Sutton & Barto RL Book Notes

Tin Vuong

Friday 23rd May, 2025

1 Chapter 2: Multi-arm Bandits

RL evaluates the actions taken rather than instructs correct actions like other forms of learning.

1.1 An n-Armed Bandit Problem

The problem

- You are faced repeatedly with a choice of n actions.
- After each choice, you receive a reward from a **stationary** probability distribution (reward for an action is sampled from the same prob dist every time).
- Objective is to maximise total reward over some time period, say 100 time steps.
- Named after of slot machine (one-armed bandit problem), but n levers instead of 1.
- Each action has an expected or mean reward based on its probability distribution. We shall call this the **value** of the action. We do not know these values with certainty. Because of this uncertainty, there is always an **exploration** vs **exploitation** problem. We always have one action that we deem to be most valuable at any instant, but it is highly likely, at least initially, that there are actions we are yet to explore that are more valuable.

1.2 Action-Value Methods

The estimated action value is

$$Q_t(a) \doteq \frac{\text{sum of rewards when } a \text{ was taken prior to } t}{\text{number of times } a \text{ was taken prior to } t} = \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbb{1}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbb{1}_{A_i=a}}, \tag{2.1}$$

where $\mathbb{1}_{predicate}$ denotes the random variable that is 1 if *predicate* is true and 0 if it is not.
If the denominator is zero, then we instead define $Q_t(a)$ as some default value, such as 0. As the denominator goes to infinity, by the law of large numbers, $Q_t(a)$ converges to $q_*(a)$.

$$Q_t(a) \xrightarrow[t \to \inf]{} q_*(a) \tag{1}$$

Call this the *sample-average* method for estimating action values because each estimate is an average of the sample of observed rewards.
The true value (mean reward) of an action is q, but the *estimated* value at the t-th time-step is $Q_t(a)$
The **greedy** action selection method is

$$A_t = \arg\max_a Q_t(a) \tag{2.2}$$

- Simplest action selection rule is to select the action with the highest estimated value.
- ϵ-greedy methods are where the agent selects the greedy option most of the time, and selects a random action with probability ϵ.
- Three algorithms are tried: one with e=0 (pure greedy), one with e=0.01 and another with e=0.1
- Greedy method gets stuck performing sub-optimal actions.
- e=0.1 explores more and usually finds the optimal action earlier, but never selects it more at 91% of the time.
- e=0.01 method improves more slowly, but eventually performs better than the e=0.1 on both performance measures.
- It is possible to reduce e over time to try to get the best of both high and low values.

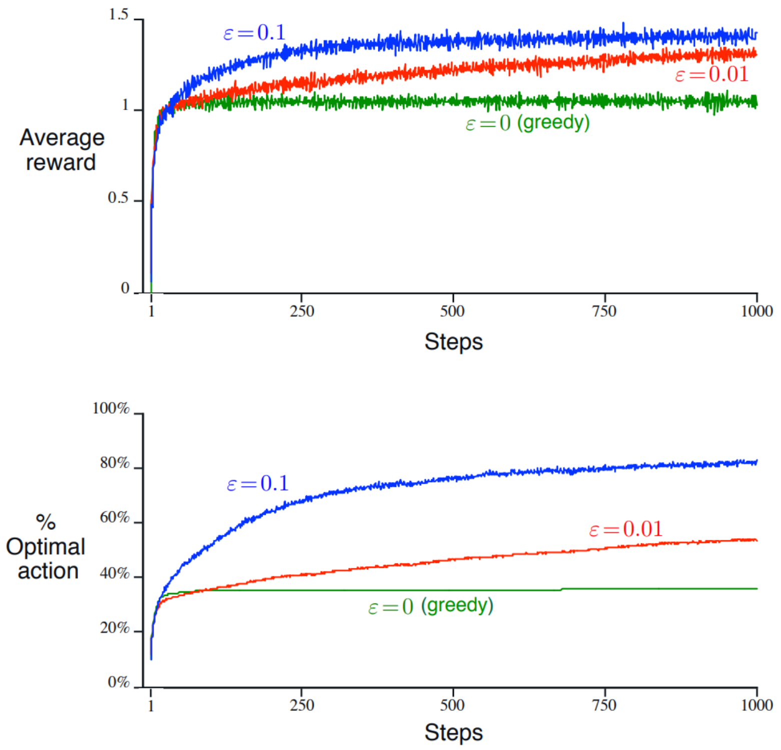


Figure 2.2: Average performance of ε-greedy action-value methods on the 10-armed testbed These data are averages over 2000 runs with different bandit problems. All methods used sample averages as their action-value estimates.

1.2.1 Exercises

Exercise 1.1. 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 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 " 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?

Answer:

1. At $t = 1$, we have $Q_1(1) = 0$, $Q_1(2) = 0$, $Q_1(3) = 0$, and $Q_1(4) = 0$. Any choice here could be greedy or exploration, so exploration *could have* occured. 1 is selected, and the action value for 1 is updated to $Q_2(1) = \frac{R_1}{1} = \frac{-1}{1} = -1$.

2. At $t = 2$, we have $Q_2(1) = -1$, $Q_2(2) = 0$, $Q_2(3) = 0$, and $Q_2(4) = 0$. Any choice but 1 here could be greedy or exploration, so exploration *could have* occured. The action value for 2 is updated to $Q_3(2) = 1$.

3. At $t = 3$, we have $Q_3(1) = -1$, $Q_3(2) = 1$, $Q_3(3) = 0$, and $Q_3(4) = 0$. 2 is selected again, and the action value for 2 is updated to $Q_4(2) = \frac{1+(-2)}{2} = \frac{-1}{2} = -0.5$.

