## 1 MDPs: Warm-Up

1. What is the Markov Property?
2. What are the Bellman equations, and when are they used?
3. What is a policy? What is an optimal policy?
4. How does the discount factor $\gamma$ affect how the agent finds the optimal policy? Why do we restrict gamma $\gamma < 1$?
5. Fill in the following table explaining the effects of having different gamma values:

| $\gamma$ | **Effect on policy search:** |
|---|---|
| $\gamma = 0$ | |
| $\gamma = 1$ | |

6. What are the two steps to Policy Iteration?
7. What is the relationship between $V^*(s)$ and $Q(s, a)$?
8. (MDP Notation Review) Draw a line connecting each term in the left column with its corresponding equation in the right column.

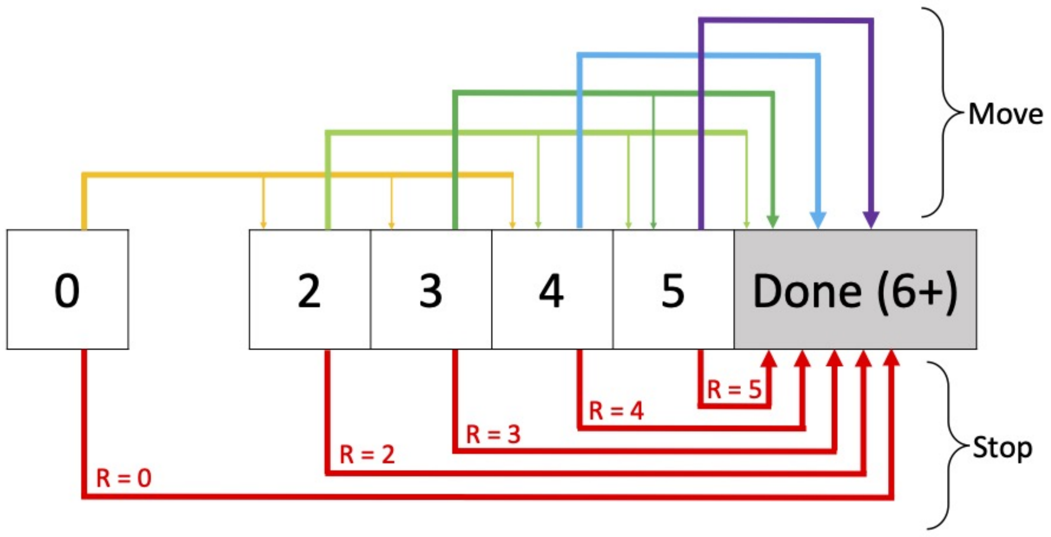
| *Term* | *Equation* |
|---|---|
| Standard Expectimax ● | ● $\pi_V(s) = \text{argmax}_a \sum_{s'} P(s'\|s,a)[R(s,a,s')] + \gamma V(s')], \forall s$ |
| Bellman Equation ● | ● $V_{k+1}(s) = \max_a \sum_{s'} P(s'\|s,a)[R(s,a,s')] + \gamma V_k(s')], \forall s$ |
| Value Iteration ● | ● $V^*(s) = \max_a \sum_{s'} P(s'\|s,a)[R(s,a,s')] + \gamma V^*(s')]$ |
| Q-Iteration ● | ● $Q_{k+1}(s,a) = \sum_{s'} P(s'\|s,a)[R(s,a,s')] + \gamma \max_{a'} Q_k(s',a')], \forall s, a$ |
| Policy Extraction ● | ● $V^{\pi}_{k+1}(s) = \sum_{s'} P(s'\|s,\pi(s))[R(s,\pi(s),s')] + \gamma V^{\pi}_k(s')], \forall s$ |
| Policy Evaluation ● | ● $\pi_{new}(s) = \text{argmax}_a \sum_{s'} P(s'\|s,a)[R(s,a,s')] + \gamma V^{\pi_{old}}(s')], \forall s$ |
| Policy Improvement ● | ● $V(s) = \max_a \sum_{s'} P(s'\|s,a)V(s')$ |

## 2 MDPs: Racing

Consider a modification of the racing robot car example seen in lecture. In this game, the car repeatedly moves a random number of spaces that is equally likely to be 2, 3, or 4. The car can either Move or Stop if the total number of spaces moved is less than 6.

If the total spaces moved is 6 or higher, the game automatically ends, and the car receives a reward of 0. When the car Stops, the reward is equal to the total spaces moved (up to 5), and the game ends. There is no reward for the Move action.

Let's formulate this problem as an MDP with the states $\{0, 2, 3, 4, 5, Done\}$.



