Reinforcement Learning - Homework 1

Andrii Skliar, 11636785
Gabriele Bani, 11636758

deadline: November 16, 2018

Problem 1 (Instructions)**:**
(empty for correct numeration)

Problem 2 (Introduction)**:**

1. Explain what is meant by the curse of dimensionality.

   > **Solution:** In general, the curse of dimensionality refers to a problem becoming exponentially difficult with increasing number of dimensions. In reinforcement learning, this happens when the state space or action space grow too large, making it difficult to learn and select optimal policies, especially when using bellman updates.

2. Suppose you are trying to design a predator agent that can learn to catch a randomly moving prey on a 5 5 toroidal grid. You have been given the (x, y)-coordinates of the predator and the (x, y)-coordinates of the prey to use as the state.

   (a) How many possible states are there in this naive approach?

   > **Solution:** As there are 5 possible choices for each coordinate, for both agents, and no combination is invalid, there are 5^4 possible states.

   (b) There is a way of reducing the state space considerably a priori. Write down how you would adapt the given state representation to reduce the size of the state space.

   > **Solution:** One way to do this is considering only the difference of coordinates between the two agents, considering the predator as center of the world (0, 0) (to avoid ambiguities).

   (c) How many possible states are there now?

   > **Solution:** There are now only 5^2 possible states

(d) What is the advantage of doing this?

> **Solution:** The advantage is that we retain all the information needed to learn to play the game while heavily reducing the number of states.

(e) Consider the Tic-Tac-Toe example in Chapter 1.5 of the book. Here, too, we can exploit certain properties of the problem to reduce the size of the state space. Give an example of how you could do this.

> **Solution:** We can exploit the invariance of the states over rotation and reflections. We can encode states using a ternary vector of size 9, so that for each of the 9 cells, we use 0 to indicate no sign, 1 for sign of first player, 2 for sign of second player. In this way, we have a unique number representing every possible state. We can use this to map states to a default rotation (or reflection), by generating all rotations and cosider the current state as the one with lowest value (given by the ternary representation). Notice that this has to be done also at runtime, and that for this game all reflections and rotations can be pre-computed.

3. Suppose you want to implement a Reinforcement Learning agent that learns to play Tic-Tac-Toe, as outlined in Chapter 1 of the book.

(a) Which agent do you think would learn a better policy in the end: a greedy agent that always chooses the action it currently believes is best, or a non-greedy agent that sometimes tries new actions? Why

> **Solution:** A non-greedy agent would learn a better policy, given enough time. This is because it has non zero probability of exploring all the possible states, and thus change its beliefs when encountering better possible moves. Instead, a greedy agent may never see states which it believes are not optimal, but that instead would lead to better rewards.

4. Assume we start with an exploration rate of ϵ, meaning that whenever the agent chooses an action, it has a probability of ϵ to pick an action at random, and a probability of 1 - ϵ to pick the greedy action. If we assume the environment has been sufficiently explored, we may want to reduce the amount of exploration after some time.

(a) Write down how you would do this.

> **Solution:** We have two objectives: first, determining when the environment has been sufficiently explored. Second, deciding how to decrease the exploration probability over time.
>
> For the first problem, we assume that we know in advance the maximum reward R_{max} that can be obtained in an episode, so that we have an upper bound to the state-value function. Then, we define a threshold d and a maximum exploration probability a, and keep $\epsilon = a$ if $R_{max} - max_s v(s) > d$. In this way, we can think that when $R_{max} - max_s v(s) \leq d$ we are confident enough about our state-value function, and can start gradually reducing the exploring probability.

To reduce the exploration probability, we can define it as $\epsilon = a * \sigma(R_{max} - max_s v(s) - d/2)$. In this way, we have a function that always gives non zero value to the exploration probability as the sigmoid is always positive, and we reach a maximum value of $0.88a$ when $R_{max} - v(s) = d$.

(b) Does your method work if the opponent changes strategies? Why/why not? If not, provide suggestions on a heuristic that can adapt to changes in the opponents strategy.

Solution: The method proposed would work. To see this, note that if the opponent changes strategy, the state-value function values would change values, and if it becomes low, then the exploration probability would increase. Notice however that our method can be easily fooled if the behavior of the opponent is not too different, which can happen for example if its policy is only slightly modified, so that the average of state-value funciton over states does not change much.

Problem 3 (Exploration)**:**

1. In ϵ-greedy action-selection for the case of n actions, what is the probability of selecting the greedy action?

   Solution: Probability of selecting the greedy action is $1 - \epsilon + \frac{\epsilon}{n}$.

2. Consider a 3-armed bandit problem with actions 1, 2, 3. If we use ϵ-greedy action-selection, initialization at 0, and sample-average action-value estimates, which of the following sequence of actions are certain to be the result of exploration? $A_1 = 1, R_1 = -1, A_2 = 2, R_2 = 1, A_3 = 2, R_3 = -2, A_4 = 2, R_4 = 2, A_5 = 3, R_5 = 1$.

   Solution: Actions A_4, A_5 are results of exploration. This can be seen by calculating average action-value estimates at each timestep and seeing when the action taken is not the one that yields best estimates.

