

Homework 2

Instructions

Points: Please see the points for each problem.

Submission Instructions: Please submit a PDF in Canvas.

Points

Question	Part	Points Possible	Points Earned
1. Minimax and α-β pruning	(a)	4	_____
	(b)	4	_____
2. MDP and Value Iteration	(a)	4	_____
	(b)	4	_____
	(c)	4	_____
3. Policy Gradient and PPO	(a)	3	_____
	(b)	3	_____
Total		26	_____

1. Minimax and α - β pruning Consider the minimax tree shown below for parts (a) and (b).

[8]

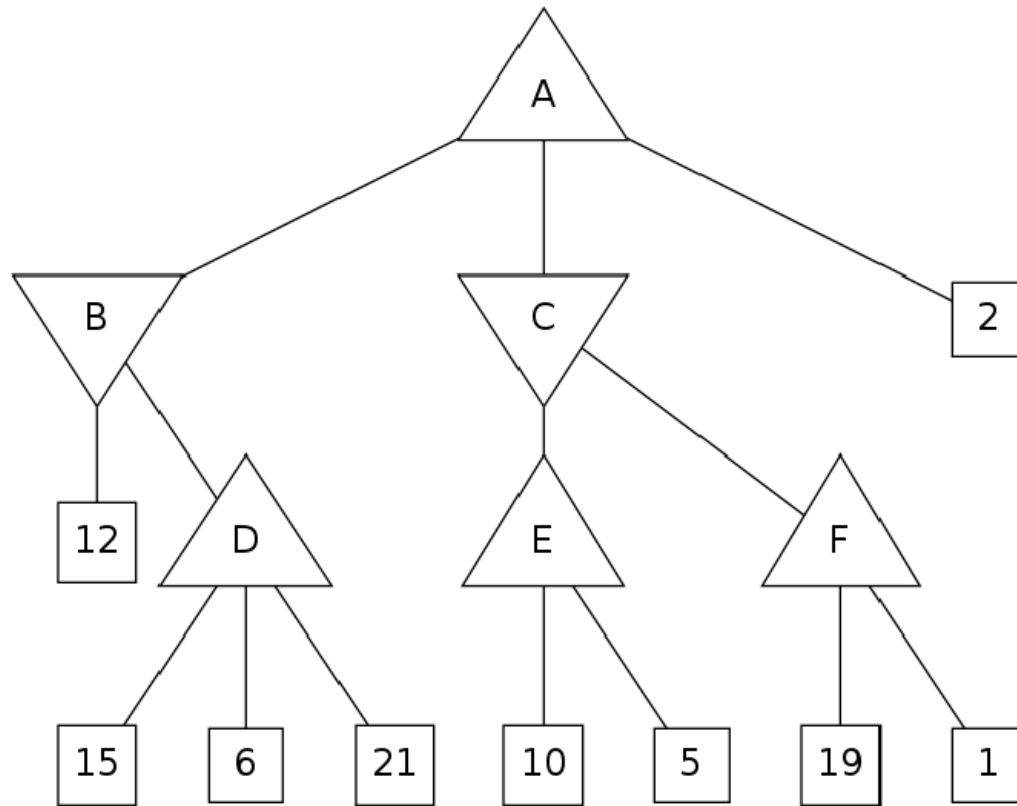


Figure 1: Minimax tree for Question 1.

- (a) What value will the root node A have?

[4]

Solution: From the leaves, compute internal node values bottom-up:

$$D = \max\{15, 6, 21\} = 21,$$

$$B = \min\{12, D\} = 12,$$

$$E = \max\{5, 10\} = 10,$$

$$F = \max\{1, 19\} = 19,$$

$$C = \min\{E, F\} = 10,$$

$$A = \max\{B, C, 2\} = 12.$$

Thus, the root value is $A = 12$.

- (b) Cross off the nodes that are pruned by α - β pruning. Assume standard left-to-right traversal. If a non-terminal state (A, B, C, D, E , or F) is pruned, cross off the entire subtree.

[4]

Solution: With left-to-right traversal, the leaf with value 6 and the leaf with value 21 under D are pruned, and the entire subtree under F is pruned.

2. Pacman is using MDPs and Value Iteration to maximize his expected utility. He has the standard actions {North, East, South, West} unless blocked by an outer wall. There is a

[12]

reward of 1 when eating a dot. The game ends when the dot is eaten.



Figure 2: Grid for Question 2.

