Final Examination

CS 540: Introduction to Artificial Intelligence

May 14, 2007

LAST NAME: _________SOLUTION____________________________

FIRST NAME: ____

Problem	**Score**	**Max Score**
1	________	15
2	________	20
3	________	10
4	________	15
5	________	20
6	________	20
Total	________	100

1. [15] **Probabilistic Reasoning**
In her guest lecture, Prof. Burnside presented data related to whether a woman has breast cancer or not, and whether the results of a mammography test are positive (i.e., indicating breast cancer is likely) or not. Let Boolean random variables *B* stand for "has breast cancer" and *M* stand for "mammography test is positive." The prior probability of having breast cancer is 1%. The probability of testing positive when you *have* breast cancer is 90%. The probability of testing negative when you do *not have* breast cancer is 89.9%.

(a) [5] What is the prior probability of having a positive mammography test, i.e., what is $P(M)$?

```
Given: P(B) = 0.01, P(M | B) = 0.9, P(¬M | ¬B) = 0.899
```

$$P(M) = P(M|B)P(B) + P(M|\neg B)P(\neg B)$$

$$= (.9)(.01) + (1-.899)(1-.01)$$

$$= 0.10899 = 10.9\%$$

(b) [5] If a patient has a positive mammography test, what is the probability that she has breast cancer? That is, compute $P(B \mid M)$.

$$P(B|M) = \frac{P(M|B)P(B)}{P(M)} = \frac{(.9)(.01)}{0.10899} = 0.0826 = 8.26\%$$

(c) [5] If a patient gets a negative mammography test, what is the probability that she has breast cancer? That is, compute $P(B \mid \neg M)$.

$$P(B|\neg M) = \frac{P(\neg M|B)P(B)}{P(\neg M)} = \frac{(1-.9)(.01)}{(1-.10899)} = 0.0011 = 0.1\%$$

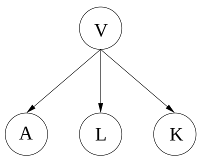
2. [20] **Naive Bayes**

Consider the problem of detecting if an email message contains a virus. Use 4 Boolean random variables to model this problem: V indicates if the message contains a virus (=1) or not (=0), A indicates if the message has an attachment (=1) or not, L indicates if the message is long (=1) or not, and K indicates if the sender is known (=1) to the receiver or not.

(a) [3] If we use a Naive Bayes Classifier to solve this problem, we make the assumption $P(A, L, K \mid V) = P(A|V)P(L|V)P(K|V)$. What does this assumption mean in terms of independence and/or conditional independence of any of the variables A, L, K, and V?

```
It means A, L and K are conditionally independent given V.
```

(b) [3] Draw the Bayesian Network (just the nodes and arcs) that corresponds to this problem with the Naive Bayes assumption in (a).



(c) [10] Given an email message with A=1, L=0, and K=0, compute the probability that it contains a virus. That is, compute $P(V = 1 \mid A = 1, L = 0, K = 0)$. Use the following probabilities as needed.

V	*P(A=1|V)*	*P(L=1|V)*	*P(K=1|V)*	*P(V)*
0	0.1	0.9	0.9	0.9
1	0.99	0.2	0.05	0.1

$$P(V = 1 \mid A = 1, L = 0, K = 0) = \frac{P(A = 1, L = 0, K = 0 \mid V = 1)P(V = 1)}{P(A = 1, L = 0, K = 0)} \quad \text{by Bayes's rule}$$

$$= \frac{P(A = 1|V = 1)P(L = 0|V = 1)P(K = 0|V = 1)P(V = 1)}{P(A = 1|V = 1)P(L = 0|V = 1)P(K = 0|V = 1)P(V = 1) + P(A = 1|V = 0)P(L = 0|V = 0)P(K = 0|V = 0)P(V = 0)}$$

