

Carnegie Mellon

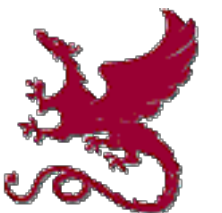
School of Computer Science

Deep Reinforcement Learning and Control

# Multi-armed bandits

Spring 2021, CMU 10-403

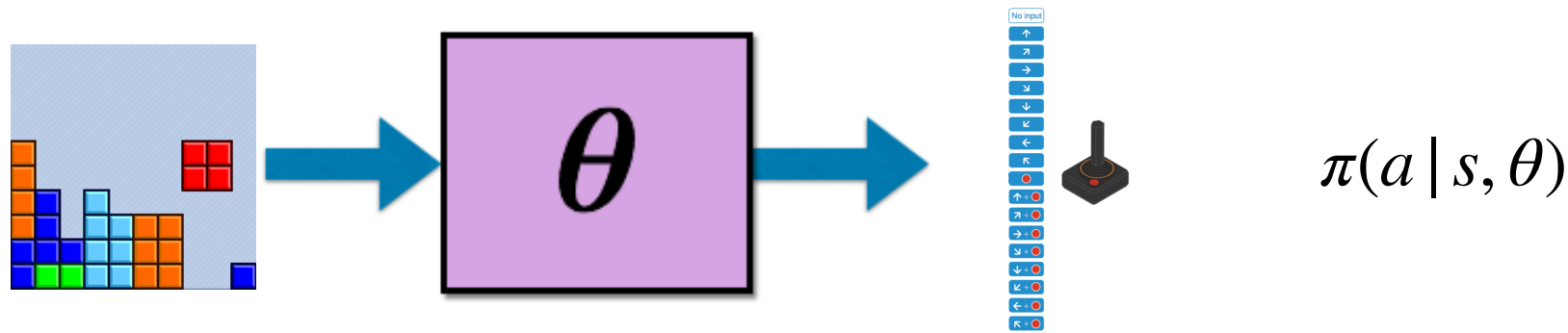
Katerina Fragkiadaki



# Used Materials

- **Disclaimer:** Some material and slides for this lecture were borrowed from Rich Sutton's lecture on multi-armed bandits.

# LL: Reinforcement Learning via Evolution



Given an initial state distribution  $\mu_0(s_0)$ , estimate parameters  $\theta$  of a policy  $\pi_\theta$  so that, the trajectories  $\tau$  sampled from this policy have maximum returns, i.e., sum of rewards  $R(\tau)$ .

$$\max_{\theta} . U(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} [R(\tau) | \pi_\theta, \mu_0(s_0)]$$

$\tau$  : trajectory, a sequence of state, action, rewards, a game fragment or a full game:

$$\tau : s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2, \dots, s_T, a_T, r_T$$

$R(\tau)$  : reward of a trajectory: (discounted) sum of the rewards of the individual state/actions

$$R(\tau) = \sum_{t=1}^T r_t$$

# LL: Evolutionary methods for policy search

$$\max_{\theta} . \quad U(\theta) = \mathbb{E}_{\tau \sim \pi_{\theta}} [R(\tau) \mid \pi_{\theta}, \mu_0(s_0)]$$

*General algorithm:*

*Initialize a population of parameter vectors (genotypes)*

- 1. Make random perturbations (mutations) to each parameter vector*
- 2. Evaluate the perturbed parameter vector (fitness)*
- 3. Keep the perturbed vector if the result improves (selection)*
- 4. GOTO 1*

*Simple and biologically plausible...*



# LL: Natural Evolutionary Strategies (NES)

- In CEM and CMA-ES, we have been selecting the best (elite) parameter offsprings.
- NES considers **every** offspring.

## Algorithm 1: Evolutionary Strategies

1. **Input:** Learning rate  $\alpha$ , noise standard deviation  $\sigma$ , initial policy parameters  $\theta_0$
2. **for**  $t = 0, 1, 2, \dots$  **do**
3.   Sample  $\epsilon_1, \dots, \epsilon_n \sim \mathcal{N}(0, \mathbf{I}_d)$
4.   Compute returns  $F_i = F(\mu_t + \sigma\epsilon_i)$  for  $i = 1, 2, \dots, n$
5.   Set  $\mu_{t+1} \leftarrow \mu_t + \alpha \frac{1}{n\sigma} \sum_{i=1}^n F_i \epsilon_i$
6. **end for**

# This lecture - Motivation

Learning to act in a non-sequential (single action) setups:

- Each action results in an **immediate** reward.
- We want to choose actions that maximize our immediate reward in expectation.
  - Q: Why in expectation?
  - A: Because rewards are not deterministic.
- For example, displaying an advertisement can generate different click rates in different days. Actions: the advertisements to be displayed, Rewards: the user click rate. We want to pick the advertisement that maximizes the click rate on average

# Multi-Armed Bandits

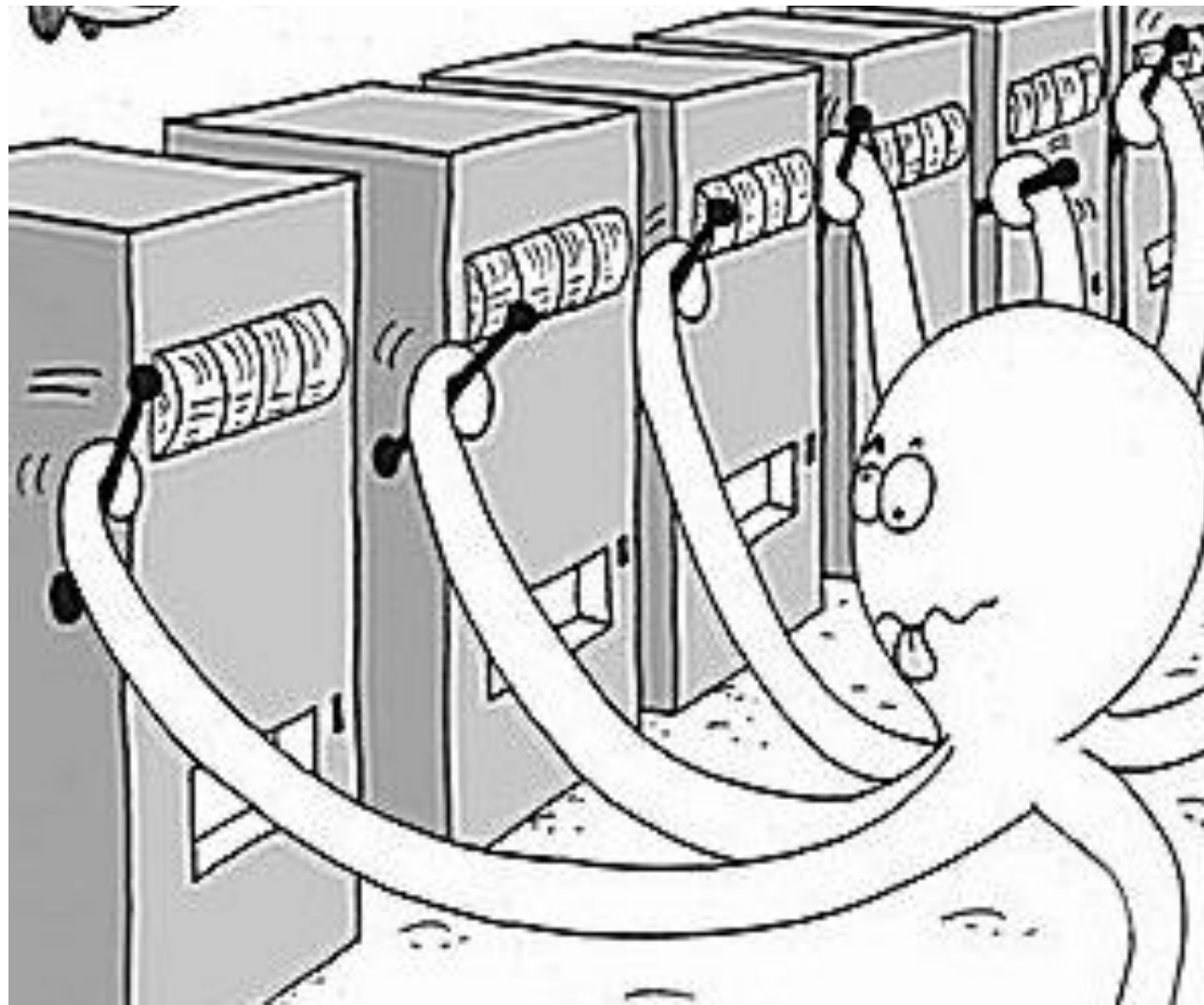
One-armed bandit= Slot machine (English slang)



source: [infoslotmachine.com](http://infoslotmachine.com)

