	0.04

A	P(C)
t	0.75
f	0.1

Compute

(i) [5] P(¬B, C | A)

$$P(\neg B, C \mid A) \;=\; P(\neg B \mid A)\,P(C \mid A) \;=\; (0.15)(0.75) \;=\; 0.1125$$

(ii) [5] P(A | ¬B, C)

$$P(A \mid \neg B, C) = \frac{P(\neg B, C \mid A)\,P(A)}{P(\neg B, C)}$$

$$= \frac{P(\neg B, C \mid A)\,P(A)}{P(\neg B, C \mid A)\,P(A) + P(\neg B, C \mid \neg A)\,P(\neg A)}$$

$$= \frac{(0.1125)(0.10664)}{(0.1125)(0.10664) + (0.096)(0.89336)}$$

$$= 0.12272$$

Now add on to the network above a fourth node containing Boolean random variable D, with arcs to it from both B and C. The remaining questions pertain to this new network.

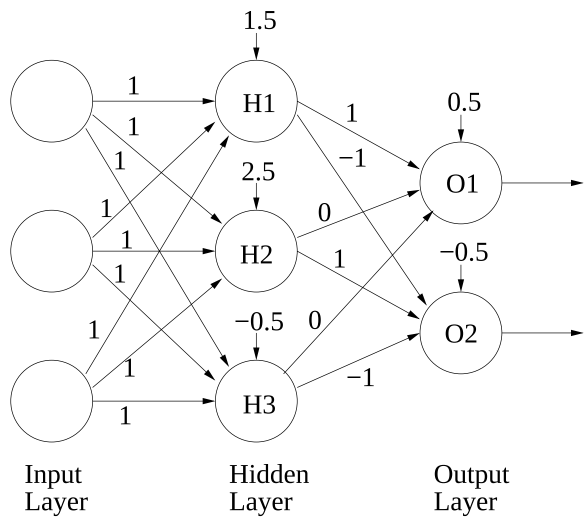
(iii) [2] Yes or No: Is A conditionally independent of D given B?

`No`

(iv) [2] Yes or No: Is B conditionally independent of C given A?

`Yes`

3. [10] **Neural Networks**
The following feedforward neural network takes three binary (0 or 1) inputs and produces two binary (0 or 1) outputs. Each node uses a *Linear Threshold Unit* as its activation function with the associated threshold value. Call an input vector that has exactly *n* inputs equal to "1" as an input vector with "count" *n*, for *n* = 0, 1, 2 or 3. Note that for this particular neural network, all input vector with the same "count" have the same output at both output units. In other words, for a given "count," the output of the network is the same no matter which particular input units are the ones with input equal to "1."



(a) [6] For a given input "count" of *n*, describe what is computed as the output of each hidden unit. Give your answer in terms of *n* and do not give simply a literal translation of each individual calculation performed.

```
Hidden node H1 computes "n > 1".  Hidden node H2 computes "n >
2".  Hidden node H3 always outputs "1".
```

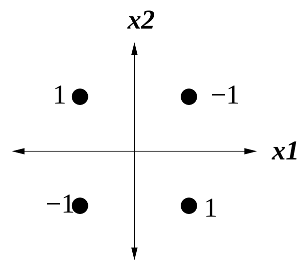
(b) [4] For a given input "count" of *n*, give an interpretation of the computed output O_1, O_2.

```
Output node O1 computes "n > 1" (i.e., "n = 2 or n = 3").  Output
node O2 always outputs "0".  (Note:  Output node O2 was orginally
intended to compute "n = 1 or n = 3" so that O1, O2 outputs the
binary value of n but, because of errors in defining the neural
net, it ended up not doing this.)
```

4. [10] **Support Vector Machines**
We want to construct a Support Vector Machine (SVM) that computes the XOR function. Instead of input and output values of 1 and 0, we'll use values 1 and –1, respectively. So, for example, if the input is [x_1 = –1, x_2 = 1] we want the output to be 1.

