Multi-armed Bandits

Deep Reinforcement Learning

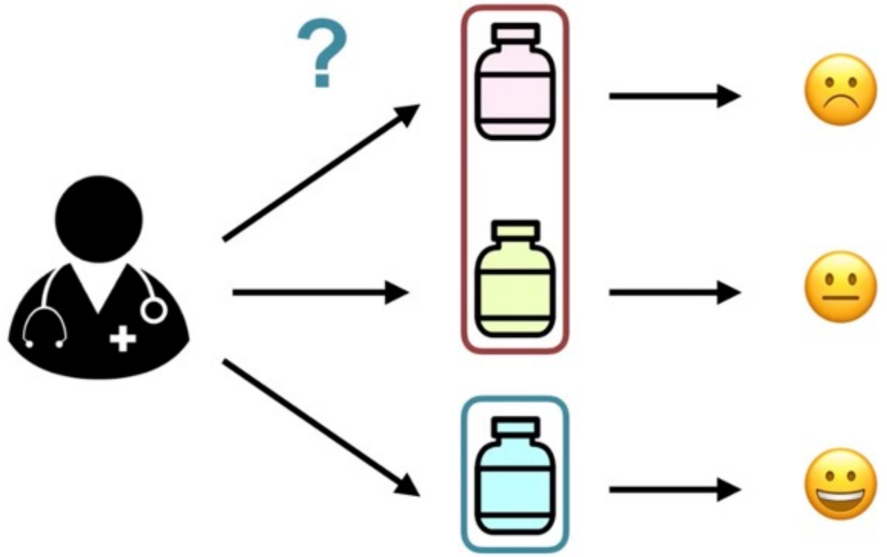
Mohammad Hossein Rohban, Ph.D.

Courtesy: Some slides are adopted from CS 285 Berkeley, and CS 234 Stanford, and Pieter Abbeel's compact series on RL.

RIML Lab, CE Department, Sharif University of Technology

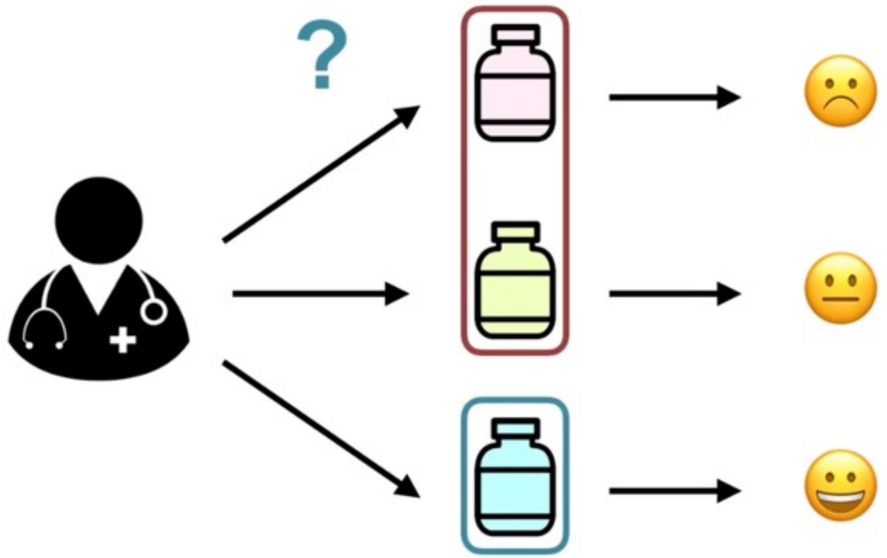
Spring 2026

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

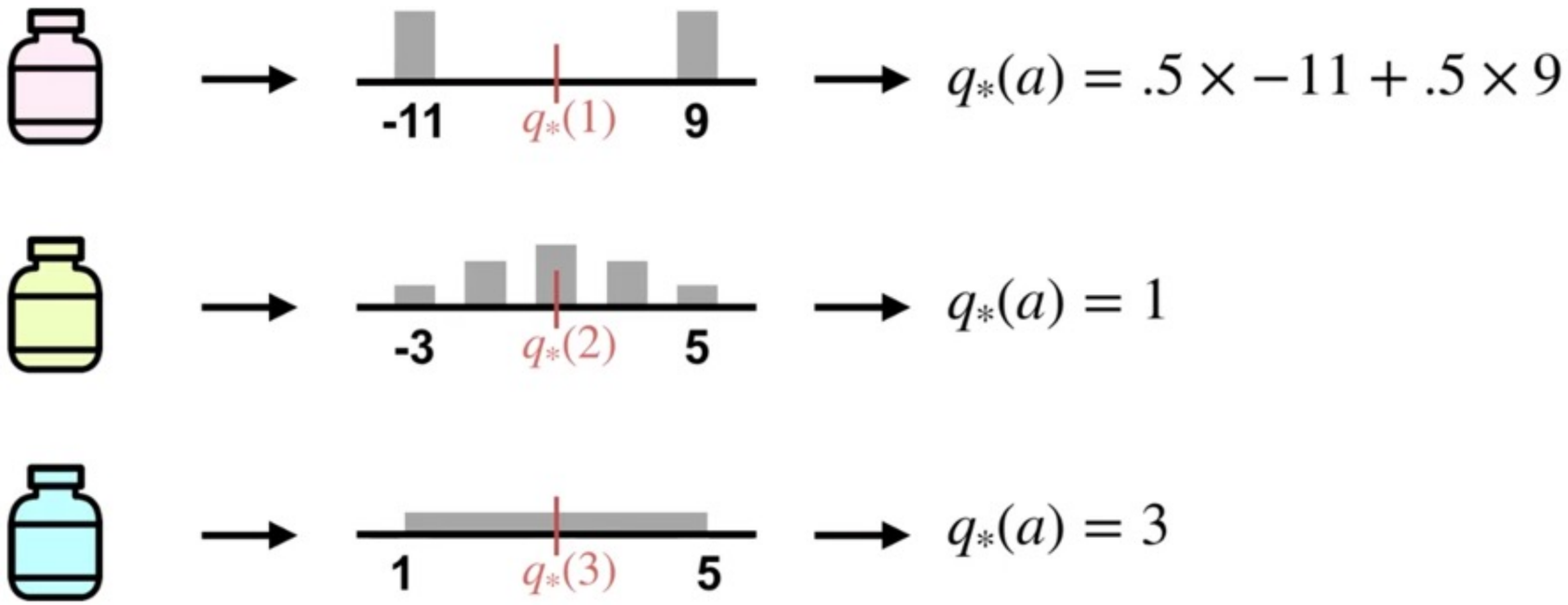
- The value of an action is the expected reward when that action is taken