4. At $t = 4$, we have $Q_4(1) = -1$, $Q_4(2) = -0.5$, $Q_4(3) = 0$, and $Q_4(4) = 0$. 2 is selected again even though its value is $0 < Q_4(3)$ and $0 < Q_4(4)$. So, *definitely* exploration. the action value for 2 is updated to $Q_5(2) = \frac{-1+2}{3} = \frac{1}{3} = 0.33$.

5. At $t = 5$, we have $Q_5(1) = -1$, $Q_5(2) = 0.33$, $Q_5(3) = 0$, and $Q_5(4) = 0$. 3 is selected even though its value is $0 < Q_5(2) = 0.33$. Therefore, we *definitely* explored here.

Time t	**Action Values $Q_t(a)$**	**Action Chosen**	**Exploration?**
1	$Q_1(a) = 0$ for all a	$A_1 = 1$	Possibly (all values tied)
2	$Q_2(1) = -1$ $Q_2(2) = 0$ $Q_2(3) = 0$ $Q_2(4) = 0$	$A_2 = 2$	Possibly (multiple actions tied for max)
3	$Q_3(1) = -1$ $Q_3(2) = 1$ $Q_3(3) = 0$ $Q_3(4) = 0$	$A_3 = 2$	Definitely greedy (2 is best)
4	$Q_4(1) = -1$ $Q_4(2) = -0.5$ $Q_4(3) = 0$ $Q_4(4) = 0$	$A_4 = 2$	Definitely exploration (2 is suboptimal)
5	$Q_5(1) = -1$ $Q_5(2) = 0.33$ $Q_5(3) = 0$ $Q_5(4) = 0$	$A_5 = 3$	Definitely exploration (3 ¡ 2)

Table 1: Analysis of exploration vs. greedy choices using ϵ-greedy algorithm.

Exercise 1.2. 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

Answer: According to Equation 1, the method with $\epsilon = 0.01$ will choose the optimal action 10x more often than the one with ϵ=0.1 because as $t \to \infty$, all $Q_\infty(a) \to q_*(a)$.

1.3 Incremental Implementation

The sample-average technique used to estimate action-values above has a problem: memory and computation requirements grow over time. This isn't necessary, we can devise an incremental solution instead:

$$Q_{k+1} = \frac{1}{k}\sum_{i=1}^{k} R_i \tag{2}$$

$$= \frac{1}{k}\left(R_k + \sum_{i=1}^{k-1} R_i\right) \tag{3}$$

$$\vdots$$

$$= Q_k + \frac{1}{k}[R_k - Q_k] \tag{4}$$

We are updating our estimate of Q_{k+1} by adding the discounted error between the reward just received and our estimate for that reward Q_k.

$$NewEstimate \leftarrow OldEstimate + StepSize[Target - OldEstimate] \tag{5}$$

α is used to denote the stepsize $(\frac{1}{k})$ in the rest of this book.

1.4 Non-Stationary Problems

Most problems in RL are **non-stationary** - the reward for an action is sampled from a prob dist that changes over time.
$\Rightarrow$ Give more weight to recent rewards, since their rewards are sampled from the most recent prob dists.
For example, the incremental update rule in Equation (5) for updating an average Q_n of the $n-1$ past rewards is modified to be

$$Q_{n+1} \doteq Q_n + \alpha[R_n - Q_n], \tag{6}$$

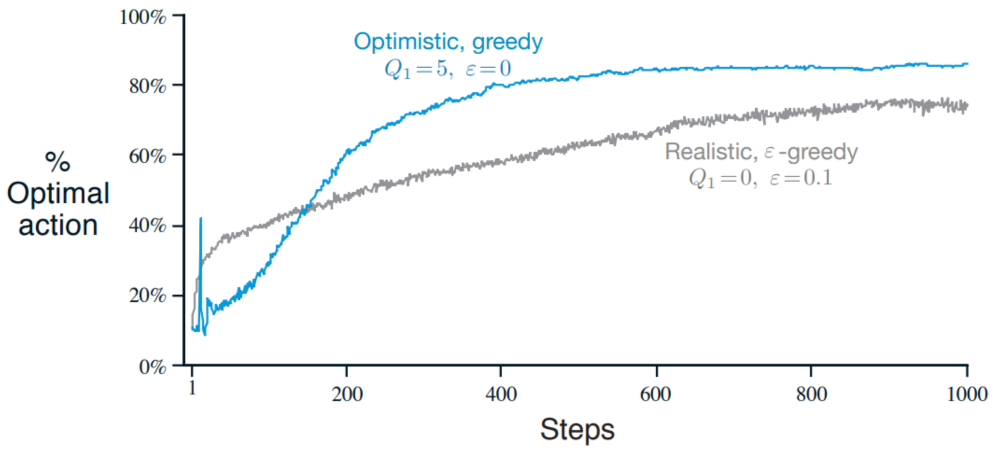
where the step-size parameter $\alpha \in (0,1]$ is **constant**. This results in Q_{n+1} being a weighted average of past rewards and the initial estimate Q_1:

$$\begin{aligned} Q_{n+1} &= Q_n + \alpha[R_n - Q_n] \\ &= \alpha R_n + (1-\alpha)Q_n \\ &= \alpha R_n + (1-\alpha)[\alpha R_{n-1} + (1-\alpha)Q_{n-1}] \\ &= \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2 Q_{n-1} \\ &= \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2 \alpha R_{n-2} + \dots \\ &\qquad \dots + (1-\alpha)^{n-1}\alpha R_1 + (1-\alpha)^n Q_1 \\ &= (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i. \end{aligned} \tag{7}$$

