15-281 Archives Practice Midterm 2b

INSTRUCTIONS

- Exam length: 80 minutes
- You are permitted to have one handwritten page of notes, double-sided
- Sole-function calculators are allowed
- No other electronic or other resources are allowed.

Name	
Andrew ID	

For staff use only

Q1. Logical Agents	/ 17
Q2. Classical Planning	/ 17
Q3. MDPs	/ 20
Q4. Reinforcement Learning	/ 31
Q5. Bayes Nets	/ 15
Total	/ 100

Q1. [17 pts] Logical Agents

Consider the logical agent from programming assignment 3 and lecture. In it, you used a SAT solver to determine a feasible plan. Recall that you had symbols P[x,y,t] for whether pacman is at (x,y) at time t. You also had symbols North[t], South[t], East[t], and West[t] for actions taken at t. You created successor state axioms representing actions available at $t-1$ (i.e., not blocked by a wall) to reach a state at time t. For example $P[2,3,4] \iff P[2,2,3] \wedge North[3]$. You also created domain rules about mutual exclusivity of states and actions.

(a) [9 pts] Assume Pacman is in its starting state at time $t = 0$ and the goal is checked at least for $t = T$. While increasing time T until you found a satisfying plan, you created a new logical sentence representing all of the facets of the Pacman problem and passed it into the SAT solver. We want to minimize the number of rules we create. For each of the facets that were **required** to be present in the sentence to find a satisfying plan, select the corresponding box. Do not include extra rules that could be applied but are not required.

☒ the initial location of Pacman $P[x, y, t]$ at time $t = 0$

□ the negation of the initial location of Pacman $P[x, y, t]$ at time $t = 1 \ldots T$

□ symbols Wall[x,y] representing the presence of a wall at location (x,y)

□ successor state axioms for time $t = T$ only

□ successor state axioms for time $t = 0 \ldots (T - 1)$

□ successor state axioms for time $t = 0 \ldots T$

□ successor state axioms for time $t = 1 \ldots (T - 1)$

☒ successor state axioms for time $t = 1 \ldots T$

☒ mutual exclusion rules for Pacman's location at time $t = 0$ only

□ mutual exclusion rules for Pacman's location for $t = 0 \ldots (T - 1)$

□ mutual exclusion rules for Pacman's location for $t = 0 \ldots T$

□ mutual exclusion rules for Pacman's action at time $t = 0$ only

☒ mutual exclusion rules for Pacman's action for $t = 0 \ldots (T - 1)$

□ mutual exclusion rules for Pacman's action for $t = 1 \ldots T$

☒ the goal state of Pacman $P[x, y, t]$ at time $t = T$ only

□ the goal state of Pacman $P[x, y, t]$ for times $t = 0 \ldots T$

□ the negation of the goal state of Pacman $\neg P[x, y, t]$ for time $t = T$ only

□ the negation of the goal state of Pacman $\neg P[x, y, t]$ for times $t = 0 \ldots (T - 1)$

I would accept mutual exclusion criteria for locations at all time steps t=0...T-1 instead probably

(b) [8 pts] Circle True or False for the following statements.

True [✓ False] If $KB \models q$, then the KB is true in every model where q is true

True [✓ False] If $KB \models q$, then $KB \vee \neg q$ is not satisfiable.

[✓ True] False Resolution can be used to decide if **any** propositional logic query is entailed by a KB

[✓ True] False Forward chaining is sound for proving entailment of definite clauses.