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

- The goal is to maximize the expected reward

$$\underset{a}{\text{argmax}} q_*(a)$$

Calculating q(a)



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

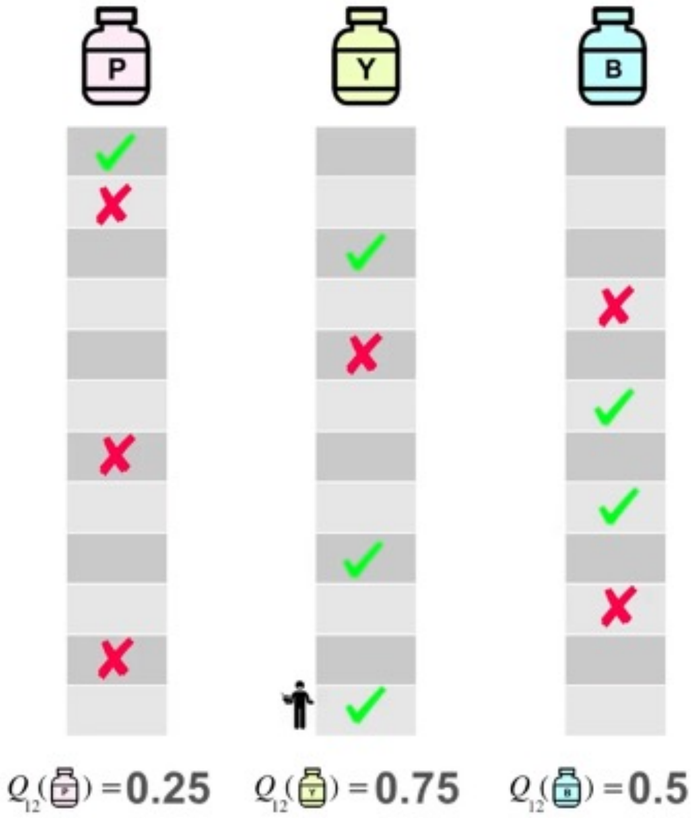
$$q_*(a) \doteq \mathbb{E}[R_t | A_t = a]$$

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

Sample Average

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

Sample Average (cont.)



Incremental Update Rule

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

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

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

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

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

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

Incremental Update Rule

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

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

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

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

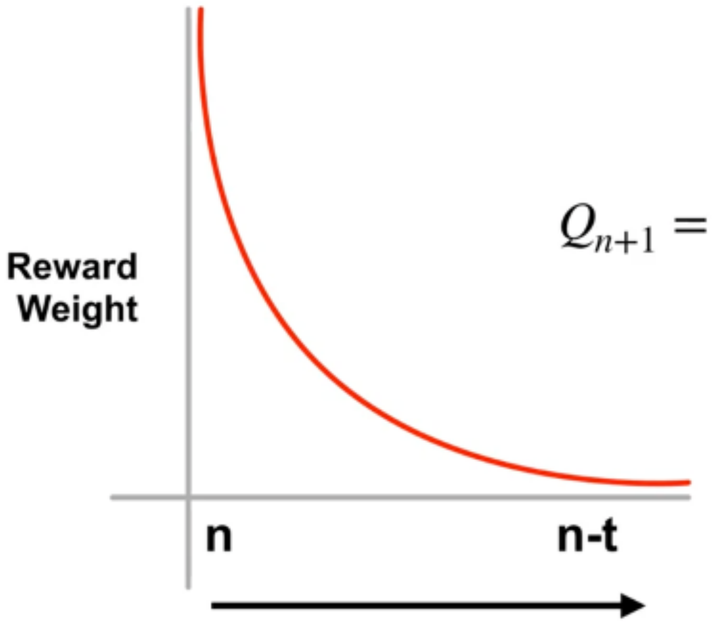
$$= \frac{1}{n}(R_n + nQ_n - Q_n)$$

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

Non-Stationary Bandit Problem

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

Non-Stationary Bandit Problem (cont.)



