Reinforcement learning

Dr. Aissa Boulmerka

The National School of Artificial Intelligence
aissa.boulmerka@ensia.edu.dz

2024-2025

Chapter 2
Multi-armed Bandits

Outline

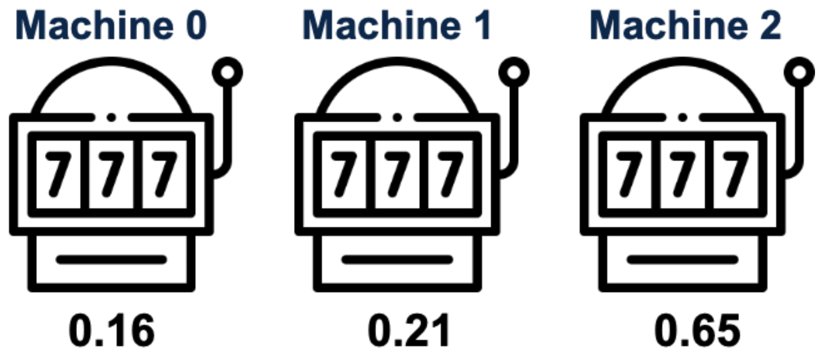
- Introduction
- k-armed Bandit Problem
- Estimating Action-Values
- Incremental Implementation
- Exploration and Exploitation Trade-off
- Summary

Introduction

- In **reinforcement learning**, the agent generates its own training data by **interacting with the world**.
- The agent must learn the consequences of his own actions through **trial and error**, rather than being told the correct action.
- In this chapter, we will study this evaluative aspect of reinforcement learning. We will focus on the problem of **decision-making** in a simplified setting called **bandits**.

K-armed bandit problem

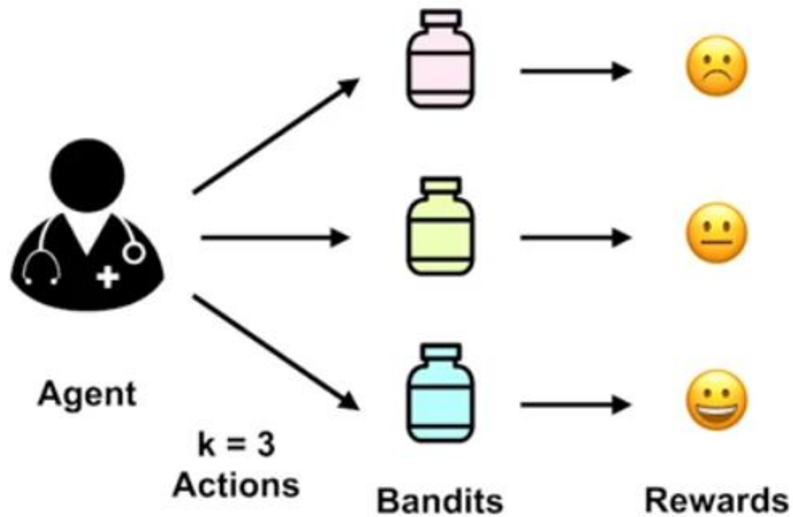
k-armed Bandit Problem



- In the **k-armed bandit problem**, we have a **decision-maker** or **agent**, who chooses between "k" different options or **actions**, and receives a numerical **reward** chosen from a stationary probability distribution based on the action it selects.
- The objective is to **maximize** the expected **total reward** over some time period, for example, over 1000 action selections, or **time steps**.

k-armed Bandit Problem

- Imagine the next problem where the doctor has to **give medicine** to a patient and based on which one he choose, he will receive a **reward** (cure the patient).
- For the doctor to decide which **action** is best, we must define the value of taking each action. We call these values the **action values** or the **action value function**.



Action-Values

- We call these values the **Action-Values** or the **action value function**. The **value** is the **expected reward.**
- We denote the action selected on time step t as A_t, and the corresponding reward as R_t.

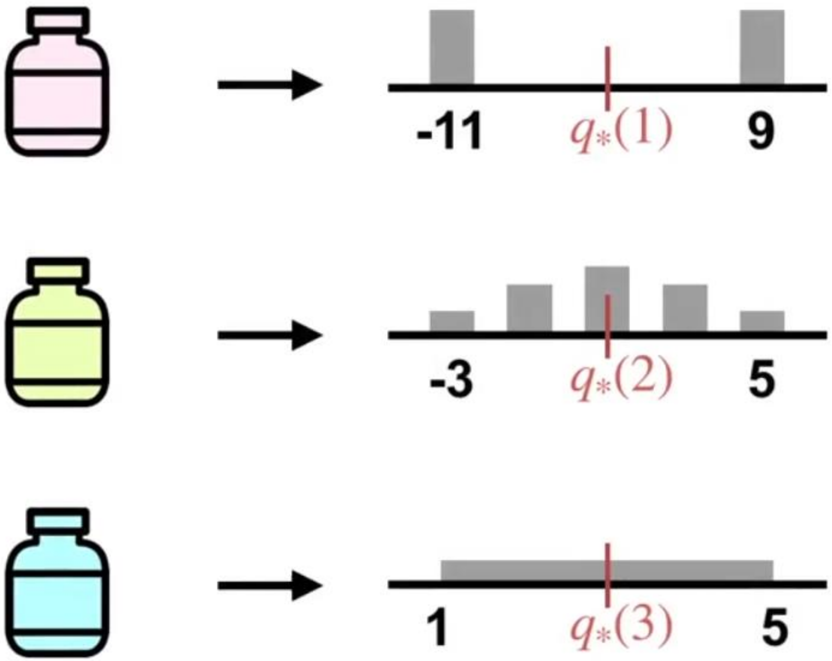
$$q_*(a) \doteq \mathbb{E}[R_t \mid A_t = a\,] \qquad \forall a \in \{1, \cdots, k\}$$

$$= \sum_r p(r|a)r$$

- The value of an arbitrary action a, denoted $q_*(a)$, is the **expected reward** given that a is selected:
- The goal is to **maximize** the **expected reward**

$$\underset{a}{\operatorname{argmax}}\, q_*(a)$$

Calculating $q_*(a)$



$$q_*(a) = 0.5 \times -11 + 0.5 \times 9 = -1$$

$$q_*(a) = 1$$

$$q_*(a) = 3$$

Some applications of K-armed bandit problem

1. Online advertising
2. Clinical trials
3. Recommendation systems
4. Portfolio optimization in finance
5. Dynamic pricing in e-commerce
6. Traffic routing and navigation
7. Fraud detection

Estimating Action-Values

Sample-Average Method

- The **sample-average method** is a method for **estimating the action values**. We will use this method to **compute the value** of each treatment in our medical trial example.

- The **value** of selecting an **action** q^* is the expected reward received after that action has been taken.

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

- q^* **is not known to the agent**, just like the doctor doesn't know the effectiveness of each treatment.

- Instead, we will need to find a way to **estimate it**.

Sample-Average Method

- One way to **estimate** q^* is to compute a **sample-average**.

$$Q_t(a) \doteq \frac{\text{sum of rewards when a taken prior to t}}{\text{number of times a taken prior to t}}$$

$$= \frac{\sum_{i=1}^{t-1} R_i}{(t-1)}$$

Medical trial example

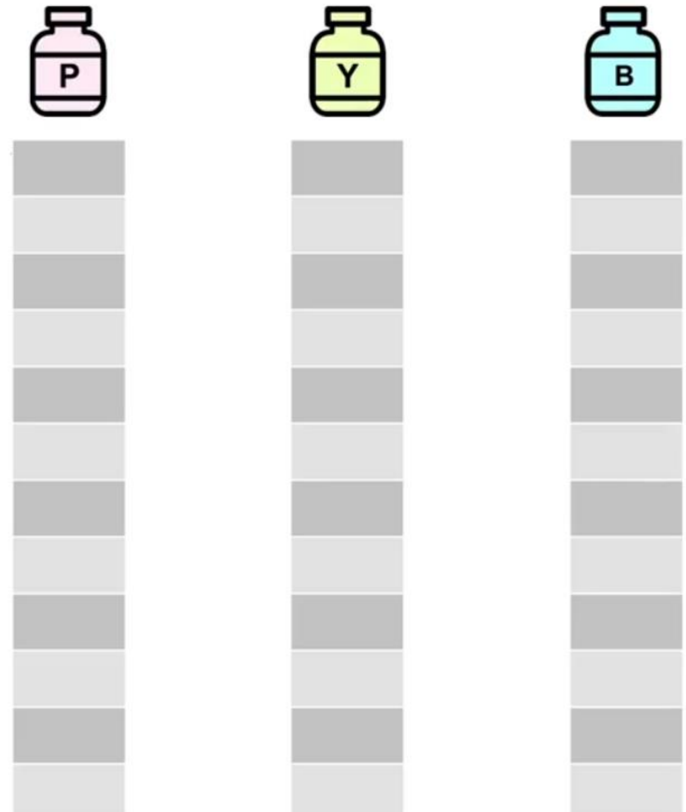
- **A reward of 1 if the treatment succeeds otherwise 0.**

