# Introduction to Artificial Intelligence (INFO8006)

## Exercises 7 – Reinforcement learning

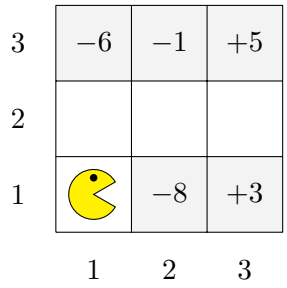
September 16, 2022

## Learning outcomes

At the end of this session you should be able to

- differentiate passive and active RL;
- define and apply direct utility estimation and temporal-difference learning;
- define and apply $Q$-learning.

## Exercise 1 Passive RL




Consider the grid-world given above and an agent who is trying to learn the optimal policy. The agent starts from the bottom-left corner and can take the actions `north` ($N$), `south` ($S$), `west` ($W$) and `east` ($E$). Rewards are only awarded for reaching the terminal (shaded) states. You observe the following trials, whose trajectories are sequences of tuples $(s_t^i, r_t^i, a_t^i, s_{t+1}^i)$.

| $t$ | Trial 1 | Trial 2 | Trial 3 | Trial 4 | Trial 5 |
|---|---|---|---|---|---|
| 0 | $(1,1), 0, N, (1,2)$ | $(1,1), 0, N, (1,2)$ | $(1,1), 0, N, (1,2)$ | $(1,1), 0, N, (1,2)$ | $(1,1), 0, N, (1,2)$ |
| 1 | $(1,2), 0, E, (2,2)$ | $(1,2), 0, E, (2,2)$ | $(1,2), 0, E, (2,2)$ | $(1,2), 0, E, (2,2)$ | $(1,2), 0, E, (2,2)$ |
| 2 | $(2,2), 0, N, (2,3)$ | $(2,2), 0, E, (3,2)$ | $(2,2), 0, S, (2,1)$ | $(2,2), 0, E, (3,2)$ | $(2,2), 0, E, (3,2)$ |
| 3 | $(2,3), -1, \varnothing, \varnothing$ | $(3,2), 0, N, (3,3)$ | $(2,1), -8, \varnothing, \varnothing$ | $(3,2), 0, W, (2,2)$ | $(3,2), 0, S, (3,1)$ |
| 4 | | $(3,3), +5, \varnothing, \varnothing$ | | $(2,2), 0, N, (2,3)$ | $(3,1), +3, \varnothing, \varnothing$ |
| 5 | | | | $(2,3), -1, \varnothing, \varnothing$ | |

Assuming a discount factor $\gamma = 1$,

1. Perform direct utility estimation of the expected utilities $V^\pi(s)$, given the four first trials.

   In this setting, the transition model $P(s'|s,a)$ and the reward function $R(s)$ are unknown. We wish to learn the expected utility $V^\pi$ of the policy $\pi$ without modeling the environment, *i.e.* without building approximates $\hat{P}(s'|s,a)$ and $\hat{R}(s)$. The principle of direct utility estimation is to approximate $V^\pi(s)$ by the average utility $\hat{V}(s)$ of the state $s$ within all trial trajectories, *i.e.*

$$\hat{V}(s) = \frac{1}{N(s)} \sum_{(i,j)\in I(s)} \sum_{t=0}^{\infty} \gamma^t r_{j+t}^i \approx V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t)\right]\Bigg|_{s_0=s}$$

   where $I(s) = \{(i,j) : s_j^i = s\}$ indexes the occurrences of $s$ and $N(s) = |I(s)|$ is the number

of occurrences of $s$. For examples, in our case and excluding the fifth trial,

$$
\begin{aligned}
I((1,1)) &= \{(1,0),(2,0),(3,0),(4,0)\} \\
\hat{V}((1,1)) &\approx \frac{1}{4}\Big(-1\,\gamma^3 + 5\,\gamma^4 - 8\,\gamma^3 - 1\,\gamma^5\Big) = \frac{-5}{4} \\
I((2,2)) &= \{(1,2),(2,2),(3,2),(4,2),(4,4)\} \\
\hat{V}((2,2)) &\approx \frac{1}{5}\Big(-1\,\gamma^1 + 5\,\gamma^2 - 8\,\gamma^1 - 1\,\gamma^3 - 1\,\gamma^1\Big) = \frac{-6}{5} \\
I((2,3)) &= \{(1,3),(4,5)\} \\
\hat{V}((2,3)) &\approx \frac{1}{2}\Big(-1\,\gamma^0 - 1\,\gamma^0\Big) = -1.
\end{aligned}
$$

