CS 188 Spring 2025

Intro to Artificial Intelligence Midterm

Solutions last updated: Tuesday, March 18, 2025

PRINT Your Name: _____

PRINT Your Student ID: _____

PRINT Student name to your left: _____

PRINT Student name to your right: _____

You have 110 minutes. There are 6 questions of varying credit. (100 points total)

Question:	1	2	3	4	5	6	Total
Points:	20	16	21	19	14	10	100

For questions with **circular bubbles**, you may select only one choice.

- ○ Unselected option (Completely unfilled)
- ◌✓ Don't do this (it will be graded as incorrect)
- ● Only one selected option (completely filled)

For questions with **square checkboxes**, you may select one or more choices.

- ■ You can select
- ■ multiple squares
- ☑ Don't do this (it will be graded as incorrect)

Anything you write outside the answer boxes or you ~~cross out~~ will not be graded. If you write multiple answers, your answer is ambiguous, or the bubble/checkbox is not entirely filled in, we will grade the worst interpretation.

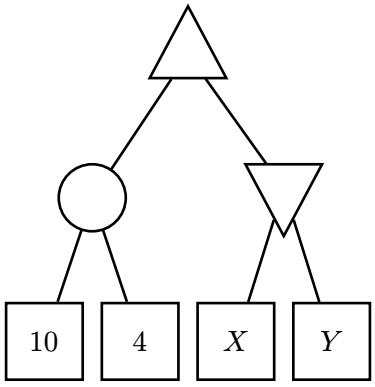
Read the honor code below and sign your name.

> By signing below, I affirm that all work on this exam is my own work. I have not referenced any disallowed materials, nor collaborated with anyone else on this exam. I understand that if I cheat on the exam, I may face the penalty of an "F" grade and a referral to the Center for Student Conduct.

SIGN your name: _____

Q1 *Potpourri* **(20 points)**

In the next 3 subparts, consider the following game tree, representing a two-player game where the players take turns. Assume that the children of the expectation node have equal probability.



Assume nodes are pruned from left to right, and we prune on equality.

Q1.1 (3 points) Can you ever prune X?

If you select "Yes", write an inequality that describes when X will be pruned. For example, you could write "$X \leq 5$" or "$X + Y > 30$".

If you select "No", leave the box blank.

○ Yes ● No

N/A

Solution: Visiting X is required because otherwise, you have no information bounding the minimizer node, so you don't know which action will be taken at the root.

Q1.2 (3 points) Can you ever prune Y?

If you select "Yes", write an inequality that describes when Y will be pruned. For example, you could write "$X \leq 5$" or "$X + Y > 30$".

If you select "No", leave the box blank.

● Yes ○ No

$X \leq 7$

Solution:

We know that the chance node has value 7.

If $X \leq 7$, then we know that the value at the minimizer node is also less than or equal to 7. At this point, we know that the root maximizer will choose the left action, and we don't need to explore Y.

Q1.3 (2 points) Suppose the maximizer node represents Pacman, and both the minimizer node and the chance node represent Blinky. What kind of situation does this specific game tree represent?

○ Pacman acts randomly in some states, but adversarially in others.

● Blinky acts randomly in some states, but adversarially in others.

○ Pacman and Blinky use different utility functions.

○ Pacman and Blinky act adversarially in all states.

Solution: Blinky is represented by a chance node and a minimizer node. After some actions by Pacman, i.e. in some successor states, Blinky acts randomly, while for others, Blinky acts adversarially. Hence, the second option is correct.

Q1.4 (4 points) Select all true statements about game trees.

■ Increasing the depth of a minimax game tree may result in a smaller value at the root node.

■ Decreasing the depth of a minimax game tree may result in a smaller value at the root node.

■ Increasing the depth of a minimax game tree may result in a larger value at the root node.

■ Decreasing the depth of a minimax game tree may result in a larger value at the root node.

○ None of the above

Solution: Changing the depth of the game tree will change the leaf nodes that are considered. This new set of leaf nodes may possess smaller or larger values, which can be propagated to the root node.

Q1.5 (3 points) Select all true statements about policy iteration.

- ☐ Policy iteration always converges in strictly fewer iterations than value iteration.
- ■ During policy evaluation, we can compute V^{π_i} using a system of linear equations.
- ■ If V^{π_i} is optimal, then the policy extraction step will not change the policy in the next iteration.
- ○ None of the above

Solution: 1) Trivially, if your random policy is optimal, then policy iteration converges faster than value iteration.

2) Trivially, if there are no rewards, then value iteration converges in the same number of iterations.

