Exercises Reinforcement Learning: An Introduction

Contents

Code for exercises can be found at github.com/brynhayder/reinforcement_learning_an_introduction

Note that equation numbers in questions will refer to the original text.

1 Chapter 1

1.1 Exercise 1.1: Self-Play

Q

Suppose, instead of playing against a random opponent, the reinforcement learning algorithm described above played against itself, with both sides learning. What do you think would happen in this case? Would it learn a different policy for selecting moves?

A

- Would learn a different policy than playing a fixed opponent since the opponent would also be changing in this case.
- May not be able to learn an optimal strategy as the opponent keeps changing also.
- Could get stuck in loops.
- Policy could remain static since on average they would draw each iteration.

1.2 Exercise 1.2: Symmetries

Q

Many tic-tac-toe positions appear different but are really the same because of symmetries. How might we amend the learning process described above to take advantage of this? In what ways would this change improve the learning process? Now think again. Suppose the opponent did not take advantage of symmetries. In that case, should we? Is it true, then, that symmetrically equivalent positions should necessarily have the same value?

A

- We could label the states as unique up to symmetries so that our search space is smaller, this way we will get a better estimate of optimal play.
- If we are playing an opponent who does not take symmetries into account when they are playing then we should not label the states as the same since the opponent is part of the environment and the environment is not the same in those states.

1.3 Exercise 1.3: Greedy Play

Q

Suppose the reinforcement learning player was greedy, that is, it always played the move that brought it to the position that it rated the best. Might it learn to play better, or worse, than a nongreedy player? What problems might occur

A

- The greedy player will not explore, so will in general perform worse than the non-greedy player
- If the greedy player had a perfect estimate of the value of states then this would be fine.

1.4 Exercise 1.4: Learning from Exploration

Q

Suppose learning updates occurred after all moves, including exploratory moves. If the step-size parameter is appropriately reduced over time (but not the tendency to explore), then the state values would converge to a set of probabilities. What are the two sets of probabilities computed when we do, and when we do not, learn from exploratory moves? Assuming that we do continue to make exploratory moves, which set of probabilities might be better to learn? Which would result in more wins?

A

I think that an estimate for the probability of the state producing a win should be based on the optimal moves from that state.

- The one in which we only record the optimal moves is the probability of our optimal agent winning. If we include exploration then this is the probability of the training agent winning.
- Better to learn the probability of winning with no exploration since this is how the agent will perform in real time play.
- Updating from optimal moves only will increase probability of winning.

1.5 Exercise 1.5: Other Improvements

Q

Can you think of other ways to improve the reinforcement learning player? Can you think of any better way to solve the tic-tac-toe problem as posed?

A

I'm not too sure here...

- We could rank the draws as better than the losses.
- We might like to try running multiple iterations of games before updating our weights as this might give a better estimate.

2 Chapter 2

2.1 Exercise 2.1

Q

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

A

0.5.

2.2 Exercise 2.2: Bandit example

Q

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 of $Q_1(a) = 0$, 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?

A

A_2 and A_5 were definitely exploratory. Any of the others *could* have been exploratory.

2.3 Exercise 2.3

Q

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.

A

The $\varepsilon = 0.01$ will perform better because in both cases as $t \to \infty$ we have $Q_t \to q_*$. The total reward and probability of choosing the optimal action will therefore be 10 times larger in this case than for $\varepsilon = 0.1$.

2.4 Exercise 2.4

Q

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 from 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?

A

Let $\alpha_0 = 1$, then

$$Q_{n+1} = \left(\prod_{i=1}^{n}(1-\alpha_i)\right) Q_1 + \sum_{i=1}^{n} \alpha_i R_i \prod_{k=i+1}^{n} (1-\alpha_k). \tag{1}$$

Where $\prod_{i=x}^{y} f(i) \doteq 1$ if $x > y$.

