



Computer Engineering Department

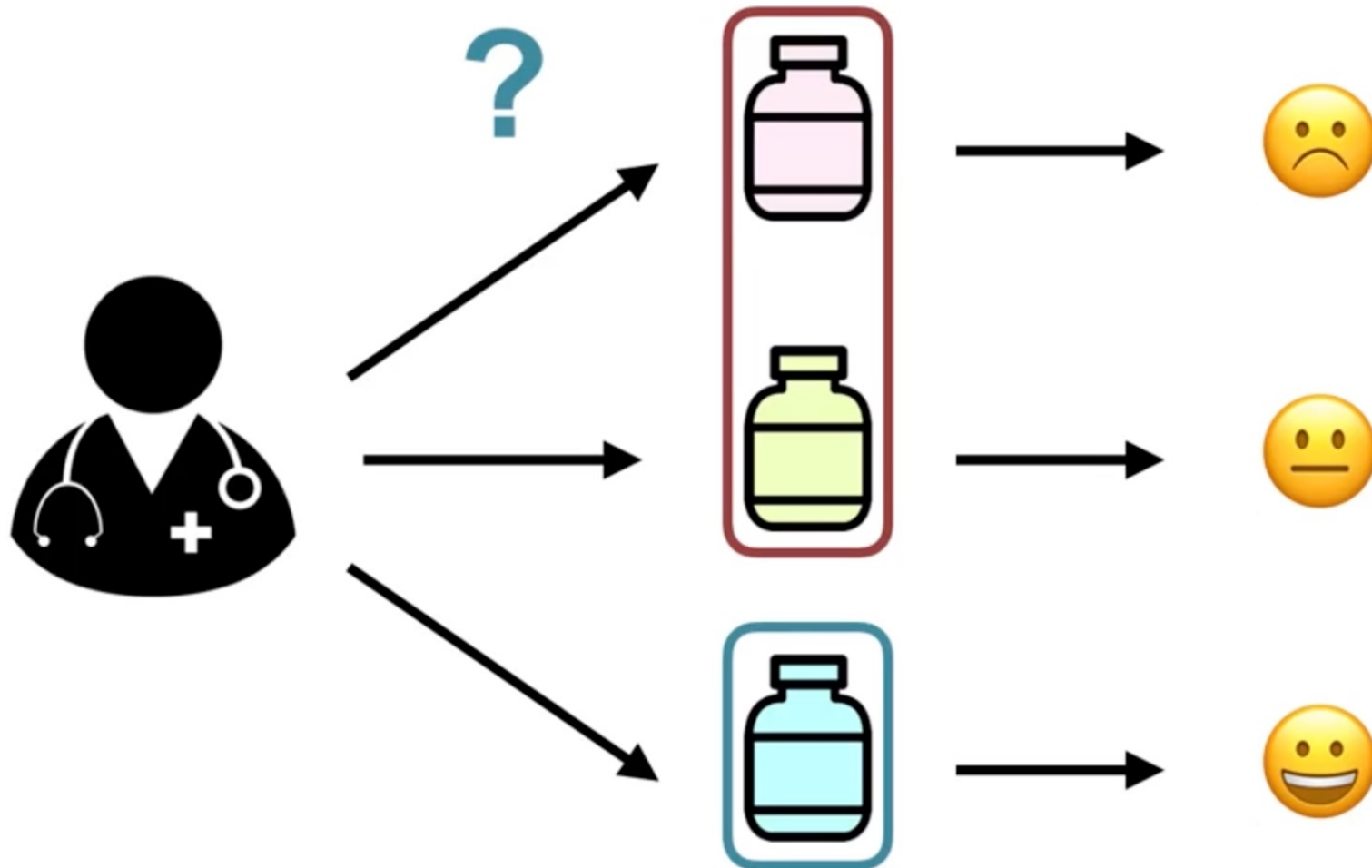
Multi-armed Bandits

Mohammad Hossein Rohban, Ph.D.

Spring 2025

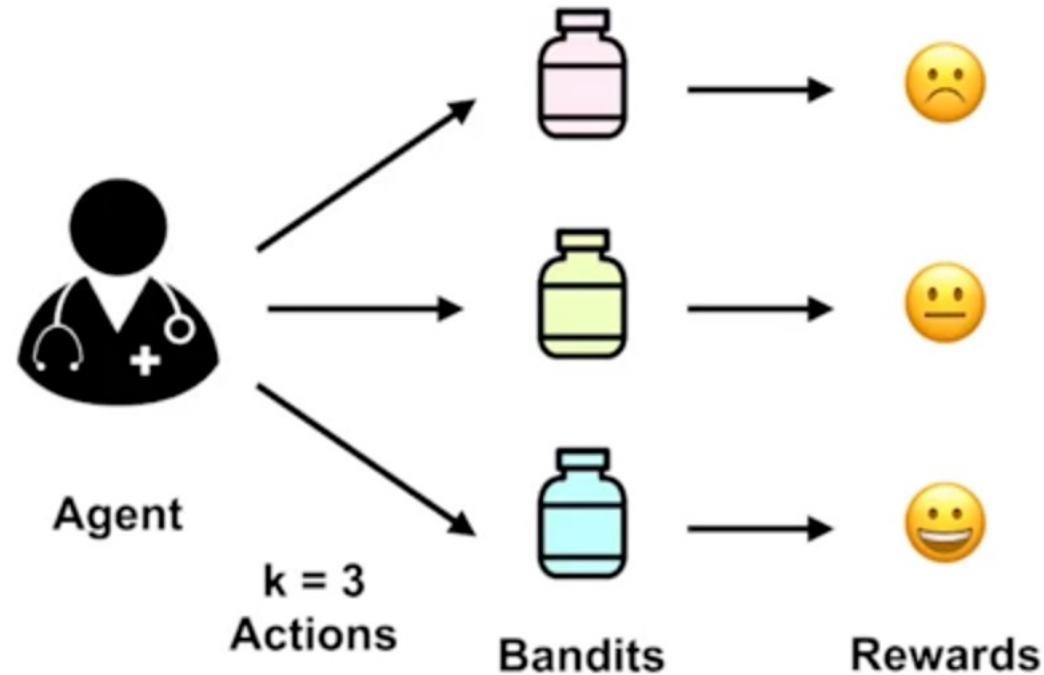
Courtesy: Most of slides are adopted from RL course in Alberta and by Qing Wang.

Clinical Trials



K-armed bandit

In the **k-armed bandit problem**, we have an **agent** who chooses between “k” **actions** and receives a **reward** based on the action it chooses.



Action Values

- The **value** is the **expected reward**

$$\begin{aligned} q_*(a) &\doteq \mathbb{E}[R_t | A_t = a] \quad \forall a \in \{1, \dots, k\} \\ &= \sum_r p(r | a) r \end{aligned}$$

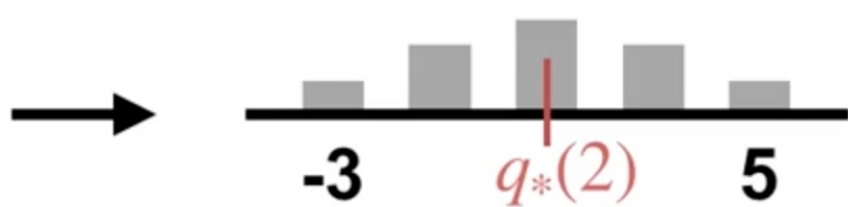
- The goal is to **maximize** the **expected reward**

$$\operatorname{argmax}_a q_*(a)$$

Calculating $q_*(a)$



$$\longrightarrow q_*(a) = .5 \times -11 + .5 \times 9$$



$$\longrightarrow q_*(a) = 1$$



$$\longrightarrow q_*(a) = 3$$

Why we discuss bandits?

