Chapter 2 - Solutions

Alessio Caciagli

March 2021

1 Exercise 2.1

In ϵ-greedy action selection, for the case of two actions and $\epsilon = 0.5$, what is the probability that the greedy action is selected?

In ϵ-greedy action selection, greedy action is *always* selected with probability (1-ϵ). On the other side, with probability ϵ a random selection from all the actions is performed with equal probability i.e. we assume a uniform distribution for the probability of selecting a certain action a. These two cases are of course mutually exclusive. In formal terms, assuming N actions are available, the probability of selecting the best action is:

$$p(x) = (1 - \epsilon) + \frac{\epsilon}{N} \tag{1}$$

The second term arises since $p(x = k) = 1/N$ for a uniform distribution. With $N = 2$ and $\epsilon = 0.5$, the result is $p(x) = 0.5 + 0.5 * 0.5 = 0.75$. $\square$

2 Exercise 2.2

Bandit Example *Consider a k-armed bandit problem with $k = 4$ actions, denoted 1, 2 3 and 4. Consider applying to this problem a bandit algorithm using ϵ-greedy action selection, sample-average action-value estimates, and initial estimates for $Q_1(a) = 0$, for all a. Suppose the initial sequence of actions and rewards is $A_1 = 1$, $R_1 = 1$, $A_2 = 2$, $R_2 = 1$, $A_3 = 2$, $R_3 = 2$, $A_4 = 2$, $R_4 = 2$, $A_5 = 3$, $R_5 = 0$. On some of these time steps the ϵ case may have occurred, causing an action to be selected at random. On which time steps did this definitely occur? On which time steps could this possibly have occurred?*

Let's tabulate the mean rewards, $Q_t(a)$, at each time step:

Time	$Q_t(1)$	$Q_t(2)$	$Q_t(3)$	$Q_t(4)$
$t = 0$	0	0	0	0
$t = 1$	1	0	0	0
$t = 2$	1	1	0	0
$t = 3$	1	3/2	0	0
$t = 4$	1	5/3	0	0
$t = 5$	1	5/3	0	0

Given this, we can reconstruct whether the ϵ case might have occurred at each time step:

- At $t = 1$, either ϵ or greedy selection might have occurred (because all actions are greedy at the start)
- At $t = 2$, ϵ selection has occurred (the greedy action is 1)
- At $t = 3$, either ϵ or greedy selection might have occurred (because both actions 0 and 1 are greedy)
- At $t = 4$, either ϵ or greedy selection might have occurred (although the greedy action 2 has been selected)
- At $t = 5$, ϵ selection has occurred (the greedy action is 2)

$\square$

3 Exercise 2.3

In the comparison shown in Figure 2.2, which method will perform best in the long run in terms of cumulative reward and probability of selecting the best action? How much better will it be? Express your answer quantitatively.

Assuming both methods have converged so that the optimal action corresponds to the greedy action (which is reasonable in the limit $t \to \infty$), we have the following by applying Equation 1:

- For the $\epsilon = 0.1$ case, the probability of optimal action selection is $p(x) = 0.9 + 0.1 * 0.1 = 0.91$
- For the $\epsilon = 0.01$ case, the probability of optimal action selection is $p(x) = 0.99 + 0.01 * 0.1 = 0.991$

Since we can disregard the transient in the long run, the method with the highest probability of optimal action selection will also yield the highest cumulative reward. Hence, method $\epsilon = 0.01$ will perform best. $\square$

4 Exercise 2.4

If the step-size parameters α_n are not constant, then the estimate Q_n is a weighted average of previously received rewards with a weighting different form that given by (2.6). What is the weighting on each prior reward for the general case, analogous to (2.6), in terms of the sequence of step-size parameters?