Q2. [17 pts] Classical Planning

Prof. Rosenthal's daughter is learning to put on her socks and shoes by herself. Prof. Rosenthal decides to use classical planning as a representational model for helping her find an optimal plan for foot accessories. She creates the following operators:

Put_Sock:
Preconditions: ¬ on_sock(foot), ¬ on_shoe(foot)
Add Effects: on_sock(foot)
Delete Effects: ¬ on_sock(foot)

Put_Boot:
Preconditions: on_sock(foot), ¬ on_shoe(foot)
Add Effects: on_shoe(foot)
Delete Effects: ¬ on_shoe(foot)

Put_Sandal:
Preconditions: ¬ on_sock(foot), ¬ on_shoe(foot)
Add Effects: on_shoe(foot)
Delete Effects: ¬ on_shoe(foot)

Remove_Sock:
Preconditions: on_sock(foot), ¬ on_shoe(foot)
Add Effects: ¬ on_sock(foot)
Delete Effects: on_sock(foot)

Remove_Shoe:
Preconditions: on_shoe(foot)
Add Effects: ¬ on_shoe(foot)
Delete Effects: on_shoe(foot)

Her daughter has two feet – left_foot and right_foot - and is getting ready for school in the morning. She starts in state ¬ on_sock(left_foot), ¬ on_sock(right_foot), ¬ on_shoe(left_foot), ¬ on_shoe(right_foot).

(a) [4 pts] First Prof. Rosenthal starts with linear planning. She sets a goal for her daughter "Put your shoes and socks on" and her daughter identifies the following goals in the following order:

on_shoe(left_foot), on_shoe(right_foot), on_sock(left_foot), on_sock(right_foot).

What plan would be returned by linear planning starting with the leftmost goal?

Linear Plan:
Put_Sandal(left_foot),
Put_Sandal(right_foot),
Remove_Shoe(left_foot),
Put_Sock(left_foot),
Put_Boot(left_foot),
Remove_Shoe(right_foot),
Put_Sock(right_foot),
Put_Boot(right_foot)

(b) [8 pts] Prof. Rosenthal decides that GraphPlan is better. Identify the type(s) of mutual exclusion relations present between each of the following pairs of actions so that her daughter can find a plan. Assume that the foot is the same for both actions (i.e., PutBoot(left) and PutSandal(left)). Select all that apply.

Put_Boot / Put_Sandal	☒ Interference	☐ Inconsistency	☒ Competing Needs	☐ None
Put_Sock / Put_Sandal	☒ Interference	☐ Inconsistency	☐ Competing Needs	☐ None
Remove_Sock / Put_Sandal	☒ Interference	☐ Inconsistency	☒ Competing Needs	☐ None
Put_Boot / Put_Sock	☒ Interference	☐ Inconsistency	☒ Competing Needs	☐ None

(c) [5 pts] How many levels of the GraphPlan graph must be created before a plan is found for the same goal in part a? Explain your answer.

Levels:
2

Explain:
PutSock(left) and PutBoot(left) interfere so they cannot be done simultaneously. Same with the right foot. But PutSock(left) and PutSock(right) don't interfere so they can be done in parallel.

Q3. [20 pts] MDPs

This gridworld MDP operates like the one we saw in class. The states are grid squares, identified by letter. The agent always starts in state S. There are two terminal goal states, L(oss) with reward $R(L, exit, x) = -5$ and transition $T(L, exit, x) = 1.0$, and W(in) with reward $R(W, exit, x) = +5$ and transition $T(W, exit, x) = 1.0$. Rewards $R(s)$ are 0 in non-terminal states. The transition function is such that the intended agent movement (North, South, West, or East) happens with probability .8. With probability .1 each, the agent ends up in one of the states perpendicular to the intended direction (see Figure 1 Right). If a collision with an outer edge happens, the agent stays in the same state.

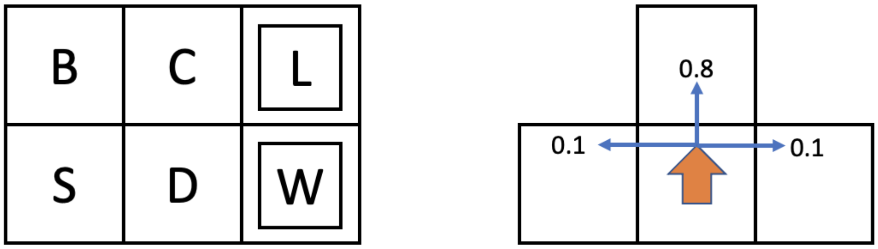


Figure 1: Left: MDP setup. Start in S. Terminal states L and W.
Figure 1: Right: Transition function. Given an action such as north, 80% of the time it will go north and 10% each will wind up east or west (perpendicular to the action direction).