- **Exploration** vs. Exploitation Dilemma
- Many **real-world** applications
 - e.g. Recommender Systems

Value of an Action

- The **value** of an action is the **expected reward** when that action is taken

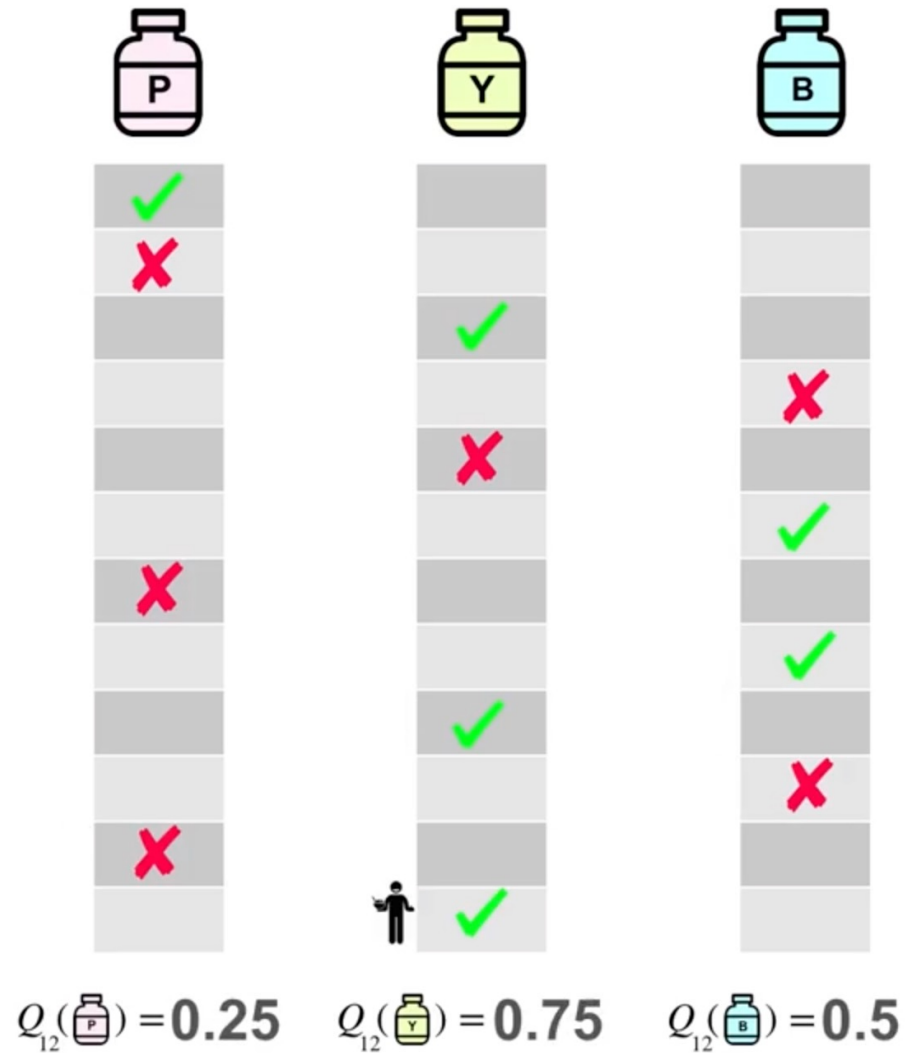
$$q_*(a) \doteq \mathbb{E}[R_t | A_t = a]$$

- $q_*(a)$ is not known, so we **estimate** it

Sample Average

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

Sample Average (cont.)



Incremental Update Rule

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


$$\begin{aligned}&= \frac{1}{n} (R_n + (n-1)Q_n) \\&= \frac{1}{n} (R_n + nQ_n - Q_n) \\&= \frac{1}{n} (R_n + nQ_n - Q_n)\end{aligned}$$