The estimate Q_{n+1} is given by the general formula:

$$Q_{n+1} = Q_n + \alpha_n (R_n - Q_n) \tag{2}$$

We can expand recursively, such that, for the first two expansions:

$$\begin{aligned}
Q_{n+1} &= \alpha_n R_n + (1-\alpha_n)Q_n \\
&= \alpha_n R_n + (1-\alpha_n)[Q_{n-1} + \alpha_{n-1}(R_{n-1} - Q_{n-1})] \\
&= \alpha_n R_n + (1-\alpha_n)[\alpha_{n-1}R_{n-1} + (1-\alpha_{n-1})Q_{n-1}] \\
&= \alpha_n R_n + (1-\alpha_n)\alpha_{n-1}R_{n-1} + (1-\alpha_n)(1-\alpha_{n-1})Q_{n-1}
\end{aligned}$$

We can refactor it in the final form:

$$Q_{n+1} = \left(\prod_{i=1}^{n}(1-\alpha_i)\right) Q_1 + \sum_{i=1}^{n} \alpha_i R_i \prod_{j=i+1}^{n} (1-\alpha_j) \tag{3}$$

As a check, we can assume a stationary problem with weights $\alpha_i = 1/i$. In this case, the first term of Equation 3 is zero, and the second term becomes:

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

which is the estimate of the action value for a stationary problem. Hence, Equation 3 is the estimate of the action value for the general case of step-size parameters α_n (either stationary or non-stationary). $\square$

5 Exercise 2.5

Design and conduct an experiment to demonstrate the difficulties that sample-average methods have for non-stationary problems. Use a modified version of the 10-armed testbed in which all the $q_*(a)$ *start out equal and then take independent random walks (say by adding a normally distributed increment with mean zero and standard deviation 0.01 to all the* $q_*(a)$ *on each step). Prepare plots like Figure 2.2 for an action-value method using sample averages, incrementally computed, and another action-value method using a constant step-size parameter,* $\alpha = 0.1$. *Use* $\epsilon = 0.1$ *and longer runs, say of 10000 steps.*

The results of the experiment are presented in Figure 1. See the companion *code* folder (for code and notebooks) for the implementation.

The non-stationary nature of the problem lead to slow convergence for both action-value methods employed, denoting the added difficulty that sample-average methods experience for non-stationary problems. The constant step-size method is superior to the incremental sample average method. Besides the higher average reward cumulated at the end of the run, the constant step-size

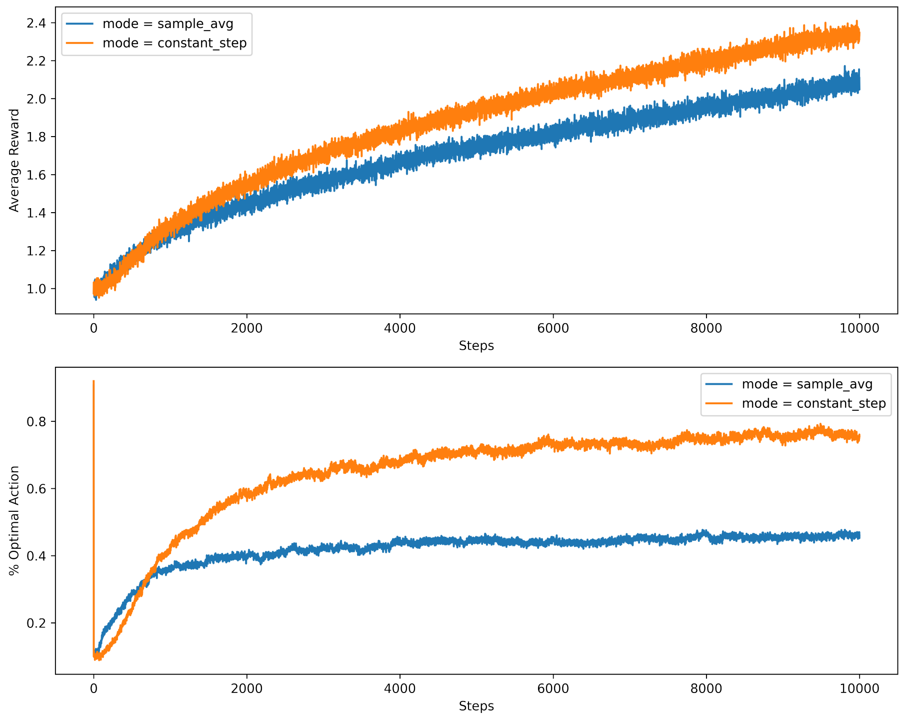

Figure 1: Average performance of ϵ-greedy action value methods on the 10-armed testbed for a non-stationary problem. These data are averages over 2000 runs with different bandit problems. The *sample average* method employs incrementally computed step-size parameters. The *constant step* method, on the contrary, employs a constant step-size parameter.

