Final, CSC242, May 2009

Name:_____

*Write your **NAME** legibly on your bluebook* **AND EXAM, WHICH YOU WILL ALSO TURN IN**. *This is a 168-minute) test. Work all problems. You may use two double-sided pages of notes. Please hand your notes in with your bluebook and exam. The best strategy is not to spend more than the indicated time on any question (minutes = points). Please start each answer on a new page*

1. Newpage! Logic: 15 Min.

(a) (5 min) Write in FOPC the assertion that *every key will eventually be lost forever.* Use only the vocabulary $Key(x)$ (x is a Key), Now (the current time), $Before(t_1, t_2)$ (time t_1 comes before time t_2), $Lost(x, t)$ (x is lost at time t).

(b) (5 min) True or False, and why? If a FOPC clause can be resolved with a copy of itself it must be logically equivalent to $True$

(c) (5 min) True or False? $(A \vee B) \wedge \neg(A \Rightarrow B)$ is satisfiable.

Answers:

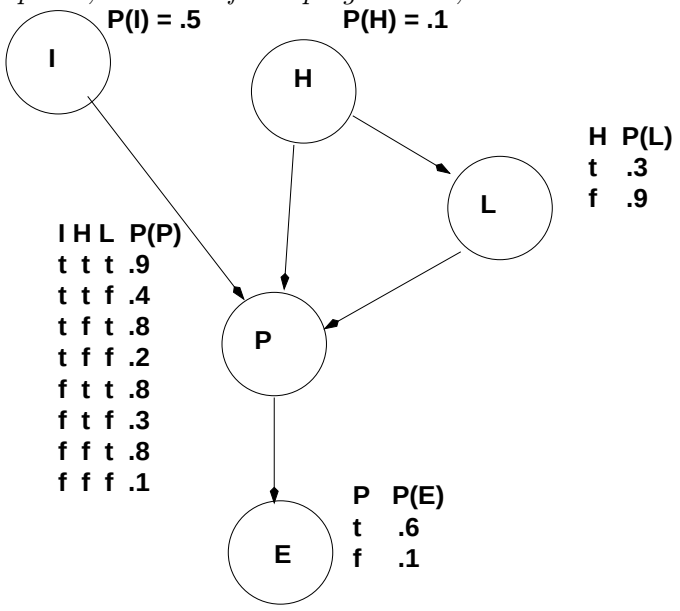
(a)

$$\forall x Key(x) \Rightarrow [\exists t_1 Lost(x, t_1) \wedge \forall t_2 Before(t_1, t_2) \Rightarrow Lost(x, t_2)].$$

(b) To resolve with self it must have a symbol ORed with its negation, so it's always true.

(c) A true and B false works.

2.New Page! Bayes Nets: 30 min. In the net below, the boolean variables have the semantics: *I: Intelligent, H: Honest, P: Popular, L: LotsOfCampaignFunds, E: Elected.*



(a) (3 min) Which of these, if any, are asserted by the **structure** of the network (leaving aside the conditional probability tables (CPTs)).

(i) $\mathbf{P}(I, L) = \mathbf{P}(I)\mathbf{P}(L)$

(ii) $\mathbf{P}(E \mid P, L) = \mathbf{P}(E|P, L, H)$

(iii) $\mathbf{P}(P \mid I, H) = \mathbf{P}(P \mid I, H, L)$

(b) (7 min) Calculate the value of $P(i, h, \neg l, p, \neg e)$. Show and explain your work.
(c) (7 min) Calculate the probability that a candidate is intelligent given that she is honest, has few campaign funds, and gets elected. That is, calculate $\mathbf{P}(I \mid h, \neg l, e)$. Show and explain your work.
(d) (5 min) Suppose we want to add the variable *S: StealsTheElection* (stuffs ballot box, bribes election judges, pays people to vote, etc.) to the network. Draw your new network and provide new or modified CPTs as needed: just come up with reasonable probabilities and justify them.
(e) (3 min) If there are two candidates in the race, then to figure the joint distribution over the two sets of variables, we just need two copies of the network, one per candidate. True or False and why?
(f) (5 min) True or False and why? Every Boolean function can be represented by some Bayesian network.