Incremental Update Rule

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

Non-Stationary Bandit Problem

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

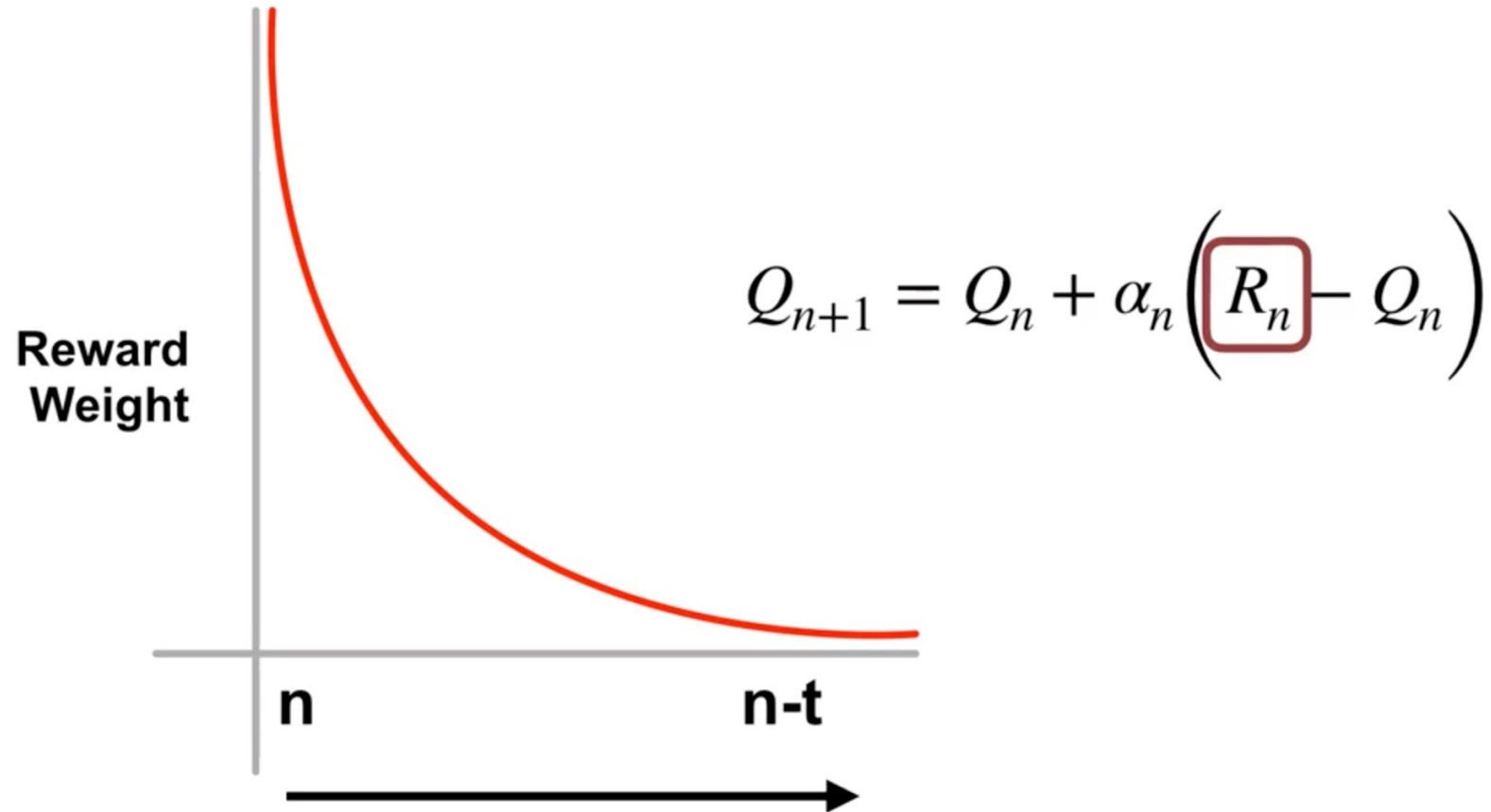


0.25

0.75

0.9

Non-Stationary Bandit Problem



Decaying Past Reward

$$\begin{aligned}Q_{n+1} &= Q_n + \alpha_n(R_n - Q_n) \\&= \alpha R_n + Q_n - \alpha Q_n \\&= \alpha R_n + (1 - \alpha)Q_n \\&= \alpha R_n + (1 - \alpha)\alpha R_{n-1} + (1 - \alpha)Q_{n-1} \\&= \alpha R_n + (1 - \alpha)\alpha R_{n-1} + (1 - \alpha)^2 Q_{n-1}\end{aligned}$$

$$\begin{aligned}Q_{n+1} &= Q_n + \alpha_n(R_n - Q_n) \\&= \alpha R_n + (1 - \alpha)\alpha R_{n-1} + (1 - \alpha)^2 \alpha R_{n-2} + \dots \\&\quad + (1 - \alpha)^{n-1} \alpha R_1 + (1 - \alpha)^n Q_1 \\&= (1 - \alpha)^n Q_1 + \sum_{i=1}^n \alpha (1 - \alpha)^{n-i} R_i\end{aligned}$$

$Q_1 \rightarrow$ initial action-value

Exploration and Exploitation

- **Exploration** - improve knowledge for long-term benefit



$$\begin{aligned}q(a) &= 0 \\ N(a) &= 0 \\ q_*(a) &= 3\end{aligned}$$



$$\begin{aligned}q(a) &= 0 \\ N(a) &= 0 \\ q_*(a) &= 4\end{aligned}$$



$$\begin{aligned}q(a) &= 0 \\ N(a) &= 0 \\ q_*(a) &= 2\end{aligned}$$

Exploration and Exploitation



$$q(a) = 2$$

$$N(a) = 2$$

$$q_*(a) = 3$$



$$q(a) = 4$$

$$N(a) = 2$$

$$q_*(a) = 4$$



$$q(a) = 1$$

$$N(a) = 2$$

$$q_*(a) = 2$$

Exploration and Exploitation

- **Exploitation** - exploit knowledge for short-term benefit



$$\begin{aligned}q(a) &= 3 \\ N(a) &= 5 \\ q_*(a) &= 3\end{aligned}$$



$$\begin{aligned}q(a) &= 0 \\ N(a) &= 0 \\ q_*(a) &= 4\end{aligned}$$



$$\begin{aligned}q(a) &= 0 \\ N(a) &= 0 \\ q_*(a) &= 2\end{aligned}$$

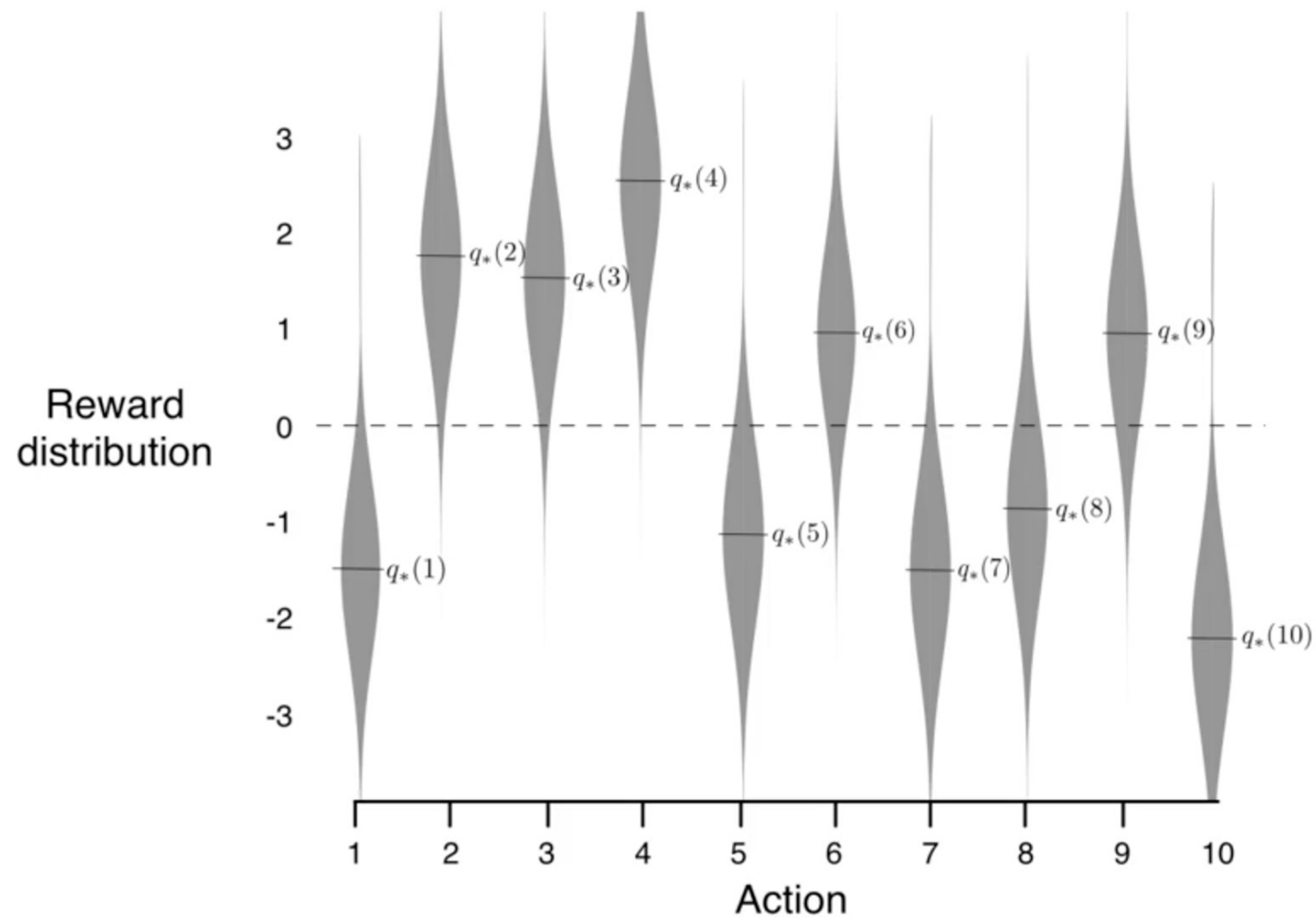
Exploration and Exploitation

- **Exploration** - **improve** knowledge for **long-term** benefit
- **Exploitation** - **exploit** knowledge for **short-term** benefit
- How do we choose when to **explore** and when to **exploit**?

