Computer Engineering Department

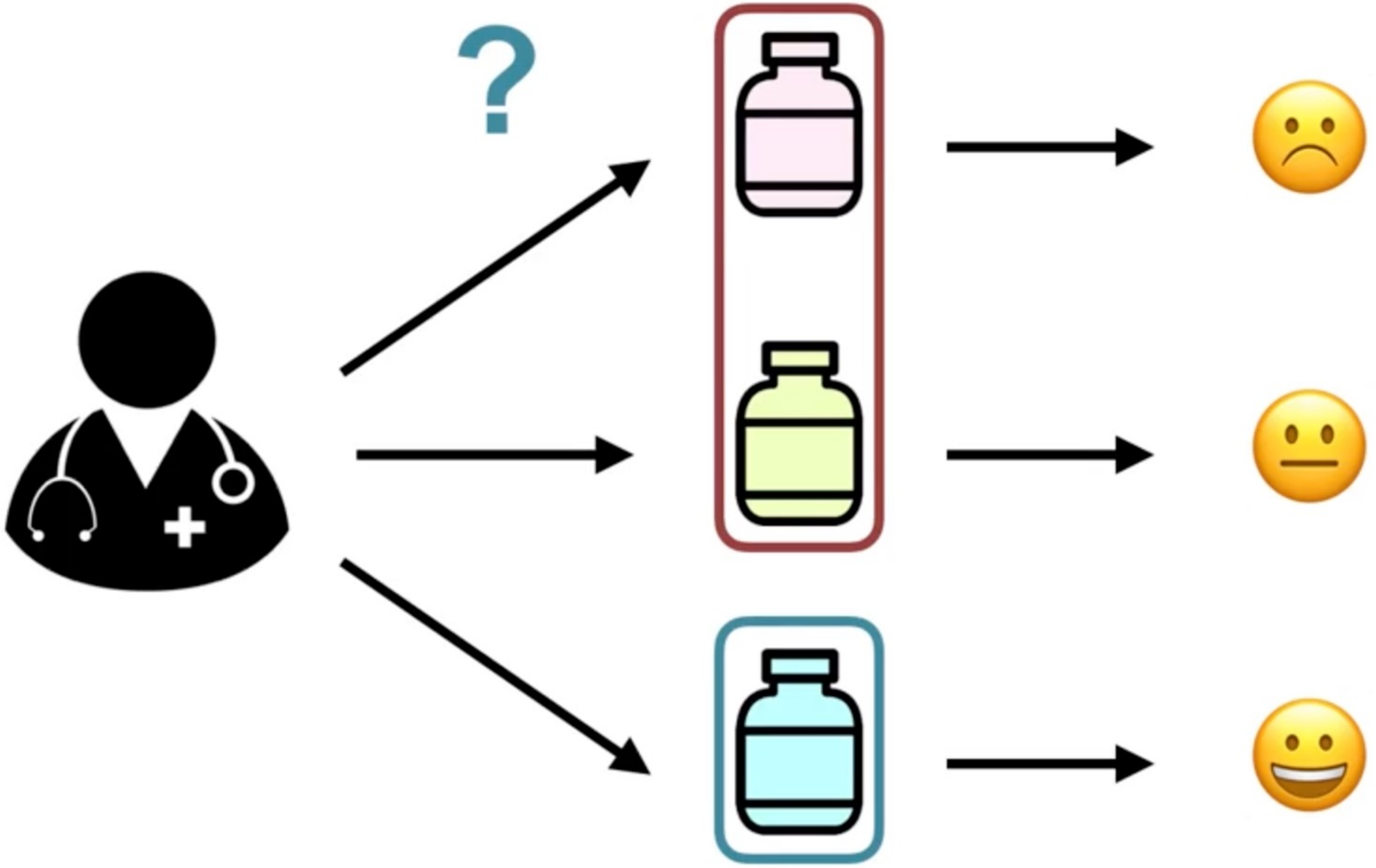
Multi-armed Bandits

Mohammad Hossein Rohban, Ph.D.

Spring 2025

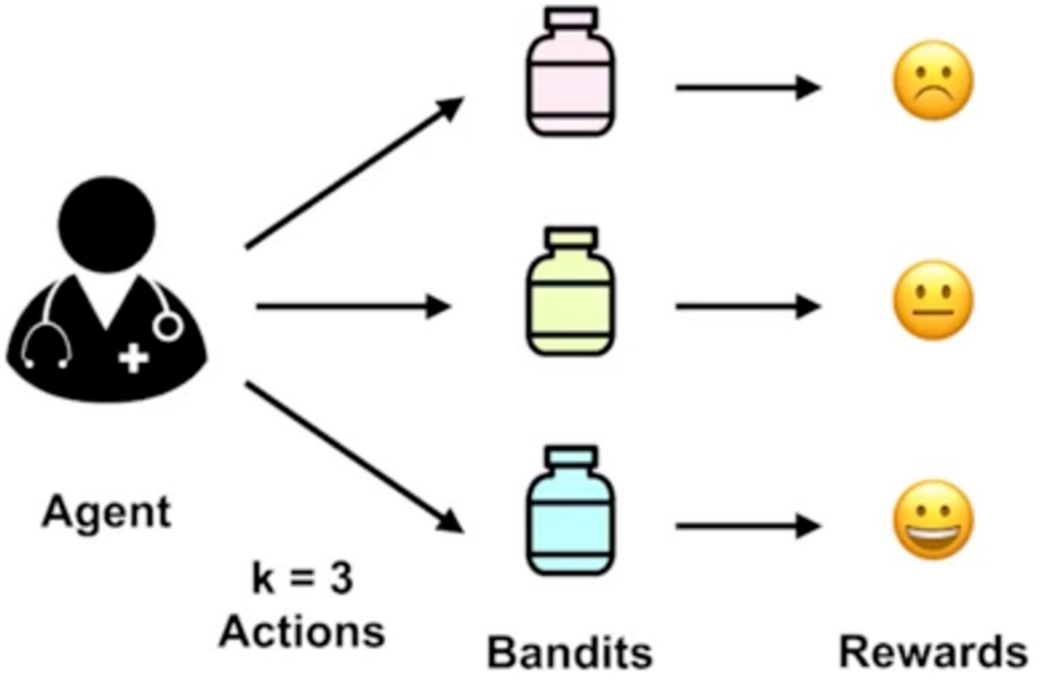
Courtesy: Most of slides are adopted from RL course in Alberta and by Qing Wang.

