# Exercise Set 2 - Reinforcement Learning
# Chapter 3,4 - Tabular Methods

## Instructions

This is the second exercise booklet for Reinforcement Learning. It covers both ungraded exercises to practice at home or during the tutorial sessions as well as graded homework exercises and graded coding assignments. The graded assignments are clearly marked.

- Make sure you deliver answers in a clear and structured format. LaTeXhas our preference. Messy handwritten answers will not be graded.
- Pre-pend the name of your TA to the file name you hand in and remember to put your name and student ID on the submission;
- The deadline for this first assignment is **September 16th 2020 at 17:00** and will cover the material of chapter 3-4. All questions marked 'Homework' in this booklet need to be handed in on Canvas. The coding assignments need to be handed in separately through the Codegra.de platform integrated on canvas.

## Contents

**3 Advanced MC and TD methods** **2**
3.1 Homework: Coding Assignment - Monte Carlo . . . . . . . . . . . . . . . . . . . . 2
3.2 Importance Sampling in Monte Carlo methods . . . . . . . . . . . . . . . . . . . . 2
3.3 Temporal Difference Learning (Theory) . . . . . . . . . . . . . . . . . . . . . . . . 2
3.4 * Exam question: Monte Carlo for control . . . . . . . . . . . . . . . . . . . . . . 2

**4 Advanced TD Methods** **4**
4.1 Off-policy TD . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
4.2 Temporal Difference Learning (Application) . . . . . . . . . . . . . . . . . . . . . 4
4.3 Homework: Coding assignment - Temporal Difference Learning . . . . . . . . . . 5
4.4 Homework: Maximization Bias . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
4.5 Contraction Mappings & Convergence of Bellman Optimality Operator . . . . . . 6
4.6 N-step Temporal Difference Learning (Theory) . . . . . . . . . . . . . . . . . . . . 7
4.7 *Exam question - N-Step Temporal Difference Learning . . . . . . . . . . . . . . 8

# Chapter 3: Advanced MC and TD methods

## 3.1 Homework: Coding Assignment - Monte Carlo

1. In the chapter 5.6 of the book, we are given incremental update rule for weighted importance sampling (equations 5.7-5.8). However, in the coding assignment we will use ordinary importance sampling which uses a different update rule. Start with $V_n = \frac{\sum_{k=1}^{n} W_k G_k}{n}$ and derive the incremental update rule for ordinary importance sampling. Your final answer should be of the form: $V_n = V_{n-1} + a * (b - V_{n-1})$.

2. Download the notebook *RLLab2_MC.zip* from canvas assignments and follow the instructions.

3. What is the difference between Dynamic Programming and Monte Carlo? When would you use the one or the other algorithm? Include your answer in the submitted homework.

## 3.2 Importance Sampling in Monte Carlo methods

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

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

## 3.3 Temporal Difference Learning (Theory)

1. We can use Monte Carlo to get value estimates of a state with $V_M(S) = \frac{1}{M} \sum_{n=1}^{M} G_n(S)$ where $V_M(S)$ is the value estimate of state $S$ after $M$ visits of the state and $G_n(S)$ the return of an episode starting from $S$. Show that $V_M(S)$ can be written as the update rule $V_M(S) = V_{M-1}(S) + \alpha_M[G_M(S) - V_{M-1}(S)]$ and identify the learning rate $\alpha_M$.

2. Consider the TD-error

$$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t). \tag{1}$$

   (a) What is $\mathbb{E}[\delta_t | S_t = s]$ if $\delta_t$ uses the true state-value function $V^\pi$

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

## 3.4 * Exam question: Monte Carlo for control

*This exercise has been taken from a previous exam (perhaps lightly edited) and can require a bit more insight. It should give you a good idea of the level of exam questions.*

The following questions refer to the pseudo-code presented in Figure 1.