3. You are trying to find the optimal policy for a two-armed bandit. You try two approaches: in the pessimistic approach, you initialize all action-values at -5, and in the optimistic approach you initialize all action-values at $+5$. One arm gives a reward of $+1$, one arm gives a reward of -1. Using a greedy policy to choose actions, compute the resulting Q-values for both actions after three interactions with the environment. In case of a tie between two Q-values, break the tie at random.

   Solution: We assume that arm A_1 gives reward of $+1$ and arm A_2 gives reward of -1.

Step	$Q_1(+1)$	$Q_2(-1)$	Action Taken
0	-5	-5	All action values are the same, select arm randomly. Assume, Q_1 has been selected.
1	1	-5	Greedily choose Q_1
2	1	-5	Greedily choose Q_1
3	1	-5	

Step	$Q_1(+1)$	$Q_2(-1)$	Action Taken
0	5	5	All action values are the same, select arm randomly. Assume, Q_1 has been selected.
1	1	5	Greedily choose Q_2
2	1	-1	Greedily choose Q_1
3	1	-1	

4. Which initialization leads to a higher (undiscounted) return? What if you had broken the tie differently?

Solution: Pessimistic initialization leads to higher return because in this case it will lead to pure exploitation without any exploration. However, if we would break tie choosing Q_2 in the first step, we would never choose Q_1 thus getting only negative rewards. Optimistic initialization, on the other hand, will lead to the same return even with any break in the first step. Asymptotically, its performance should also be better due to the fact described in the next part of the question.

5. Which initialization leads to a better estimation of the Q-values?

Solution: Optimistic initialization leads to a better estimation of the Q-values as, unlike pessimistic initialization, it allows for exploration and not just pure exploitation strategy. Pessimistic initialization asymptotically can only lead to a good estimation of a Q-value of a single hand (the one chosen first), while optimistic initialization can asymptotically lead to a good estimation of all Q-values.

6. Explain why one of the two initialization methods is better for exploration.

Solution: Optimistic initialization is better for exploration due to the fact that it makes initial Q-value larger than any possible action-value, so after taking any action, greedy policy forces us to use other, not yet explored actions as their action-value is higher. At the same time, pessimistic initialization will choose the same action at all timesteps. It is due to the fact that all the initial action-values are smaller than then action-value of an action chosen at the first timestep and will enforce that only this action will be chosen at any future timestep.

Problem 4 (Markov Decision processes)**:**

1. (a) For the first four examples outlined in Section 1.2 of the book, describe the state space, action space and reward signal.

Solution:

i. **Chess**

- **State space** It contains all the possible chess board configurations that are valid. We can encode every piece with a tuple in which the first element is a letter $\{B, W\}$ for indicating whether the owner of the piece is black or white, and the second element is a number encoding the type of piece. We can then have a 8×8 matrix whose elements are tuples (or an empty placeholder) to represent the chessboard. Finally, we must also keep a variable to identify whose player is currently playing.
- **Action space** It is composed of all legal actions that the current player can make.
- **Reward signal** 1 if the player wins the game, 0 if the opponent wins.

ii. **Adaptive controller**

- **State space** The set of marginal costs and the parameters.
- **Action space** The actions are setting specific values for any subset of the parameters of the refinery.
- **Reward signal** The yield/cost/quality value

iii. **Gazelle**

- **State space** As this is a real world example, we need to discretize the world in small timesteps. Furthermore, we have to decide to which abstraction level consider. We decide to consider a high level abstraction, with states
- **Action space** The action space is { walking, running, standing, eating, drinking, sleeping}.
- **Reward signal** The reward can be seen as +1 for each small discrete timestep in which the gazelle is alive.

iv. **Robot**

- **State space** The state space could be a composed of the coordinates of the robot and the recharging station.
- **Action space** At each timestep, the robot can switch between two behaviors H and C. While in H, the robot tries to head towards the recharge station. While in C, the robot continues searching for trash to collect. The actual movement done is calculated through a navigation algorithm based on the current behavior.
- **Reward signal** The reward can be 1 whenever trash is found, and $-C$ when the robot is discharged, with C being a high positive number (such as 100). Furthermore, to further help the robot learning not to discharge completely, we could assign a small negative reward for every timestep where the battery level is below a certain threshold.

(b) Come up with an example of your own that you might model with an MDP. State the action space, state space and reward signal.

Solution: We have a movie website, and want to recommend the next movie to watch to users.

- **State space** The list of all previously watched movies of the user.
- **Action space** a value between 1 and N, which indicates the index of the next movie to watch
- **Reward signal** 1 if the user watches the suggested movie as next, 0 if the next movie is different from the suggested one.

(c) Come up with an example for a problem that you might have trouble solving with an MDP. Why doesnt this fit the framework?

Solution:
Life is generally non-MDP. Even if we do assume that we have almost zero discretization step and full memory of all the previous event, we can't actually know all possible states. The reason for that are so-called black swan events. This metaphor describes events, which were not expected to happen and were unprecedented yet were considered to be bound to happen after being analyzed by domain expert (also due to hindsight bias). Due to the fact that we don't know about the existence of those states and don't know about our absence of knowledge about those, we can't consider life a Markov Decision Process as we don't have finite state of actions anymore.