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

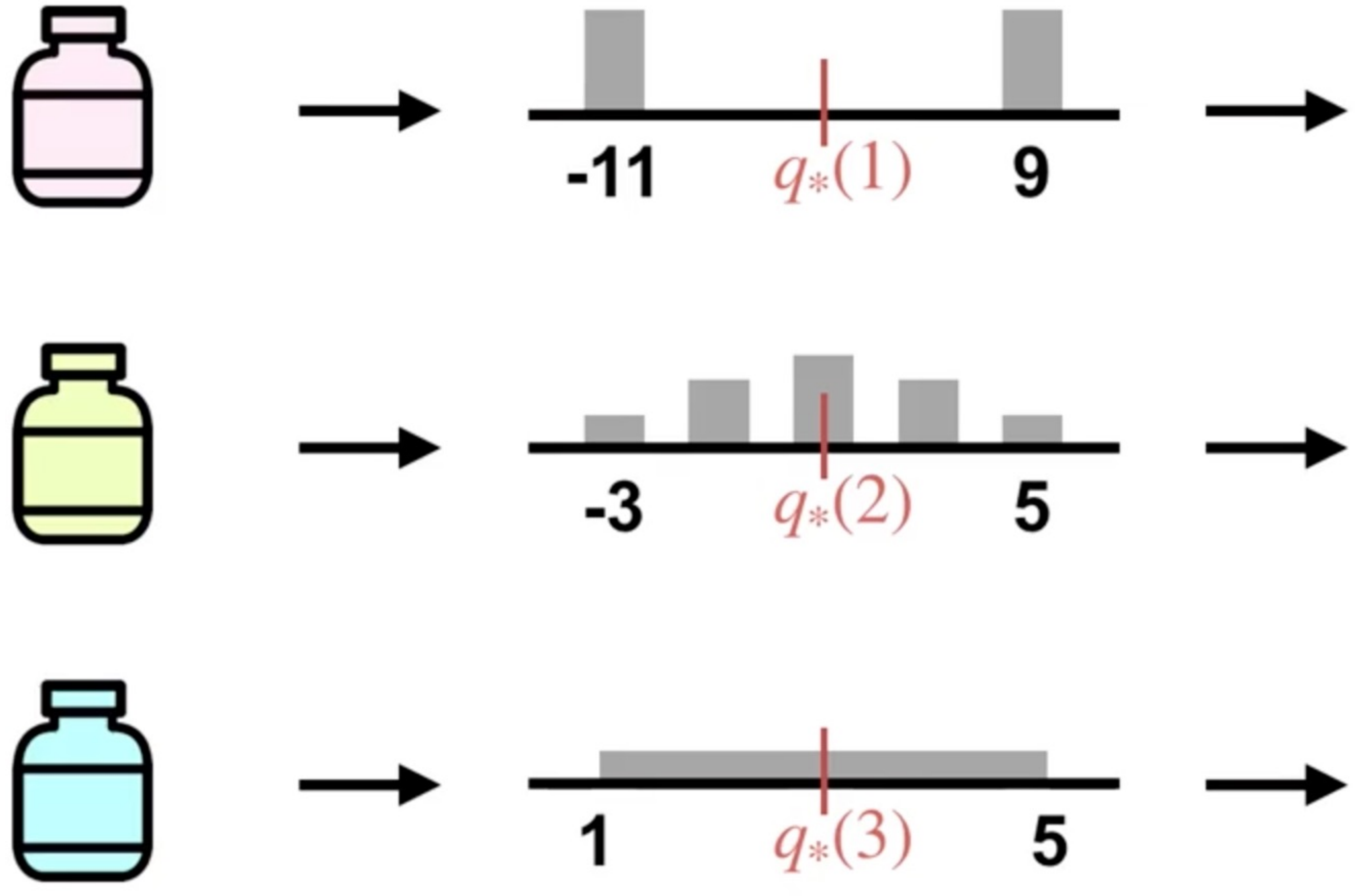
- The **value** is the **expected reward**

$$q_*(a) \doteq \mathbb{E}[R_t \mid A_t = a] \quad \forall a \in \{1, \ldots, k\}$$
$$= \sum_r p(r \mid a)\, r$$

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

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

Calculating $q_*(a)$



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

$$q_*(a) = 1$$

$$q_*(a) = 3$$

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

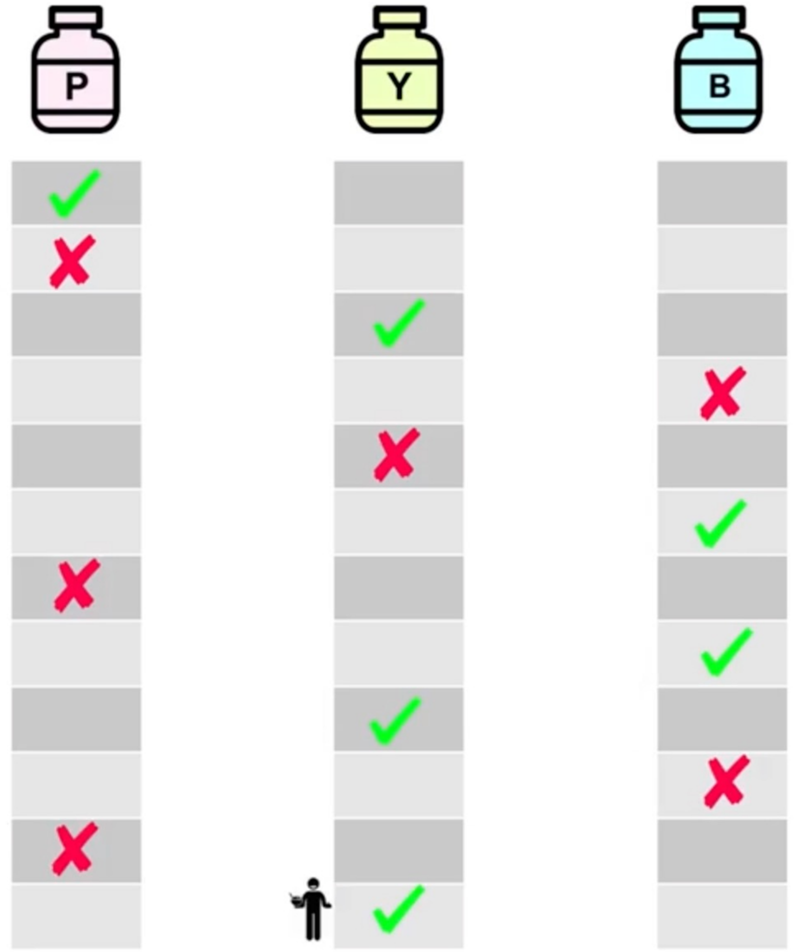
$$q_*(a) \doteq \mathbb{E}[R_t \mid A_t = a]$$

- $q_*(a)$ 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.)