Value	**0.25**	**0.75**	**0.50**

Medical trial example

A reward of 1 if the treatment succeeds otherwise 0.

$$\boldsymbol{Q_t(a) = \frac{\sum_{i=1}^{t-1} R_i}{(t-1)}}$$

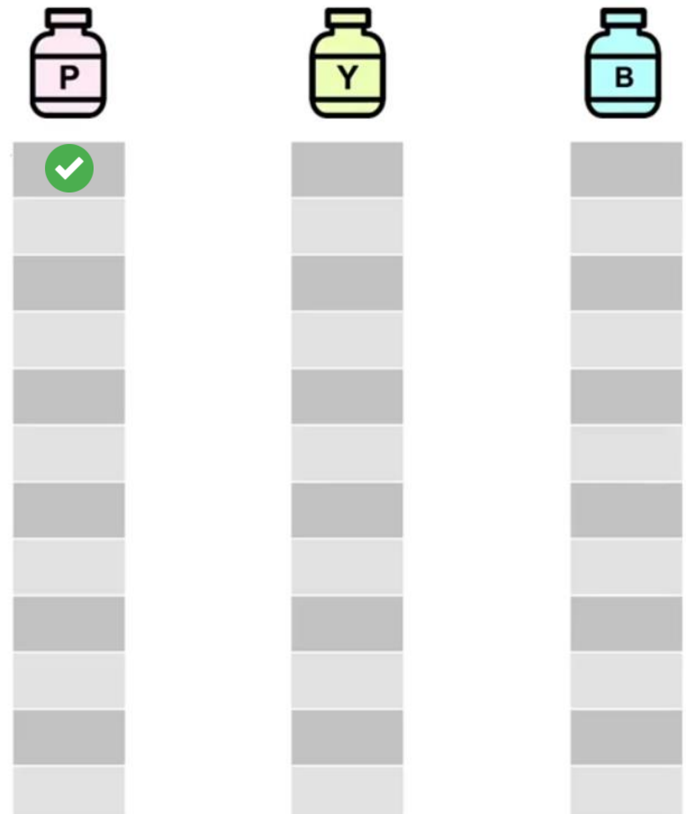


$Q_0(\text{P}) = \mathbf{0.0}$ $Q_0(\text{Y}) = \mathbf{0.0}$ $Q_0(\text{B}) = \mathbf{0.0}$

Medical trial example

A reward of 1 if the treatment succeeds otherwise 0.

$$\boldsymbol{Q_t(a) = \frac{\sum_{i=1}^{t-1} R_i}{(t-1)}}$$

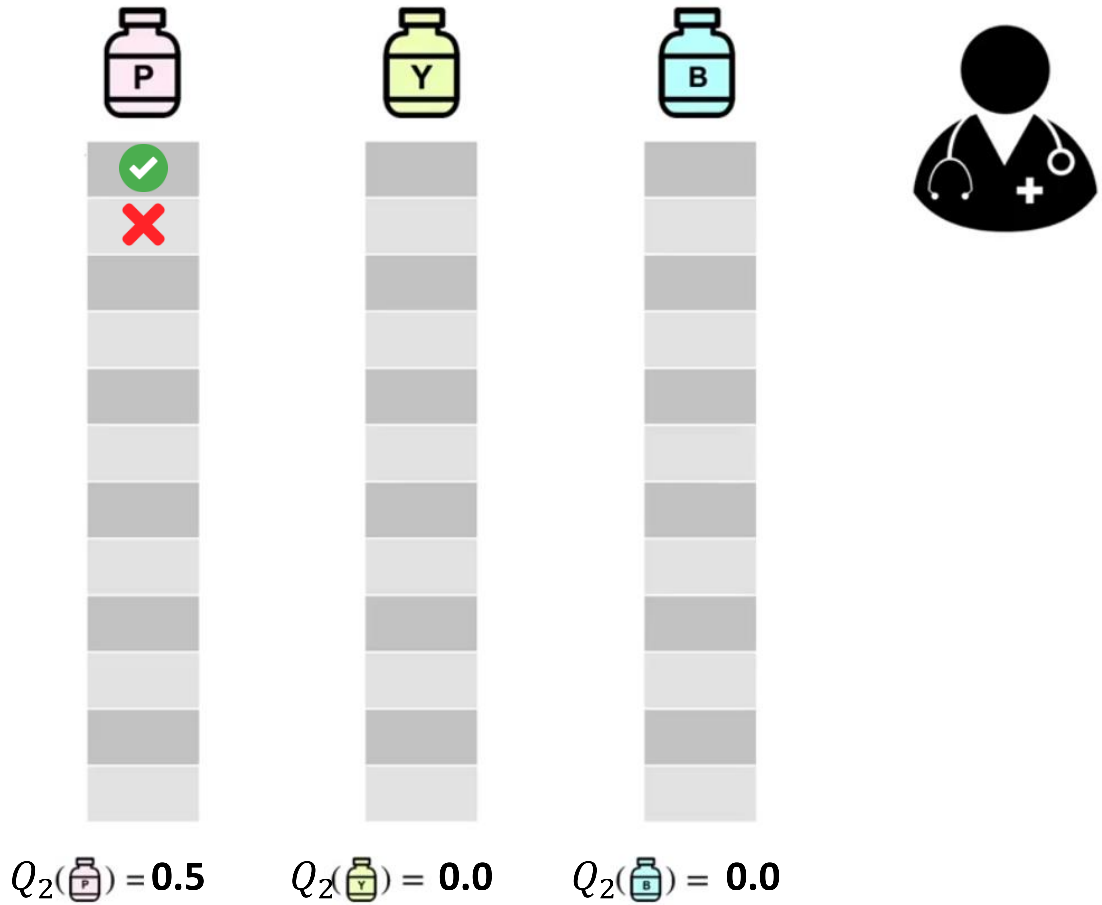


$Q_1(\text{P}) = \mathbf{1.0}$ $Q_1(\text{Y}) = \mathbf{0.0}$ $Q_1(\text{B}) = \mathbf{0.0}$

Medical trial example

A reward of 1 if the treatment succeeds otherwise 0.

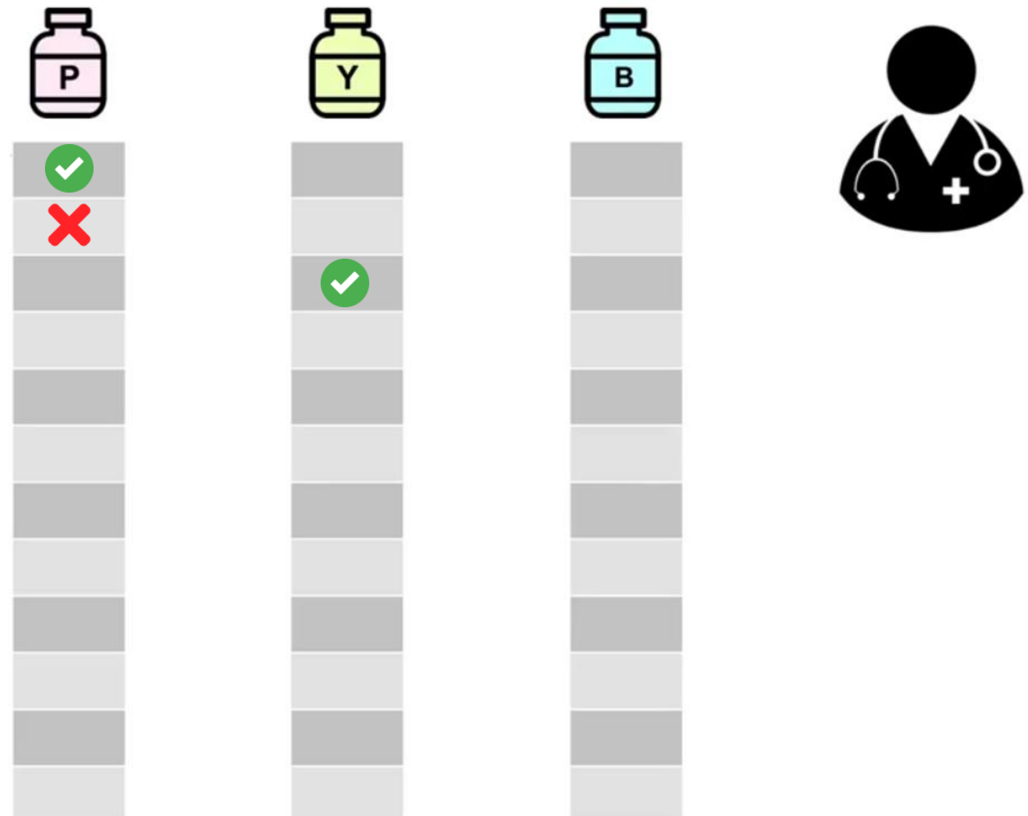
$$Q_t(a) = \frac{\sum_{i=1}^{t-1} R_i}{(t-1)}$$