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

Decaying Past Reward

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

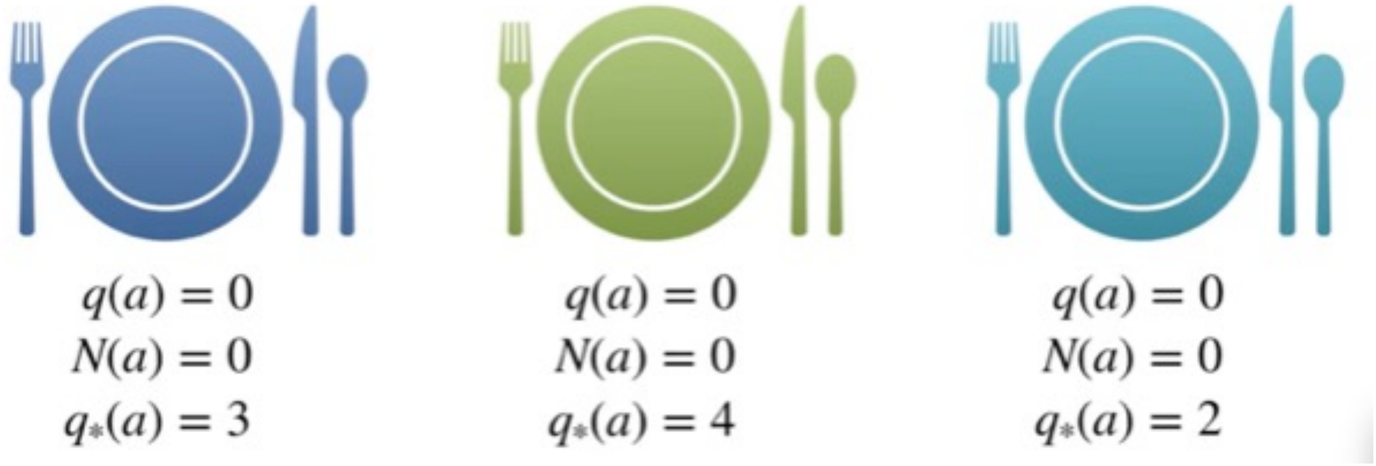
$$
\begin{aligned}
Q_{n+1} &= Q_n + \alpha(R_n - Q_n) \\
&= \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2 \alpha R_{n-2} + \cdots + (1-\alpha)^{n-1} \alpha R_1 + (1-\alpha)^n Q_1
\end{aligned}
$$

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

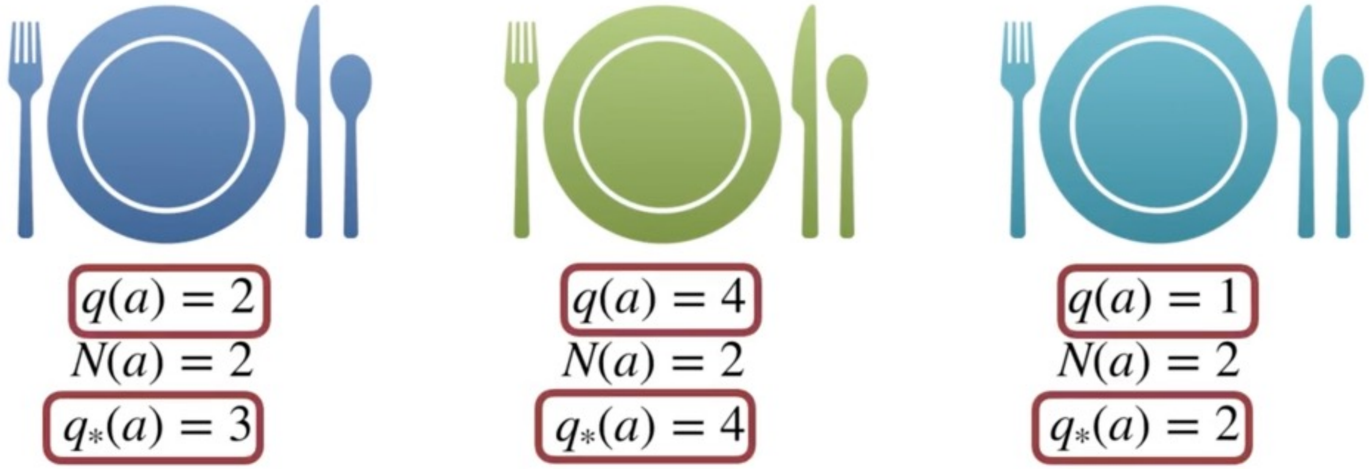
$Q_1 \rightarrow$ initial action-value

Exploration and Exploitation

- Exploration - improve knowledge for long-term benefit

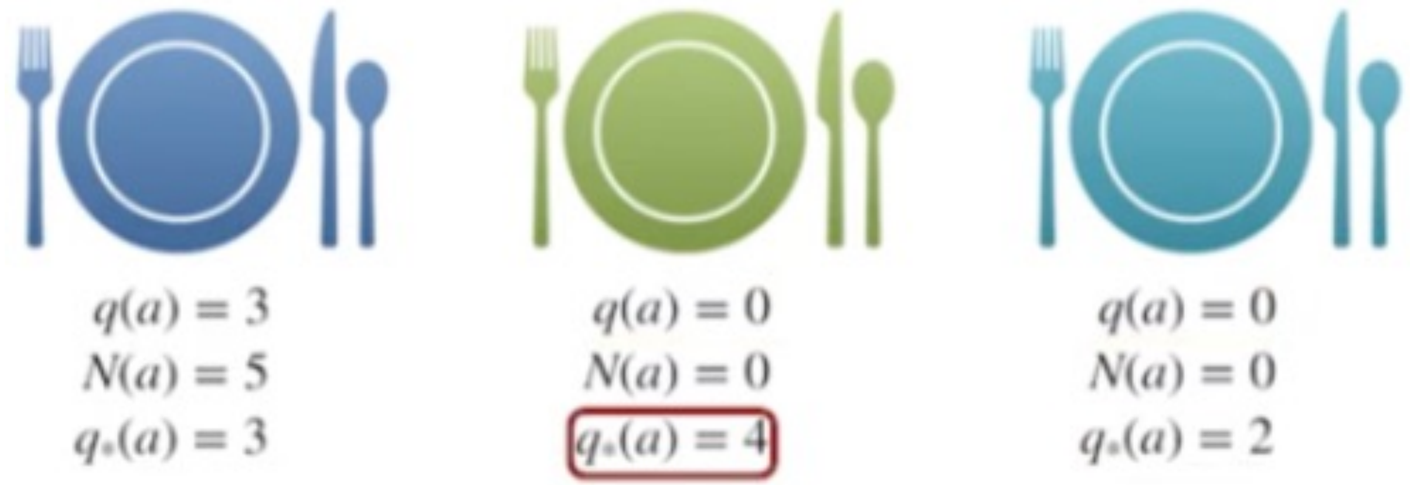


Exploration and Exploitation (cont.)



Exploration and Exploitation (cont.)

