TD 9 – Learning `III` Reinforcement Learning

1 Markov Decision Process

① *Return*

The return is the total future rewards :

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... = \sum_{k=0}^{+\infty} \gamma^k r_{t+k+1}$$

The recursive form is obtained by factorizing by γ :

$$G_t = r_{t+1} + \gamma(r_{t+2} + \gamma r_{t+3} + ...) = r_{t+1} + \gamma G_{t+1}$$

Useful for the questions below :

By definition, the state-action value function is the expected return if the agent starts in state s_t, chooses the action a_t, and then follows the policy π :

$$Q_\pi(a_t, s_t) = \mathbb{E}(G_t | a_t, s_t)$$

By definition, the state value function is the expected return if the agent starts in state s_t and then follows the policy π :

$$V_\pi(s_t) = \mathbb{E}(G_t | s_t)$$

② *Value of an action in a state* The subjective value of an action decomposes in two contributions (by the linearity of the expectancy) : the reward expected at the next time $t+1$, and the return expected after.

$$Q_\pi(a_t, s_t) = \mathbb{E}(G_t | a_t, s_t) = \mathbb{E}(r_{t+1} + \gamma G_{t+1} \mid a_t, s_t) = \mathbb{E}(r_{t+1} \mid a_t, s_t) + \gamma \mathbb{E}(G_{t+1} \mid a_t, s_t)$$

• Once an action a_t and a state s_t are fixed, the random variable r_{t+1} becomes determined by the reward function :

$$\mathbb{E}(r_{t+1} \mid a_t, s_t) = R(a_t, s_t)$$

• To introduce the state-value function in the expression of $\mathbb{E}(G_{t+1} \mid a_t, s_t)$, it is possible to use the law of total expectation over all the possible states s_{t+1} :

$$\mathbb{E}(G_{t+1} \mid a_t, s_t) = \sum_{s_{t+1}} \mathbb{P}(s_{t+1} | a_t, s_t) \mathbb{E}(G_{t+1} \mid s_{t+1}, a_t, s_t)$$

The random variable G_{t+1} only depends on the state s_{t+1} : $\mathbb{E}(G_{t+1} \mid s_{t+1}, a_t, s_t) = \mathbb{E}(G_{t+1} \mid s_{t+1}) = V_\pi(s_{t+1})$.
Moreover, the probability $\mathbb{P}(s_{t+1} | a_t, s_t)$ corresponds to the transition probability function.
Therefore :

$$\mathbb{E}(G_{t+1} | a_t, s_t) = \sum_{s_{t+1}} p_{tr}(s_{t+1} | a_t, s_t) V_\pi(s_{t+1})$$

• Gathering both terms leads to the recursive expression :

$$Q_\pi(a_t, s_t) = R(s_t, a_t) + \gamma \sum_{s_{t+1}} p_{tr}(s_{t+1} | a_t, s_t) V_\pi(s_{t+1})$$

③ *Value of a state*

The value of a state is the expected return if the agent starts in state s_t and follow its policy :

$$V(s_t) = \mathbb{E}(G_t | s_t) = \mathbb{E}(r_{t+1} + \gamma G_{t+1} | s_t) = \mathbb{E}(r_{t+1} | s_t) + \gamma \mathbb{E}(G_{t+1} | s_t)$$

- The random variable r_{t+1} depends on the choice of the action a_t in the current state s_t :

$$\mathbb{E}(r_{t+1}|s_t) = \sum_{a_t} \pi(a_t|s_t) R(s_t, a_t)$$

- Similarly as before, with the law to total expectation over all the possible states s_{t+1} :

$$\mathbb{E}(G_{t+1} \mid s_t) = \sum_{s_{t+1}} \mathbb{P}(s_{t+1}|s_t) \mathbb{E}(G_{t+1} \mid s_{t+1}, s_t)$$

As before : $\mathbb{E}(G_{t+1} \mid s_{t+1}, s_t) = \mathbb{E}(G_{t+1} \mid s_{t+1}) = V_\pi(s_{t+1})$.
Now, the conditional probability of the state can be further decomposed over all the action choices :

$$\mathbb{P}(s_{t+1}|s_t) = \sum_{a_t} \mathbb{P}(s_{t+1}|a_t, s_t) \mathbb{P}(a_t|s_t) = \sum_{a_t} p_{tr}(s_{t+1}|a_t, s_t) \pi(a_t|s_t)$$

Therefore : $\mathbb{E}(G_{t+1}|s_t) = \sum_{a_t} \pi(a_t|s_t) \sum_{s_t} p_{tr}(s_{t+1}|a_t, s_t) V_\pi(s_{t+1})$

- Gathering both terms and factorizing leads to the recursive expression :

$$V_\pi(s_t) = \sum_{a_t} \pi(a_t|s_t) \left[R(s_t, a_t) + \gamma \sum_{s_{t+1}} p_{tr}(s_{t+1}|a_t, s_t) V_\pi(s_{t+1}) \right]$$

2 Algorithms for Decision-Making

2.1 Dynamic Programming

④ *Characterization of the optimal policy in terms of the distribution* $\pi^*(a|s)$

- The optimal policy is deterministic : in any state, it specifies *the optimal action*, i.e. the action which gives the maximum expected reward.