Medical trial example

A reward of 1 if the treatment succeeds otherwise 0.

$$Q_t(a) = \frac{\sum_{i=1}^{t-1} R_i}{(t-1)}$$

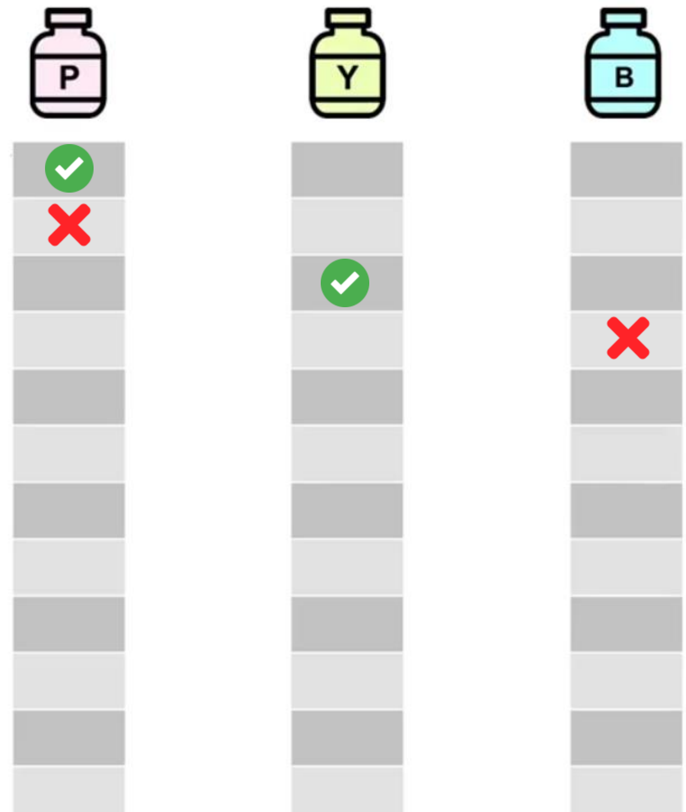


$Q_3(\text{P}) =$ **0.5** $Q_3(\text{Y}) =$ **1.0** $Q_3(\text{B}) =$ **0.0**

Medical trial example

A reward of 1 if the treatment succeeds otherwise 0.

$$Q_t(a) = \frac{\sum_{i=1}^{t-1} R_i}{(t-1)}$$

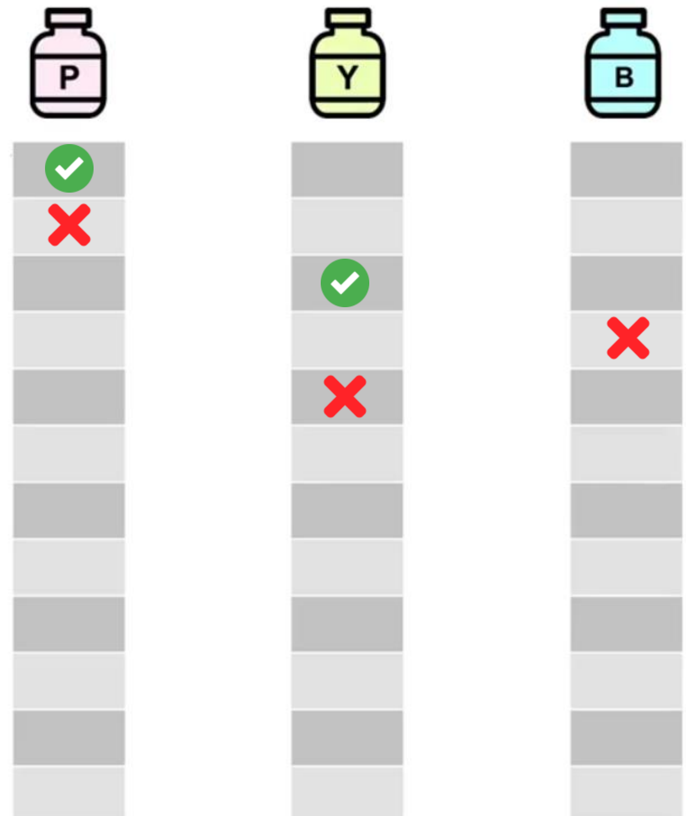


Q_4(P) = **0.5** Q_4(Y) = **1.0** Q_4(B) = **0.0**

Medical trial example

A reward of 1 if the treatment succeeds otherwise 0.

$$Q_t(a) = \frac{\sum_{i=1}^{t-1} R_i}{(t-1)}$$

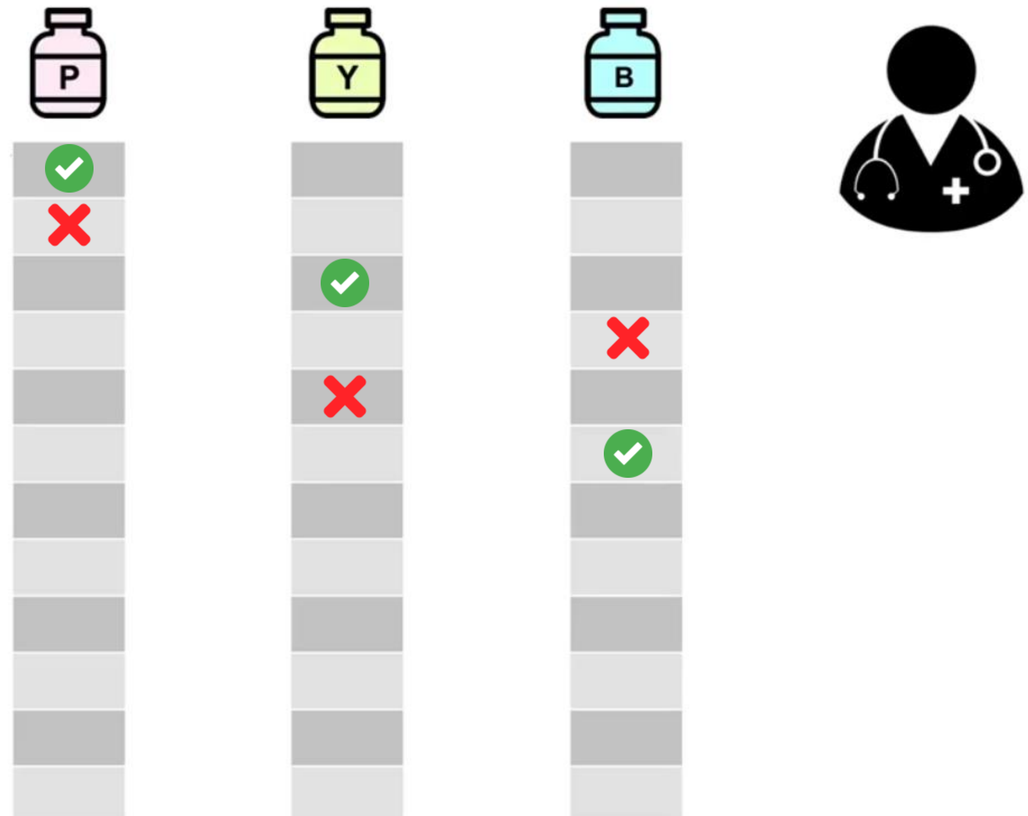


$Q_5(\text{P}) = \mathbf{0.5}$ $Q_5(\text{Y}) = \mathbf{0.5}$ $Q_5(\text{B}) = \mathbf{0.0}$

Medical trial example

A reward of 1 if the treatment succeeds otherwise 0.

