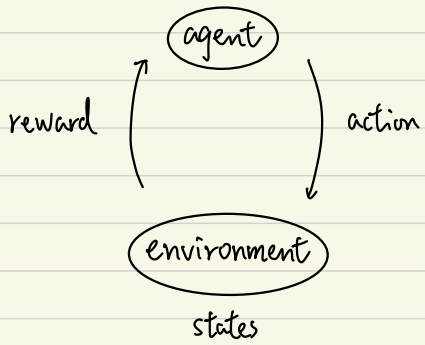
CH2. Multi-armed Problem



2-1 A k-armed bandit problem

Imagine that you have k choices and each choice corresponds to a reward. At time t (discrete time), you are going to choose one of them. Our goal is to estimate the expected reward given an action a since once we know the value, expected reward, we know which choice is the optimal choice. In other words, we want to estimate

$$q_*(a) = \mathbb{E}[R_t \mid A_t = a],$$

where

R_t : the reward at time t

A_t : the selected action at time t.

* The value refers to $q_*(a)$ in the following context.

In practice, we don't know what $q_*(a)$ is, so we estimate it with $Q_t(a)$ which denotes the estimated value.

exploration vs. exploitation

exploration：尋找更多可能

exploitation：根據已知資訊做出最佳選擇

* In this chapter, we will
  - provide balancing methods for k-armed problems.
  - show that these methods are better than the methods that always exploit.

2-2 Action-value Methods

Define $Q_t(a)$ in the following fashion:

$$Q_t(a) := \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbb{1}_{A_i = a}}{\sum_{i=1}^{t-1} \mathbb{1}_{A_i = a}}$$ (stationary case: reward doesn't change with time t)

This is called the sample-average method. By strong law of large numbers, $Q_t(a) \longrightarrow q_*(a)$.

greedy-algorithm : always choose the optimal one "according to our information"

$$A_t = \underset{a}{\operatorname{argmax}}\ Q_t(a)$$

How about other choices? Probably the unexplored choices are better than our current optimal choice.

2.3 10-armed Testbed

ε-greedy :

$$A_t = \begin{cases} \underset{a}{\operatorname{argmax}}\ Q_t(a) & , 1-\varepsilon \\ 0 & , \varepsilon \end{cases}$$

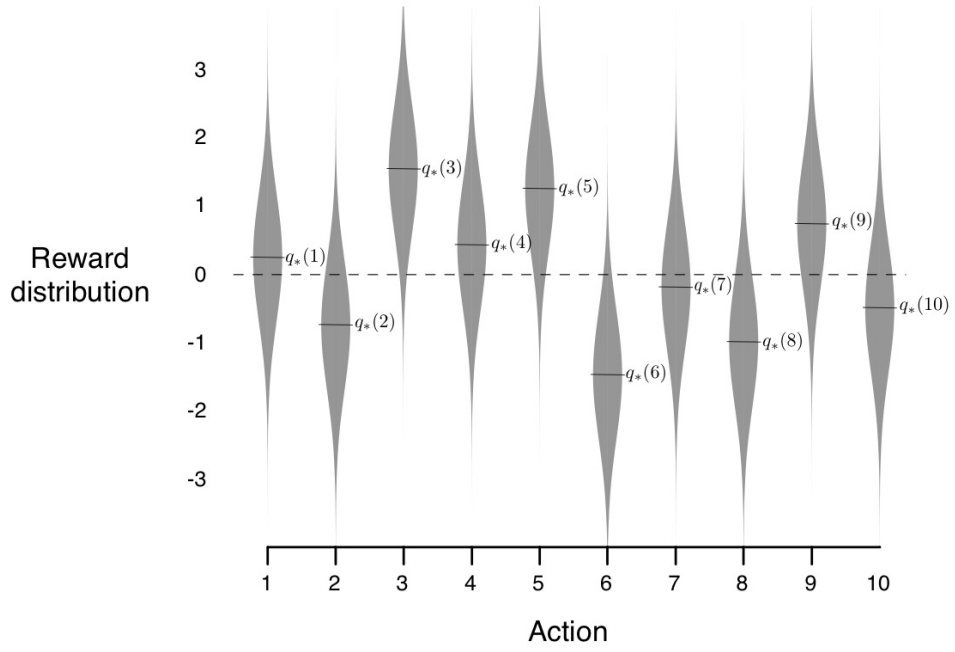
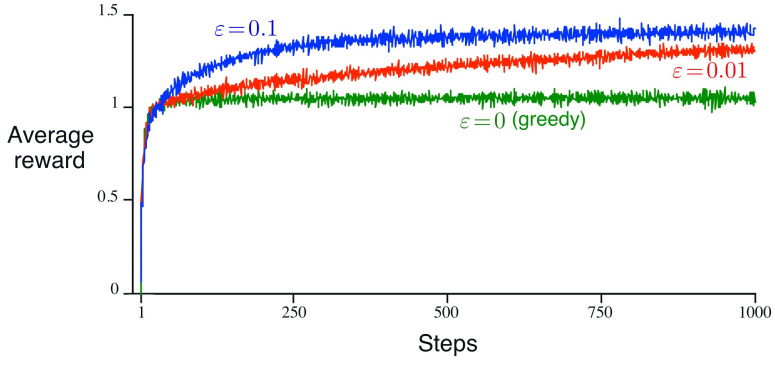
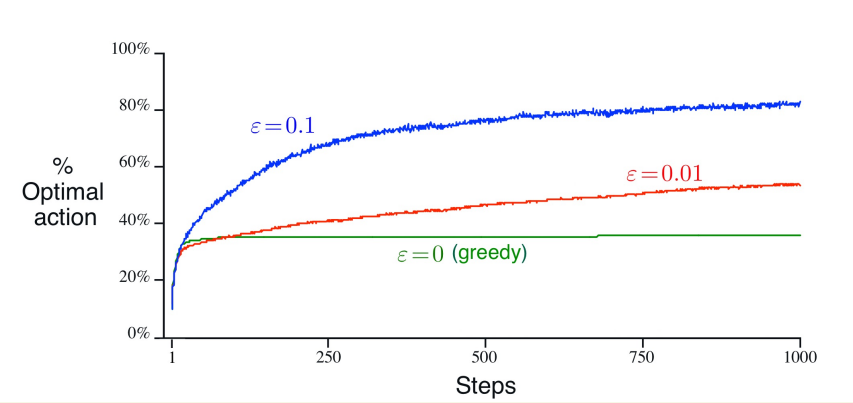