- Call this the **constant-stepsize** method for estimating action values.
- The sum of the weights add up to 1.
- Since $\alpha \in (0,1]$, the weight given to R_i decreases exponentially. We call this method *exponential recency-weighted average.*

1.5 Optimistic Initial Values

- The methods discussed so far are dependent to some extent on the initial action-value estimate i.e. they are biased by their initial estimates.
- For methods with constant α this bias is permanent.
- In effect, the initial estimates become a set of parameters for the model that must be picked by the user.
- In the above problem, by initial bias to +5 rather than 0 we encourage exploration, even in the greedy case. The agent will almost always be disappointed with it's samples because they are less than the initial estimate and so will explore elsewhere until the values converge.
- The above method of exploration is called *Optimistic Initial Values*
- ONLY suited for **stationary** problems. NOT suited for **non-stationary** problems since the drive for exploratory is only temporary.



1.5.1 Exercises

Exercise 1.3. 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 particular early steps?

Answer: The spikes could happen because the model initially selected the optimistic action, but then its reward was almost always less than the large initial bias, so the model was disappointed and explored other actions.

Exercise 1.4.

Answer:
We are given the update rule for the n-th reward for a particular action: $Q_n \leftarrow Q_{n-1} + \beta_n[R_n - Q_{n-1}]$, where Q_0 is the initial estimate. This can be written as:

$$Q_n = (1-\beta_n)Q_{n-1} + \beta_n R_n \quad (*) \tag{8}$$

The step size β_n is defined as:

$$\beta_n = \alpha/\bar{o}_n \quad \text{(cf. Eq. 2.8 in source)} \tag{9}$$

where $\alpha > 0$ is a constant step size, and $\bar{o}_n$ is a trace updated by:

$$\bar{o}_n = \bar{o}_{n-1} + \alpha(1-\bar{o}_{n-1}) = (1-\alpha)\bar{o}_{n-1} + \alpha, \quad \text{for } n > 0, \text{ with } \bar{o}_0 = 0 \quad \text{(cf. Eq. 2.9 in source).} \tag{10}$$

Analyze $\bar{o}_n$

The recurrence for $\bar{o}_n$ is $\bar{o}_n = (1-\alpha)\bar{o}_{n-1} + \alpha$. We can find a closed form for $\bar{o}_n$. Let $\bar{o}_n = 1 - \delta_n$.

$$\begin{aligned}1-\delta_n &= (1-\alpha)(1-\delta_{n-1}) + \alpha\\ 1-\delta_n &= 1-\delta_{n-1} - \alpha + \alpha\delta_{n-1} + \alpha\\ 1-\delta_n &= 1-(1-\alpha)\delta_{n-1}\end{aligned}$$

So, $\delta_n = (1-\alpha)\delta_{n-1}$. Given $\bar{o}_0 = 0$, we have $1-\delta_0 = 0 \implies \delta_0 = 1$. Thus, $\delta_n = (1-\alpha)^n\delta_0 = (1-\alpha)^n$. Therefore,

$$\bar{o}_n = 1-(1-\alpha)^n. \tag{11}$$

Substitute β_n into the update rule for Q_n

From $Q_n = (1-\beta_n)Q_{n-1} + \beta_n R_n$: Using $\beta_n = \alpha/\bar{o}_n$:

$$\begin{aligned} Q_n &= \left(1 - \frac{\alpha}{\bar{o}_n}\right) Q_{n-1} + \frac{\alpha}{\bar{o}_n} R_n \\ \bar{o}_n Q_n &= (\bar{o}_n - \alpha) Q_{n-1} + \alpha R_n. \end{aligned}$$

We know $\bar{o}_n = (1-\alpha)\bar{o}_{n-1} + \alpha$, so $\bar{o}_n - \alpha = (1-\alpha)\bar{o}_{n-1}$. Substituting this into the equation for $\bar{o}_n Q_n$:

$$\bar{o}_n Q_n = (1-\alpha)\bar{o}_{n-1} Q_{n-1} + \alpha R_n. \tag{12}$$

Expand the recurrence for $\bar{o}_n Q_n$

Let $X_n = \bar{o}_n Q_n$. The recurrence is $X_n = (1-\alpha)X_{n-1} + \alpha R_n$. This is a standard first-order linear recurrence relation. We can expand it:

$$\begin{aligned} X_n &= (1-\alpha)X_{n-1} + \alpha R_n \\ X_n &= (1-\alpha)((1-\alpha)X_{n-2} + \alpha R_{n-1}) + \alpha R_n \\ X_n &= (1-\alpha)^2 X_{n-2} + (1-\alpha)\alpha R_{n-1} + \alpha R_n \\ &\vdots \\ X_n &= (1-\alpha)^n X_0 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i. \end{aligned}$$

Analyze the initial condition X_0

$X_0 = \bar{o}_0 Q_0$. Since $\bar{o}_0 = 0$, we have $X_0 = 0 \cdot Q_0 = 0$. This is a crucial step. The initial estimate Q_0 is multiplied by zero.
Substituting $X_0 = 0$ into the expression for X_n:

$$X_n = \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i. \tag{13}$$

Express Q_n

Since $X_n = \bar{o}_n Q_n$: $Q_n = \frac{X_n}{\bar{o}_n} = \frac{\sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i}{\bar{o}_n}$. Substitute $\bar{o}_n = 1 - (1-\alpha)^n$:

$$Q_n = \frac{\sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i}{1-(1-\alpha)^n}. \tag{14}$$