Formally, for any state s : $\begin{cases} \pi^*(a^*|s) = 1 \text{ for the action } a^* = \arg\max_a Q^*(a, s) \\ \pi^*(a|s) = 0 \text{ otherwise} \end{cases}$

- Thus, the optimal policy can be deduced straight-forward from the state-action value function Q^*.

⑤ *Relation between* $V^*(s)$ *and* $Q^*(a, s)$

- Under the optimal policy, the action taken in any state is deterministic : it is he one which achieves the maximal expected reward in this state, $a^* = \arg\max_a Q^*(a, s)$.

Consequently, the value of a state is equivalent to the value of the optimal action in this state : $V^*(s) = \max_a Q^*(a, s)$.

- Thus, the optimal state-action value function Q^* rewrites :

$$Q^*(a, s) = R(a, s) + \gamma \sum_{s'} p_{tr}(s'|a, s) V^*(s') = R(a, s) + \gamma \sum_{s'} p_{tr}(s'|a, s) \max_{a'} Q^*(a', s')$$

⑥ *Numerical method for finding the optimal policy*

The formula above can expressed with matrices of the following dimensions :

- $\mathbf{Q}$: shape $(|\mathcal{A}|, |\mathcal{S}|)$, with $\mathbf{Q}_{i,j} = Q(a_i, s_j)$
- $\mathbf{R}$: shape $(|\mathcal{A}|, |\mathcal{S}|)$, with $\mathbf{R}_{i,j} = R(a_i, s_j)$
- $\mathbf{P}$: shape $(|\mathcal{S}|, |\mathcal{A}|, |\mathcal{S}|)$, with $\mathbf{P}_{k,i,j} = p_{tr}(s_k|a_i, s_j)$

The optimal state-action value function $\mathbf{Q}^*$ can be viewed as the fixed point of a function f such that :

$$f(\mathbf{Q}_{i,j}) = \mathbf{R}_{i,j} + \gamma \sum_k \mathbf{P}_{k,i,j} \max_l \mathbf{Q}_{l,k}$$

It is possible to use the iterative fixed point method.

2.2 Reinforcement Learning

⑦ *Type of reinforcement learning*

Those two algorithms belong to *model-free* reinforcement learning.

- Model-free agents (MF-RL) directly estimate their subjective values with experience at time t, which is the case here for V (Temporal Difference Learning) and Q (Q-Learning).
- Model-based agents (MB-RL) build models of the expected transitions $\widehat{P}$ and expected rewards $\widehat{R}$, which is not the case in Temporal Difference Learning and Q-Learning.

⑧ *Difference between both strategies*

In TD-Learning, the agent estimates the $|\mathcal{S}|$ values of the state value function V. Each value $V(s)$ is updated whenever the agent happens in state s (whatever the action taken).

In Q-Learning, the agent estimates the $|\mathcal{A}| \times |\mathcal{S}|$ values of the state value function Q. Each $Q(a,s)$ is updated selectively whenever the agent happens in state s **and** chooses the action a.

⑨ *Temporal-Difference Learning*

- Estimation error :

It is defined as the difference between the actual reward r_{t+1} received at this step, and the reward expected by the agent when it is in the current state s_t :

$$\delta_t = r_{t+1} - \widehat{\mathbb{E}}(r_{t+1}|s_t)$$

The *estimated expected reward* in state s_t is, according to question ③ : $\widehat{\mathbb{E}}(r_{t+1}|s_t) = \widehat{\mathbb{E}}(G_t|s_t) - \gamma\widehat{\mathbb{E}}(G_{t+1}|s_t)$

The estimate $\widehat{\mathbb{E}}(G_t|s_t)$ can be assimilated to the value of $V(s_t)$, and the estimate $\widehat{\mathbb{E}}(G_{t+1}|s_t)$ can be assimilated to the value of the new state $V(s_{t+1})$ that the agent has reached (the transitions probabilities are unknown). Therefore :

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$$

- Online update of the state value function :

The update can be made proportional to the error with a learning rate α :

$$V(s_t) \leftarrow V(s_t) + \alpha\left[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)\right]$$

⑩ *Q-Learning*

- Estimation error :

Now, the *estimated expected reward* in state s_t *when the agent performs the action* a_t is, according to question ② : $\widehat{\mathbb{E}}(r_{t+1}|a_t,s_t) = \widehat{\mathbb{E}}(G_t|a_t,s_t) - \gamma\widehat{\mathbb{E}}(G_{t+1}|a_t,s_t)$