(d) Consider the example in exercise 3.3 of the book. Why might you choose to view the actions as handling the accelerator, brake and steering wheel? What is the disadvantage of doing this?

Solution: We can choose these as they allow the driver to fully control the behavior of the car. The disadvantage of this is that planning can be hard as we are controlling low level actions.

(e) Why might you choose to view the actions as choosing where to drive? What is the disadvantage of doing this?

Solution: We can choose to do this as the actions are directly related to the goal, and planning can become much easier. The disadvantage is that we cannot model anymore low level actions such as steering, breaking and turning the wheel, and we assume we have a system that can do that according to the selected action. However, this assumption is often hard to meet in practice.

(f) Can you think of some way to combine both approaches?

Solution: To combine both approaches, we can use a hierarchical method. We would have different two policies, a high level one witch decides the destination, and a lower level one which decides which low level actions to take based on the action chosen by the first policy. This has the advantage of decoupling the two types of actions, and allowing the ruse of information learned by the low level policy.

2. (a) Eq. 3.8 in the book gives the discounted return for the continuous case. Write down the formula for the discounted return in the episodic case.

Solution:

$$G_t = R_{t+1} + _{t+2} + \gamma^2 R_{t+3} .. + \gamma^T R_T = \sum_{k=t}^{T} \gamma^k R_{t+1+k}$$

(b) Show that $\sum_{k=0}^{\infty} \gamma^k = \frac{1}{1-\gamma}$ if $\gamma < 1$.

Solution: We first work out the expression for finite summation of n terms

$$\sum_{k=0}^{n-1} \gamma^k = \gamma^k = 1 + \gamma + \gamma^2 + ...\gamma^{n-1}$$
$$\gamma \sum_{k=0}^{n-1} \gamma^k = \gamma + \gamma^2 + ...\gamma^n$$
$$\gamma \sum_{k=0}^{n-1} \gamma^k = -1 + 1 + \gamma + \gamma^2 + ...\gamma^n$$
$$\gamma \sum_{k=0}^{n-1} \gamma^k = \gamma^n - 1 + \sum_{k=0}^{n-1} \gamma^k$$
$$(\gamma - 1) \sum_{k=0}^{n-1} \gamma^k = \gamma^n - 1$$
$$\sum_{k=0}^{n-1} \gamma^k = \frac{\gamma^n - 1}{\gamma - 1} = \frac{1 - \gamma^n}{1 - \gamma}$$

Thus, if we consider the limit with $n \to \infty$, the term $\gamma^n \to 0$ when $\gamma < 1$, so the equality $\sum_{k=0}^{\infty} \gamma^k = \frac{1}{1-\gamma}$ holds.

(c) Consider exercise 3.7 in the book. Why is there no improvement in the agent?

Solution: There is no improvement in the agent because every time it exits from the maze it gets a reward of +1, regardless of the number of steps taken to reach the end. This is a problem when using approaches based on state-value functions or action-value function, as all states or state/action pairs would have the same value and this would give no useful information on how to solve the problem.

(d) How would adding a discount factor of γ ¡ 1 help solve this problem?

Solution: Adding a discount factor of $\gamma < 1$ would make the rewards higher when the agent finds the exit of the maze quickly, and make the reward lower when the agent takes a long time to find the exit. So, for example, value function approaches would now be able to distinguish which states (or actions) are better, which translates in this case in lead to the exit in a lower number of time steps.

(e) How might changing the reward function help solve this problem?

Solution: We can change the rewards so that the agent gets reward based on the time steps needed to exit the maze. One way to do this would be changing the reward for all non terminal transitions to a small $\epsilon < 0$.

Problem 5 (Dynamic Programming)**:**

1. Write the value, $\nu^\pi(s)$, of a state s under policy π, in terms of π and $q^\pi(s,a)$. Write down both the stochastic and the deterministic policy case.

Solution:

$$\begin{aligned}\nu^\pi(s) &= \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma \nu^\pi(s')] \\ &= \sum_a \pi(a|s) q^\pi(s,a) && \text{(stochastic case)} \\ &= q^\pi(s,a') && \text{(deterministic case with policy } \pi(s) = a')\end{aligned}$$

2. In Policy Iteration, we first evaluate the policy by computing its value function, and then update it using a Policy Improvement step. You will now change Policy Iteration as given on page 80 of the book to compute action-values. First give the new policy evaluation update in terms of $Q^\pi(s,a)$ instead of $V^\pi(s)$. Note that Policy Improvement uses deterministic policies.