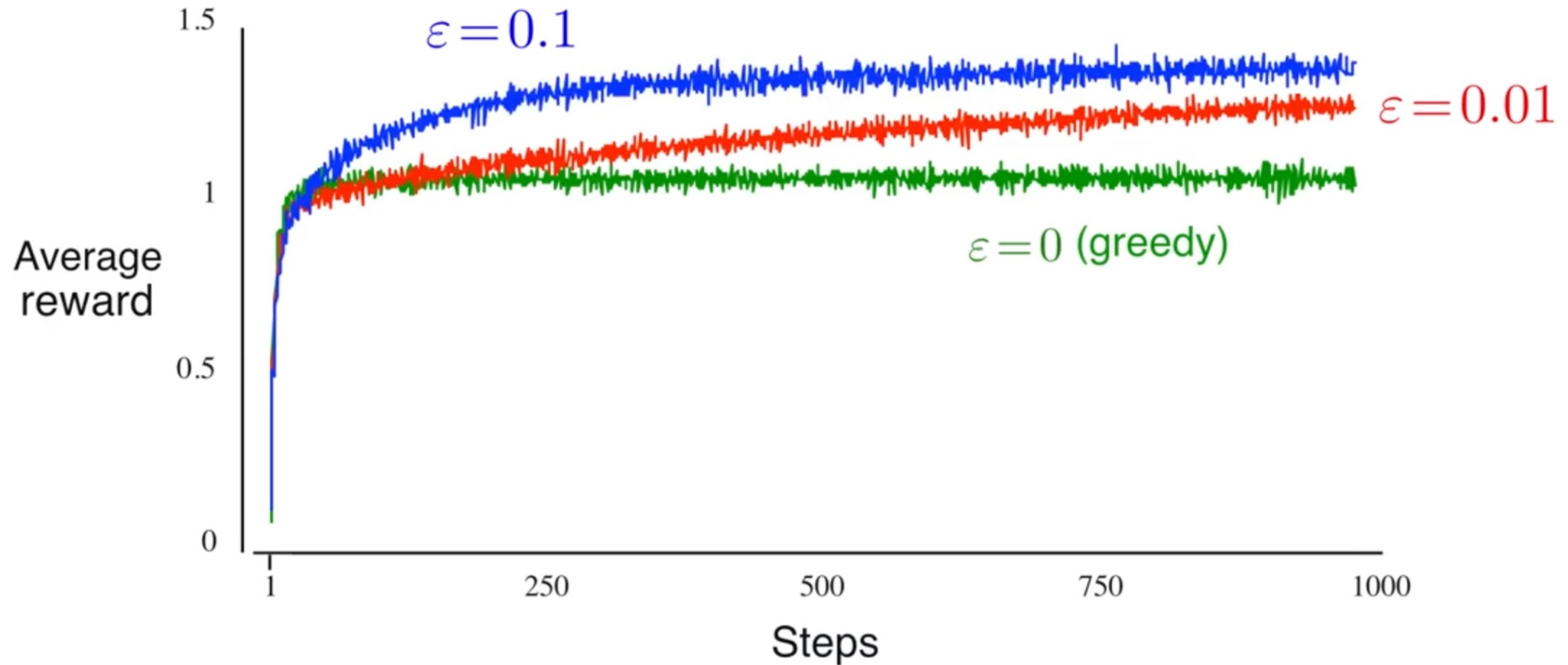
epsilon-Greedy Action Selection

$$A_t \leftarrow \begin{cases} \operatorname{argmax}_a \underline{Q_t(a)} & \text{with probability } 1 - \epsilon \\ a \sim \text{Uniform}(\{a_1 \dots a_k\}) & \text{with probability } \epsilon \end{cases}$$