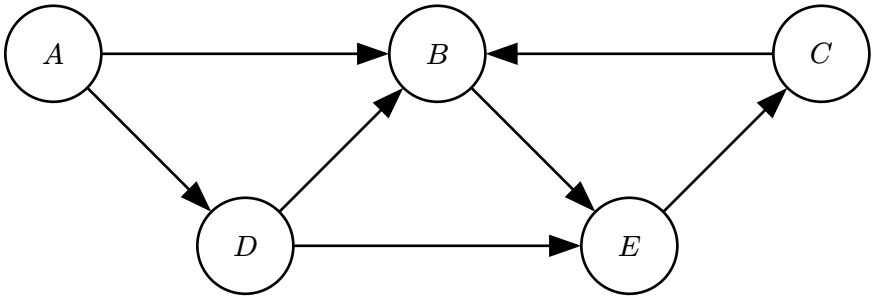
3) During policy evaluation, there are no max's in the equations, and so the value of the policy consists of entirely linear terms. You can then solve for $V_i^{\pi}(s)$ by solving a system of linear equations, where the value of each state are the unknowns.

Q1.6 (2 points) Consider policies π_1, π_2 for the same MDP. Select all true statements about their value functions.

- ■ If $\pi_1 = \pi_2$, then $Q^{\pi_1}(s, a) = Q^{\pi_2}(s, a)$ for all (s, a).
- ■ If $\pi_1 = \pi_2$, then $V^{\pi_1}(s) = V^{\pi_2}(s)$ for all s.
- ○ None of the above

Solution: When two policies are the same in an MDP, then you can solve for the V and Q values explicitly (and there is a ground truth on the values.)

Q1.7 (3 points) Consider the graph below:



Select the edges that, when removed **in isolation** (i.e., independently of the other answer choices), make the graph a valid Bayes Net.

- ☐ A → B
- ☐ A → D
- ☐ D → B
- ☐ B → E
- ☐ C → B
- ☐ E → C
- ○ None of the above

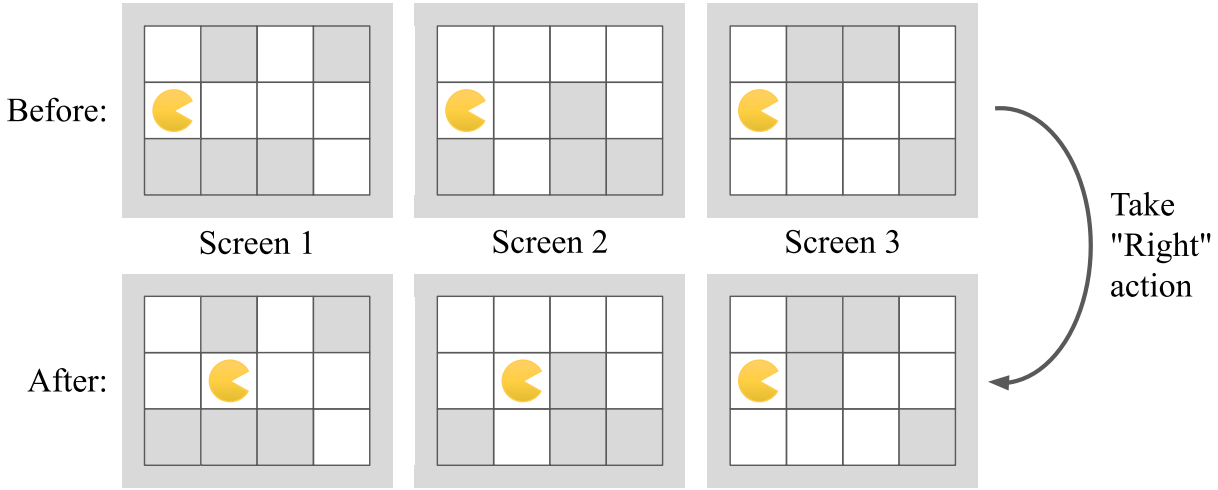
Solution: A Bayes Net must be acyclic, so removing any edge in the BEC cycle will make the graph a valid Bayes Net.

Q2 *Pac Trifecta* **(16 points)**

You are playing a new arcade game called Pac Trifecta.

- There are three screens, with one Pacman on each screen. Pacmen cannot move between screens.
- Each screen is an $M \times N$ grid. Each screen may have different walls.
- When you press a button (Up, Down, Left, or Right), each Pacman simultaneously takes that action. For example, when you press "Right", all Pacmen will attempt to move right on their screen.
- If moving in a direction causes a Pacman to hit a wall, then that Pacman stays in its current position.
- All actions cost 1.

Here is an example. Your answers should work for any arbitrary problem, **not** just the example shown.



For the next 3 subparts, suppose that our goal is to get all Pacmen in the top-left corner at the same time.

Q2.1 (3 points) What is the size of the smallest state space representation for this problem?

○ $3 \times M \times N$	● $(M \times N)^3$	○ $M^3 \times N$
○ $M \times N$	○ $M \times N^3$	○ $3^{M \times N}$

Solution: $(M \times N)^3$. Each screen has $M \times N$ possible locations for its Pacman, and we have three screens. There are no other variables in the state space that vary.

Q2.2 (2 points) Regardless of your answer to the previous subpart, assume that you use the following state space representation: three $M \times N$ boolean arrays, where a 1 denotes Pacman's location.