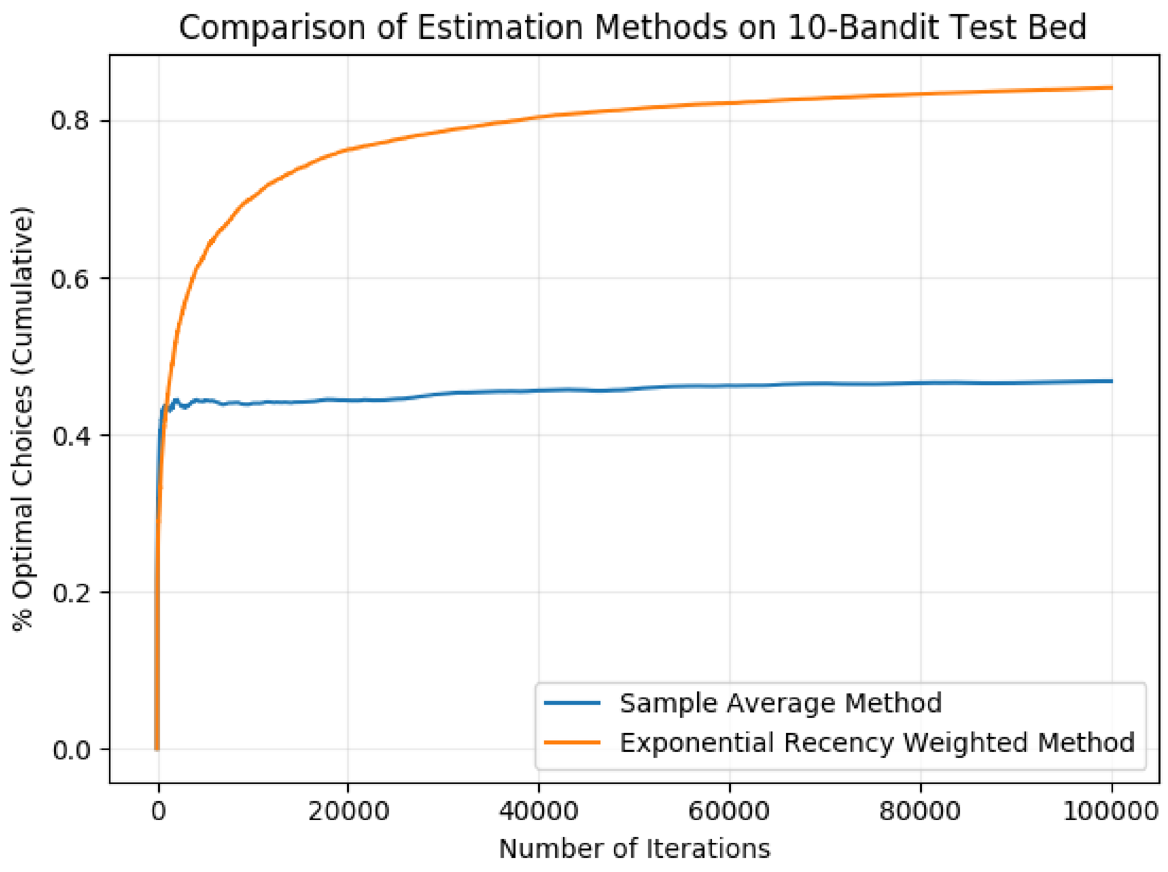
2.5 Exercise 2.5 (programming)

Q

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 $\varepsilon = 0.1$ and longer runs, say of 10,000 steps.

A

This is a programming exercise. For the relevant code/notebook please see the repo.



2.6 Exercise 2.6: Mysterious Values

Q

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 particular early steps?

A

At some point after step 10, the agent will find the optimal value. It will then choose this value greedily. The small step-size parameter (small relative to the initialisation value of 5) means that the estimate of the optimal value will converge slowly towards its true value.

It is likely that this true value is less than 5. This means that, due to the small step size, one of the sub-optimal actions will still have a value close to 5. Thus, at some point, the agent begins to act sub-optimally again.

2.7 Exercise 2.7: Unbiased Constant Step Trick

Q

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

$$\beta_t \doteq \alpha / \bar{o}_t, \tag{2}$$

where $\alpha > 0$ is a conventional constant step size and $\bar{o}_t$ is a trace of one that starts at 0:

$$\bar{o}_{t+1} = \bar{o}_t + \alpha(1 - \bar{o}_t) \tag{3}$$

for $t \geq 1$ and with $\bar{o}_1 \doteq \alpha$.

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

A

Consider the answer to Exercise 2.4. There is no dependence of Q_k on Q_1 for $k > 1$ since $\beta_1 = 1$. Now it remains to show that the weights in the remaining sum decrease as we look further into the past. That is

$$w_i = \beta_i \prod_{k=i+1}^{n} (1 - \beta_k) \tag{4}$$

increases with i for fixed n. For this, observe that

$$\frac{w_{i+1}}{w_i} = \frac{\beta_{i+1}}{\beta_i(1 - \beta_{i+1})} = \frac{1}{1 - \alpha} > 1 \tag{5}$$

where we have assumed $\alpha < 1$. If $\alpha = 1$ then $\beta_t = 1 \; \forall \, t$.