$$Q_t(a) = \frac{\sum_{i=1}^{t-1} R_i}{(t-1)}$$

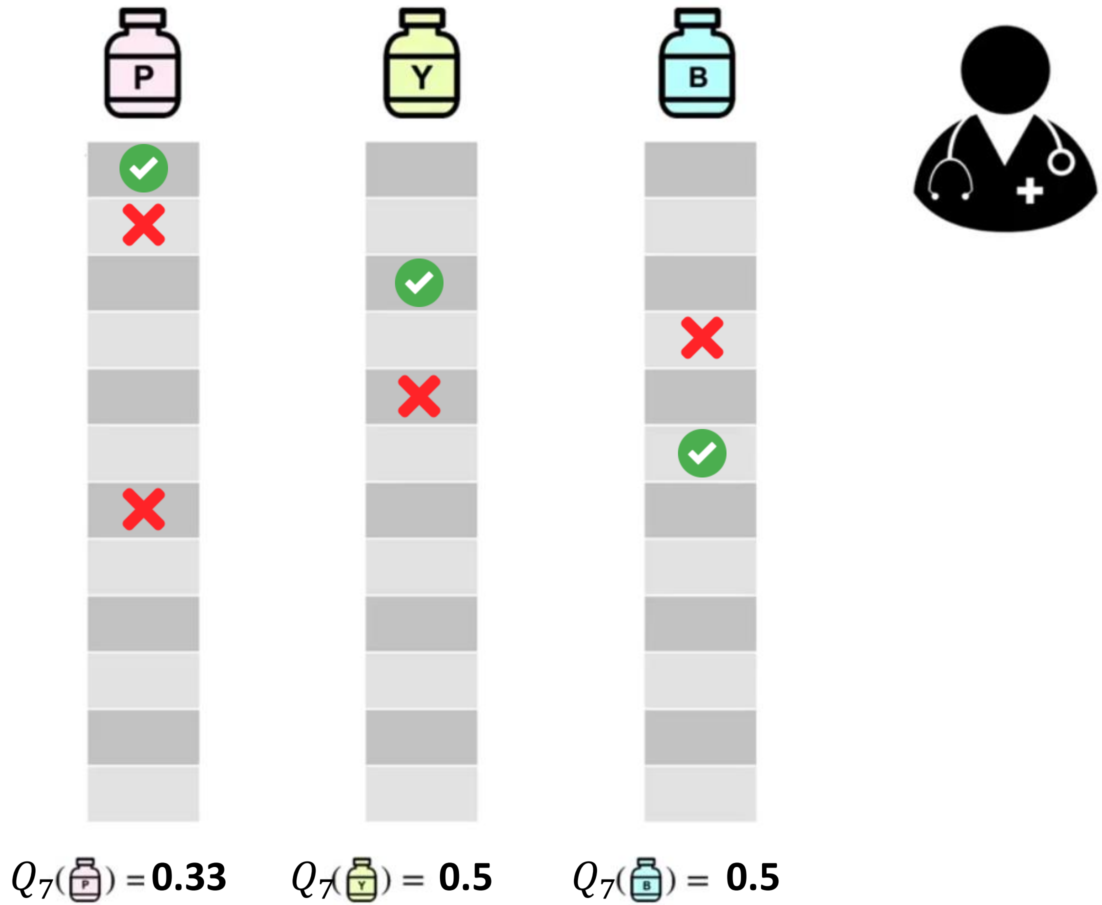


Q_6(P) = **0.5** Q_6(Y) = **0.5** Q_6(B) = **0.5**

Medical trial example

A reward of 1 if the treatment succeeds otherwise 0.

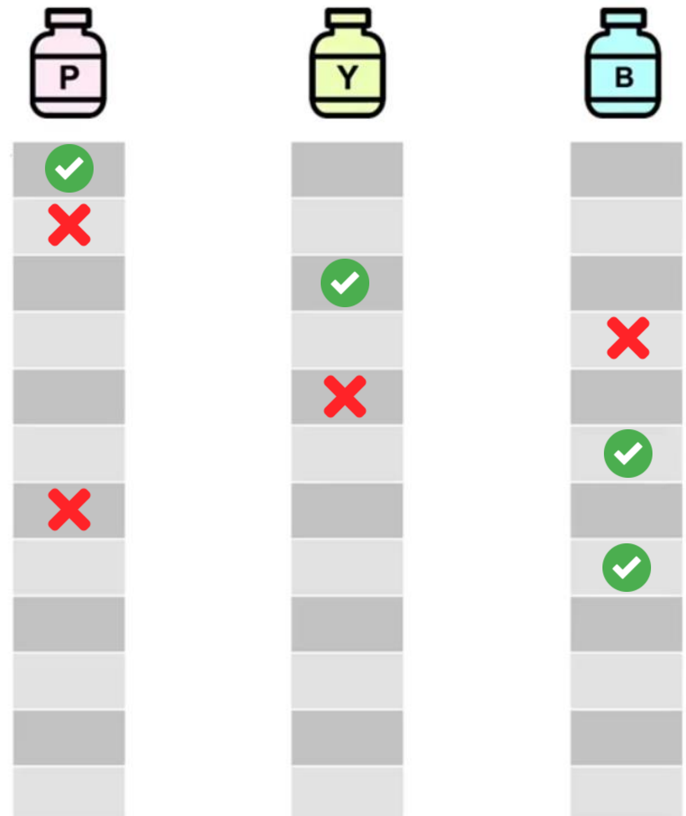
$$Q_t(a) = \frac{\sum_{i=1}^{t-1} R_i}{(t-1)}$$



Medical trial example

A reward of 1 if the treatment succeeds otherwise 0.

$$Q_t(a) = \frac{\sum_{i=1}^{t-1} R_i}{(t-1)}$$

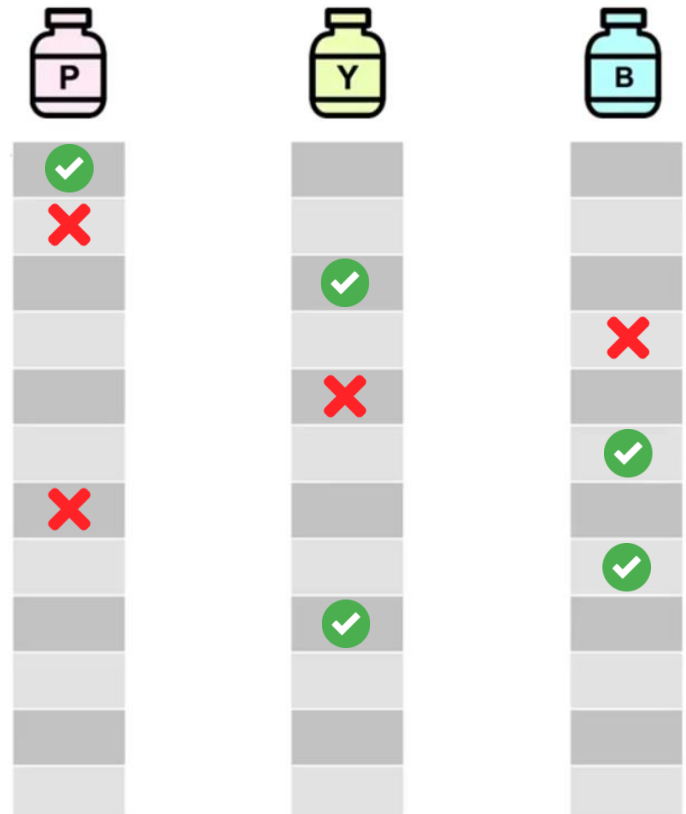


$Q_8(\text{P}) =$ **0.33** $Q_8(\text{Y}) =$ **0.5** $Q_8(\text{B}) =$ **0.66**

Medical trial example

A reward of 1 if the treatment succeeds otherwise 0.

$$Q_t(a) = \frac{\sum_{i=1}^{t-1} R_i}{(t-1)}$$

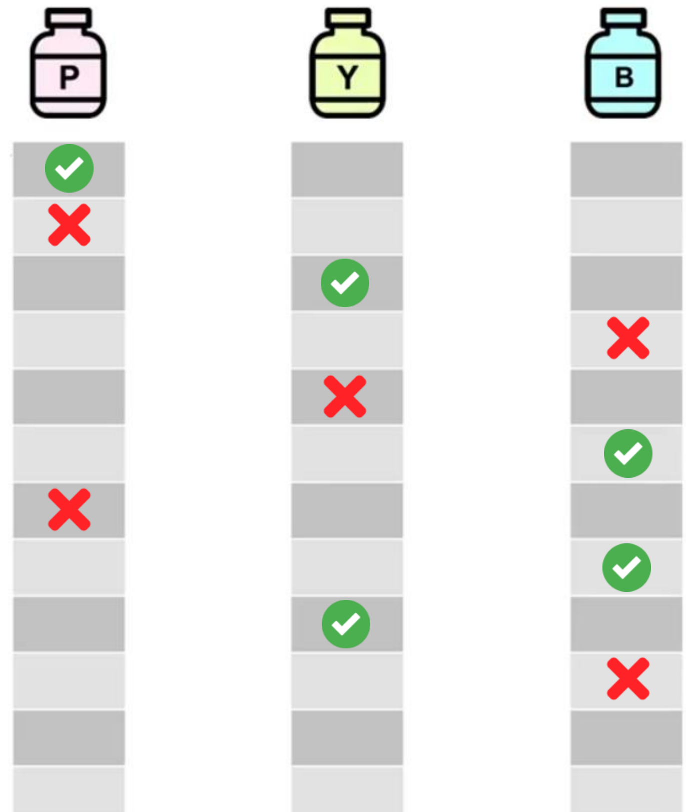


$Q_9(\text{P}) =$ **0.33** $\quad Q_9(\text{Y}) =$ **0.66** $\quad Q_9(\text{B}) =$ **0.66**