What is the asymptotic runtime to run the goal test on a given state?

● $O(1)$	○ $O(M)$	○ $O((M \times N)^3)$
○ $O(N)$	○ $O(M \times N)$	○ $O(3^{M \times N})$

Solution: $O(1)$. Regardless of how large the screens are, we only need to check the top-left corner of each $M \times N$ array.

Q2.3 (4 points) Let D_i be the Manhattan distance from the Pacman on the ith screen to the top-left corner of the ith screen.

Select all admissible heuristics.

- [x] $\max(D_1, D_2, D_3)$
- [x] $\min(D_1, D_2, D_3)$
- [ ] $D_1 + D_2 + D_3$
- [x] $\frac{D_1 + D_2 + D_3}{3}$
- ○ None of the above

Solution: If there were no walls at all, the Manhattan distance would be a lower bound for each Pacman to walk to the top-left corner, and adding walls can only increase the true cost. Thus, the maximum of the three Manhattan distances must be admissible. The minimum and average of the Manhattan distances are also admissible as they are strictly equal or less than the max option.

Now, consider this modified problem: The goal is to bring all Pacmen to the same position on their respective screens, not necessarily the top-left corner. The specific position doesn't matter, as long as they are all in the same position at the same time step.

Q2.4 (2 points) Assume that you use the following state representation: three $M \times N$ boolean arrays, where a 1 denotes Pacman's location.

For the modified problem, what is the asymptotic runtime to run the goal test on a given state?

- ○ $O(1)$
- ○ $O(M)$
- ○ $O((M \times N)^3)$
- ○ $O(N)$
- ● $O(M \times N)$
- ○ $O(3^{M \times N})$

Solution: $O(M \times N)$. We need to check each possible position, and see if all three arrays are True.

Oh no! Blinky has taken control of the third screen! Now, when you press a button, only the Pacmen on screens 1 and 2 follow your action, while screen 3 remains unchanged. Then, Blinky presses a button and controls the Pacman on screen 3, leaving screens 1 and 2 unchanged. You and Blinky take turns.

Q2.5 (3 points) For this subpart, your goal is to get all three Pacmen to the same position, but Blinky doesn't necessarily have this goal.

We decide to model this using a game tree.

For a depth of one (where you take a turn, then Blinky takes a turn), what is the maximum number of leaves in the game tree?

16

Solution: 16, because you take an action (with 4 possible choices) and then Blinky takes an action (with 4 possible choices).

Q2.6 (2 points) For this subpart, suppose that you both wish to bring all three Pacmen to the top-left corner in as few time steps as possible. Blinky still has separate control of the third screen.

We can formulate a single search problem and compute the optimal solution.

True or false: We can also formulate two separate search problems with strictly smaller state spaces (compared to the single search problem) and still compute the optimal solution.

● True ○ False

Solution: If the goal is top-left, we can split the problem into two independent search problems, reducing state space and therefore runtime. Your action has no bearing on what Blinky's action should be.

Q3 *Constrained Student Problem* **(21 points)**

We want to assign 100 students ($s_1, s_2, ..., s_{100}$) to either Dwinelle (D) or VLSB (V). We'd like to model this as a CSP. For the first 4 subparts, suppose each room is infinitely large.

Assume that only the first 10 students that are assigned ($s_1, ..., s_{10}$) are left-handed. VLSB has at least 10 left-handed desks and Dwinelle has no left-handed desks. Left-handed students must be assigned to left-handed desks.

Q3.1 (2 points) Is this CSP easy or hard to solve, and why?

- ● Easy, because it is under-constrained.
- ○ Easy, because it is over-constrained.
- ○ Hard, because it is under-constrained.
- ○ Hard, because it is over-constrained.

Solution: This is an under-constrained CSP, since it has a very lenient constraint. This makes the CSP easy to solve because almost any assignment is a valid solution.

Q3.2 (2 points) How many possible assignments exist (including invalid assignments)?

○ 10 ○ 100 ○ 100^2 ● 2^{100} ○ 100!

Recall the naive DFS-based algorithm from lecture. In particular, notice that we only check constraints when all variables have been assigned values, and we do not pre-process by applying unary constraints.

```
1 dfs(assignment, csp):
2     if assignment is complete and satisfies all constraints:
3         print(assignment) # found solution
4         exit # terminate program
5     else:
6         for each value in domain of the next unassigned variable:
7             new_assignment ← assignment with (variable, value) assigned
8             dfs(new_assignment, csp)
```

Q3.3 (2 points) If you always assign students to Dwinelle first on line 6, how many complete assignments does DFS examine?

○ 1 ○ Between 2 and 100 ● More than 100

Solution: If you assign to Dwinelle first, the first solution you try is "all students in Dwinelle". Then you try "99 students in Dwinelle, and one right-handed student in VLSB", "98 students in Dwinelle, the 99th student in VLSB, the 100th student in Dwinelle"... the amount of leaves grows combinatorially. It certainly tries more than 100 assignments.