Importantly, since the agent hasn't reached the states $(1,3)$ and $(3,3)$ yet, it is not possible to estimate their expected utility. Instead, we assume a default value of 0.

| | | | |
|---|---|---|---|
| 3 | 0 | $-1$ | $+5$ |
| 2 | $\frac{-5}{4}$ | $\frac{-6}{5}$ | $+2$ |
| 1 | $\frac{-5}{4}$ | $-8$ | 0 |
| | 1 | 2 | 3 |

2. Update the estimated expected utilities with respect to the fifth trial using temporal-difference learning. Assume a learning rate $\alpha = 0.5$.

   Direct utility estimation misses the fact that the expected utilities for different states are not independent, since they obey the Bellman equation for a fixed policy

   $$V^\pi(s) = R(s) + \gamma \sum_{s'} P(s'|s,\pi(s))V^\pi(s').$$

   Then, we would like our estimates $\hat{V}(s)$ to verify the Bellman equation, *i.e.* to minimize

   $$L = \mathop{\mathbb{E}}_{s'|s}\Big[\Big(R(s) + \gamma\hat{V}(s') - \hat{V}(s)\Big)^2\Big].$$

   One way to reach this objective is to follow the opposite of its gradient

   $$\nabla L = 2\mathop{\mathbb{E}}_{s'|s}\Big[\hat{V}(s) - R(s) - \gamma\hat{V}(s')\Big]$$

   with respect to $\hat{V}(s)$, *i.e.* to perform *gradient descent*. Unfortunately, due to the expectation, we don't have access to the true gradient. Instead, we use a *stochastic* approximation of the gradient from observed[1] tuples $(s,r,a,s')$ and update $\hat{V}(s)$ as

   $$
   \begin{aligned}
   \hat{V}(s) &\leftarrow \hat{V}(s) - \alpha\Big(\hat{V}(s) - r - \gamma\hat{V}(s')\Big) \\
   &\leftarrow (1-\alpha)\hat{V}(s) + \alpha\Big(r + \gamma\hat{V}(s')\Big)
   \end{aligned}
   $$

   where $\alpha$ is the learning rate. For a fixed learning rate, the average of $\hat{V}(s)$ converge towards (oscillates around) the true expected utility $V^\pi(s)$. For a (slowly) decreasing learning rate, $\hat{V}(s)$ itself converges to $V^\pi(s)$. For historical reasons, this algorithm is called temporal-difference (TD) learning.

[1] We are not limited to perform only one update for each tuple $(s,r,a,s')$. Indeed, as we assume the policy to be stationary, past trials are as likely to happen again as more recent ones. Therefore, reusing tuples several times (in other orders) is a good way to improve estimates while generating less trials.

Following the trajectory of the fifth trial, we perform the updates

$$V^\pi((1,1)) \leftarrow (1-\alpha)V^\pi((1,1)) + \alpha(0 + \gamma V^\pi((1,2))) = \frac{1}{2}\frac{-5}{4} + \frac{1}{2}\frac{-5}{4} = \frac{-5}{4}$$
$$V^\pi((1,2)) \leftarrow (1-\alpha)V^\pi((1,2)) + \alpha(0 + \gamma V^\pi((2,2))) = \frac{1}{2}\frac{-5}{4} + \frac{1}{2}\frac{-6}{5} = \frac{-49}{40}$$
$$V^\pi((2,2)) \leftarrow (1-\alpha)V^\pi((2,2)) + \alpha(0 + \gamma V^\pi((3,2))) = \frac{1}{2}\frac{-6}{5} + \frac{1}{2}2 = \frac{+4}{10}$$
$$V^\pi((3,2)) \leftarrow (1-\alpha)V^\pi((3,2)) + \alpha(0 + \gamma V^\pi((3,1))) = \frac{1}{2}2 + \frac{1}{2}0 = +1$$
$$V^\pi((3,1)) \leftarrow (1-\alpha)V^\pi((3,1)) + \alpha(3 + 0) = \frac{1}{2}0 + \frac{1}{2}3 = \frac{+3}{2}.$$

