Chapter 4

Project 3: An introduction to reinforcement learning I

4.1 Introduction

Reinforcement learning is a learning paradigm that states that one learns by interacting with the environment in which one is in, rather than by being provided explicit rules to follow. Specifically, reinforcement learning states that an *agent* (e.g. a human or a robot) learns which actions to take in the *environment* so as to maximize a specific *reward* signal. That is, the agent initially has no information about the set of desirable actions to take and only learns them gradually by interacting with its environment, which will periodically issue rewards to the agent. Reinforcement learning is fundamentally different from supervised learning where the agent would be explicitly taught which behaviour to take [6].

Thus, to cast a problem as a reinforcement learning problem one needs to define an environment, an agent acting in this environment and a reward signal. Among the many particular problems that can be cast as a reinforcement learning problem is learning to drive a car. Here the environment is a network of roads, the agent is a human or a robot (such as a self-driving vehicle), the set of actions are to accelerate forward and backward, stopping, turning left and right, and the reward signal could be defined as the time that has passed without crashing the car. Each action taken by the agent brings the agent from one state of the environment to another state of the environment. For driving a car, say, your initial state is the parking spot in front of a house (which belongs to the network of roads) and once you start to drive (i.e. accelerate and turn left or right) you find yourself in another state of the environment (another part of the network of roads). You maximize your reward by simply not crashing your car.

While the concept is astonishingly simple, reinforcement learning (and its combination with deep neural networks, the so-called deep reinforcement learning) is responsible for most of the recent breakthroughs in artificial intelligence, such as self-driving cars and computers beating humans in the games of Chess and Go [5] as well as in several video games such as the Atari games, StarCraft II or Dota II [4]. What makes reinforcement learning superior to other learning approaches is that we do not teach the agent which particular rules to follow. In the case of (video) games, we do not explicitly teach the agent which behaviour we think is best. The agent learns only by maximizing its expected reward (e.g. how to win the game as fast as possible). This has led to surprising new strategies learned by the various agents for solving problems, which were previously unknown to humans. In fact, reinforcement learning is currently believed to be our best shot at realizing true artificial intelligence.

Remark 4. This entire course can be regarded as an example for reinforcement learning. You are provided with which results are expected for each project, and your task is to figure out how to obtain these results using trial-and-error. Doing so as best as possible then maximizes your cumulative reward (your final grade on this course).

4.2 Background

In this first part of the introduction to reinforcement learning, we consider the problem of a *multi-armed bandit*: We are presented with k slot machines, each of which produces a particular numerical reward. The problem is that we do not know ahead of time which slot machine (or bandit) pays the most reward, which is the bandit we should be playing all the time. This setting simplifies the reinforcement learning problem in that there is only a single state of the environment, and we aim to learn which action, i.e. which bandit to play, will yield the highest payout in the long run.

Each of the k actions, i.e. each of the k bandits we can be playing, has an expected (mean) reward given that this action has been selected. We denote this as the *value* of that action. The action selected at time step t is denoted by A_t, with the corresponding obtained reward being denoted as R_t. The value $q_*(a)$ of the action a is the expected reward provided that we have selected action a at time step t,

$$q_*(a) = \mathbb{E}(R_t | A_t = a).$$

The above equation reads that the value of the action a is obtained as the expected value of the reward R_t given that we select action a at step t.

The goal of the reinforcement learning problem is to learn the true value $q_*(a)$, i.e. to obtain as good an estimate $Q_t(a)$ for the true value $q_*(a)$ of the action a as possible, having played the bandits $t-1$ times. The following example provides some clarification.

Example 1. Assume that there is only one bandit, i.e. $k = 1$. You play the bandit 9 times and obtain the following rewards:

$$\begin{aligned} &R_1 = 0.79110577, \quad R_2 = 1.58662319, \quad R_3 = 1.83898341, \\ &R_4 = 1.93110208, \quad R_5 = 1.28558733, \quad R_6 = 1.88514116, \\ &R_7 = 0.24560206, \quad R_8 = 2.25286816, \quad R_9 = 1.51292982. \end{aligned}$$

The mean reward after playing this bandit 9 times is then

$$Q_{10} = \frac{1}{9}(R_1 + R_2 + \cdots + R_9) = \frac{1}{9}\sum_{i=1}^{9} R_i \approx 1.4811.$$

Note that we call this value estimate Q_{10} instead of Q_9 since this is the expected reward we will receive when playing the bandit a 10th time. Thus, after playing the bandit 9 times, we estimate the value of the single action $a = 1$ (we only have one bandit to play so $a = 1$ means playing bandit number 1) is

$$q_*(1) \approx Q_{10}(1).$$

The more often we play the bandit, the more accurate this estimate will become.