Q3.4 (2 points) If you always assign students to VLSB first on line 6, how many complete assignments does DFS examine?

● 1 ○ Between 2 and 100 ○ More than 100

Solution: You immediately assign all 100 students to VLSB, which is a valid assignment.

For the rest of the question, we add a new constraint: Each room can hold at most 50 students.

Q3.5 (3 points) Suppose we enforce Dwinelle's room capacity using only one higher-order constraint. In the worst case, how many variables do you need to check to determine if this constraint is satisfied?

100

Solution: You need to check all 100, because you don't know which 51 students could be violating the constraints. (e.g. the 51st student violating the constraint could be the last student.)

Q3.6 (2 points) If you always assign students to VLSB first on line 6 with this new room constraint, how many complete assignments does DFS examine?

○ 1 ○ Between 2 and 100 ● More than 100

Solution: The naive DFS algorithm will first attempt to place 100 students in VLSB, then 99 and 1 student in Dwinelle (going through all 100 students individually), then 98, etc. It certainly tries more than 100 assignments before reaching a valid solution.

Recall backtracking search from lecture. In particular, notice that unlike DFS, we now check constraints immediately following each variable assignment. We do not pre-process by applying unary constraints.

```
1 backtracking(assignment, csp):
2     if assignment is complete:
3         print(assignment) # found solution
4         exit # terminate program
5     else:
6         for each value in domain of the next unassigned variable:
7             new_assignment ← assignment with (variable, value) assigned
8             if new_assignment does not violate any constraints:
9                 backtracking(new_assignment, csp)
```

Q3.7 (2 points) Suppose you always assign students to VLSB first. During backtracking search, how many `new_assignment`s violate a constraint on line 8?

○ 1 ● Between 2 and 100 ○ More than 100

Solution: 50 assignments will violate a constraint— the last 50 people will try to get placed in VLSB first, then assigned to Dwinelle instead.

s_1 through s_{50} in VLSB, s_{51} in Dwinelle (fail, backtrack)

s_1 through s_{50} in VLSB, s_{51} in Dwinelle, this is good

1-50 in VLSB, 51st in Dwinelle, 52nd in VLSB (fail, backtrack) ...

So there are about 50 constraints checked.

Q3.8 (2 points) Consider two unassigned students s_1 (left-handed) and s_{88} (non-left-handed).

If you enforce arc consistency between s_1 and s_{88}, what happens?

- ○ At least one value gets removed from s_1's domain.
- ○ At least one value gets removed from s_{88}'s domain.
- ○ At least one value gets removed from both variables' domains.
- ● No values get removed from either variable's domain.

Solution: No values are removed because there is not a binary constraint between them. Recall that arc consistency tests values in the head and prunes values in the tail by applying binary constraints.

Q3.9 (2 points) Suppose for this subpart only, we pre-process by applying unary constraints. What happens if you use the MRV (minimum remaining values) heuristic on this CSP?

- ○ Students are assigned to VLSB first.
- ○ Students are assigned to Dwinelle first.
- ● Left-handed students are assigned first.
- ○ Non-left-handed students are assigned first.

Solution: Left-handed students have the fewest values remaining in their domain after enforcing unary constraints, so MRV will assign them first.

Q3.10 (2 points) Pacman suggests using local search to solve this CSP.

Local search _____ guaranteed to find a solution, and when local search finds a solution, the runtime is _____ than exponential time on average.
(1) (2)

- ○ (1) is (2) slower
- ○ (1) is (2) faster
- ○ (1) is NOT (2) slower
- ● (1) is NOT (2) faster

Solution: Local search can get stuck in a local minima, and thus is not guaranteed to find a solution. However, it will often find a solution (in this case) because a majority of assignments are successful or close-to-successful, and it will run faster than exponential time.

Q4 *Modified MDP Equation* **(19 points)**

Recall the standard Bellman equation discussed in lecture:

$$Q(s,a) = \sum_{s'} T(s,a,s') \Big[R(s,a,s') + \gamma \max_{a'} Q(s',a') \Big]$$

In this equation, $\max_{a'} Q(s',a')$ is calculated by computing $Q(s',a')$ for every a', and then taking the **maximum** of all the resulting Q-values.

Consider a modified Bellman equation, where we replace the max function with an average function:

$$Q_\mu(s,a) = \sum_{s'} T(s,a,s') \Big[R(s,a,s') + \gamma \operatorname{avg}_{a'} Q_\mu(s',a') \Big]$$

In this equation, $\operatorname{avg}_{a'} Q_\mu(s',a')$ is calculated by computing $Q_\mu(s',a')$ for every a', and then taking the **average** of all the resulting Q_μ values.

Q4.1 (1 point) It is always possible to solve the **standard** Bellman equation (shown above) using a system of linear equations.

○ True ● False

Solution: This is because the max operation makes the equation nonlinear.