| | | | |
|---|---|---|---|
| 3 | $0$ | $-1$ | $+5$ |
| 2 | $\frac{-49}{40}$ | $\frac{+4}{10}$ | $+1$ |
| 1 | $\frac{-5}{4}$ | $-8$ | $\frac{+3}{2}$ |
| | 1 | 2 | 3 |

## Exercise 2 Q-learning

An agent is in an unknown environment where there are three states $\{A, B, C\}$ and two actions $\{0, 1\}$. We are given the following tuples $(s, a, r, s')$, generated by taking actions in the environment.

| $s$ | $a$ | $r$ | $s'$ |
|---|---|---|---|
| $A$ | 0 | $+2$ | $A$ |
| $C$ | 1 | $-2$ | $A$ |
| $B$ | 1 | $+1$ | $B$ |
| $A$ | 0 | $-1$ | $B$ |
| $B$ | 1 | $-2$ | $C$ |
| $C$ | 0 | $+4$ | $B$ |
| $B$ | 0 | $+1$ | $A$ |

Assuming a discount factor $\gamma = 0.5$ and a learning rate $\alpha = 0.75$,

1. Apply the $Q$-learning algorithm to obtain state-action-value $Q(s, a)$ estimates. Estimates are initialized to 0.

   As seen in the previous exercise, given trial trajectories of some policy $\pi$, it is possible to estimate its expected utility $V^\pi$ even without knowing the transition model $P(s'|s, a)$ or the reward function $R(s)$ of the environment. However, estimating $V^\pi$ only evaluates the quality of $\pi$, but does not describe how to improve it.

   By definition, the optimal policy $\pi^*$ is the one that maximizes the expect utility of the state $s$, *i.e.*

   $$\pi^*(s) = \arg\max_a \sum_{s'} P(s'|s, a)V(s')$$

   where $V(s) = \max_\pi V^\pi(s)$ is the state-value of $s$. Unfortunately, even if we knew $V$, we could not find the optimal actions without knowing the transition model. However, if we knew the state-action-value $Q(s, a)$ of taking action $a$ in state $s$, we would be able to select the optimal action

   $$\pi^*(s) = \arg\max_a Q(s, a),$$

   where

   $$\begin{aligned} Q(s, a) &= R(s) + \gamma \sum_{s'} P(s'|s, a)V(s') \\ &= R(s) + \gamma \sum_{s'} P(s'|s, a) \max_{a'} Q(s', a') \end{aligned}$$

   since $V(s) = \max_a Q(s, a)$. Fortunately, state-action-values $Q(s, a)$ can be learned in a model-free fashion using the $Q$-learning algorithm. Similarly to temporal-difference learning, in $Q$-learning, we perform stochastic gradient descent updates

   $$\begin{aligned} \hat{Q}(s, a) &\leftarrow \hat{Q}(s, a) - \alpha\Big(\hat{Q}(s, a) - r - \gamma \max_{a'} \hat{Q}(s', a')\Big) \\ &\leftarrow (1 - \alpha)\hat{Q}(s, a) + \alpha\Big(r + \gamma \max_{a'} \hat{Q}(s', a')\Big) \end{aligned}$$

   from observed tuples $(s, r, a, s')$.