The 10-armed testbed



Average reward for epsilon-greedy

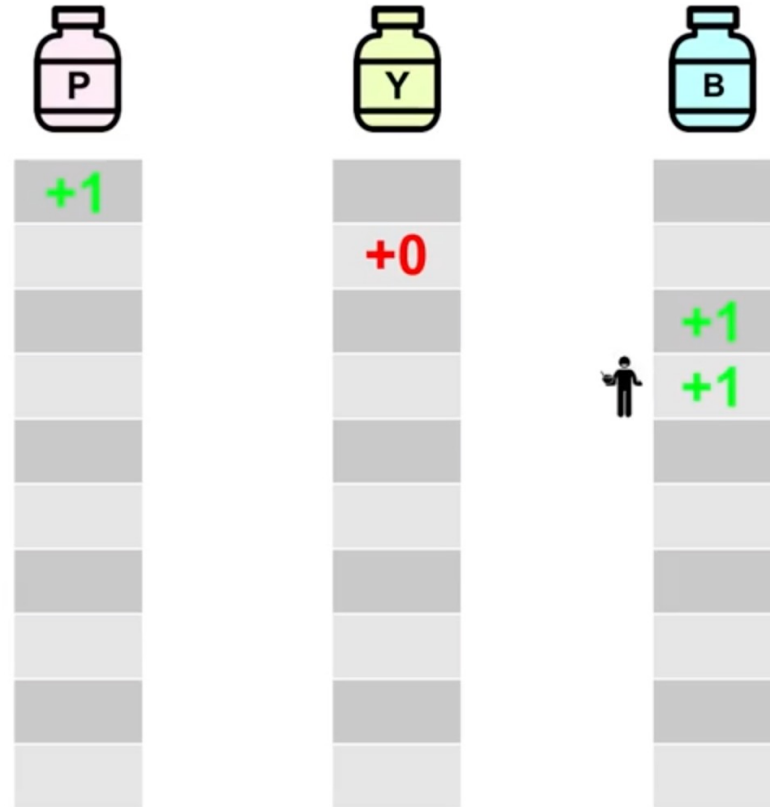


Optimistic Initial Values

A reward of 1 if the treatment succeeds otherwise 0

$$Q_{n+1} \leftarrow Q_n + \alpha (R_n - Q_n)$$

Let $\alpha = 0.5$



$$Q_5(\text{P}) = 1.5$$

$$Q_5(\text{Y}) = 1.0$$

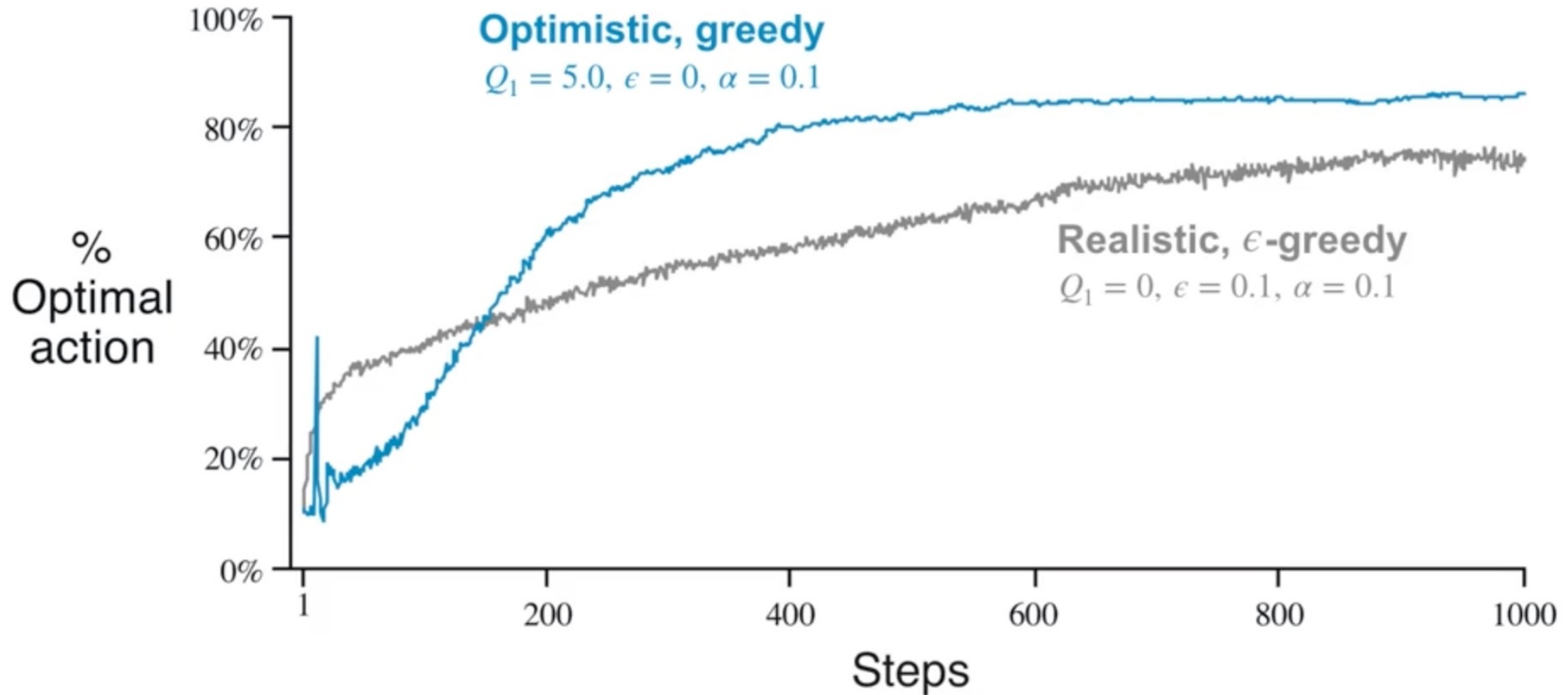
$$Q_5(\text{B}) = 1.75$$

$$q_*(\text{P}) = 0.25$$

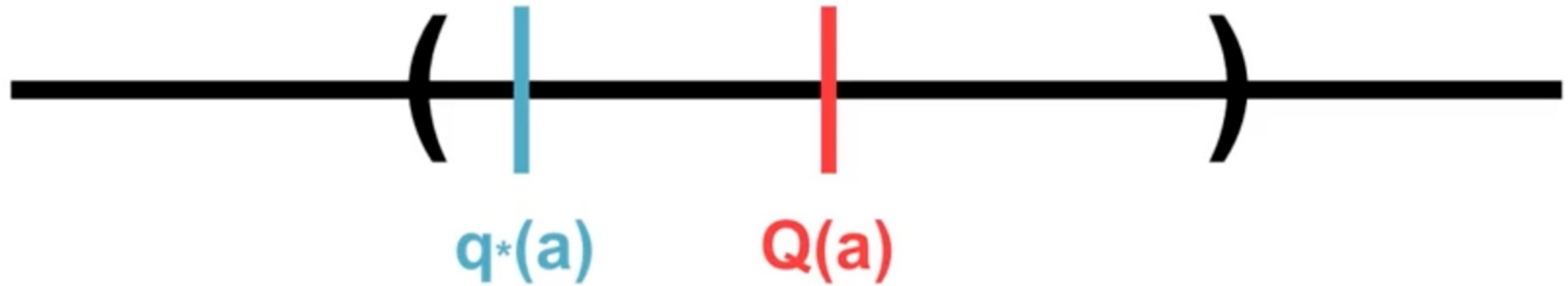
$$q_*(\text{Y}) = 0.75$$

$$q_*(\text{B}) = 0.5$$

Optimistic Value Initialization in action

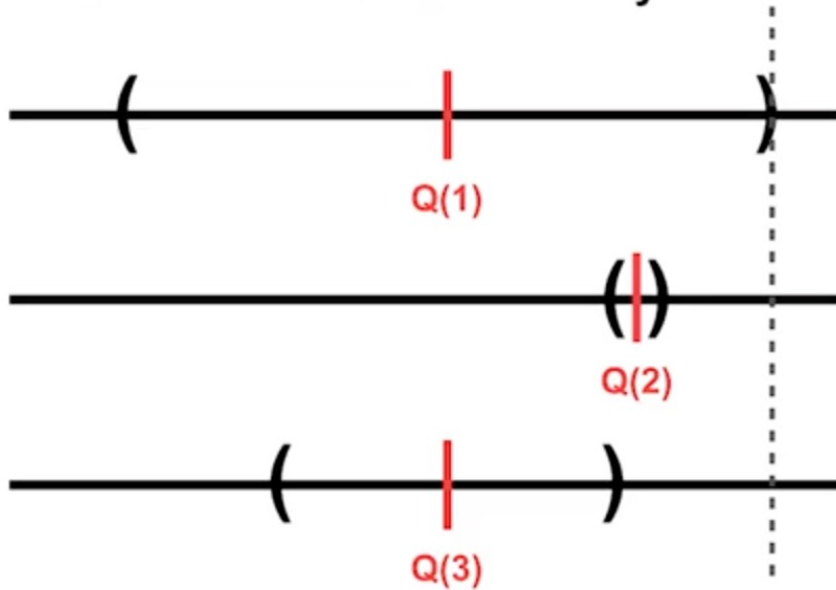


Uncertainty in Estimates

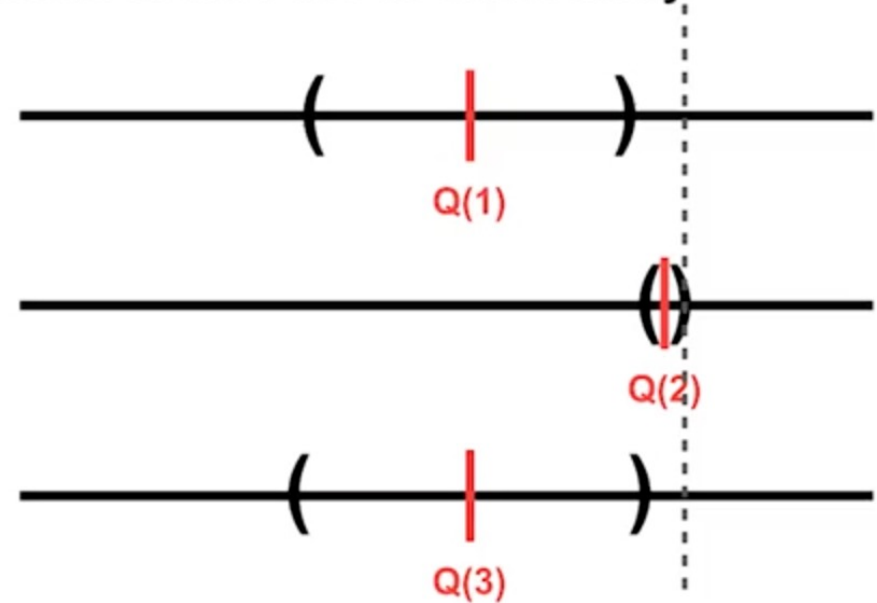


Optimism

Optimism in the Face of Uncertainty



Optimism in the Face of Uncertainty



Upper Confidence Bound Action Selection

empirical mean $\leftarrow Q_t(a) = \frac{1}{N_t(a)} \sum R^a$ $\leadsto \text{Var}[Q_t(a)]$