- (a) Consider the grid where there is a single food pellet in the bottom-right corner (F) as shown in figure 2. The discount factor is $\gamma = 0.5$. There is no living reward. The states are the grid locations A, B, C, D, E, F . What is the optimal policy for each state? [4]

State s	Policy $\pi(s)$
A	
B	
C	
D	
E	

Solution: An optimal policy (one of possibly several, where ties are allowed) is:

State s	Policy $\pi(s)$
A	East or South
B	East or South
C	South
D	East
E	East

- (b) What is the optimal value for the upper-left corner state A ? [4]

Solution: $V^*(A) = 0.375$.

We use value iteration with the Bellman equation. The reward $R(s, a, s') = 1$ when transitioning into the terminal state F (eating the dot), and 0 otherwise. The value iteration update is:

$$V^k(s) = \max_a \left[R(s, a, s') + \gamma V^{k-1}(s') \right]$$

where s' is the state reached by taking action a from state s , and $\gamma = 0.5$.

Value Iteration Table:

k	$V^k(A)$	$V^k(B)$	$V^k(C)$	$V^k(D)$	$V^k(E)$	$V^k(F)$
0	0	0	0	0	0	0
1	0	1	0	0	1	1
2	0	0.5	1.5	0.5	1.5	1
3	0.25	0.75	1.5	0.75	1.5	1
4	0.375	0.75	1.5	0.75	1.5	1
5	0.375	0.75	1.5	0.75	1.5	1

Explanation by iteration:

Iteration $k = 0$: All values initialized to 0, including $V^0(F) = 0$.

Iteration $k = 1$: The terminal state F gets its value:

$$V^1(F) = 1 \quad (\text{terminal state: when you're in } F, \text{ you've already received reward } R = 1 \text{ and the game ends})$$

Since F is terminal, once you reach it, you get the reward of 1 and the episode ends. Therefore, $V^k(F) = 1$ for all $k \geq 1$. The value stays 1 because there are no future actions or rewards after reaching the terminal state.

States that can reach F in one step receive the reward:

$$V^1(E) = R(E, \text{East}, F) + \gamma V^0(F) = 1 + 0.5 \cdot 0 = 1$$

$$V^1(B) = R(B, \text{action}, F) + \gamma V^0(F) = 1 + 0.5 \cdot 0 = 1 \quad (\text{can reach } F \text{ via optimal path})$$

$$V^1(A) = \max_a [0 + \gamma \cdot 0] = 0 \quad (\text{cannot reach } F \text{ in one step})$$

$$V^1(C) = \max_a [0 + \gamma \cdot 0] = 0 \quad (\text{cannot reach } F \text{ in one step})$$

$$V^1(D) = \max_a [0 + \gamma \cdot 0] = 0 \quad (\text{cannot reach } F \text{ in one step})$$

Iteration $k = 2$: States update based on values from iteration 1:

$$V^2(F) = 1 \quad (\text{terminal state, unchanged})$$

$$V^2(E) = R(E, \text{East}, F) + \gamma V^1(F) = 1 + 0.5 \cdot 1 = 1.5$$

$$V^2(C) = R(C, \text{South}, E?) + \gamma V^1(E) = 0 + 0.5 \cdot 1 = 0.5, \text{ or } \max\{\text{other actions}\} = 1.5 \quad (\text{optimal path gives higher value})$$

$$V^2(D) = R(D, \text{South}, F) + \gamma V^1(F) = 1 + 0.5 \cdot 1 = 1.5, \text{ or } \max\{R(D, \text{East}, E) + \gamma V^1(E)\} = 0 + 0.5 \cdot 1 = 0.5$$

$$V^2(B) = \max_a [0 + \gamma V^1(\text{next state})] = \max\{0.5 \cdot 1, 0.5 \cdot 0\} = 0.5$$

$$V^2(A) = \max_a [0 + \gamma V^1(\text{next state})] = \max\{0.5 \cdot 0, 0.5 \cdot 0\} = 0$$

Iteration $k = 3$:

$$V^3(F) = 1 \quad (\text{terminal state, unchanged})$$

$$V^3(E) = 1 + 0.5 \cdot 1 = 1.5 \quad (\text{unchanged, optimal to go East to } F)$$

$$V^3(C) = 1.5 \quad (\text{unchanged, optimal path established})$$

$$V^3(D) = \max\{1 + 0.5 \cdot 1, 0 + 0.5 \cdot 1.5\} = \max\{1.5, 0.75\} = 1.5, \text{ or } 0.75$$

$$V^3(B) = \max\{0 + 0.5 \cdot 1.5, 0 + 0.5 \cdot 0.5\} = \max\{0.75, 0.25\} = 0.75$$

$$V^3(A) = \max\{0 + 0.5 \cdot 0.5, 0 + 0.5 \cdot 1.5\} = \max\{0.25, 0.75\} = 0.75, \text{ or } 0.25$$