- Exploration - improve knowledge for long-term benefit



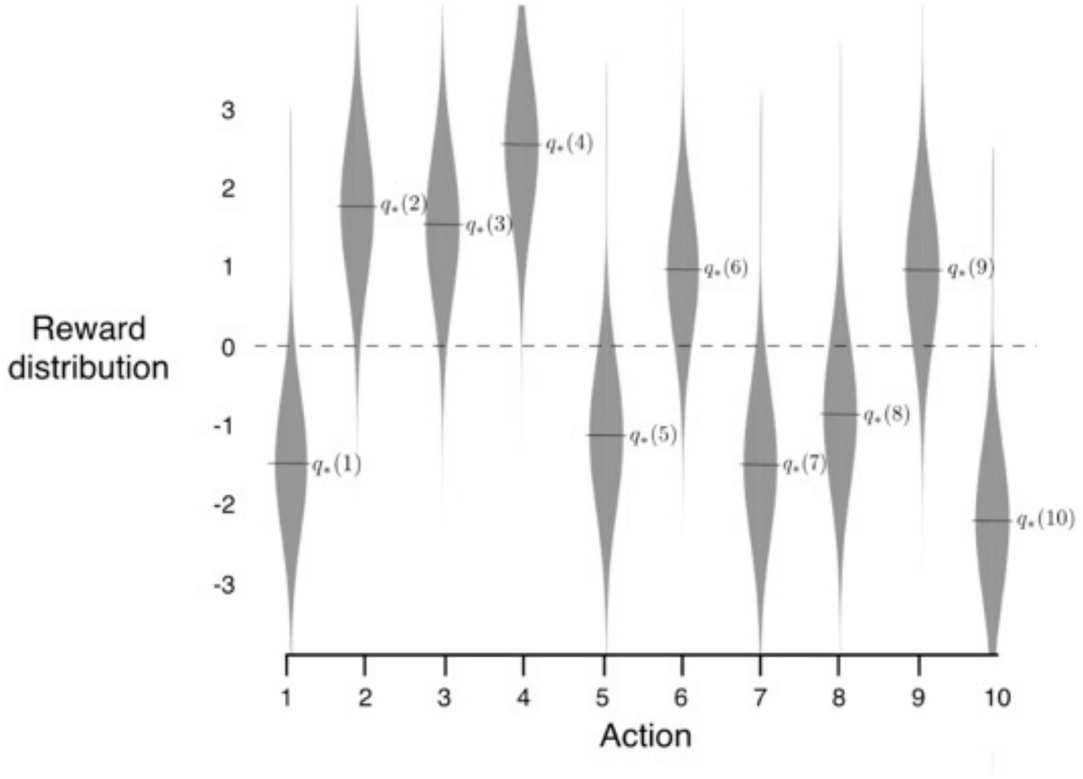
Exploration and Exploitation (cont.)

