# Homework 5. Reinforcement learning

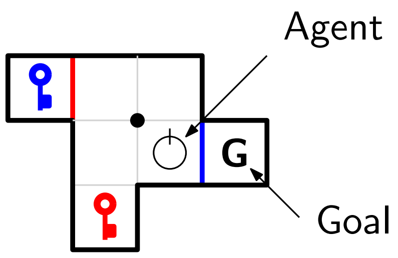

Figure 1: Grid world where an agent must reach the cell marked with "**G**".

Consider once again the maze world from Lab 2, where an agent moves in the grid-world environment of Fig. 1. The agent must reach the goal cell, marked with "**G**".

At each step, the agent may move in any of the four directions—up, down, left and right. Movement across a *grey* cell division succeeds with a 0.8 probability and fails with a 0.2 probability. Movements across colored cell divisions (blue or red) succeed with a 0.8 probability *only if the agent has the corresponding colored key*. Otherwise, they fail with probability 1. When the movement fails, the agent remains in the same cell.

To get a colored key, the agent simply needs to stand in the corresponding cell. In other words, as soon as the agent stands on the cell of a colored key, you may consider that it holds that key thereafter.

This problem can be modeled as a Markov decision problem where, numbering the cells from 1 to 7, starting from left to right and from top to bottom, we have the state space

$$\begin{aligned}\mathcal{X} = \big\{&(1,b,r),(2,\bar{b},\bar{r}),(2,\bar{b},r),(2,b,r),(3,\bar{b},\bar{r}),(3,\bar{b},r),(3,b,r),\\ &(4,\bar{b},\bar{r}),(4,\bar{b},r),(4,b,r),(5,\bar{b},\bar{r}),(5,\bar{b},r),(5,b,r),(6,b,r),(7,\bar{b},r),(7,b,r)\big\},\end{aligned}$$

where we assume that the agent can never get to the blue key without the red key and can never get to the goal without both keys, and the action space

$$\mathcal{A} = \{u,d,l,r\}.$$

Suppose that the agent is allowed to move around the environment for $T$ time steps, experimenting the different actions and visiting the different states. A excerpt of its trajectory is

$$\begin{gathered}
\dots \\
x_{t-1} = (4, \bar{b}, r), \\
a_{t-1} = r, \\
c_{t-1} = 1.0, \\
x_t = (5, \bar{b}, r), \\
a_t = u, \\
c_t = 1.0, \\
x_{t+1} = (5, \bar{b}, r), \\
a_{t+1} = r, \\
c_{t+1} = 1.0, \\
\dots
\end{gathered}$$

At time step $t$, the $Q$-values estimated by the for state $(4, \bar{b}, r)$ are:

$$\mathbf{Q}^{(t)}_{(4,\bar{b},r),:} = \begin{bmatrix} 3.23 & 3.34 & 3.25 & 3.22 \end{bmatrix}$$

and for state $(5, \bar{b}, r)$,

$$\mathbf{Q}^{(t)}_{(5,\bar{b},r),:} = \begin{bmatrix} 3.08 & 3.25 & 3.57 & 3.22 \end{bmatrix}.$$

## Exercise 1.

(a) Indicate the $Q$-values after a $Q$-learning update with step-size $\alpha = 0.1$, resulting from the transition at time step $t$. Use $\gamma = 0.99$.

(b) Indicate the $Q$-values after a SARSA update with step-size $\alpha = 0.1$, resulting from the transition at time step $t$.

(c) Explain the difference between on-policy and off-policy learning using Questions 1a and 1b to illustrate your explanation.

**Solution 1:**

(a) The $Q$-learning update is given by

$$Q^{(t+1)}(x_t, a_t) = Q^{(t)}(x_t, a_t) + \alpha(c_t + \gamma \min_{a' \in \mathcal{A}} Q^{(t)}(x_{t+1}, a') - Q^{(t)}(x_t, a_t))$$

where, from the provided transitions,

$$x_t = (5, \bar{b}, r), \qquad a_t = u, \qquad c_t = 1.0, \qquad x_{t+1} = (5, \bar{b}, r).$$

This results in the update

$$Q^{(t+1)}((5, \bar{b}, r), u) = 3.08 + 0.1 \times (1 + 0.99 \times 3.08 - 3.08) = 3.177.$$

(b) The SARSA update is given by

$$Q^{(t+1)}(x_t, a_t) = Q^{(t)}(x_t, a_t) + \alpha(c_t + \gamma Q^{(t)}(x_{t+1}, a_{t+1}) - Q^{(t)}(x_t, a_t))$$

where, from the provided transitions,

$$x_t = (5, \bar{b}, r), \qquad a_t = u, \qquad c_t = 1.0, \qquad x_{t+1} = (5, \bar{b}, r) \qquad a_{t+1} = r.$$

This results in the update

$$Q^{(t+1)}((5, \bar{b}, r), u) = 3.08 + 0.1 \times (1.0 + 0.99 \times 3.22 - 3.08) = 3.191.$$

(c) An on-policy learning algorithm learns the values associated with the policy used to sample the MDP. For example, SARSA is an on-policy algorithm, as it learns the $Q$-values for the learning policy. This is apparent in its dependence on the action $a_{t+1}$, as seen in Question 1b.

An off-policy learning algorithm, on the other hand, learns the value associated with some target policy—not necessarily the one used to sample the MDP. For example, $Q$-learning is an off-policy algorithm, as it learns the $Q$-values for the optimal policy independently of the learning policy. This is apparent in its computation of the action $a'$ that minimizes $Q(x_{t+1}, a')$, as seen in Question 1a.