Medical trial example

A reward of 1 if the treatment succeeds otherwise 0.

$$Q_t(a) = \frac{\sum_{i=1}^{t-1} R_i}{(t-1)}$$

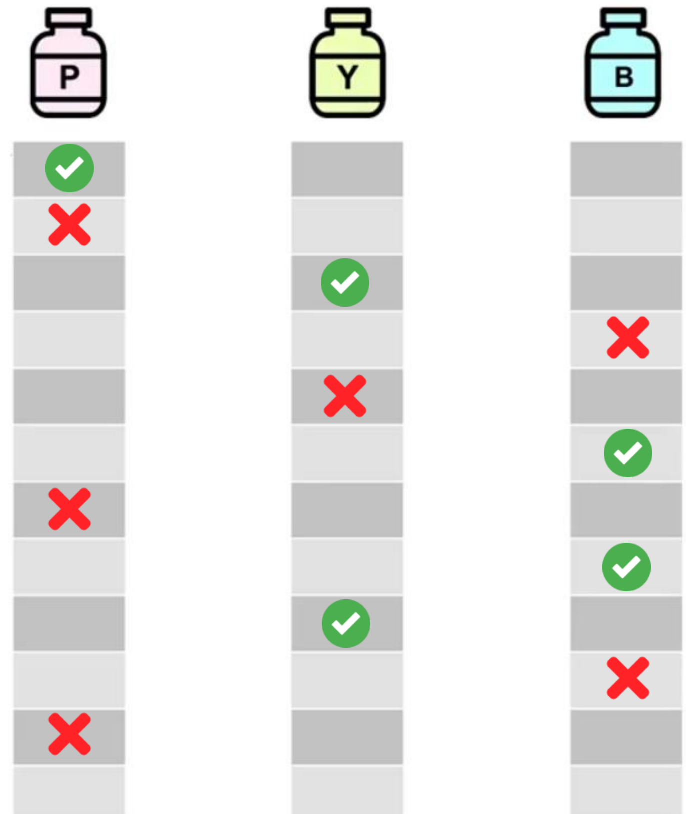


$Q_{10}(\text{P}) =$ **0.33** $\quad Q_{10}(\text{Y}) =$ **0.66** $\quad Q_{10}(\text{B}) =$ **0.5**

Medical trial example

A reward of 1 if the treatment succeeds otherwise 0.

$$Q_t(a) = \frac{\sum_{i=1}^{t-1} R_i}{(t-1)}$$

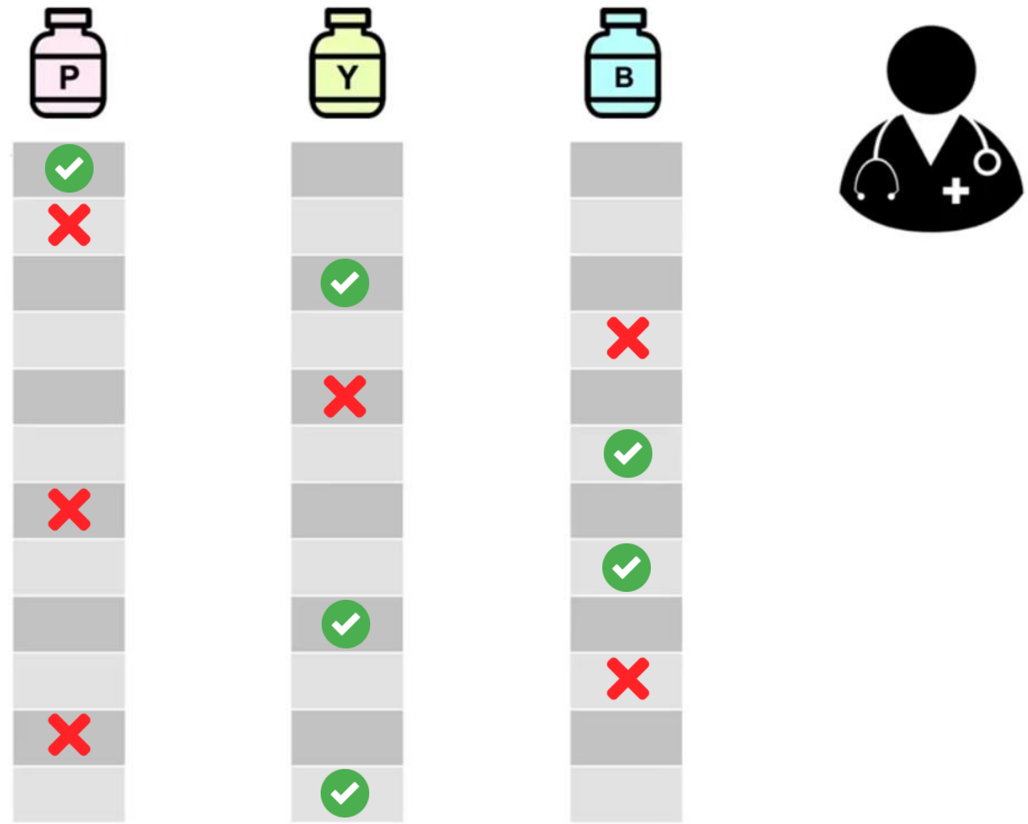


$Q_{11}(\text{P}) =$ **0.25** $Q_{11}(\text{Y}) =$ **0.66** $Q_{11}(\text{B}) =$ **0.5**

Medical trial example

A reward of 1 if the treatment succeeds otherwise 0.

$$Q_t(a) = \frac{\sum_{i=1}^{t-1} R_i}{(t-1)}$$



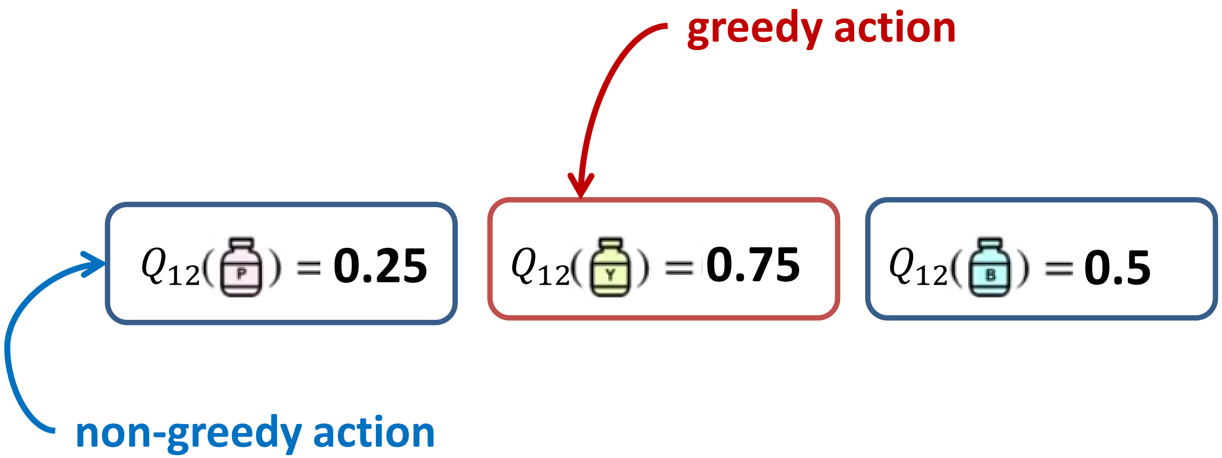
Q_{12}(P) = **0.25** Q_{12}(Y) = **0.75** Q_{12}(B) = **0.5**

Action Selection

- The simplest **action selection** rule is to select one of the actions with the **highest estimated value**.
- We call this method of choosing actions **greedy**. The **greedy action** is the action that currently has the **largest estimated value**.
- Selecting the **greedy action** means the agent is **exploiting** its current knowledge. It is trying to get the most reward it can right now.
- If there is more than one greedy action, then a selection is made among them in some arbitrary way, perhaps **randomly**.
- We write this **greedy** action selection method as:

$$A_t \doteq \underset{a}{argmax}\, Q_t(a)$$

Action Selection



$$A_t \doteq \underset{a}{argmax}\, Q_t(a)$$

ε-greedy action selection

- Greedy action selection maximizes **immediate rewards**, without **exploring** less promising options that might **improve the solution**.
- Instead of that, we can behave **greedily most of the time**, but every once with small probability epsilon (ε), select randomly from **among all the actions** with equal probability, independently of the action-value estimates.
- We call methods using this near-greedy action selection rule ***ε* -greedy methods**.
- An advantage of these methods is that, in the limit as the number of steps increases, every action will be sampled an infinite number of times, thus ensuring that all the $Q_t(a)$ **converges** to $q_*(a)$.
- This of course implies that the probability of **selecting the optimal action** converges to greater than $1-\varepsilon$, that is, to near **certainty**.