In our case, the sequence of updates would be

$$\hat{Q}(A,0) \leftarrow (1-\alpha)\hat{Q}(A,0) + \alpha\Big(2+\gamma\max\{\hat{Q}(A,0),\hat{Q}(A,1)\}\Big) = \frac{1}{4}0 + \frac{3}{4}(2+\frac{1}{2}0) = \frac{+3}{2}$$
$$\hat{Q}(C,1) \leftarrow (1-\alpha)\hat{Q}(C,1) + \alpha\Big(-2+\gamma\max\{\hat{Q}(A,0),\hat{Q}(A,1)\}\Big) = \frac{1}{4}0 + \frac{3}{4}(-2+\frac{1}{2}\frac{3}{2}) = \frac{-15}{16}$$
$$\hat{Q}(B,1) \leftarrow (1-\alpha)\hat{Q}(B,1) + \alpha\Big(2+\gamma\max\{\hat{Q}(B,0),\hat{Q}(B,1)\}\Big) = \frac{1}{4}0 + \frac{3}{4}(1+\frac{1}{2}0) = \frac{+3}{4}$$
$$\hat{Q}(A,0) \leftarrow (1-\alpha)\hat{Q}(A,0) + \alpha\Big(-1+\gamma\max\{\hat{Q}(B,0),\hat{Q}(B,1)\}\Big) = \frac{1}{4}\frac{3}{2} + \frac{3}{4}(-1+\frac{1}{2}\frac{3}{4}) = \frac{-3}{32}$$
$$\hat{Q}(B,1) \leftarrow \dots$$

It should be noted that, as for TD learning, we are not limited to perform only one update for each tuple. Reusing tuples several times (in other orders) is a good way to reach convergence faster while generating less trials.

2. We now switch to a feature-based estimator $\hat{Q}(s,a) = w_0 + w_1 f_1(s,a)$, with $f_1(s,a) = 2a-1$. Starting from weights $w_0 = w_1 = 0$, update the weights according to the approximate $Q$-learning algorithm.

   We would like our estimates $\hat{Q}(s,a)$ to satisfy the Bellman equation, *i.e.* to minimize

   $$L = \mathop{\mathbb{E}}_{s'|s,a}\left[\left(R(s) + \gamma\max_{a'}\hat{Q}(s',a') - \hat{Q}(s,a)\right)^2\right].$$

   To reach this objective we follow the opposite of its gradient

   $$\nabla_w L = 2\mathop{\mathbb{E}}_{s'|s,a}\left[\left(\hat{Q}(s,a) - R(s) - \gamma\max_{a'}\hat{Q}(s',a')\right)\nabla_w\hat{Q}(s,a)\right]$$

   with respect to the weights[2] $w$ of $\hat{Q}(s,a)$. Once again, due to the expectation, we are forced to use a stochastic approximation of the gradient from observed tuples $(s,a,r,s')$ and to update the weights as

   $$w \leftarrow w - \alpha\left(\hat{Q}(s,a) - r - \gamma\max_{a'}\hat{Q}(s',a')\right)\nabla_w\hat{Q}(s,a),$$

   where $\alpha$ is the learning rate. In our case,

   $$\nabla_w\hat{Q}(s,a) = \begin{pmatrix}\partial_{w_0}\\ \partial_{w_1}\end{pmatrix}(w_0 + w_1 f_1(s,a)) = \begin{pmatrix}1\\ f_1(s,a)\end{pmatrix} = \begin{pmatrix}1\\ 2a-1\end{pmatrix}$$

   and the two first weight updates would be

   $$\begin{aligned} w &\leftarrow w - \alpha\Big(\hat{Q}(A,0) - 2 - \gamma\max\{\hat{Q}(A,0),\hat{Q}(A,1)\}\Big)\nabla_w\hat{Q}(A,0)\\ &\leftarrow \begin{pmatrix}0\\0\end{pmatrix} - \frac{3}{4}\left(-2 - \frac{1}{2}\max\{0,0\}\right)\begin{pmatrix}1\\-1\end{pmatrix} = \begin{pmatrix}\frac{+3}{2}\\ \frac{-3}{2}\end{pmatrix}\\ w &\leftarrow w - \alpha\Big(\hat{Q}(C,1) + 2 - \gamma\max\{\hat{Q}(A,0),\hat{Q}(A,1)\}\Big)\nabla_w\hat{Q}(C,1)\\ &\leftarrow \begin{pmatrix}\frac{+3}{2}\\ \frac{-3}{2}\end{pmatrix} - \frac{3}{4}\left(2 - \frac{1}{2}\max\{0,3\}\right)\begin{pmatrix}1\\1\end{pmatrix} = \begin{pmatrix}\frac{+9}{8}\\ \frac{-15}{8}\end{pmatrix}. \end{aligned}$$