(a) [12 pts] Suppose that $\gamma = 0.9$. Compute V_0, V_1, and V_2 using value iteration by filling in the table below.

s / V_k	S	B	C	D	W	L
V_0	0	0	0	0	0	0
V_1	0	0	0	0	5	-5
V_2	0	0	0	3.6	5	-5

(b) [4 pts] Write out the equation for $Q_3(C, East)$, substituting proper state and action names. Do not write any max's and $\sum$'s. For summations, expand the summation into all parts. For maxes, you should be able to substitute the values you found from part (a). Be sure to use the correct subscripts as necessary. Do not substitute numbers yet.

$Q_3(C, East) =$

$[T(C, East, C)[R(C) + \gamma * V_2(C)]] + [T(C, East, L)[R(C) + \gamma * V_2(L)]] + [T(C, East, D)[R(C) + \gamma * V_2(D)]]$

(c) [4 pts] Now substitute values in for your variables in $Q_3(C, East)$. CIRCLE your final q-value answer in the box below.

$Q_3(C, East) =$

$[0.1[0 + 0.9 * 0]] + [0.8[0 + 0.9 * -5]] + [0.1[0 + 0.9 * 3.6]] = -3.276$

Q4. [32 pts] Reinforcement Learning

Use the same MDP from above, but now assume that the transition function and reward function are unknown. The states are grid squares, identified by letter. The agent always starts in state S. There are two terminal goal states, L(oss) and W(in) If a collision with an outer edge happens, the agent stays in the same state.

The agent starts with the policy that always chooses to go East, and executes the following two trials.

- (S,East,D,-1), (D,East,W,-1), (W,exit,x,5)
- (S,East,D,-1), (D,East,C,-1), (C,East,L,-1), (L,exit,x,-5)

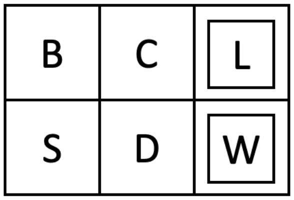


Figure 2: MDP setup. Start in S. L and W are the terminal states.

(a) [14 pts] Initialize all q-values to 0. Use $\alpha = 0.5$ and $\gamma = 0.9$. Fill in the following table as you update your values. You have space to show your work for partial credit. Leave any extra rows blank.

(s,a) to update	Updated Q-value	Show your work
(S,East)	-0.5	0.5*0 + 0.5*(-1 + 0.9*0)
(D,East)	-0.5	0.5*0 + 0.5*(-1 + 0.9*0)
(W,exit)	2.5	0.5*0 + 0.5*(5 + 0.9*0)
(S,East)	-0.75	0.5*-0.5 + 0.5*(-1+ 0.9*0)
(D,East)	-0.75	0.5*-0.5 + 0.5*(-1+0.9*0)
(C,East)	-0.5	0.5*0 + 0.5(-1 + 0.9*0)
(L,exit)	-2.5	0.5*0 + 0.5*(-5 + 0)

(b) [4 pts] Draw arrows on the map below in each non-terminal state below to represent the policy that is learned. Draw darkly so we can see them easily. Break ties randomly.

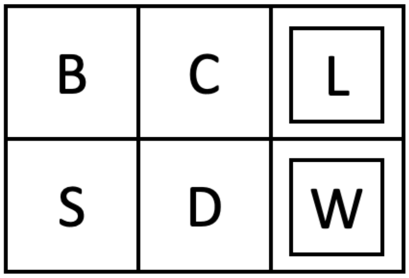


Figure 3: Draw learned policy in each non-terminal state.