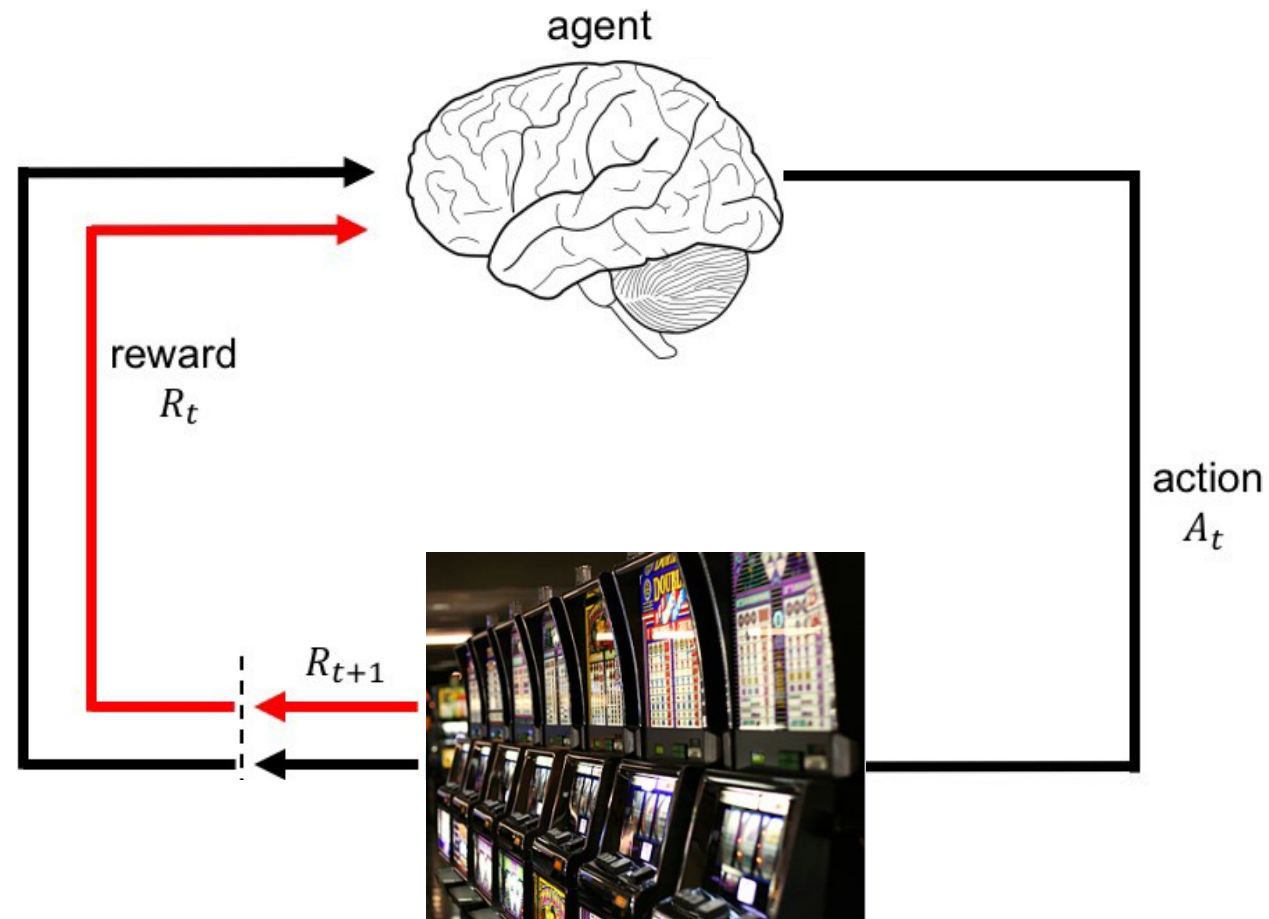
# Multi-Armed Bandits

Multi-Armed bandit = Multiple Slot Machine



source: Microsoft Research

# Multi-Armed Bandits



$$A_t, R_{t+1}, A_{t+1}, R_{t+2}, A_{t+2}, A_{t+3}, R_{t+3}, \dots$$

The state does not change! (a.k.a. stateless)

# Multi-Armed Bandit Problem

At each timestep  $t$  the agent chooses one of the  $K$  arms and plays it.

The  $k$  th arm produces reward  $r_{k,t}$  when played at timestep  $t$ .

The rewards  $r_{k,t}$  are drawn from a probability distribution  $\mathcal{P}_k$  with mean  $\mu_k$ .

The agent does not know neither the full arm reward distributions neither their means.



source: Pandey et al.'s slide

**Agent's Objective:** maximize cumulative rewards (over a finite or infinite horizon).

I can maximize cumulative rewards over a finite or infinite horizon if i just play **the arm with the highest mean reward  $\mu_k$**  each time. (but i do not know those..)



# Multi-Armed Bandit Problem

At each timestep  $t$  the agent chooses one of the  $K$  arms and plays it.

The  $k$  th arm produces reward  $r_{k,t}$  when played at timestep  $t$ .

The rewards  $r_{k,t}$  are drawn from a probability distribution  $\mathcal{P}_k$  with mean  $\mu_k$ .

The agent does not know neither the full arm reward distributions neither their means.



source: Pandey et al.'s slide

Definition: The **action-value** for action  $a$  (here arm  $k$ ) is **its mean reward**:

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

# The Exploration/Exploitation Dilemma

- Suppose you form estimates

$$Q_t(a) \approx q_*(a), \quad \forall a \quad \textit{action-value estimates}$$



# The Exploration/Exploitation Dilemma

- Suppose you form estimates

$$Q_t(a) \approx q_*(a), \quad \forall a \quad \text{action-value estimates}$$

- Define the *greedy action* at time  $t$  as

$$A_t^* \doteq \arg \max_a Q_t(a)$$

# The Exploration/Exploitation Dilemma

- Suppose you form estimates

$$Q_t(a) \approx q_*(a), \quad \forall a \quad \text{action-value estimates}$$

- Define the *greedy action* at time  $t$  as

$$A_t^* \doteq \arg \max_a Q_t(a)$$

- If  $A_t = A_t^*$  then you are *exploiting*  
If  $A_t \neq A_t^*$  then you are *exploring*

# The Exploration/Exploitation Dilemma

- Suppose you form estimates

$$Q_t(a) \approx q_*(a), \quad \forall a \quad \text{action-value estimates}$$

- Define the *greedy action* at time  $t$  as

$$A_t^* \doteq \arg \max_a Q_t(a)$$

- If  $A_t = A_t^*$  then you are *exploiting*  
If  $A_t \neq A_t^*$  then you are *exploring*
- You can't do both, but you need to do both

# The Exploration/Exploitation Dilemma

- Suppose you form estimates

$$Q_t(a) \approx q_*(a), \quad \forall a \quad \text{action-value estimates}$$

- Define the *greedy action* at time  $t$  as

$$A_t^* \doteq \arg \max_a Q_t(a)$$

- If  $A_t = A_t^*$  then you are *exploiting*  
If  $A_t \neq A_t^*$  then you are *exploring*
- You can't do both, but you need to do both
- You can never stop exploring, but maybe you should explore less with time.

# Exploration vs Exploitation Dilemma

- Online decision-making involves a fundamental choice:
  - **Exploitation**: Make the best decision given current information
  - **Exploration**: Gather more information
- The best long-term strategy may involve **short-term sacrifices**
- Gather enough information to make the best overall decisions

# Exploration vs Exploitation Dilemma

- Online decision-making involves a fundamental choice:
  - **Exploitation**: Make the best decision given current information
  - **Exploration**: Gather more information
- The best long-term strategy may involve **short-term sacrifices**
- Gather enough information to make the best overall decisions
- The exploration/exploitation dilemma is not a problem encountered in computational RL or deep RL: It is a fundamental problem in decision making of any intelligent agent.

# Exploration vs. Exploitation Dilemma

- Restaurant Selection
  - **Exploitation:** Go to your favorite restaurant
  - **Exploration:** Try a new restaurant
- Oil Drilling
  - **Exploitation:** Drill at the best known location
  - **Exploration:** Drill at a new location
- Game Playing
  - **Exploitation:** Play the move you believe is best
  - **Exploration:** Play an experimental move

# Example: Bernoulli Bandits

Recall: The **Bernoulli distribution** is the discrete probability distribution of a random variable which takes the value 1 with probability  $p$  and the value 0 with probability  $q=1-p$ , that is, the probability distribution of any single experiment that asks a yes-no question.

- Each action (arm when played) results in success or failure, **rewards are binary**.
- Mean reward for each arm represents the probability of success.
- Action (arm)  $k \in \{1 \dots K\}$  produces a success with probability  $\theta_k \in [0,1]$ .



$\theta_1$

win 0.6  
of time



$\theta_2$

win 0.4  
of time



$\theta_3$

win 0.45  
of time



# Real world motivation: content presentation

We have two variations of content of a webpage, A and B, and we want to decide which one to display to engage more users.

- Two arm bandits: each arm corresponds to a content variation shown to users (not necessarily the same user).
- Reward: 1 if the user clicks, 0 otherwise.
- Mean reward (success probability) for each invitation: the click-through-rate, the percentage of users that would click on it

OBAMA BIDEN

**DINNER WITH BARACK**  
Your chance to meet the President

[GET STARTED](#)

**DINNER WITH BARACK**

YOU'RE INVITED.  
WE'LL COVER YOUR AIRFARE.

No purchase, payment, or contribution necessary to enter or win. Contributing will not improve chances of winning. Void where prohibited. Entries must be received by September 20, 2012. You may enter by contributing to Obama Victory Fund 2012 here or click here to enter without contributing. Three winners will each receive the following prize package: round-trip tickets for winner from within the fifty U.S. States, DC, or Puerto Rico to a destination to be determined by the Sponsor; hotel accommodations; and dinner with President Obama on a date to be determined by the Sponsor (approximate retail value of all prizes \$4,800). Odds of winning depend on number of entries received. Promotion open only to U.S. citizens, or lawful permanent U.S. residents who are legal residents of 50 United States, District of Columbia and Puerto Rico and 18 or older (or age of majority under applicable law). Promotion subject to Official Rules Official rules and additional restrictions on eligibility. Sponsor: Obama for America, 130 E. Randolph St., Chicago, IL 60601.

OBAMA BIDEN

[Privacy Policy](#) [Terms of Service](#)

Contributions or gifts to Obama Victory Fund 2012 are not tax deductible.

PAID FOR BY OBAMA VICTORY FUND 2012, A JOINT FUNDRAISING COMMITTEE AUTHORIZED BY OBAMA FOR AMERICA, THE DEMOCRATIC NATIONAL COMMITTEE, AND THE STATE DEMOCRATIC PARTIES IN THE FOLLOWING STATES: CO, FL, IA, IN, NH, NC, OH, PA, VA, AND WI.

© 2011–2012 Obama for America. All Rights Reserved.

OBAMA BIDEN

**DINNER WITH BARACK**  
Your chance to meet the President

[GET STARTED](#)

**DINNER WITH BARACK**

You're invited.  
We'll cover your airfare.

No purchase, payment, or contribution necessary to enter or win. Contributing will not improve chances of winning. Void where prohibited. Entries must be received by September 20, 2012. You may enter by contributing to Obama Victory Fund 2012 here or click here to enter without contributing. Three winners will each receive the following prize package: round-trip tickets for winner from within the fifty U.S. States, DC, or Puerto Rico to a destination to be determined by the Sponsor; hotel accommodations; and dinner with President Obama on a date to be determined by the Sponsor (approximate retail value of all prizes \$4,800). Odds of winning depend on number of entries received. Promotion open only to U.S. citizens, or lawful permanent U.S. residents who are legal residents of 50 United States, District of Columbia and Puerto Rico and 18 or older (or age of majority under applicable law). Promotion subject to Official Rules Official rules and additional restrictions on eligibility. Sponsor: Obama for America, 130 E. Randolph St., Chicago, IL 60601.

OBAMA BIDEN

[Privacy Policy](#) [Terms of Service](#)

Contributions or gifts to Obama Victory Fund 2012 are not tax deductible.