$Q_{12}(\text{P}) = 0.25$ $Q_{12}(\text{Y}) = 0.75$ $Q_{12}(\text{B}) = 0.5$

Incremental Update Rule

$$Q_{n+1} = \frac{1}{n}\sum_{i=1}^{n} R_i$$

$$= \frac{1}{n}\left(R_n + \sum_{i=1}^{n-1} R_i\right)$$

$$= \frac{1}{n}\left(R_n + (n-1)\frac{1}{n-1}\sum_{i=1}^{n-1} R_i\right)$$

$$= \frac{1}{n}\left(R_n + (n-1)Q_n\right)$$

$$= \boxed{\frac{1}{n}}\left(R_n + \boxed{nQ_n} - Q_n\right)$$

$$= \frac{1}{n}\left(R_n + nQ_n - Q_n\right)$$

Incremental Update Rule

$$Q_{n+1} = \frac{1}{n}\sum_{i=1}^{n} R_i$$

$$= \frac{1}{n}\left(R_n + \sum_{i=1}^{n-1} R_i\right)$$

$$= \frac{1}{n}\left(R_n + (n-1)\frac{1}{n-1}\sum_{i=1}^{n-1} R_i\right)$$

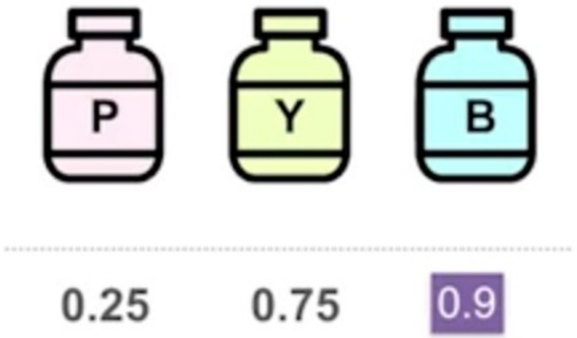
$$= \frac{1}{n}\left(R_n + (n-1)Q_n\right)$$

$$= \boxed{\frac{1}{n}}\left(R_n + \boxed{nQ_n} - Q_n\right)$$

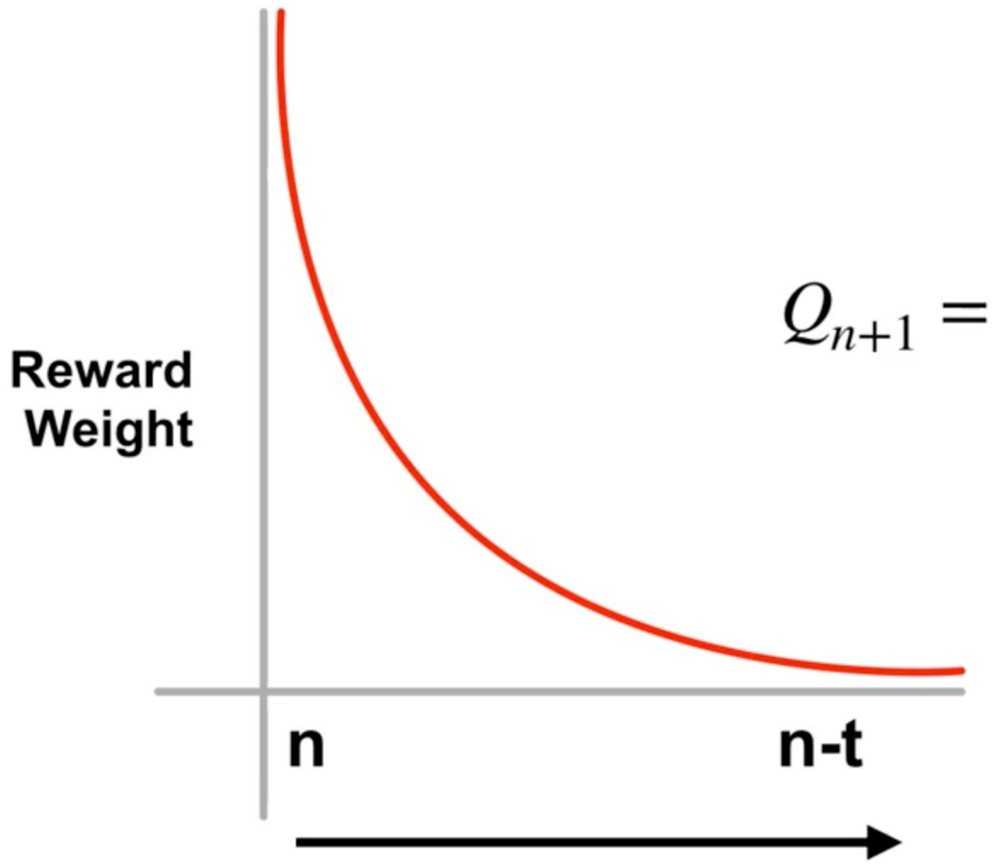
$$= \boxed{Q_n} + \frac{1}{n}\left(R_n - Q_n\right)$$

Non-Stationary Bandit Problem

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



Non-Stationary Bandit Problem



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

Decaying Past Reward

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

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

$Q_1 \rightarrow$ **initial action-value**

Exploration and Exploitation

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

$$q(a) = 0$$
$$N(a) = 0$$
$$q_*(a) = 3$$

$$q(a) = 0$$
$$N(a) = 0$$
$$q_*(a) = 4$$

$$q(a) = 0$$
$$N(a) = 0$$
$$q_*(a) = 2$$

Exploration and Exploitation

$\boxed{q(a) = 2}$

$N(a) = 2$

$\boxed{q_*(a) = 3}$

$\boxed{q(a) = 4}$

$N(a) = 2$

$\boxed{q_*(a) = 4}$

$\boxed{q(a) = 1}$

$N(a) = 2$

$\boxed{q_*(a) = 2}$

Exploration and Exploitation

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

$q(a) = 3$	$q(a) = 0$	$q(a) = 0$
$N(a) = 5$	$N(a) = 0$	$N(a) = 0$
$q_*(a) = 3$	$\boxed{q_*(a) = 4}$	$q_*(a) = 2$