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

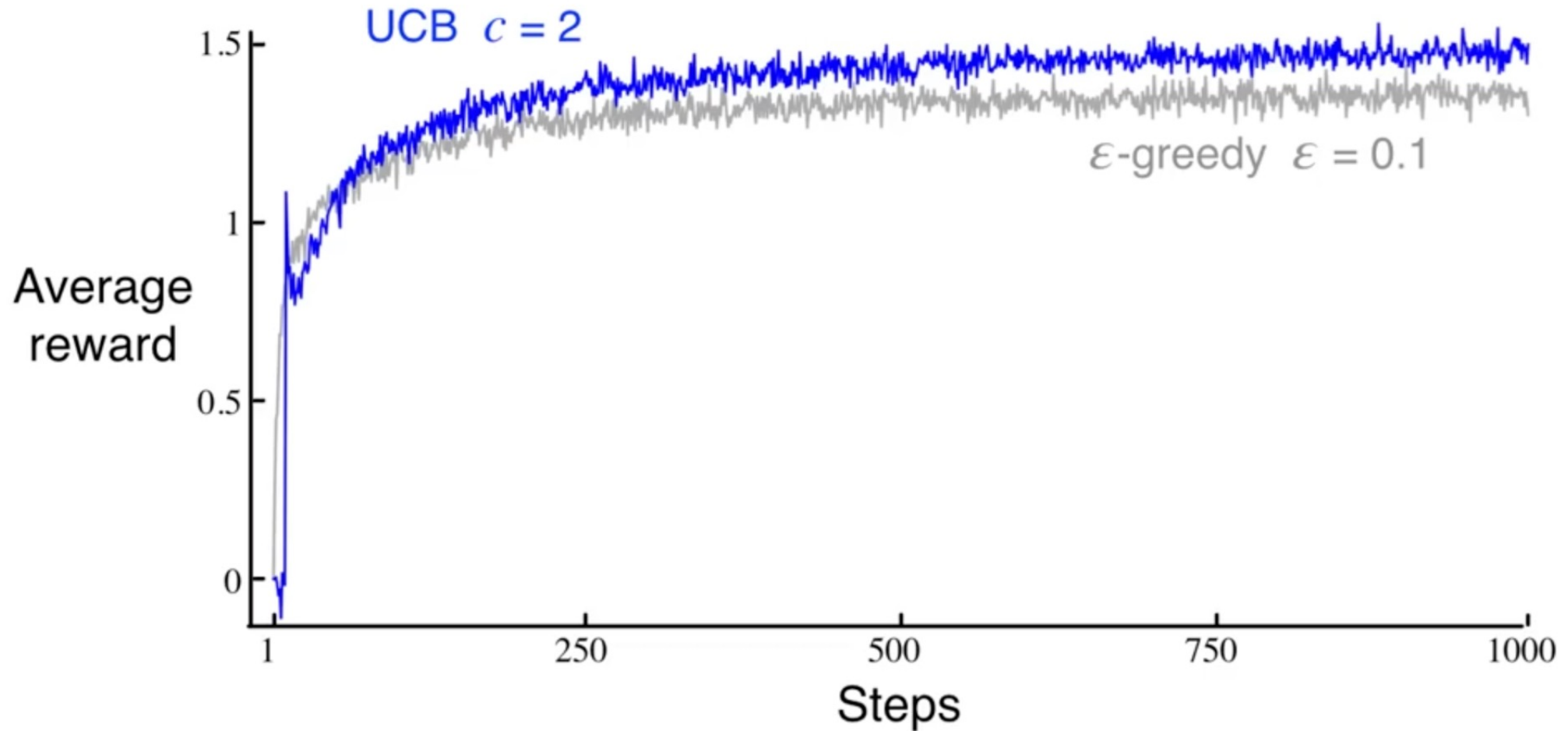
$= \frac{1}{N_t(a)} \sum \sigma_{R^a}^2$
 ~~$\frac{1}{N_t(a)} \sigma_{R^a}^2$~~

$\text{Std}[Q_t(a)]$
 $\propto \sqrt{\frac{1}{N_t(a)}}$

Upper Confidence Bound Action Selection

$$\begin{array}{lcl} c\sqrt{\frac{\ln t}{N_t(a)}} & \longrightarrow & c\sqrt{\frac{\ln \text{ timesteps}}{\text{times action } a \text{ taken}}} \\ & & \swarrow \searrow \\ & & c\sqrt{\frac{\ln 10000}{5000}} \longrightarrow 0.043c \\ & & c\sqrt{\frac{\ln 10000}{100}} \longrightarrow 0.303c \end{array}$$

Eps-greedy vs UCB



Contextual Bandits

- Contextual bandit algorithm in round t
 - Algorithm observes user $\mathbf{u}_t$ and a set $\mathbf{A}$ of arms together with their features $\mathbf{x}_{t,a}$ (context)
 - Based on payoffs from previous trials, algorithm chooses arm $\mathbf{a} \in \mathbf{A}$ and receives payoff $\mathbf{r}_{t,a}$
 - Algorithm improves arm selection strategy with each observation $(\mathbf{x}_{t,a}, \mathbf{a}, \mathbf{r}_{t,a})$

LinUCB Algorithm

- Expectation of reward of each arm is modeled as a linear function of the context.