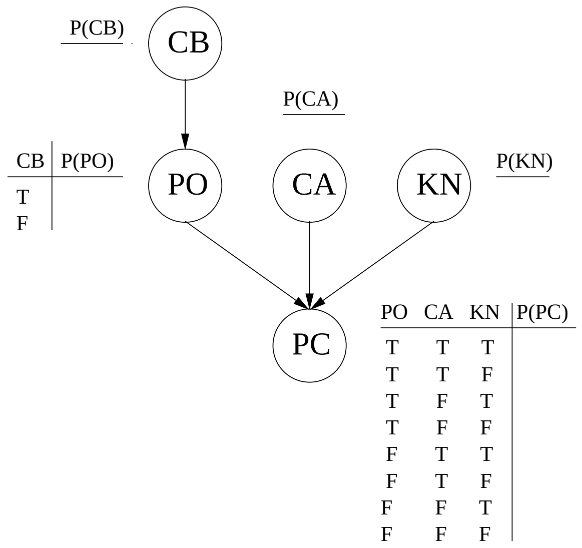
$$= \frac{(.99)(.8)(.95)(.1)}{(.99)(.8)(.95)(.1) + (.1)(.1)(.1)(.9)} = 0.9882$$

(d) [4] What constraints are there, in general, on the values filled in the 8 entries in the table above? That is, can we fill in the table with 8 arbitrary real numbers? If not, what properties must hold to make it valid?

```
All entries must be between 0.0 and 1.0, and P(V=0) + P(V=1) =
1.0.
```

3. [10] **Bayesian Networks**
Consider the following cell phone domain. Having a charged battery (CB) enables a cell phone to be operational (PO). Having an operational cell phone, being in the coverage area (CA), and knowing the number to call (KN) enables the placing of a call (PC). Assume there are no other dependencies. All random variables are Boolean.

(a) [7] Draw the Bayesian Network for this domain, including nodes, arcs and the form of each conditional probability table (CPT) at each node.



CB	P(PO)
T	
F	

PO	CA	KN	P(PC)
T	T	T	
T	T	F	
T	F	T	
T	F	F	
F	T	T	
F	T	F	
F	F	T	
F	F	F	

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

This was a poorly written question. I meant how many degrees of freedom are there in the full joint probability distribution table entries, with the answer: $2^5 - 1 = 31$.

4. [15] **Neural Networks**

(a) [3] Can a Perceptron learn the function defined by the following 5 data points? If so, construct a Perceptron that classifies all these values correctly; if not, argue why there is none.

Input1	**Input2**	**Desired Output**
-1	0	0
0	0	1
2	1	0
1	0	1
1	-2	0

`Cannot be learned by a Perceptron because the points are not linearly separable.`

(b) [2] Which of the following describes the process of *classifying* a test example using a feed-forward neural network that uses back-propagation learning?

(i) Activation values are propagated from the input nodes through the hidden layers to the output nodes.
(ii) Activation values are propagated from the output nodes through the hidden layers to the input nodes.
(iii) Weights on the arcs are modified based on values propagated from input nodes to output nodes.
(iv) Weights on the arcs are modified based on values propagated from output nodes to input nodes.
(v) Arcs in the network are modified, gradually shortening the path from input nodes to output nodes.
(vi) Weights on arcs from the input nodes are compared to the weights on the arcs coming into the output nodes, and then these weights are modified to reduced the difference.

`(i)`

(c) [2] Which of the options in (b) describes the process of *learning* in a feed-forward neural network that uses back-propagation?

`(iv)`

4. (cont.)

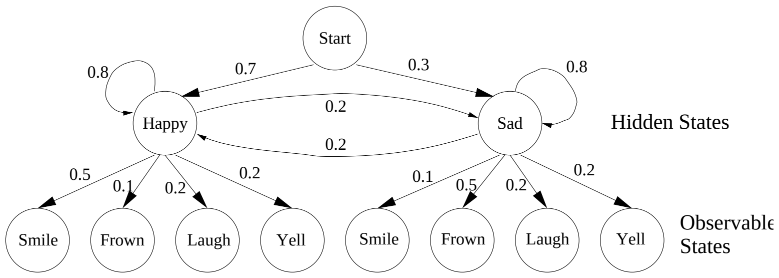
(d) [4] It is often stated that local minima can cause difficulties for a feed-forward neural network that uses back-propagation. Local minima of what function of what arguments? Also explain why this is a problem in terms of the type of search method that is performed by back-propagation.