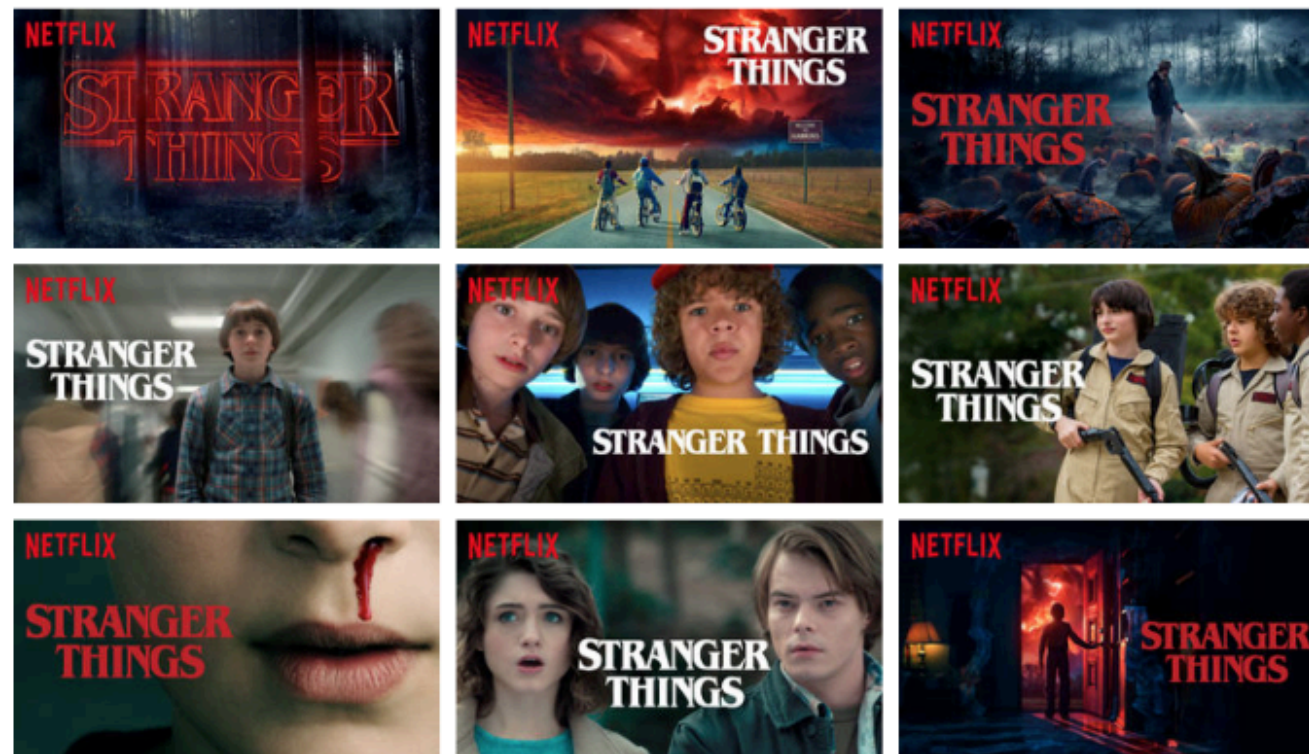
PAID FOR BY OBAMA VICTORY FUND 2012, A JOINT FUNDRAISING COMMITTEE AUTHORIZED BY OBAMA FOR AMERICA, THE DEMOCRATIC NATIONAL COMMITTEE, AND THE STATE DEMOCRATIC PARTIES IN THE FOLLOWING STATES: CO, FL, IA, IN, NH, NC, OH, PA, VA, AND WI.

© 2011–2012 Obama for America. All Rights Reserved.

# Real world motivation: NETFLIX artwork

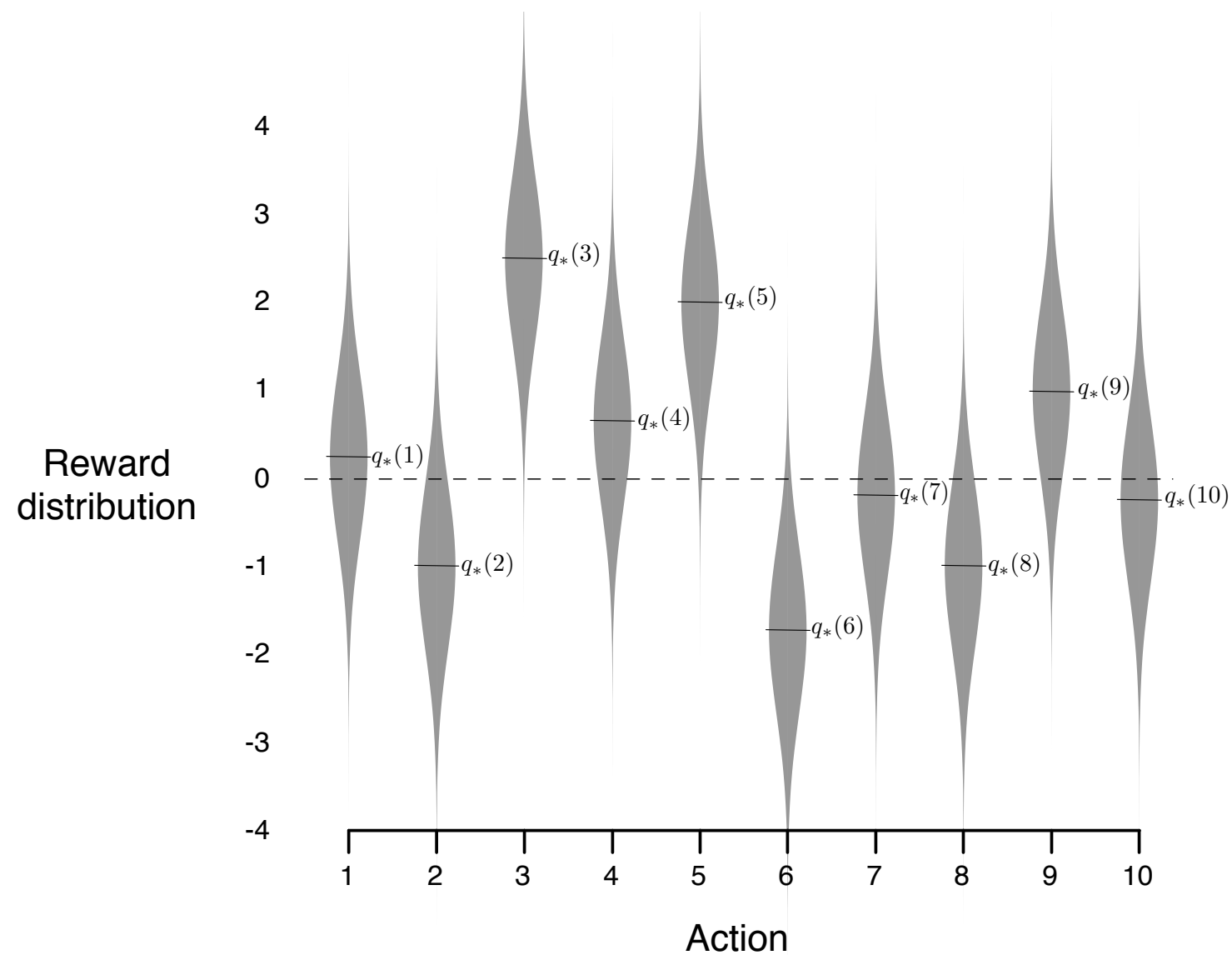
For a particular movie, we want to decide what image to show (to all the NETFLIX users)

- Actions: uploading one of the  $K$  images to a user's home screen
- Reward: 1 if the user clicks and watches, 0 otherwise.
- Mean reward (success probability) for each image: the percentage of users that clicked and watched (quality engagement, not clickbait)



# Example: Gaussian Bandits

- Each action (arm when played) results in a **real number**.
- Action (arm)  $k \in \{1 \dots K\}$  produces on average reward equal to the mean of its Gaussian distribution.



# Regret

- The **action-value** is the mean reward for action  $a$ ,

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

- The **optimal value** is

$$v_* = q(a^*) = \max_{a \in \mathcal{A}} q_*(a)$$

- The **regret** is the opportunity loss for one step. For an algorithm that selects action  $a_t$  at timestep  $t$  it reads:

$$I_t = \mathbb{E}[v_* - q_*(a_t)] \quad \text{reward} = - \text{regret}$$

- The **total regret** is the total opportunity loss

$$L_T = \mathbb{E} \left[ \sum_{t=1}^T v_* - q_*(a_t) \right]$$

- Maximize cumulative expected reward = minimize total regret

# Regret

- The **count**  $N_t(a)$ : the number of times that action  $a$  has been selected prior to time  $t$
- The **gap**  $\Delta a$  is the difference in value between action  $a$  and optimal action  $a_*$ :  $\Delta_a = v_* - q_*(a)$
- Regret is a function of gaps and the counts

$$\begin{aligned} L_T &= \mathbb{E} \left[ \sum_{t=1}^T v_* - q_*(a_t) \right] \\ &= \sum_{a \in \mathcal{A}} \mathbb{E}[N_t(a)](v_* - q_*(a)) \\ &= \sum_{a \in \mathcal{A}} \mathbb{E}[N_t(a)] \Delta_a \end{aligned}$$

# Forming Action-Value Estimates

- To simplify notation, let us focus on one action
  - We consider only its rewards, and its estimate after  $n+1$  rewards:

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

- How can we do this incrementally (without storing all the rewards)?
- Could store a running sum and count (and divide), or equivalently:

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

- This is a standard form for learning/update rules:

$$\text{NewEstimate} \leftarrow \text{OldEstimate} + \text{StepSize} [\text{Target} - \text{OldEstimate}]$$

# Forming Action-Value Estimates

- To simplify notation, let us focus on one action
  - We consider only its rewards, and its estimate after  $n+1$  rewards:

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

- How can we do this incrementally (without storing all the rewards)?
- Could store a running sum and count (and divide), or equivalently:

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

- This is a standard form for learning/update rules: error

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

# Derivation of incremental update

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

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



# Non-stationary bandits

- Suppose the true action values change slowly over time
  - then we say that the problem is *nonstationary*
- In this case, sample averages are not a good idea
  - Why?

