CS 188
Fall 2022

Regular Discussion 5 Solutions

Solving MDPs

Here, Transition Probabilities and Reward Functions are known, and we want to find the optimal policy for the agent to maximize its discounted total utility. There are two ways:

1. Value Iteration with Policy Extraction
Repeat Value Iteration until convergence, then run Policy Extraction.
Value Iteration: $\forall s \in S,\ V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V_k(s')]$
Policy Extraction: $\forall s \in S,\ \pi^*(s) = \underset{a}{\operatorname{argmax}}\, Q^*(s,a) = \underset{a}{\operatorname{argmax}} \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$

2. Policy Iteration
Repeatedly perform Policy Evaluation then Policy Improvement.
Policy Evaluation:
Primary Method: $V^\pi(s) = \sum_{s'} T(s,\pi(s),s')[R(s,\pi(s),s') + \gamma V^\pi(s')]$
Solve the system of equations for each $V^\pi(s)$.
Alternate Method: $V^{\pi_i}_{k+1}(s) \leftarrow \sum_{s'} T(s,\pi_i(s),s')[R(s,\pi_i(s),s') + \gamma V^{\pi_i}_k(s')]$
Repeat until convergence.
Policy Improvement: $\pi_{i+1}(s) = \underset{a}{\operatorname{argmax}} \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^{\pi_i}(s')]$

Reinforcement Learning

In RL problems, T and R are not known. Instead, we observe episodes of the agent's behavior with several (s,a,s',r) samples.

Model-Based Learning

Create a model to approximate $\hat{T}(s,a,s')$ and $\hat{R}(s,a,s')$. We keep counts of the (s,a,s',r) samples observed, then calculate $\hat{T}$ and $\hat{R}$ based on these counts.
$\hat{T}(s,a,s') = \frac{\text{count}(s,a,s')}{\text{count}(s,a)}$, and $\hat{R}(s,a,s') = r$
As the number of samples increases, we converge to the true T and R.

Model-Free Learning

On-policy methods try to learn the value of states according to a certain (fixed) policy. Off-policy methods try to learn other information that could be used to find a good policy.

Direct Evaluation (On-policy)
For each state, average out the discounted total reward from all the episodes that visited that state.
$V^\pi(s) = \frac{\text{sum of discounted total rewards of episodes that visit s}}{\text{number of episodes that visit s}}$

Temporal Difference Learning (On-policy)

Learn from every experience by incorporating sample information into an exponential moving average.

$\text{sample}_k = R(s, \pi(s), s') + \gamma V^{\pi}_{k-1}(s')$

$V^{\pi}_k(s) \leftarrow (1-\alpha)V^{\pi}_{k-1}(s) + \alpha \cdot \text{sample}_k$

Q-Learning (Off-policy)

Q-value Iteration. $Q_{k+1}(s,a) \leftarrow \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma \max_{a'} Q_k(s',a')]$

Incorporate samples into exponential moving average:

$\text{sample} = R(s,a,s') + \gamma \max_{a'} Q(s',a')$

$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha \cdot \text{sample}$

Approximate Q-learning (Off-policy)

Approximate Q and V functions as linear combinations of features.

$V(s) = w_1 \cdot f_1(s) + w_2 \cdot f_2(s) + \ldots + w_n \cdot f_n(s) = \vec{w} \cdot \vec{f}(s)$

$Q(s,a) = w_1 \cdot f_1(s,a) + w_2 \cdot f_2(s,a) + \ldots + w_n \cdot f_n(s,a) = \vec{w} \cdot \vec{f}(s,a)$

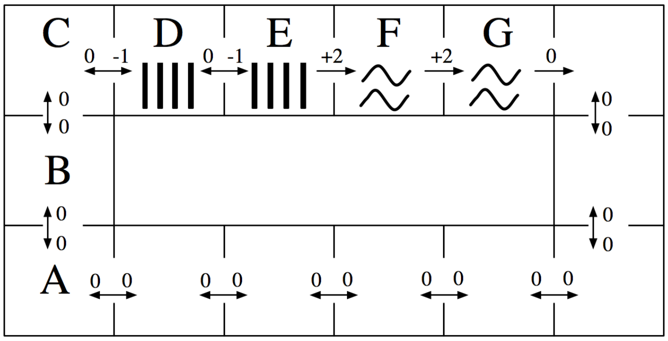
Update Rule:

