

# My Solutions for Exercises of Sutton and Barto 2nd Edition

Andres Espinosa

May 4, 2024

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Exercise 1.1 . . . . .	2
1.2	Exercise 1.2 . . . . .	2
1.3	Exercise 1.3 . . . . .	2
1.4	Exercise 1.4 . . . . .	2
1.5	Exercise 1.5 . . . . .	2
<b>2</b>	<b>Multi-armed Bandits</b>	<b>3</b>
2.1	Exercise 2.1 . . . . .	3
2.2	Exercise 2.2 . . . . .	3
2.3	Exercise 2.3 . . . . .	3
2.4	Exercise 2.4 . . . . .	3
2.5	Exercise 2.5 . . . . .	4
2.6	Exercise 2.6 . . . . .	4
2.7	Exercise 2.7 . . . . .	4
2.8	Exercise 2.8 . . . . .	4
2.9	Exercise 2.9 . . . . .	5
2.10	Exercise 2.10 . . . . .	5

# 1 Introduction

## 1.1 Exercise 1.1

The games would eventually result in a tie for every game. Since the model learns the optimal decision, and tic-tac-toe is a game where it is possible to always end in a tie in the case of optimal playing, the models would converge to always ending in a tie.

## 1.2 Exercise 1.2

We could model the value function in a way where symmetrical states have the same value function. This would benefit exploration because the agent would be able to learn the value function for a state that it has never truly reached. If the opponent did not take advantage of symmetries but the agent does, the opponent would have to experience a state before getting an approximation of the value function. The agent does not need to reach a state because it can use a symmetrical state as an accurate approximation of the value function for the other symmetrical states.

## 1.3 Exercise 1.3

Not necessarily, greedy actions can cause agents to get stuck in local minima. An agent could perform better if the greedy actions happen to be the best ones, but there could be a better set of actions that are in states the agent has never seen, and since it hasn't seen the state it may never due to its greedy actions.

## 1.4 Exercise 1.4

If an agent decides to learn from exploration, then it will learn the policy that it is using; a policy that combines optimal and suboptimal actions. If an agent explores but only learns from the greedy actions, then the value function it learns would be the value function that represents the optimal policy, not the policy it is actually executing (a mix of optimal and explorative).

## 1.5 Exercise 1.5

The tic-tac-toe is a rather simple problem but an agent could learn faster if it learned the value function of the opponent as well as itself. It could maybe use the opponent's value function to inform it's own since the environment is symmetrical to both of them.

## 2 Multi-armed Bandits

### 2.1 Exercise 2.1

Define  $a_1, a_2$  as the two actions that can be taken,  $\varepsilon$  as the probability of selecting a random action, and  $r$  as a randomly generated number  $\in [0, 1]$ .

$$\text{action} = \begin{cases} \text{random choice of } a_1 \text{ or } a_2 & \text{if } r < \varepsilon \\ \text{greedy choice} & \text{if } r \geq \varepsilon \end{cases} \quad (1)$$

if  $\varepsilon$  is 0.5, there is a 0.5 probability of taking the greedy choice and 0.5 probability of picking the random choice. The random choice has two actions, therefore there is a choice of picking  $a_1$  or  $a_2$  with 0.25 prob each.  $0.25 + 0.50 = 0.75$ , therefore the probability of the greedy action getting selected is 0.75

### 2.2 Exercise 2.2

$k$ -armed bandits with  $k = 4$ .  $\varepsilon$ -greedy algorithm setting  $Q_1(a) = 0 \forall a$ . Sample a trajectory  $(a_1, r_1, a_2, \dots)$  as  $(1, -1, 2, 1, 2, -2, 2, 2, 3, 0)$

- $A_1 = 1$ , Greedy (tie broken) or random
- $A_2 = 2$ , Greedy (tie broken) or random
- $A_3 = 2$ , Greedy or random
- $A_4 = 2$ , Must have been random because average reward of  $a_2$  is now  $-1/2$
- $A_5 = 3$ , Greedy (tie broken) or random

### 2.3 Exercise 2.3

In the long run, the  $\varepsilon = 0.01$  will perform the best. On average, it will pick the optimal action more frequently while still sampling each action an infinite number of times. In the long run, as  $Q(a) \approx q_*(a)$ ,  $\varepsilon = 0.01$  will sample the optimal action 99.1% of the time, whereas  $\varepsilon = 0.1$  will sample the optimal action 91% of the time, resulting in a smaller frequency of the optimal action being picked and a smaller total reward over the long run.