Answers:

(a) (i) and (ii)

(b) $P(i, h, \neg l, p, \neg e) = P(i)P(h)P(\neg l \mid h)P(p \mid i, h, \neg l)P(\neg e \mid p). = .5 * .1 * .7 * .4 * .4 = .0056.$

(c)

$$\mathbf{P}(I \mid h, \neg l, e) = \alpha \mathbf{P}(I, h, \neg l, e) =$$

$$\alpha(\mathbf{P}(I, h, \neg l, p, e) + \mathbf{P}(I, h, \neg l, \neg p, e)) = \alpha(\langle 084, .063\rangle + \langle .021, .0245\rangle) \approx \langle .545, 455\rangle.$$

(d) S(teals election) is parent of E and child of H and possibly P (since someone popular probably doesn't need to rig the election). The CPT for E is the same as before given $S = false$ but when $S = true$ it has a high probability for E, presumably with less dependence on P. The CPT for S should show $S = true$ less likely when $H = true$.

(e) False. Such independence is too simple, and would mean that both candidates can be elected, for instance.

(f) True. In the worst case have one Boolean output node with n Boolean parents and an n-dimensional, 2^n-big CPT.

3. New Page! Language: 22 Min.

A *probabilistic context-free grammar* (PCFG) is a CFG with probabilities on the rules, so that in generative mode it induces a probability distribution over all allowable sentences in the language. As you'd expect, the probabilities are independent, for each non-terminal they sum to one, and each defines how likely this rule is used to expand the non-terminal. Here's a PCFG, with Λ denoting the empty string:

$0.6 : NP \rightarrow Det \;\; AdjString \;\; Noun$
$0.4 : NP \rightarrow Det \;\; AdjString \;\; NounNounCompound$
$0.5 : AdjString \rightarrow Adj \;\; AdjString$
$0.5 : AdjString \rightarrow \Lambda$
$1.0 : NounNounCompound \rightarrow Noun \;\; Noun$
$0.8 : Det \rightarrow$ **the**
$0.2 : Det \rightarrow$ **a**
$0.5 : Adj \rightarrow$ **small**
$0.5 : Adj \rightarrow$ **green**
$0.6 : Noun \rightarrow$ **village**
$0.4 : Noun \rightarrow$ **green**

(a) (3 min) What is the longest NP that can be generated by the grammar?
(b) (5 min) Which of the following have a nonzero probability of being generated as complete NPs?
(i) **a small green village** (ii) **a green green green** (iii) **a small village green.**
(c) (5 min) What is the probability of generating **the green green**? Show and explain your work.

(d) (3 min) What types of ambiguity are exhibited by the phrase in (c)?
(i) lexical, (ii) syntactic, (iii) referential, (iv) stochastic, (v) none
(e) (3 min) Give a compelling argument for or against this statement: Given any PCFG and any finite word sequence, it is possible to calculate the probability of that sequence being generated by the PCFG.
(f) (3 min) True or False and why? Context free grammars fall to capture English grammar because the meaning of an English sentence may depend on the context of the utterance.

Answers: (a) With exponentially decreasing probability we can have NPs of any length.

(b) (i), (ii), and (iii).

(c) 'the green green' can be generated two different ways. If first 'green' is Adj, $p1 = .6*.8*.5*.5*.5*.4 = .024$, If noun $p2 = .4*.8*.5*1*.4*.4 = .0256$: The probabilities are independent so we can sum (marginalize!) and get p = .0496.