Show it is an exponential recency-weighted average without initial bias

- **No initial bias:** The expression for Q_n depends only on the rewards $R_1, R_2, \ldots, R_n$ and the constant parameter α. The initial estimate Q_0 does not appear in the final expression because its effective coefficient (X_0) became zero. This means the estimate Q_n is **unbiased** by the choice of Q_0.
- **Exponential recency-weighted average:** The term $\sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i$ is a sum where each reward R_i is weighted by $\alpha(1-\alpha)^{n-i}$. The term $(1-\alpha)^{n-i}$ gives exponentially less weight to rewards that are further in the past (i.e., when i is small, $n-i$ is large). More recent rewards (i close to n, so $n-i$ is small) get higher weights. The denominator $1-(1-\alpha)^n$ is the sum of these unnormalized weights:

$$\sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} = \alpha \sum_{j=0}^{n-1} (1-\alpha)^j = \alpha \frac{1-(1-\alpha)^n}{1-(1-\alpha)} = \alpha \frac{1-(1-\alpha)^n}{\alpha} = 1-(1-\alpha)^n.$$

  So, Q_n can be written as $Q_n = \sum_{i=1}^{n} W_i R_i$, where the weights are $W_i = \frac{\alpha(1-\alpha)^{n-i}}{1-(1-\alpha)^n}$. These weights W_i sum to 1:

$$\sum_{i=1}^{n} W_i = \frac{\sum_{i=1}^{n} \alpha(1-\alpha)^{n-i}}{1-(1-\alpha)^n} = \frac{1-(1-\alpha)^n}{1-(1-\alpha)^n} = 1.$$

  Thus, Q_n is indeed an **exponential recency-weighted average** of the past rewards $R_1, \ldots, R_n$.

This analysis, similar to the one leading to (2.6) but with the crucial introduction of $\bar{o}_n$ and its properties (especially $\bar{o}_0 = 0$), demonstrates that Q_n is an exponential recency-weighted average of rewards without the initial bias introduced by Q_0 that is present in the constant step-size case.

Alternatively, performing the expansion "like that in (2.6)":

$$\begin{aligned}
Q_n &= (1-\beta_n)Q_{n-1} + \beta_n R_n \\
Q_n &= \beta_n R_n + (1-\beta_n)Q_{n-1} \\
Q_n &= \beta_n R_n + (1-\beta_n)[\beta_{n-1}R_{n-1} + (1-\beta_{n-1})Q_{n-2}] \\
Q_n &= \beta_n R_n + (1-\beta_n)\beta_{n-1}R_{n-1} + (1-\beta_n)(1-\beta_{n-1})Q_{n-2} \\
&\vdots \\
Q_n &= \beta_n R_n + (1-\beta_n)\beta_{n-1}R_{n-1} + \cdots + \left(\prod_{j=2}^{n}(1-\beta_j)\right)\beta_1 R_1 + \left(\prod_{j=1}^{n}(1-\beta_j)\right)Q_0.
\end{aligned}$$

We need to evaluate β_1. $\bar{o}_1 = (1-\alpha)\bar{o}_0 + \alpha = (1-\alpha)(0) + \alpha = \alpha$. So, $\beta_1 = \alpha/\bar{o}_1 = \alpha/\alpha = 1$. This means $1 - \beta_1 = 0$. The coefficient of Q_0 in the expansion is $\prod_{j=1}^{n}(1-\beta_j) = (1-\beta_n)(1-\beta_{n-1})\ldots(1-\beta_1)$. Since $1-\beta_1 = 0$, this entire product is zero. So,

$$Q_n = \beta_n R_n + (1-\beta_n)\beta_{n-1}R_{n-1} + \cdots + \left(\prod_{j=2}^{n}(1-\beta_j)\right)\beta_1 R_1. \tag{15}$$

This explicitly shows there is no bias from Q_0.
The weight for a reward R_k (for $1 \le k \le n$) is $W_k = \beta_k \prod_{j=k+1}^{n}(1-\beta_j)$. (Where the product is 1 if $k = n$). Using $1 - \beta_j = \frac{\bar{o}_j - \alpha}{\bar{o}_j} = \frac{(1-\alpha)\bar{o}_{j-1}}{\bar{o}_j}$ and $\beta_k = \frac{\alpha}{\bar{o}_k}$:

$$\begin{aligned}
W_k &= \frac{\alpha}{\bar{o}_k}\prod_{j=k+1}^{n}\frac{(1-\alpha)\bar{o}_{j-1}}{\bar{o}_j} \\
W_k &= \frac{\alpha}{\bar{o}_k}(1-\alpha)^{n-(k+1)+1}\frac{\bar{o}_k \cdot \bar{o}_{k+1} \cdot \ldots \cdot \bar{o}_{n-1}}{\bar{o}_{k+1} \cdot \bar{o}_{k+2} \cdot \ldots \cdot \bar{o}_n} \\
W_k &= \frac{\alpha}{\bar{o}_k}(1-\alpha)^{n-k}\frac{\bar{o}_k}{\bar{o}_n} = \frac{\alpha(1-\alpha)^{n-k}}{\bar{o}_n}.
\end{aligned}$$

So, $Q_n = \sum_{k=1}^{n} W_k R_k = \sum_{k=1}^{n} \frac{\alpha(1-\alpha)^{n-k}}{\bar{o}_n} R_k = \frac{\sum_{k=1}^{n}\alpha(1-\alpha)^{n-k}R_k}{1-(1-\alpha)^n}$. This confirms the previous result and demonstrates the exponential recency-weighting.