(c) [14 pts] Answer the following True/False questions.

True [✓ False] Policy extraction is required after TD-learning to find the optimal policy.

True [✓ False] Policy evaluation is required after Q-learning to find the optimal policy.

True [✓ False] Decreasing α to 0 during learning will always cause Q-learning to converge to π^*.

True [✓ False] Decreasing γ to 0 during learning will always cause Q-learning to converge to π^*.

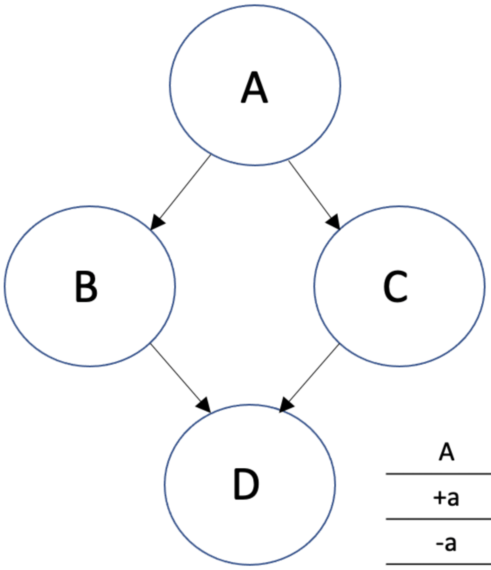
True [✓ False] An agent does not need to explore every state to learn the value of every state.

[✓ True] False An agent does not need to act optimally during Q-learning to converge to π^*.

True [✓ False] Approximate Q-learning will always converge to a policy.

Q5. [15 pts] Bayes Nets

Use the Bayes Net below for the following questions.



A	B	P(B|A)
+a	+b	0.5
+a	-b	0.5
-a	+b	0.8
-a	-b	0.2

A	C	P(C|A)
+a	+c	0.9
+a	-c	0.1
-a	+c	0.6
-a	-c	0.4

B	C	D	P(D|C,B)
+b	+c	+d	0.4
+b	+c	-d	0.6
+b	-c	+d	0.7
+b	-c	-d	0.3
-b	+c	+d	0.2
-b	+c	-d	0.8
-b	-c	+d	0.9
-b	-c	-d	0.1

A	P(A)
+a	.7
-a	.3

(a) [4 pts] Write the full equation for the probability $P(A, B, C)$ using the conditional probability tables for this Bayes net. Make sure to indicate what variables you need to sum over by using lower case letters as the summation variables. Do not substitute probabilities from the tables. Do not cancel values that sum to 1.

$P(A, B, C) =$
$\sum_d P(A)P(B|A)P(C|A)P(d|B, C)$

(b) [7 pts] Write the equation for the conditional probability $P(C|A, D)$ in terms of the given conditional probabilities $P(A)$, $P(B|A)$, $P(C|A)$ and $P(D|C, B)$. Make sure to indicate what variables you need to sum over by using lower case letters as the summation variables. Do not substitute probabilities from the tables.

$P(C|A, D) =$
$\frac{\sum_b P(A)P(b|A)P(C|A)P(D|b,C)}{\sum_b \sum_c P(A)P(b|A)P(c|A)P(D|b,c)}$

(c) [4 pts] Find the probability $P(+b, -c, +d)$. Round to the nearest thousandth. Show your work for partial credit.

$P(+b, -c, +d) =$
$\sum_a P(a)P(+b|a)P(-c|a)P(+d| + b, -c)$
$= P(+d| + b, -c) * \sum_a P(a)P(+b|a)P(-c|a)$
$= 0.7 * [[.7 * .5 * .1] + [.3 * .8 * .4]]$
$= 0.7 * [0.035 + 0.096] = 0.092$

THIS PAGE INTENTIONALLY LEFT BLANK