method has a optimal action selection probability of around 80% at convergence, while the incrementally computed sample average method plateaus at only 40%. The initial drop in the optimal action plot is due to the fact that, at the beginning, every action is optimal (they all start from the same initial value).
Note that the average reward does not plateau, but it increases in time. This is a consequence of the random walk the expected rewards perform. For a discrete random walk, $\langle r^2(t) \rangle \propto t$ where r^2 is the *mean square displacement* from the initial position. Due to the stochastic nature of the random walks performed by the expected rewards, we can on average expect that at least one of the arms exhibits a significant positive drift in its random walk. This action will be the optimal action selected by the bandit algorithm. For our case, we can therefore expect that the average reward exhibits a power-law behavior $Q_*(a) \propto t^\alpha$, with $0 < \alpha < 1$.

6 Exercise 2.6

Mysterious spikes *The results shown in Figure 2.3 should be quite reliable because they are averages over 2000 individual, randomly chosen 10-armed bandit tasks. Why, then, are there oscillations and spikes in the early part of the curve for the optimistic method? In other words, what might make this method perform particularly better or worse, on average, on particularly early steps?*

The oscillations in the early steps are caused by the "relaxation" of the initial optimistic values for each action. Each action will be considered optimal before it is sampled the first time. Afterwards, it will require some relaxation steps before the sample average for each sub-optimal action (incrementally computed) converges below the optimal action's one. This occurs also in the non-optimistic case. However, with a non-optimistic initialization, the convergence is smooth as a consequence of the exploration policy (i.e., the optimal action is initially only selected during exploratory moves on average, because all the action values start from 0). On the other side, with an optimistic initialization and greedy selection, after k steps every action has been sampled once. On the $k+1$-step, therefore, optimal action selection occurs in roughly 42% of the cases since the first sampling of each action yields a normally distributed reward centered of the action value. This is the cause of the first spike. The second spike arises following a similar explanation, but this time after $2*(k+1)$ steps. Subsequent spikes are less and less prominent until convergence is reached and the evolution of the optimal action curve becomes smooth.

7 Exercise 2.7

Unbiased constant-step size trick *In most of this chapter we have used sample averages to estimate action values because sample averages do not produce the initial bias that constant step sizes do (see the analysis in (2.6)). However, sample averages are not a completely satisfactory solution because they may perform poorly on non-stationary problems. Is it possible to avoid the bias of constant step-sizes while retaining their advantages on non-stationary problems? One way is to use a step size of:*

$$\beta_n \coloneqq \alpha/\bar{o}_n \tag{4}$$

to process the n^{th} reward for a particular action, where $\alpha > 0$ is a conventional constant step size and $\bar{o}_n$ is a trace of one that starts at 0:

$$\begin{aligned} &\bar{o}_n \coloneqq \bar{o}_{n-1} + \alpha(1 - \bar{o}_{n-1}) \quad \text{for}\, n > 0 \\ &\bar{o}_0 \coloneqq 0 \end{aligned} \tag{5}$$

Carry out an analysis like that in (2.6) to show that Q_n is an exponential recency-weighted average without initial bias.