Exploration and Exploitation

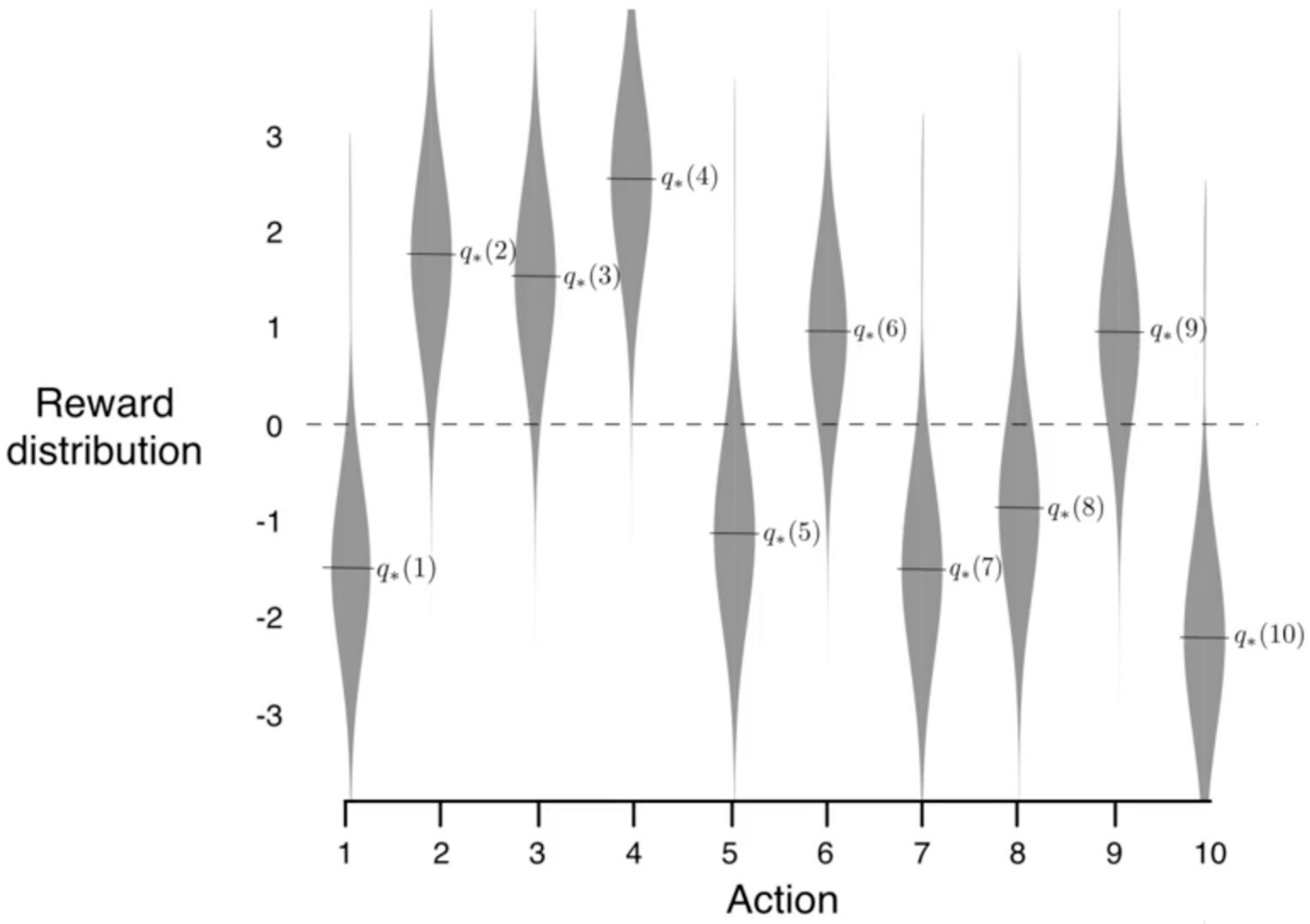
- Exploration - improve knowledge for long-term benefit
- Exploitation - exploit knowledge for short-term benefit

- How do we choose when to explore and when to exploit?