# Non-stationary bandits

- Suppose the true action values change slowly over time
  - then we say that the problem is *nonstationary*
- In this case, sample averages are not a good idea
- Better is an “exponential, recency-weighted average”:

$$\begin{aligned} Q_{n+1} &\doteq Q_n + \alpha \left[ R_n - Q_n \right] \\ &= (1 - \alpha)^n Q_1 + \sum_{i=1}^n \alpha (1 - \alpha)^{n-i} R_i, \end{aligned}$$

where  $\alpha \in (0,1]$  and constant

The smaller the  $i$ , the smaller  $(1 - \alpha)^{n-i}$  -> forgetting earlier rewards

# Action selection in multi-armed bandits

# Fixed exploration period + Greedy

1. Allocate a fixed time period to exploration when you try bandits **uniformly at random**

2. Estimate mean rewards for all actions:  $Q_t(a) = \frac{1}{N_t(a)} \sum_{i=1}^{t-1} r_i \mathbf{1}(A_i = a)$

3. Select the action that is optimal for the estimated mean rewards, breaking ties at random:  $a_t = \operatorname{argmax}_{a \in \mathcal{A}} Q_t(a)$

4. GOTO 2

# Fixed exploration period + Greedy

- After the fixed exploration period we have formed the following reward estimates



$$Q_t(a_1) = 0.3$$



$$Q_t(a_2) = 0.5$$



$$Q_t(a_3) = 0.1$$

**Q1:** Will the greedy method always pick the second action?

**Q2:** Can greedy lock onto a suboptimal action forever?

⇒ Greedy has linear total regret

# $\epsilon$ -Greedy Action Selection

- In greedy action selection, you always exploit
- In  $\epsilon$ -greedy, you are usually greedy, but with probability  $\epsilon$  you instead pick an action at random (possibly the greedy action again)
- This is perhaps the simplest way to balance exploration and exploitation

# $\epsilon$ -Greedy Action Selection

## A simple bandit algorithm

Initialize, for  $a = 1$  to  $k$ :

$$Q(a) \leftarrow 0$$

$$N(a) \leftarrow 0$$

Repeat forever:

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

$$R \leftarrow \text{bandit}(A)$$

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

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

# $\epsilon$ -Greedy Algorithm

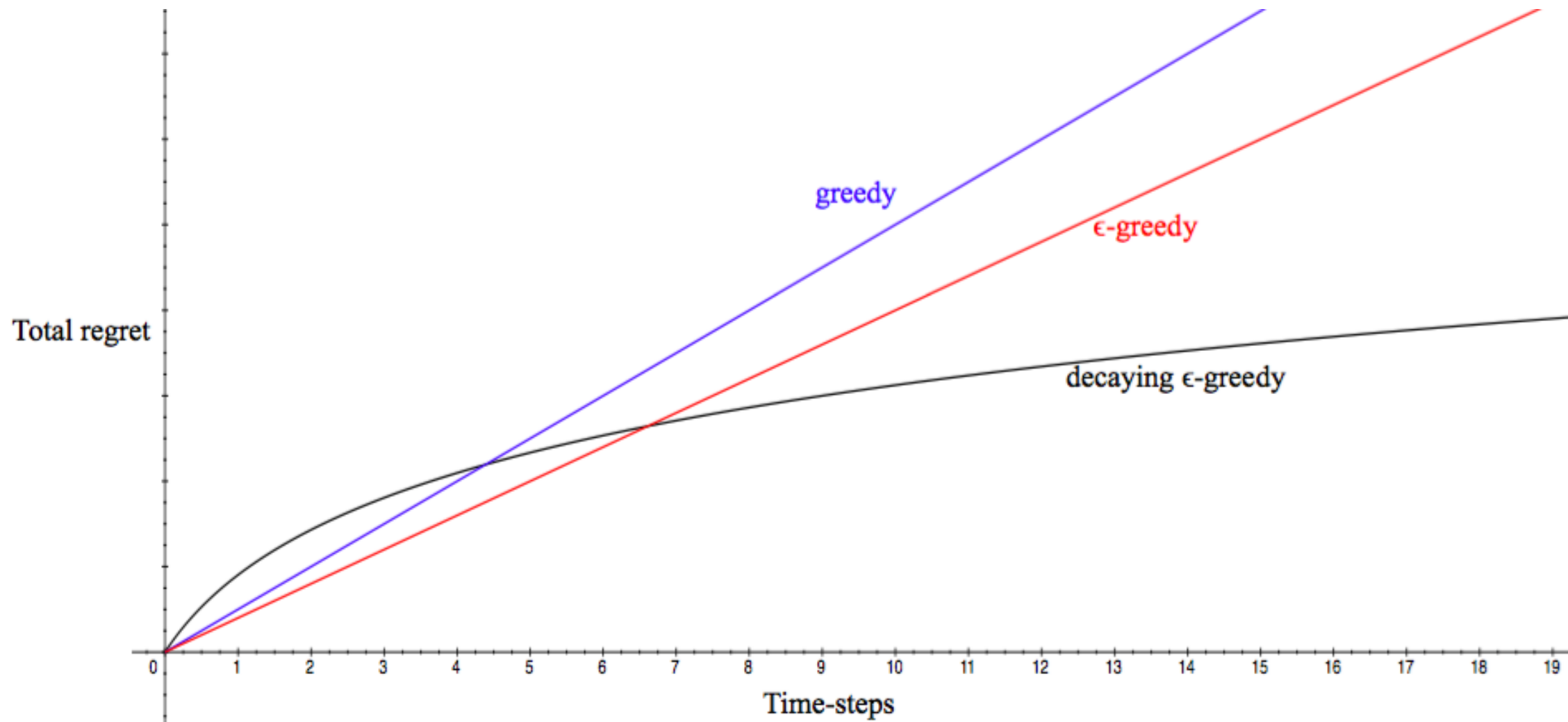
- The  $\epsilon$ -greedy algorithm continues to explore forever
  - With probability  $1 - \epsilon$  select  $a_t = \operatorname{argmax}_{a \in \mathcal{A}} Q_t(a)$
  - With probability  $\epsilon$  select a random action (independent of its Q estimate)
- Constant  $\epsilon$  ensures **minimum regret**

$$I_t \geq \frac{\epsilon}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} \Delta_a$$

- $\Rightarrow$   $\epsilon$ -greedy has linear total regret



# Counting Regret

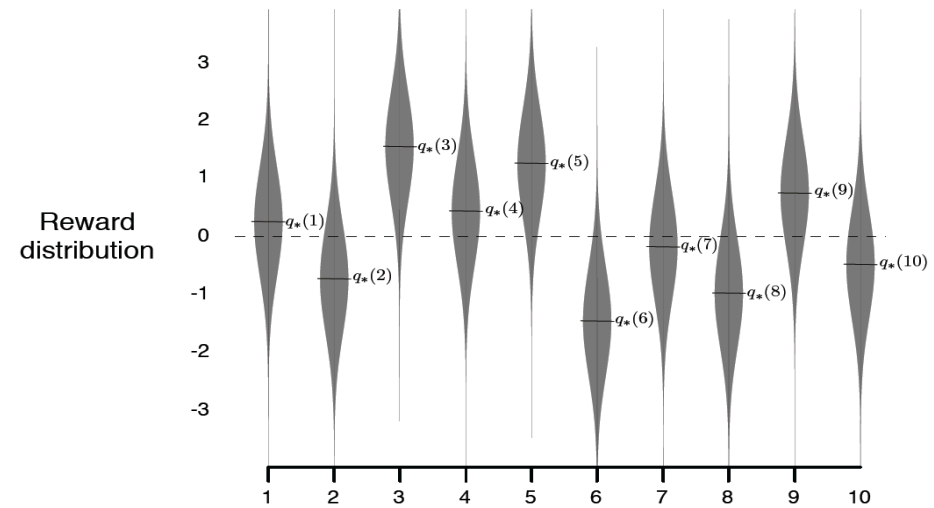


- If an algorithm forever explores it will have linear total regret
- If an algorithm never explores it will have linear total regret

# Average reward for three algorithms

We sample 10 arm bandits instantiations:

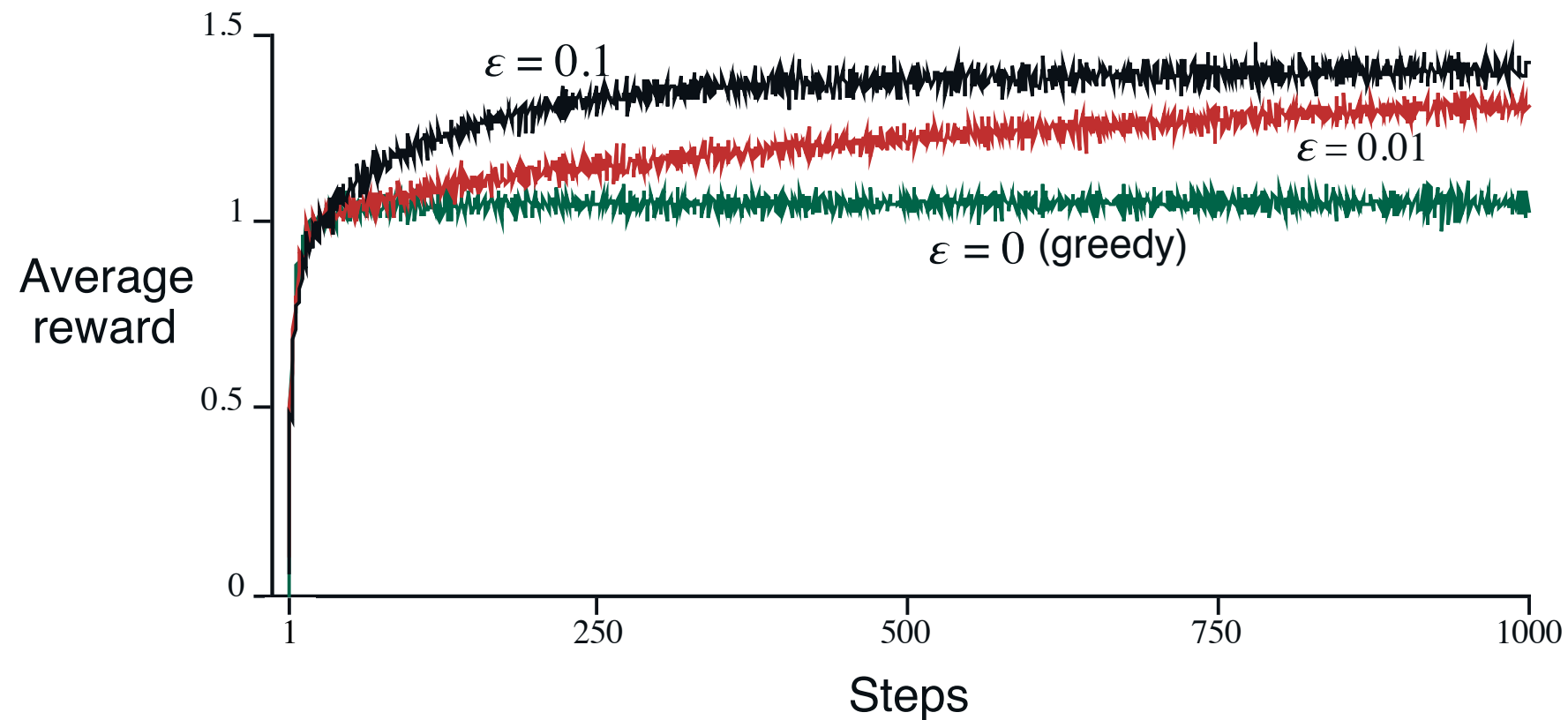
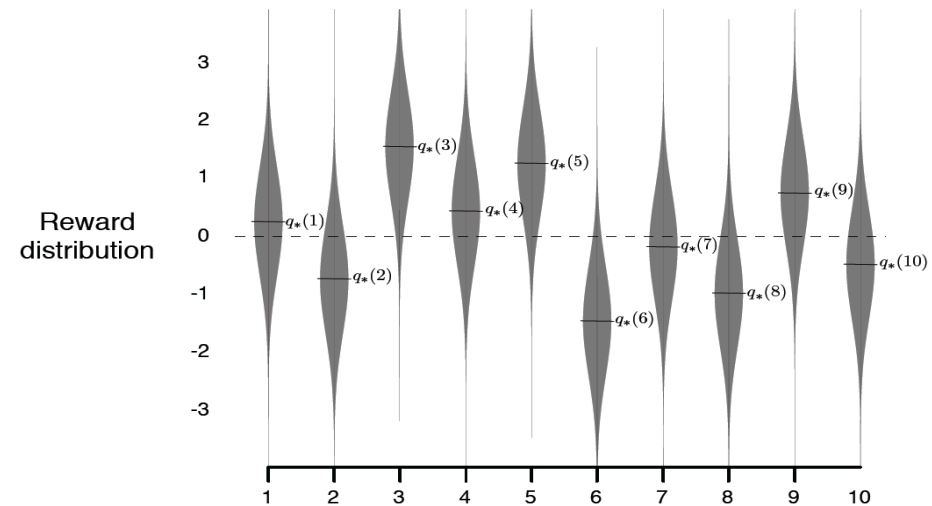
$$q_*(a) \sim \mathcal{N}(0, 1)$$
$$R_t \sim \mathcal{N}(q_*(a), 1)$$