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

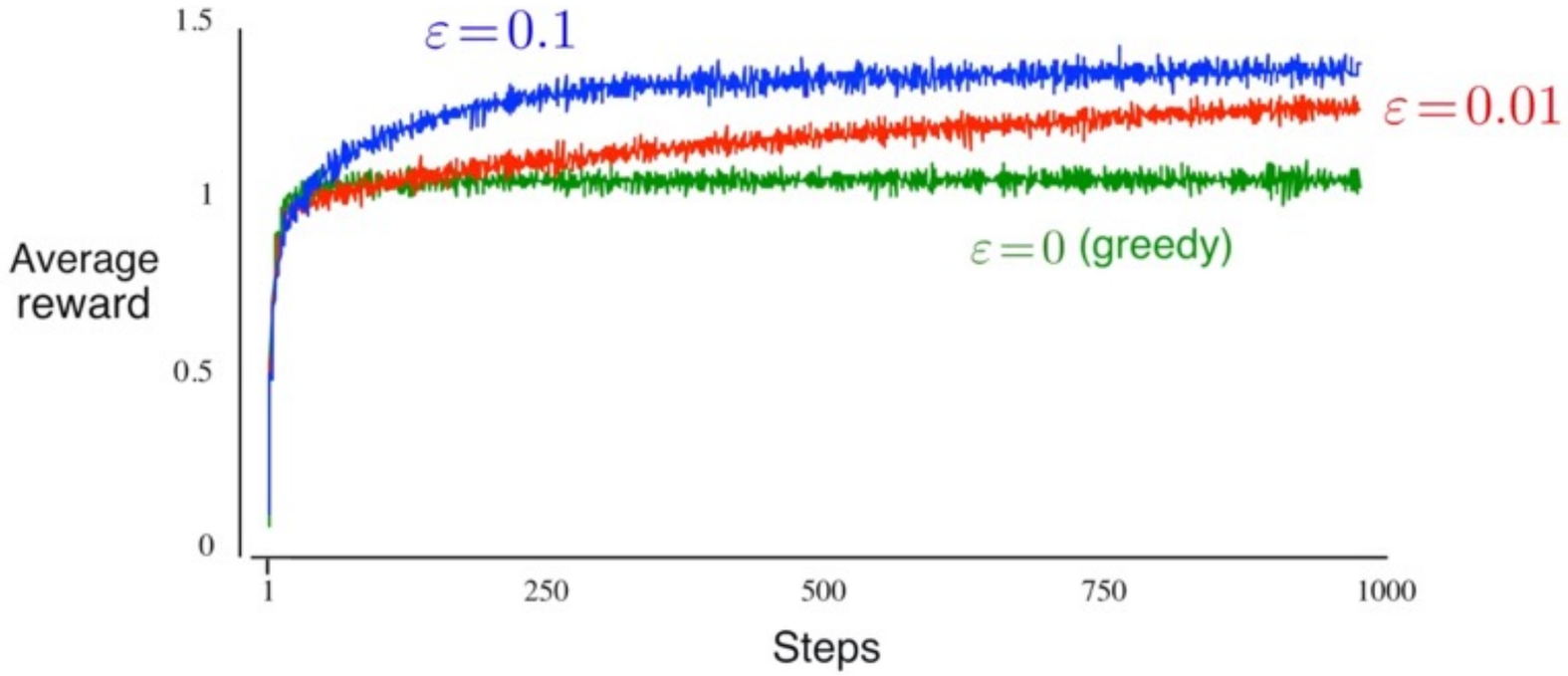
ϵ-Greedy Action Selection

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

The 10-armed testbed



Average reward for ϵ-greedy



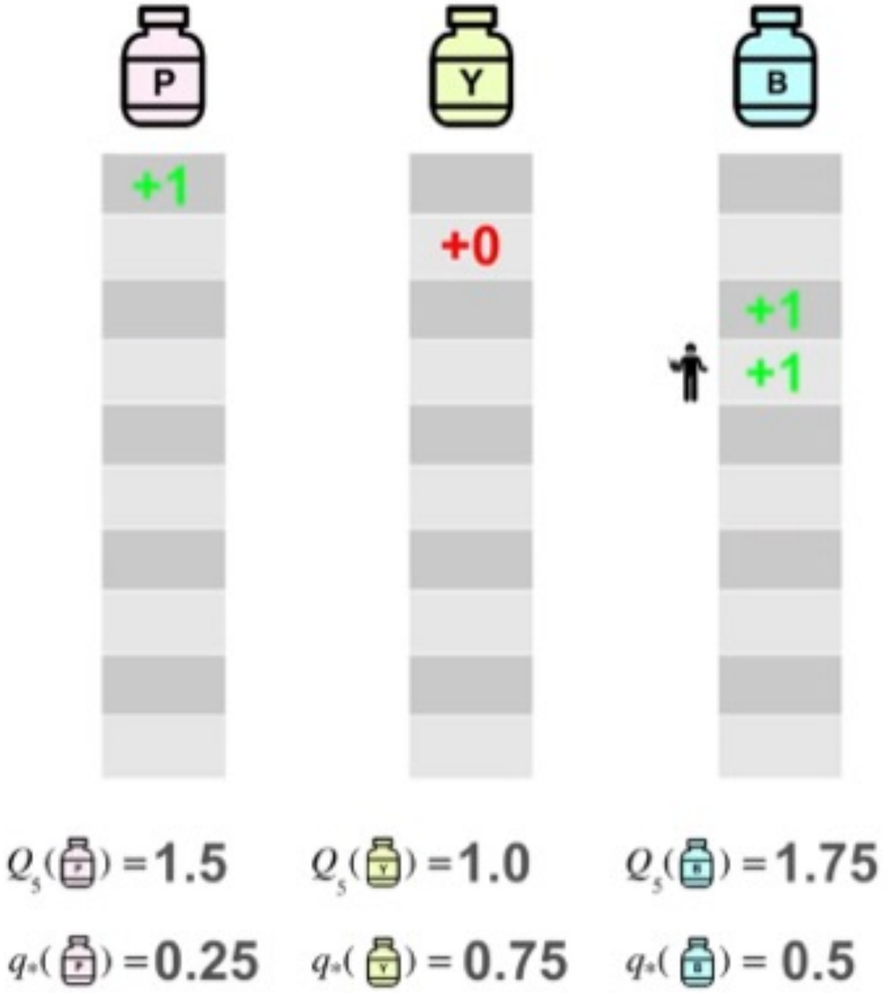
Optimistic Initial Values

- A reward of 1 if the treatment succeeds