$\text{diff} = [R(s,a,s') + \gamma \max_{a'} Q(s',a')] - Q(s,a)$

$w_i \leftarrow w_i + \alpha \cdot \text{diff} \cdot f_i(s,a)$

1 MDPs: Grid-World Water Park

Consider the MDP drawn below. The state space consists of all squares in a grid-world water park. There is a single waterslide that is composed of two ladder squares and two slide squares (marked with vertical bars and squiggly lines respectively). An agent in this water park can move from any square to any neighboring square, unless the current square is a slide in which case it must move forward one square along the slide. The actions are denoted by arrows between squares on the map and all deterministically move the agent in the given direction. The agent cannot stand still: it must move on each time step. Rewards are also shown below: the agent feels great pleasure as it slides down the water slide (+2), a certain amount of discomfort as it climbs the rungs of the ladder (-1), and receives rewards of 0 otherwise. The time horizon is infinite; this MDP goes on forever.



(a) How many (deterministic) policies π are possible for this MDP?

2^{11}

(b) Fill in the blank cells of this table with values that are correct for the corresponding function, discount, and state. *Hint: You should not need to do substantial calculation here.*

	γ	$s = A$	$s = E$
$V_3(s)$	1.0	0	4
$V_{10}(s)$	1.0	2	4
$V_{10}(s)$	0.1	0	2.2
$Q_1(s, \text{west})$	1.0	——	0
$Q_{10}(s, \text{west})$	1.0	——	3
$V^*(s)$	1.0	∞	∞
$V^*(s)$	0.1	0	2.2

$V^*_{10}(A), \gamma = 1$: In 10 time steps with no discounting, the rewards don't decay, so the optimal strategy is to climb the two stairs (-1 reward each), and then slide down the two slide squares (+2 rewards each). You only have time to do this once. Summing this up, we get $-1 - 1 + 2 + 2 = 2$.

$V^*_{10}(E), \gamma = 1$: No discounting, so optimal strategy is sliding down the slide. That's all you have time for. Sum of rewards $= 2 + 2 = 4$.

$V^*_{10}(A), \gamma = 0.1$. The discount rate is 0.1, meaning that rewards 1 step further into the future are discounted by a factor of 0.1. Let's assume from A, we went for the slide. Then, we would have to take the actions $A \to B, B \to C, C \to D, D \to E, E \to F, F \to G$. We get the first -1 reward from $C \to D$, discounted by γ^2 since it is two actions in the future. $D \to E$ is discounted by γ^3, $E \to F$ by γ^4, and $F \to G$ by γ^5. Since γ is low, the positive rewards you get from the slide have less of an effect as the larger negative rewards you get from climbing up. Hence, the sum of rewards of taking the slide path would be negative; the optimal value is 0.

$V^*_{10}(E), \gamma = 0.1$. Now, you don't have to do the work of climbing up the stairs, and you just take the slide down. Sum of rewards would be 2 (for $E \to F$) + 0.2 (for $F \to G$, discounted by 0.1) = 2.2.

$Q^*_{10}(E, west), \gamma = 1$. Remember that a Q-state (s,a) is when you start from state s and are committed to taking a. Hence, from E, you take the action West and land in D, using up one time step and getting an immediate reward of 0. From D, the optimal strategy is to climb back up the higher flight of stairs and then slide down the slide. Hence, the rewards would be $-1(D \to E) + 2(E \to F) + 2(F \to G) = 3$.

$V^*(s), \gamma = 1$. Infinite game with no discount? Have fun sliding down the slide to your content from anywhere.

$V^*(s), \gamma = 0.1$. Same reasoning apply to both A and E from $V^*_{10}(s)$. With discounting, the stairs are more costly to climb than the reward you get from sliding down the water slide. Hence, at A, you wouldn't want to head to the slide. From E, since you are already at the top of the slide, you should just slide down.

(c) Fill in the blank cells of this table with the Q-values that result from applying the Q-update for the transition specified on each row. You may leave Q-values that are unaffected by the current update blank. Use discount $\gamma = 1.0$ and learning rate $\alpha = 0.5$. Assume all Q-values are initialized to 0. (Note: the specified transitions would not arise from a single episode.)