We can substitute the sequence α_n in Equation 3 with $\beta_n = \alpha/\bar{o}_n$. The initial bias term is:

$$\begin{aligned}\prod_{i=1}^{n}(1-\alpha_i) &= \prod_{i=1}^{n}\left(1-\frac{\alpha}{\bar{o}_i}\right)\\ &= \prod_{i=1}^{n}\left(1-\frac{\alpha}{\bar{o}_{i-1}+\alpha(1-\bar{o}_{i-1})}\right)\\ &= \prod_{i=1}^{n}\left(\frac{\bar{o}_{i-1}(1-\alpha)}{\bar{o}_{i-1}+\alpha(1-\bar{o}_{i-1})}\right) = 0\end{aligned}$$

following from $\bar{o}_0 \coloneqq 0$. $\square$

8 Exercise 2.8

UCB Spikes *In Figure 2.4 the UCB algorithm shows a distinct spike in performance on the 11th step. Why is this? note that for your answer to be fully satisfactory it must explain both why the reward increases on the 11th step and why it decreases on the subsequent steps. Hint: if $c = 1$, the spike is less prominent.*

The answer is similar to that of Exercise 2.6. During the first k steps, all actions are tried once since each action will be considered optimal before it is sampled the first time. The first sampling of each action yields a normally distributed reward centered of the action value. In 42% of the cases, therefore, the optimal action is going to exhibit the highest estimated reward after one sampling. This is the cause of the peak on the $k+1$-step. Since the c parameter controls the degree of exploration (with higher values meaning more exploration), for higher c values the subsequent steps will again tend to favour sampling of all the moves, leading to a decrease in the average reward until convergence is reached. For $c = 1$ (and in general for low c values), the peak in the average reward curve at the $k+1$-step is still present and at the same value, but the drop in the subsequent steps is less prominent since action selection will favour exploitation over exploration.

9 Exercise 2.9

Show that in the case of two actions, the soft-max distribution is the same as that given by the logistic, or sigmoid, function often used in statistics and artificial neural networks.

In the case of two actions a and b, we can rewrite $\pi_t(a)$ as follows:

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

without loss of generality. Since only the relative preference of action a over action b is important (and not the absolute value of the preference function), we can denote $x = H_t(a) - H_t(b)$. We then obtain:

$$\pi_t(a) = \frac{1}{1+e^{-x}} = S(x) \tag{6}$$

which is the functional form of the sigmoid function. It's trivial to verify:

$$\pi_t(a) = S(x) = 1 - S(-x) = 1 - \pi_t(b) \quad \square$$

10 Exercise 2.10

Suppose you face a 2-armed bandit task whose true action values change randomly from time step to time step. Specifically, suppose that, for any time step, the true values of actions 1 and 2 are respectively 0.1 and 0.2 with probability 0.5 (case A), and 0.9 and 0.8 with probability 0.5 (case B). If you are not able to tell which case you face at any step, what is the best expectation of success you can achieve and how should you behave to achieve it? Now suppose that on each step you are told whether you are facing case A or case B (although you still don't know the true action values). This is an associative search task. What is the best expectation of success you can achieve in this task, and how should you behave to achieve it?

In the first case, let us denote the probability of selecting action 1 with p_1. Of course, $p_2 = 1 - p_1$. We have $q_t(1) = 0.5 * 0.1 + 0.5 * 0.9 = 0.5 = q_t(2)$ at each time step. The expected reward is:

$$p_1 * q_t(1) + (1 - p_1) * q_t(2) = 0.5$$

irrespective of p_1. Hence, any strategy will yield the same expected reward (from always selecting the same action to randomly selecting an action at each time step).
In the second case, we should now swap our probability computation. For scenarios A and B, the expected reward is:

$$\begin{aligned} p_1 * 0.1 + (1-p_1) * 0.2 &= 0.2 - 0.1 * p_1 && (\text{case A}) \\ p_1 * 0.9 + (1-p_1) * 0.8 &= 0.8 + 0.1 * p_1 && (\text{case B}) \end{aligned}$$

We can maximise the expected reward by having $p_1 = 0$ for case A, and $p_1 = 1$ for case B. Hence, the best expectation of success is $0.5 * 0.2 + 0.5 * 0.9 = 0.55$. In order to achieve it, we should set two separate agents for the two cases - any method is suitable for this task. If the rewards have small variance, greedy with optimistic initial values might lead to faster convergence and highest cumulative rewards.