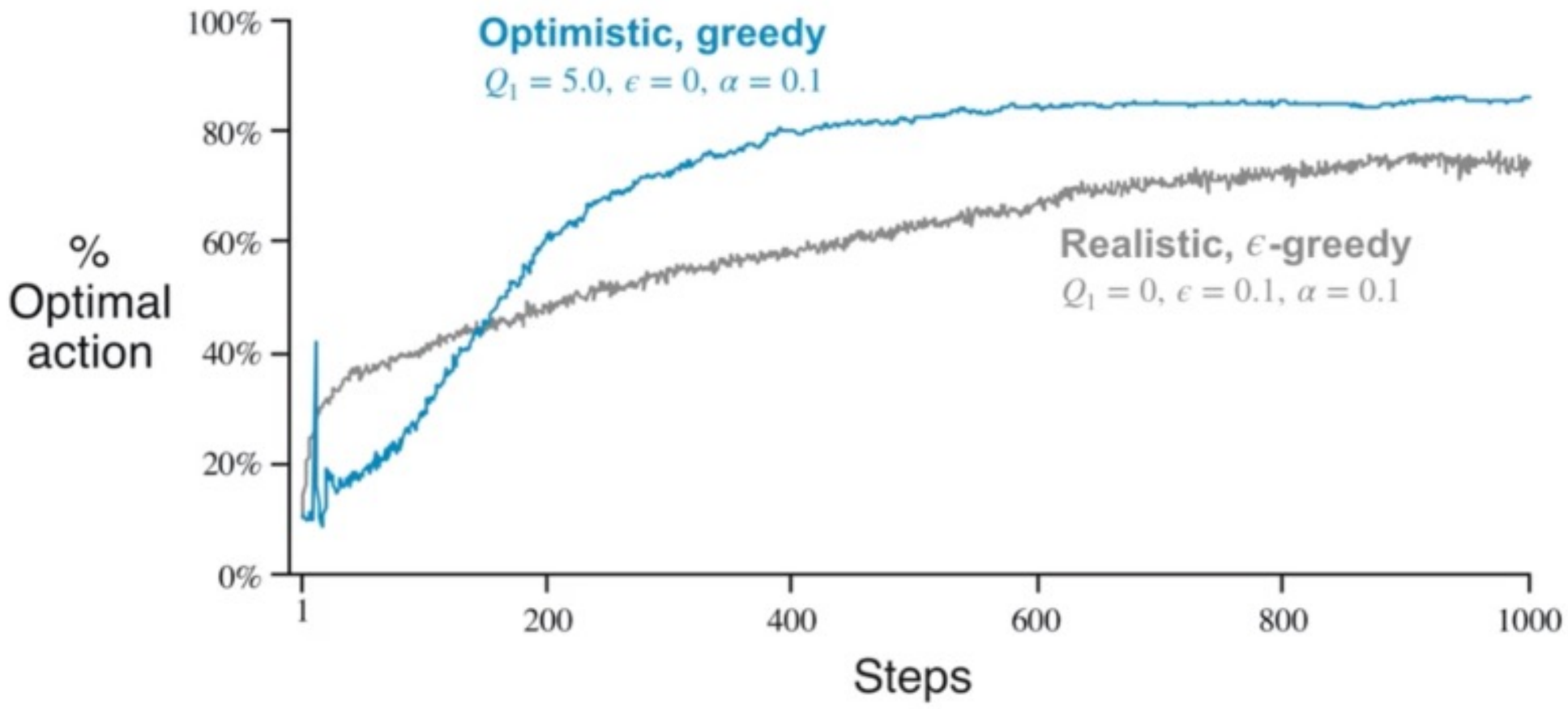
  otherwise 0

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

let $\alpha = 0.5$

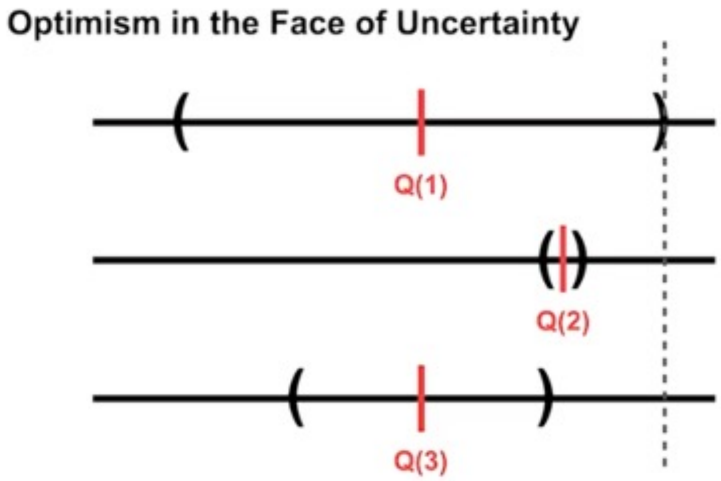


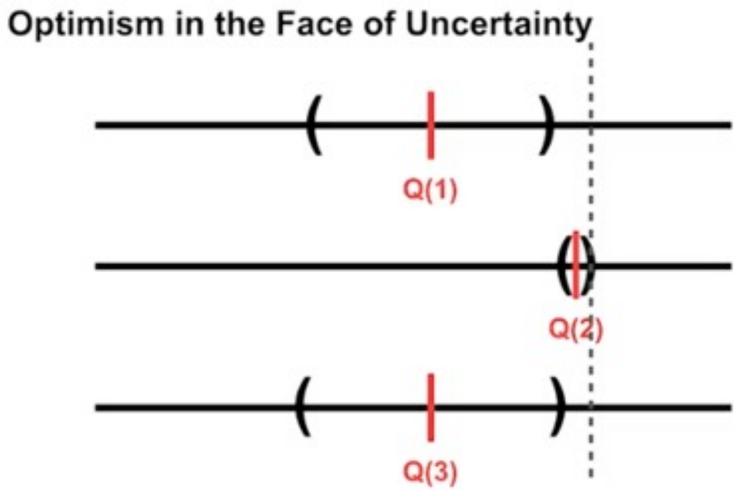
Optimistic Value Initialization in action



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)