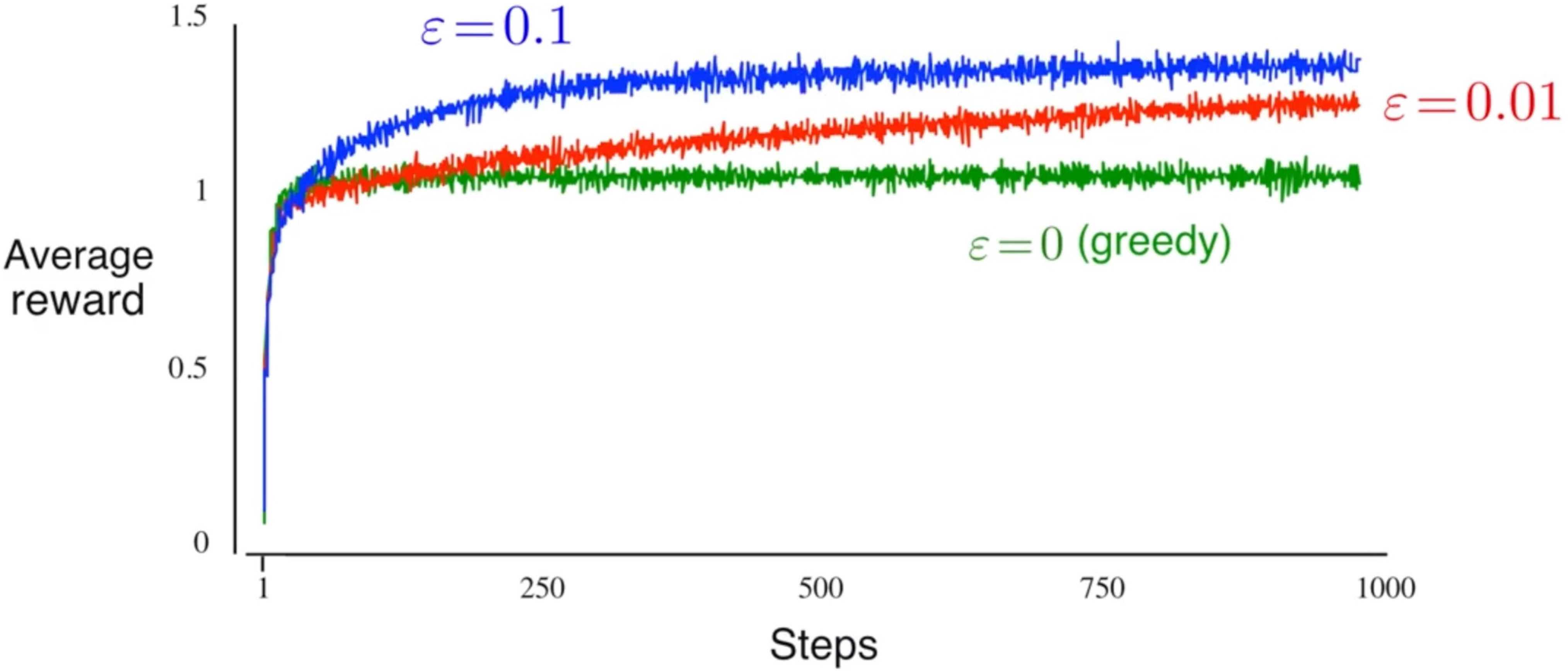
epsilon-Greedy Action Selection

$$A_t \leftarrow \begin{cases} \underset{a}{\text{argmax}}\ Q_t(a) & \textbf{with probability } 1-\epsilon \\ \boxed{a \sim Uniform(\{a_1 \ldots a_k\})} & \textbf{with probability } \epsilon \end{cases}$$

The 10-armed testbed



Average reward for epsilon-greedy

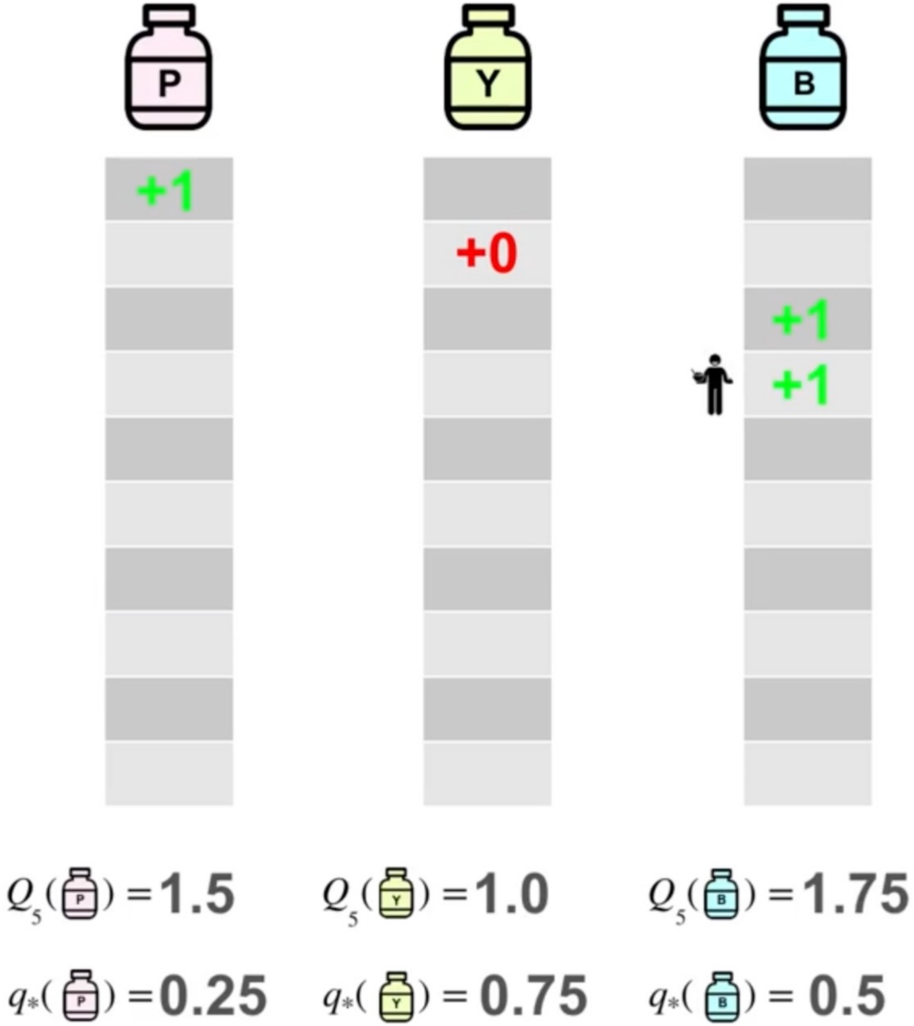


Optimistic Initial Values

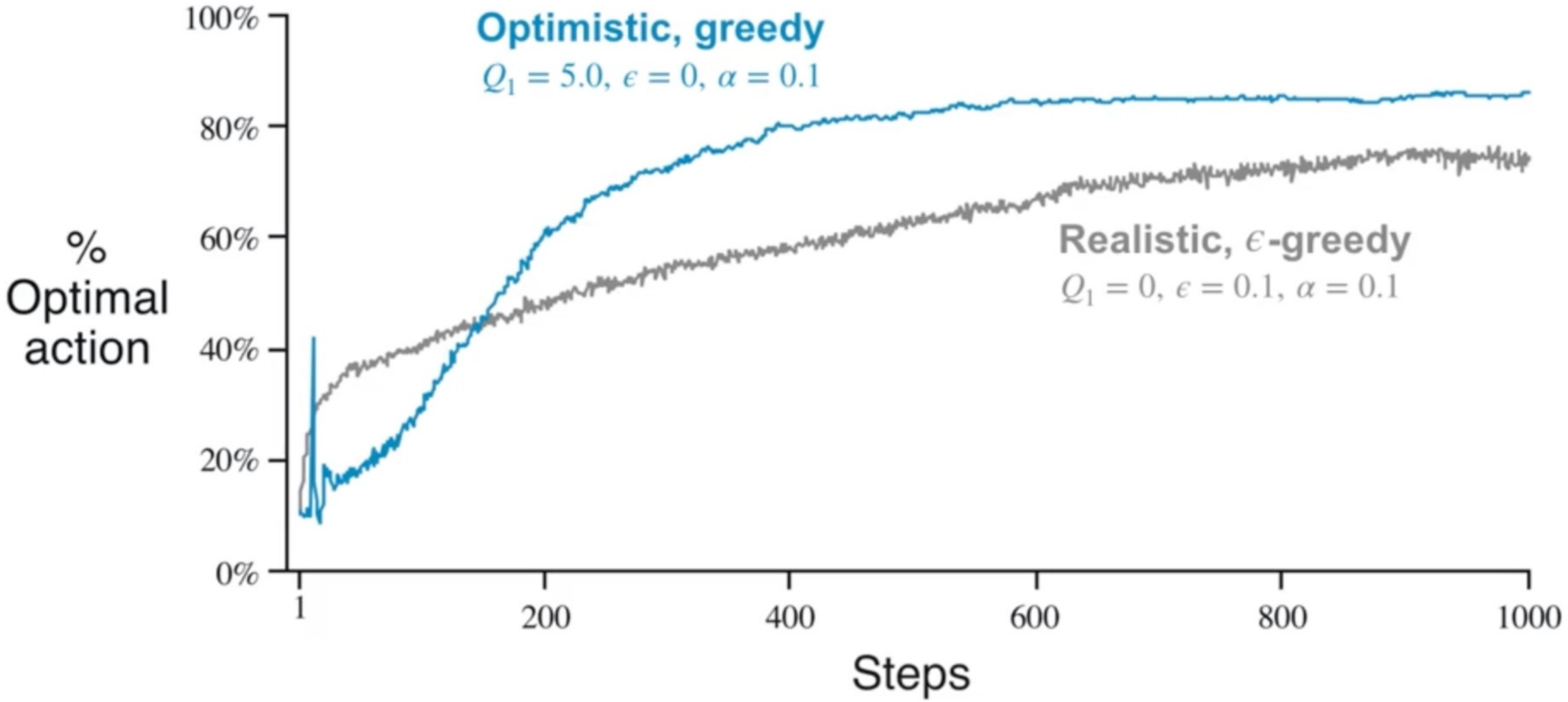
A reward of 1 if the treatment succeeds otherwise 0