Solution: Note, that we are using deterministic policy $\pi(s)$.
Policy Evaluation
Loop:
    $\Delta \leftarrow 0$
    Loop for each $s \in \mathcal{S}$:
        Loop for each $a \in \mathcal{A}(s)$:
            $q \leftarrow Q(s,a)$
            $Q(s,a) \leftarrow \sum_{s',r} p(s',r|s,a)[r + \gamma Q^\pi(s', \pi(s'))]$
            $\Delta \leftarrow max(\Delta, |q - Q(s,a)|)$
until $\Delta < \theta$

3. Now change the Policy Improvement update in terms of $Q^\pi(s,a)$ instead of $V^\pi(s)$.

Solution:
Policy Improvement
policy-stable $\leftarrow$ *true*
For each $s \in \mathcal{S}$:
    old-action $\leftarrow \pi(s)$
    $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s',r|s,a)[r + \gamma Q^\pi(s', \pi(s'))]$
    If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow false$
If *policy-stable*, then stop and return $Q \approx q_*$ and $\pi \approx \pi_*$; else go to Policy Evaluation

4. The Value Iteration update, given in the book in Eq. 4.10 can also be rewritten in terms of Q-values. Give the Q-value Iteration update.

Solution:

$$q(s,a) = \sum_{s',r} p(s',r|s,a)[r + \gamma \max_{a'} q(s',a')]$$

Problem 6 (Monte Carlo)**:**

1. Consider an MDP with a single state s 0 that has a certain probability of transitioning back onto itself with a reward of 0, and will otherwise terminate with a reward of 5. Your agent has interacted with the environment and has gotten the following three trajectories: [0, 0, 5], [0, 0, 0, 0, 5], [0, 0, 0, 5]. Use $\gamma = 0.9$.

   Solution:

   (a) Following the pseudocode in section 5.1, we get the following result. For the first episode, the return will be added only at the last iteration of the inner loop, as s_0 is the only state present, and the value appended to Returns will be $\gamma^2 5$. For the second and third episode, the same holds, and the values appendded to Returns will be $\gamma^4 5$ and $\gamma^3 5$ respectively. Thus, the value of state s_0 will be $\frac{0.9^2*5+0.9^4*5+0.9^3*5}{3} = 3.6585$

   (b) This time, we do not check whether we have previously visited the state during the current run, so the return for the first episode will be $\frac{1}{2}\sum_{i=0}^{i=2} G^{2-i} = 0.9^2 * 5 + 0.9 * 5 + 5$. For the second and third episodes, the values appended to returns will be $\frac{1}{4}\sum_{i=0}^{i=4} G^{4-i}$ and $\frac{1}{3}\sum_{i=0}^{i=3} G^{3-i}$ respectively.

   Thus, the value of state s_0 will be 4.304

2. What is a disadvantage of using ordinary importance sampling in off-policy Monte Carlo?

   Solution: One disadvantage is that the variance of ordinary importance sampling can be unbounded, as it is determined by the ratio between the target probability and the behavior policy, which can assume any positive (possibly large or infinite) value.

3. What is a disadvantage of using weighted importance sampling in off-policy Monte Carlo?

   Solution: One disadvantage is that for both first-visit and every-visit approaches, weighted importance sampling produces a biased estimator of the desired value function, and the only theoretical guarantee that the bias goes to zero is that the number of samples tends to infinity.

Problem 7 (Temporal Difference Learning (Application))**:**

1. What are the state(-action) value estimates $V(s)$ (or $Q(s,a)$) after observing the sample episode when applying:

   (a) TD(0)

   Solution:

   $$V(A) = -0.893$$
   $$V(B) = 0.433$$

   (b) 3-step TD

   Solution:

   $$V(A) = -0.983$$
   $$V(B) = -0.17$$

   (c) SARSA

   Solution:

   $$Q(A,1) = -0.57$$
   $$Q(A,2) = -0.43$$
   $$Q(B,1) = 0.4$$
   $$Q(B,2) = 0.1$$

   (d) Q-learning

   Solution:

   $$Q(A,1) = -0.53$$
   $$Q(A,2) = -0.4$$
   $$Q(B,1) = 0.4$$
   $$Q(B,2) = 0.1$$

2. Choose a deterministic policy that you think is better than the random policy given the data. Refer to any of the state(-action) value estimates to explain your reasoning.

Solution: Using the Q-values, learned in Q-learning algorithm (SARSA produces similar results, so we can use those as well), we can change our policy to be deterministic :

heyyy the first one could be also 1, depends on how you argue

$$\pi(S = A) = 2$$
$$\pi(S = B) = 1$$

As we can see, such policy gives rise to the highest state-action value among all the possible options.

3. Let π_{random} denote the random policy used so far and $\pi_{student}$ denote the new policy you proposed. Suppose you can draw new sample episodes indefinitely until convergence of the value estimates.

   (a) Discuss how do you expect the final value estimates to differ if you ran Q-Learning with π_{random} as compared to $\pi_{student}$.

   Solution: The main reason to use Q-learning is that it is an off-policy algorithm, which means that it uses one policy for exploring and the other one for exploiting the information provided by the exploration to update Q-values. The main consequence of that is that Q-learning convergence can only be proven for the case when **all** the state-action pairs are continuously updated, which is not the case if we use $\pi_{student}$, which is a greedy policy. Therefore, I would expect that π_{random} would converge to real state-action values, while $\pi_{student}$ would not be able to converge to real values as most of the action-state pairs would never be explored enough to estimate anything meaningful. However, if we are interested in a short-term estimation, $\pi_{student}$ can potentially produce good estimation of the pairs $(A, 2)$ and $(B, 1)$ in a short run as they will be visited more often than if we would use π_{random}. However, this statement highly depends on the transition probabilities of the MDP.