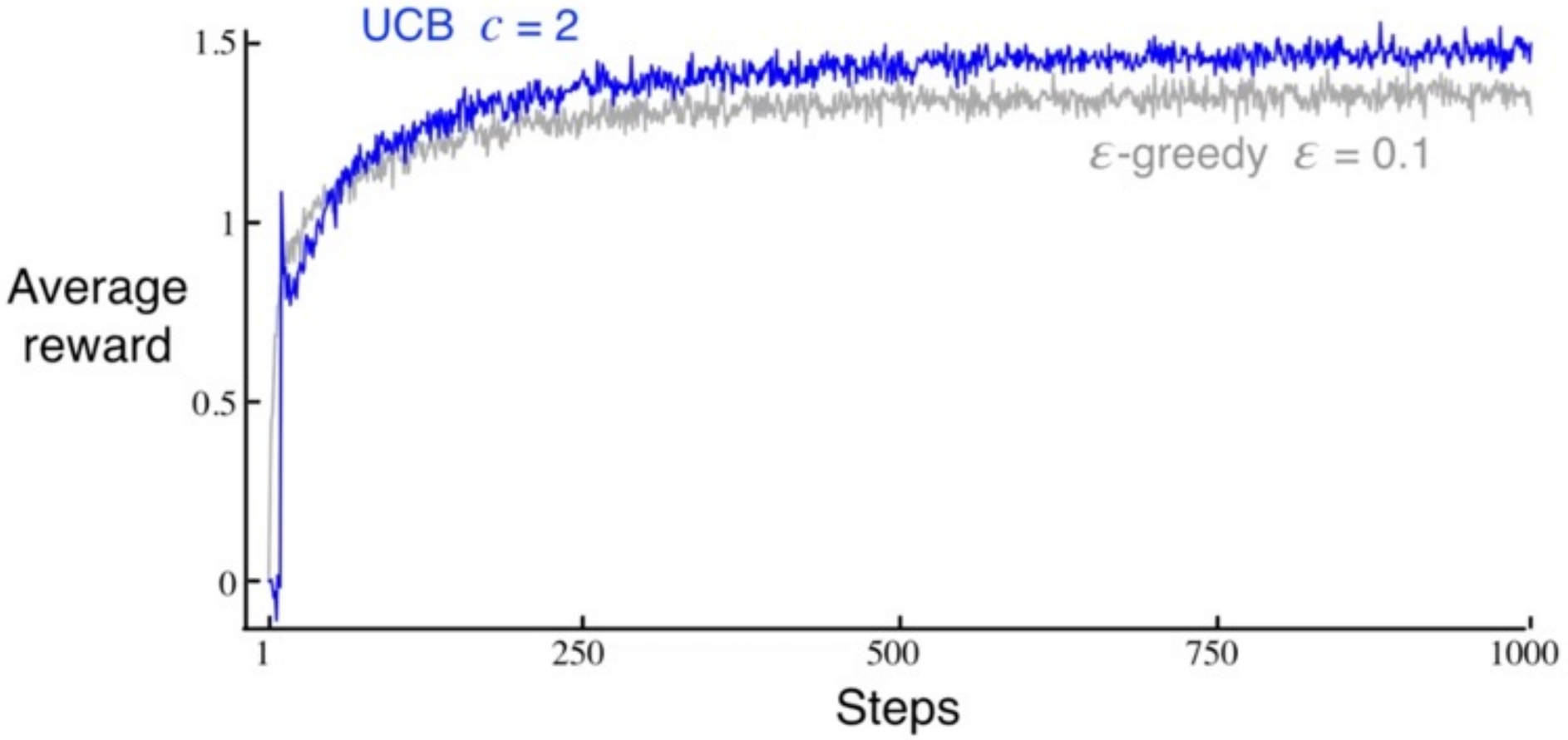
Upper Confidence Bound Action Selection

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

Upper Confidence Bound Action Selection (cont.)

$$c\sqrt{\frac{\ln t}{N_t(a)}} \rightarrow c\sqrt{\frac{\ln \text{timesteps}}{\text{times action } a \text{ taken}}} \begin{cases} c\sqrt{\frac{\ln 10000}{5000}} \rightarrow 0.043c \\ c\sqrt{\frac{\ln 10000}{100}} \rightarrow 0.303c \end{cases}$$

ϵ-greedy vs UCB



Contextual Bandits

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

LinUCB Algorithm

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

$\theta *_a$ is the unknown coefficient vector we aim to learn

Payoff of arm a: $\mathbb{E}[r_{t,a} \mid x_{t,a}] = x_{t,a}^{\top}\theta_a^*$

$x_{t,a}$ 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.

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

LinUCB Algorithm (cont.)

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

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

We can get:

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

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 (cont.)

- Using similar techniques as we used for UCB

$$\left| x_{t,a}^{\top} \hat{\theta}_a - \mathbb{E}[r_{t,a} \mid x_{t,a}] \right| \leq \alpha \sqrt{x_{t,a}^{\top} (D_a^{\top} D_a + I_d)^{-1} x_{t,a}}$$

where

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

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

$$a_t \overset{def}{=} \underset{a \in A_t}{\arg max} \left(x_{t,a}^{\top} \hat{\theta}_a + \alpha \sqrt{x_{t,a}^{\top} (D_a^{\top} D_a + I_d)^{-1} x_{t,a}} \right)$$

Estimated μ_a — confidence interval term

LinUCB Algorithm (cont.)

LinUCB Algorithm

1. **Initialization:** $A_a \overset{def}{=} D_a^T D_a + I_d$
2. **For each arm :**
3. $A_a = I_d$ — identity matrix $d \times d$
4. $b_a = [0]_d$ — vector of zeros

6. **Online algorithm:**
7. **For t=[1:T]:**
8. **Observe features for all arms** $a: x_{t,a} \in R^d$
9. **For each arm** a**:**
10. $\theta_a = A_a^{-1} b_a$ — regression coefficients
11. $p_{t,a} = [x_{t,a}]^T \theta_a + \alpha \sqrt{[x_{t,a}]^T A_a^{-1} x_{t,a}}$
12. **Choose arm** $a_t = \text{argmax}_a p_{t,a}$ — choose arm
13. $A_{a_t} = A_{a_t} + x_{t,a_t} [x_{t,a_t}]^T$ — update A for the chosen arm a_t
14. $b_{a_t} = b_{a_t} + r_t x_{t,a_t}$ — 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: $x \sim Bernoulli(\theta)$
- Let's assume that
  - $\theta \sim Beta(\alpha_H, \alpha_T)$ (Beta distribution)
  - $P(\theta) \propto \theta^{\alpha_H - 1}(1-\theta)^{\alpha_T - 1}$
- $P(\theta|X) = \frac{P(X|\theta)P(\theta)}{\sum_\theta P(X|\theta)}$ (Prior)

Posterior

the prior is conjugate!

Algorithm

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

Thompson Sampling for Bernoulli bandits

$S_i = 0, F_i = 0.$

for each $t = 1,2, \dots,$ do

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

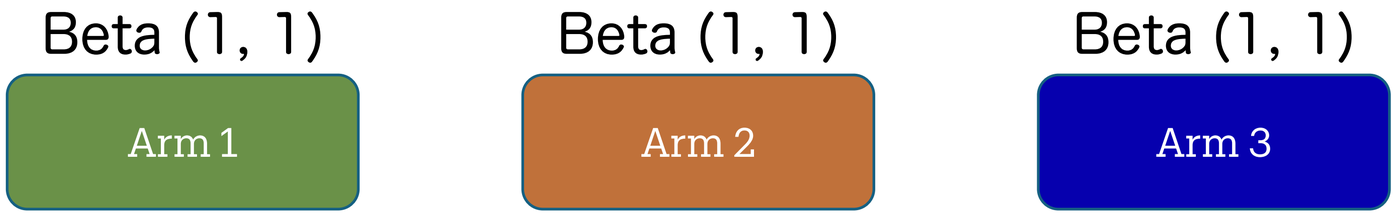
Play arm $i(t) := \arg max_i \theta_i(t)$ and observe reward r_t.

