

MULTI-ARMED BANDITS

Scott O'Hara

Metrowest Boston Developers Machine Learning Group

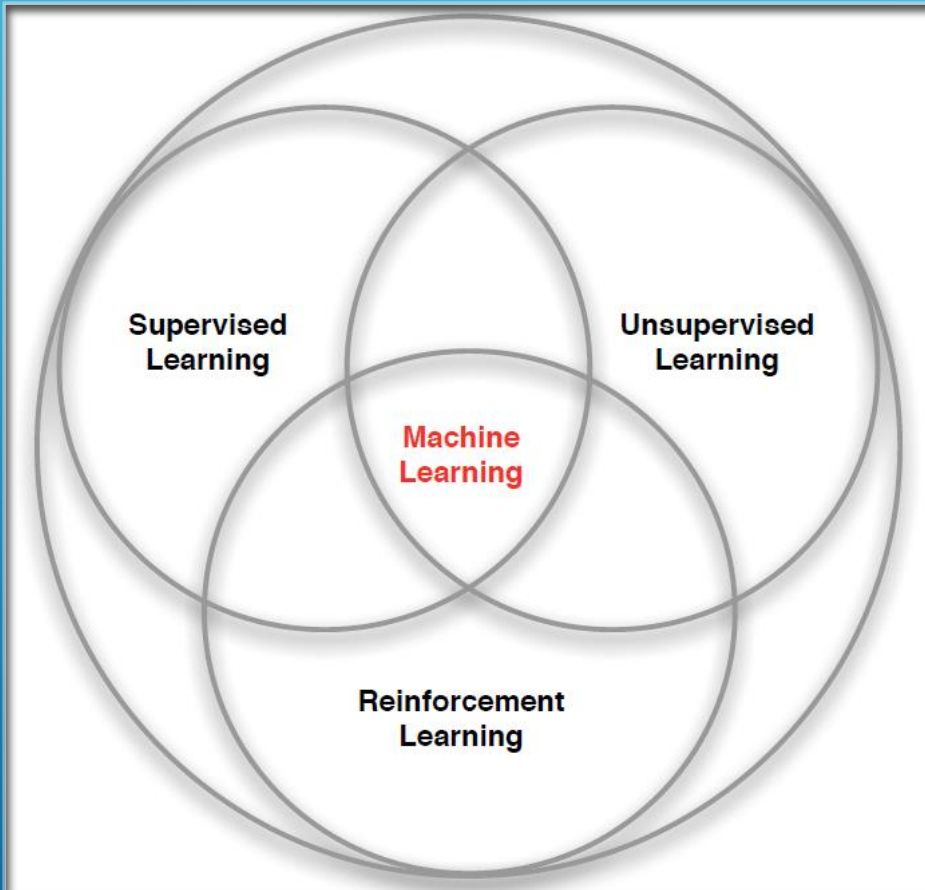
02/12/2020

REFERENCES

The material for this talk is primarily drawn from these resources:

- ▶ R. S. Sutton and A. G. Barto, ***Reinforcement learning: an introduction***, Second edition. Cambridge, Massachusetts: The MIT Press, 2018.
- ▶ M. White and A. White, “**Fundamentals of Reinforcement Learning**,” **Coursera**. [Online]. Available: <https://www.coursera.org/specializations/reinforcement-learning#courses>. [Accessed: 14-Feb-2020].
- ▶ R. Dechter, “**CompSci 295 Reinforcement Learning, Fall 2019, UC Irving**.” [Online]. This is a graduate-level seminar with pointers to many advanced resources on reinforcement learning. Available: <https://www.ics.uci.edu/~dechter/courses/ics-295/fall-2019/>. [Accessed: 14-Feb-2020]
- ▶ S. Arora and E. Hazan, “**Machine Learning and Artificial Intelligence**.” [Online]. Available: <https://www.cs.princeton.edu/courses/archive/fall16/cos402/>. [Accessed: 14-Feb-2020].

3 TYPES OF MACHINE LEARNING



Supervised Learning – Learn a function from labeled data that maps input attributes to an output label e.g., linear regression, decision trees, SVMs.

Unsupervised Learning – Learn patterns in unlabeled data e.g., principle component analysis or clustering algorithms such as K-means, HAC, or Gaussian mixture models.

Reinforcement Learning – An agent learns to maximize rewards while acting in an uncertain environment.

SOME CHARACTERISTICS OF REINFORCEMENT LEARNING

- Learning happens as the agent interact with the world.
- There are no training or test sets. Training is guided by rewards and punishments obtained by acting in an environment.
- The amount of data an agent receives is not fixed. More information is acquired as you go.
- Actions are not always rewarded or punished immediately. “Delayed gratification” is possible.
- Agent actions can affect the subsequent data it receives. E.g., closing a door that can’t be opened again.

MARKOV DECISION PROCESSES

- The **Markov Decision Process** (MDP) provides a mathematical framework for reinforcement learning.
- An MDP is used to model optimal decision-making in situations where outcomes are uncertain.

THE MDP DECISION FRAMEWORK

Markov decision processes model uncertainty

Use probability to model uncertainty about the domain.



Markov decision processes model an agent's objectives

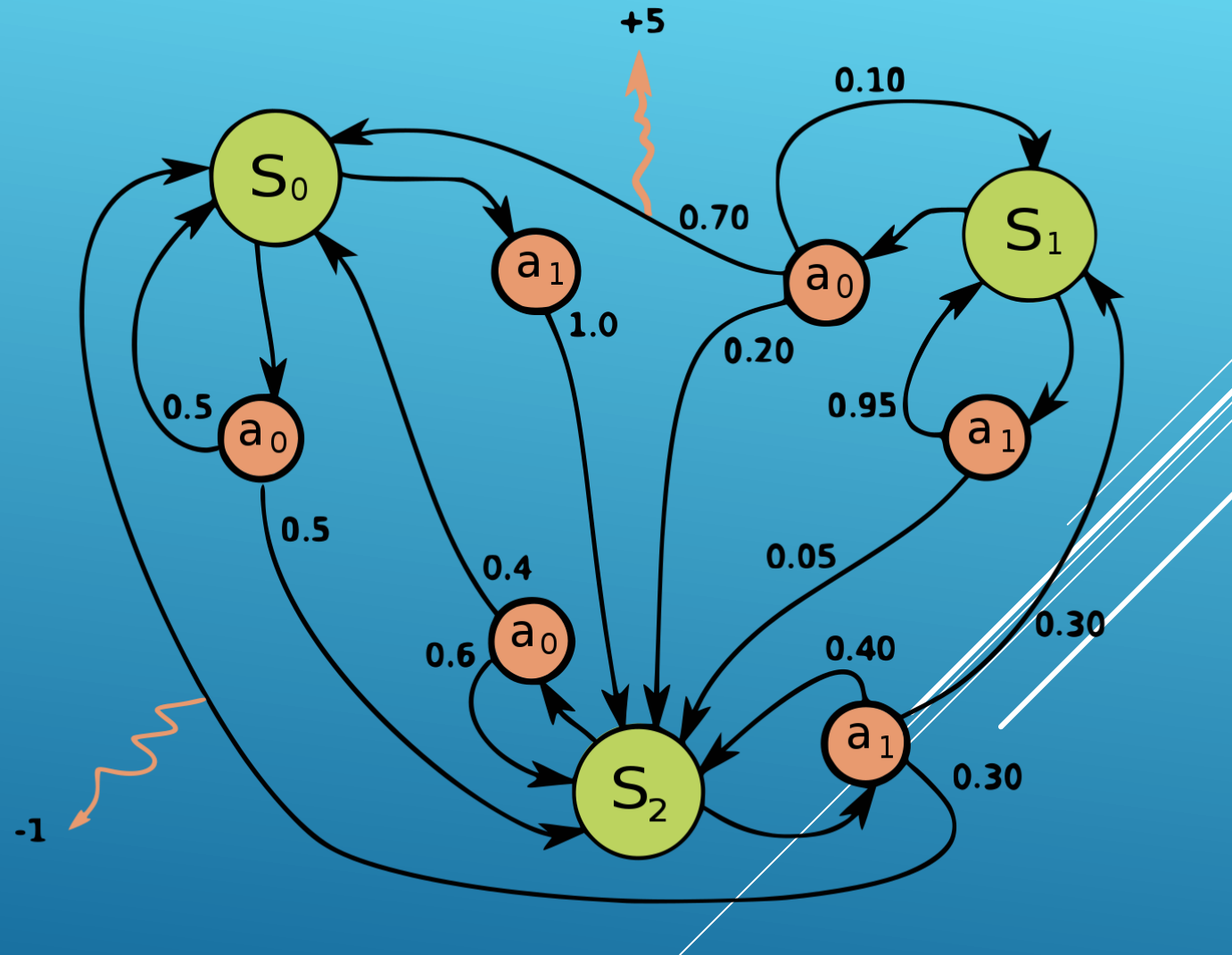
Use utility to model an agent's objectives. The higher the utility, the “happier” your agent is.