1. Part of the algorithm is covered by a black square. What is the missing information?

2. Is this a Monte-Carlo algorithm or a TD-based algorithm? Explain your answer based on the given pseudo-code.

3. What is stored in $C(S_t, A_t)$ ?

4. Why is the inner loop stopped when $A_t \neq \pi(S_t)$?

Figure 1: Algorithm pseudo-code.

```
Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
    $Q(s, a) \in \mathbb{R}$ (arbitrarily)
    $C(s, a) \leftarrow 0$
    $\pi(s) \leftarrow \arg\max_a Q(s, a)$    (with ties broken consistently)

Loop forever (for each episode):
    $b \leftarrow$ any soft policy
    Generate an episode using $b$: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
    $G \leftarrow 0$
    $W \leftarrow 1$
    Loop for each step of episode, $t =$ [redacted]:
        $G \leftarrow \gamma G + R_{t+1}$
        $C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
        $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)} [G - Q(S_t, A_t)]$
        $\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$    (with ties broken consistently)
        If $A_t \neq \pi(S_t)$ then exit inner Loop (proceed to next episode)
        $W \leftarrow W \frac{1}{b(A_t | S_t)}$
```

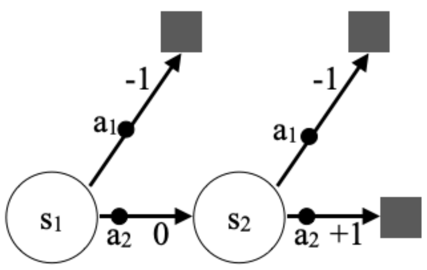

Figure 2: Example MDP

# Chapter 4: Advanced TD Methods

## 4.1 Off-policy TD

Consider the MDP in Figure 2. Consider a uniform behavior policy $b$ (probability of $a_1$ and $a_2$ is 0.5 in both states). Additionally, consider the target policy $\pi$ which takes $a_1$ with probability 0.1 and $a_2$ with probability 0.9 in both states. We consider the undiscounted case ($\gamma = 1$).

1. What are the Q functions $Q^b$ and $Q^\pi$ under both policies?
2. Now consider a dataset gathered using $b$ $(s_1, a_2, 0, s_2, a_1, -1), (s_1, a_2, 0, s_2, a_2, +1)$. Consider a Q-function that is initialized as per the following table

   |  | $a_1$ | $a_2$ |
   |---|---|---|
   | $s_1$ | -1 | 0.5 |
   | $s_2$ | -1 | +1 |

   Apply one pass of Sarsa on the dataset with a learning rate of 0.1. How does the change in Q function relate to the two functions you calculated in sub-question 1?
   *hint: throughout this question, only $Q(s_1, a_2)$ will change. Why?*
3. We can use expected Sarsa to estimate the Q-function $Q^\pi$. Apply a single pass, and note how the change in Q function relates to the two functions of sub-question 1.
4. Another possibility for off-policy learning is applying Sarsa with importance weight. Again do one pass and notice the change in Q-function.
5. We could also learn a $V$ function, e.g. through TD(0), off policy. For example, by using importance weights. Why do you think the book doesn't cover this?
6. Could you do something like expected Sarsa for learning a $V$ function? If yes, apply one pass. If not, explain why this is the case.
7. Why is Q-learning considered an off-policy control method? (ex. 6.11 in RL:AI)

## 4.2 Temporal Difference Learning (Application)

Consider an undiscounted Markov Decision Process (MDP) with two states A and B, each with two possible actions 1 and 2, and a terminal state T with $V(T) = 0$. The transition and reward functions are unknown, but you have observed the following episode using a random policy:

- $A \xrightarrow[r_3=-3]{a_3=1} B \xrightarrow[r_4=4]{a_4=1} A \xrightarrow[r_5=-4]{a_5=2} A \xrightarrow[r_6=-3]{a_6=1} B \xrightarrow[r_7=1]{a_7=2} T$

where the the arrow ($\rightarrow$) indicates a transition and $a_t$ and $r_t$ take the values of the observed actions and rewards respectively.

1. What are the state(-action) value estimates $V(s)$ (or $Q(s, a)$) after observing the sample episode when applying:
   (a) TD(0) (1-step TD)
   (b) 3-step TD
   (c) SARSA
   (d) Q-learning

   where we initialize state values to 0 and use a learning rate $\alpha = 0.1$

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.

3. Let $\pi_{random}$ 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 $\pi_{random}$ as compared to $\pi_{student}$.
   (b) What problems may arise with $\pi_{random}$ or $\pi_{student}$ respectively?
   (c) Do you think using an $\epsilon$-greedy policy as behavior policy would be beneficial? Explain why/why not?

## 4.3 Homework: Coding assignment - Temporal Difference Learning

1. Download the notebook *RLLab3_TD.zip* from canvas assignments and follow the instructions.

2. In the lab notebook you plotted the average returns during training for Q-learning and SARSA algorithms. Which algorithm achieves higher average return? Do you observe the same phenomenon as in the Example 6.6 in the book? Explain why or why not.