Q4.2 (1 point) It is always possible to solve the **modified** Bellman equation (shown above) using a system of linear equations.

● True ○ False

Solution: The avg function makes the equation linear, since it is just a sum and scalar multiple of other operands.

Q4.3 (2 points) The optimal policy extracted from the Q_μ values (in the modified equation) is ____ the same as the optimal policy extracted from the Q-values (in the standard equation).

○ never ● sometimes ○ always

Solution: In a case where there is only one available action from every state, the two policies will be the same. In a case where a state has many good actions compared to one great action and many terrible actions, the Q_μ formulation will prefer the former compared to the standard Q-values.

Q4.4 (2 points) For this subpart only, consider a state s with many successor states, all of which are terminal states. Recall that once you enter a terminal state, no future rewards are available.

Which of these statements is always true?

- ○ $Q_\mu(s,a) < Q(s,a)$ for all actions a
- ○ $Q_\mu(s,a) > Q(s,a)$ for all actions a
- ● $Q_\mu(s,a) = Q(s,a)$ for all actions a
- ○ None of the above

Solution: Note that the averaging only applies to future values. Since terminal states, by definition, acquire no future rewards, the Q-values will be the same.

Q4.5 (2 points) Given a set of Q_μ values for all state-action pairs, which of these equations extracts a policy from the Q_μ values?

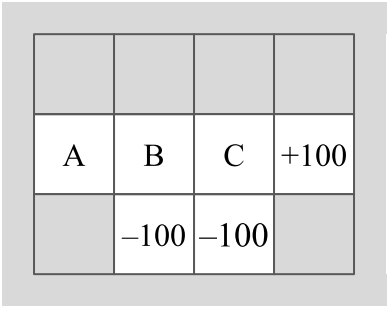
- ● For each state s, compute $\arg\max_a Q_\mu(s,a)$.
- ○ For each state s, compute $\text{avg}_a Q_\mu(s,a)$.
- ○ For each action a, compute $\max_s Q_\mu(s,a)$.
- ○ For each action a, compute $\arg\max_s Q_\mu(s,a)$.

Solution: This is the same as policy extraction using standard Q-values. Note that none of the incorrect options will even result in a policy, as we need something that gives us a choice of actions given a state.

For the rest of this question, consider the Gridworld shown. Some states are labeled with letters (A, B, C).

Assume all actions succeed with 100% probability, $\gamma = 1$, and there is 0 living reward.

Reminder: In a Gridworld, there is only one action, "Exit", available from the squares labeled $+100$ and -100, that gives the rewards of $+100$ and -100, respectively.



Q4.6 (6 points) For this Gridworld, fill in the tables for $Q(s,a)$, the Q-values from the **standard** equation.

Write one number per box. The last row has been filled in for you as an example.

(s,a)	$Q(s,a)$
$(A,\rightarrow)$	**Solution:** 100
$(B,\leftarrow)$	**Solution:** 100
$(B,\downarrow)$	**Solution:** −100
$(B,\rightarrow)$	**Solution:** 100

(s,a)	$Q(s,a)$
$(C,\leftarrow)$	**Solution:** 100
$(C,\downarrow)$	**Solution:** −100
$(C,\rightarrow)$	100

Solution: Since $\gamma = 1$, and the actions succeed with 100% probability, and there is no living reward, $Q(s,a)$ is the maximum terminal reward attainable from s given action a.

For the remaining subparts, select the correct expression for $Q_\mu(s,a)$, the Q_μ values from the **modified** equation.

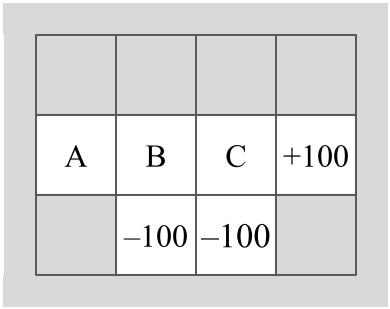
Q4.7 (1 point) $Q_\mu(C,\rightarrow)$

- ○ -100
- ● 100
- ○ $\frac{1}{3}\left[Q_\mu(A,\downarrow)+Q_\mu(B,\downarrow)+Q_\mu(C,\downarrow)\right]$
- ○ $\frac{1}{3}\left[Q_\mu(C,\rightarrow)+Q_\mu(C,\downarrow)+Q_\mu(C,\leftarrow)\right]$
- ○ $\frac{1}{3}\left[Q_\mu(B,\rightarrow)+Q_\mu(B,\downarrow)+Q_\mu(B,\leftarrow)\right]$
- ○ $\frac{1}{3}\left[Q_\mu(A,\rightarrow)+Q_\mu(B,\rightarrow)+Q_\mu(C,\rightarrow)\right]$