Markov decision processes find an optimal decision policy

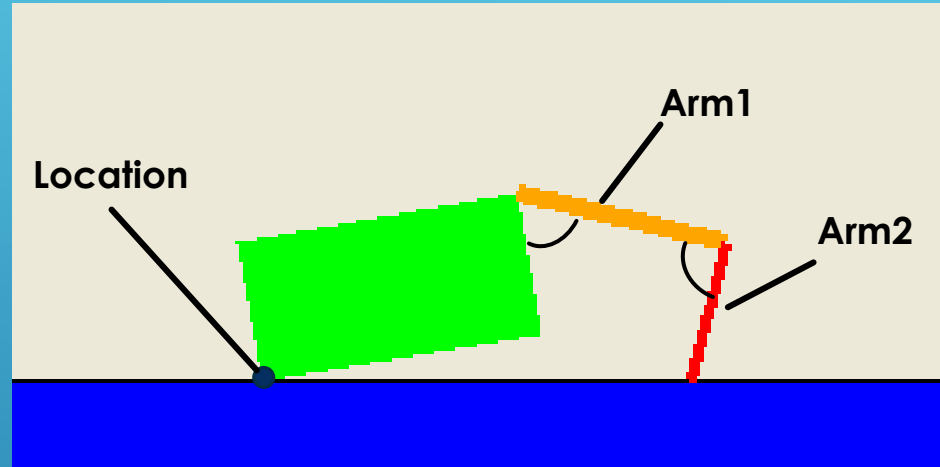
The goal is to discover an optimal decision policy π specifying how the agent should act in all possible states in order to maximize its expected utility.

MARKOV DECISION PROCESSES

- **States:** s_0, \dots, s_n
- **Actions:** a_0, \dots, a_m
- **Reward Function:** 
- **Transition model:** 
- **Discount factor:** $\gamma \in [0, 1]$



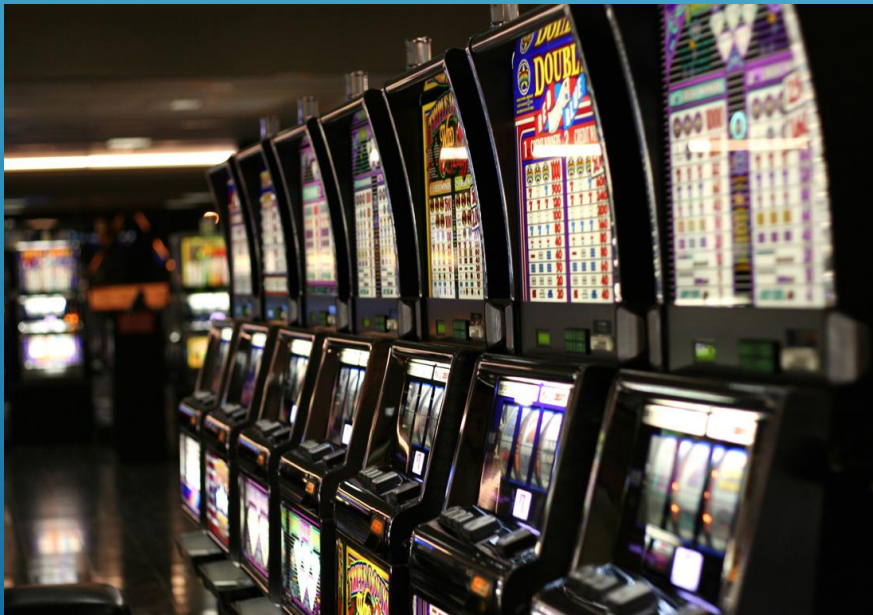
APPLICATION: CRAWLER ROBOT



- **States:** <Location, Arm1 angle, Arm2 angle>
- **Actions:** increase Arm1 angle, decrease Arm1 angle, increase Arm2 angle, decrease Arm2 angle.
- **Reward Function:** +1 if robot moves right, -1 if robot moves left.
- **Transition model:** model of box movement caused by arm movements.

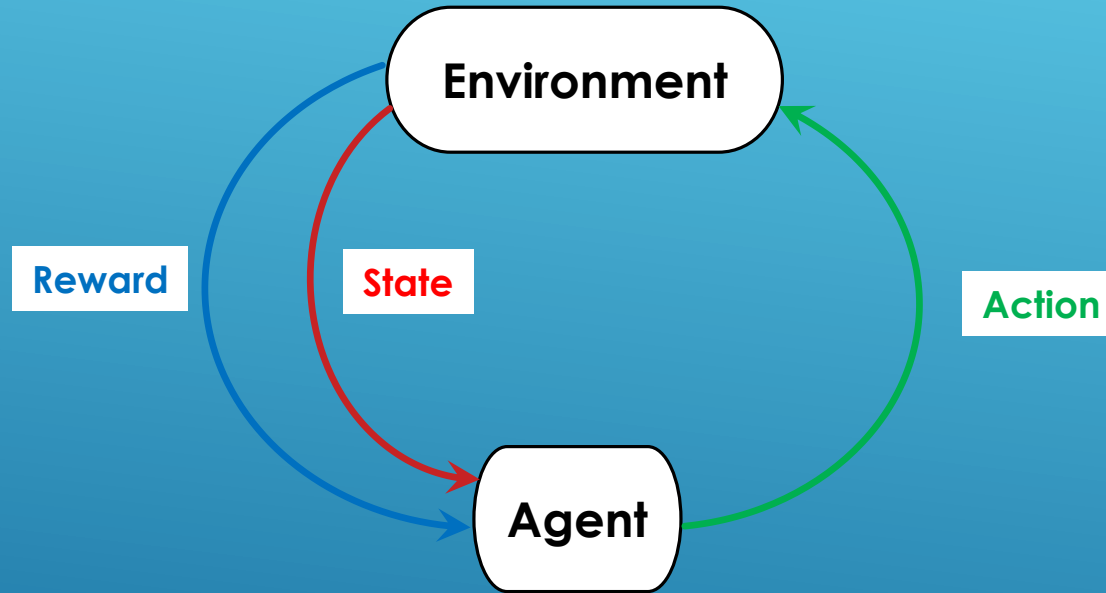
APPLICATION: THE MULTI-ARMED BANDIT

A Simple Reinforcement Learning Problem



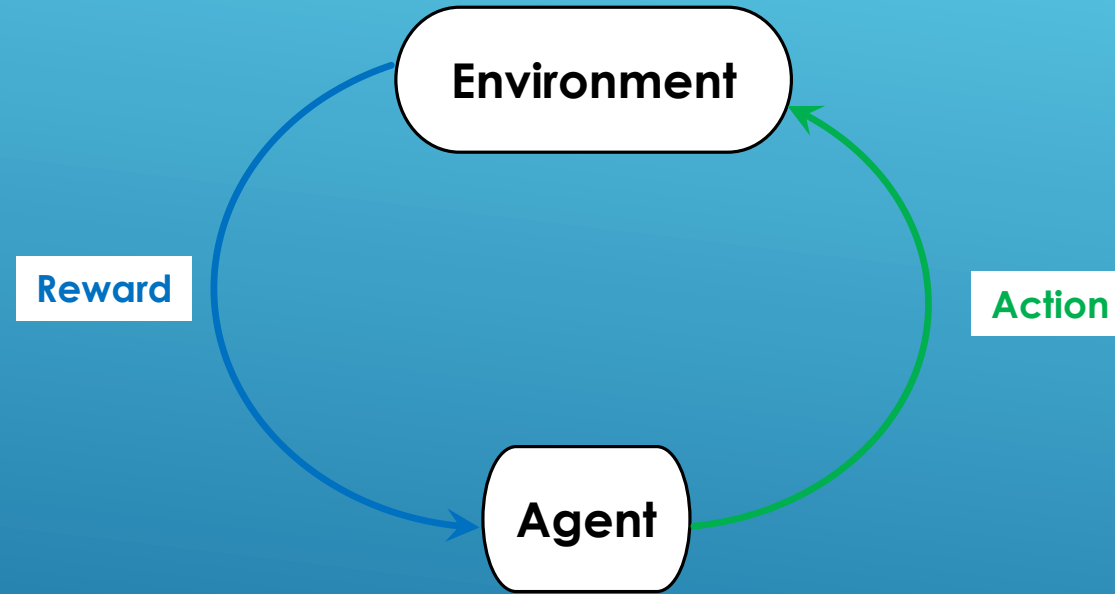
- **States:** only 1 state
- **Actions:** pull one of k arms
- **Reward Function:** receive reward based on a probability distribution associated with each arm.
- **Transition model:** no transitions

THE REINFORCEMENT LEARNING PROBLEM




$s_0, a_0, r_0, s_1, a_1, r_1, s_2 \dots, s_{n-1}, a_{n-1}, r_{n-1}, s_n$

THE MULTI-ARM BANDIT LEARNING PROBLEM




$a_0, r_0, a_1, r_1, a_2, r_2, \dots, a_n, r_n$


WHY STUDY MULTI-ARMED BANDITS?

- Multi-armed bandits can be used to create agents to solve real-world problems.
 - A simple model can make it easier to understand important issues that are hard to address using a more complex model.
 - Here, we will use multi-armed bandits to approach the exploration vs. exploitation trade-off in reinforcement learning.
- 
- A decorative graphic consisting of several parallel white lines of varying lengths, slanted diagonally from the bottom right towards the top right, located in the lower right quadrant of the slide.

MULTI-ARMED BANDIT APPLICATIONS

- Medical treatment in clinical trials
 - Online ad placement
 - Webpage personalization
 - Network packet routing
- 
- A series of white diagonal lines of varying lengths and thicknesses are positioned in the bottom right corner of the slide, creating a modern, abstract graphic element.


CHOOSING AN ACTION: EXPLORATION VS EXPLOITATION

- How should an agent choose an action? An obvious answer is simply to follow the current policy. However, this is often not the best way to improve your model.
 - **Exploit:** use your current model to maximize the expected utility now.
 - **Explore:** choose an action that will help you improve your model.
- 
- Three white lines of varying lengths and slopes are positioned in the bottom right corner of the slide, serving as a decorative element.

HOW/WHEN SHOULD WE EXPLORE / EXPLOIT ?

