Exercices of Reinforcement Learning from Sutton and Barto

Paul Aubin
paul@eolrobotics.fr

February 2022

1 Chapter 2

Exercice 2.4

$$
\begin{aligned}
Q_{n+1} &= Q_n + \alpha_n[R_n + Q_n] \\
Q_{n+1} &= \alpha_n R_n + \alpha_{n-1}(1-\alpha_n)R_{n-1} + (1-\alpha_n)(1-\alpha_{n-1})\left[(1-\alpha_{n-2})Q_{n-2} + \alpha_{n-2}R_{n-2}\right] \\
Q_{n+1} &= \alpha_n R_n + \alpha_{n-1}(1-\alpha_n)R_{n-1} \\
&\quad + \alpha_{n-2}(1-\alpha_n)(1-\alpha_{n-1})R_{n-2} + (1-\alpha_n)(1-\alpha_{n-1})(1-\alpha_{n-2})Q_{n-2} \\
Q_{n+1} &= \left[\Pi_{i=1}^{n}(1-\alpha_i)\right]Q_1 + \alpha_n R_n + \sum_{i=1}^{n-1}\alpha_i R_i \Pi_{k=i}^{n-1}(1-\alpha_{k+1}) \qquad (1)
\end{aligned}
$$

Exercice 2.5 Programming

Simulation for 10 000 steps averaged over 1000 runs with an independent random walk of the mean agent value.

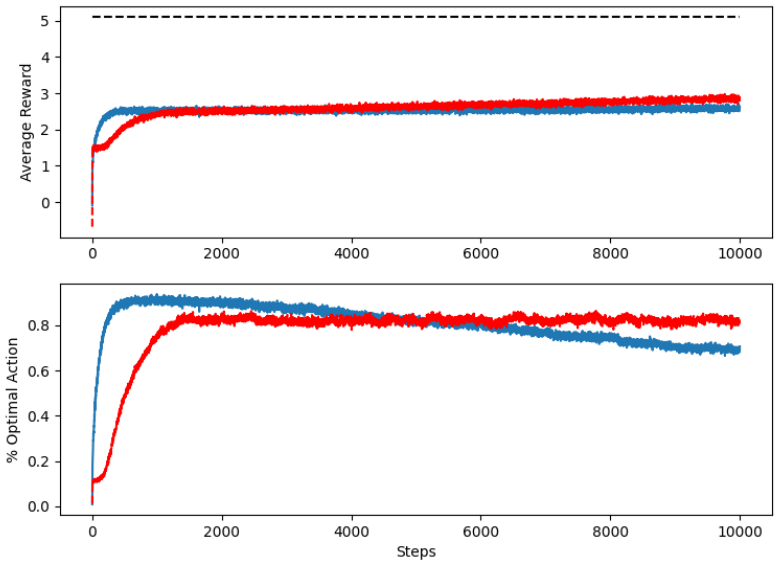

Figure 1: The blue line represents an average of 1000 runs with a sample averaged action-value method and the red line represents an average of 1000 runs with a constant parameter action-value method on a non-stationnary bandit problem

We can see on the long run the better performance of the constant parameter action-value method compared to the sample averaged action-value method, both in term of optimal action and in term of average reward

Exercice 2.6 Mysterious Spikes

Let's first assume that the average is strong enough to consider that the reward corresponds to the mean value of the true action value.

First let's consider the early steps of a realistic ϵ-greedy with constant step-size parameter method. At the very first steps it can either pick a positive value or a negative value. If it picks a negative value it will keep exploring until it picks a positive value. Once it picks a positive value it will play it again and again at a rate of $(1 - \epsilon)$ to start maximizing the gains. It may not be the best pick, but at least it is likely a good one. So it will start strong. Also, it will need time to find the best pick.

Let's consider the early steps of an optimistic ϵ-greedy with constant step-size parameter method in comparison. At the very first steps it will pick each pos-

sible action at least once. Since the parameter is constant, the updated value will be stronger on the bad actions than on the good ones. For instance, if there were only 2 values : +2 and -2, then the +2 action would be updated to $5 - \alpha(2 - 5) = 5 - 0.3 = 4.7$ whereas the bad action would be updated to $5 - \alpha(-2 - 5) = 5 - 0.7 = 4.3$. The best action is then ranked with the best score after a first update of all the actions, so the algorithm will choose it again (at a probability of $(1 - \epsilon)$) on the next step.
This effect lasts until the estimated value of the best action-value drops below the other estimated action-values, and then it repeats, creating a damped periodicity, observed by the spikes.