θ_a^* is the unknown coefficient vector we **aim to learn**

Payoff of arm a : $E[r_{t,a} | x_{t,a}] = \underbrace{[x_{t,a}]^T}_{\text{feature vector}} \theta_a^*$

$x_{t,a}$ is a **d**-dimensional feature vector

- The goal is to minimize regret, defined as the difference between the expectation of the reward of best arms and the expectation of the reward of selected arms.

$$R_t(T) \stackrel{\text{def}}{=} E \left[\sum_{t=1}^T r_{t,a_t^*} \right] - E \left[\sum_{t=1}^T r_{t,a_t} \right]$$

LinUCB Algorithm *Context*

- $E[r_{t,a} | x_{t,a}] = [x_{t,a}]^T \theta_a^*$
 - How to estimate θ_a ?
 - Linear regression solution to θ_a is

$$\widehat{\theta}_a = \underset{\theta}{\operatorname{argmin}} \sum_{m \in D_a} ([x_{t,a}]^T \theta_a - b_a^{(m)})^2$$

We can get:

$$\widehat{\theta}_a = (D_a^T D_a + I_d)^{-1} D_a^T b_a$$

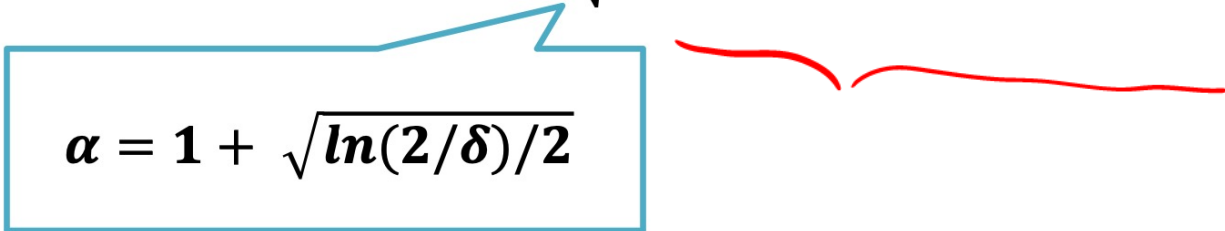
D_a is a $m \times d$ matrix of m training inputs $[x_{t,a}]$

b_a is a m -dimension vector of responses to a (click/no-click)

LinUCB Algorithm

- Using similar techniques as we used for UCB

$$|[x_{t,a}]^T \widehat{\boldsymbol{\theta}}_a - \mathbb{E}[r_{t,a} | x_{t,a}]| \leq \alpha \sqrt{[x_{t,a}]^T (\mathbf{D}_a^T \mathbf{D}_a + \mathbf{I}_d)^{-1} x_{t,a}}$$


$$\alpha = 1 + \sqrt{\ln(2/\delta)/2}$$

- For a given context, we estimate the reward and the confidence interval.

$$\mathbf{a}_t \stackrel{\text{def}}{=} \underset{\substack{\text{Estimated } \mu_a}}{\operatorname{argmax}_{a \in A_t}} \left(\underbrace{[x_{t,a}]^T \widehat{\boldsymbol{\theta}}_a}_{\text{Estimated } \mu_a} + \underbrace{\alpha \sqrt{[x_{t,a}]^T (\mathbf{D}_a^T \mathbf{D}_a + \mathbf{I}_d)^{-1} x_{t,a}}}_{\text{Confidence interval}} \right)$$

LinUCB Algorithm

- Initialization:

$$A_a \stackrel{\text{def}}{=} \mathbf{D}_a^T \mathbf{D}_a + \mathbf{I}_d$$

- For each arm a :

- $A_a = \mathbf{I}_d$

//identity matrix $d \times d$

- $b_a = [0]_d$

//vector of zeros

- Online algorithm:

- For $t=[1:T]$:

- Observe features for all arms $a : x_{t,a} \in R^d$

- For each arm a :

- $\theta_a = A_a^{-1} b_a$

//regression coefficients

- $p_{t,a} = [x_{t,a}]^T \theta_a + \alpha \sqrt{[x_{t,a}]^T A_a^{-1} x_{t,a}}$

- Choose arm $a_t = \operatorname{argmax}_a p_{t,a}$

//choose arm

- $A_{a_t} = A_{a_t} + x_{t,a_t} [x_{t,a_t}]^T$

//update A for the chosen arm a_t

- $b_{a_t} = b_{a_t} + r_t x_{t,a_t}$

//update b for the chosen arm a_t

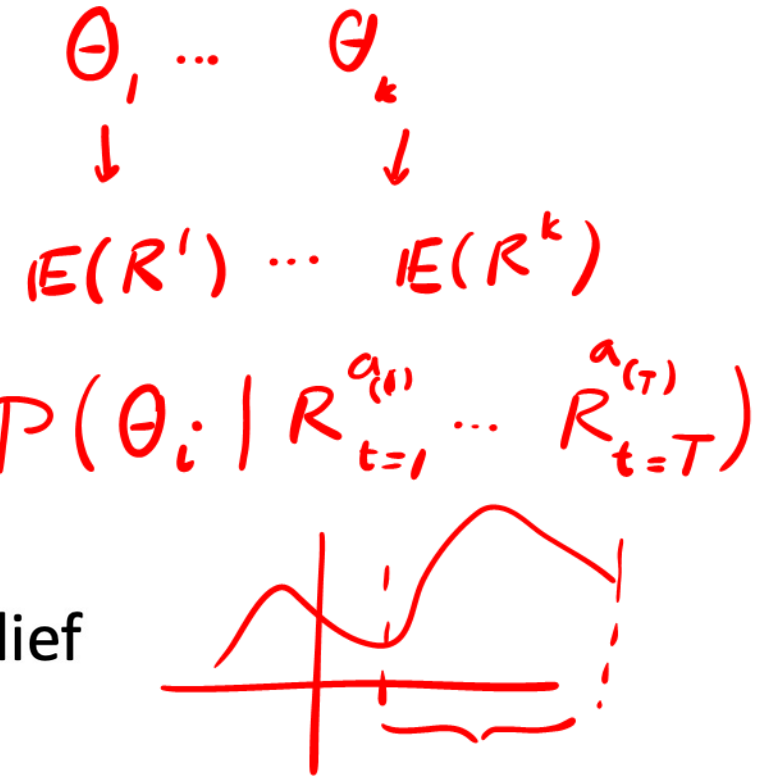
Thompson Sampling

- A simple natural Bayesian heuristic
 - Maintain a belief(distribution) for the unknown parameters

- Each time, pull arm a and observe a reward r

- Initialize priors using belief distribution
 - For $t=1:T$:

- Sample random variable X from each arm's belief distribution
 - Select the arm with largest X
 - Observe the result of selected arm
 - Update prior belief distribution for selected arm



A Simple Example

- Coin toss: $x \sim \text{Bernoulli}(\theta)$

- Let's assume that

- $\theta \sim \text{Beta}(\alpha_H, \alpha_T)$

Beta distribution

- $P(\theta) \propto \theta^{\alpha_H-1} (1-\theta)^{\alpha_T-1}$

$$Q_t(a) = \mathbb{E}(R^a)$$

- $P(\theta|X) = \frac{P(X|\theta)P(\theta)}{\sum_{\theta} P(X|\theta)}$

Posterior

Prior

The prior is conjugate!

$$\frac{1}{K} \underbrace{\theta^x (1-\theta)^{1-x}}_{\text{likelihood } P(x|\theta)} \cdot \theta^{\alpha_H-1} (1-\theta)^{\alpha_T-1} = \text{Beta}(\alpha_H + x, \alpha_T + (1-x))$$



Algorithm

- Theorem [Emilie et al. 2012]
 - Initially assumes arm i with prior $\text{Beta}(1,1)$ on μ_i
 - $S_i = \text{\#“Success”}$, $F_i = \text{\#“Failure”}$

Algorithm 1: Thompson Sampling for Bernoulli bandits

$S_i = 0, F_i = 0.$

→ **foreach** $t = 1, 2, \dots$, **do**

 For each arm $i = 1, \dots, N$, sample $\theta_i(t)$ from the $\text{Beta}(S_i + 1, F_i + 1)$ distribution.