Local minima of the error function over the set of all training examples, as a function of the set of weights on the arcs in the network. Since back-propagation is doing gradient descent search, it will get stuck at a local minimum and therefore never find the optimal network that minimizes the error function.

(e) [4] How can over-fitting be avoided when training a feed-forward neural network?

Use a "Tuning Set" to decide when to stop. That is, train the network using a Training Set to update the weights, but use a separate Tuning Set to evaluate each network’s performance in terms of classification accuracy. Stop when the error on the Tuning set reaches a minimum.

5. [20] **Hidden Markov Models**
The following HMM describes Chuck's emotional state. Each day he is either *Sad* or *Happy*. But he hides his emotional state so all you can observe is his facial expression each day, which is either a *Smile*, *Frown*, *Laugh*, or *Yell*. Let q_t define his emotional state on day t. Let O_t indicate his observed facial expression on day t. The arcs from the Start state give the probabilities of his initial emotional state on day 1 (i.e., they specify the π vector in the HMM).



(a) [6] What is P(q_2 = Happy)?

$$P((q_2 = Happy) = P((q_2 = Happy \mid q_1 = Happy)P(q_1 = Happy)$$
$$+ P((q_2 = Happy \mid q_1 = Sad)P(q_1 = Sad)$$
$$= (.8)(.7) + (.2)(.3) = 0.62$$

(b) [7] What is P(O_2 = Frown)?

$$P(O_2 = Frown) = P(O_2 = Frown, q_2 = Happy, q_1 = Happy)$$
$$+ P(O_2 = Frown, q_2 = Happy, q_1 = Sad)$$
$$+ P(O_2 = Frown, q_2 = Sad, q_1 = Happy)$$
$$+ P(O_2 = Frown, q_2 = Sad, q_1 = Sad)$$
$$= (.7)(.8)(.1) + (.3)(.2)(.1) + (.7)(.2)(.5) + (.3)(.8)(.5)$$
$$= 0.056 + 0.006 + 0.07 + 0.12 = 0.252$$

(c) [7] What is P(q_1 = Sad | O_1 = Frown)?

$$P(q_1 = Sad \mid O_1 = Frown) = \frac{P(O_1 = Frown \mid q_1 = Sad)\ P(q_1 = Sad)}{P(O_1 = Frown)} \quad \text{by Bayes's rule}$$

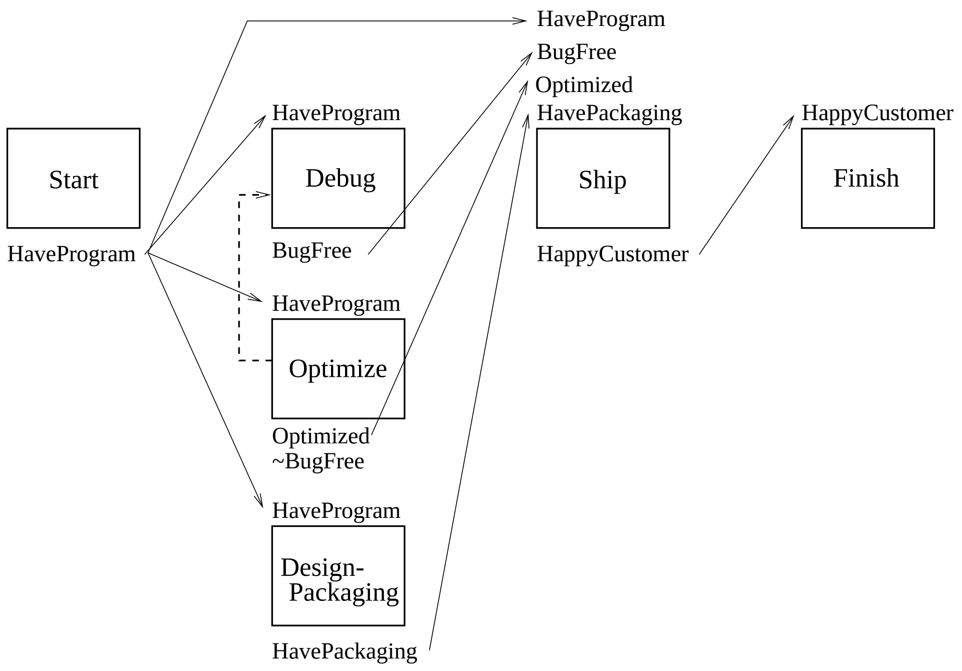
$$= \frac{P(O_1 = Frown \mid q_1 = Sad)\ P(q_1 = Sad)}{P(O_1 = Frown|q_1 = Sad)\ P(q_1 = Sad) + P(O_1 = Frown|q_1 = Happy)\ P(q_1 = Happy)}$$