- **How to Exploit?** use the current policy.
- **How to Explore?**
 - choose an action randomly
 - be optimistic about the rewards you will receive.
 - choose an action you have used less often
- **When to exploit? When to explore? How much time in exploration vs time in exploitation?**

5 MULTI-ARMED BANDIT ALGORITHMS

1. Greedy
 2. ϵ -Greedy
 3. Greedy with optimistic initial values
 4. ϵ -Greedy with decreasing ϵ .
 5. Upper-confidence bound (UCB) action selection
- 
- A series of three parallel white diagonal lines in the bottom right corner of the slide, extending from the middle of the right edge towards the bottom left.

GREEDY METHOD

- At time t , estimate the expected value for each action:

$$Q_t(a) = \frac{\text{Sum of rewards when action } a \text{ is taken prior to } t}{\text{Number of times action } a \text{ is taken prior to } t}$$

- Select the action with the maximum value.

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

GREEDY METHOD WEAKNESSES

- Always exploits current knowledge, no exploration.
 - Can get stuck with a suboptimal action.
- 
- A series of three parallel white diagonal lines in the bottom right corner of the slide, extending from the bottom right towards the center.

E-GREEDY METHOD

- At time t , estimate the expected value for each action:

$$Q_t(a) = \frac{\text{Sum of rewards when action } a \text{ is taken prior to } t}{\text{Number of times action } a \text{ is taken prior to } t}$$

- With probability $1 - \epsilon$, select the action with the maximum value.

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

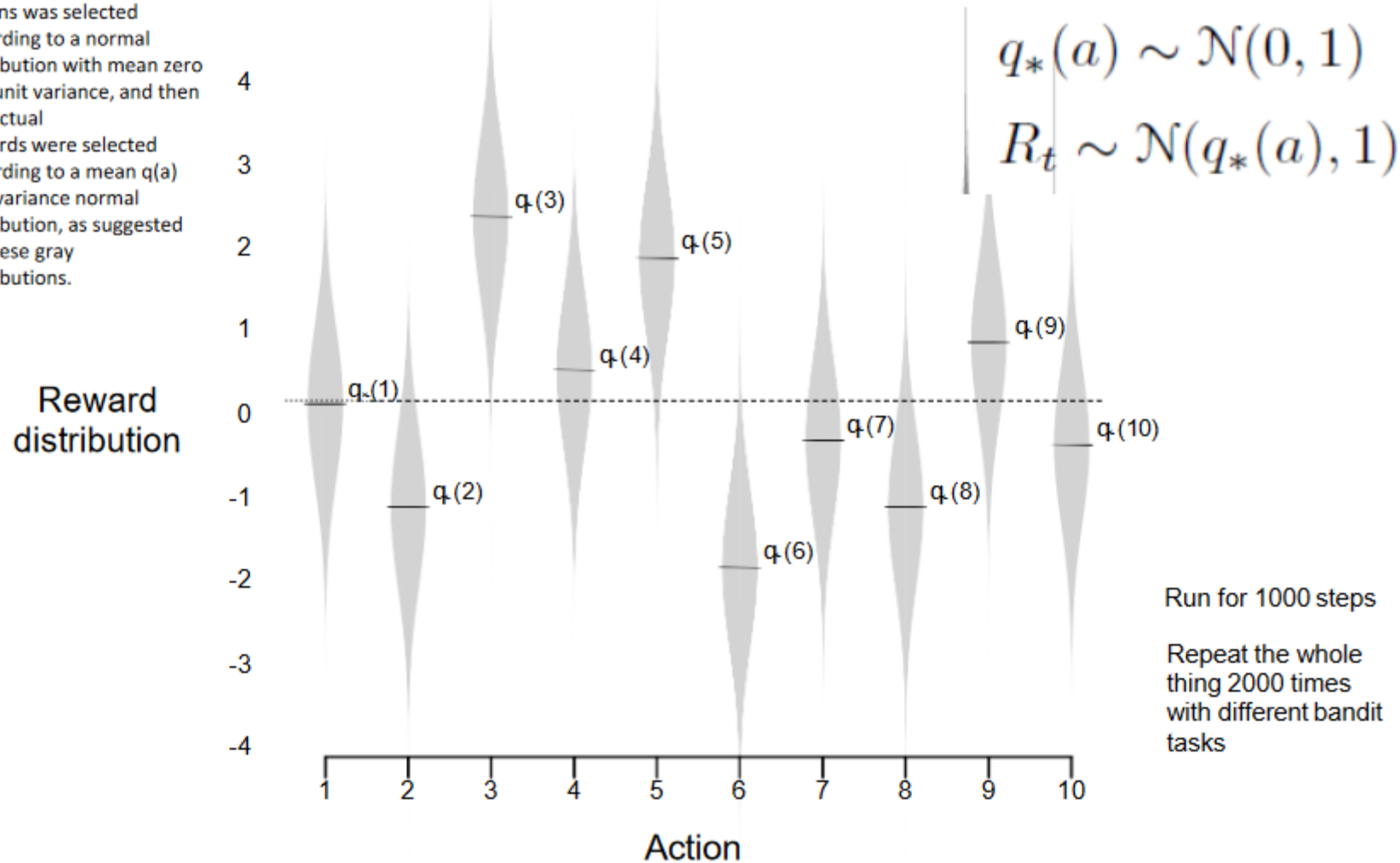
- With probability ϵ , randomly select an action from all the actions with equal probability.

EXPERIMENTS

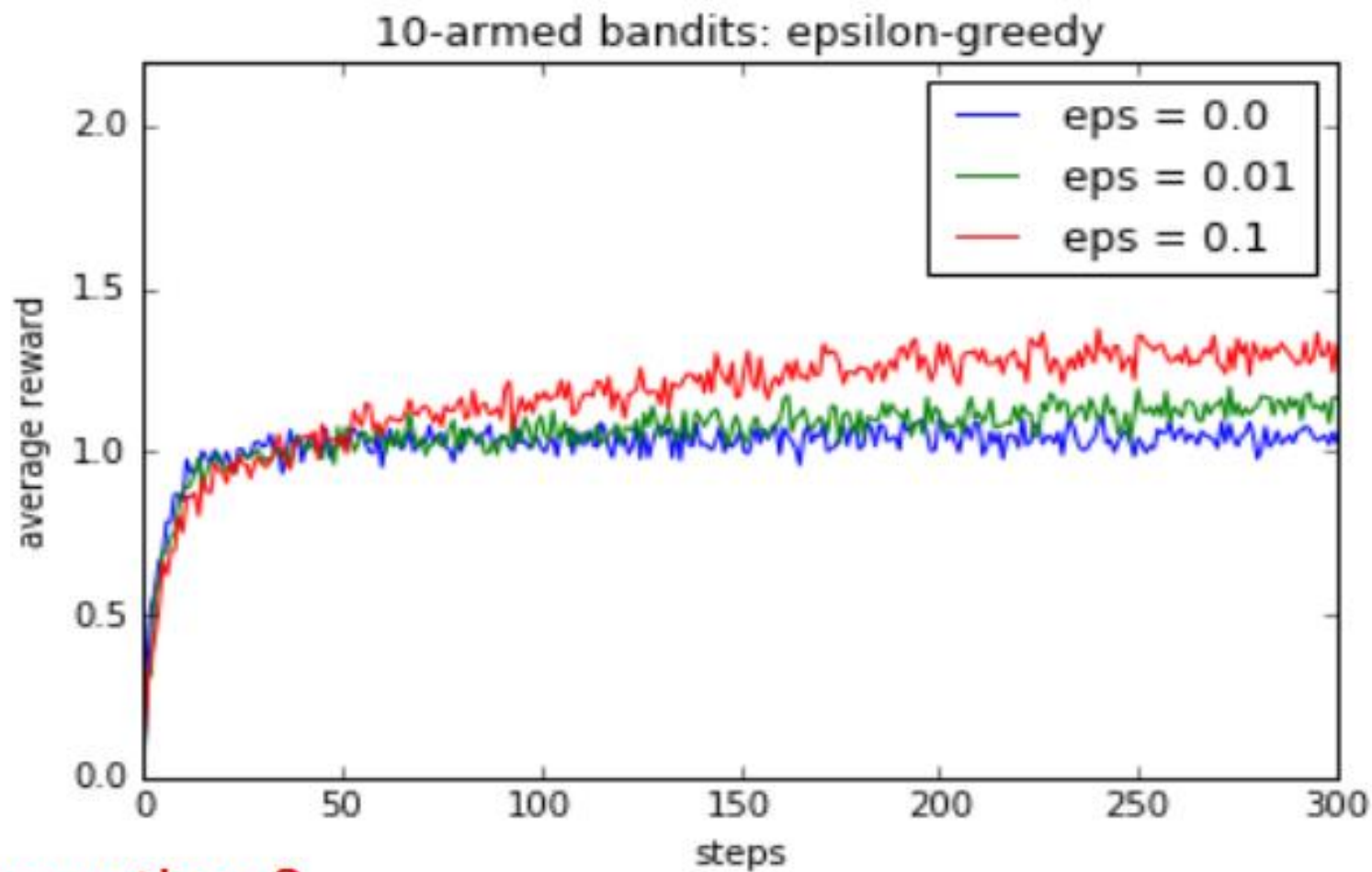
- Setup: (one run)
 - 10-armed bandits
 - Draw u_i from $Gaussian(0,1), i = 1, \dots, 10$
 - the expectation/mean of rewards for action i .
 - Rewards of action i at time t : $x_i(t)$
 - $x_i(t) \sim Gaussian(u_i, 1)$
 - Play 2000 rounds/steps
 - Average return at each time step
- Average over 1000 runs

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.