Example 2. If there is only one bandit, there is no learning problem since there is no choice to be made. Now consider that there are two bandits. You play the first bandit at $t = 1$ and obtain

$$R_1 = 2.12948391.$$

You then try the second bandit at $t = 2$ and obtain

$$R_2 = 1.36126959.$$

Disappointed with the lower payout of the second bandit, you return to the first bandit and exclusively play the first bandit afterwards. Is this a smart choice?

The previous example is known as the *exploration–exploitation* problem. Exploitation refers to using your current knowledge to choose the action that maximizes your reward over a single step. Exploration in turn refers to choosing an action that gives you a lower reward over a single step but may lead to a higher reward in the long run.

Example 3. Continuing Example 2, assume we first play the first bandit ($a = 1$) 9 times and obtain the following rewards:

$$\begin{aligned} R_1 &= 2.12948391, \quad R_2 = 0.79924193, \quad R_3 = 1.18656139, \\ R_4 &= 1.41005165, \quad R_5 = 1.19829972, \quad R_6 = 1.11900865, \\ R_7 &= 0.32933771, \quad R_8 = 1.37756379, \quad R_9 = 1.12182127. \end{aligned}$$

This allows us to estimate the value of the action $a = 1$ as

$$q_*(1) \approx Q_{10}(1) = \frac{1}{9} \sum_{i=1}^{9} R_i \approx 1.1857.$$

Now assume we instead played the second bandit ($a = 2$) 9 times, where we obtain the following rewards

$$\begin{aligned} R_1 &= 1.36126959, \quad R_2 = 3.19891788, \quad R_3 = 2.18515642, \\ R_4 &= 1.62471505, \quad R_5 = 2.42349435, \quad R_6 = 2.07734007, \\ R_7 &= 1.65614632, \quad R_8 = 2.04359686, \quad R_9 = 1.37999916. \end{aligned}$$

We can thus estimate the value of the action $a = 2$ as

$$q_*(2) \approx Q_{10}(2) = \frac{1}{9} \sum_{i=1}^{9} R_i \approx 1.9945.$$

In other words, the second bandit has almost twice the mean payout as the first bandit. We were just unlucky that the first time we played each bandit, that the first one gave more reward than the second one. Over a total of 9 plays each, the mean payout per play if of course much better for the second bandit.

Thus, had we followed a strict exploitation strategy after having played each bandit only once, we would have always played bandit 1. Exploration over a period of time allowed us to discover that bandit 2 actually seems more promising than bandit 2, regardless of the disappointing performance after a single play. Going forward, based on the information we have gathered over 18 plays it thus would make more sense to exclusively play the second bandit. But is this the right strategy?

The main problem with our two-armed bandit problem is that after 18 plays we still do not know which bandit pays out the most in the long run. Maybe after 100 plays the payouts will be again different, this time favouring once again the first bandit. The problem is that you'd have to play each bandit an infinite number of times to find the true value of each action. Playing a "game" infinitely many times is impossible for real-world reinforcement learning problems.

While the exploration–exploitation problem seems relatively simple, it is important to note that it has not been solved for general reinforcement learning problems. Whether you notice or not, the exploration–exploitation problem is a constant companion in your day-to-day life.

Example 4. Suppose you move to a new city and spend the first few weeks exploring different restaurants. Having tried 5 different restaurants once, you then pick your favourite 2 restaurants and go to them exclusively in the future. How can you be sure that you really found the 2 best restaurants in town (or, for that matter, even the 2 best among the 5 you tried)?

Example 5. You have to pick a section for your Calculus I course. You browse through the student evaluations of the 3 instructors that are offering the course and find the following numerical ratings (with 5 being the highest):

1. Instructor A: 4.2 (from 3 evaluations)
2. Instructor B: 4.0 (from 10 evaluations)
3. Instructor C: 5.0 (from 1 evaluation)

Exploitation would compel you to choose Instructor C, since this one gives the highest reward (a perfect rating of 5), but there was only one student rating for this instructor. In other words, this could have been a lucky (for the Instructor) outlier. Instructor A has still a quite good score but 3 evaluations is again rather little and if your experience would be negative and you would finally add this to the score, Instructor A would drop below Instructor B (as would Instructor C). So should you choose Instructor B, which has the lowest overall score but this score being overall the most established?

Regardless of your problem, at each step t you can maintain an estimate of all the action values, which will become more and more accurate the larger t becomes. Then there is at any time step at least one action that would give you the highest estimated value. We call this the *greedy action*. Exploitation means that you always choose the greedy action. Exploration refers to choosing a nongreedy action.