1.6 Upper-Confidence-Bound Action Selection

ϵ-greedy action selection forces the agent to explore new actions, but it does so indiscriminately. It would be better to select among non-greedy actions according to their potential for actually being optimal, taking into account both how close their estimates are to being maximal and the uncertainty in those estimates. One way to do this is to select actions as:

$$A_t = \underset{a}{\operatorname{argmax}}\left[Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}}\right] \tag{10}$$

where $c > 0$ controls the degree of exploration.

- The square root term is a measure of the uncertainty in our estimate. It is proportional to t i.e. how many timesteps have passed and inversely proportional to $N_t(a)$ i.e. how many times that action has been visited. The more time has passed, and the less we have sampled an action, the higher our upper-confidence-bound.

- As the timesteps increases, the denominator dominates the numerator as the ln term flattens.
- Each time we select an action our uncertainty decreases because N is the denominator of this equation.
- UCB will often perform better than ϵ-greedy methods

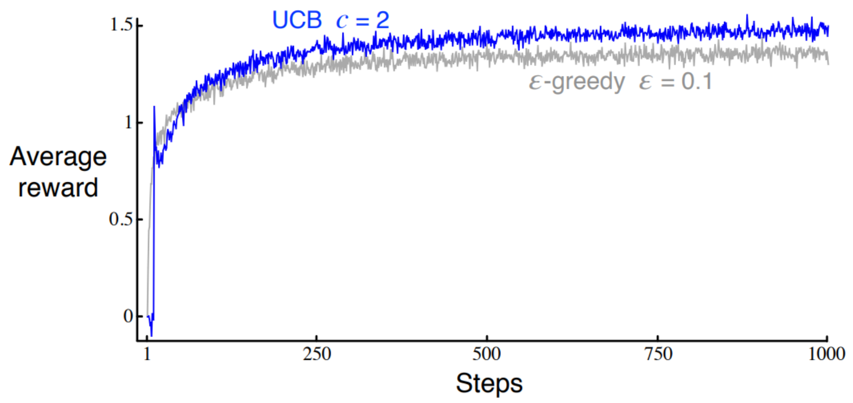


Figure 2.4: Average performance of UCB action selection on the 10-armed testbed. As shown, UCB generally performs better than ε-greedy action selection, except in the first k steps, when it selects randomly among the as-yet-untried actions.

1.6.1 Exercises

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 $c = 1$, then the spike is less prominent.
After 10 timesteps the UCB algorithm has explored all 10 actions as, until they are selected, their upper confidence bound is infinite (as $N_t(a) = 0$) as so it guarenteed to be selected once in the first 10 actions. At this point the agent has one sample to assess the expected value of each arm and the same confidence/uncertainty in each action. With the same c for every action, it is likely to pick the action with highest return from first sample, which will likely give it an similarly large reward, creating the spike. Now, the UCB for that action will decrease and the agent will select another, less valuable action, causing the decrease in performance at the next timestep.

1.7 Gradient Bandit Algorithms

- Gradient bandit algorithms are a class of algorithms that use the gradient of the action-value function to select actions.
- They are particularly useful in problems where the action space is continuous or large.
- Gradient bandit algorithms maintain a preference for each action, which is updated based on the received rewards.
- The action selection is based on a softmax function of the preferences, which allows for exploration and exploitation.
- The preferences are updated using a gradient *ascent* approach, where the update is proportional to the received reward and the difference between the action's preference and the average preference.
- See the official book for the proofs of the equations.

$$P(a) = \frac{e^{H(a)}}{\sum_{b=1}^{k} e^{H(b)}} = \pi_t(a) \tag{16}$$

for k actions available, and $\pi_t(a)$ is the probability of selecting action a at time t.
The preference for action a is denoted by $H(a)$ is given by:

$$H(a) \leftarrow H(a) + \alpha[R - \bar{R}] \cdot \frac{\partial P(a)}{\partial H(a)} \tag{17}$$

where $\bar{R}$ is the average reward of received up to but not including time t, which serves as the **baseline**. The baseline improves the performance significantly by adapting to the reward distribution shift by averaging the past rewards.

1.8 Associative Search

- So far, all the methods discussed has the learner's goal is to:
  - find the single best action if the task is stationary (reward probabilities don't change) or
  - to track the best action if the task is nonstationary (reward probabilities change over time). Note that it does NOT have knowledge about the changing state/situation and how they are relevant to action selection. A situation can change suddenly, and it will have to relearn the best action. It doesn't associate the change with a new **type** of situation it needs to recognize; it just recognizes that the **effectiveness** of its actions has changed.
  - There is no need to associate different actions with different situations.
- However, in RL, there are many situations/environments with different best actions.
- The goal then is to learn a **poliicy** that *maps states to actions* rather than just a single action.
- In **associative search / contextual bandits**, the agent is given a set of actions and a set of contexts/clues. The agent must learn to select the best action for each context.
- Associative tasks are intermediate between the n-armed bandit problem and the full RL problem: They are like the full RL problem in that they involve learning a policy, but they are also like our version of the k-armed bandit problem in that each action affects only the *immediate* reward. If actions are allowed to affect the *next situation* as well as the reward, then we have the full RL problem.

1.8.1 Exercises

Exercise 1.5. 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 10 and 20 with probability 0.5 (case A), and 90 and 80 with probability 0.5 (case B). If you are not able to tell which case you face at any step, what is the best expected reward 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 expected reward you can achieve in this task, and how should you behave to achieve it?