	$Q(D, \text{west})$	$Q(D, \text{east})$	$Q(E, \text{west})$	$Q(E, \text{east})$
Initial:	0	0	0	0
Transition 1: $(s = D, a = \text{east}, r = -1, s' = E)$		-0.5		
Transition 2: $(s = E, a = \text{east}, r = +2, s' = F)$				1.0
Transition 3: $(s = E, a = \text{west}, r = 0, s' = D)$				
Transition 4: $(s = D, a = \text{east}, r = -1, s' = E)$		-0.25		

2 Pacman with Feature-Based Q-Learning

We would like to use a Q-learning agent for Pacman, but the size of the state space for a large grid is too massive to hold in memory. To solve this, we will switch to feature-based representation of Pacman's state.

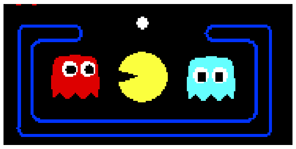
(a) We will have two features, F_g and F_p, defined as follows:

$$F_g(s,a) = A(s) + B(s,a) + C(s,a)$$
$$F_p(s,a) = D(s) + 2E(s,a)$$

where

$A(s)$ = number of ghosts within 1 step of state s
$B(s,a)$ = number of ghosts Pacman touches after taking action a from state s
$C(s,a)$ = number of ghosts within 1 step of the state Pacman ends up in after taking action a
$D(s)$ = number of food pellets within 1 step of state s
$E(s,a)$ = number of food pellets eaten after taking action a from state s

For this pacman board, the ghosts will always be stationary, and the action space is $\{left, right, up, down, stay\}$.



Calculate the features for the actions $\in \{left, right, up, stay\}$ from the current state shown in the figure.

$$F_p(s, up) = 1 + 2(1) = 3$$
$$F_p(s, left) = 1 + 2(0) = 1$$
$$F_p(s, right) = 1 + 2(0) = 1$$
$$F_p(s, stay) = 1 + 2(0) = 1$$
$$F_g(s, up) = 2 + 0 + 0 = 2$$
$$F_g(s, left) = 2 + 1 + 1 = 4$$
$$F_g(s, right) = 2 + 1 + 1 = 4$$
$$F_g(s, stay) = 2 + 0 + 2 = 4$$

(b) After a few episodes of Q-learning, the weights are $w_g = -10$ and $w_p = 100$. Calculate the Q value for each action $\in \{left, right, up, stay\}$ from the current state shown in the figure.

$$Q(s, up) = w_pF_p(s, up) + w_gF_g(s, up) = 100(3) + (-10)(2) = 280$$
$$Q(s, left) = w_pF_p(s, left) + w_gF_g(s, left) = 100(1) + (-10)(4) = 60$$
$$Q(s, right) = w_pF_p(s, right) + w_gF_g(s, right) = 100(1) + (-10)(4) = 60$$
$$Q(s, stay) = w_pF_p(s, stay) + w_gF_g(s, stay) = 100(1) + (-10)(4) = 60$$

(c) We observe a transition that starts from the state above, s, takes action *up*, ends in state s' (the state with the food pellet above) and receives a reward $R(s,a,s') = 250$. The available actions from state s' are *down* and *stay*. Assuming a discount of $\gamma = 0.5$, calculate the new estimate of the Q value for s based on this episode.

$$\begin{aligned} Q_{new}(s,a) &= R(s,a,s') + \gamma * \max_{a'} Q(s',a') \\ &= 250 + 0.5 * \max\{Q(s', down), Q(s', stay)\} \\ &= 250 + 0.5 * 0 \\ &= 250 \end{aligned}$$

where

$$\begin{aligned} Q(s', down) &= w_p F_p(s', down) + w_g F_g(s', down) = 100(0) + (-10)(2) = -20 \\ Q(s', stay) &= w_p F_p(s', stay) + w_g F_g(s', stay) = 100(0) + (-10)(0) = 0 \end{aligned}$$

(d) With this new estimate and a learning rate (α) of 0.5, update the weights for each feature.

$$\begin{aligned} w_p &= w_p + \alpha * (Q_{new}(s,a) - Q(s,a)) * F_p(s,a) = 100 + 0.5 * (250 - 280) * 3 = 55 \\ w_g &= w_g + \alpha * (Q_{new}(s,a) - Q(s,a)) * F_g(s,a) = -10 + 0.5 * (250 - 280) * 2 = -40 \end{aligned}$$