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

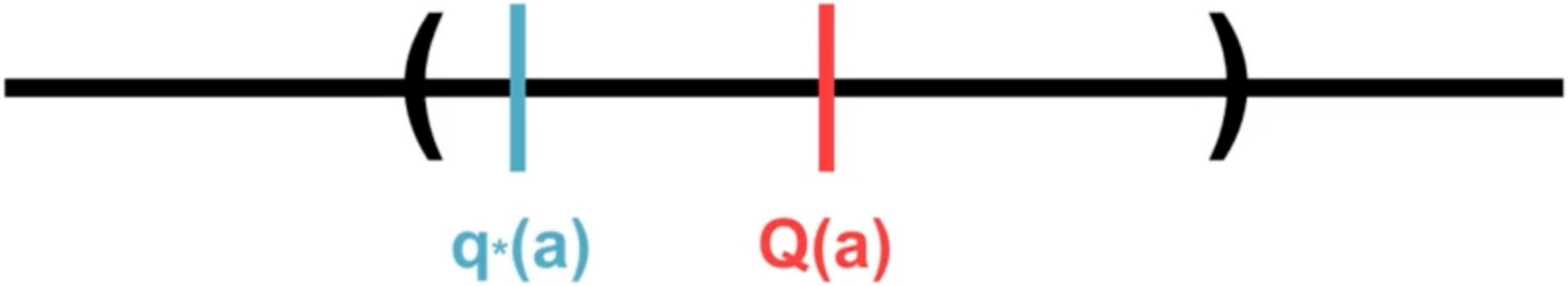
Let $\alpha = 0.5$



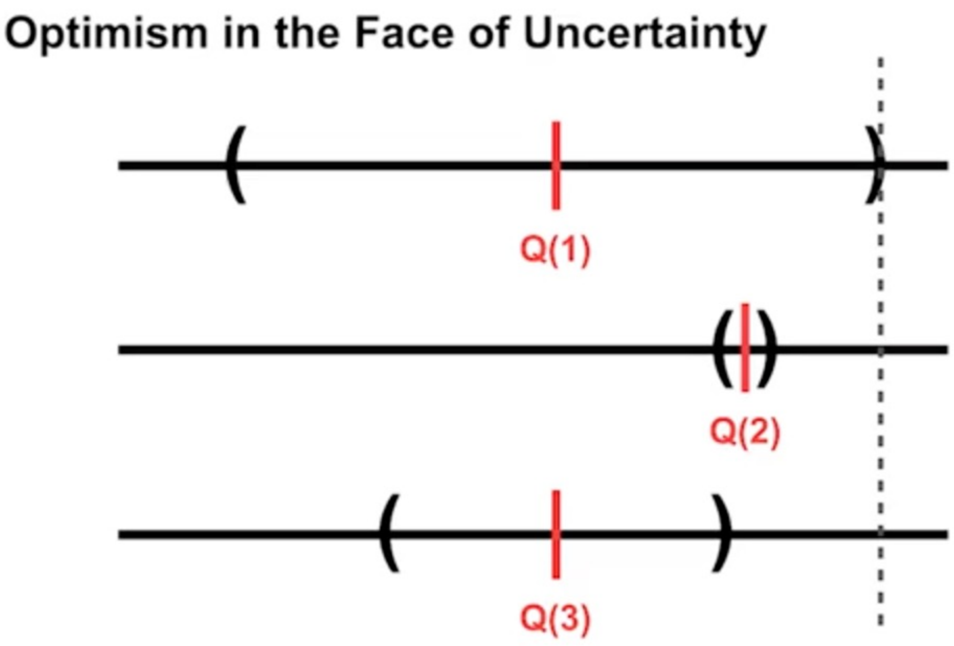
Optimistic Value Initialization in action