2.8 Exercise 2.8: UCB Spikes

Q

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 $c = 1$, then the spike is less prominent.

A

In the first 10 steps the agent cycles through all of the actions because when $N_t(a) = 0$ then a is considered maximal. On the 11th step the agent will most often then choose greedily. The agent will continue to choose greedily until $\ln(t)$ overtakes $N_t(a)$ for one of the other actions, in which case the agent begins to explore again hence reducing rewards.

Note that, in the long run, $N_t = O(t)$ and $\ln(t)/t \to 1$. So this agent is 'asymptotically greedy'.

2.9 Exercise 2.9

Q

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.

A

Let the two actions be denoted by 0 and 1. Now

$$\mathbb{P}(A_t = 1) = \frac{e^{H_t(1)}}{e^{H_t(1)} + e^{H_t(0)}} = \frac{1}{1 + e^{-x}}, \tag{6}$$

where $x = H_t(1) - H_t(0)$ is the relative preference of 1 over 0.

2.10 Exercise 2.10

Q

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?

A

I assume the rewards are stationary.

One should choose the action with the highest expected reward. In the first case, both action 1 and 2 have expected value of 0.5, so it doesn't matter which you pick.

In the second case one should run a normal bandit method separately on each colour. The expected reward from identifying the optimal actions in each case is 0.55.

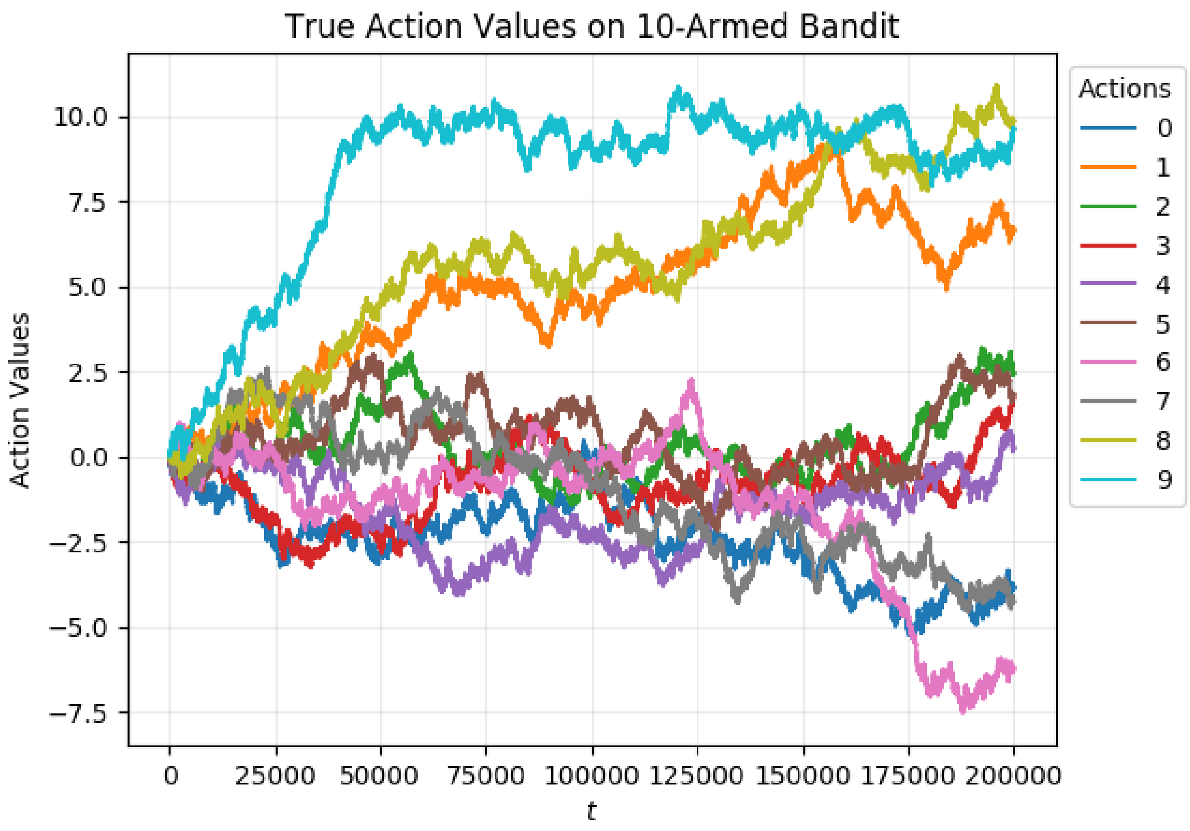
2.11 Exercise 2.11 (programming)