Q4.8 (1 point) $Q_\mu(B,\downarrow)$

- ● -100
- ○ 100
- ○ $\frac{1}{3}\left[Q_\mu(A,\downarrow)+Q_\mu(B,\downarrow)+Q_\mu(C,\downarrow)\right]$
- ○ $\frac{1}{3}\left[Q_\mu(C,\rightarrow)+Q_\mu(C,\downarrow)+Q_\mu(C,\leftarrow)\right]$
- ○ $\frac{1}{3}\left[Q_\mu(B,\rightarrow)+Q_\mu(B,\downarrow)+Q_\mu(B,\leftarrow)\right]$
- ○ $\frac{1}{3}\left[Q_\mu(A,\rightarrow)+Q_\mu(B,\rightarrow)+Q_\mu(C,\rightarrow)\right]$

The Gridworld, reprinted for your convenience:



Q4.9 (1 point) $Q_\mu(A, \rightarrow)$

- ○ -100
- ○ 100
- ○ $\frac{1}{3}\left[Q_\mu(A,\downarrow) + Q_\mu(B,\downarrow) + Q_\mu(C,\downarrow)\right]$
- ○ $\frac{1}{3}\left[Q_\mu(C,\rightarrow) + Q_\mu(C,\downarrow) + Q_\mu(C,\leftarrow)\right]$
- ● $\frac{1}{3}\left[Q_\mu(B,\rightarrow) + Q_\mu(B,\downarrow) + Q_\mu(B,\leftarrow)\right]$
- ○ $\frac{1}{3}\left[Q_\mu(A,\rightarrow) + Q_\mu(B,\rightarrow) + Q_\mu(C,\rightarrow)\right]$

Q4.10 (1 point) $Q_\mu(B, \rightarrow)$

- ○ -100
- ○ 100
- ○ $\frac{1}{3}\left[Q_\mu(A,\downarrow) + Q_\mu(B,\downarrow) + Q_\mu(C,\downarrow)\right]$
- ● $\frac{1}{3}\left[Q_\mu(C,\rightarrow) + Q_\mu(C,\downarrow) + Q_\mu(C,\leftarrow)\right]$
- ○ $\frac{1}{3}\left[Q_\mu(B,\rightarrow) + Q_\mu(B,\downarrow) + Q_\mu(B,\leftarrow)\right]$
- ○ $\frac{1}{3}\left[Q_\mu(A,\rightarrow) + Q_\mu(B,\rightarrow) + Q_\mu(C,\rightarrow)\right]$

Q4.11 (1 point) $Q_\mu(C, \leftarrow)$

- ○ -100
- ○ 100
- ○ $\frac{1}{3}\left[Q_\mu(A,\downarrow) + Q_\mu(B,\downarrow) + Q_\mu(C,\downarrow)\right]$
- ○ $\frac{1}{3}\left[Q_\mu(C,\rightarrow) + Q_\mu(C,\downarrow) + Q_\mu(C,\leftarrow)\right]$
- ● $\frac{1}{3}\left[Q_\mu(B,\rightarrow) + Q_\mu(B,\downarrow) + Q_\mu(B,\leftarrow)\right]$
- ○ $\frac{1}{3}\left[Q_\mu(A,\rightarrow) + Q_\mu(B,\rightarrow) + Q_\mu(C,\rightarrow)\right]$

Solution: For 4.7 and 4.8, the Q_μ values only average over one value, so they are the same as the standard Q-values.

For 4.9 through 4.11, we may expand out the given equation for Q_μ and see that we average over the Q_μ values over all the possible actions in the successor states. In this scenario, there is only one successor state given a state-action pair (s, a), so we only need to consider the possible actions at the successor state.

Note that an alternate interpretation of this question involved having all actions $\uparrow, \leftarrow, \rightarrow, \downarrow$ be available, with taking an action that would move you into a wall leaving you in the same state, as opposed to having those actions be illegal to begin with. However, since $\gamma = 1$, this doesn't actually change the Q_μ values.

For example, for $Q(B, \rightarrow)$, under the first interpretation, we would have

$$Q(B, \rightarrow) = \frac{1}{3}(Q(C, \rightarrow) + Q(C, \downarrow) + Q(C, \leftarrow))$$

Under the second interpretation, we would have

$$Q(B, \rightarrow) = \frac{1}{4}(Q(C, \uparrow) + Q(C, \rightarrow) + Q(C, \downarrow) + Q(C, \leftarrow))$$

and

$$\begin{aligned} Q(C, \uparrow) &= \frac{1}{4}(Q(C, \uparrow) + Q(C, \rightarrow) + Q(C, \downarrow) + Q(C, \leftarrow)) \\ \frac{3}{4}Q(C, \uparrow) &= \frac{1}{4}(Q(C, \rightarrow) + Q(C, \downarrow) + Q(C, \leftarrow)) \\ Q(C, \uparrow) &= \frac{1}{3}(Q(C, \rightarrow) + Q(C, \downarrow) + Q(C, \leftarrow)) \end{aligned}$$

Substituting for $Q(C, \uparrow)$, we would have

$$\begin{aligned} Q(B, \rightarrow) &= \frac{1}{4}\left(\frac{1}{3}(Q(C, \rightarrow) + Q(C, \downarrow) + Q(C, \leftarrow)) + Q(C, \rightarrow) + Q(C, \downarrow) + Q(C, \leftarrow)\right) \\ Q(B, \rightarrow) &= \frac{1}{4}\left(\frac{4}{3}(Q(C, \rightarrow) + Q(C, \downarrow) + Q(C, \leftarrow))\right) \\ Q(B, \rightarrow) &= \frac{1}{3}(Q(C, \rightarrow) + Q(C, \downarrow) + Q(C, \leftarrow)) \end{aligned}$$

so the other interpretation leads to the same answer.