Exercice 2.7 Unbiased Constant Step-Size Trick

Let's start from equation 1 :

$$Q_{n+1} = \left[\Pi_{i=1}^{n}(1 - \beta_i)\right] Q_1 + \beta_n R_n + \sum_{i=1}^{n-1} \beta_i R_i \Pi_{k=i}^{n-1}(1 - \beta_{k+1})$$

Consider the first term of the equation that accounts for the initial bias :

$$\Pi_{i=1}^{n}(1 - \beta_i)Q_1 \tag{2}$$

Then, let's calculate β_1 :

$$\begin{aligned} & o_0 = 0 \\ \Rightarrow & o_1 = o_0 + \alpha(1 - o_0) = \alpha \\ \Rightarrow & \beta_1 = \frac{\alpha}{o_1} = \frac{\alpha}{\alpha} = 1 \end{aligned}$$

Then equation 2 can be calculated :

$$\Pi_{i=1}^{n}(1 - \beta_i)Q_1 = (1 - 1)\Pi_{i=2}^{n}(1 - \beta_i)Q_1 = 0 \tag{3}$$

Let's propagate the result to equation 1 :

$$Q_{n+1} = \beta_n R_n + \sum_{i=1}^{n-1} \beta_i R_i \Pi_{k=i}^{n-1}(1 - \beta_{k+1})$$

The initial bias has been removed

Exercice 2.8 UCB Spikes

In the formula

$$A_t = \underset{a}{argmax} \left[Q_t(a) + c\sqrt{\frac{lnt}{N_t(a)}} \right] \tag{4}$$

For any given $N_t(a) = 0$, A_t will tend to positive infinity. This ensures a first try on every action at the first steps. Since there are 10 actions, the 10 first moves are then exploratory, which explains the flatness at the beginning. Once all actions have been tried, equation 1 gives :

$$A_{11} = \underset{a}{argmax} \left[Q_{11}(a) + c\sqrt{ln11} \right]$$

Where $\sqrt{ln11} \approx 1.55$

We can consider that the random walk is too slow to have generated a strong drift of the different mean action values. Also since the curve is averaged over 2000 samples, we can consider that on average, a good reward on the first 10 steps corresponds to a good action. On the 11th step, since the exploratory term $c\sqrt{\frac{lnt}{N_t(a)}}$ is constant for all actions, the algorithm will then pick the action with the highest estimated action-value. It is a good action, which gives a high reward).

On the 12th step it is a bit different. If c tends to 0 then there is no exploration : the algorithm will keep making the same good action, but probably not the best. If c tends to infinity there is no exploitation and only exploration, which would lead to poor rewards and a high drop. In both cases the curve goes flat. Either constant at the peak, or constant around 0.
When c = 2, the value of the highest estimated action-value is lowered by $c(\sqrt{ln12} - \sqrt{ln(12)/2}) \approx 0.92$ compared to other actions. Since the mean action-value are distributed normally with a standard deviation of 1, this gives a high probability to select other actions for the following steps, with lower rewards. This is why there is a drop after the peak

Exercice 2.9

$$Pr\{A_t = a\} = \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}}$$

If we consider only two actions,

$$Pr\{A_t = a\} = \frac{e^{H_t(a)}}{e^{H_t(a)} + e^{H_t(b)}}$$
$$Pr\{A_t = a\} = \frac{1}{1 + e^{-(H_t(a) - H_t(b))}}$$

Which is the sigmoid function for the variable $H_t(a) - H_t(b)$

Exercice 2.10

When facing a 2 armed-bandit task (arm A and arm B), whose true action value change randomly from time step to time step with a probability of 0.5,

then there is no strategy available to decide which action value should be favored.

	arm A	arm B	probability
case A	10	20	0.5
case B	90	80	0.5

Table 1: 2 armed-bandit task with case A and B

As represented in table 1, when facing case A one could pick on average either 10 or 20, when facing case B one could pick on average either 90 or 80. Let's assume that one always pick arm A, the averaged reward is 50. Let's assume that one always pick arm B, the average reward is also 50. Let's assume that one picks arm A or arm B with equal probability, the average reward remains 50. This is because there is equal probability of being in case A or B and no prior information to know which case we are facing. So the best expected reward is 50, there is no better strategy to increase the average reward, but selecting arm B will minimize the variance of the reward, which is still interesting.

If we have a way to know in which case we are, then in case A the best action would be to select arm B, in case B it would be to select arm A. That would lead to an average reward of 55. It is higher than the average reward of 50 when the information on the current case is not exploited.
To achieve this reward we should train two reinforcement learning algorithms, one for each case. They should have an action-value method that converges to the true action-value, such as sample averages action-value method. Once trained we should only perform exploitation with each algorithm corresponding to each case.