Reinforcement Learning Lab

Lesson 4: Temporal Difference Methods

Gabriele Roncolato and Alberto Castellini

University of Verona
email: gabriele.roncolato@univr.it

Academic Year 2024-25

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_4_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_4_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 $Q(\mathit{terminal}, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  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 $Q(\mathit{terminal}, \cdot) = 0$

Loop for each episode:
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Loop for each step of episode:
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') - Q(S, A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  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*, α, γ, *expl_func*, *expl_param*
Ensure: *policy*, *rewards*, *lengths*

$\forall a \in A, \forall s \in S$ initialize $Q(s, a)$ arbitrarily
rewards, *lengths* $\leftarrow [0, ..., 0]$ ▷ Null vectors of length *episodes*
for $i \leftarrow 0$ **to** *episodes* **do**
    Initialize s
    repeat
        $a \leftarrow \text{EXPL\_FUNC}(Q, s, expl\_param)$
        $s', r \leftarrow$ take action a from state s ▷ Act and observe
        $Q(s, a) \leftarrow Q(s, a) + \alpha(R + \gamma \max_{a' \in A_s} Q(s', a) - Q(s, a))$ ▷ TD
        $s \leftarrow s'$
    until s is terminal
    Update *rewards*, *lengths*
$\pi \leftarrow [0, ..., 0]$ ▷ Null vector of length $|S|$
for each s **in** S **do** ▷ Extract policy
    $\pi_s \leftarrow \operatorname{argmax}_{a \in A_s} Q(s, a)$
return π, *rewards*, *lengths*

Pseudocode: SARSA

Require: *environment* $[A, S]$, *episodes*, α, γ, *expl_func*, *expl_param*
Ensure: *policy*, *rewards*, *lengths*

$\forall a \in A, \forall s \in S$ initialize $Q(s, a)$ arbitrarily
rewards, *lengths* $\leftarrow [0, ..., 0]$ ▷ Null vectors of length *episodes*
for $i \leftarrow 0$ **to** *episodes* **do**
    Initialize s
    $a \leftarrow$ EXPL_FUNC($Q, s, expl\_param$)
    repeat
        $s', r \leftarrow$ take action a from state s ▷ Act and observe
        $a' \leftarrow$ EXPL_FUNC($Q, s', expl\_param$)
        $Q(s, a) \leftarrow Q(s, a) + \alpha(R + \gamma Q(s', a') - Q(s, a))$ ▷ TD
        $s \leftarrow s'$, $a \leftarrow a'$
    until s is terminal
    Update *rewards*, *lengths*
$\pi \leftarrow [0, ..., 0]$ ▷ Null vector of length $|S|$
for each s **in** S **do** ▷ Extract policy
    $\pi_s \leftarrow \underset{a \in A_s}{\text{argmax}}\, Q(s, a)$
return π, *rewards*, *lengths*