   (b) What problems may arise with π_{random} or $\pi_{student}$ respectively?

   Solution: The main problem of $\pi_{student}$ has been discussed in the previous subquestion - as it is a greedy policy, it doesn't contribute to the exploration of the state-action values, which is a necessity for a good performance of Q-learning in a long run.

   However, in a short term, it might not explore states, which potentially bring the highest expected reward, which is not the case for $\pi_{student}$, which will only explore those. $\pi_{student}$ though can only provide the forementioned benefit in case the initial estimations were representative and system is stationary as in this case we can already expect the final state-action values not to change much compared to the estimation provided in the previous part of the question.

   (c) Do you think using an ϵ-greedy policy as behavior policy would be benecial? Explain why/why not?

Solution: Using ϵ-greedy policy for the case of $\pi_{student}$ would be highly beneficial as it would eliminate the problem of not sufficient exploration that arises when using previously suggested policy. However, using it for the π_{random} depends on the current values of the policy. Assuming that they are uniform (which is the most extreme case of randomness), it would be beneficial as it would allow us not only to explore all the states equally but also to explore states which are known to be good more. However, the closer π_{random} is to epsilon-greedy policy, the less benefit changing the policy will give. It's important to notice, that ϵ-greedy policy, however, is only useful when we haven't converged yet. If we have converged, exploration is not needed anymore and we can simply use greedy policy for exploitation.

Problem 8 (Temporal Difference Learning (Theory))**:**

1. We can use Monte Carlo to get value estimates of a state with

Solution:

$$V_M(S) = V_{M-1}(S) + \alpha_M[G_M(S) - V_{M-1}(S)]$$
$$V_M(S) = V_{M-1}(S) + \alpha_M G_M(S) - \alpha_M V_{M-1}(S)$$
$$V_M(S) = (1-\alpha_M)V_M(S) + \alpha_M G_t$$
$$\frac{1}{M}\sum_{n=1}^{M} G_n(S) = (1-\alpha)\frac{1}{M-1}\sum_{n=1}^{M-1} G_n(S) + \alpha_M G_t$$
$$\frac{1}{M}\sum_{n=1}^{M-1} G_M(S) + \frac{1}{M}G_M(S) = (1-\alpha)\frac{1}{M-1}\sum_{n=1}^{M-1} G_n(S) + \alpha_M G_M(S)$$

In order for the right side to be the same as the left side, we need $\alpha_M = \frac{1}{M}$. This is the obvious choice for equating the $G_M(S)$ terms, and we can see that, for the summation on the right hand side, $(1-\alpha_M)\frac{1}{M-1} = \frac{M-1}{M(M-1)} = \frac{1}{M}$, so the summation terms both have the same coefficient, and the equality holds.

2. Consider the TD-error

   (a) What is $\mathbb{E}[\delta t | S_t = s]$ if δt uses the true state-value function V^π.

Solution:
For clarity, we first write down the bellman equations for state-value function V under the policy π

$$
\begin{aligned}
V_\pi(s) &= \mathop{\mathbb{E}}_{\pi}[G_t|S_t = s] \\
&= \mathop{\mathbb{E}}_{\pi}[R_{t+1} + \gamma G_{t+1}|S_t = s] \\
&= \mathop{\mathbb{E}}_{\pi}[R_{t+1} + \gamma V_\pi(S_{t+1})|S_t = s]
\end{aligned}
$$

We can now write the TD error, using the bellman equations, find the result

$$
\begin{aligned}
\mathbb{E}[\delta_t|S_t = s] &= \mathop{\mathbb{E}}_{\pi}[R_{t+1} + \gamma V^\pi(S_{t+1}) - V^\pi(S_t)|S_t = s] \\
&= \mathop{\mathbb{E}}_{\pi}[R_{t+1} + \gamma V^\pi(S_{t+1})|S_t = s] - V^\pi(S_t) \\
&= V^\pi(S_t) - V^\pi(S_t)
\end{aligned}
$$

(b) What is $\mathbb{E}[\delta t|S_t = s, A_t = a]$ if δt uses the true state-value function V^π

Solution:
For action-value function, we know that from bellman equation

$$
\begin{aligned}
q_\pi(s, a) &= \mathop{\mathbb{E}}_{\pi}[G_{t_{+1}}|S_t = s, A_t = a] \\
&= \mathop{\mathbb{E}}_{\pi}[R_{t+1} + \gamma G_{t+1}|S_t = s, A_t = a] \\
&= \mathop{\mathbb{E}}_{\pi}[R_{t+1} + \gamma V_\pi(S_{t+1})|S_t = s, A_t = a]
\end{aligned}
$$

We have

$$
\begin{aligned}
\mathbb{E}[\delta_t|S_t = s, A_t = a] &= \mathop{\mathbb{E}}_{\pi}[R_{t+1} + \gamma V^\pi(S_{t+1}) - V^\pi(S_t)|S_t = s, A_t = a] \\
&= \mathop{\mathbb{E}}_{\pi}[R_{t+1} + \gamma V^\pi(S_{t+1})|S_t = s, A_t = a] - V^\pi(S_t) \\
&= Q^\pi(S_t, A_t) - V^\pi(S_t)
\end{aligned}
$$

The Monte-Carlo error can be written as the sum of TD-errors if the value estimates dont change (cf. equation 6.6 in the book) as

(a) If V changes during the episode, then this equation only holds approximately, what would the difference be between the two sides? Let V_t denote the array of state values used at time t in the TD errorδ_t and in the TD update $V_{t+1}(S_t) = V_t(S_t) + \alpha\delta_t$. Redo the derivation above to determine the additional amount that must be added to the sum of TD errors in order to equal the Monte Carlo error.