Answer: If we select the actions at random, the expected reward is:

$$E[R] = \frac{1}{2} \cdot E[R[A]] + \frac{1}{2} \cdot E[R[B]] = \frac{1}{2}(\cdot 10 + \frac{1}{2} \cdot 20) + \frac{1}{2}(\cdot 90 + \frac{1}{2} \cdot 80) = \frac{1}{2}(15 + 85) = 50 \tag{18}$$

In the associative search task, after a while, we will find out the best action for each situation. Then, we should keep picking that action for that situation, and the expected reward is:

$$E[R] = \cdot E[R[A]] + \cdot E[R[B]] = (\frac{1}{2} \cdot 20) + (\frac{1}{2} \cdot 90) = (10 + 45) = 55.0 \tag{19}$$

2 Chapter 3: Finite Markov Deicsion Process (MDP)

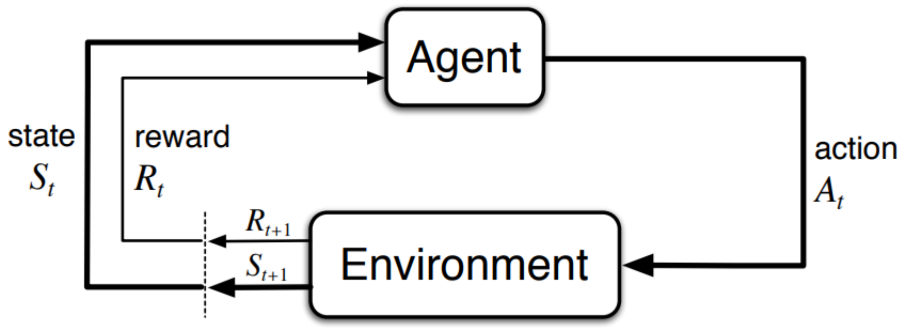


Figure 3.1: The agent–environment interaction in a Markov decision process.

- The reward is a consequence of an action at a state. A state (S_i, A_i) will receive a reward at time $i+1:\ R_{i+1}$.
- A state or an action can be anything. A reward must be a single number.
- The (state, actionn, reward) triplets form the **trajectory**:

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, ... \tag{20}$$

$$p(s', r|s, a) = Pr\{S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a\}, \tag{21}$$

$$\sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r|s, a) = 1, \forall s \in S, a \in A(s). \tag{22}$$

- In general, actions can be any decision we want to learn how to make, and states can be any interpretation of the world that might inform those actions.
- The boundary between agent and environment is much closer to the agent than is first intuitive.
- we always consider the reward computation to be external to the agent because it defines the task facing the agent and thus must be beyond its ability to change arbitrarily.
- The general rule we follow is that anything that cannot be changed arbitrarily by the agent is considered to be outside of it and thus part of its environment. The agent may still know everything about the environment, but still unable to solve it. E.g. solving a Rubik's cube.

2.0.1 Exercises

Exercise 2.1. Consider the problem of driving. You could define the actions in terms of the accelerator, steering wheel, and brake, that is, where your body meets the machine. Or you could define them farther out—say, where the rubber meets the road, considering your actions to be tire torques. Or you could define them farther in—say, where your brain meets your body, the actions being muscle twitches to control your limbs. Or you could go to a really high level and say that your actions are your choices of where to drive. What is the right level, the right place to draw the line between agent and environment? On what basis is one location of the line to be preferred over another? Is there any fundamental reason for preferring one location over another, or is it a free choice?

Answer: There's a trade-off between state-action space complexity, computational expense and accuracy. If we draw the boundary at the brain we would create a state-action space contingent on the number of neurons in the brain and their interplay with physical actions like turning the steering wheel; too large to be stored or computed efficiently. Equally, if we draw the boundary at the journey level, then we miss the detail required to act on second-by-second state changes on the road that could lead to a crash. The fundamental limiting factor in this selection is whether the goal can be achieved safely at your chosen layer of abstraction, indeed this feels like one of the tasks of engineering more widely

2.1 Goals and Rewards

- The *reward hypothesis*: "That all of what we mean by goals and purposes can be well thought of as the maximization of the expected value of the cumulative sum of a received scalar signal (called reward)."
- It is thus critical that the rewards we set up truly indicate what we want accomplished. In particular, the reward signal is not the place to impart to the agent prior knowledge about *how* to achieve what we want it to do. For example, a chess-playing agent should be rewarded only for actually winning, not for achieving subgoals such as taking its opponent's pieces or aining control of the center of the board. If achieving these sorts of subgoals were rewarded, then the agent might find a way to achieve them without achieving the real goal. For example, it might find a way to take the opponent's pieces even at the cost of losing the game.
- The reward signal is your way of communicating to the agent *what* you want achieved, not *how* you want it achieved.

2.2 Returns and Episodes

- An agent needs to maximize the cummulative reward in the long run, not just the immediate reward.
- Can we use the sum of all rewards as the metric? it can grow without bound if the task doesn't have a terminal state.
- Instead, use discounts:

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{23}$$

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \cdots) = R_{t+1} + \gamma G_{t+1} \tag{24}$$

2.2.1 Exercises

Exercise 2.2. Suppose you treated pole-balancing as an episodic task but also used discounting, with all rewards zero except for 1 upon failure. What then would the return be at each time? How does this return differ from that in the discounted, continuing formulation of this task?

Answer: For *episodic* tasks, each episode has a terminate state. And the corresponding metric, with discounts, for each episode is:

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-t-1} R_T = 0 + \gamma^{T-t-1} R_T = \gamma^{T-t-1}(-1) = -\gamma^{T-t-1}.$$

Exercise 2.3. Imagine that you are designing a robot to run a maze. You decide to give it a reward of +1 for escaping from the maze and a reward of zero at all other times. The task seems to break down naturally into episodes—the successive runs through the maze—so you decide to treat it as an episodic task, where the goal is to maximize expected total reward (3.7). After running the learning agent for a while, you find that it is showing no improvement in escaping from the maze. What is going wrong? Have you effectively communicated to the agent what you want it to achieve?