1. What is the transition function for this MDP? (You should specify discrete values for specific state/action inputs.)
2. What is the reward function for this MDP?
3. Recall the value iteration update equation:

$$V_{k+1}(s) = \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V_k(s')]$$

   Perform value iteration for 4 iterations with $\gamma = 1$.

| States | 0 | 2 | 3 | 4 | 5 | Done |
|---|---|---|---|---|---|---|
| $V_0$ | | | | | | 0 |
| $V_1$ | | | | | | 0 |
| $V_2$ | | | | | | 0 |
| $V_3$ | | | | | | 0 |
| $V_4$ | | | | | | 0 |

4. You should have noticed that value iteration converged above. What is the optimal policy?

| States | 0 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $\pi^*$ | | | | | |

5. How would our results change with $\gamma = 0.1$?
6. Now imagine we changed the rules so that moving to spaces after 5 does not immediately end the game, but rather wraps back around to the beginning states (this is the case where the states exist side by side in a loop).

   For example if you start in state 5, select the action Move, and move 2 spaces, you would end up in state 1 (this is a new state we would need introduce).

   To be clear, the states would now be states {0,1,2,3,4,5, *Done*}.

   How would this modification change our results? (Assume we again use $\gamma$=1)

## 3 MDPs: Policy Iteration

Recall the racing MDP from the prior problem. In this game, the car repeatedly moves a random number of spaces that is equally likely to be 2, 3, or 4. The car can either Move or Stop if the total number of spaces moved is less than 6.
If the total spaces moved is 6 or higher, the game automatically ends, and the car receives a reward of 0. When the car Stops, the reward is equal to the total spaces moved (up to 5), and the game ends. There is no reward for the Move action.
States: $\{0, 2, 3, 4, 5, Done\}$

| Transition | Reward |
|---|---|
| $T(s, Stop, Done) = 1, \quad \forall s \neq Done$ | $R(s, Stop, Done) = s, \quad \forall s \leq 5$ |
| $T(0, Move, s') = \frac{1}{3}, \quad \forall s' \in \{2, 3, 4\}$ | $R(s, a, s') = 0$ otherwise |
| $T(2, Move, s') = \frac{1}{3}, \quad \forall s' \in \{4, 5, Done\}$ | |
| $T(3, Move, 5) = \frac{1}{3}$ | |
| $T(3, Move, Done) = \frac{2}{3}$ | |
| $T(4, Move, Done) = 1$ | |
| $T(5, Move, Done) = 1$ | |
| $T(s, a, s') = 0$ otherwise | |

Now recall the policy evaluation and policy improvement equations, which together make up policy iteration.

$$\text{Policy Evaluation: } V_{k+1}^{\pi}(s) = \sum_{s'} T(s, \pi(s), s')[R(s, \pi(s), s') + \gamma V_k^{\pi}(s')]$$

$$\text{Policy Improvement: } \pi_{new}(s) \leftarrow \text{argmax}_a \sum_{s'} T(s, a, s')[R(s, a, s') + \gamma V^{\pi_{old}}(s')]$$

Perform two iterations of policy iteration for one step of this MDP, starting from the fixed policy below. Use the initial $\gamma = 1$.

| States | 0 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $\pi_0$ | Move | Stop | Move | Stop | Move |
| $V_0^{\pi_0}$ | | | | | |
| $V_1^{\pi_0}$ | | | | | |
| $V_2^{\pi_0}$ | | | | | |
| $V_3^{\pi_0}$ | | | | | |
| $\pi_1$ | | | | | |
| $V_0^{\pi_1}$ | | | | | |
| $V_1^{\pi_1}$ | | | | | |
| $V_2^{\pi_1}$ | | | | | |
| $V_3^{\pi_1}$ | | | | | |
| $\pi_2$ | | | | | |

## 4 MDPs: Conceptual Questions

1. What are the key distinctions between the value iteration and policy iteration algorithms, and when might you prefer one to the other?
2. What are some limitations of value iteration? What are some limitations of policy iteration?
3. When does policy iteration end? Immediately after policy iteration ends (without performing additional computation), do we have the values of the optimal policy?
4. What changes if during policy iteration, you only run one iteration of Bellman update instead of running it until convergence? Do you still get an optimal policy?

## 5 RL: What's Changed Since MDPs?

1. Recall the Bellman Equation we used in MDPs to determine the value of a given state:

$$V^*(s) = \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V(s')]$$

   What information do we no longer have direct access to in the transition to RL?