A main challenge balancing exploration and exploitation is that most real-world reinforcement learning problems are not stationary. In our bandit problem, non-stationary would mean that over time the mean payout of each bandit changes. Since the environment is constantly changing, maintaining some level of exploration is usually advantageous.

One simple way of balancing exploration and exploitation is to use an ϵ-greedy strategy. Suppose we set $\epsilon = 0.1$ this would mean that from 10 actions we choose 9 times the greedy action and 1 time randomly any of the other actions available; in other words, we explore 10% of the time and exploit the remaining 90% of the time. The larger ϵ, the larger the rate of exploration will be. If we set $\epsilon = 1$ then we will explore all the time and never exploit. If we set $\epsilon = 0$ then we are always using the greedy action, i.e. we exploit all the time and never explore.

Remark 5. For the single bandit case we estimated the value of the single action $a = 1$ at time step $t = n$ by forming the arithmetic mean of all the rewards R_i obtained up to time $t = n - 1$,

i.e.

$$Q_n = \frac{1}{n-1}\sum_{i=1}^{n-1} R_i.$$

If we play the bandit one more time, i.e. moving to $t = n+1$ we get one additional reward R_n. To compute the new estimate Q_{n+1} it is luckily not required to compute

$$Q_{n+1} = \frac{1}{n}\sum_{i=1}^{n} R_i.$$

Instead, we note that

$$\begin{aligned} Q_{n+1} &= \frac{1}{n}\sum_{i=1}^{n} R_i = \frac{1}{n}\left(R_n + \sum_{i=1}^{n-1} R_i\right) \\ &= \frac{1}{n}\left(R_n + (n-1)\frac{1}{n-1}\sum_{i=1}^{n-1} R_i\right) \\ &= \frac{1}{n}\left(R_n + (n-1)Q_n\right) = \frac{1}{n}(R_n + nQ_n - Q_n) \\ &= Q_n + \frac{1}{n}(R_n - Q_n). \end{aligned}$$

In other words, having computed Q_n and after having received the new reward R_n, we simply compute Q_{n+1} by

$$Q_{n+1} = Q_n + \frac{1}{n}(R_n - Q_n), \tag{4.1}$$

which is much quicker to compute than having to evaluate the sum over all past rewards again and again.

Remark 6. Note that Eq. (4.1) has to be adapted to the k-armed bandit case! In particular, a k-dimensional counter array n has to be maintained, with the value $n(a)$ being increased by one if action a is selected at step t.

4.3 Project

Here we assess the effectiveness of the greedy and ϵ-greedy strategies for a 10-armed bandit problem.

4.3.1 Epsilon-greedy strategy

Write a `Python` routine that sets the true action values $q_*(a)$, $a = 1, \ldots, 10$ for each bandit as taken randomly from a normal distribution with mean 0 and variance 1. This will give you one specific bandit configuration, see Fig. 4.1 for an example. Set $Q_1(a)$, i.e. your estimate for the value of each action a at the first time step $t = 1$ (i.e. before any action was taken) to zero for all a.

Run this bandit problem for a total of 1000 steps using the greedy and ϵ-greedy strategies for various values of ϵ, such as $\epsilon = 0.01$, $\epsilon = 0.1$ and $\epsilon = 0.2$. Each learning method will select a particular action A_t (i.e. a particular bandit) at step t, which will give you a particular reward R_t, which we choose to be selected from a normal distribution with mean $q_*(A_t)$ and

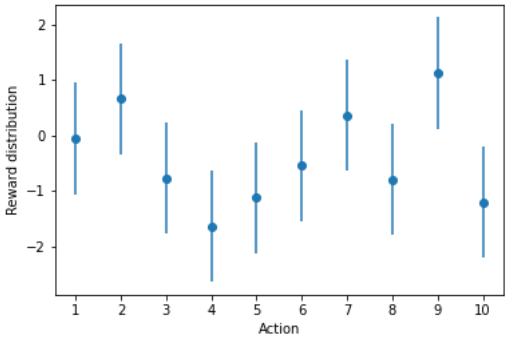

Figure 4.1: One particular bandit problem, with mean reward $q_*(a)$ distribution selected as `qstar = np.random.rand(10)`. The actual reward at each step t is then selected from a normal distribution with mean $q_*(A_t)$ and variance 1.

variance 1. In other words, the mean reward you will obtain from each bandit is indeed the mean true reward $q_*(A_t)$, but there will be some variance to this reward as well. You can then track the obtained rewards over all of the 1000 steps. This will give you the performance of these strategies for that particular bandit problem.