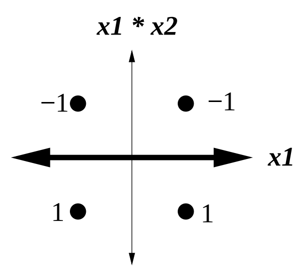
(a) [5] Using the four possible input vectors and their associated outputs, can an SVM be constructed to correctly compute XOR? If it can, show how by drawing the four possible input value pairs in the 2D input space, x_1, x_2, and the separator (i.e., decision boundary) computed by the SVM. If it cannot, explain or show why not.

```
Not possible because XOR is not linearly separable as required by
SVMs.
```



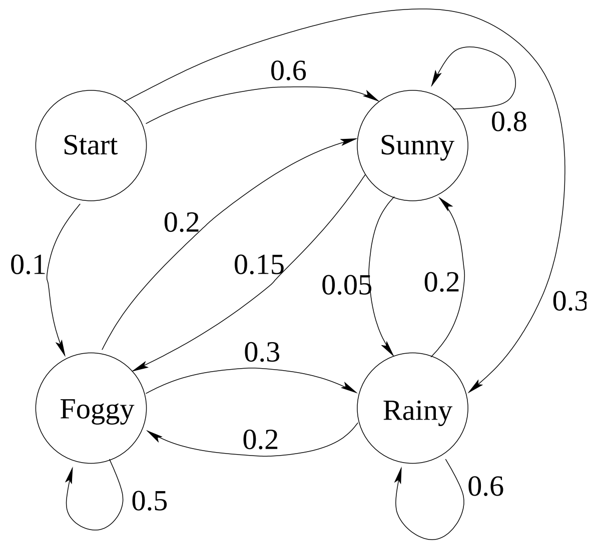
(b) [5] Suppose we re-express the input data using the computed features [x_1, $x_1 * x_2$] instead of the original [x_1, x_2] pair of values. So now the above example would have the input [–1, –1] and the same output value of 1 for the XOR function. Can an SVM be constructed to correctly compute XOR using the computed features rather than the raw features? If it can, show how by drawing the four possible input value pairs in the 2D input space, x_1, $x_1 * x_2$, and the separator (i.e., decision boundary) computed by the SVM. If it cannot, explain or show why not.

It can. The support vector that maximizes the margin in this case is the horizontal x_1 axis as shown below.



5. [15] **Markov Models**

(a) [5] Consider the following 1st-order Markov Model that describes the probability of tomorrow's weather given today's weather. The Start state is where you are to start and the arcs from that node specify the probability of the *first* day's weather (this corresponds to the Π vector that's used in the notes).



What is the probability of the following sequence of weather on 5 consecutive days: Rainy, Sunny, Sunny, Rainy, Foggy? That is, compute P(w_1=Rainy, w_2=Sunny, w_3=Sunny, w_4=Rainy, w_5=Foggy) where w_i specifies the weather on the i^{th} day.

P(w_1=R, w_2=S, w_3=S, w_4=R, w_5=F)

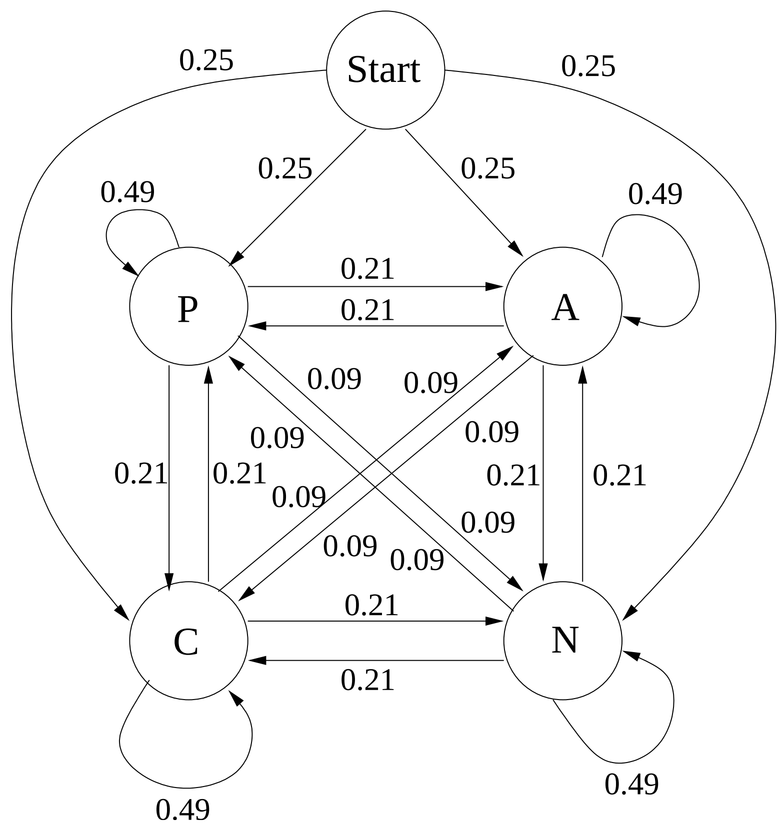
= P(w_5=F | w_4=R) * P(w_4=R | w_3=S) * P(w_3=S | w_2=S) * P(w_2=S | w_1=R) * P(w_1=R)