The 10-armed Testbed

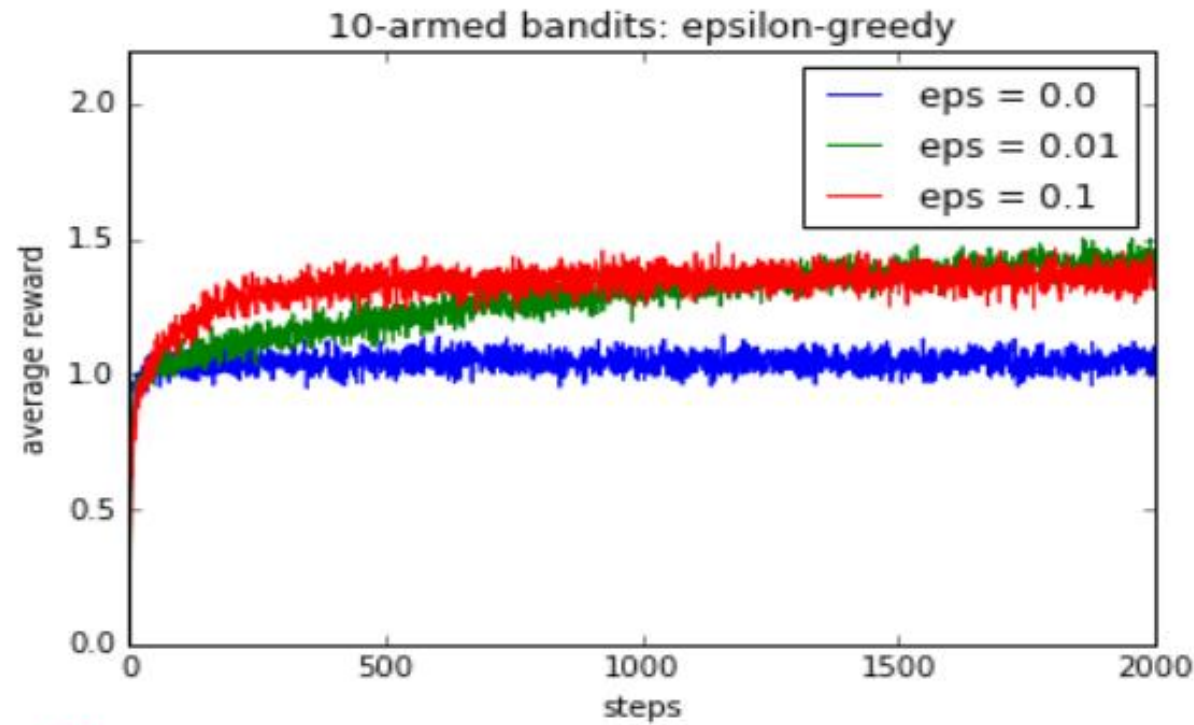


Experimental results: average over 1000 runs



Observations?

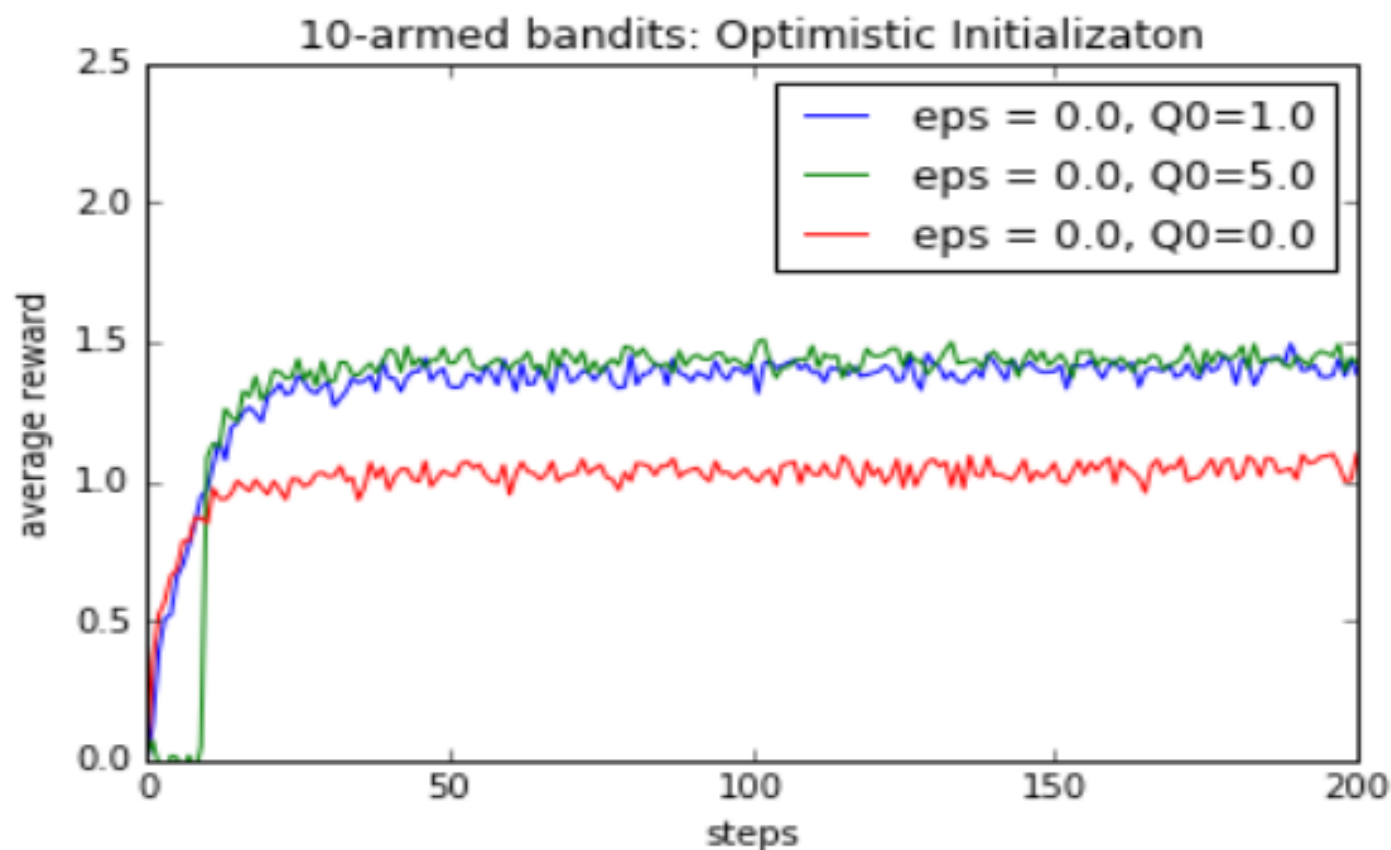
What will happen if we run more steps?



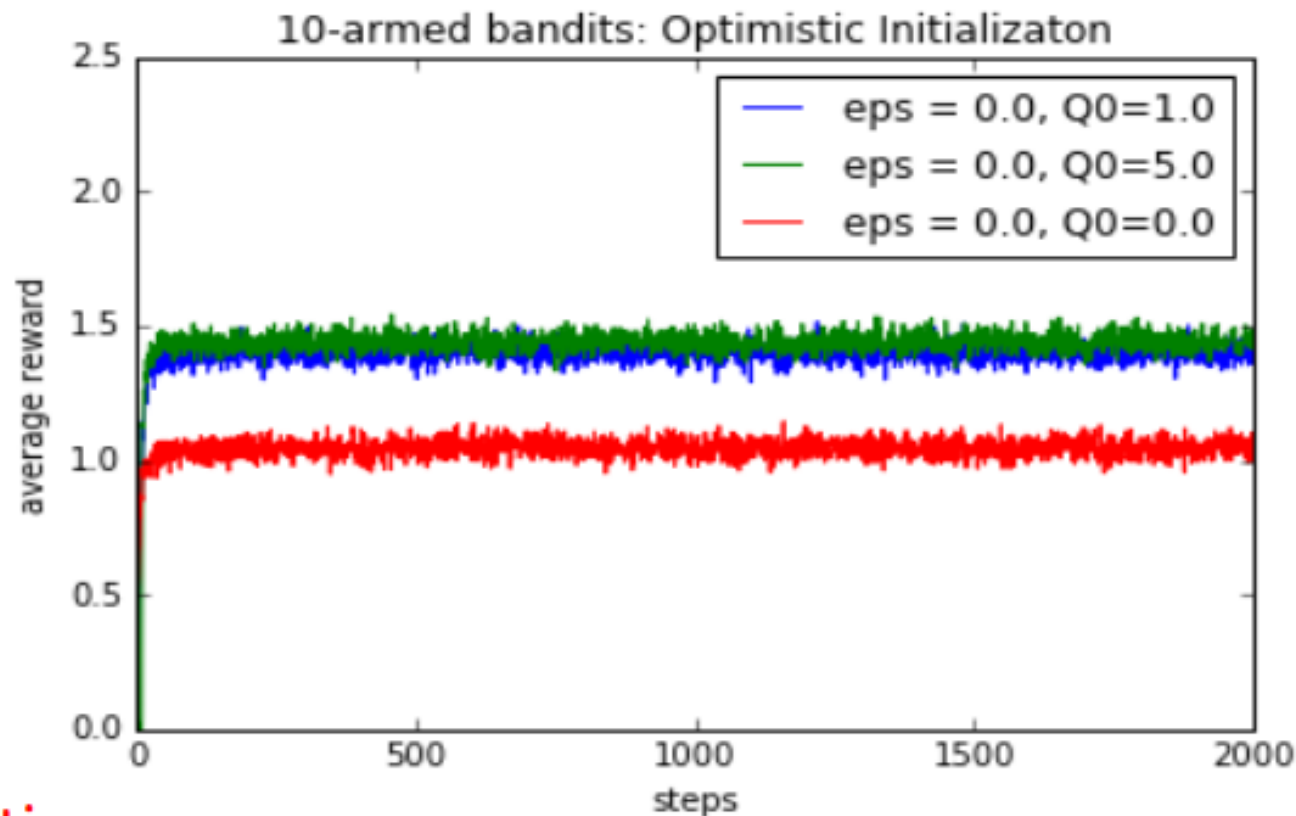
Observations:

- Greedy method improved faster at the very beginning, but level off at a lower level.
- ϵ - Greedy methods continue to Explore and eventually perform better.
- The $\epsilon = 0.01$ method improves slowly, but eventually performs better than the $\epsilon = 0.1$ method.