## 4.4 Homework: Maximization Bias

Consider the undiscounted MDP in Figure 3 where we have a starting state A with two actions (L,R), one ending in the terminal state T (with $V(T) = 0$) and always yielding a reward of 0.7 and another action that transitions to state B with zero reward. From state B we can take four different actions each transitioning to the terminal state and yielding a reward of either 0 or 1 with equal probability. Suppose that we sample two episodes where we try action L and eight episodes where try action R followed by an action from state B (each action in B is used twice). The observed rewards for each of the four actions from state B are indicated in the Figure, e.g. the rightmost action received a reward of 0 and a reward of 1. *Note: The scenario in this exercise is specifically designed to allow for a simple demonstration of the Maximization Bias.*

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

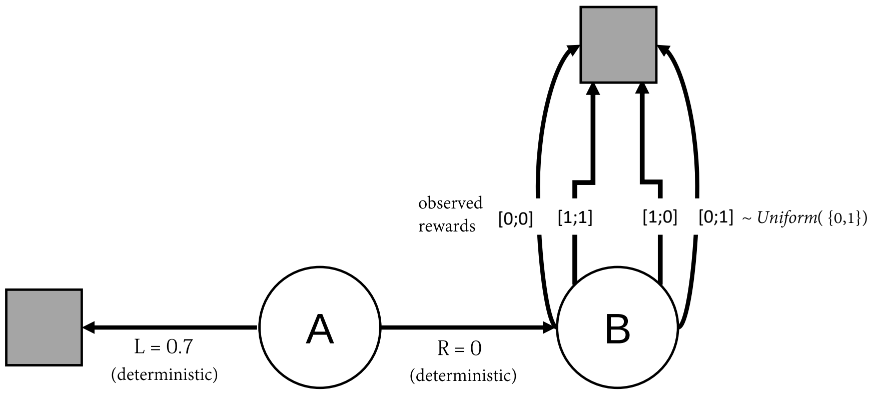

Figure 3: MDP: Maximization Bias

2. What are the true state-action values that we would expect to get (after convergence) if we continued sampling episodes.
3. 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?
4. 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.

### 4.5 Contraction Mappings & Convergence of Bellman Optimality Operator

It is important to understand the convergence behavior of reinforcement learning methods. In one iteration of the value iteration algorithm, an 'input' value function $v_n$ is mapped to an 'output' value function $v_{n+1}$. This mapping is called the *Bellman optimality operator* $B^*$, which is given by

$$(B^*V)(s) = \max_a \left[ R(s,a) + \gamma \sum_{s' \in \mathcal{S}} Pr(s'|s,a)V(s') \right], \tag{2}$$

where $V$ be the value function which gives the value per state, and $R(s,a)$ the expected immediate reward for action $a$ in state $s$ and $Pr(s'|s,a)$ the probability of transitioning to state $s'$ after taking action $a$ in state $s$.

To understand convergence of value iteration, we can look at the behavior of this operator. In this exercise, we will prove that this operator is a contraction mapping. This is very helpful, as the contraction mappings are known to converge.

A contraction mapping is a mapping $T : X \to X$ (on metric space $(X, ||\cdot||)$) if there exists a constant $0 \leq c < 1$ such that

$$||T(x) - T(y)|| \leq c||x - y|| \quad \forall x \in X, \forall y \in X \tag{3}$$

Thus, after applying the operator, the points are closer together than before by at least the factor $c$. Informally, this implies that no matter how a sequence is initialized, repeated applications of the operator will push the points closer and closer together to a single fixed point that is mapped to itself by the operator.

Formally, this results is stated in the following Theorem, from Csaba Szepesvari's *Algorithms for Reinforcement Learning*:

**Theorem 1.** (Banach's Fixed Point Theorem). Let $V$ be a Banach space and $T : V \to V$ be a contraction mapping. Then $T$ has a unique fixed point. Further, for any $v_0 \in V$, if $v_{n+1} = Tv_n$, then $v_n \to_{||\cdot||} v$, where $v$ is the unique fixed point of $T$ and the convergence is geometric:

$$||v_n - v|| \leq \gamma^n ||v_0 - v|| \tag{4}$$

(Note: a Banach space is a special kind of metric space. The precise definition is not relevant for this question).