Q5 *Reinforcement Learning* **(14 points)**

All subparts of this question are **independent**.

Pacman is running approximate Q-learning with two features. Recall that we can calculate features for a given state-action pair, and we have a weight for each feature.

$$f(s,a) = \begin{bmatrix} f_1(s,a) \\ f_2(s,a) \end{bmatrix}, \text{ and } w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$

Suppose Pacman introduces a third new feature, f_3, with corresponding weight w_3. The existing features and weights don't change.

Q5.1 (2 points) Before introducing f_3, a given state-action pair (s,a) had an approximate Q-value of 10.

After introducing f_3, what is the new approximate Q-value of (s,a)?

- ○ $10 + f_3(s,a) + w_3$
- ○ $(10 \cdot w_3) + f_3(s,a)$
- ● $10 + (w_3 \cdot f_3(s,a))$
- ○ $10 \cdot (w_3 + f_3(s,a))$

Solution: This comes from expanding the approximate Q-value calculation. Recall that we perform a dot product; i.e. after introducing f_3, the new approximate Q-value should be: $w_1 \cdot f_1(s,a) + w_2 \cdot f_2(s,a) + w_3 \cdot f_3(s,a)$. The first two products are exactly equal to the "old" approximate Q-value of 10, and so we just need to add $(w_3 \cdot f_3(s,a))$.

Q5.2 (2 points) Suppose we had already determined w_1 and w_2 such that the policy extracted using approximate Q-learning was optimal.

After introducing f_3, what should we set w_3 to be such that the policy remains optimal?

Express your answer as a single decimal number.

0

Solution: Since the original w_1 and w_2 produced the optimal policy, the only way to guarantee that the policy after introducing f_3 remains optimal is by ensuring that $w_3 \cdot f_3 = 0$. This is only satisfied if $w_3 = 0$.

As a (somewhat lengthy) proof for any non-zero value for w_3, consider two state-action pairs, (s,a_1) and (s,a_2). Note that the state s is the same.

Suppose without loss of generality that $Q(s,a_1) - Q(s,a_2) = k > 0$; that is, a_1 is optimal, with difference k. From 5.1, we know that their new Q-values after introducing f_3 will be: $Q'(s,a_1) = Q(s,a_1) + w_3 \cdot f_3(s,a_1)$, and similarly for a_2.

For any non-zero w_3, it is possible to choose a value for $f_3(s,a_1)$ and $f_3(s,a_2)$ such that the Q-value for $Q(s,a_2)$ is now better than $Q(s,a_1)$; i.e. a feature f_3 that makes a_2 the better action. The simplest choice is to set $f_3(s,a_1)$ to zero, and $f_3(s,a_2)$ to any number greater than k such as $k+1$. Then $Q'(s,a_1) = Q(s,a_1)$, but $Q'(s,a_2) = Q(s,a_2) + k + 1$— making a_2 better.

Q5.3 (2 points) In this subpart, consider this scenario:

1. We run approximate Q-learning with only the original two features f_1 and f_2. We extract an *old policy* π from the converged approximate Q-values.
2. Then, we introduce the new feature f_3, and continue running approximate Q-learning (with all three features) until convergence.
3. Finally, we extract a *new policy* π' from the converged approximate Q-values.

At the end of Step 3, it is ____ the case that $\pi(s) = \pi'(s)$ for all states s.

○ never ● sometimes ○ always

Solution: Consider that f_3 is a constant number. Then, it is possible that our third feature converges to a nonzero number without changing the policy. However, it is also possible that the new feature provides new beneficial information (e.g. if $f_1 = f_2 = 0$ and f_3 is useful), and we get a better policy from the new information.

Q5.4 (4 points) We extract a feature vector $f(s_x, a_x)$ from a state-action pair (s_x, a_x).

We extract another feature vector $f(s_y, a_y)$ from another state-action pair (s_y, a_y).

We notice that $f(s_x, a_x)$ and $f(s_y, a_y)$ are identical feature vectors.

What can we conclude about the states and actions that generated these features?
Select all that apply.

- ☐ s_x and s_y must be the same state.
- ☐ a_x and a_y must be the same action.
- ■ After convergence, the approximate Q-values computed for (s_x, a_x) and (s_y, a_y) are equal.
- ☐ The optimal actions from s_x and s_y are the same.
- ○ None of the above

Solution: The first and second options are false as (s_x, a_x) and (s_y, a_y) may just appear the same given our features, but if we had more features, then they could be different.