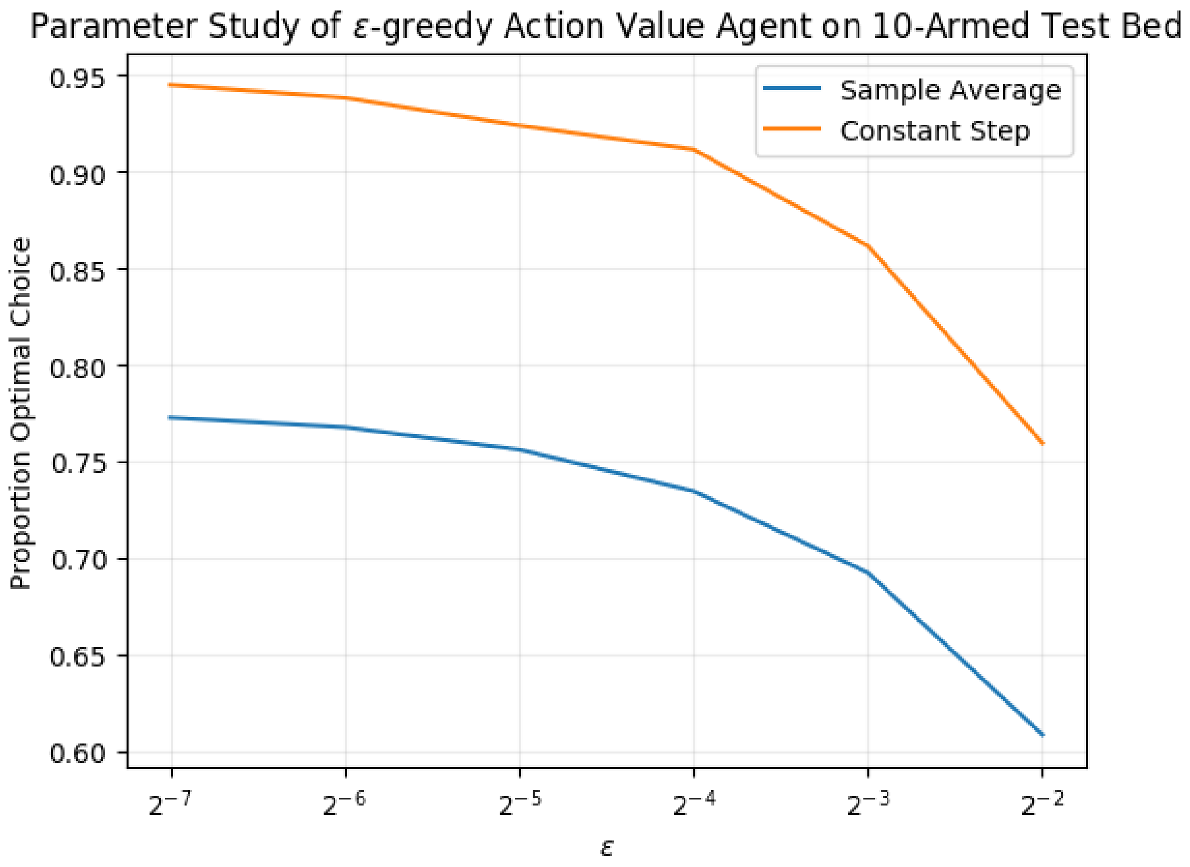
Q

Make a figure analogous to Figure 2.6 for the non-stationary case outlined in Exercise 2.5. Include the constant-step-size ε-greedy algorithm with $\alpha = 0.1$. Use runs of 200,000 steps and, as a performance measure for each algorithm and parameter setting, use the average reward over the last 100,000 steps.

A

This is a programming exercise. For the relevant code/notebook please see the repo.


True Action Values on 10-Armed Bandit
Action Values
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
0
25000
50000
75000
100000
125000
150000
175000
200000
t
Actions
0
1
2
3
4
5
6
7
8
9


Parameter Study of ε-greedy Action Value Agent on 10-Armed Test Bed
Proportion Optimal Choice
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
2^{-7}
2^{-6}
2^{-5}
2^{-4}
2^{-3}
2^{-2}
ε
Sample Average
Constant Step

3 Chapter 3