Figure 2.1: An example bandit problem from the 10-armed testbed. The true value $q_*(a)$ of each of the ten actions was selected according to a normal distribution with mean zero and unit variance, and then the actual rewards were selected according to a mean $q_*(a)$, unit-variance normal distribution, as suggested by these gray distributions.

$q_*(a)$：從 $N(0,1)$ 中隨機抽取

The actual reward is selected from a normal distribution with mean $q_*(A_t)$ and variance 1, i.e. for an action a, $R_t \sim N(q_*(A_t), 1)$.





⇒ ε-greedy performs better in this task.

2-4 Increment Implementation

There is an issue when implementing sample-average: it will consume lots of memory since the computer has to memorize each R_i.

$$Q_n = \frac{R_1 + \cdots + R_{n-1}}{n-1} \quad \text{(sample-average)}$$

We can modify Q_{n+1} as

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

$$NewEstimate \leftarrow OldEstimate + StepSize\left[Target - OldEstimate\right].$$

A simple bandit algorithm

Initialize, for $a = 1$ to k:
$Q(a) \leftarrow 0$
$N(a) \leftarrow 0$

Loop forever:
$A \leftarrow \begin{cases} \arg\max_a Q(a) & \text{with probability } 1-\varepsilon \quad \text{(breaking ties randomly)} \\ \text{a random action} & \text{with probability } \varepsilon \end{cases}$
$R \leftarrow bandit(A)$
$N(A) \leftarrow N(A) + 1$
$Q(A) \leftarrow Q(A) + \frac{1}{N(A)}\left[R - Q(A)\right]$

2-5 Tracking a Nonstationary Problem

stationary: reward 隨 t 變動 (e.g. above cases)
nonstationary: reward 不隨 t 變動

For nonstationary cases, it makes sense to give more weight to recent rewards. First, note that