Solution: First, we can notice that in case $S_{t+1} \neq S_T$, then the value of $V_t(S_t)$ does not get updated, so we have $V_{t+1}(S_{t+1}) = V_t(S_{t+1}$. Instead, when $S_{t+1} = S_t$, $V_{t+1}(S_{t+1}) = V_t(S_{t+1} + \alpha\delta_t$ also includes the delta term. We can write this concisely as $V_{t+1}(S_{t+1}) = V_t(S_{t+1}) + w_t$, where $w_t = \begin{cases} \alpha, & \text{if } S_{t+1} = S_t \\ 0, & \text{if } S_{t+1} \neq S_t \end{cases}$.

$$\begin{aligned}
G_t - V(S_t) &= R_{t+1} + \gamma G_{t+1} - V_t(S_t) + \gamma V_{t+1}(S_{t+1}) - \gamma V_{t+1}(S_{t+1}) \\
&= R_{t+1} - V_t(S_t) + \gamma V_{t+1}(S_{t+1}) + \gamma(G_{t+1} - V_{t+1}(S_{t+1}) \\
&= R_{t+1} - V_t(S_t) + \gamma V_t(S_{t+1}) + \gamma\delta_t w_t + \gamma(G_{t+1} - V_{t+1}(S_{t+1}) \\
&= \delta_t + \gamma\delta_t w_t + \gamma(G_{t+1} - \gamma V_t(S_{t+1})) \\
&= \delta_t + \gamma\delta_t w_t + \gamma(\delta_{t+1} + \gamma\alpha\delta_{t+1} + \gamma(G_{t+2} - \gamma V(S_{t+2})) \\
&= \sum_{k=t}^{T-1} \gamma^{k-t}\delta_k + \gamma^{k-t+1}\delta_t w_t \\
&= \sum_{k=t}^{T-1} \gamma^{k-t}\delta_k(1 + \gamma w_t)
\end{aligned}$$

Which means that the term added to the monte carlo error, which is $\sum_{k=t}^{T-1} \gamma^{k-t}\delta_k$, can be written as $\sum_{k=t}^{T-1} \gamma^{k-t+1}\delta_k\gamma w_t$

(b) Show that the n-step error as used in can also be written as a sum TD errors (assuming the value estimates dont change)

Solution:

$$\begin{aligned}
G_{t:t+n} - V_{t+n-1}(S_t) &= R_{t+1} + \gamma G_{t+1:t+n} - V_{t+n-1}(S_t) + \gamma V_{t+n-1}(S_{t+1}) - \gamma V_{t+n-1}(S_{t+1}) \\
&= R_{t+1} - V_{t+n-1}(S_t) + \gamma V_{t+n-1}(S_{t+1}) + \gamma(G_{t+1:t+n} - V_{t+n-1}(S_{t+1})) \\
&= \delta_t + \gamma(G_{t+1:t+n} - \gamma V_{t+n-1}(S_{t+1})) \\
&= \delta_t + \gamma(\delta_{t+1} + \gamma(G_{t+2:t+n} - \gamma V_{t+n-1}(S_{t+2}))) \\
&= \delta_t + \gamma(\delta_{t+1} + \gamma(\delta_{t+2} + \gamma(... + \gamma(\delta_{t+n-1} + \gamma(V_{t+n-1}(S_{t+n)} - V_{t+n-1}(S_{t+n}))..) \\
&= \delta_t + \gamma\delta_{t+1} + \gamma^2\delta_{t+2} + ... + \gamma^{n-1}\delta_{n-1} + 0 \\
&= \sum_{k=0}^{n-1} \gamma^t\delta_{t+k}
\end{aligned}$$

Problem 9 (Maximization Bias)**:** Consider the undiscounted MDP in Figure 1.

1. We repeatedly apply Q-learning and SARSA on the observed data until convergence. Give all nal state-action values for Q-learning and SARSA respectively.

Solution: First, we need to specify notation for actions. We use following notation:

- Action taking from state A to state T is denoted as a_{AT}
- Action taking from state A to state B is denoted as a_{AB}
- Actions taking from state B to state T are denoted (from left to right) as $\{a^1_{BT}, a^2_{BT}, a^3_{BT}, a^4_{BT}\}$

Then, we get following state-action values for Q-Learning:

$$
\begin{aligned}
Q(A, a_{AT}) &= 1.5 \\
Q(A, a_{AB}) &= 2 \\
Q(A, a^1_{BT}) &= 1 \\
Q(A, a^1_{BT}) &= 1 \\
Q(A, a^2_{BT}) &= 2 \\
Q(A, a^3_{BT}) &= 0
\end{aligned}
$$

And following values for SARSA:

$$
\begin{aligned}
Q(A, a_{AT}) &= 1.5 \\
Q(A, a_{AB}) &= 1 \\
Q(A, a^1_{BT}) &= 1 \\
Q(A, a^1_{BT}) &= 1 \\
Q(A, a^2_{BT}) &= 2 \\
Q(A, a^3_{BT}) &= 0
\end{aligned}
$$

2. This problem suffers from maximization bias. Explain where this can be observed. Do both Q-learning and SARSA suffer from this bias? Why/why not?