$$= \frac{(.5)(.3)}{(.5)(.3) + (.1)(.7)} = 0.68$$

6. [20] **Planning**
(a) [12] Consider using a partial-order planner for the development and release of a software product. Assume there exist the following four STRIPS operators:

Operator	**Precondition**	**Effect**
Optimize	HaveProgram	Optimized $\wedge$ ¬BugFree
Debug	HaveProgram	BugFree
Ship	HaveProgram $\wedge$ BugFree $\wedge$ Optimized $\wedge$ HavePackaging	HappyCustomer
DesignPackaging	HaveProgram	HavePackaging

The initial state specifies that *HaveProgram* is true, and the goal state is one in which *HappyCustomer* is true. From the above information, say we have constructed the following partial-order plan (which shows only causal links, and has preconditions on top and effects on the bottom of each step):



(i) [6] Add steps and causal links to the above figure to create a new partial-order plan in which there are no open preconditions.

(ii) [4] What threat(s) exist, if any, in your partial-order plan produced in (i)? If there are threats, add temporal links (as dotted arcs) to the above figure to resolve them all.

`The` ¬*BugFree* `effect of the step Optimize threatens the causal link from the Debug step to the Ship step. Threat is resolved by demotion, i.e., forcing Optimize to occur sometime before Debug.`

6. (cont.)

(iii) [2] Give one possible solution plan (i.e., total ordering of steps) that is implied by your partial-order plan produced in (i) and (ii).

```
  There are three possible solutions that are consistent
with the final partial-order plan:

1.  DesignPackaging, Optimize, Debug, Ship
2.  Optimize, DesignPackaging, Debug, Ship
3.  Optimize, Debug, DesignPackaging, Ship
```

(b) [4] Define a STRIPS operator, *move(x,y)*, that gives the preconditions and effects for a move of the blank square, *B*, from position *x* to position *y* in the 8-puzzle. Use the following predicates:

(i) $At(t, x)$ means tile *t* is at position *x*

(ii) $At(B, x)$ means the blank square is at position *x*

(iii) $Movable(t)$ means tile *t* is adjacent (north, south, east or west) to the blank square

`Op(Action:` $move(x, y)$,
`Precond:` $At(B, x) \wedge At(t, y) \wedge Movable(t)$,
`Effects:` $\neg At(B, x) \wedge \neg At(t, y) \wedge At(B, y) \wedge At(t, x)$)

(c) [4] Define one or more effect axioms in the Situation Calculus that capture the same meaning as the operator *move(x,y)* in (b). (You may modify the arguments of the predicates to include a situation variable, if desired.)

$$\forall x, y, s, t \; (At(B, x, s) \wedge At(t, y, s) \wedge Movable(t, s)) \Rightarrow$$

$$(At(B, y, Result(move(x, y), s)) \wedge At(t, x, Result(move(x, y), s))$$