11 Exercise 2.11

Make a figure analogous to Figure 2.6 for the nonstationary case outlined in Exercise 2.5. Include the constant-step-size ϵ-greedy algorithm with $\alpha = 0.1$.

Use runs of 200000 steps and, as a performance measure for each algorithm and parameter setting, use the average reward over the last 100000 steps.

The results of the experiment are presented in Figure 2. See the companion *code* folder (for code and notebooks) for the implementation.

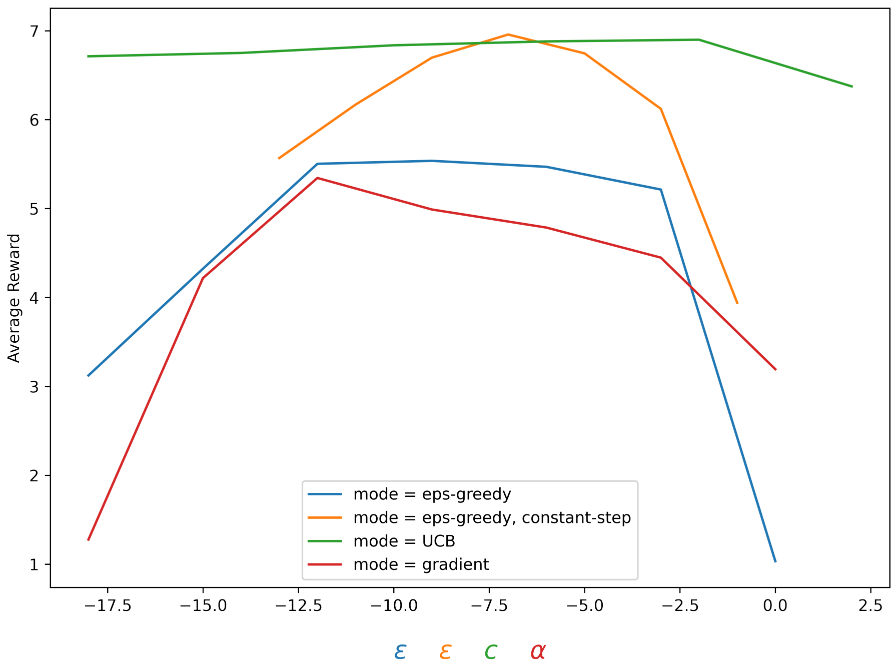

Figure 2: Parameter study of the various bandit algorithms on the 10-armed testbed for a non-stationary problem. Each point is the average reward obtained over 1000 steps with a particular algorithm at a particular setting of its parameter. These data are averages over 2000 runs with different bandit problems. x-axis values are reported as powers of 2.

In comparison with Figure 2.6, we can see that the parameter space for all the algorithm moves toward lower values. This means that lower learning rates are generally needed for non-stationary problems, in order to track the changing true action values.
The gradient bandit algorithm and ϵ-greedy with incremental sample averages perform the worst. While it is expected for the latter, the poor performance of the former is due to the skewed action preferences $H(t)$, which prevent exploration. In the early stages, the gradient bandit's action preference tend to skew toward the optimal choice and become overwhelmingly peaked on one arm only. However, while the optimal choice might in time change toward another arm, action preferences remain "stuck" on the sub-optimal arm due to the skew in the action preferences, hence preventing exploration.

The best performers are the constant-step size ϵ-greedy and the UCB algorithm. Note that a modified version of the UCB algorithm was implemented, called *discounted UCB* and appropriate for non-stationary problems (see [1]). Although the constant-step size ϵ-greedy algorithm performs slightly better than UCB for a particular choice of ϵ, UCB is fairly insensitive to its parameter c and performs well over a very broad range of parameter values. Therefore, once again UCB seems to perform best on this problem.

References

[1] Aurélien Garivier and Eric Moulines. On upper-confidence bound policies for non-stationary bandit problems, 2008.