# Average reward for three algorithms

We sample 10 arm bandits instantiations:

$$q_*(a) \sim \mathcal{N}(0, 1)$$
$$R_t \sim \mathcal{N}(q_*(a), 1)$$

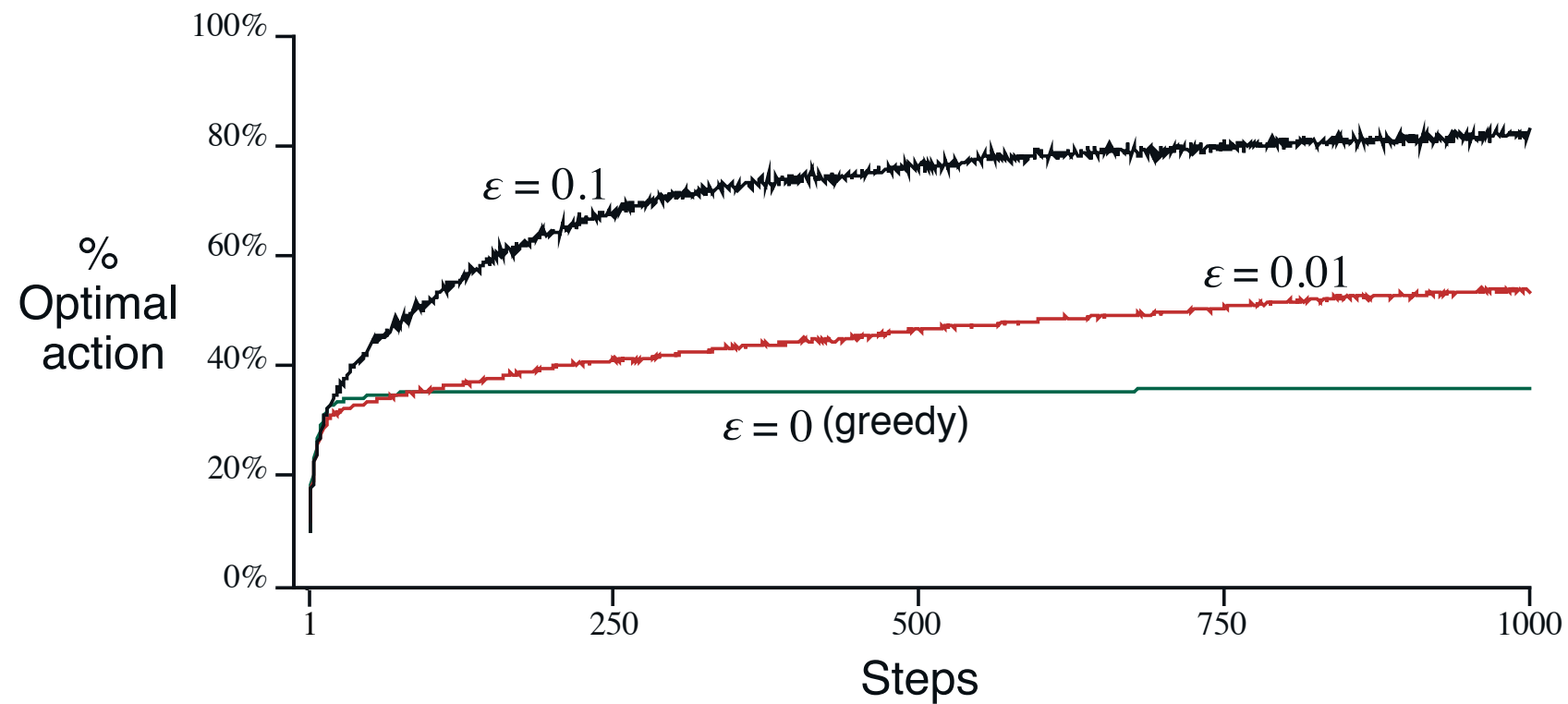
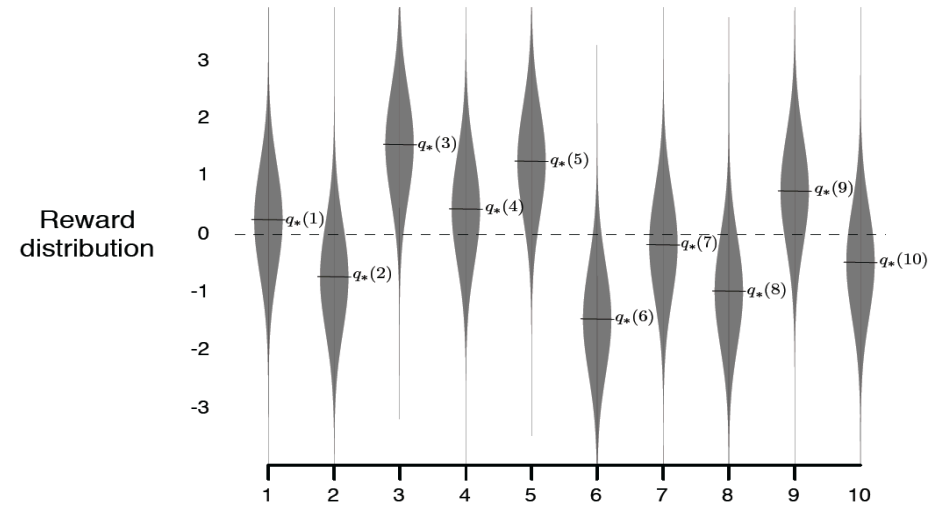


Q: In the limit (after infinite number of steps), which method will result in the largest average reward?

# Optimal action for three algorithms

We sample 10 arm bandits instantiations:

$$q_*(a) \sim \mathcal{N}(0, 1)$$
$$R_t \sim \mathcal{N}(q_*(a), 1)$$

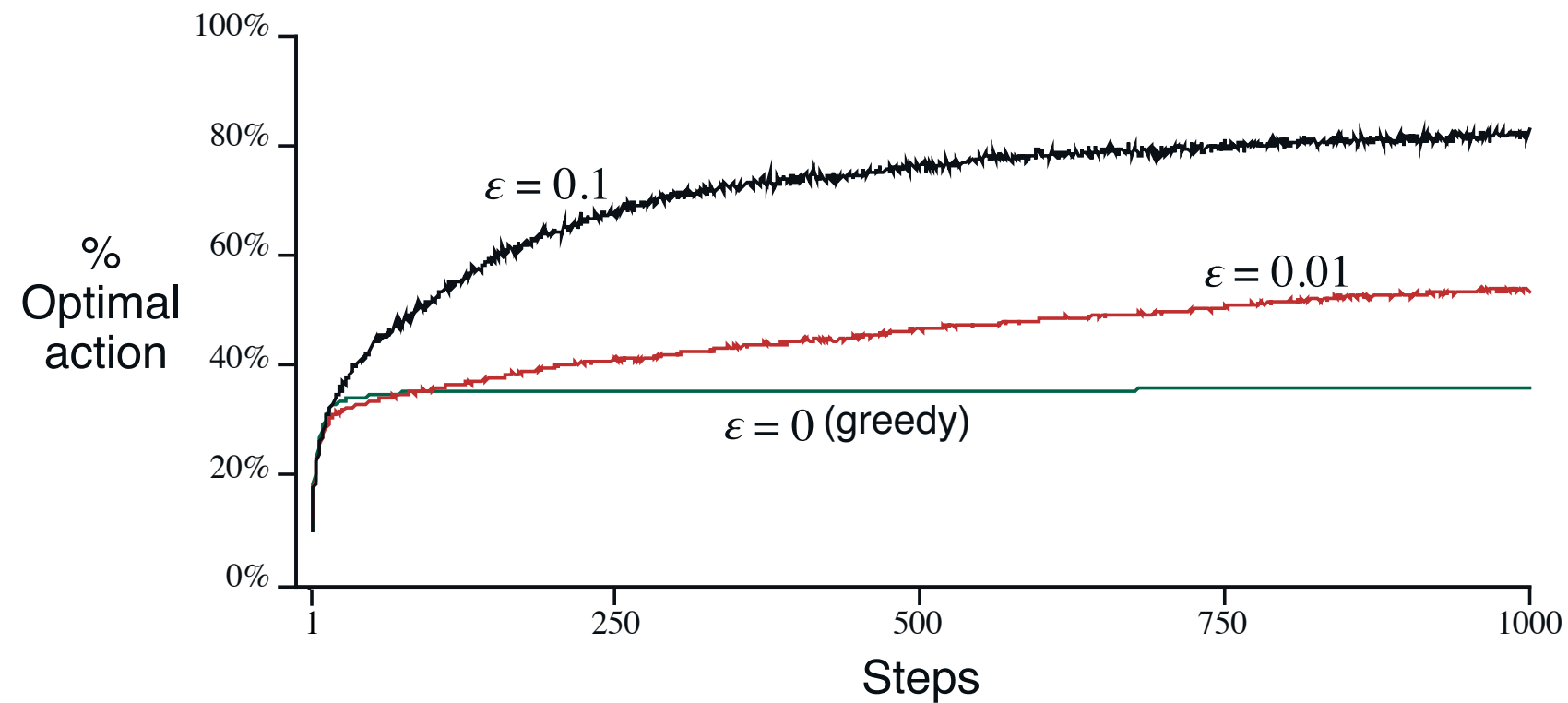
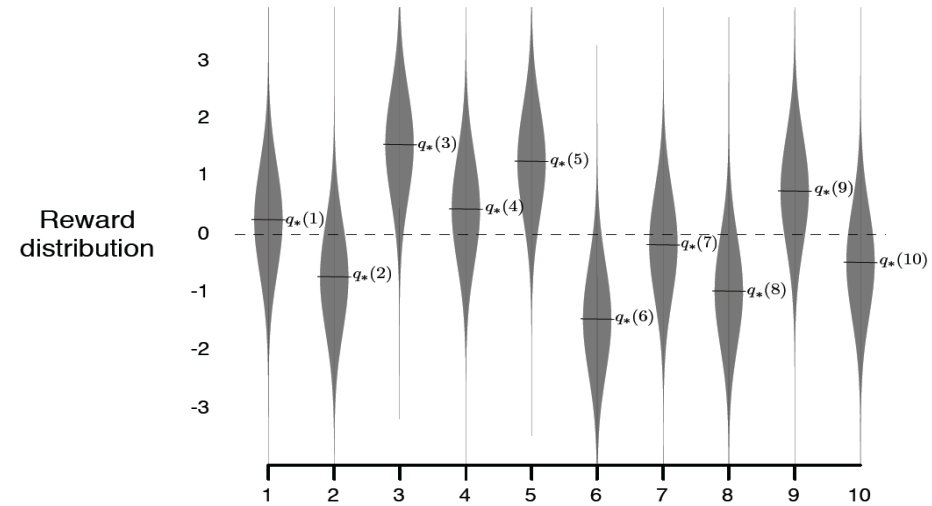