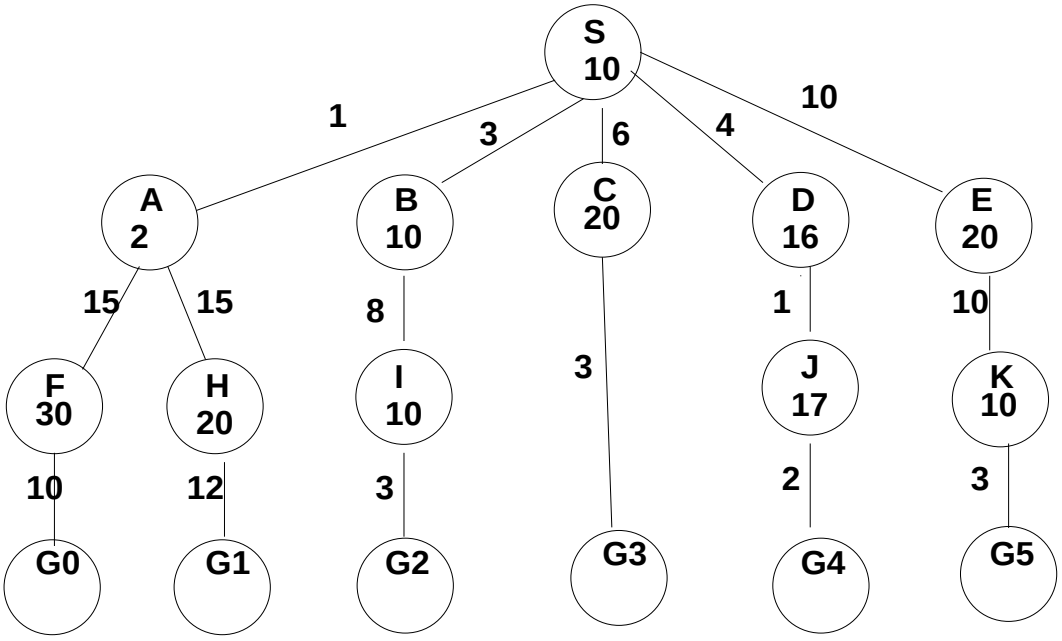
(d) (i) and (ii)

(e) True: there's a finite number of parses and each parse has a probability that is the product of its rules' probabilities...

(f) False: the C in CFG refers to syntactic context of a non-terminal.

4. New Page! Search: 31 Min.

Below is a search space with S the start state and G0 – G5. satisfying the goal test. Arcs are labeled with the cost of traversing them and the estimated cost to a goal is reported inside nodes. The space is a tree, so you have an OPEN list but no CLOSED list. (ONLY) where applicable, search proceeds "left to right" through successors. Where applicable, in case of ties, nodes with earlier letters are expanded first.



For each of the following search strategies (a) – (e), list in order all the states popped off the open list. (Hint: all answers start with S).
(a) (5 min) Hill Climbing (Greedy)
(b) (5 min) Iterative Deepening: Assume S is put on OPEN at the start of each iteration.
(c) (5 min) Uniform Cost
(d) (5 min) Depth First
(e) (5 min) A*

(f) (3 min) We proved that A* is an optimal search strategy. Verify that it found the optimal (cheapest) path to a goal here, or explain what happened otherwise.
(g) (3 min) Uniform cost seems like it should act like BFS, but its complexity is not measurable in terms of branching factor and depth. In fact, its time and space complexity can be much worse than BFS: why?

Answers

(a) S A H G1

(b) S S A B C D E S A F H B I C G3

(c) S A B D J C G4

(d) S A F G0

(e) S A B D I G2

(f) Node J overestimates the cost to the goal, so the heuristic is not admissible, and A* proceeds down I instead, missing best solution. Inadmissibility means the optimality theorem does not apply.

(g) UC can spend lots of time and space exploring cheap actions that may not have much effect. Its worst-case complexity goes as $b^{C/e}$, where b is branching factor, C is the cost of the optimal solution and e is the minimal cost of an action.

5. New Page! Reinforcement Learning: 25 Min.

You are a passive learning agent; you follow a fixed policy. You always start in state S1. You make three trials in your state space, each of which ends in the terminal state S3, which has a reward of 10. Your experience (actions, rewards) are as follows (your state sequence reads left to write interspersed by the actions that cause state transitions, with reward below each state). Trials 2 and 3 are identical, that's not a misprint.

Trial	State	action	State	action	State	action	State
	Reward		Reward		Reward		Reward
1	S1	A	S1	A	S2	B	S3
	-1		-1		-2		10
2	S1	A	S2	B	S2	B	S3
	-1		-2		-2		10
3	S1	A	S2	B	S2	B	S3
	-1		-2		-2		10

(a) (2 min) At this point, what can you say about the number of states in your state space?

For the following, initially your guess at the utilities of all states is 0. When you hit a terminal state you are entitled immediately to set its utility to its reward. Your discounting constant $\gamma = .9$. Your learning constant $\alpha = .5$,

First, assume you're using Adaptive Dynamic Programming (ADP) to learn.