ε-greedy action selection

- ε-greedy action:

$$A_t = \begin{cases} \underset{a}{\boldsymbol{argmax}}\, \boldsymbol{Q_t(a)} & \boldsymbol{with\ probability\ 1-\varepsilon} \\ \boldsymbol{randomly\ selected\ action} & \boldsymbol{with\ probability\ \varepsilon} \end{cases}$$

Exercise:

- 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?

Answer:

$$\text{Action} = \begin{cases} 50\%\ \text{Exploit} - \text{Greedy} \\ 50\%\ \text{Explore} \begin{cases} 50\%\ \text{Greedy} \\ 50\%\ \text{Explore} \end{cases} \end{cases} = \begin{cases} 75\%\ \text{Greedy} \\ 25\%\ \text{Explore} \end{cases}$$

Incremental Implementation

Incremental Implementation

- The **action-value** methods we have discussed so far all estimate action values as **sample averages** of observed rewards.
- How these averages can be computed in a **computationally efficient** manner with constant memory and constant per-time-step computation?
- Let R_i denote the **reward** received after the ith selection of one action.
- Let Q_n denote the **estimate** of its action value after it has been selected $n-1$ times, which can be written simply as

$$Q_n \doteq \frac{R_1 + R_2 + \cdots + R_{n-1}}{\mathrm{n}-1}.$$

Incremental update rule

$$
\begin{aligned}
Q_{n+1} &\doteq \frac{1}{\mathrm{n}}\sum_{i=1}^{n} R_i \\
&= \frac{1}{\mathrm{n}}\left(R_n + \sum_{i=1}^{n-1} R_i\right) \\
&= \frac{1}{\mathrm{n}}\left(R_n + (n-1)\frac{1}{n-1}\sum_{i=1}^{n-1} R_i\right) \\
&= \frac{1}{\mathrm{n}}\left(R_n + (n-1)Q_n\right) \\
&= \boldsymbol{Q_n} + \frac{\mathbf{1}}{\mathbf{n}}[\boldsymbol{R_n} - \boldsymbol{Q_n}]
\end{aligned}
$$

Recall:

$$
Q_n = \frac{1}{n-1}\sum_{i=1}^{n-1} R_i
$$

- This implementation requires memory only for Q_n and n, and only the previous small computation for each new reward.

Incremental update rule

- The general form is

$$NewEstimate \leftarrow OldEstimate + StepSize[Target - OldEstimate]$$

$$Q_{n+1} = Q_n + \frac{1}{\mathrm{n}}[R_n - Q_n]$$

- The expression $[Target - OldEstimate]$ is an **error** in the estimate. It is reduced by taking a step toward the $Target$.

- The **target** is presumed to indicate a desirable **direction** in which to move. In the case above, for example, the target is the nth reward.

- We denote the step-size parameter by $\alpha = \frac{1}{n}$ or, more generally, by $\alpha_n \to [0,1]$.

A simple Bandit algorithm

- Pseudo-code for a complete **bandit algorithm** using incrementally computed **sample averages** and ε-greedy action selection.
- The function **bandit(a)** is assumed to take an action and return a corresponding reward.

A simple bandit algorithm

Initialize, for $a = 1$ to k:

$Q(a) \leftarrow 0$

$N(a) \leftarrow 0$

Loop forever:

$$A \leftarrow \begin{cases} \arg\max_a Q(a) & \text{with probability } 1-\varepsilon \quad \text{(breaking ties randomly)} \\ \text{a random action} & \text{with probability } \varepsilon \end{cases}$$

$R \leftarrow bandit(A)$

$N(A) \leftarrow N(A) + 1$

$Q(A) \leftarrow Q(A) + \frac{1}{N(A)}\left[R - Q(A)\right]$

Non-stationary Bandit Problem

- The **averaging methods** discussed so far are appropriate for **stationary bandit problems**, that is, for bandit problems in which the reward probabilities **do not change over time**.
- We often encounter reinforcement learning problems that are effectively **non-stationary**. In such cases it makes sense to give **more weight to recent rewards than to long-past rewards**.
- One of the most popular ways of doing this is to use a **constant step-size** parameter.
- For example, the **incremental update rule** for updating an average Q_n of the $n-1$ past rewards is modified to be

$$Q_{n+1} = Q_n + \alpha_n [R_n - Q_n]$$

where the step-size parameter $\alpha_n \in (0, 1]$ is constant.

Non-stationary Bandit Problem

- This results in Q_{n+1} being a **weighted average** of past rewards and the initial estimate Q_1
- Proof:

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

Non-stationary Bandit Problem

$$Q_{n+1} = (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i$$

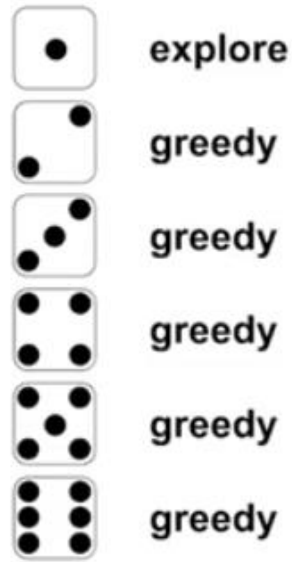
- The **updating equation** can be written as the initial value of Q plus a **weighted sum of the rewards** over time.
- The first term tells us that the contribution of Q_1 **decreases exponentially** with time.
- The second term tells us that the rewards further back in time **contribute exponentially less to the sum**.
- The influence of our **initialization** of Q goes to zero with **more and more data**.
- The most recent rewards **contribute most** to our current estimate.

Exploration and exploitation trade-off

Exploration and Exploitation

- **Exploration** allows the agent to improve his knowledge about each action. Hopefully, leading to **long-term benefit**.
- By **improving the accuracy** of the estimated action values, the agent can make more informed **decisions** in the future.
- **Exploitation** on the other hand, exploits the agent's current estimated values. It chooses the **greedy action** to try to get the **most reward**.
- But by being greedy with respect to estimated values, may not get the most reward.
- How do we choose **when to explore**, and **when to exploit**?

