CS 188
Fall 2023

Midterm Review RL Solutions

Q1. Q-uagmire

Consider an unknown MDP with three states (A, B and C) and two actions ($\leftarrow$ and $\rightarrow$). Suppose the agent chooses actions according to some policy π in the unknown MDP, collecting a dataset consisting of samples (s, a, s', r) representing taking action a in state s resulting in a transition to state s' and a reward of r.

s	a	s'	r
A	$\rightarrow$	B	2
C	$\leftarrow$	B	2
B	$\rightarrow$	C	-2
A	$\rightarrow$	B	4

You may assume a discount factor of $\gamma = 1$.

(a) Recall the update function of Q-learning is:

$$Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha\left(r_t + \gamma \max_{a'} Q(s_{t+1}, a')\right)$$

Assume that all Q-values are initialized to 0, and use a learning rate of $\alpha = \frac{1}{2}$.

(i) Run Q-learning on the above experience table and fill in the following Q-values:

$Q(A, \rightarrow) =$ $5/2$ $\qquad Q(B, \rightarrow) =$ $-1/2$

$$Q_1(A, \rightarrow) = \frac{1}{2} \cdot Q_0(A, \rightarrow) + \frac{1}{2}\left(2 + \gamma \max_{a'} Q(B, a')\right) = 1$$
$$Q_1(C, \leftarrow) = 1$$
$$Q_1(B, \rightarrow) = \frac{1}{2}(-2+1) = -\frac{1}{2}$$
$$Q_2(A, \rightarrow) = \frac{1}{2} \cdot 1 + \frac{1}{2}\left(4 + \max_{a'} Q_1(B, a')\right)$$
$$= \frac{1}{2} + \frac{1}{2}(4+0) = \frac{5}{2}.$$

(ii) After running Q-learning and producing the above Q-values, you construct a policy π_Q that maximizes the Q-value in a given state:

$$\pi_Q(s) = \arg\max_a Q(s, a).$$

What are the actions chosen by the policy in states A and B?

$\pi_Q(A)$ is equal to:

- ◯ $\pi_Q(A) = \leftarrow$.
- ● $\pi_Q(A) = \rightarrow$.
- ◯ $\pi_Q(A) =$ Undefined.

$\pi_Q(B)$ is equal to:

- ● $\pi_Q(B) = \leftarrow$.
- ◯ $\pi_Q(B) = \rightarrow$.
- ◯ $\pi_Q(B) =$ Undefined.

Note that $Q(B, \leftarrow) = 0 > -\frac{1}{2} = Q(B, \rightarrow)$.

(b) Use the empirical frequency count model-based reinforcement learning method described in lectures to estimate the transition function $\hat{T}(s, a, s')$ and reward function $\hat{R}(s, a, s')$. (Do not use pseudocounts; if a transition is not observed, it has a count of 0.)

Write down the following quantities. You may write N/A for undefined quantities.

$\hat{T}(A, \rightarrow, B) =$ 1 $\hat{R}(A, \rightarrow, B) =$ 3

$\hat{T}(B, \rightarrow, A) =$ 0 $\hat{R}(B, \rightarrow, A) =$ N/A

$\hat{T}(B, \leftarrow, A) =$ N/A $\hat{R}(B, \leftarrow, A) =$ N/A

(c) This question considers properties of reinforcement learning algorithms for *arbitrary* discrete MDPs; you do not need to refer to the MDP considered in the previous parts.

(i) Which of the following methods, at convergence, provide enough information to obtain an optimal policy? (Assume adequate exploration.)

- ■ Model-based learning of $T(s, a, s')$ and $R(s, a, s')$.
- ☐ Direct Evaluation to estimate $V(s)$.
- ☐ Temporal Difference learning to estimate $V(s)$.
- ■ Q-Learning to estimate $Q(s, a)$. Given enough data, model-based learning will get arbitrarily close to the true model of the environment, at which point planning (e.g. value iteration) can be used to find an optimal policy. Q-learning is similarly guaranteed to converge to the optimal Q-values of the optimal policy, at which point the optimal policy can be recovered by $\pi^*(s) = \arg\max_a Q(s, a)$. Direct evaluation and temporal difference learning both only recover a value function $V(s)$, which is insufficient to choose between actions without knowledge of the transition probabilities.

(ii) In the limit of infinite timesteps, under which of the following exploration policies is Q-learning guaranteed to converge to the optimal Q-values for all state? (You may assume the learning rate α is chosen appropriately, and that the MDP is ergodic: i.e., every state is reachable from every other state with non-zero probability.)

- ■ A fixed policy taking actions uniformly at random.
- ☐ A greedy policy.
- ■ An ϵ-greedy policy
- ☐ A fixed optimal policy. For Q-learning to converge, every state-action pair (s, a) must occur infinitely often. A uniform random policy will achieve this in an ergodic MDP. A fixed optimal policy will not take any suboptimal actions and so will not explore enough. Similarly a greedy policy will stop taking actions the current Q-values suggest are suboptimal, and so will never update the Q-values for supposedly suboptimal actions. (This is problematic if, for example, an action most of the time yields no reward but occasionally yields very high reward. After observing no reward a few times, Q-learning with a greedy policy would stop taking that action, never obtaining the high reward needed to update it to its true value.)