(b) (5 min) At this point what is your estimate of $T(s, a, s')$?

(c) (5 min) You decide to use value iteration to compute the utilities of states. What is your first estimate of U(S2)?

(d) (3 min) Besides value iteration, how might you have computed this utility? (hint: you would also get U(S1)).
(e) (5 min) Now assume instead you're using Temporal Difference to learn. After trial 1 above and the TD computation, what is your estimate of U(S2)?
(f) (5 min) For realistically high-dimensional states and maybe actions with parameters, $T(s, a, s')$ would be impossibly huge to learn or even to represent. What can we do to address these problems?

Answers:

(a) there are at least 3 states (and the minimum reward is 10). More you don't know.

(b) The table should show T(S1,A,S1) = .25, T(S1,A,S2) = .75, T(S2,B,S2) = .4, T(S2,B,S3) = .6.

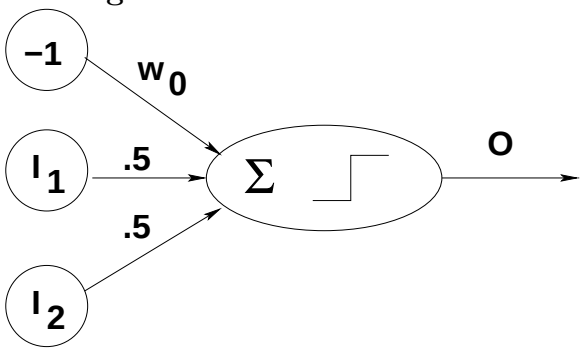
(c) Russell and Norvig eq. 17.6 is the Bellman update equation. using it, $U(S2) \longleftarrow -2+.9*(6) = 3.4$, where the factor in parens is the result of a max operation.

(d) You could substitute all the information you've gathered into the simplified Bellman equation (Russell and Norvig eq. 21.2) and solve the linear system for all utilities.

(e) Using one form of the standard TD equation (R&N eq. 22.3), $U(S2) = 0+.5*(-2+.9*10-0) = 3.5$.

(f) Don't use raw state vectors, use a smaller number of relevant features based on them, and use a function approximation (linear, neural net, splines, whatever) to get a compact representation that "generalizes" (i.e. interpolates) over experiences and can thus supply some sort of useful guess (you hope) for unexplored parts of the space.

6. New Page! Perceptron Learning: 20 Min.



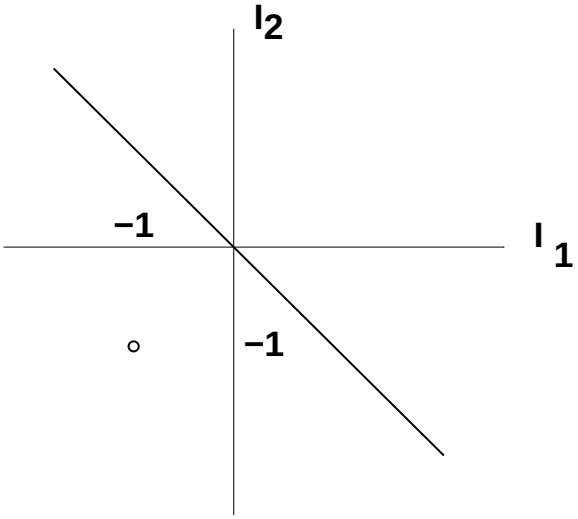
Here is a two-input perceptron with its threshold implemented as a weight w_0 applied to an input whose value is a constant -1. Say weights w_1, w_2 for inputs I_1, I_2 respectively are both set to .5 and .5. Its output is labeled O (that's not a zero!).
(a) (5 min) Plot the decision boundary of the perceptron with these settings for w_1 and w_2, and with $w_0 = 0$, in the (I_1, I_2) input space. Also plot the point (-1,-1), which represents the point in input space with both inputs equal to -1. Let's say this input is on the WRONG side of the decision boundary: successful learning will get the boundary below (or on) the (-1,-1) point somehow.