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

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

2. (a) In the Bellman optimality operator (2), the max plays an important role. Show that

   $$|\max_z f(z) - \max_z h(z)| \leq \max_z |f(z) - h(z)| \tag{5}$$

   for arbitrary $f(z)$ and $h(z)$.

   Consider an MDP with finite state space $\mathcal{S}$, finite action space $\mathcal{A}$, and discount factor $\gamma$ and the Bellman optimality operator as given in (2)

   In our case, the metric space is $(\mathcal{B}(\mathcal{S}), ||\cdot||_\infty)$, where $\mathcal{B}(\mathcal{S})$ is the space of bounded functions on $\mathcal{S}$. $\mathcal{B}(\mathcal{S}) = \{V : \mathcal{S} \to \mathbb{R} : ||V||_\infty < +\infty\}$. $(\mathcal{B}(\mathcal{S}), ||\cdot||_\infty)$ is a normed vector space. Thus, the distance between value functions is given by the supremum norm $||V_1 - V_2||_\infty = \max_{s \in \mathcal{S}} |V_1(s) - V_2(s)|$, or the largest difference between state values.

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

   $$||(B^*V_1) - (B^*V_2)||_\infty \leq c||V_1 - V_2||_\infty \tag{6}$$

   (c) Why is this an important result?

### 4.6 N-step Temporal Difference Learning (Theory)

1. The Monte-Carlo error can be written as the sum of TD-errors if the value estimates don't change (cf. equation 6.6 in the book) as

   $$G_t - V(S_t) = \sum_{k=t}^{T-1} \gamma^{k-t}\delta_k. \tag{7}$$

   Show that the n-step error as used in

   $$V_{t+n}(S_t) = V_{t+n-1}(S_t) + \alpha[G_{t:t+n} - V_{t+n-1}(S_t)], \quad 0 \leq t < T \tag{8}$$

   can also be written as a sum TD errors (assuming the value estimates don't change).

### 4.7 *Exam question - N-Step Temporal Difference Learning

*This exercise has been taken from a previous exam (perhaps lightly edited) and can require a bit more insight. It should give you a good idea of the level of exam questions.*

Consider the undiscounted and deterministic random walk environment in Figure 4 with 19 states and two terminal states. The state in the middle (J) is always the start state and we want to apply n-step TD using a random behavior policy choosing between going left or right with equal probability at each step. The rewards are indicated above each transition arrow. The initial value of each state is set to 0. We run n-step TD for different values of n and learning rate $\alpha$. To the right you can see the average RMS error over all states compared to the true state values after 10 episodes.

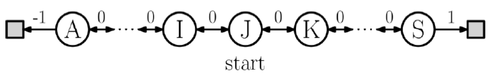

Figure 4: Random walk environment.

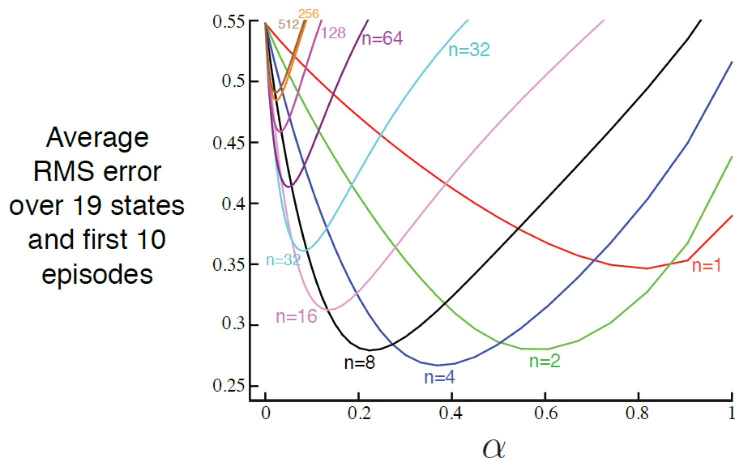

Figure 5: Performance of n-step TD methods as a function of $\alpha$ for various values of n on a 19-state random walk task.

1. We observe that we need a small learning rate when n is big in order to reduce the error. Why is this the case compared to when using a smaller n?
2. Why does an intermediate value for n work best in this scenario (assuming a good choice for $\alpha$)? You can argue about the disadvantages of the corner cases.
3. To use off-policy data with n-step methods, we can use importance weights. Do you think off-policy n-step learning with a greedy target policy (like in Q-learning) would be effective? Explain your answer.