[2] It should be noted that the target $R(s) + \gamma\max_{a'}\hat{Q}(s',a')$ should not affected by the gradient operation as it is considered constant in the objective. In practice, this is not true as modifying the weights in a parametric function $\hat{Q}$ does so for all state-action pairs $(s,a)$ at once. This generally makes the approximate $Q$-learning algorithm very unstable.

## Exercise 3 Football

ULiège's football team is playing against UCL's team next week. With a lot of losses this season, Liège needs to improve their attack strategy to win the game and increase their popularity. Luckily, the team captain follows INFO8006 and knows how to model the attack as a Markov Decision Process. The captain considers four states `close` (`C`), `away` (`A`), `fail` (`F`), and `goal` (`G`), and two actions `pass` (`P`) and `shoot` (`S`). Although the transition probabilities are unsure, the possible transitions $(s, a, s')$ are known. To each transition is associated an increase/decrease of the team's popularity.

| $s$ | $a$ | $s'$ | $R(s, a, s')$ |
|---|---|---|---|
| C | P | C | $+1$ |
| C | P | A | $-1$ |
| C | P | F | $-2$ |
| C | S | C | $+3$ |
| C | S | F | $-5$ |
| C | S | G | $+10$ |
| A | P | C | $+2$ |
| A | P | A | $0$ |
| A | P | F | $-3$ |
| A | S | C | $+3$ |
| A | S | F | $-10$ |
| A | S | G | $+20$ |

The current strategy of the team is to always shoot. Last match, they had several attack opportunities, resulting in the following actions.

| $s$ | C | C | C | C | A | A | A | A |
|---|---|---|---|---|---|---|---|---|
| $a$ | S | S | S | S | S | S | S | S |
| $s'$ | G | C | G | F | F | C | F | F |

Assuming a discount factor $\gamma = 0.75$ and a learning rate $\alpha = 0.25$,

1. Build an estimator of the transition model $P(s'|s, a)$ and, from it, determine the expected utility $V^\pi$ of the team's current policy $\pi$. Given the previous table, we can estimate the transition probabilities for $s \in \{\texttt{C}, \texttt{A}\}$ and $a = \texttt{S}$.

| $s$ | $a$ | $P(s'\|s, a)$ | | | |
|---|---|---|---|---|---|
| | | C | A | F | G |
| C | S | 0.25 | 0 | 0.25 | 0.5 |
| A | S | 0.25 | 0 | 0.75 | 0 |

   We can now determine the expected utility $V^\pi$ of the team's current policy $\pi(s) = \texttt{S}$ using the following Bellman operator

$$V^\pi_{i+1}(s) = \sum_{s'} P(s'|s, \pi(s))\big[R(s, \pi(s), s') + \gamma V^\pi_i(s')\big],$$

   with $V^\pi_0(s) = 0$ for all state $s$. Because the states `F`and `G`are terminal, their expected utility

remains constant and we only have to update Cand A. For the first iteration,

$$\begin{aligned} V_1^\pi(\texttt{C}) &= \sum_{s'} P(s'|\texttt{C},\texttt{S})\big[R(\texttt{C},\texttt{S},s') + \gamma V_0^\pi(s')\big] \\ &= 0.25(3+\gamma\,0) + 0.25(-5+\gamma\,0) + 0.5(10+\gamma\,0) = 4.5 \end{aligned}$$

and

$$\begin{aligned} V_1^\pi(\texttt{A}) &= \sum_{s'} P(s'|\texttt{A},\texttt{S})\big[R(\texttt{A},\texttt{S},s') + \gamma V_0^\pi(s')\big] \\ &= 0.25(3+\gamma\,0) + 0.75(-10+\gamma\,0) = -6.75. \end{aligned}$$