The estimate $\widehat{\mathbb{E}}(G_t|a_t,s_t)$ can be assimilated to the value $Q(a_t,s_t)$. The expected return $\widehat{\mathbb{E}}(G_{t+1}|a_t,s_t)$ could be estimated based on the new state s_{t+1} but also the action taken at the next step (not chosen yet). It is conventional to assume the optimal action is taken, therefore :

$$\delta_t = r_t + \gamma \max_{a'} Q(a', s_{t+1}) - Q(a_t, s_t)$$

- Online update of the state-action value function :

$$Q(a_t,s_t) \leftarrow Q(a_t,s_t) + \alpha\delta_t$$

⑪ *Soft-max policy*

'Exploration-exploitation' dilemma :

- In an "exploitation" regime, the agent uses the current information it has gathered so far about the environment to optimize its reward. However, this information might be incomplete due to stochastic sampling, thus it might miss potential better options.
- In an "exploration" regime, the agent tests new solutions and refines its model of the environment. However, the rewards it collects are more random.

The soft-max formula is also found in statistical physics. The parameter β plays the role of an inverse temperature, which switches the balance between exploration and exploitation.

- A high parameter β (low temperature) favors exploitation. The policy distribution $\pi(a|s)$ becomes dominated by the highest exponential terms, corresponding to the action maximizing the expected future rewards.
- A low parameter β (high temperature) favors exploration. All the exponential terms become equivalent, such that the policy distribution $\pi(a|s)$ becomes more uniform.

3 Applications to Maze Learning

3.1 Analytical study – Model-free agent performing Temporal-Difference Learning

12 *Model of the situation*

- States of the environment : $\mathcal{S} = \{-n, ..., -1, 0, 1, ..., n\}$, corresponding to the positions in the corridor, with $-n$ the left end of the maze, n the right end of the maze, and 0 the middle of the maze.
- Actions : $\mathcal{S} = \{R, L\}$, correponding to the displacements in both direction 'right' and 'left'.
- Reward : 1 at the exit, 0 everywhere else.

13 *Value function $V(k)$ (after convergence)*

For any position $k \in [\![-(n-1), (n-1)]\!]$, two actions 'right' and 'left' are possible, each with probability $1/2$ and no reward. Therefore, $V(k) = \frac{1}{2}V(k+1) + \frac{1}{2}V(k-1)$.

At the extreme positions, exploration stops. A reward may or may not be received depending on the side : $V(n) = 1$ and $V(-n) = 0$.

14 *Ansatz for solving the equations*

With the previous relation, the Laplacian cancels : $\Delta V(k) = 0$.

In the continuous limit, the Laplacian would correspond to a second order differential equation. Indeed, replacing k by x and 1 by an increment h :

$$\begin{aligned}\Delta V(x) &= \frac{V(x-h) - 2V(x) + V(x+h)}{h^2} = \frac{\frac{(V(x+h)-V(x))-(V(x)-V(x-h))}{h}}{h} = \frac{\frac{V(x+h)-V(x)}{h} - \frac{V(x)-V(x-h)}{h}}{h} \\ &\approx \frac{V'(x) - V'(x-h)}{h} \approx \partial_x^2 V(x)\end{aligned}$$

The solution of is an affine function of x : $\partial_x^2 V = 0 \implies V(x) = Ax + B$.

By analogy, an ansatz of this form can be proposed : $V(k) = Ak + B$ (it can be checked that is satisfies the equation).

The free constants can be fixed by the boundary conditions :

$$V(-n) = 0 \quad \Rightarrow \quad B = An$$
$$V(n) = 1 \quad \Rightarrow \quad An + B = 2An = 1 \quad \Rightarrow \quad A = \frac{1}{2n}$$

Conclusion : $\boxed{V(k) = \frac{1}{2}\left(1 + \frac{k}{n}\right)}$

15 *Estimated value V during the first learning trial*

On the first trial, all positions are equivalent for the subjective values of the agent. The TD error is $R(k) + \widehat{V}(k \pm 1) - \widehat{V}(k) = 0$, except at the boundaries, such that no updates to $\widehat{V}$ are made until the agent reaches one of the ends of the maze. If the left end is reached, then the trial stops without any update. If the right end is reached, then an update is made at the position $n-1$ using the value of the reward. Progressively, with many essays, V will be updated from position to position, back-propagating from the right end of the maze.

3.2 Simulations – Model-based agent

16 *Python objects/structures*

- Reward function $R(a, s)$ and estimated reward function $\widehat{R}(a, s)$: 2D matrix of dimensions $|\mathcal{A}| \times |\mathcal{S}|$.

- Transition probabilities $p_{tr}(s_{t+1}|a_t, s_t)$ and estimated transition probabilities $\widehat{P}(s_{t+1}|a_t, s_t)$: 3D matrix of dimensions $|\mathcal{S}| \times |\mathcal{A}| \times |\mathcal{S}|$.
- State-action value function $Q(a, s)$: 2D matrix of dimensions $|\mathcal{A}| \times |\mathcal{S}|$.
- Number of events of each (state, action) couple $N(a, s)$: 2D matrix of dimensions $|\mathcal{A}| \times |\mathcal{S}|$.

17 – 24 num Jupyter Notebook