Improve the Greedy method with optimistic initialization



Greedy with optimistic initialization



Observations:

- Big initial Q values force the Greedy method to explore more in the beginning.
- No exploration afterwards.

Example: Clinical Trials

A reward of 1 if the treatment succeeds otherwise 0



Example: Clinical Trials

A reward of 1 if the treatment succeeds otherwise 0

$$Q_1(\text{P}) = 0.0$$

$$Q_1(\text{Y}) = 0.0$$

$$Q_1(\text{B}) = 0.0$$



Example: Clinical Trials

A reward of 1 if the treatment succeeds otherwise 0

$$Q_1(\text{P}) = 2.0$$

$$Q_1(\text{Y}) = 2.0$$

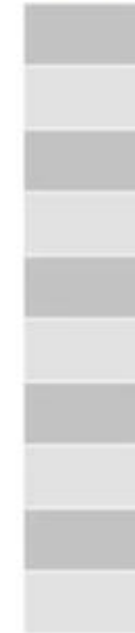
$$Q_1(\text{B}) = 2.0$$



A reward of 1 if the
treatment succeeds
otherwise 0

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

Let $\alpha = 0.5$



$$Q_1(\text{P}) = 2.0$$

$$Q_1(\text{Y}) = 2.0$$

$$Q_1(\text{B}) = 2.0$$

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

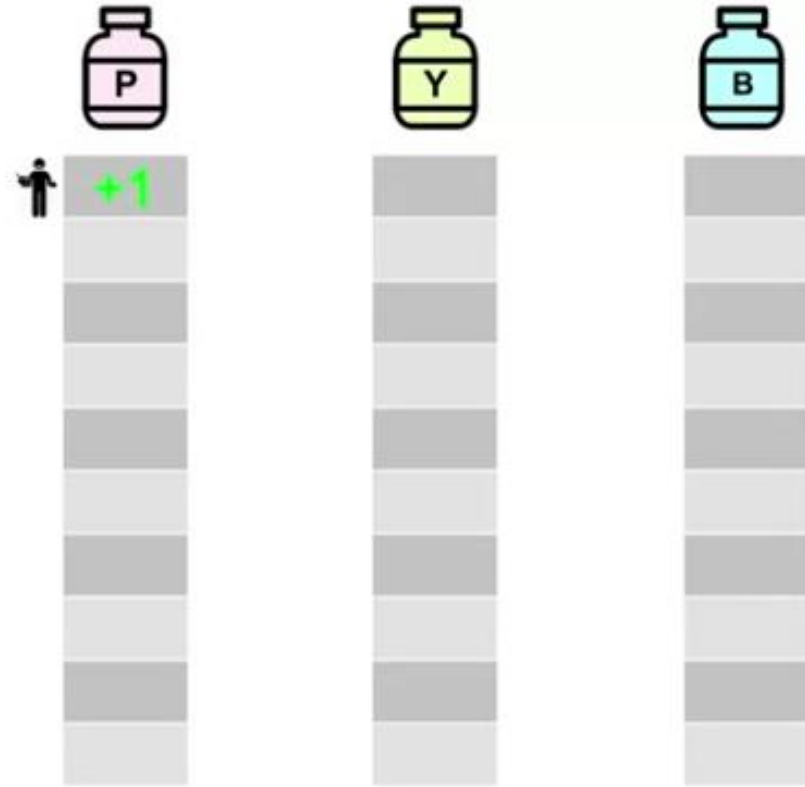
$$q_*(\text{Y}) = 0.75$$

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

A reward of 1 if the treatment succeeds otherwise 0

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

Let $\alpha = 0.5$



$$Q_2(\text{P}) = 1.5$$

$$Q_2(\text{Y}) = 2.0$$

$$Q_2(\text{B}) = 2.0$$

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

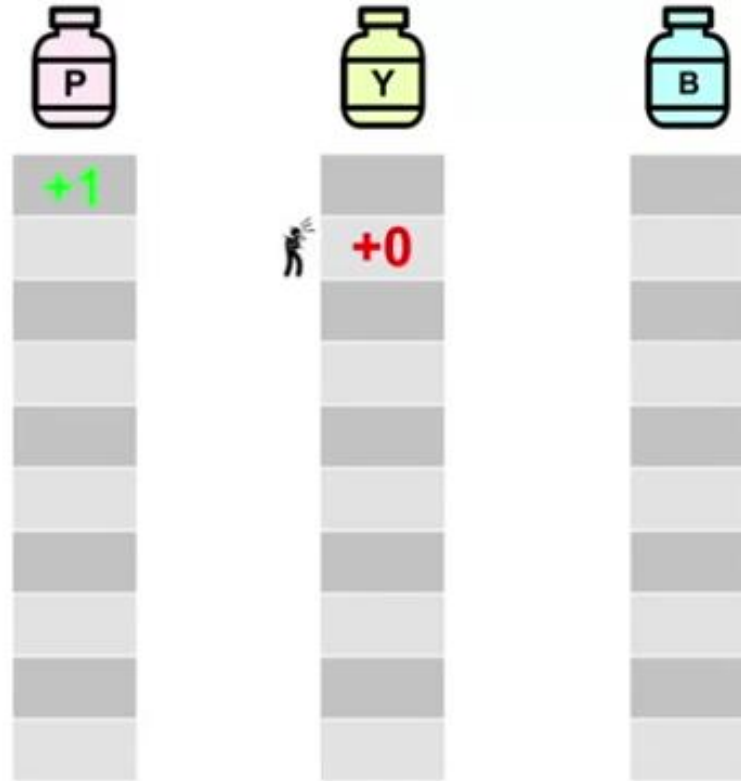
$$q_*(\text{Y}) = 0.75$$

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

A reward of 1 if the treatment succeeds
otherwise 0

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

Let $\alpha = 0.5$



$$Q_3(\text{P}) = 1.5$$

$$Q_3(\text{Y}) = 1.0$$

$$Q_3(\text{B}) = 2.0$$

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

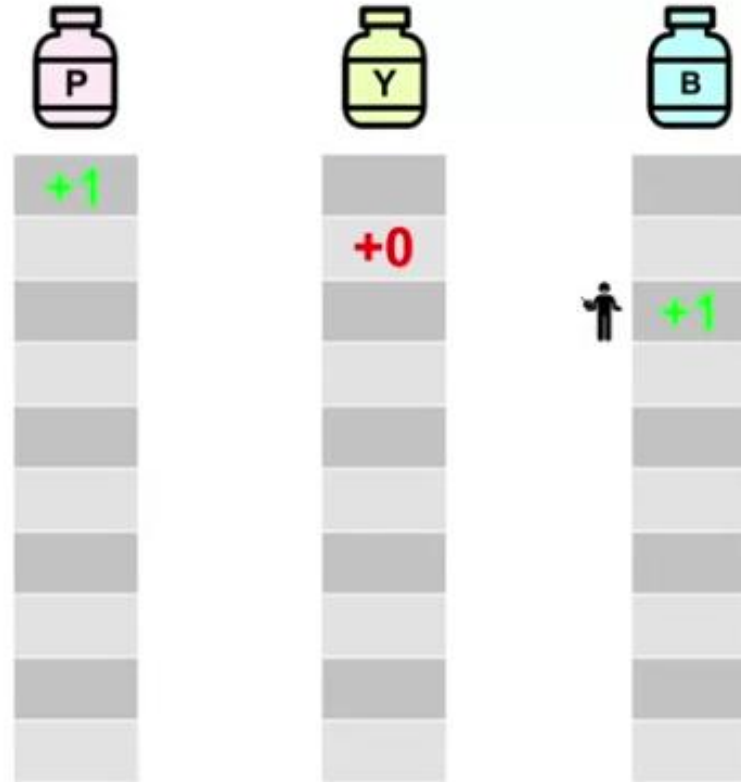
$$q_*(\text{Y}) = 0.75$$

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

A reward of 1 if the treatment succeeds
otherwise 0

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

Let $\alpha = 0.5$



$$Q_4(\text{P}) = 1.5$$

$$Q_4(\text{Y}) = 1.0$$

$$Q_4(\text{B}) = 1.5$$

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

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

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

A reward of 1 if the treatment succeeds otherwise 0

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