Answer: We only reward it for exiting the maze, but did not penalize it for the time it takes to exit the maze. We can improve on this by giving negative rewards as the time it takes to exit the maze increases.

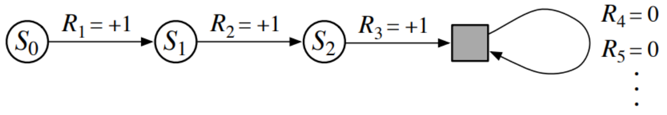
Exercise 2.4. Suppose $\gamma = 0.5$ and the following sequence of rewards is received $R_1 = 1, R_2 = 2, R_3 = 6, R_4 = 3,$ and $R_5 = 2,$ with $T = 5$. What are $G_0, G_1, ..., G_5$? Hint: Work backwards.

Answer: We know $G_t = R_{t+1} + \gamma \cdot G_{t+1}$. So

1. $G_5 = 0$
2. $G_4 = R_5 + \gamma \cdot G_5 = 2 + 0 = 2$
3. $G_3 = R_4 + \gamma \cdot G_4 = 3 + 0.5 \cdot 2 = 4$
4. $G_2 = R_3 + \gamma \cdot G_3 = 6 + 0.5 \cdot 4 = 8$
5. $G_1 = R_2 + \gamma \cdot G_2 = 2 + 0.5 \cdot 8 = 6$
6. $G_0 = R_1 + \gamma \cdot G_1 = 1 + 0.5 \cdot 6 = 4$

2.3 Unified Formulation for Episodic and Continuing Tasks

- Use a self-absorbing state S_T that returns a reward of 0 to represent the termination of an episode.



- The sum remains the same for both terminating (episodic) and continuing task.
- To include discounting, either $\gamma = 1$ or $T = \infty$ but not both. Cause if both are true, then the sum diverges.

2.4 Value Functions and Action-Value Functions

- The **state-value function** for a policy π, denoted $v_\pi(s)$, is the expected return when starting in state s and following policy π thereafter. It is formally defined as:

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t | S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s\right], \quad \text{for all } s \in \mathcal{S} \tag{25}$$

  where $\mathbb{E}_\pi[\cdot]$ denotes the expected value given that the agent follows policy π, G_t is the total discounted return from time step t, S_t is the state at time t, R_{t+k+1} is the reward at time step $t + k + 1$, and γ is the discount factor. The value of the terminal state, if any, is always zero.

- Similarly, the **action-value function** for a policy π, denoted $q_\pi(s, a)$, is the expected return starting from state s, taking action a, and thereafter following policy π. It is formally defined as:

$$q_\pi(s, a) \doteq \mathbb{E}_\pi[G_t | S_t = s, A_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s, A_t = a\right] \tag{26}$$

  where A_t is the action taken at time step t.

- v_π measures *how good* a *state* is, while q_π measures *how good* a *state-action pair* is.

- The **Bellman equation** states that the value of the start state must equal the (discounted) value of the expected next state, plus the *expected* reward along the way:

$$\begin{aligned} v_\pi(s) &\doteq \mathbb{E}_\pi[G_t|S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1}|S_t = s] \\ &= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a) \left[r + \gamma \mathbb{E}_\pi[G_{t+1}|S_{t+1} = s']\right] \\ &= \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_\pi(s')\right], \quad \text{for all } s \in \mathcal{S}, \end{aligned} \tag{27}$$

- v_π satisfies a recursive relationship and is a unique solutionn to Bellman's equation.
item v_π satisfies a recursive relationship and is a unique solutionn to Bellman's equation.

- Similarly, the Bellman equation for the action-value function $q_\pi(s, a)$ is:

$$\begin{aligned} q_\pi(s,a) &\doteq \mathbb{E}_\pi[G_t|S_t = s, A_t = a] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1}|S_t = s, A_t = a] \\ &= \sum_{s'} \sum_r p(s', r|s, a) \left[r + \sum_{a'} \pi(a'|s') \gamma \mathbb{E}_\pi[G_{t+1}|S_{t+1} = s', A_{t+1} = a']\right] \\ &= \sum_{s'} \sum_r p(s', r|s, a) \left[r + \gamma \sum_{a'} \pi(a'|s') q_\pi(s', a')\right] \\ &= \sum_{s',r} p(s', r|s, a) \left[r + \gamma \sum_{a'} \pi(a'|s') q_\pi(s', a')\right] \end{aligned} \tag{28}$$

2.4.1 Exercises

Exercise 2.5. If the current state is S_t, and actions are selected according to a stochastic policy π, then what is the expectation of $R_t + 1$ in terms of π and the four-argument function p in (21)?

Answer:

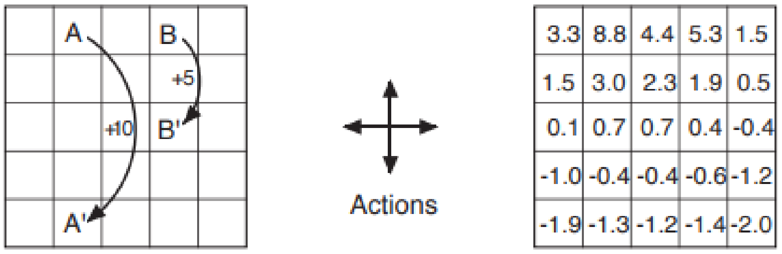
$$\mathbb{E}_\pi[R_{t+1}|S_t] = \sum_a \pi(a|S_t) \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} r \cdot p(s', r|S_t, A_t = a).$$

Exercise 2.6. Give an equation for v_π in terms of q_π and π.