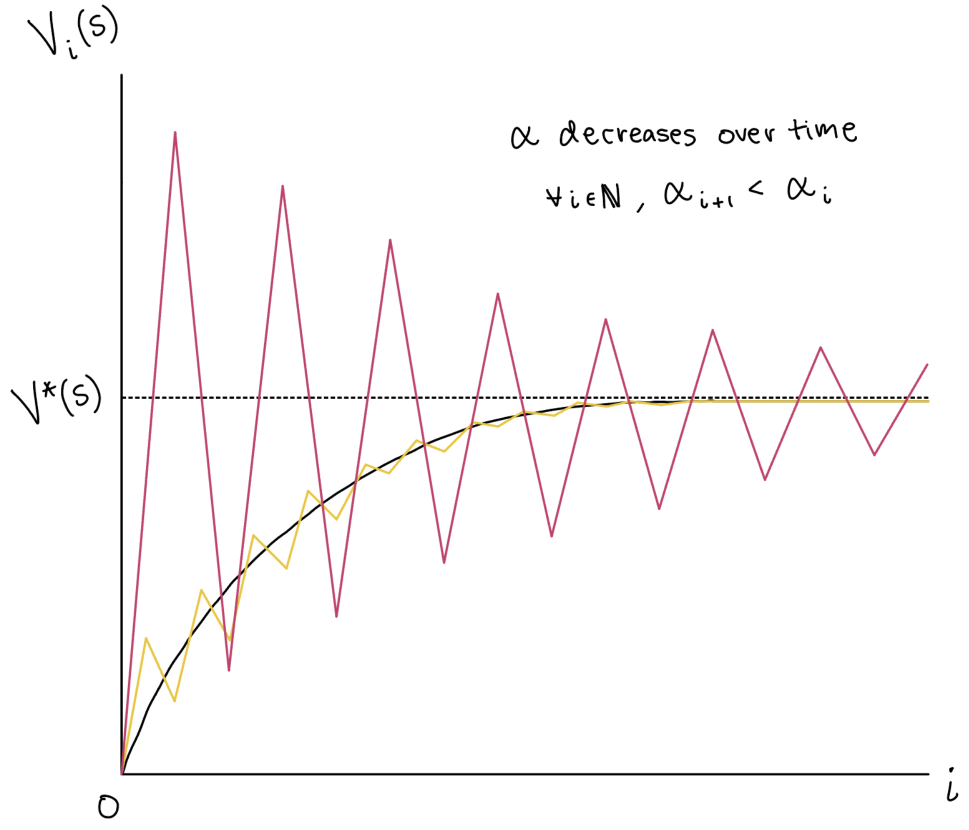
2. What is the difference between online and offline learning? Which type of learning does MDP use? How about RL?

## 6 RL: Conceptual Questions

Recall that in Q-learning, we continually update the values of each Q-state by learning through a series of episodes, ultimately converging upon the optimal policy.

1. What's the main shortcoming of TD learning that Q-learning resolves?

2. We are given two runs of TD-learning using the same sequence of samples but different $\alpha$ values depicted in the plot below. Assume the dashed horizontal line represents the optimal value for a specific state $s$ and the black curve represents the smoothest transition to the optimal value given this sequence of samples. In both runs $\alpha$ decreases over time (or iterations), but one run has $\alpha$ values larger than the other run at any point in time. Which run (red or yellow) corresponds to the smaller values of $\alpha$? How do the relative sizes of $\alpha$ affect the rate of convergence to the optimal value?



3. We are given a pre-existing table of current estimate of Q-values (and its corresponding policy), and asked to perform $\epsilon$-greedy Q-learning. Individually, what effect does setting each of the following constants to 0 have on this process?
   Remember that in $\epsilon$-greedy Q-learning, we follow the following formulation for choosing our action:

$$\text{action at time } t = \begin{cases} \underset{Q(s,a)}{\arg\max} & \text{with probability } 1-\epsilon \\ \text{any action } a & \text{with probability } \epsilon \end{cases}$$

   (a) $\alpha$
   (b) $\gamma$
   (c) $\epsilon$

4. Consider a variant of the $\epsilon$-greedy Q-learning algorithm that is changed such that instead of using the policy extracted from our current Q-values, we use a fixed policy instead. We still perform exploration with probability $\epsilon$. If this fixed policy happens to be optimal, how does the performance of this algorithm compare to normal $\epsilon$-greedy Q-learning?

5. Recall the count exploration function used in the modified Q-update:

$$f(u,n) = u + \frac{k}{n+1}$$

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \text{max}_{a'} f(Q(s',a'), N(s',a')) - Q(s,a)]$$

   Remember that $k$ is a hyperparameter that the designer chooses, and $N(s',a')$ is the number of times we've visited the $(s',a')$ pair. What is the effect of increasing or decreasing $k$?

6. Let's revisit the Nim code. What RL strategies does `AgentRL` employ? Does it evaluate states or Q-states?

7. Contrast the following pairs of reinforcement learning terms:
   (i) Off-policy vs. on-policy learning
   (ii) Model-based vs. model-free
   (iii) Passive vs. active