STATISTICAL MACHINE LEARNING (SML2016)
4. COMPUTER LAB
Reinforcement Learning

Jan Drchal

1 Overview

Your task for this computer lab is to implement and perform experiments with several reinforcement learning methods. In all cases the methods will search for an optimal *action-value* function $q_*(s, a)$:

$$q_*(s, a) = \max_\pi q_\pi(s, a) \tag{1}$$

where the *action-value* function $q_\pi(s, a)$ is the expected return of starting in state s, taking action a and following policy π:

$$q_\pi(s, a) = \mathbb{E}_\pi \left(G_t \mid S_t = s, A_t = a \right). \tag{2}$$

The methods will be:

1. **Value iteration**: a dynamic programming approach requiring knowledge of Markov Decision Process (MDP) state transition probabilities $\mathcal{P}^a_{ss'} = \mathbb{P}\left(S_{t+1} = s' \mid S_t = s, A_t = a\right)$ and reward function $\mathrm{R}^a_s = \mathbb{E}\left(R_{t+1} \mid S_t = s, A_t = a\right)$.
2. **Policy Iteration**: an alternative to the *value iteration*. The implementation of this method is a bonus task.
3. **SARSA**: an on-policy method based on Temporal-Difference (TD(0)).
4. **Q-learning**: an off-policy method based on TD(0).

To evaluate the methods you will run experiments on the *windy gridworld* as defined in [1][1]. The gridworld is depicted and described in Figure 1.

You will evaluate the convergence of SARSA and Q-learning with respect to optimal action-value function obtained by means of the value iteration method. Use mean-squared error to assess quality of the action-value function approximation $q(s, a)$:

$$E = \frac{1}{|\mathcal{S}| \cdot |\mathcal{A}|} \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \left[q_*(s, a) - q(s, a) \right]^2. \tag{3}$$

2 Download

Download an archive from the following link:

`https://cw.fel.cvut.cz/wiki/_media/courses/be4m33ssu/rl_data.zip`

[1] Better download a draft to the second edition here: `https://webdocs.cs.ualberta.ca/~sutton/book/bookdraft2016sep.pdf`.

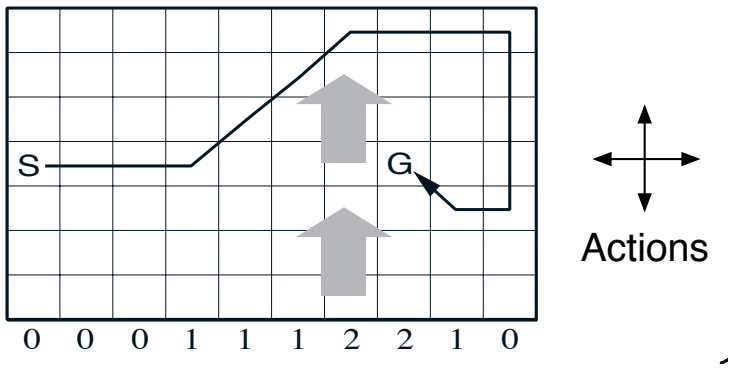

Figure 1: **Windy gridworld.** The task is to navigate an agent from the starting state (`S`) to the goal state (`G`). The possible actions are `left`, `right`, `up` and `down`. The results of actions are deterministic and work as expected, however, wind will move agent upward by the number of steps determined by the wind strength values of which are shown below the grid. Moves leading agent out of the grid will be always truncated to keep him inside. The rewards are also deterministic: -1 for each step until the goal state is reached. Use undiscounted rewards ($\gamma = 1$) in all cases. Figure was taken from [1].

The zip file contains a single Python source file `snippets.py`. It defines the Windy gridworld MDP as well as other useful functions including visualization. The functions are commented. If you are not into Python a translation to other programming language should be straightforward using this source code as a reference.

3 Task assignment

Assignment 1 (3 points) *Implement the value iteration method for action-value function. Note that the lecture slides as well as the book [1] give a version of the algorithm for the state-value function. Your implementation should work directly with action-value function, do not convert between the two representations. Give:*

- *The number of iterations needed to find an optimal action-value function. Start with the $q_0(s, a) = 0$ for all states s and actions a.*
- *The optimal action-value function as a list of four matrices, one per each available action.*

Assignment 2 (4 points) *Implement the SARSA algorithm using ϵ-greedy policy. Decide for reasonable three settings of the ϵ as well as three settings for the step-size parameter α to demonstrate their influence on the convergence. Measure the error using mean-squared error (MSE) with respect to the optimal action-value function obtained in the Assignment 1. Give:*

- *Plots showing convergence for all $3 \times 3 = 9$ parameter combinations. Repeat computations for each parameter combination and show averages only. The plots will show MSE evaluated every 100 steps (state transitions). The maximum number of steps (and hence episodes) is up to you.*
- *Minimum average error achieved for each parameter combination.*
- *Shortly discuss the results.*

Assignment 3 (3 points) *Implement the Q-learning algorithm using ϵ-greedy policy. Give exactly the same type of outputs as in the case of SARSA.*

Assignment 4 (2 bonus points) *Implement the policy iteration algorithm working natively with action-value function as does the value iteration in the Assignment 1. Both algorithms converge quickly to the exact optimum. Which one is faster?*

References

[1] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.