Q: Which method will find the optimal action in the limit?

# Optimal action for three algorithms

We sample 10 arm bandits instantiations:

$$q_*(a) \sim \mathcal{N}(0, 1)$$
$$R_t \sim \mathcal{N}(q_*(a), 1)$$



Q: Does the performance of those methods depend on the initialization of the action value estimates?

# Optimistic Initialization

- Simple and practical ideas: initialise  $Q(a)$  to a high value
- Update action value by incremental Monte-Carlo evaluation
- Starting with  $N(a) > 0$ ,

$$Q_t(a_t) = Q_{t-1}(a_t) + \frac{1}{N_t(a_t)} \left( r_t - Q_{t-1}(a_t) \right)$$

just an incremental estimate of sample mean, including one 'hallucinated' initial optimistic value

- Encourages systematic exploration early on
- But optimistic greedy can still lock onto a suboptimal action if rewards are stochastic.

# Optimistic Initial Values

We initialize with the following reward estimates for Bernoulli bandits



$$Q_t(a_1) = 1$$



$$Q_t(a_2) = 1$$



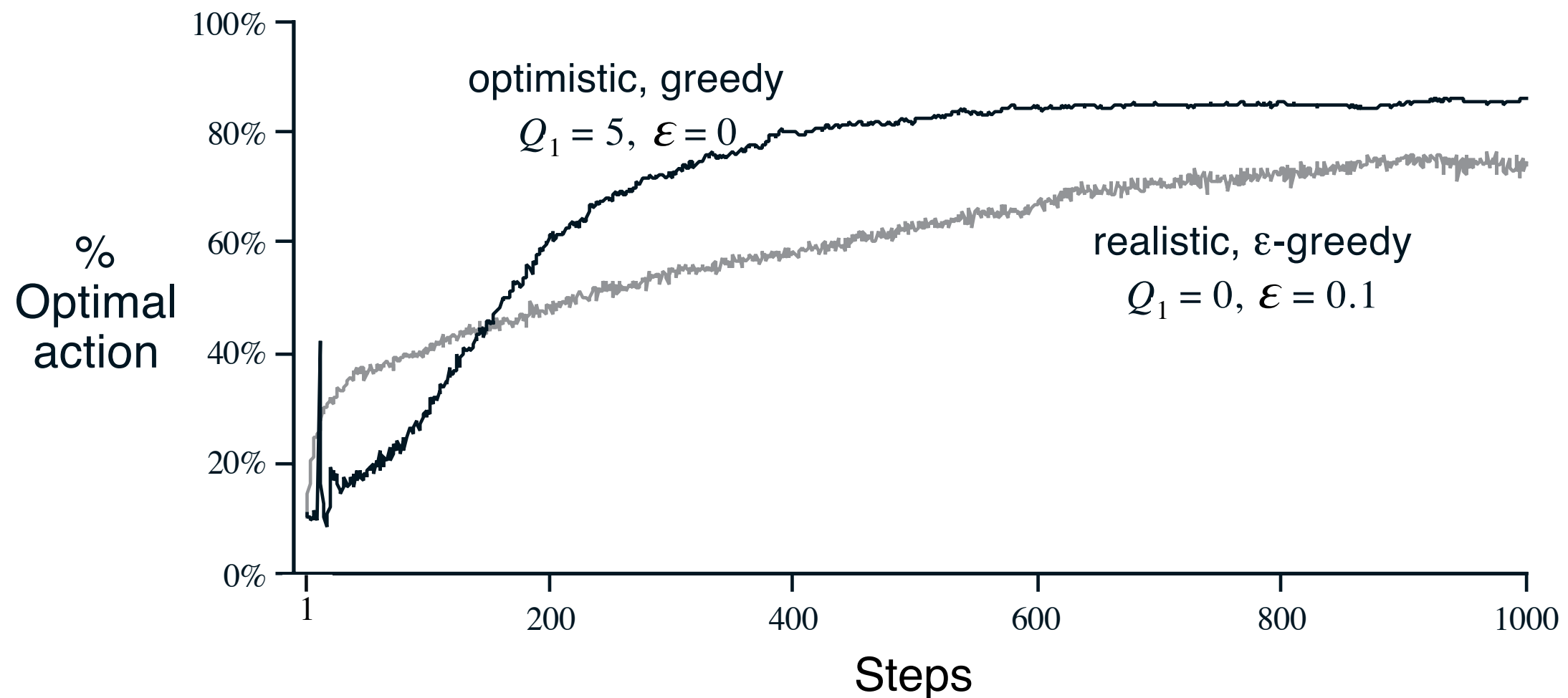
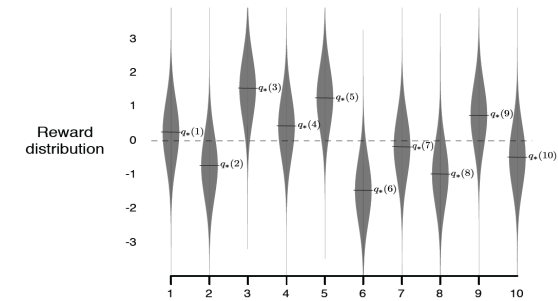
$$Q_t(a_3) = 1$$

Q: When it is possible that greedy action selection will not try out all the actions?

# Optimistic Initial Values

- Suppose we initialize the action values optimistically ( $Q_1(a) = 5$ ), e.g., on the 10-armed testbed

$$q_*(a) \sim \mathcal{N}(0, 1)$$
$$R_t \sim \mathcal{N}(q_*(a), 1)$$





# Achieving sub-linear total regret

We need to reason about **uncertainty** of our action value estimates



1000 pulls,  
600 wins  
 $Q_t(a_1)=0.6$



1000 pulls,  
400 wins  
 $Q_t(a_2)=0.4$



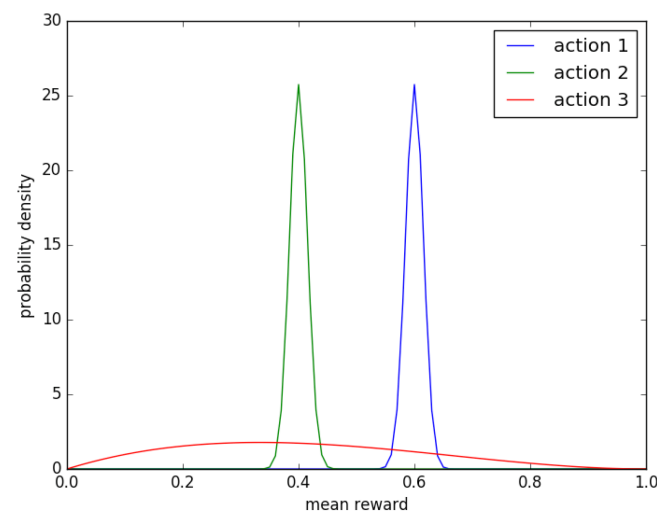
10 pulls,  
4 wins  
 $Q_t(a_1)=0.4$

Epsilon-greedy

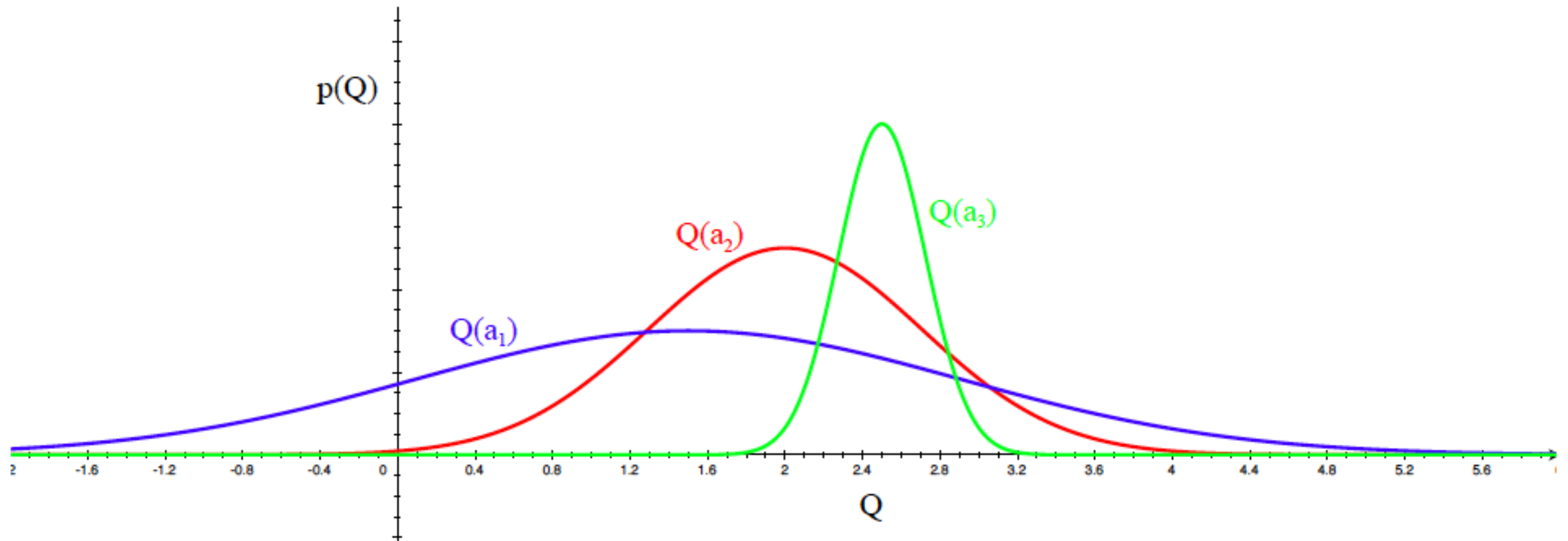
Repeat forever:

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

The problem with using mean estimates is that we do not reason about **uncertainty** of those estimates.