### 2.4 Exercise 2.4

Deriving the weighting on prior reward for general case  $\alpha_n \neq \alpha$

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

$$Q_{n+1} = \alpha_n R_n + (1 - \alpha_n) Q_n \quad (3)$$

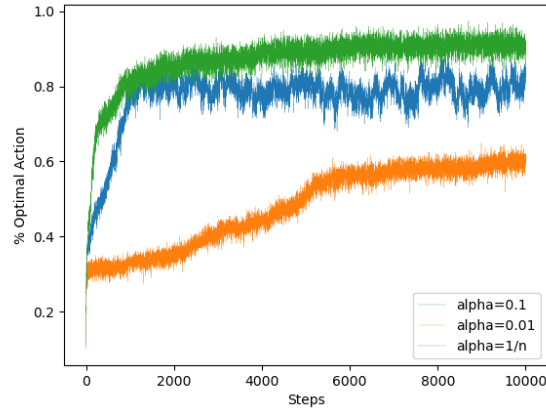
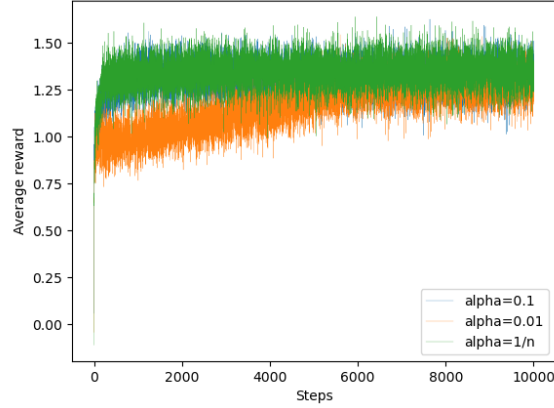
$$Q_{n+1} = \alpha_n R_n + (1 - \alpha_n)[\alpha_{n-1} R_{n-1} + (1 - \alpha_{n-1}) Q_{n-1}] \quad (4)$$

$$Q_{n+1} = \alpha_n R_n + (1 - \alpha_n)\alpha_{n-1} R_{n-1} + (1 - \alpha_n)(1 - \alpha_{n-1}) Q_{n-1} \quad (5)$$

$$Q_{n+1} = Q_1 \prod_{i=1}^n (1 - \alpha_i) + \sum_{i=1}^n R_i \alpha_i \prod_{j=i+1}^n (1 - \alpha_j) \quad (6)$$

## 2.5 Exercise 2.5

Completed in code



## 2.6 Exercise 2.6

Optimism will cause every option to look appealing until the option has been sampled enough times. If the option that is appealing happens to be an optimal choice, the decay of the Q value will be slower than that of the other choices.

## 2.7 Exercise 2.7

$\bar{o}_0 = 0$ , therefore  $\bar{o}_1 = 0 + \alpha(1 - 0) = \alpha$ .  $\beta_1 = \alpha/\alpha = 1$ . Using the equation derived at the end of exercise 2.4,  $Q_1 \prod_{i=1}^n (1 - \alpha_i)$  is 0 since  $\alpha_1$  is 0.  $Q_n$  does not include  $Q_1$  so it is without initial bias.

## 2.8 Exercise 2.8

In the UCB algorithm, each arm must be sampled once. Each arm that has not been pulled ( $N_t(a) = 0$ ), is classified as a maximizing action and will likely be pulled in the first  $|A|$  actions.

Since each arm has been sampled, on average, the optimal action will be the most appealing as it will provide the most reward in the long run. Therefore, each action will be made once and then the best seeming action is most likely to be selected.

## 2.9 Exercise 2.9

Soft-max as defined by equation 2.11 in the book is:

$$\pi_t(a) = \frac{e^{H_t(a)}}{\sum_{b=1}^k e^{H_t(b)}} \quad (7)$$

Sigmoid is defined as:

$$\sigma(x) = \frac{e^x}{e^x + 1} \quad (8)$$

When there are only two actions,  $a_1$  and  $a_2$ , they end up being the same thing. This can be shown since the preference  $H_t(a)$  only matters in relative importance to the other actions possible. To illustrate how this means the sigmoid is the same as soft-max, define  $H_t(a_1) = n$  and  $H_t(a_2) = m$ . Subtracting  $m$  from both results in  $H_t(a_1) = n - m$  and  $H_t(a_2) = 0$ . Since the relative aspect is the only thing that matters, the relative preference is still retained and this is a valid transformation. Substituting  $n$  and  $m$  into the softmax equation yields:

$$\pi_t(a_1) = \frac{e^{H_t(a_1)}}{e^{H_t(a_1)} + e^{H_t(a_2)}} \quad (9)$$

$$\pi_t(a_1) = \frac{e^{n-m}}{e^{n-m} + e^0} \quad (10)$$

$$\pi_t(a_1) = \frac{e^{n-m}}{e^{n-m} + 1} \quad (11)$$

$$\therefore \pi_t(a_1) = \sigma(n - m) \quad (12)$$

## 2.10 Exercise 2.10

If you do not know the case, then the problem can be framed as a 2-armed bandit where the reward structure is:  $a_1$  gives 10 w/ probability 0.50 and 90 w/ probability 0.50,  $a_2$  gives 20 w/ probability 0.50 and 80 w/ probability 0.50.

The expected reward for always picking  $a_1$  is  $E[a_1] = \frac{1}{2} * 10 + \frac{1}{2} * 90 = 50$  and the expected reward for always picking  $a_2$  is  $E[a_2] = \frac{1}{2} * 20 + \frac{1}{2} * 80 = 50$ . Since the expected rewards are the same for both actions, the best expected reward is 50 and the policy does not affect the expected reward.

In event that the case is known, it is indeed important which action you take in order to maximize reward. You should pick  $a_2$  when in case B and  $a_1$  when in case A. The best expected reward that can be achieved with this strategy is  $E[p_i] = \frac{1}{2} * 20 + \frac{1}{2} * 90 = 55$