Let $\alpha = 0.5$



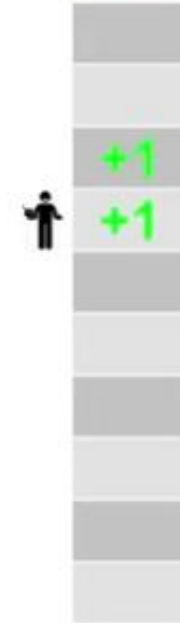
$$Q_s(\text{P}) = 1.5$$

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



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

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



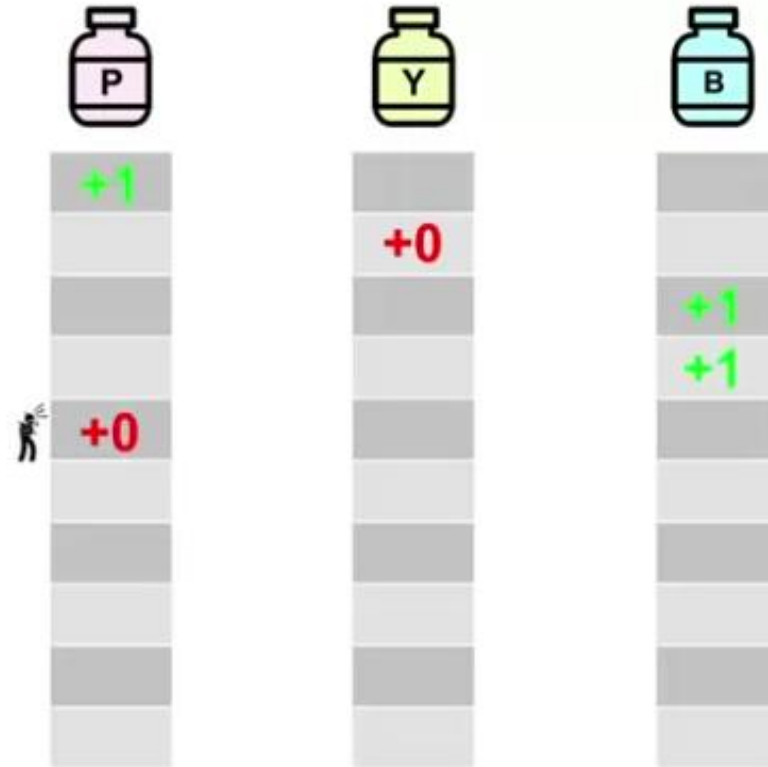
$$Q_s(\text{B}) = 1.25$$

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

A reward of 1 if the treatment succeeds otherwise 0

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

Let $\alpha = 0.5$



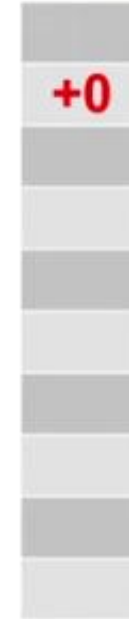
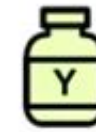
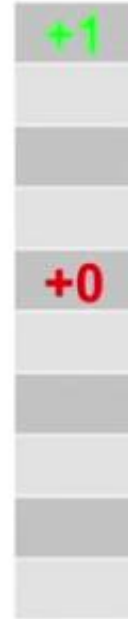
$$Q_6(\text{P}) = 0.75 \quad Q_6(\text{Y}) = 1.0 \quad Q_6(\text{B}) = 1.25$$

$$q_*(\text{P}) = 0.25 \quad q_*(\text{Y}) = 0.75 \quad q_*(\text{B}) = 0.5$$

A reward of 1 if the treatment succeeds otherwise 0

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

Let $\alpha = 0.5$



$$Q_7(\text{P}) = 0.75$$

$$Q_7(\text{Y}) = 1.0$$

$$Q_7(\text{B}) = 0.625$$

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

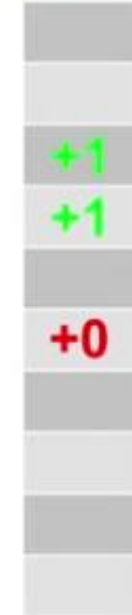
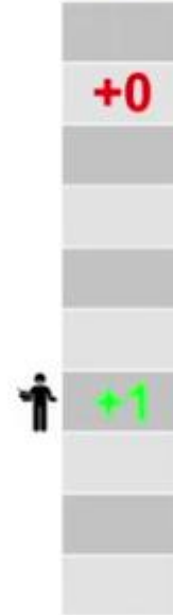
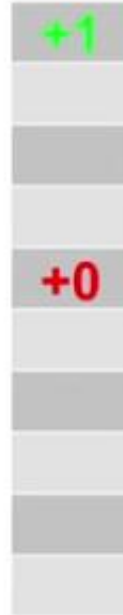
$$q_*(\text{Y}) = 0.75$$

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

A reward of 1 if the treatment succeeds otherwise 0

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

Let $\alpha = 0.5$



$$Q_8(\text{P}) = 0.75$$

$$Q_8(\text{Y}) = 1.0$$

$$Q_8(\text{B}) = 0.625$$

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

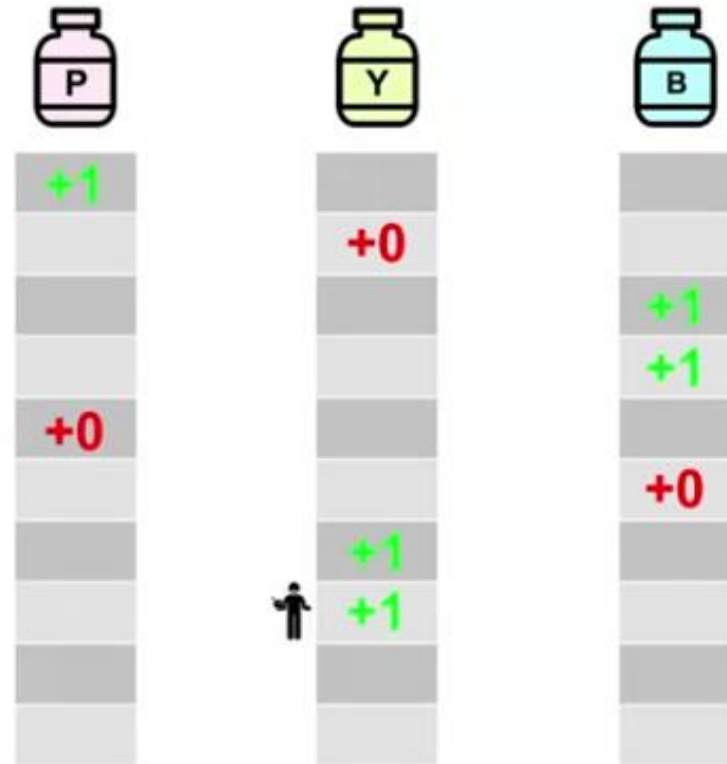
$$q_*(\text{Y}) = 0.75$$

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

A reward of 1 if the treatment succeeds otherwise 0

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

Let $\alpha = 0.5$



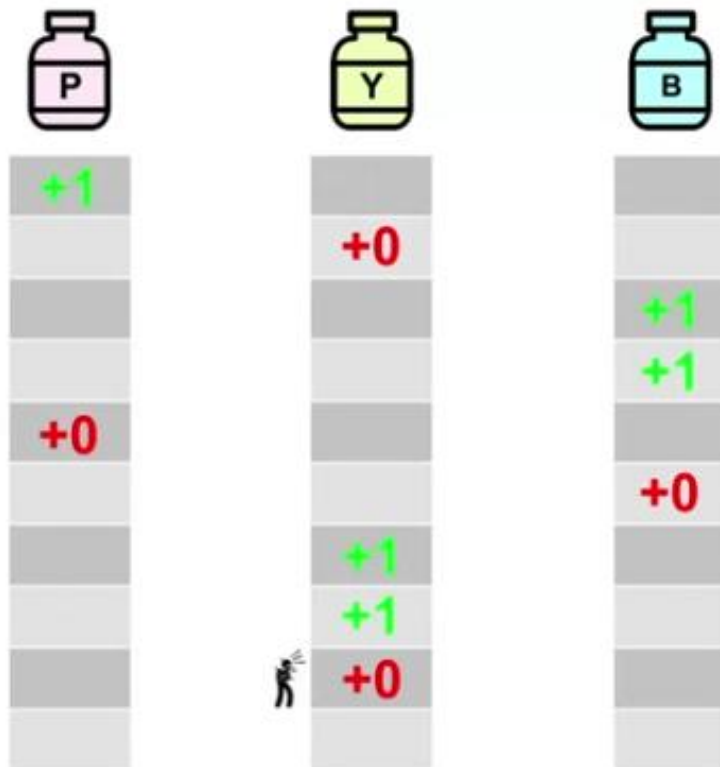
$$Q_y(\text{P}) = 0.75 \quad Q_y(\text{Y}) = 1.0 \quad Q_y(\text{B}) = 0.625$$

$$q_*(\text{P}) = 0.25 \quad q_*(\text{Y}) = 0.75 \quad q_*(\text{B}) = 0.5$$

A reward of 1 if the treatment succeeds otherwise 0

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

Let $\alpha = 0.5$

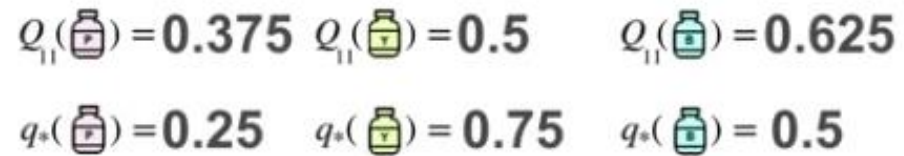


$$Q_{10}(\text{P}) = 0.75 \quad Q_{10}(\text{Y}) = 0.5 \quad Q_{10}(\text{B}) = 0.625$$