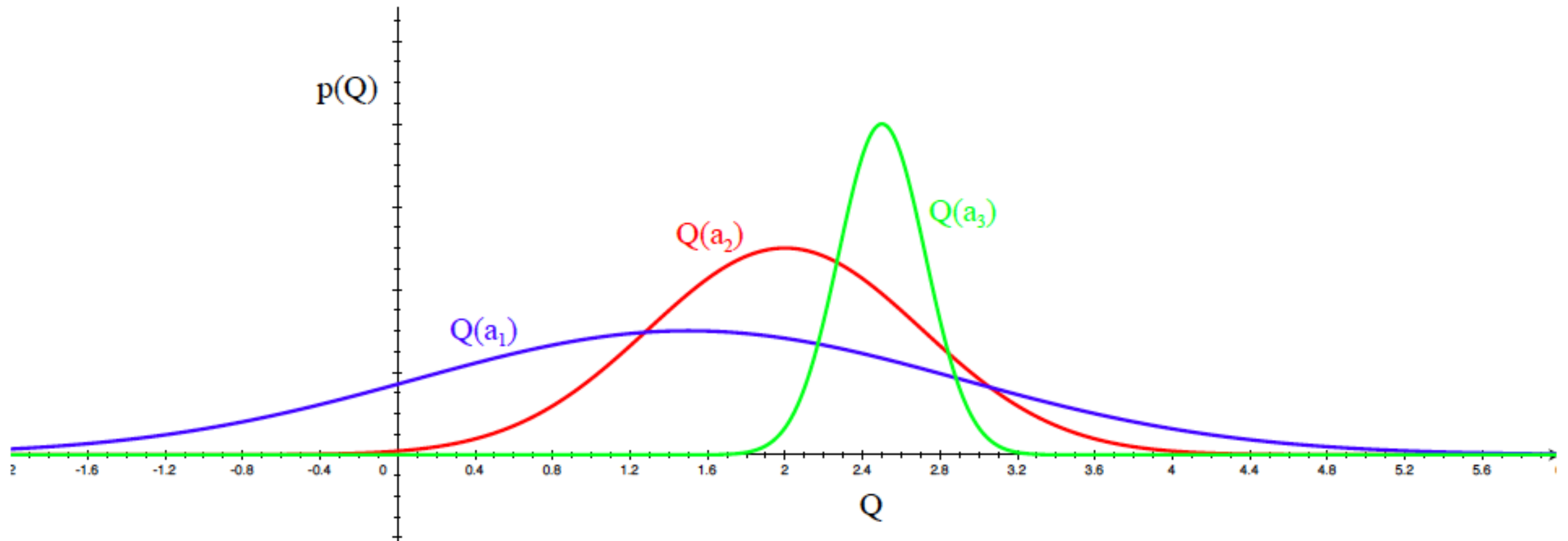


# Uncertainty guides Exploration



- The more **uncertain** we are about an action-value
- The more important it is to explore that action
- It could turn out to be the best action

# Uncertainty guides Exploration



- We are then less uncertain about the value
- And more likely to pick another action
- Until we converge to the best action

# Upper Confidence Bounds

- Estimate an **upper confidence**  $U_t(a)$  for each action value
- Such that with high probability

$$q_*(a) \leq Q_t(a) + U_t(a)$$



- This upper confidence depends on the number of times action  $a$  has been selected
  - Small  $N_t(a) \Rightarrow$  large  $U_t(a)$  (estimated value is uncertain)
  - Large  $N_t(a) \Rightarrow$  small  $U_t(a)$  (estimated value is accurate)
- Select action maximizing **Upper Confidence Bound** (UCB)

$$a_t = \operatorname{argmax}_{a \in \mathcal{A}} Q_t(a) + U_t(a)$$

# Hoeffding's Inequality

Let  $X_1, \dots, X_t$  be independent random variables in the range  $[0,1]$  with  $\mathbb{E}(X_i) = \mu$ . Then for  $u > 0$ ,

sample mean

$$\mathbb{P}\left(\frac{1}{n} \sum_{i=1}^n X_i \geq \mu + u\right) \leq e^{-2u^2n}$$
$$\mathbb{P}\left(\frac{1}{n} \sum_{i=1}^n X_i \leq \mu - u\right) \leq e^{-2u^2n}$$

**Hoeffding's inequality** provides an **upper bound** on the **probability** that the sum of bounded **independent random variables** deviates from its **expected value** by more than a certain amount.

# Hoeffding's Inequality

Let  $X_1, \dots, X_n$  be independent random variables in the range  $[0,1]$  with  $\mathbb{E}(X_i) = \mu$ . Then for  $u > 0$ ,

$$\mathbb{P} \left( \frac{1}{n} \sum_{i=1}^n X_i \geq \mu + u \right) \leq e^{-2u^2 n}$$

I made the margin to depend on the amount of interactions  $t$



- We will apply Hoeffding's Inequality to the rewards obtained from each action (bandit)  $a$ :

$$\mathbb{P} \left( \hat{Q}_t(a) \geq q(a) + U_t(a) \right) \leq e^{-2U_t(a)^2 N_t(a)}$$

- $t$ : how many times I have played any action,
- $N_t(a)$ : how many times I have played action  $a$  in  $t$  interactions

# Calculating Upper Confidence Bounds

- Pick a probability  $p$  that the value estimate deviates from its mean
- Now solve for  $U_t(a)$

$$e^{-2U_t(a)^2 N_t(a)} = p$$

$$U_t(a) = \sqrt{\frac{-\log p}{2N_t(a)}}$$

- Reduce  $p$  as we play more, e.g.  $p = t^{-c}$ ,  $c = 4$
- Ensures we select optimal action as  $t \rightarrow \infty$

$$U_t(a) = \sqrt{\frac{2 \log t}{2N_t(a)}}$$



# Upper Confidence Bound (UCB)

- A clever way of reducing exploration over time
- Estimate an upper bound on the true action values
- Select the action with the largest (estimated) upper bound

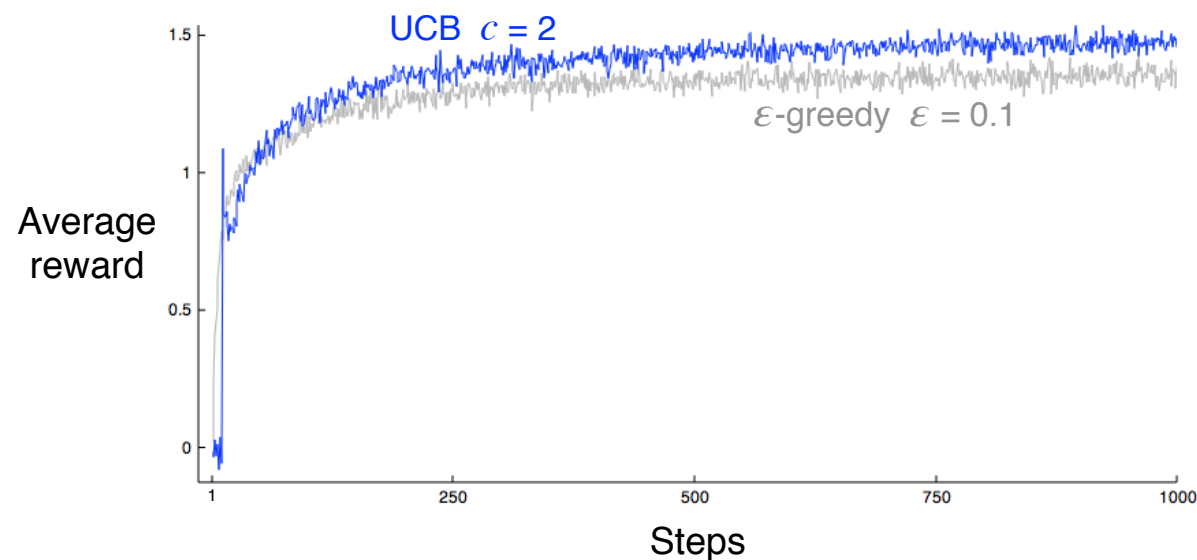
$$A_t \doteq \arg \max_a \left[ Q_t(a) + c \sqrt{\frac{\log t}{N_t(a)}} \right]$$

- $c$  is a hyper-parameter that trades-off explore/exploit
- the confidence bound grows with the total number of actions we have taken  $t$  but shrinks with the number of times we have tried this particular action  $N_t(a)$ . This ensures each action is tried infinitely often but still balances exploration and exploitation.

# Upper Confidence Bound (UCB)

- A clever way of reducing exploration over time
- Estimate an upper bound on the true action values
- Select the action with the largest (estimated) upper bound

$$A_t \doteq \arg \max_a \left[ Q_t(a) + c \sqrt{\frac{\log t}{N_t(a)}} \right]$$



# UCB1 Algorithm

- ▶ This leads to the **UCB1** algorithm

$$a_t = \operatorname{argmax}_{a \in \mathcal{A}} Q(a) + \sqrt{\frac{2 \log t}{N_t(a)}}$$

## Theorem

*The UCB algorithm achieves logarithmic asymptotic total regret*

$$\lim_{t \rightarrow \infty} L_t \leq 8 \log t \sum_{a | \Delta_a > 0} \Delta_a$$

# Bayesian Bandits

- So far we have made no assumptions about the reward distributions.
  - In UCB we just considered some bounds on rewards
- Bayesian bandits exploit **prior knowledge of rewards**,  $p[\mathcal{R}]$
- They compute **posterior** distribution of rewards  $p[\mathcal{R} | h_t]$   
$$h_t = a_1, r_1, \dots, a_{t-1}, r_{t-1}$$
- Use posterior to guide exploration: we simply sample from the posterior!