Answer: We have $v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s] = \sum_a \pi(a|s) q_\pi(s, a)$. This is the expected value of q_π over all actions a taken in state s according to policy π.

$$\begin{gathered} v_\pi(s) = \sum_a \pi(a|s)\ \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]. \\ q_\pi(s,a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]. \\ \Rightarrow v_\pi(s) = \sum_a \pi(a|s)\ q_\pi(s,a). \end{gathered}$$

Figure 3.2 (left) shows a rectangular gridworld representation of a simple finite MDP. The cells of the grid correspond to the states of the environment. At each cell, four actions are possible: north, south, east, and west, which deterministically cause the agent to move one cell in the respective direction on the grid. Actions that would take the agent off the grid leave its location unchanged, but also result in a reward of 1. Other actions result in a reward of 0, except those that move the agent out of the special states A and B. From state A, all four actions yield a reward of $+10$ and take the agent to A' From state B, all actions yield a reward of $+5$ and take the agent to B'



3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

Figure 3.2: Gridworld example: exceptional reward dynamics (left) and state-value function for the equiprobable random policy (right).

Exercise 2.7. The Bellman equation (27) must hold for each state for the value function v_ϕ shown in Figure 3.2 (right) of the gridworld above. Show numerically that this equation holds for the center state, valued at $+0.7$, with respect to its four neighboring states, valued at $+2.3$, $+0.4$, 0.4, and $+0.7$. (These numbers are accurate only to one decimal place.)

Answer: Using the Bellman equation, we have:

$$\begin{aligned}
v_\pi(s) &= \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\left[r + \gamma v_\pi(s')\right] \\
&= 0.25 \cdot (0.9 \times 0.7) + 0.25 \cdot (0.9 \times 2.3) + 0.25 \cdot (0.9 \times 0.4) + 0.25 \cdot (0.9 \times -0.4) \\
&= 0.25 \cdot 0.9 \cdot (0.7 + 2.3 + 0.4 - 0.4) \\
&= 0.675 \approx 0.7.
\end{aligned}$$

Exercise 2.8. In the gridworld example, rewards are positive for goals, negative for running into the edge of the world, and zero the rest of the time. Are the signs of these rewards important, or only the intervals between them? Prove, using (24), that adding a constant c to all the rewards adds a constant, v_c, to the values of all states, and thus does not affect the relative values of any states under any policies. What is v_c in terms of c and γ?

Answer: Adding a constant c to all rewards results in the following equation:

$$\begin{aligned}
G_t &= \sum_{k=0}^{\infty} \gamma^k (R_{t+k+1} + c) \\
&= \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} + \sum_{k=0}^{\infty} \gamma^k c
\end{aligned}$$

Thus, the value of each state increases by:

$$\begin{aligned} v_c &= \sum_{k=0}^{\infty} \gamma^k c \\ &= c \cdot \sum_{k=0}^{\infty} \gamma^k \\ &= c \cdot \frac{1}{1-\gamma} \quad (\text{for } |\gamma| < 1) \\ &= \frac{c}{1-\gamma}. \end{aligned}$$

Exercise 2.9. Now consider adding a constant c to all the rewards in an episodic task, such as maze running. Would this have any effect, or would it leave the task unchanged as in the continuing task above? Why or why not? Give an example.

Answer: Yes, it would affect the agent. Episodes have a finite length, so the total return is finite. Adding a constant c to all rewards would change the total return for each episode by c times the number of time steps in that episode. This would affect the agent's learning and policy, as it would now be incentivized to complete episodes faster or slower depending on the sign of c. For example, if c is positive, it will overwhelm the negative rewards for "bad decisions", so the agent is not incentivized to learn from its mistakes, and continues collecting intermediate rewards indefinitely.
*Consequently, adding a constant to all rewards in an episodic task can alter the **relative** values of different states and policies, and may therefore change the optimal policy, especially by influencing the agent's preference for **episode length**.*
Consider a simple maze with a starting state S and a terminal goal state T. The agent receives a reward R_{step} for each step taken. The episode ends upon reaching T.

- Path 1 (Short, 2 steps toTG): Total Return $G_1 = \sum_{k=1}^{2} R_{step} = 2 \times (-1) = -2$.
- Path 2 (Long, 3 steps to G): Total Return $G_2 = \sum_{k=1}^{3} R_{step} = 3 \times (-1) = -3$.

Since $-2 > -3$, Path 1 is preferred.
Now, add a constant $c = +5$ to all rewards. The new reward per step is $R'_{step} = R_{step} + c = -1 + 5 = 4$. The new total return G'_t for an episode of length L starting from a state leading to original return G_t is $G'_t = G_t + L \cdot c$.

- Path 1 (Short, $L = 2$ steps): $G'_1 = G_1 + 2c = -2 + 2(5) = -2 + 10 = 8$. Alternatively, $G'_1 = 2 \times R'_{step} = 2 \times 4 = 8$.
- Path 2 (Long, $L = 3$ steps): $G'_2 = G_2 + 3c = -3 + 3(5) = -3 + 15 = 12$. Alternatively, $G'_2 = 3 \times R'_{step} = 3 \times 4 = 12$.

Since $12 > 8$, Path 2 (the longer path) is now preferred. The addition of a positive constant favored the policy leading to a longer episode.

Exercise 2.10. What is the Bellman equation for action values, that is, for q_π? It must give the action value $q_\pi(s, a)$ in terms of the action values, $q_\pi(s', a')$, of possible successors to the state-action pair (s, a). Hint: The backup diagram to the right corresponds to this equation. Show the sequence of equations analogous to (28), but for action values.

Answer: See (28).