Solution: It can be observed if we look at the state-action values for Q-learning. We know that the real state-action value for $Q(A, a_{AB}) = 1$ as it is an average of all possible state-action values for B. However, estimated $Q(A, a_{AB}) = 2$. Therefore, even though it is more optimal to perform action a_{AT} from the initial state, model will use action a_{AB} as it is has larger Q-value. This appears due to the fact that the state-action value of each state is the maximum over the state-action values of the next observed state. Because SARSA doesn't use the maximum and simply calculates the average of estimated state-action values of the neighboring states, it doesn't suffer from this bias, while Q-learning does.

3. To circumvent the issue of maximization bias, we can apply Double Q-learning. Use the

given example to explain how Double Q-learning alleviates the problem of maximization bias.

Solution: In this case, instead of defining $Q(A, a^2_{BT})$ to be the maximum over the state-action values of $\{a^1_{BT}, a^2_{BT}, a^3_{BT}, a^4_{BT}\}$, we will use another $Q'(A, a^2_{BT})$ to estimate the real action-state value for $Q(A, a^2_{BT})$. Randomly, based on a coin flip, we would swap two policies and use one to estimate the other or vice versa. Basically, this way we eliminate the fact that we blindly use the maximum state-action value and actually check that it is correct by comparing to another policy. This allows us to make estimated state-action value unbiased.

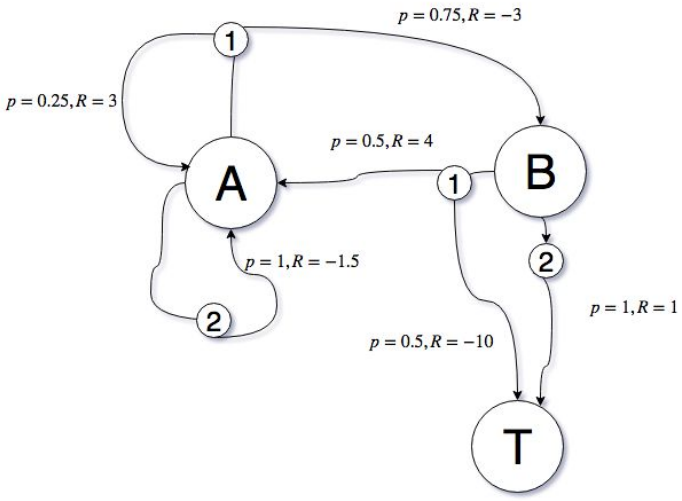
4. What are the true state-action values that we would expect to get (after convergence) if we continued sampling episodes.

Solution:

$$Q(A, a_{AT}) = 1.5$$
$$Q(A, a_{AB}) = 1$$
$$Q(A, a^1_{BT}) = 1$$
$$Q(A, a^1_{BT}) = 1$$
$$Q(A, a^2_{BT}) = 1$$
$$Q(A, a^3_{BT}) = 1$$

Problem 10 (Model-based RL)**:**

1. Draw a diagram of the MDP that best explains the observed episodes (i.e. the model that maximizes the likelihood of the data). Show rewards and transition probabilities on your diagram. Rewards can be expressed as point estimates. Transitions can be probabilistic.



Solution: [b]

2. Consider the Dyna-Q algorithm as described in the book in Section 8.2.

Solution:

(a) In this MDP, the transitions between states are stochastic, while Dyna-Q assumes them to be deterministic.

(b) To make Dyna-Q work with stochastic state transitions, we would have to modify line (e) in the pseudo code 8.2 in the book. If we kept it, every time we see a different state transition we would overwrite the model.

To solve this, we could keep a dictionary of possible next states for every state, with associated transition probabilities (based on the number of visits) and rewards.

(c) It would deal with changing environment, but not well. In particular, it would be slow to adapt to the new environment when there are many observations relative to the past environment. Also, as the environment changes multiple times, the memory of the past environment behaviors would still be present and interfere with the new observations.

To solve this, we can select a number N, which will be the maximum size of the memory for each action transition. So, we would only average the N most recent observations, and be certain to adapt to the new environments in up to N episodes.

3. Suppose we have two agents that we initialize equally. The first agent uses the Dyna-Q algorithm with infinite planning steps (i.e. after observing one episode the agent uses its planning module until the Q-values converge). The second agent uses Q-learning, but after each episode it repeatedly applies its update rule on the sampled episode until the Q-values converge. Afterwards the episode data is discarded and a new episode is sampled. If we give both agent the same experience sampled from a random policy, do we expect that their Q-values differ after

Solution:

(a) After the first episode, the Q-values will be the same. This is because both algorithms will update the Q-values regarding the exact same states until convergence. In both cases, the updates are performed using Q-learning update rule, which is guaranteed to converge to the optimal policy if all state-action pairs are visited long enough.