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.

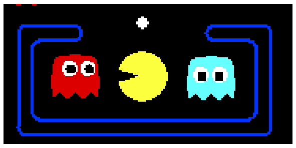
(a) We will have two features, F_g and F_p, defined as follows:

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

where

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

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 $\in \{left, right, up, stay\}$

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

(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

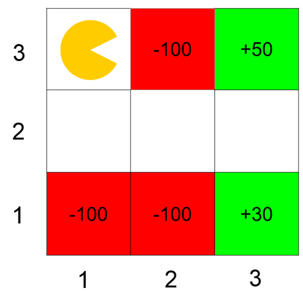
$$\begin{aligned} Q(s', down) &= w_p F_p(s, down) + w_g F_g(s, down) = 100(0) + (-10)(2) = -20 \\ Q(s', stay) &= w_p F_p(s, stay) + w_g F_g(s, stay) = 100(0) + (-10)(0) = 0 \end{aligned}$$

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

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

3 Deep inside Q-learning

Consider the grid-world given below and an agent who is trying to learn the optimal policy. Rewards are only awarded for taking the *Exit* action from one of the shaded states. Taking this action moves the agent to the Done state, and the MDP terminates. Assume $\gamma = 1$ and $\alpha = 0.5$ for all calculations. All equations need to explicitly mention γ and α if necessary.



(a) The agent starts from the top left corner and you are given the following episodes from runs of the agent through this grid-world. Each line in an Episode is a tuple containing (s, a, s', r).

Episode 1	Episode 2	Episode 3	Episode 4	Episode 5
(1,3), S, (1,2), 0	(1,3), S, (1,2), 0	(1,3), S, (1,2), 0	(1,3), S, (1,2), 0	(1,3), S, (1,2), 0
(1,2), E, (2,2), 0	(1,2), E, (2,2), 0	(1,2), E, (2,2), 0	(1,2), E, (2,2), 0	(1,2), E, (2,2), 0
(2,2), E, (3,2), 0	(2,2), S, (2,1), 0	(2,2), E, (3,2), 0	(2,2), E, (3,2), 0	(2,2), E, (3,2), 0
(3,2), N, (3,3), 0	(2,1), Exit, D, -100	(3,2), S, (3,1), 0	(3,2), N, (3,3), 0	(3,2), S, (3,1), 0
(3,3), Exit, D, +50		(3,1), Exit, D, +30	(3,3), Exit, D, +50	(3,1), Exit, D, +30

Fill in the following Q-values obtained from direct evaluation from the samples:

Q((3,2), N) = 50 Q((3,2), S) = 30 Q((2,2), E) = 40

Direct evaluation is just averaging the discounted reward after performing action a in state s.

(b) Q-learning is an online algorithm to learn optimal Q-values in an MDP with unknown rewards and transition function. The update equation is:

$$Q(s_t, a_t) = (1 - \alpha)Q(s_t, a_t) + \alpha(r_t + \gamma \max_{a'} Q(s_{t+1}, a'))$$

where γ is the discount factor, α is the learning rate and the sequence of observations are $(\cdots, s_t, a_t, s_{t+1}, r_t, \cdots)$. Given the episodes in (a), fill in the time at which the following Q values first become non-zero. When updating the Q values, You should only go through each transition once, and the order in which you are to go through them is: transitions in ep 1, transitions in ep 2 and so on. Your answer should be of the form (**episode#,iter#**) where **iter#** is the Q-learning update iteration in that episode. If the specified Q value never becomes non-zero, write *never*.

Q((1,2), E) = *never* Q((2,2), E) = (5,3) Q((3,2), S) = (5,4)

This question was intended to demonstrate the way in which Q-values propagate through the state space. Q-learning is run in the following order - observations in ep 1 then observations in ep 2 and so on.

(c) In Q-learning, we look at a window of (s_t, a_t, s_{t+1}, r_t) to update our Q-values. One can think of using an update rule that uses a larger window to update these values. Give an update rule for $Q(s_t, a_t)$ given the window $(s_t, a_t, r_t, s_{t+1}, a_{t+1}, r_{t+1}, s_{t+2})$.

$Q(s_t, a_t) = (1-\alpha)Q(s_t, a_t) + \alpha(r_t + \gamma r_{t+1} + \gamma^2 \max_{a'} Q(s_{t+2}, a'))$
(Sample of the expected discounted reward using r_{t+1})

$Q(s_t, a_t) = (1-\alpha)Q(s_t, a_t) + \alpha(r_t + \gamma((1-\alpha)Q(s_{t+1}, a_{t+1}) + \alpha(r_{t+1} + \gamma \max_{a'} Q(s_{t+2}, a'))))$
(Nested Q-learning update)

$Q(s_t, a_t) = (1-\alpha)Q(s_t, a_t) + \alpha(r_t + \gamma \max((1-\alpha)Q(s_{t+1}, a_{t+1}) + \alpha(r_{t+1} + \gamma \max_{a'} Q(s_{t+2}, a')), \max_{a'} Q(s_{t+1}, a')))$
(Max of normal Q-learning update and one step look-ahead update)