However, during approximate Q-learning, if our features can't distinguish the state-action pairs, then the approximate Q-values will be identical as the updates will be identical, so the third option is true.

The fourth option is false because it is possible for two states to look the same to our features, but have different actions available.

In the next 2 subparts, consider the exploration function $K(s, a) = \frac{1}{N(s,a)}$, where $N(s, a)$ is the number of times we have seen the state-action pair (s, a).

Blinky suggests modifying approximate Q-learning by incorporating $K(s, a)$.

Notation:
- $|S|$ is the number of states.
- $|A|$ is the number of actions available per state. Assume all states have the same actions available.
- $|F|$ is the number of features. Assume $|F|$ is much less than $|S|$.

Q5.5 (2 points) What is the space complexity of standard approximate Q-learning from lecture, without $K(s, a)$?

- ○ $O(|S| \times |A|)$
- ● $O(|F|)$
- ○ $O(|S|^2 \times |A|)$
- ○ $O(|F| \times |S|)$

Solution: Approximate Q-learning only stores the feature weights, not the Q-values for all state-action pairs. Therefore, it requires only $O(|F|)$ space.

Q5.6 (2 points) What is the space complexity of modified approximate Q-learning, with $K(s, a)$?

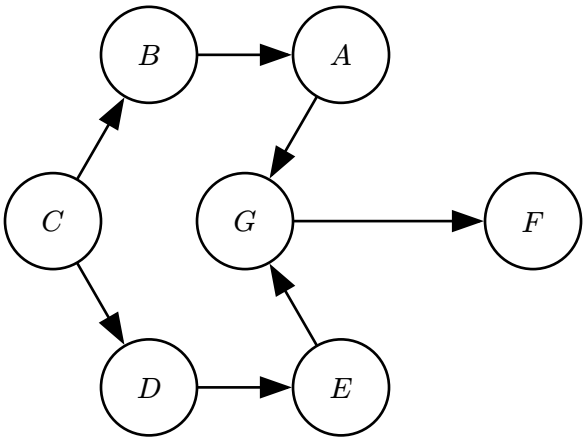
- (●) $O(|S| \times |A|)$
- ( ) $O(|F|)$
- ( ) $O(|S|^2 \times |A|)$
- ( ) $O(|F| \times |S|)$

Solution: To compute $K(s, a)$, we now have to store how often we visit each state-action pair. Therefore, we must store a table of $O(|S| \times |A|)$ values containing the values of K.

Q6 *Pac Net* **(10 points)**

Consider the Bayes Net below. All random variables are binary (each variable has two possible values).

Solution: Oh no! Somebody has turned Pacman into a Bayes Net!



Q6.1 (2 points) C is conditionally independent of F given A and E.

Solution: *If Pacman knows how his mouth functions (he knows Ate (A) and Eat (E)), then is the food (F) independent from his cerebrum (C)?*

Knowing A and E will block off the paths from C to F.

● True ○ False

Q6.2 (2 points) B is conditionally independent of D given F.

Solution: *Is Pacman's brain (B) independent of his diet (D) given food (F)?*

This is not true, because this is a common consequence; i.e. both B and D influence F, so given F they are not conditionally independent.

○ True ● False

Q6.3 (2 points) Which CPT (conditional probability table) has the fewest number of entries?

○ A ● C ○ E ○ G

○ B ○ D ○ F

Solution: C is the only CPT that represents a prior distribution— i.e. it has no parents, so it only contains 2 values. All other tables have at least 4; e.g. $P(B|C)$, the CPT for B, has four values: it conditions on C (2 possibilities) and has a value for B (2 possibilities), for $2 \times 2 = 4$.

Q6.4 (2 points) What is the sum of all values in D's CPT?

○ 0 ○ 1/2 ○ 1 ● 2 ○ 4

○ Not enough information

Solution: Since D is conditioned on C, and both are binary variables, there are two complete probability distributions ($D \mid c$, $D \mid \neg c$). The sum of a complete probability distribution is one (by rules of probability), so the answer is 2.

Q6.5 (2 points) What is the sum of all values in G's CPT?

○ 0 ○ 1/2 ○ 1 ○ 2 ● 4

○ Not enough information

Solution: Similarly, the CPT for G is $P(G \mid A, E)$. There are four probability distributions, given by the four unique conditioned upon case. The sum of a complete probability distribution is one (by rules of probability), so the answer is 4.

i.e. the four probability distributions are:

$$P(G \mid a, e),$$
$$P(G \mid a, \neg e),$$
$$P(G \mid \neg a, e),$$
$$P(G \mid \neg a, \neg e).$$

Comment Box

Congrats for making it to the end of the exam! Leave any thoughts, comments, feedback, or doodles here.

Nothing in the comment box will affect your grade.