

# Reinforcement Learning Lab

## Lesson 5: Temporal Difference Methods

Luca Marzari and Alberto Castellini

University of Verona  
*email: luca.marzari@univr.it*

Academic Year 2023-24



**UNIVERSITÀ**  
**di VERONA**  
Dipartimento  
di **INFORMATICA**

# Environment Setup

The first step for the setup of the laboratory environment is to update the repository and load the **miniconda** environment.

## Safe Procedure

Always back up the previous lessons' solutions before executing the repository update.

- Update the repository of the lab:

```
cd RL-Lab  
git stash  
git pull  
git stash pop
```

- Activate the *miniconda* environment:

```
conda activate rl-lab
```

# Today Assignment

In today's lesson, we implement the **Q-Learning** and **SARSA** algorithms in Python. In particular, the file to complete is:

---

`RL-Lab/lessons/lesson_5_code.py`

---

Inside the file, two functions are partially implemented. The objective of this lesson is to complete them.

- **def QLearning()**
- **def SARSA()**

Expected results can be found in:

---

`RL-Lab/results/lesson_5_results.txt`

---

# Algorithm: Q-Learning

Q-learning (off-policy TD control) for estimating  $\pi \approx \pi_*$

Algorithm parameters: step size  $\alpha \in (0, 1]$ , small  $\varepsilon > 0$

Initialize  $Q(s, a)$ , for all  $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$ , arbitrarily except that  $Q(\text{terminal}, \cdot) = 0$

Loop for each episode:

    Initialize  $S$

    Loop for each step of episode:

        Choose  $A$  from  $S$  using policy derived from  $Q$  (e.g.,  $\varepsilon$ -greedy)

        Take action  $A$ , observe  $R, S'$

$Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma \max_a Q(S', a) - Q(S, A)]$

$S \leftarrow S'$

    until  $S$  is terminal

**Figure:** Pseudocode for Q-Learning, from the Sutton and Barto book *Reinforcement Learning: An Introduction*

# Algorithm: SARSA

## Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Algorithm parameters: step size  $\alpha \in (0, 1]$ , small  $\varepsilon > 0$

Initialize  $Q(s, a)$ , for all  $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$ , arbitrarily except that  $Q(\text{terminal}, \cdot) = 0$

Loop for each episode:

    Initialize  $S$

    Choose  $A$  from  $S$  using policy derived from  $Q$  (e.g.,  $\varepsilon$ -greedy)

    Loop for each step of episode:

        Take action  $A$ , observe  $R, S'$

        Choose  $A'$  from  $S'$  using policy derived from  $Q$  (e.g.,  $\varepsilon$ -greedy)

$Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma Q(S', A') - Q(S, A)]$

$S \leftarrow S'; A \leftarrow A';$

    until  $S$  is terminal

**Figure:** Pseudocode for SARSA, from the Sutton and Barto book *Reinforcement Learning: An Introduction*

# Assignment Notes

Today's assignment is based on the *DangerousGridWorld* environment, the same as the previous lessons. The suggested assignment's solutions use an exploration function provided in the code, `epsilon_greedy()`. However, any other exploration strategy can be used.

## Hint

The class *DangerousGridWorld* (i.e., our environment) comes with a useful function to reset the agent to a random state: `random_initial_state()`. *The suggested assignment's solutions use it as a state-initialization function.*

## Results Disclaimer

Given the (high) stochasticity of the method, the obtained results may differ from those suggested. The crucial requirement is to obtain a policy that reaches the goal position.

# Pseudocode: Q-Learning

**Require:** *environment*  $[A, S]$ , *episodes*,  $\alpha, \gamma$ , *expl\_func*, *expl\_param*

**Ensure:** *policy*, *rewards*, *lengths*

```
1:  $\forall a \in A, \forall s \in S$  initialize  $Q(s, a)$  arbitrarily
2:  $\text{rewards}, \text{lengths} \leftarrow [0, \dots, 0]$ 
3: for  $i \leftarrow 0$  to episodes do
4:   Initialize  $s$ 
5:   repeat
6:      $a \leftarrow \text{EXPL\_FUNC}(Q, s, \text{expl\_param})$ 
7:      $s', r \leftarrow \text{take action } a \text{ from state } s$ 
8:      $Q(s, a) \leftarrow Q(s, a) + \alpha(R + \gamma \max_{a' \in A_s} Q(s', a) - Q(s, a))$ 
9:      $s \leftarrow s'$ 
10:  until  $s$  is terminal
11:  Update rewards, lengths
12:  $\pi \leftarrow [0, \dots, 0]$ 
13: for each  $s$  in  $S$  do
14:    $\pi_s \leftarrow \operatorname{argmax}_{a \in A_s} Q(s, a)$ 
15: return  $\pi, \text{rewards}, \text{lengths}$ 
```

▷ Null vectors of length *episodes*

▷ Act and observe  
▷ TD

▷ Null vector of length  $|S|$   
▷ Extract policy

# Pseudocode: SARSA

**Require:** *environment*  $[A, S]$ , *episodes*,  $\alpha, \gamma$ , *expl\_func*, *expl\_param*

**Ensure:** *policy*, *rewards*, *lengths*

- 1:  $\forall a \in A, \forall s \in S$  initialize  $Q(s, a)$  arbitrarily
- 2:  $\text{rewards}, \text{lengths} \leftarrow [0, \dots, 0]$
- 3: **for**  $i \leftarrow 0$  **to** *episodes* **do**
- 4:     Initialize  $s$
- 5:      $a \leftarrow \text{EXPL\_FUNC}(Q, s, \text{expl\_param})$
- 6:     **repeat**
- 7:          $s', r \leftarrow$  take action  $a$  from state  $s$  ▷ Act and observe
- 8:          $a' \leftarrow \text{EXPL\_FUNC}(Q, s', \text{expl\_param})$
- 9:          $Q(s, a) \leftarrow Q(s, a) + \alpha(R + \gamma Q(s', a') - Q(s, a))$  ▷ TD
- 10:          $s \leftarrow s', a \leftarrow a'$
- 11:     **until**  $s$  is terminal
- 12:     Update *rewards*, *lengths*
- 13:  $\pi \leftarrow [0, \dots, 0]$  ▷ Null vector of length  $|S|$
- 14: **for each**  $s$  **in**  $S$  **do** ▷ Extract policy
- 15:      $\pi_s \leftarrow \underset{a \in A_s}{\operatorname{argmax}} Q(s, a)$
- 16: **return**  $\pi, \text{rewards}, \text{lengths}$