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

end

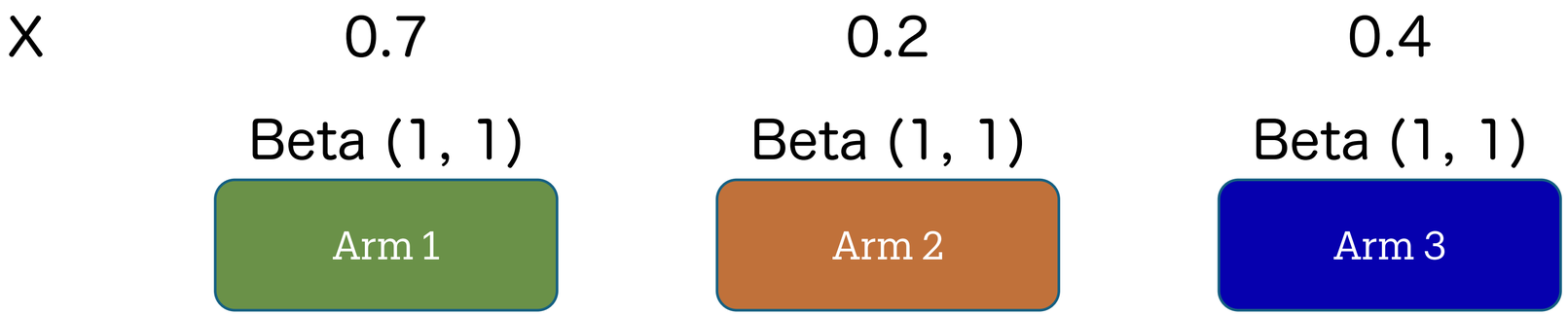
Algorithm (cont.)

- Initializing



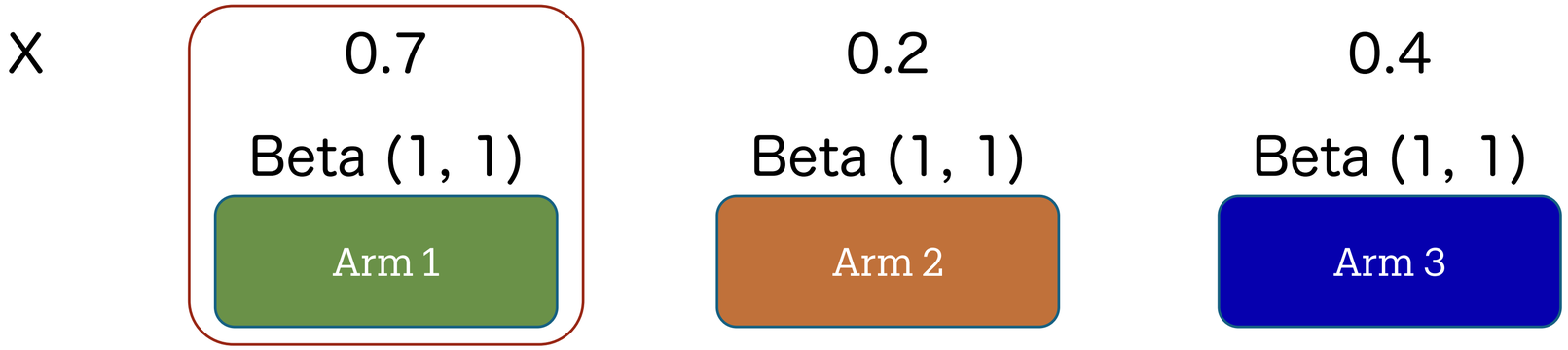
Algorithm (cont.)

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



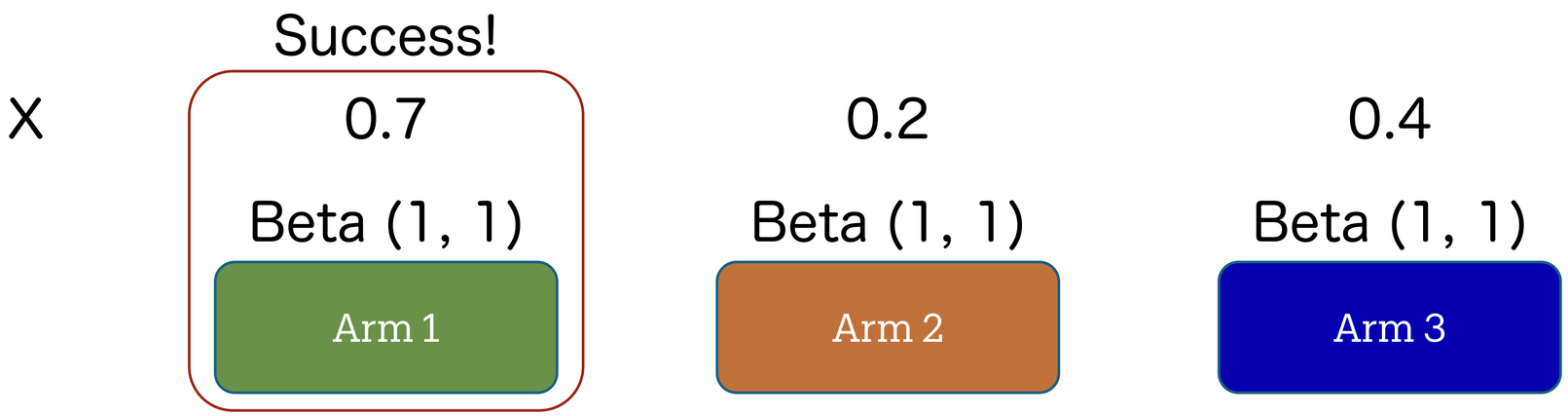
Algorithm (cont.)

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



Algorithm (cont.)

- 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 (cont.)

- 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 secelted arm