= (0.2)(0.05)(0.8)(0.2)(0.3)

= 0.00048

5. (continued)

(b) [10] Draw a 1st-order Markov Model in the style of the model shown in (a), but for the following domain: You work at the ACME Chocolate Factory and you watch chocolates as they pass by on a conveyor belt, one at a time. Four types of chocolates are made: Plain (P), with Almonds (A), with Coconut (C), and with Peanuts (N). Software controls a 4-position switch that determines which type of chocolate is made. Chocolate type P corresponds to a setting of 00, A to 01, C to 10, and N to 11. When the control computer is booted each morning there is an equal likelihood of each of the two bits being 0 or 1. Between each chocolate made, a bug in the program that controls the switch causes each of the two bits to switch to its other value (i.e., from 0 to 1 and from 1 to 0) with chance 30%. The switching/not-switching of each bit is independent of the other bit's switching behavior.



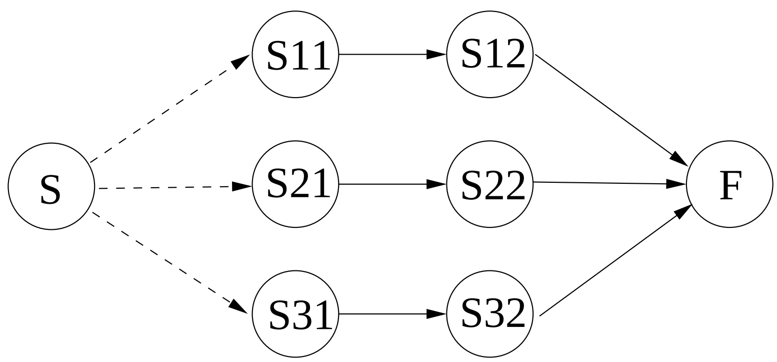
6. [15] **Planning**

(a) [6] Instead of using "causal links" in Partial-Order Planning, say we just mark a goal "done" when it is achieved (by "Step Addition" or "Simple Establishment"). Does this modification of the original POP algorithm guarantee finding correct solution plans? Briefly explain why or why not.

No, because without the causal links there is no way to detect when threats occur; that is, because a threat is detected when a step conflicts with a causal link, and with only a "done" mark it cannot be determined which step corresponded to the "producer" step that is being used to solve the given goal.

(b) [6] Show the final partial-order plan, including any causal links and temporal links, that corresponds to a problem that consists of 3 initial goals and each one can be solved independently by a sequence of 2 steps. Be sure to include Start and Finish nodes.

Note: The following partial-order solution plan does not show all the temporal links.



(c) [3] How many total-order plans (i.e., linearizations) are possible from your solution in (b)?

There are 3 possibilities for the first step, S11, S21 or S31. Say S11 is selected; then the next step can be either S12, S21 or S31. Next, notice that the two sequences consisting of steps S21, S22, S31 and S32 can, on their own, be totally ordered in 6 possible ways ([S21,S22,S31,S32], [S21,S31,S22,S32], [S21,S31,S32,S22], [S31,S32,S21,S22], [S31,S21,S32,S22], [S31,S21,S22,S32]). Since S12 could occur anwhere within, before or after these four steps in five places in each of these six sequences, there are 5 * 6 = 30 possible orderings of the steps S12, S21, S22, S31 and S32. Finally, since the first step could be any of three steps, there are a grand total of 3 * 30 = 90 total orderings possible from the partial-order solution graph in (b).