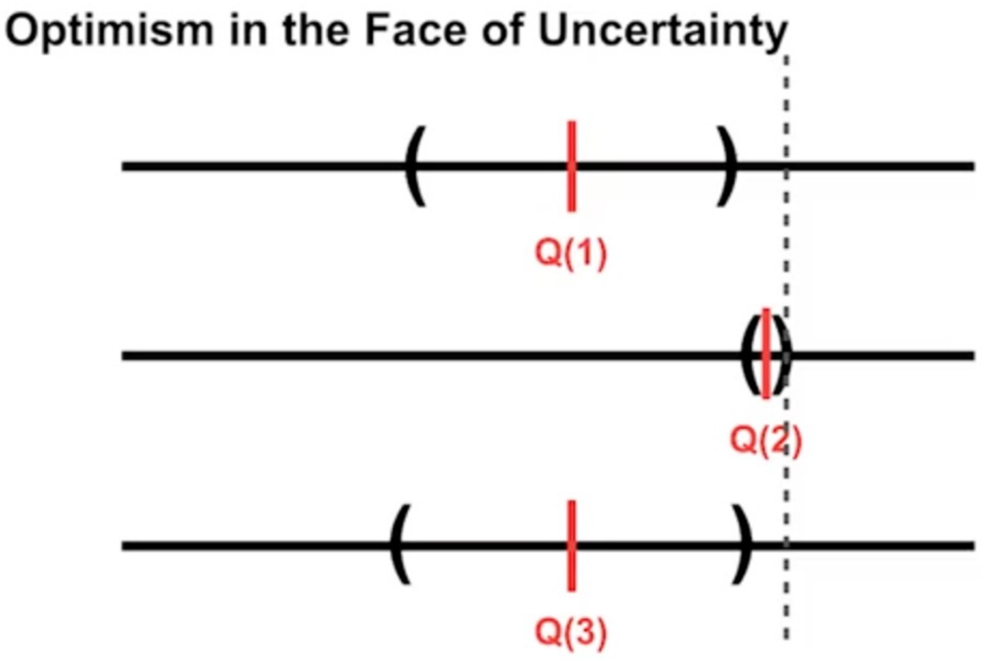
Uncertainty in Estimates



Optimism



Optimism in the Face of Uncertainty
Q(1)
Q(2)
Q(3)



Optimism in the Face of Uncertainty
Q(1)
Q(2)
Q(3)

Upper Confidence Bound Action Selection

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

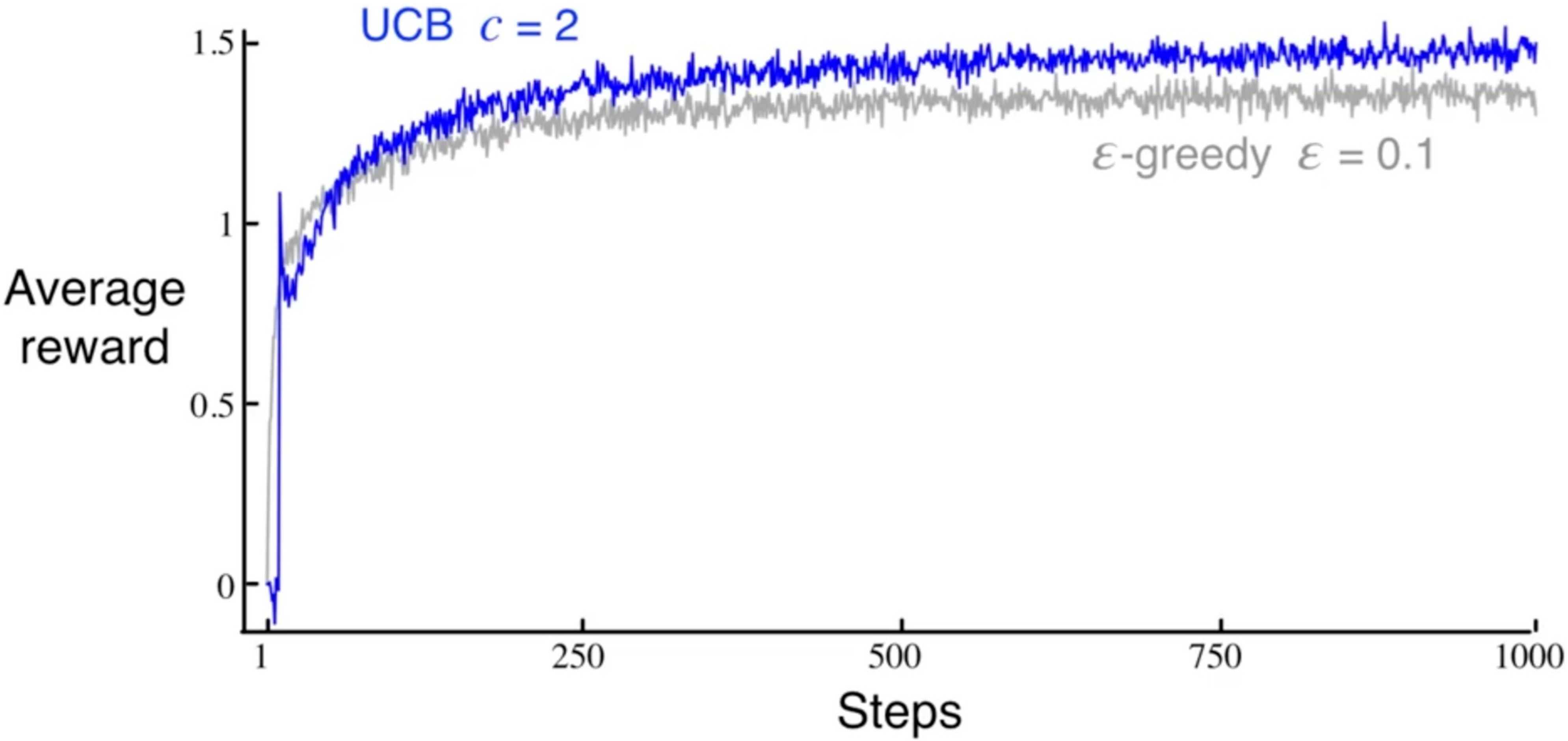
Upper Confidence Bound Action Selection

$$c\sqrt{\frac{\ln t}{N_t(a)}} \longrightarrow c\sqrt{\frac{\ln \text{timesteps}}{\text{times action } a \text{ taken}}}$$

$$c\sqrt{\frac{\ln 10000}{5000}} \longrightarrow 0.043c$$

$$c\sqrt{\frac{\ln 10000}{100}} \longrightarrow 0.303c$$

Eps-greedy vs UCB



Contextual Bandits

- Contextual bandit algorithm in round t
  - Algorithm observers user $\boldsymbol{u_t}$ and a set $\mathbf{A}$ of arms together with their features $\boldsymbol{x_{t,a}}$(context)
  - Based on payoffs from previous trials, algorithm chooses arm $\boldsymbol{a} \in \mathbf{A}$ and receives payoff $\boldsymbol{r_{t,a}}$
  - Algorithm improves arm selection strategy with each observation($\boldsymbol{x_{t,a}}, \boldsymbol{a}, \boldsymbol{r_{t,a}}$)

LinUCB Algorithm

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

$\boldsymbol{\theta_a^*}$ is the unknown coefficient vector we **aim to learn**

$$\text{Payoff of arm a : } \mathrm{E}\left[r_{t,a} \middle| x_{t,a}\right] = [x_{t,a}]^T \theta_a^*$$