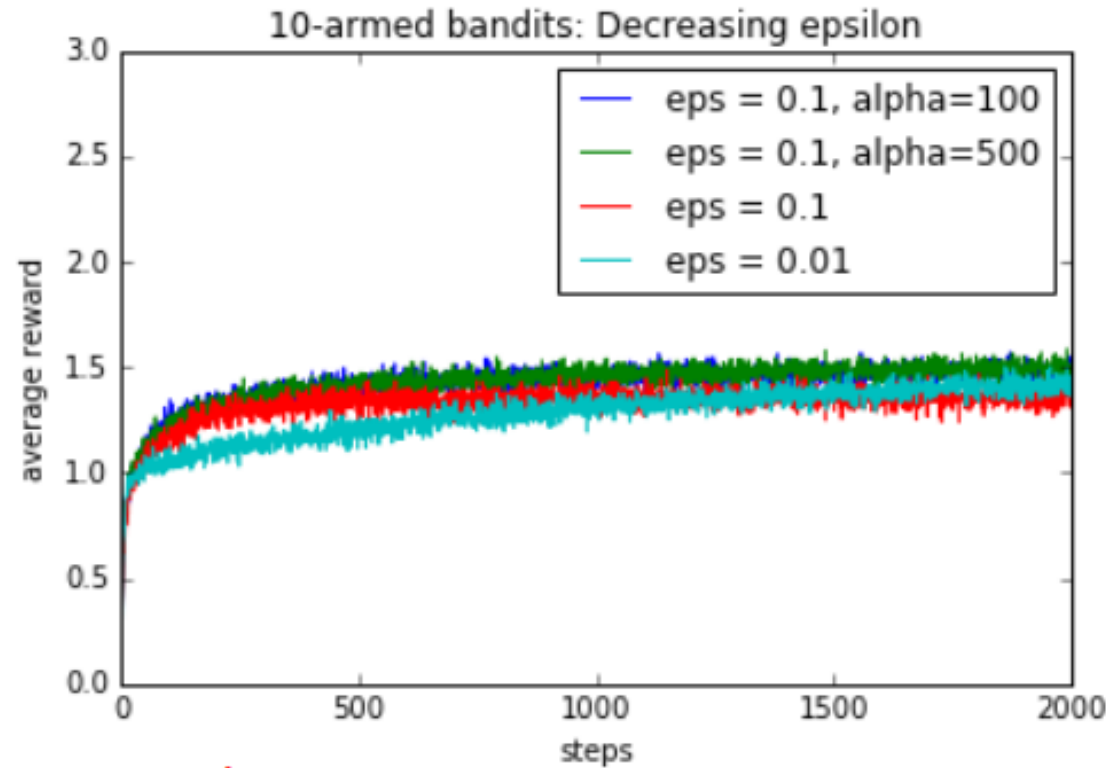
$$q^*(\text{P}) = 0.25 \quad q^*(\text{Y}) = 0.75 \quad q^*(\text{B}) = 0.5$$

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

Let $\alpha = 0.5$



Improve ϵ -greedy with decreasing ϵ over time



Decreasing over time:

- $\epsilon_t = \epsilon_t * (\alpha)/(t + \alpha)$
- Improves faster in the beginning, also outperforms fixed ϵ -greedy methods in the long run.

E-GREEDY METHOD WEAKNESSES

- Randomly selects an action to explore, does not explore more “promising” actions.
- Does not consider prior experience with an action. If an action has been taken many times, there is less need to explore it.

UPPER CONFIDENCE BOUND (UCB) ALGORITHM

- Take each action once.
- At any time $t > K$,

$$- A_t = \operatorname{argmax}_a (Q_t(a) + c \sqrt{\frac{\ln t}{N_a(t)}}), \text{ where}$$

$Q_t(a)$ is the average reward obtained from action a .

$N_a(t)$ is the number of times action a has been taken.

c is a user-specified parameter that controls the amount exploration that takes place.

UPPER CONFIDENCE BOUND (UCB) ALGORITHM

$$A_t = \operatorname{argmax}_a (Q_t(a) + c \sqrt{\frac{\ln t}{N_a(t)}}),$$

Exploitation



Exploration



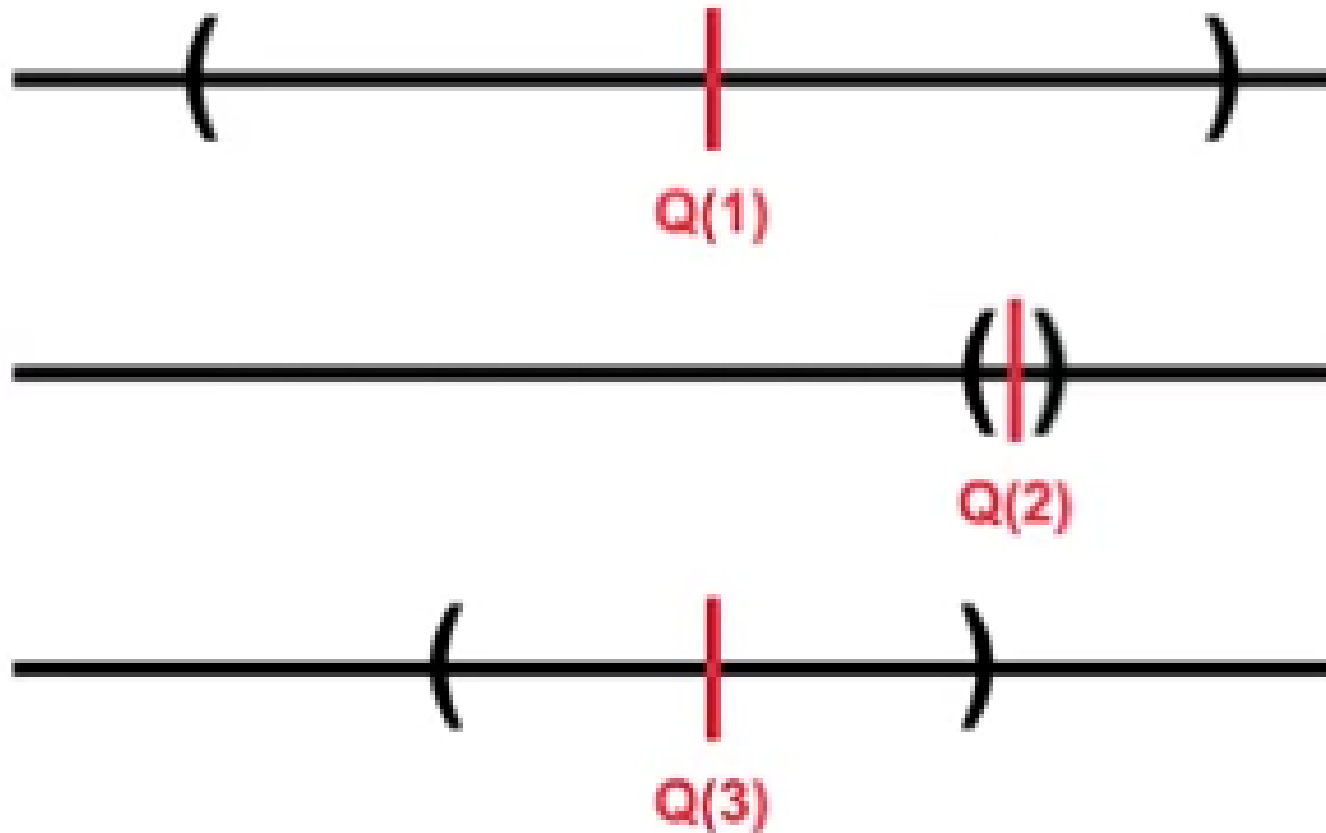
Upper-Confidence Bound (UCB) Action Selection