To assess the mean performance of each strategy, repeat this experiment for 2000 randomly generated 10-armed bandits. That is, select 2000 different 10-armed bandits (with true action values $q_*(a)$, $a = 1, \ldots, 10$ for each bandit selected from a normal Gaussian distribution with mean 0 and variance 1) and then run each 10-armed bandit problem for 1000 steps, picking the reward R_t at step t for taking action A_t from the associated normal distribution with mean $q_*(A_t)$ and variance 1. Then average the reward R_t as a function of t over all 2000 bandit problems. To give you some idea of the results you may obtain, Fig. 4.2 shows the mean performance of the greedy action selection strategy as a function of the time step.

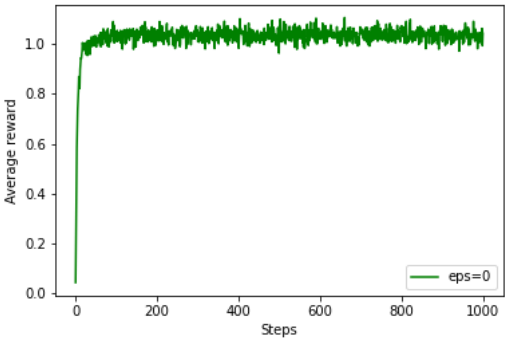

Figure 4.2: Average performance of the greedy action ($\epsilon = 0$) selection strategy over 2000 different realizations of a 10-armed bandit problem. Reproduce this result and also add the respective curves for $\epsilon = 0.01$, $\epsilon = 0.1$ and $\epsilon = 0.2$.

Since in this particular case we have access to the true reward distribution $q_*(a)$, we also know which bandit is the optimal for each 10-armed bandit problem, namely the one giving the highest mean reward. In the particular bandit problem depicted in Fig. 4.1 this would be bandit 9. You can thus assess the percentage of time each action selection strategy gave you the optimal action, i.e. whether the learning algorithm has learned which bandit is the most

profitable. An example of the results you may obtain is depicted in Fig. 4.3 for the greedy action selection strategy.

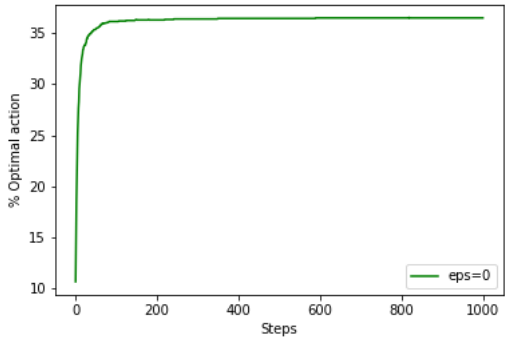

Figure 4.3: Mean percentage of times the greedy action ($\epsilon = 0$) selection strategy found the optimal action over 2000 different realizations of a 10-armed bandit problem. Reproduce this result and also add the respective curves for $\epsilon = 0.01$, $\epsilon = 0.1$ and $\epsilon = 0.2$.

Interpret all of the results you obtain!

4.3.2 Optimistic initial values

In the previous experiment we have assumed that the initial estimate $Q_1(a)$, before playing any bandit at all, for all action values is zero. Indeed, in the present case we choose our bandit reward distribution from a normal distribution with zero mean and unit variance, thus making the initial assumption for $Q_1(a)$ realistic. However, without prior knowledge of the true reward distribution (which for a real-world bandit problem we just would not have), the particular choice of the initial value estimates injects some assumed prior knowledge of the action values that is not always justified.

To assess the influence of the initial value estimate $Q_1(a)$ on the long-term reward distribution, repeat the above experiments by setting $Q_1(a) = 5$. We refer to this as *optimistic initial values*. Indeed this value is extremely optimistic since we choose $q_*(a)$ from a normal distribution with zero mean and unit variance. For reference, also try these experiments with $Q_1(a) = -5$, which would be an extremely pessimistic initial guess for the present setting.

Can you explain these results in terms of how the initial value estimates could encourage the agent to explore its environment?

Remark 7. The results more generally underpin a mindset which may be summarized as *optimistic in the face of uncertainty*: Faced with an uncertain situation, it is better to be optimistic rather than realistic (or even pessimistic).

4.4 Why this project is relevant

While a vastly simplified form of the full reinforcement learning problem, the multi-armed bandit problem introduced important concepts such as the exploration–exploitation dilemma and the need for defining an explicit learning rule for how the agent interacts with its environment. The next project expands on these concepts in substantial ways.