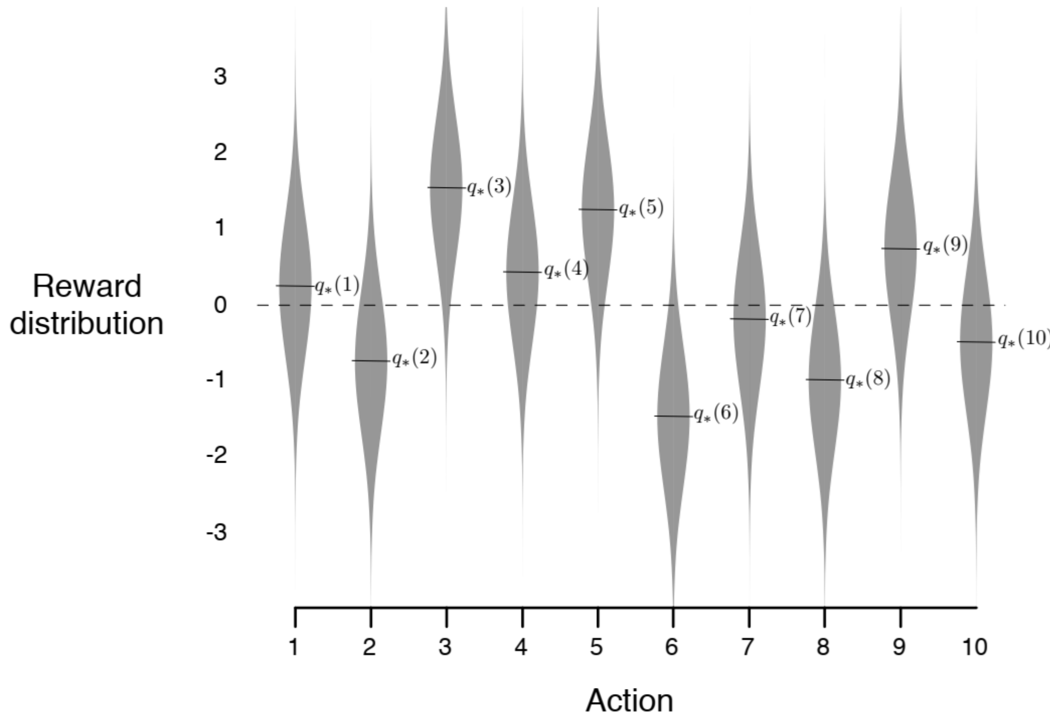
Epsilon-Greedy Action Selection



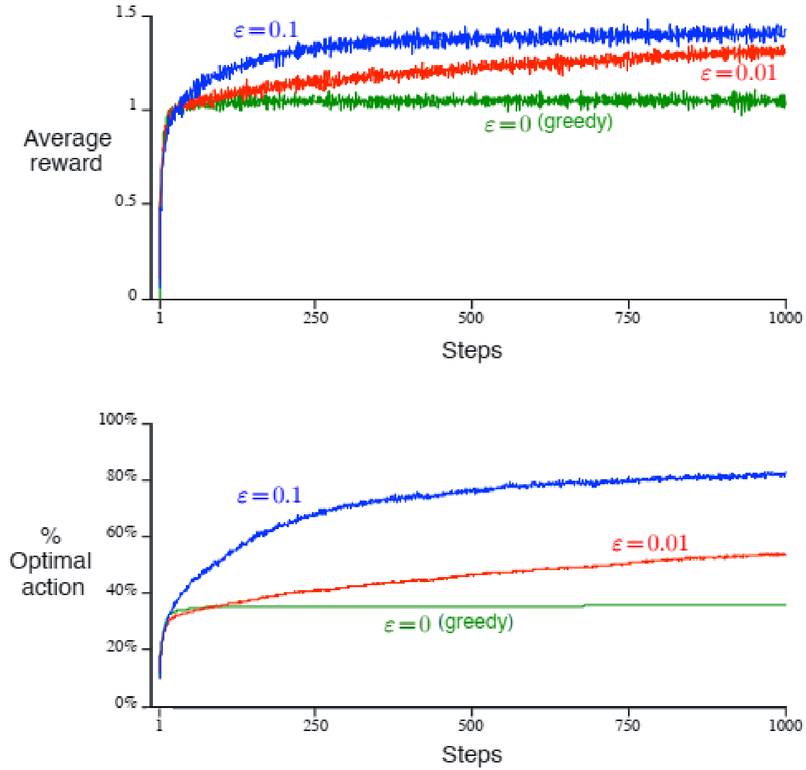
Epsilon-Greedy Action Selection.

The 10-armed Testbed

- We take a set of **2000 randomly generated** k-armed bandit problems with $k = 10$.
- For each bandit problem, **the action values**, $q_*(a), a = 1, \cdots, 10$ were selected according to a **normal (Gaussian) distribution** with mean 0 and variance 1
- The **reward**, R_t, was selected from a **normal distribution** with mean $q_*(A_t)$ and variance 1.



The 10-armed Testbed

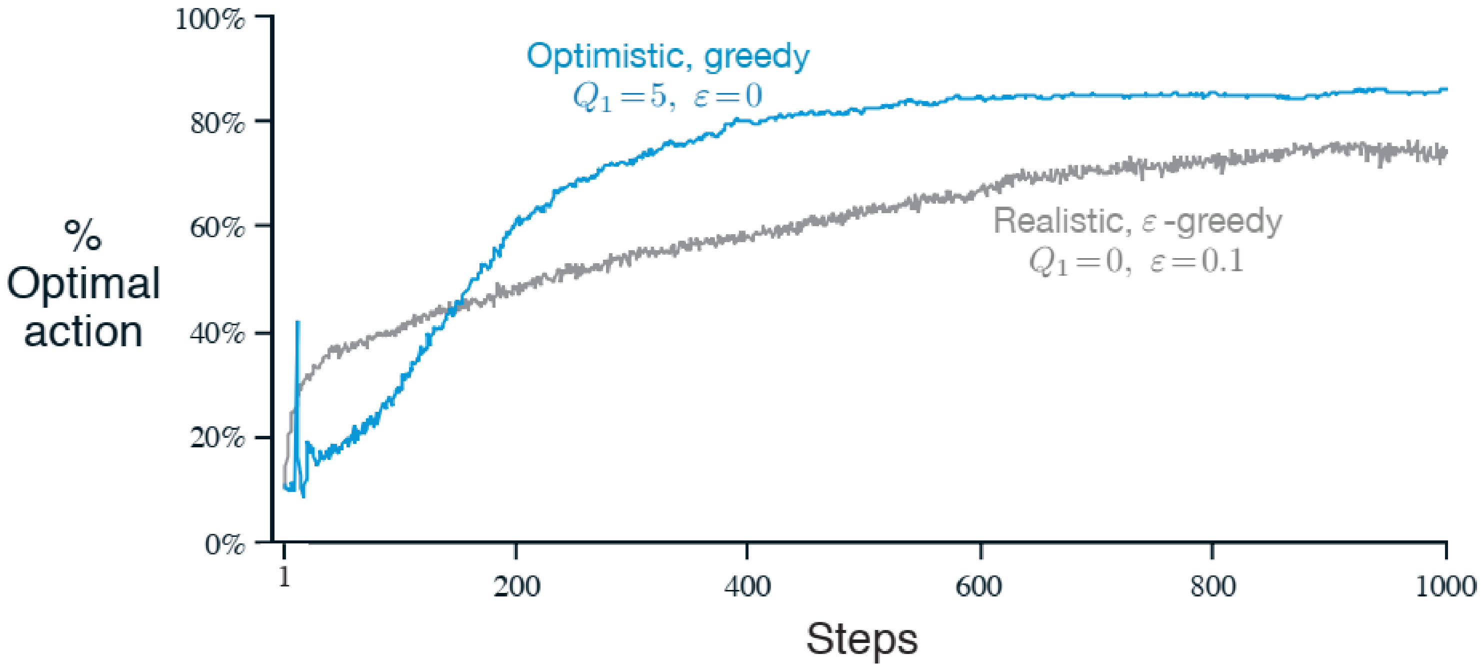


Average performance of ε-greedy action-value methods on the 10-armed testbed. These data are averages over 2000 runs with different bandit problems. All methods used sample averages as their action-value estimates.

Optimistic Initial Values

- The methods above are dependent to some extent on the **initial action-value estimates**, $Q_1(a)$.
- For the **sample-average methods ($\boldsymbol{\alpha_n(a) = 1/n}$)**, the bias disappears once all actions have been selected at least once.
- For methods with **constant $\boldsymbol{\alpha}$**, the bias is permanent.
- The **downside** of this bias is that the initial estimates become a set of parameters that must be picked by the user, if only to set them all to zero.
- The **upside** is that they provide an easy way to supply some prior knowledge about what level of rewards can be expected.
- Initial action values can also be used as a simple way to encourage exploration.

Optimistic Initial Values



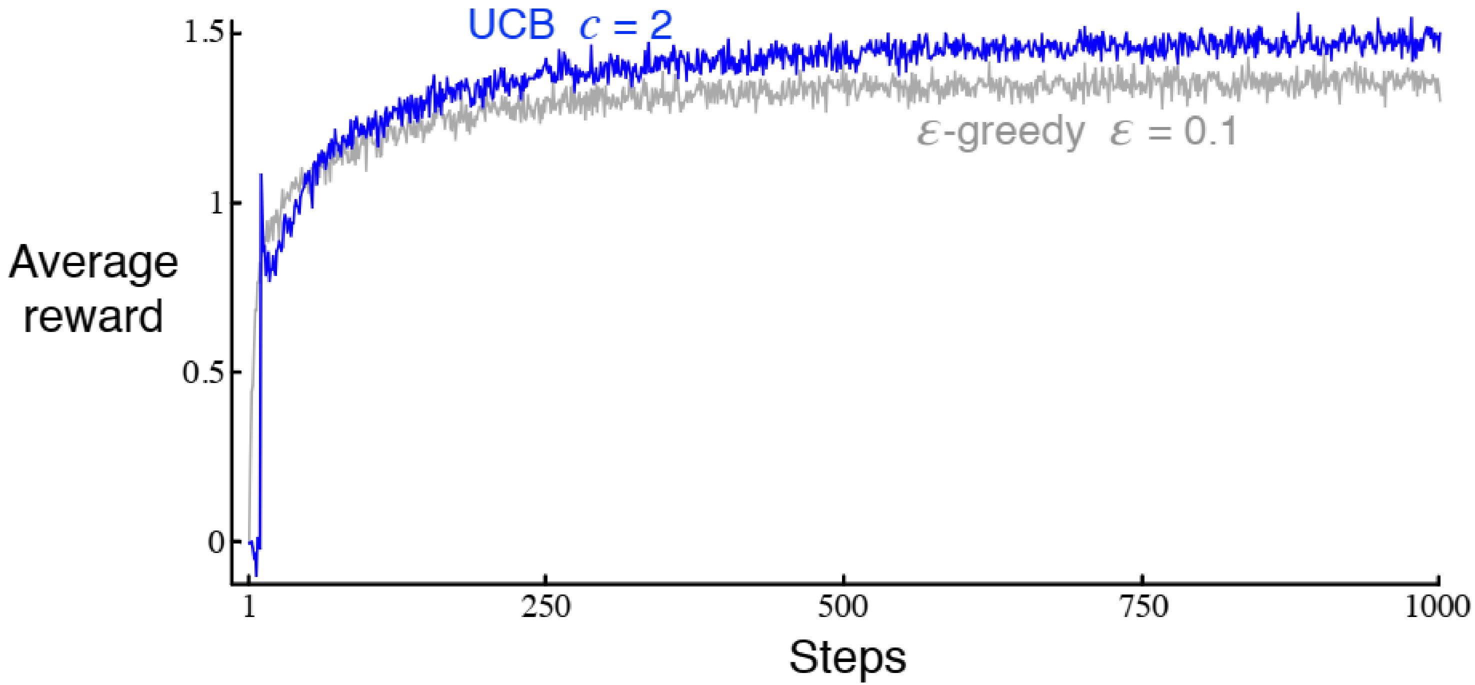
Upper-Confidence-Bound (UCB) Action Selection

- ε-greedy action selection forces the **non-greedy actions to be tried**, but **indiscriminately**.
- It would be better to select among the non-greedy actions according to their **potential for actually being optimal**.
- One effective way of doing this is to select actions according to :

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

- $N_t(a)$ denotes the number of times that action a has been selected prior to time t.
- $c > 0$ controls the degree of exploration.
- **Difficulty 1** : non-stationary problems.
- **Difficulty 2** : large state spaces.