We're going to learn *only* w_0, our threshold, by (repeatedly) using only one input training instance: $I_1 = I_2 = -1$, which you plotted in (a). Thus we shall hold all three inputs and two of the weights constant ($w_1 = w_2 = .5$). We start with $w_0 = 0$. Our perceptron's activation function is a step function that evaluates to 1 (accepts the input) if its weighted-input sum is ≥ 0 and evaluates to 0 (rejects the input) if the sum is < 0. We want our perceptron to output 1 on the (-1,-1) input.
(b) (10 min) Here's the first row of a table that makes everything explicit. You are to extend the table until the perceptron learns (rows stop changing) or You're convinced it isn't learning. Of course you use the perceptron learning rule to update w_0. Use learning rate constant $\alpha = .5$. The $Error = Correct - Output$. (Hint: only numbers in * columns can change).

*trial	I0	*w0	I1	w1	I2	w2	*sum	*Output	Correct	*Error
1	-1	0	-1	.5	-1	.5	-1	0	1	1
2										
3										
...										

(c) (5 min) Suppose our training point was (-10, -10): How much would w_0 change per iteration in this case? If in general we wanted more error to cause faster learning, what could we do? Discuss the merits of adjusting α and of changing the activation function.

Answer:



(a) 45 degree line , oriented NW-SE. It goes thru origin at $w_0 = 0$: changing w_0 moves it parallel to itself.

(b) $w_j = w_j + \alpha I_j Err$

*trial	I0	*w0	I1	w1	I2	w2	*sum	*Output	Correct	*Error
1	-1	0	-1	.5	-1	.5	-1	0	1	1
2	-1	-.5	-1	.5	-1	.5	-.5	0	1	1
3	-1	-1.0	-1	.5	-1	.5	0	1	1	0

(c) Same old .5. We'd need the error to increase for worse answers, so we'd abandon the simple step activation and go for something like a ramp or sigmoid function. Changing the learning rate alone would still give a constant change since errors are either 0 or 1.

7. New Page! Planning: 10 Min.

A simple computer has a bunch of memory cells M and some registers R. Two computer instructions could be LOAD(M,R) (copy contents of M into R, overwriting what's there) and ADD(M,R) (add contents of M to contents of R, leaving result in R).

(a) (5 min) Express these instructions as STRIPS rules.

(b) (5 min) With several registers, you could imagine a compiler being inefficient with register usage (overwriting partial results that could be reused, for instance). Relate what you know or can imagine about code optimization to what you know about planning.

Ans: (a) want precondition, add and delete lists: here, preconditions are null it seems, others obvious.

(b) partial order planning an obvious one, and in fact I guess conditional planning (for guessing about IF statements), or generally graph planning could have rules for literal and figurative 'mutex' computer constraints.

Chetan notes: Just an interesting note of what i had learnt from the compilers class taught by Chen. This register efficiency problem is converted to a graph coloring problem (constraint satisfaction problem). The graph consists of nodes that are the registers and the dependences in the statements are the edges.

8. New Page! Robotics: 15 Min.

Figures of 2-DOF robot and configuration space are here unintentionallyh left blank.

This robot arm has a telescoping upper arm fixed at its left-hand end, which is also the origin. Its minimum length is $d = 1$ and maximum length (shown) is $d = 3$. The forearm is a thin wand of constant length 3 that can rotate at the elbow, but not past vertical in either direction. Two obstacles are shown with left edges at $x = 4$, separated by a distance of 2. The diagram shows a start configuration and a goal configuration.

(a) (5 min) the next page shows four configuration spaces. Circle the correct one for this problem.

(b) (5 min) On that space mark the locations of the starting configuration with S and the goal with G and draw an appropriate plan for the robot to move from S to G.

(c) (5 min) Malfunction!! Robot loses all position (and angle) sensing! Contact, motion limit, and time sensing are still working, however. Linear and rotary motions are always at a constant (but unknown) speed. Describe a plan to reach the goal from the start state, or explain why you cannot.

Answers:

(a) (iii).

(b) S is at d=3 in upper gap, G at d=3 in center, path goes off to left, around the obstacle, back in to G.

(c) E.g. retract arm to d=1, bend arm to minimum angle limit, start timer, bend arm to maximum angle limit, stop timer, bend arm back for elapsed time; it should be now be pointing straight right, so extend arm to d=3.