Iteration $k = 4$: Values update from iteration 3:

$$V^4(F) = 1 \quad (\text{terminal state, unchanged})$$

$$V^4(E) = R(E, \text{East}, F) + \gamma V^3(F) = 1 + 0.5 \cdot 1 = 1.5 = V^3(E) \quad (\text{unchanged})$$

$$V^4(C) = 1.5 = V^3(C) \quad (\text{unchanged, optimal path established})$$

$$\begin{aligned} V^4(D) &= \max\{R(D, \text{South}, F) + \gamma V^3(F), R(D, \text{East}, E) + \gamma V^3(E)\} \\ &= \max\{1 + 0.5 \cdot 1, 0 + 0.5 \cdot 1.5\} = \max\{1.5, 0.75\} = 0.75 = V^3(D) \end{aligned}$$

$$\begin{aligned} V^4(B) &= \max\{0 + \gamma V^3(C), 0 + \gamma V^3(D)\} \\ &= \max\{0 + 0.5 \cdot 1.5, 0 + 0.5 \cdot 0.75\} = \max\{0.75, 0.375\} = 0.75 = V^3(B) \end{aligned}$$

$$\begin{aligned} V^4(A) &= \max\{0 + \gamma V^3(B), 0 + \gamma V^3(C)\} \\ &= \max\{0 + 0.5 \cdot 0.75, 0 + 0.5 \cdot 1.5\} = \max\{0.375, 0.75\} = 0.375 \end{aligned}$$

Note that $V^4(A) = 0.375$ changed from $V^3(A) = 0.25$, while other states remained unchanged from iteration 3.

Iteration $k = 5$: No change in any state value:

$$V^5(F) = 1 = V^4(F)$$

$$V^5(E) = 1.5 = V^4(E)$$

$$V^5(C) = 1.5 = V^4(C)$$

$$V^5(D) = 0.75 = V^4(D)$$

$$V^5(B) = 0.75 = V^4(B)$$

$$V^5(A) = 0.375 = V^4(A)$$

Since $V^5(s) = V^4(s)$ for all states s , the algorithm has converged. We stop value iteration when there is no change in any state value between consecutive iterations, which indicates we have reached the optimal values $V^*(s)$.

The optimal value $V^*(A) = 0.375$ is reached at iteration $k = 4$. This represents the expected discounted reward from state A following the optimal policy, where the discount factor $\gamma = 0.5$ reduces the value of future rewards.

- (c) Using value iteration initialized with $V^0(\cdot) = 0$, at which iteration k does $V^k(A)$ first equal $V^*(A)$? [4]

Solution: At iteration $k = 4$. Iteration $k = 5$ matches iteration $k = 4$.

3. Consider some of the newer RL algorithms such as Policy Gradient and Proximal Policy Optimization. [6]

- (a) What issues with RL are they designed to solve? [3]

Solution: Policy Gradient and PPO algorithms address several key issues in RL:

- **Sample inefficiency:** Traditional value-based methods (like Q-learning) require many samples to learn good policies, especially in high-dimensional or continuous

action spaces.

- **High variance:** Policy gradient methods suffer from high variance in gradient estimates, making training unstable.
- **Stability:** Vanilla policy gradients can make large, destructive policy updates that degrade performance.
- **On-policy data requirement:** Many RL algorithms require on-policy data, making sample reuse difficult and inefficient.
- **Continuous action spaces:** Value-based methods struggle with continuous actions, while policy gradients handle them naturally.

Note: Stating any 3 of these is sufficient for full credit.

(b) How do they improve on previous RL algorithms?

[3]

Solution: Policy Gradient and PPO improve on previous RL algorithms in several ways:

- **Direct policy optimization:** Policy gradients optimize the policy directly, avoiding the need to learn value functions first (unlike value-based methods like Q-learning).
- **PPO clipping:** PPO uses a clipped objective function that prevents large policy updates, maintaining training stability while allowing multiple updates from the same data.
- **Better sample efficiency:** PPO can perform multiple gradient updates on the same batch of data, improving sample efficiency compared to vanilla policy gradients.
- **Natural continuous actions:** Policy gradients parameterize policies directly, making them well-suited for continuous action spaces without discretization.
- **Reduced variance:** PPO's clipped objective and trust region approach reduce variance compared to standard policy gradient methods.
- **Stable learning:** The clipping mechanism ensures policy updates stay within a trust region, preventing performance collapse that can occur with large updates.

Note: Stating any 3 of these is sufficient for full credit.