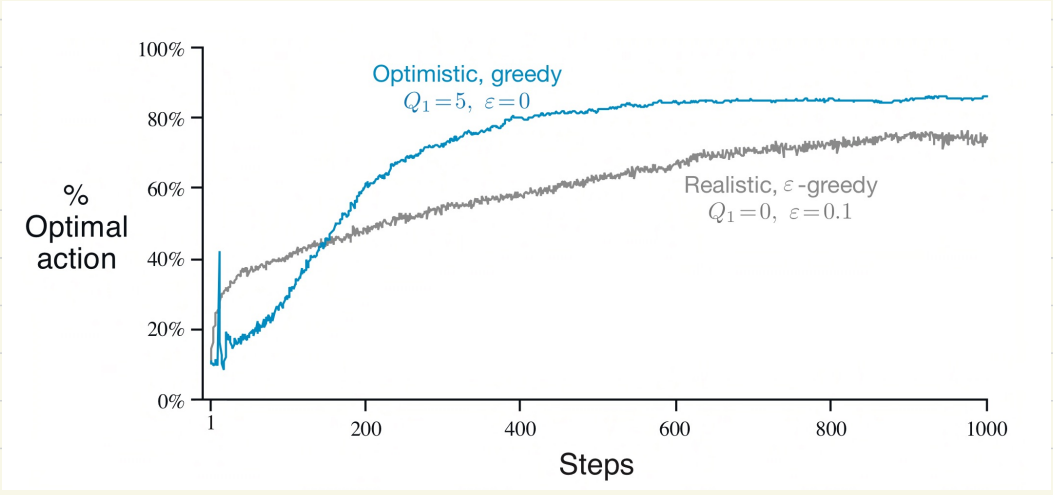
$$
\begin{aligned}
Q_{n+1} &= Q_n + \alpha\left[R_n - Q_n\right] \\
&= \alpha R_n + (1-\alpha)Q_n \\
&= \alpha R_n + (1-\alpha)\left[\alpha R_{n-1} + (1-\alpha)Q_{n-1}\right] \\
&= \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2 Q_{n-1} \\
&= \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2 \alpha R_{n-2} + \\
&\qquad \cdots + (1-\alpha)^{n-1}\alpha R_1 + (1-\alpha)^n Q_1 \\
&= (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i. \quad \text{(weighted average)}
\end{aligned}
$$

Denote $\alpha_n(a)$ the step-size after n^{th} selection of action a. $\{\alpha_n(a)\}$ converges with prob. 1 if

(1) $\sum_{n=1}^{\infty} \alpha_n(a) < \infty$

(2) $\sum_{n=1}^{\infty} \alpha_n^2(a) < \infty$.

2-6 Optimistic Initial Values



Main idea: the choice of initial values will influence the performance.

2-7 Upper-Confidence-Bound Action Selection

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

$N_t(a)$ = # a is selected before t, c : fixed constant

$N_t(a) = 0 \Rightarrow$ maximizing action

Remark

1. We can view $\sqrt{\frac{\ln t}{N_t(a)}}$ as the "uncertainty"
   The more time a is selected, the less the uncertainty is.
2. One can view $\sqrt{\frac{\ln t}{N_t(a)}}$ as a balance between exploration and exploitation.

e.g. Fix t. Suppose there are only two choices of action a_1, a_2 and that $Q_t(a_1) = Q_t(a_2)$.
If $N_t(a_1) < N_t(a_2)$, then we'll choose a_1 (exploration).

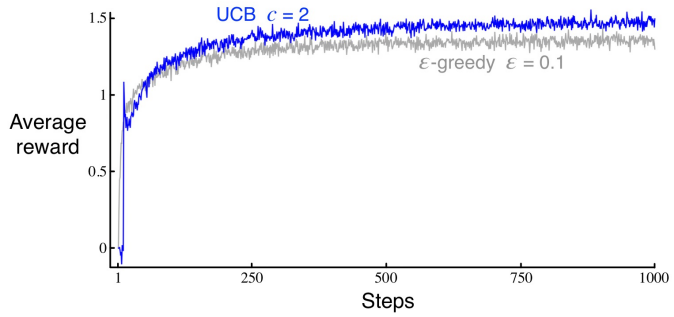


Figure 2.4: Average performance of UCB action selection on the 10-armed testbed. As shown, UCB generally performs better than ε-greedy action selection, except in the first k steps, when it selects randomly among the as-yet-untried actions.

Remark

Two difficulties of UCB (the author doesn't explain why)

- Difficulty of dealing with nonstationary problems
- Difficulty of large state space

2-8 Gradient Bandit Algorithms

Goal: We want to learn a preference of decision making.

$H_t(a)$:= numerical preference for a at time t

We define the soft-max distribution :

$$\pi_t(a) := \mathbb{P}(A_t = a) = \frac{e^{H_t(b)}}{\sum_{b=1}^{k} e^{H_t(b)}}$$

The learning algorithm:

$$\begin{aligned} H_{t+1}(A_t) &\doteq H_t(A_t) + \alpha\big(R_t - \bar{R}_t\big)\big(1 - \pi_t(A_t)\big), && \text{and} \\ H_{t+1}(a) &\doteq H_t(a) - \alpha\big(R_t - \bar{R}_t\big)\pi_t(a), && \text{for all } a \neq A_t, \end{aligned} \tag{2.12}$$

where $\bar{R}_t = \frac{1}{t-1}\sum_{i=1}^{t-1} R_i$.