We repeat the same procedure until $V_{i+1}^\pi(s) \approx V_i^\pi(s)$ for all state $s$.

| $s$ | C | A | F | G |
|---|---|---|---|---|
| $V_0^\pi(s)$ | 0 | 0 | 0 | 0 |
| $V_1^\pi(s)$ | 4.5 | -6.75 | 0 | 0 |
| $V_2^\pi(s)$ | 5.3437 | −5.9063 | 0 | 0 |
| $V_3^\pi(s)$ | 5.5019 | −5.7480 | 0 | 0 |
| $V_4^\pi(s)$ | 5.5316 | −5.7183 | 0 | 0 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| $V_8^\pi(s)$ | 5.5385 | −5.7115 | 0 | 0 |

The captain found the tapes of the previous season where they had much more success. Together with the team, the captain selects the following instructive actions.

| $s$ | C | C | A | A | C | A | A | C |
|---|---|---|---|---|---|---|---|---|
| $a$ | S | S | S | P | P | P | S | P |
| $s'$ | G | C | F | C | C | C | F | F |

2. Given the selected tuples, apply the $Q$-learning algorithm to obtain state-action-value $Q(s,a)$ estimates. Estimates are initialized to 0.

   In $Q$-learning, the state-action-value $Q(s,a)$ estimates are updated following

$$\begin{aligned} Q(s,a) &\leftarrow Q(s,a) - \alpha\Big(Q(s,a) - r - \gamma \max_{a'} Q(s',a')\Big) \\ &\leftarrow (1-\alpha)Q(s,a) + \alpha\Big(r + \gamma \max_{a'} Q(s',a')\Big) \end{aligned}$$

   for observed tuples $(s,r,a,s')$. In our case, we have a list a 16 (8 + 8) tuples $(s,a,s')$ and we have access to the reward function $R(s,a,s')$. Using the tuples in arbitrary order, we

have

$$
\begin{aligned}
Q(\texttt{C},\texttt{S}) &\leftarrow (1-\alpha)Q(\texttt{C},\texttt{S}) + \alpha\Big(R(\texttt{C},\texttt{S},\texttt{G}) + \gamma \max_{a'} Q(\texttt{G},a')\Big) \\
&\leftarrow 0.75 \times 0 + 0.25(10+0) = 2.5 \\
Q(\texttt{C},\texttt{S}) &\leftarrow (1-\alpha)Q(\texttt{C},\texttt{S}) + \alpha\Big(R(\texttt{C},\texttt{S},\texttt{C}) + \gamma \max_{a'} Q(\texttt{C},a')\Big) \\
&\leftarrow 0.75 \times 2.5 + 0.25(3 + 0.75\max\{2.5,0\}) = 3.094 \\
Q(\texttt{A},\texttt{S}) &\leftarrow (1-\alpha)Q(\texttt{A},\texttt{S}) + \alpha\Big(R(\texttt{A},\texttt{S},\texttt{F}) + \gamma \max_{a'} Q(\texttt{F},a')\Big) \\
&\leftarrow 0.75 \times 0 + 0.25(-10+0) = -2.5 \\
Q(\texttt{A},\texttt{P}) &\leftarrow (1-\alpha)Q(\texttt{A},\texttt{P}) + \alpha\Big(R(\texttt{A},\texttt{P},\texttt{C}) + \gamma \max_{a'} Q(\texttt{C},a')\Big) \\
&\leftarrow 0.75 \times 0 + 0.25(2 + 0.75\max\{3.094,0\}) = 4.320 \\
Q(\texttt{C},\texttt{P}) &\leftarrow \ldots
\end{aligned}
$$

3. Determine the optimal policy according to the state-action-value estimates.

$$
\pi(s) = \begin{cases} \texttt{S} & \text{if } s = \texttt{C} \\ \texttt{P} & \text{if } s = \texttt{A} \end{cases}
$$

## Supplementary materials

- Playing Atari with Deep Reinforcement Learning

- Chapter 21 of the reference textbook.