Upper-Confidence-Bound (UCB) Action Selection



Gradient Bandit Algorithms

- We define a **preference** $H_t(a)$ for each action a.
- The **larger the preference**, the **more often that action is taken**.
- We determine the action probabilities, according to the **Softmax distribution** :

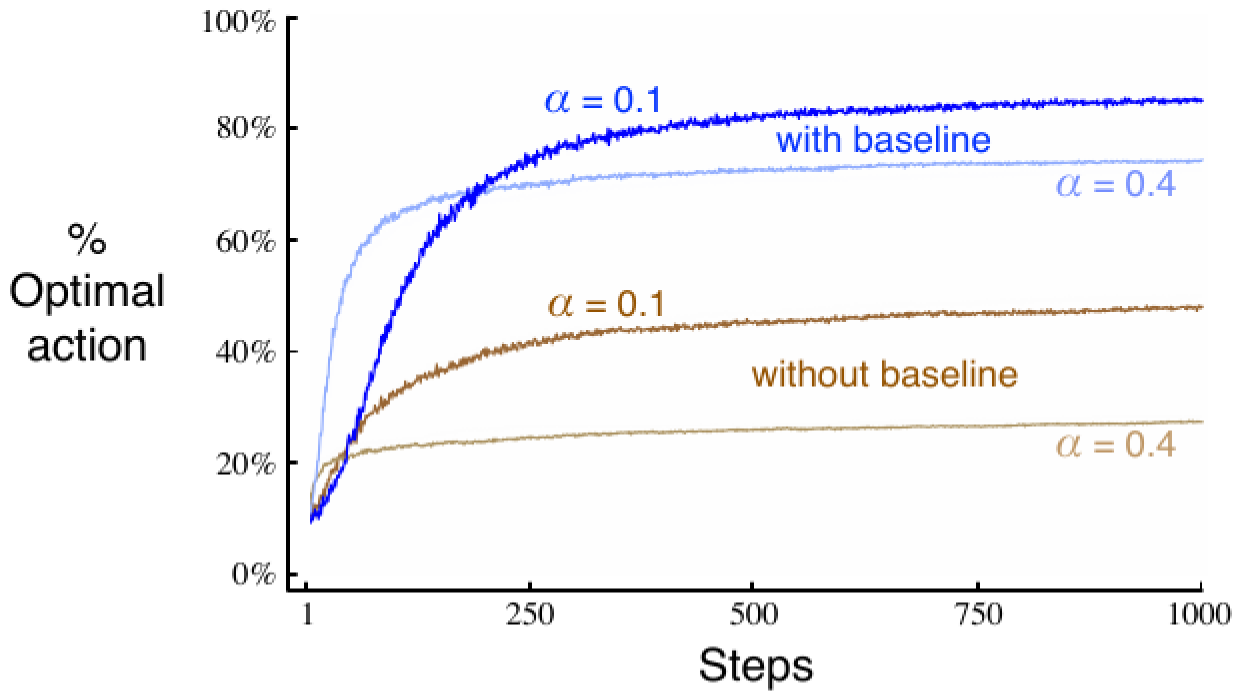
$$Pr\{A_t = a\} \doteq \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} \doteq \pi_t(a)$$

- A natural learning algorithm for Softmax action preferences based on the idea of **stochastic gradient ascent** :
- On each step, after selecting action A_t and receiving the reward R_t, the action preferences are updated :

$$H_{t+1}(A_t) = H_t(A_t) + \alpha(R_t - \bar{R}_t)\big(1 - \pi_t(A_t)\big), \qquad \text{and}$$
$$H_{t+1}(a) = H_t(a) + \alpha(R_t - \bar{R}_t)\pi_t(a), \qquad \text{for all } a \neq A_t$$

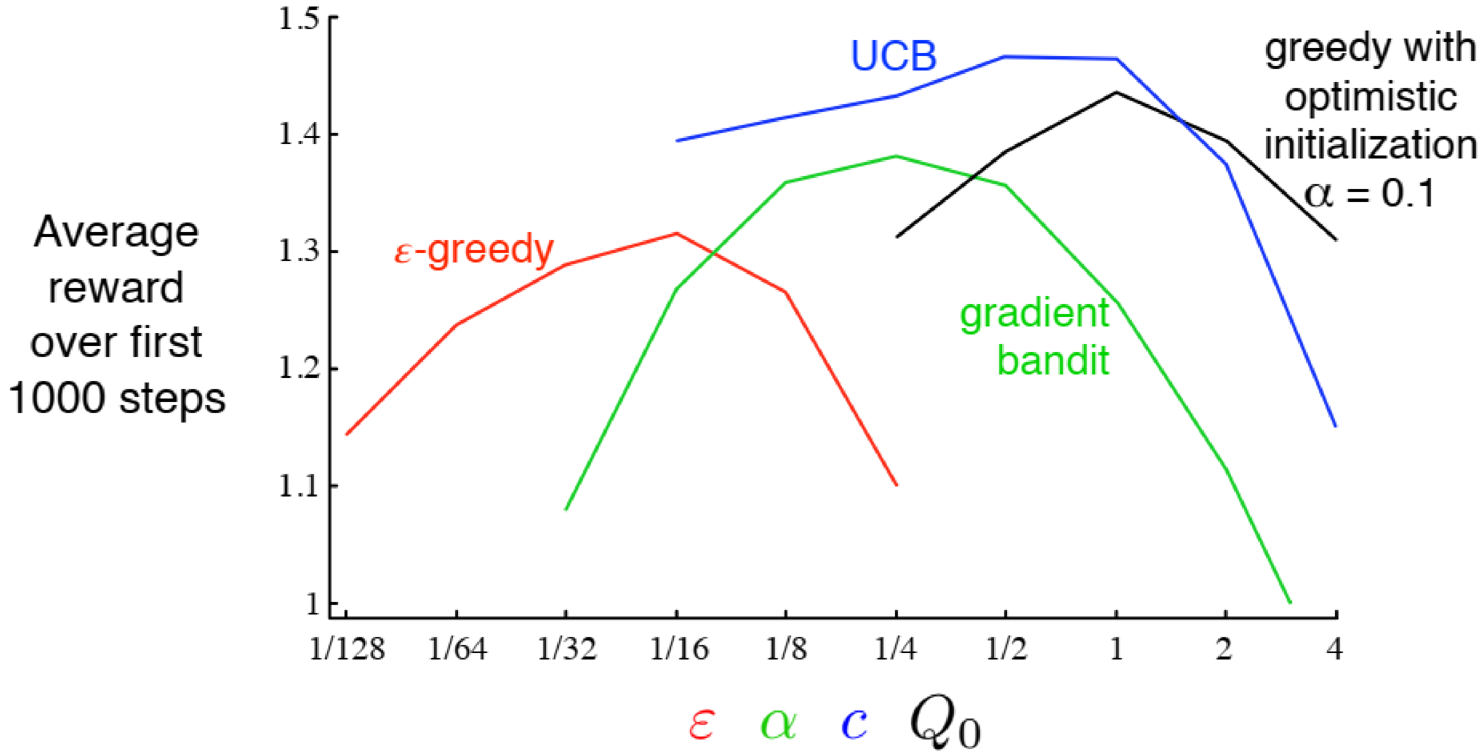
- α : step size parameter,
- $\bar{R}_t$: the average of the rewards up to but not including t (baseline).

Gradient Bandit Algorithms



Average performance of the gradient bandit algorithm with and without a reward baseline ($\bar{R}_t \neq 0$ or $= 0$) on the 10-armed testbed when the $q_*(a)$ are chosen to be near $+4$ rather than near zero.

Summary



Each point is the average reward obtained over 1000 steps with a particular algorithm at a particular setting of its parameter.

Thank you!

Q/A