$\boldsymbol{x_{t,a}}$ 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

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

$$\widehat{\boldsymbol{\theta_a}} = \boldsymbol{argmin_\theta} \sum_{\boldsymbol{m \in D_a}} ([x_{t,a}]^T \theta_a \; - \; \boldsymbol{b_a^{(m)}})^2$$

We can get:

$$\widehat{\boldsymbol{\theta_a}} = (\boldsymbol{D_a^T D_a} + \boldsymbol{I_d})^{-1}\, \boldsymbol{D_a^T b_a}$$

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

$\boldsymbol{b_a}$ is a m-dimension vector of responses to $\boldsymbol{a}$(click/no-click)

LinUCB Algorithm

- Using similar techniques as we used for UCB

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

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

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

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

LinUCB Algorithm

$A_a \stackrel{\text{def}}{=} \boldsymbol{D_a^T D_a} + \boldsymbol{I_d}$

- Initialization:
  - For each arm a:
    - $A_a = I_d$ //identity matrix d × 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
      - $p_{t,a} = [x_{t,a}]^T \theta_a + \boldsymbol{\alpha} \sqrt{[x_{t,a}]^T A_a^{-1} x_{t,a}}$
    - Choose arm $a_t = argmax_a p_{t,a}$ //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 ~ Bernoulli(θ)
- Let's assume that
  - θ ~ Beta(α_H, α_T) Beta distribution
  - P(θ) $\propto \theta^{\alpha_H - 1}\,(1-\theta)^{\alpha_T - 1}$

Prior

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

Posterior

The prior is conjugate!

Algorithm

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

Algorithm 1: Thompson Sampling for Bernoulli bandits

$S_i = 0, F_i = 0.$

foreach $t = 1, 2, \ldots,$ **do**

    For each arm $i = 1, \ldots, N$, sample $\theta_i(t)$ from the 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

Beta(1,1) Beta(1,1) Beta(1,1)

Arm 1 Arm 2 Arm 3

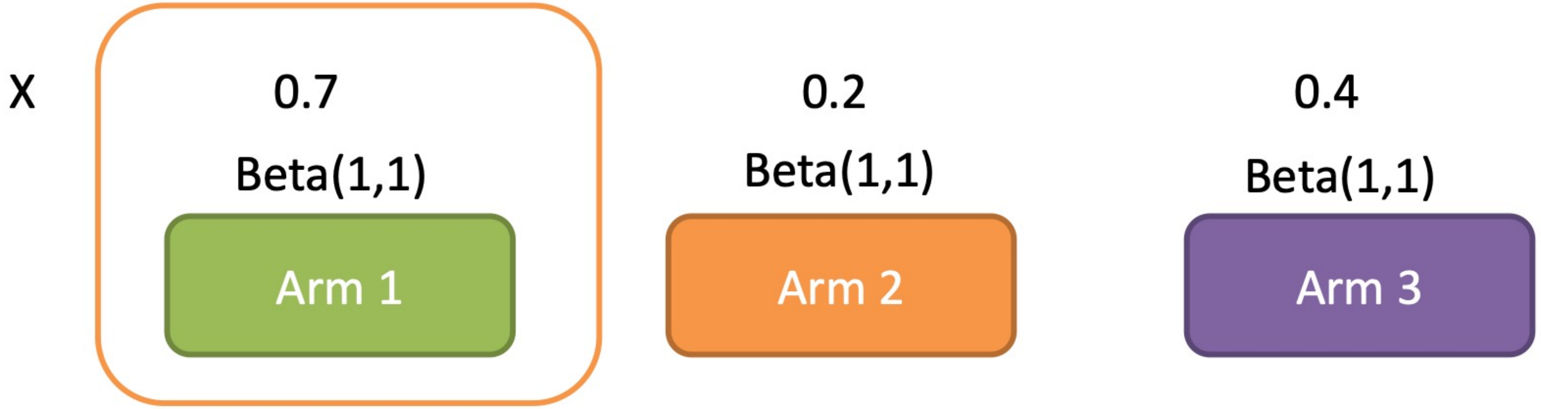
Algorithm

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

X	0.7	0.2	0.4
	Beta(1,1)	Beta(1,1)	Beta(1,1)
	Arm 1	Arm 2	Arm 3

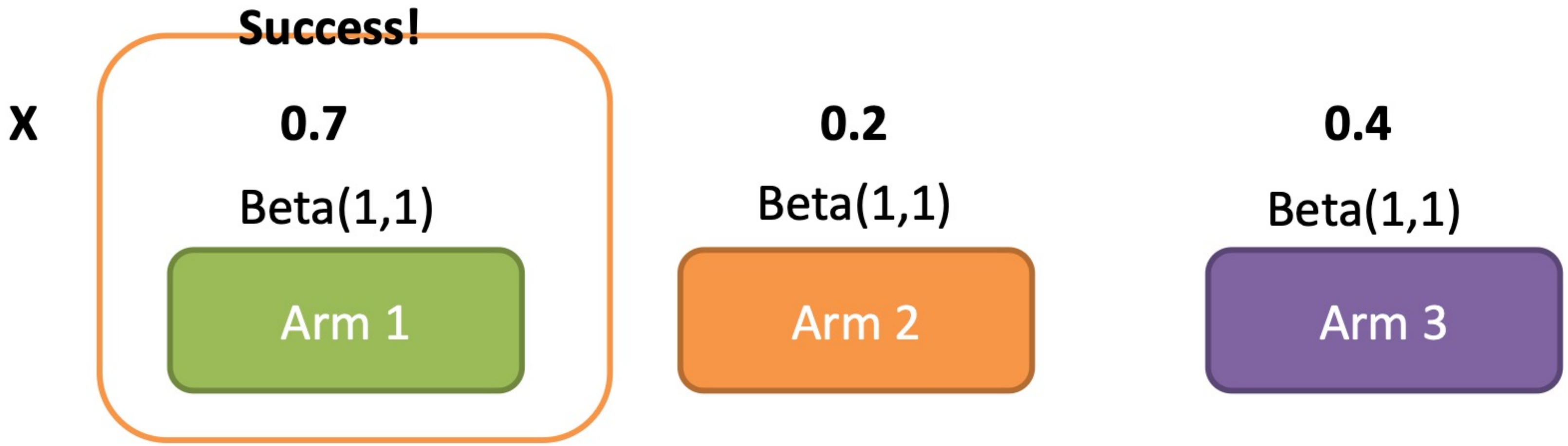
Algorithm

- 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