 Play arm $i(t) := \arg \max_i \theta_i(t)$ and observe reward r_t .

 If $r = 1$, then $S_i = S_i + 1$, else $F_i = F_i + 1$.

end

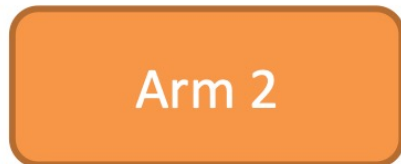
Algorithm

- Initialization

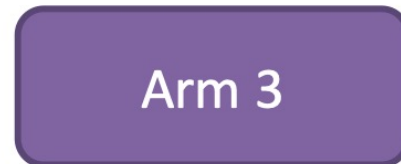
$\text{Beta}(1,1)$



$\text{Beta}(1,1)$

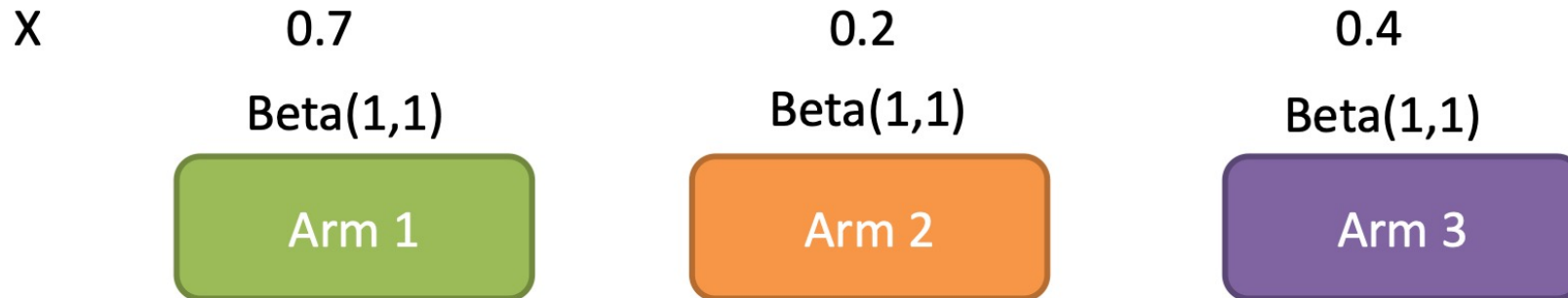


$\text{Beta}(1,1)$



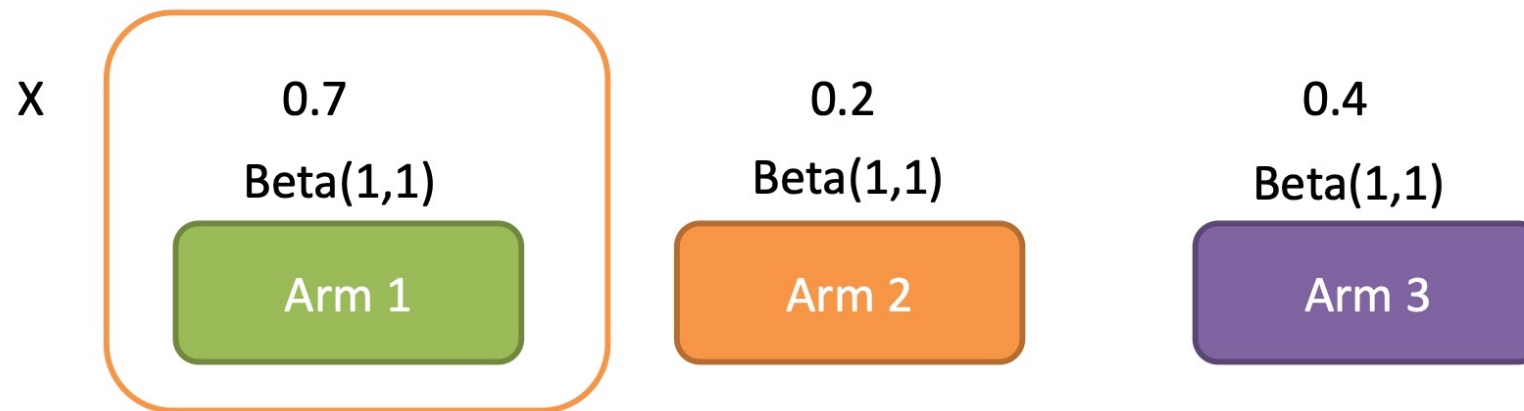
Algorithm

- For each round:
 - Sample random variable X from each arm's Beta Distribution



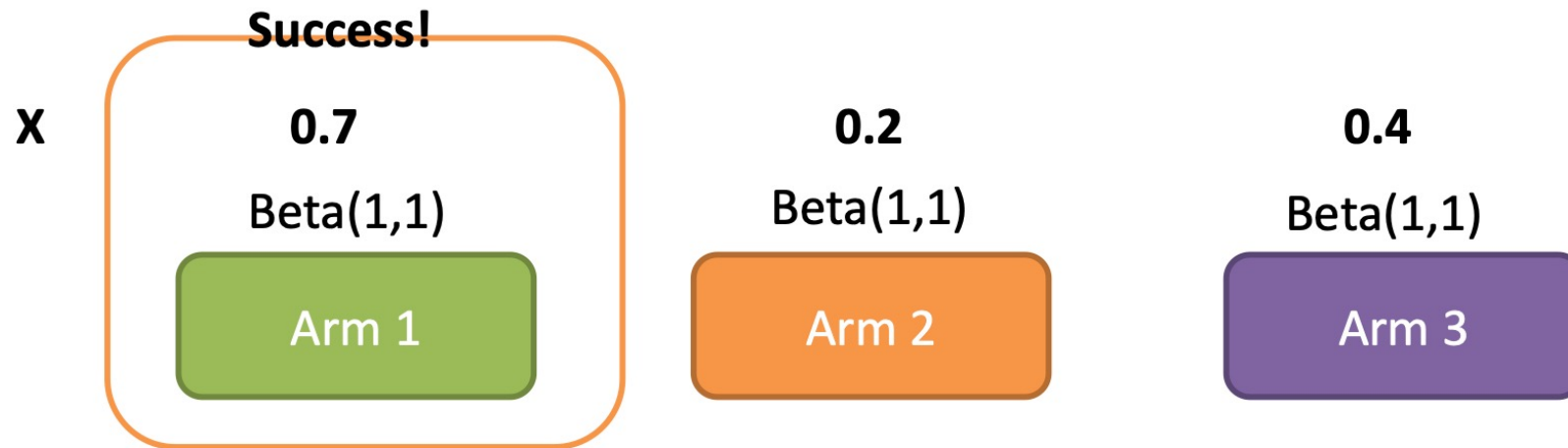
Algorithm

- For each round:
 - Sample random variable X from each arm's Beta Distribution
 - Select the arm with largest X



Algorithm

- For each round:
 - Sample random variable X from each arm's Beta Distribution
 - Select the arm with largest X
 - Observe the result of selected arm



Algorithm

- For each round:
 - Sample random variable X from each arm's Beta Distribution
 - Select the arm with largest X
 - Observe the result of selected arm
 - Update prior Beta distribution for selected arm

$$\int \frac{1}{K} g(x) dx = 1$$

$$K = \int g(x) dx$$