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

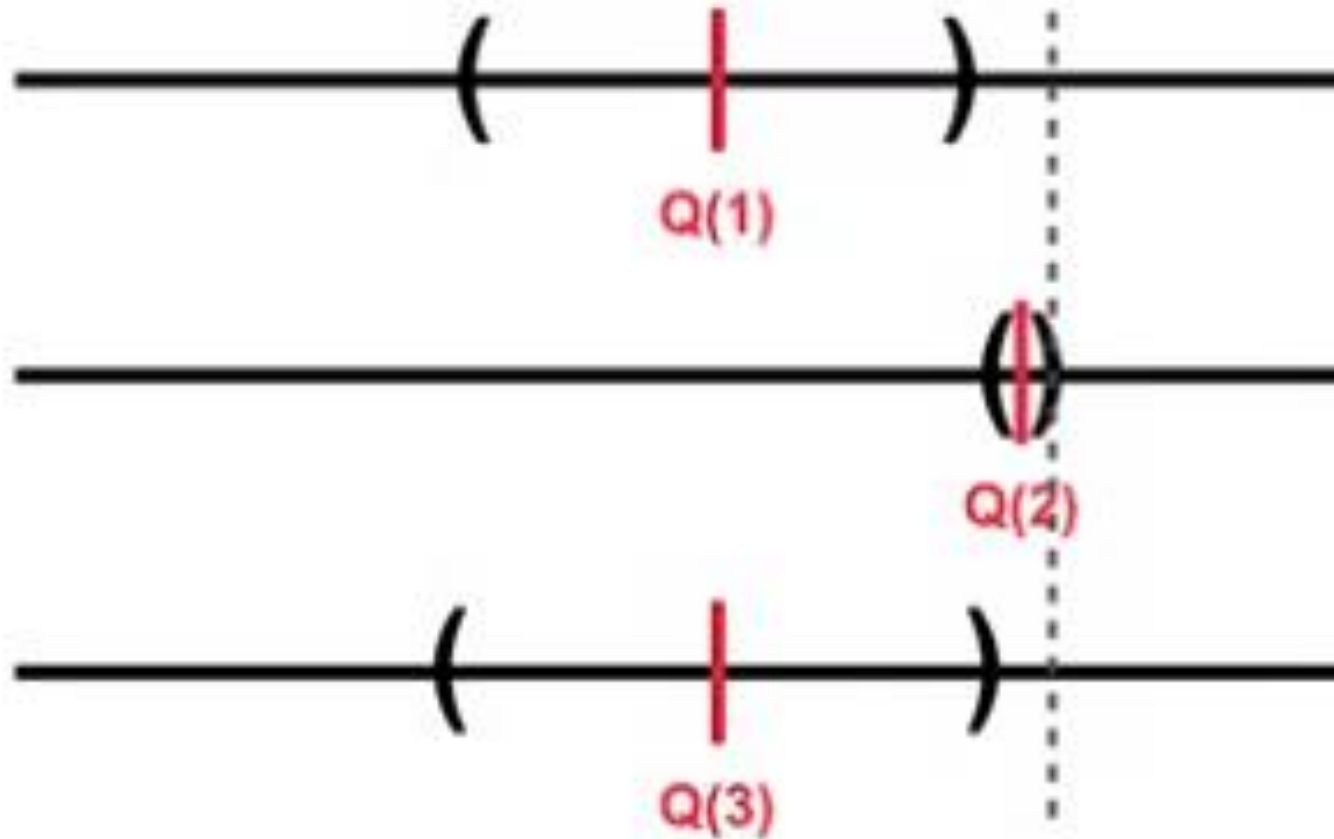
UPPER CONFIDENCE BOUND (UCB) ALGORITHM

- $A_t = \operatorname{argmax}_a (Q_t(a) + c \sqrt{\frac{\ln t}{N_a(t)}})$
- $c \sqrt{\frac{\ln t}{N_a(t)}}$ is the confidence interval for the average reward.
- Small $N_a(t)$ indicates a large confidence interval i.e., $Q_t(a)$ is more uncertain.
- Large $N_a(t)$ indicates a small confidence interval i.e., $Q_t(a)$ is more accurate.

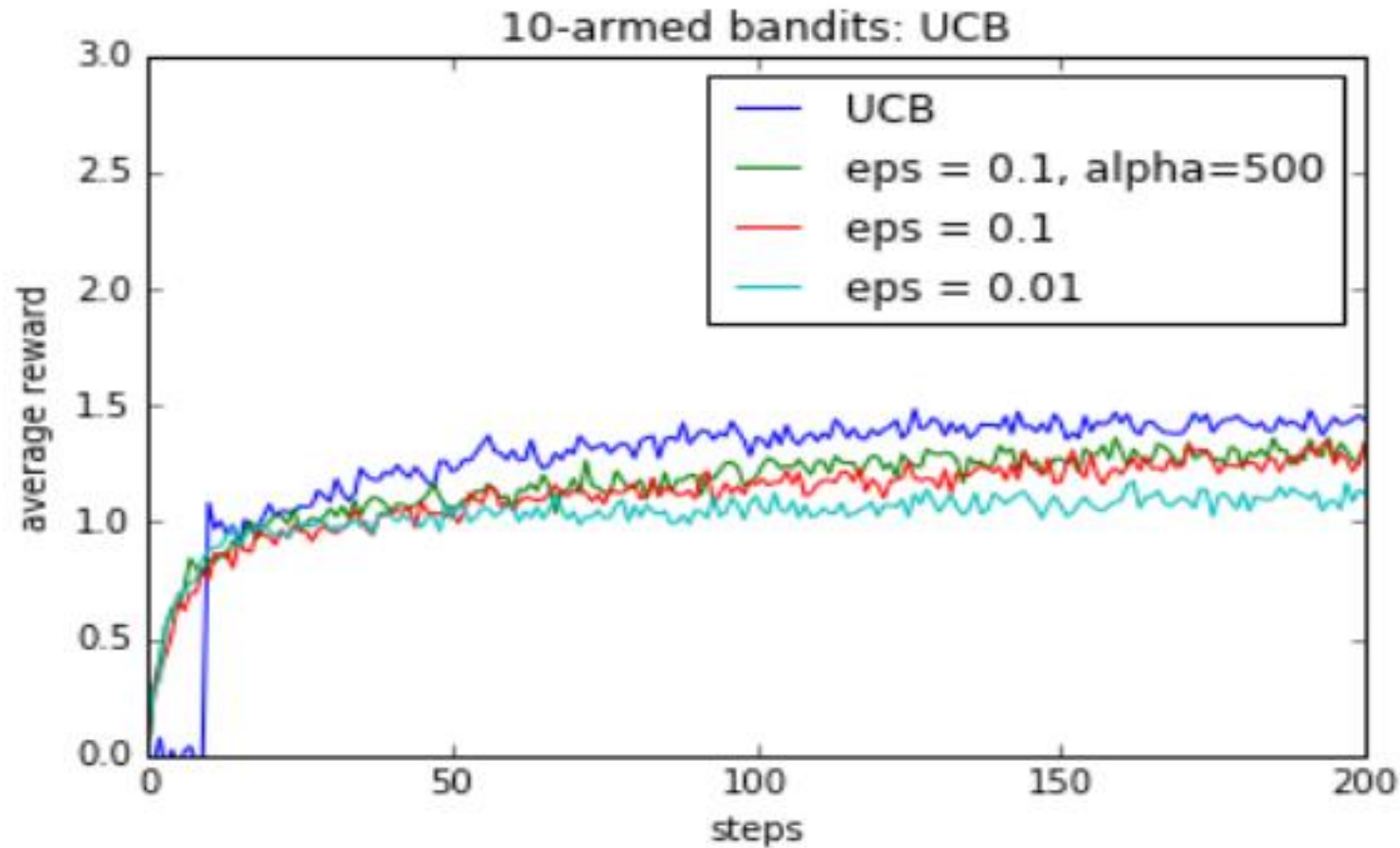
Optimism in the Face of Uncertainty



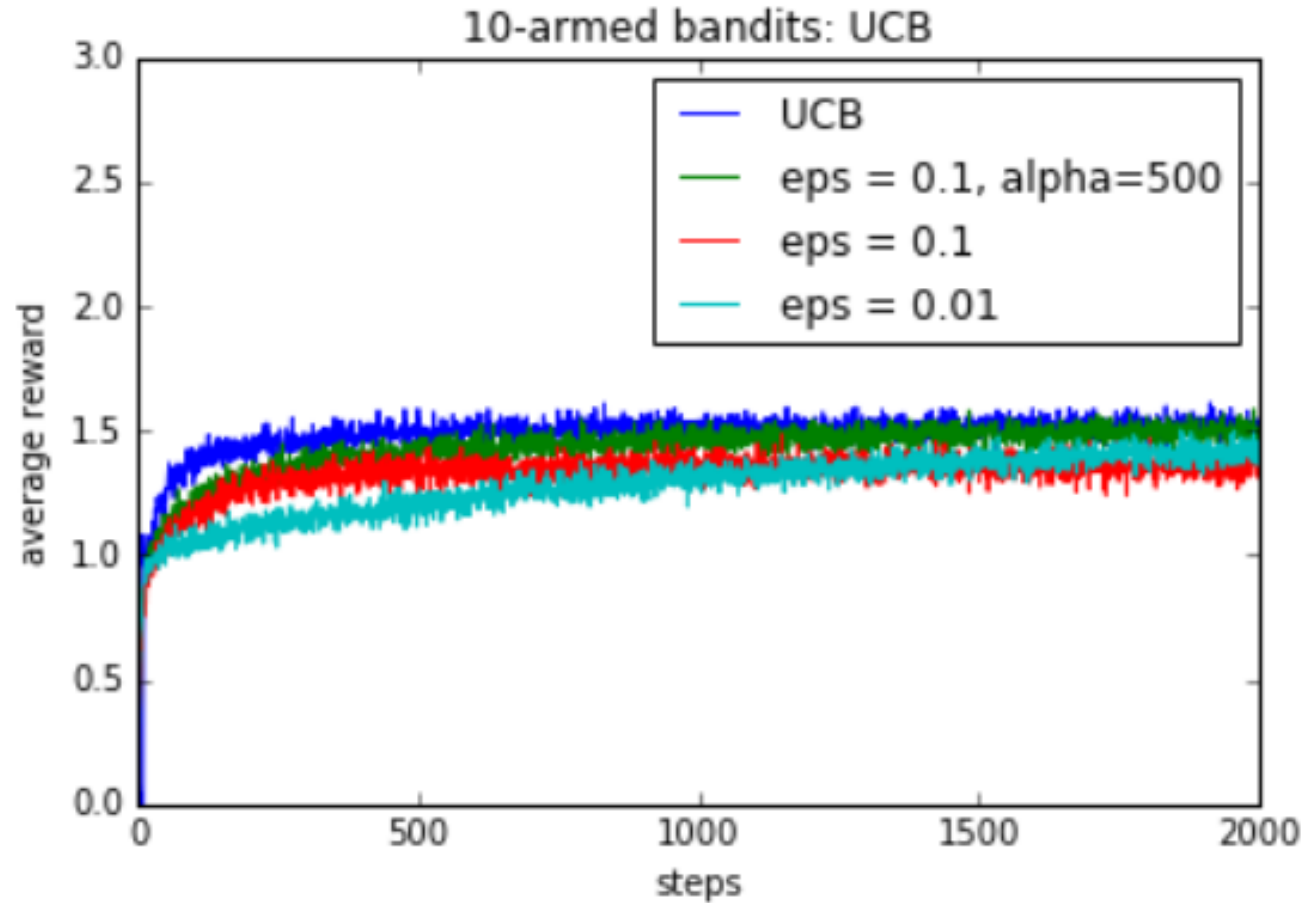
Optimism in the Face of Uncertainty



How good is UCB? Experimental results



How good is UCB? Experimental results



Observations: After initial 10 steps, improves quickly and outperforms ϵ -Greedy methods.