(b) After the second episode, Q-values will be in general different if the second episode is different from the first one. Q-learning in this case would only refine Q-values for the new episodes, while Dyna-Q would also update values for transitions of the previous episode.

(c) Dyna-q would converge, as would Q-learning, as all transitions would be visited infinitely many times, making the Q-function converge as the conditions for convergence are met.

Problem 11 (Bonus Exercise: Contraction Mapping)**:**

1. a Consider the contraction mapping $T(x) := 1 + \frac{1}{3}x$. Find the fixed point of this mapping.

Solution:

$$
\begin{aligned}
T(x) &= x \\
1 + \frac{1}{3}x &= x \\
3 + x &= 3x \\
2x &= 3 \\
x &= \frac{3}{2}
\end{aligned}
$$

b Consider the contraction mapping $(Tf)(s) := -\frac{1}{2}f(s) + g(s)$. Find the fixed point of this mapping in terms of $g(s)$.

Solution:

$$
\begin{aligned}
&(Tf)(s) = f(s) \\
&-\frac{1}{2}f(x) + g(s) = f(s) \\
&-f(s) + 2g(s) = 2f(s) \\
&3f(s) = 2g(s) \\
&s = f^{-1}(\frac{2}{3}g(s))
\end{aligned}
$$

2. (a) Show that $|max_z f(z) - max_z g(z)| \leq max_z |f(z) - g(z)|$ for arbitrary $f(z)$ and $g(z)$

Solution:
First, we write the absolute values as max operations

$$|max_z f(z) - max_z g(z)| = max\{max_z f(z) - max_z g(z), max_z g(z) - max_z f(z)\} \tag{1}$$

$$max_z |f(z) - g(z)| = max\{max_z [f(z) - g(z)], max_z [g(z) - f(z)]\} \tag{2}$$

Now, by calling $z_f := argmax_z(f(z))$, and $z_g := argmax_z(g(z))$, we can rewrite the first equation as

$$|max_z f(z) - max_z g(z)| = max\{f(z_f) - max_z g(z), g(z_g) - max_z f(z)\} \tag{3}$$

As both terms of equation 2 have a max of differences evaluated at the same z for both functions, we evaluate the function at corresponding values, and obtain

$$
\begin{aligned}
|max_z f(z) - max_z g(z)| &= max\{f(z_f) - max_z g(z), g(z_g) - max_z f(z)\} && (4)\\
&\leq max\{f(z_f) - g(z_f), g(z_g) - f(z_g)\} && (5)\\
&\leq max\{max_z [f(z) - g(z)], max_z [g(z) - f(z)]\} && (6)\\
&= max_z |f(z) - g(z)| && (7)
\end{aligned}
$$

(b) Show that B^8 is a contraction mapping in supremum norm, i.e.

$$||(B^* V_1)(s) - (B^* V_2)(s)||_\infty \leq c||[V_1(s) - V_2(s)]||_\infty$$

Solution:
As the bellman operators inside the norm both include a max operation, we can use the above proven inequality. We have

$$||(B^*V_1)(s) - (B^*V_2)(s)||_\infty \tag{8}$$
$$= ||max_a\big[R(s,a) + \gamma \sum_{s' \in \mathcal{S}} Pr(s'|s,a)V_1(s')\big] \tag{9}$$
$$- max_a\big[R(s,a) + \gamma \sum_{s' \in \mathcal{S}} Pr(s'|s,a)V_2(s')\big]||_\infty \tag{10}$$
$$\leq ||max_a\big[R(s,a) + \gamma \sum_{s' \in \mathcal{S}} Pr(s'|s,a)V_1(s') - R(s,a) + \gamma \sum_{s' \in \mathcal{S}} Pr(s'|s,a)V_2(s')\big]||_\infty \tag{11}$$
$$= ||max_a \gamma \sum_{s' \in \mathcal{S}} Pr(s'|s,a)(V_1(s') - V_2(s'))||_\infty \tag{12}$$
$$= max_s max_a \gamma \sum_{s' \in \mathcal{S}} Pr(s'|s,a)(V_1(s') - V_2(s')) \tag{13}$$
$$\leq max_s max_a \gamma max_{s'}(V_1(s') - V_2(s')) \tag{14}$$
$$= \gamma max_{s'}(V_1(s') - V_2(s')) \tag{15}$$
$$\leq \gamma max_{s'}|V_1(s') - V_2(s')| \tag{16}$$
$$= \gamma|||V_1(s') - V_2(s')|||_\infty \tag{17}$$

Where for the first inequality we have used the above proven inequality. For the second inequality we have used the fact that the maximum over s' is always greater or equal than the expected value, and for the third one we have used the fact that the difference of two values is always less than or equal to the absolute difference of the same values. Another important point is the removal of the max operations on s and a from equation 14 to 15, which can be done as there is no dependence on s or a in the function.

Note that we also found that $c = \gamma$, which means that for B to be a contraction mapping, we need $0 \leq \gamma < 1$.

References