Or equivalently,

$$H_{t+1}(A_t) = H_t(A_t) + \alpha(R_t - \bar{R}_t)(\mathbb{1}_{a=A_t} - \pi_t(a)).$$

The idea of the algorithm is based on stochastic gradient ascent.

* stochastic gradient ascent: find the maximum of a known function.
  ↳ theoretically work

Now we want to show that the algorithm is an instance of stochastic gradient ascent.

pf. In SGA, we update H_{t+1} as follows:

$$H_{t+1}(a) \doteq H_t(a) + \alpha \frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)} \quad (2.13)$$

and

$$\mathbb{E}[R_t] = \sum_x \pi_t(x) q_*(x)$$

Note that we don't know $q_*(x)$ when we perform gradient bandit.

Calculate the partial derivative:

$$\begin{aligned}
\frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)} &= \frac{\partial}{\partial H_t(a)}\left[\sum_x \pi_t(x) q_*(x)\right] \\
&= \sum_x q_*(x) \frac{\partial \pi_t(x)}{\partial H_t(a)} \\
&= \sum_x (q_*(x) - B_t) \frac{\partial \pi_t(x)}{\partial H_t(a)},
\end{aligned}$$

B_t: baseline

Since $\sum_x \pi_t(x) = 1$,

$$\frac{\partial}{\partial H_t(a)} \sum_x \pi_t(x) = \sum_x \frac{\partial \pi_t(x)}{\partial H_t(a)} = 0.$$

Thus, we have

$$\frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)} = \sum_x \pi_t(x)(q_*(x) - B_t) \frac{\partial \pi_t(x)}{\partial H_t(a)} / \pi_t(x).$$

$$\begin{aligned}
&= \mathbb{E}\left[(q_*(A_t) - B_t) \frac{\partial \pi_t(A_t)}{\partial H_t(a)} / \pi_t(A_t)\right] \\
&= \mathbb{E}\left[(R_t - \bar{R}_t) \frac{\partial \pi_t(A_t)}{\partial H_t(a)} / \pi_t(A_t)\right],
\end{aligned}$$

$q_*(A_t) = \mathbb{E}[R_t | A_t = a]$. iterated expectation!!

(A)

Now we claim that

$$\frac{\partial \pi_t(x)}{\partial H_t(a)} = \pi_t(x)\left(\mathbb{1}_{a=x} - \pi_t(a)\right).$$

$$\begin{aligned}
\frac{\partial \pi_t(x)}{\partial H_t(a)} &= \frac{\partial}{\partial H_t(a)} \pi_t(x) \\
&= \frac{\partial}{\partial H_t(a)} \left[\frac{e^{H_t(x)}}{\sum_{y=1}^{k} e^{H_t(y)}} \right] \\
&= \frac{\frac{\partial e^{H_t(x)}}{\partial H_t(a)} \sum_{y=1}^{k} e^{H_t(y)} - e^{H_t(x)} \frac{\partial \sum_{y=1}^{k} e^{H_t(y)}}{\partial H_t(a)}}{\left(\sum_{y=1}^{k} e^{H_t(y)}\right)^2} && \text{(by the quotient rule)} \\
&= \frac{\mathbb{1}_{a=x} e^{H_t(x)} \sum_{y=1}^{k} e^{H_t(y)} - e^{H_t(x)} e^{H_t(a)}}{\left(\sum_{y=1}^{k} e^{H_t(y)}\right)^2} && \text{(because } \tfrac{\partial e^x}{\partial x} = e^x\text{)} \\
&= \frac{\mathbb{1}_{a=x} e^{H_t(x)}}{\sum_{y=1}^{k} e^{H_t(y)}} - \frac{e^{H_t(x)} e^{H_t(a)}}{\left(\sum_{y=1}^{k} e^{H_t(y)}\right)^2} \\
&= \mathbb{1}_{a=x} \pi_t(x) - \pi_t(x)\pi_t(a) \\
&= \pi_t(x)\left(\mathbb{1}_{a=x} - \pi_t(a)\right). && \text{Q.E.D.}
\end{aligned}$$

Plug the result into (Δ). We get

$$\begin{aligned}
&= \mathbb{E}\left[(R_t - \bar{R}_t)\pi_t(A_t)\left(\mathbb{1}_{a=A_t} - \pi_t(a)\right)/\pi_t(A_t)\right] \\
&= \mathbb{E}\left[(R_t - \bar{R}_t)\left(\mathbb{1}_{a=A_t} - \pi_t(a)\right)\right].
\end{aligned}$$

However, we don't know $(R_t - \bar{R}_t)(\mathbb{1}_{a=A_t} - \pi_t(a))$ in reality, so we simply plug the above expectation into (2.13). Then we've showed that

the algorithm is an instance of stochastic gradient ascent.