# Bayesian learning for model parameters

**Step 1:** Given  $n$  data,  $D = x_{1\dots n} = \{x_1, x_2, \dots, x_n\}$  write down the expression for likelihood:

$$p(D | \theta)$$

**Step 2:** Specify a prior:  $p(\theta)$

**Step 3:** Compute the posterior:

$$p(\theta | D) = \frac{p(D | \theta)p(\theta)}{p(D)}$$

# Thompson Sampling

Represent a distribution for the mean reward of each bandit as opposed to the mean reward estimate alone. At each timestep:

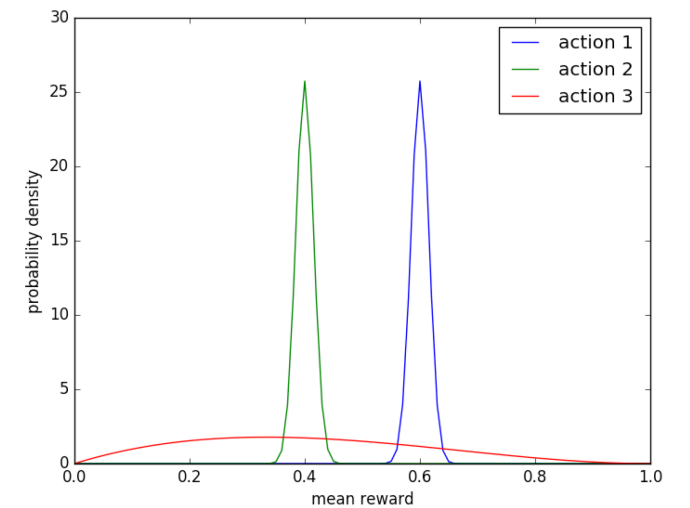
1. Sample from the mean reward distributions:

$$\bar{\theta}_1 \sim \hat{p}(\theta_1), \bar{\theta}_2 \sim \hat{p}(\theta_2), \dots, \bar{\theta}_k \sim \hat{p}(\theta_k)$$

2. Choose action  $a = \arg \max_a \mathbb{E}_{\bar{\theta}}[r(a)]$

3. Observe the reward.

4. Update the mean reward posterior distributions:  $\hat{p}(\theta_1), \hat{p}(\theta_2) \dots \hat{p}(\theta_k)$



Q: why we use argmax in step 2 and we do not add any noise?

# Bernoulli bandits - Prior

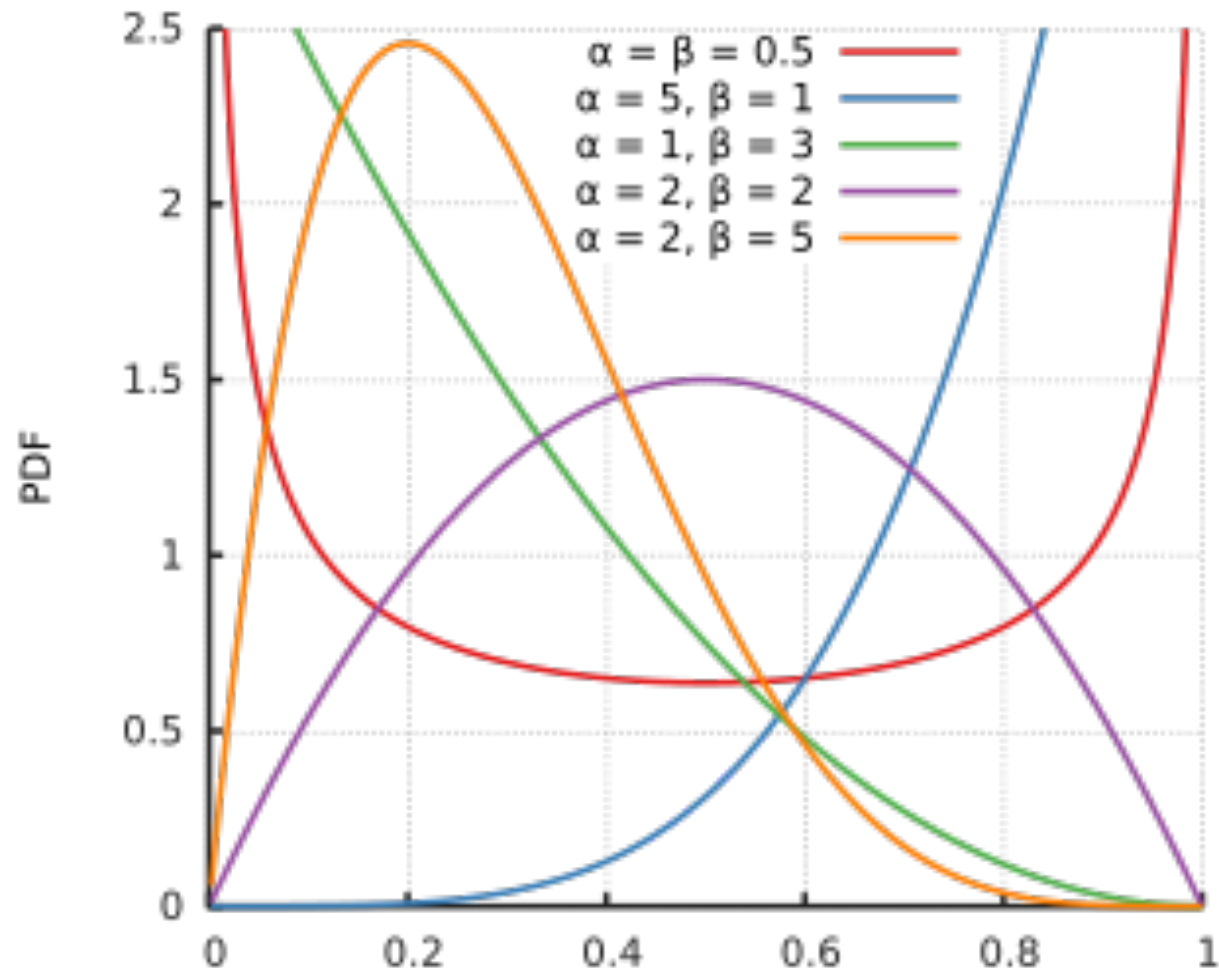
Let's consider a Beta distribution **prior** over the mean rewards of the Bernoulli bandits:

$$p(\theta_k) = \frac{\Gamma(\alpha_k + \beta_k)}{\Gamma(\alpha_k)\Gamma(\beta_k)} \theta_k^{\alpha_k-1} (1 - \theta_k)^{\beta_k-1} \quad \Gamma(n) = (n - 1)!$$

The mean is  $\frac{\alpha}{\alpha + \beta}$

The larger the  $\alpha + \beta$  the more concentrated the distribution

Beta( $\alpha, \beta$ )



# Bernoulli bandits-Posterior

Let's consider a Beta distribution prior over the mean rewards of the Bernoulli bandits:

$$p(\theta_k) = \frac{\Gamma(\alpha_k + \beta_k)}{\Gamma(\alpha_k)\Gamma(\beta_k)} \theta_k^{\alpha_k - 1} (1 - \theta_k)^{\beta_k - 1} \quad \Gamma(n) = (n - 1)!$$

The posterior is also a Beta. Because beta is conjugate distribution for the Bernoulli distribution.

A closed form solution for the bayesian update, possible only for conjugate distributions.

$$(\alpha_k, \beta_k) \leftarrow \begin{cases} (\alpha_k, \beta_k) & \text{if } x_t \neq k \\ (\alpha_k, \beta_k) + (r_t, 1 - r_t) & \text{if } x_t = k. \end{cases}$$



# Greedy VS Thompson for Bernoulli bandits

---

## Algorithm 1 BernGreedy( $K, \alpha, \beta$ )

---

1: **for**  $t = 1, 2, \dots$  **do**  
2:   #estimate model:  
3:   **for**  $k = 1, \dots, K$  **do**  
4:      $\hat{\theta}_k \leftarrow \alpha_k / (\alpha_k + \beta_k)$   
5:   **end for**  
6:  
7:   #select and apply action:  
8:    $x_t \leftarrow \operatorname{argmax}_k \hat{\theta}_k$   
9:   Apply  $x_t$  and observe  $r_t$   
10:  
11:   #update distribution:  
12:    $(\alpha_{x_t}, \beta_{x_t}) \leftarrow (\alpha_{x_t}, \beta_{x_t}) + (r_t, 1 - r_t)$   
13: **end for**

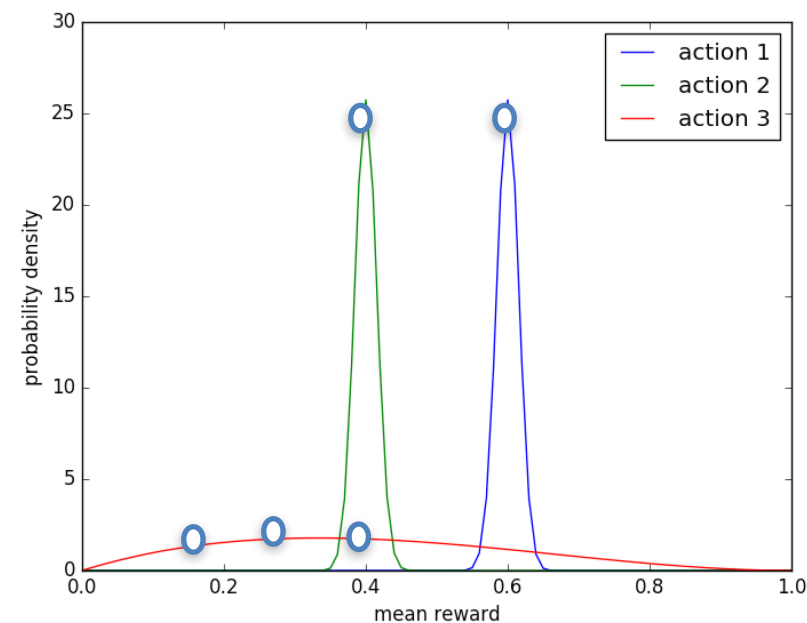
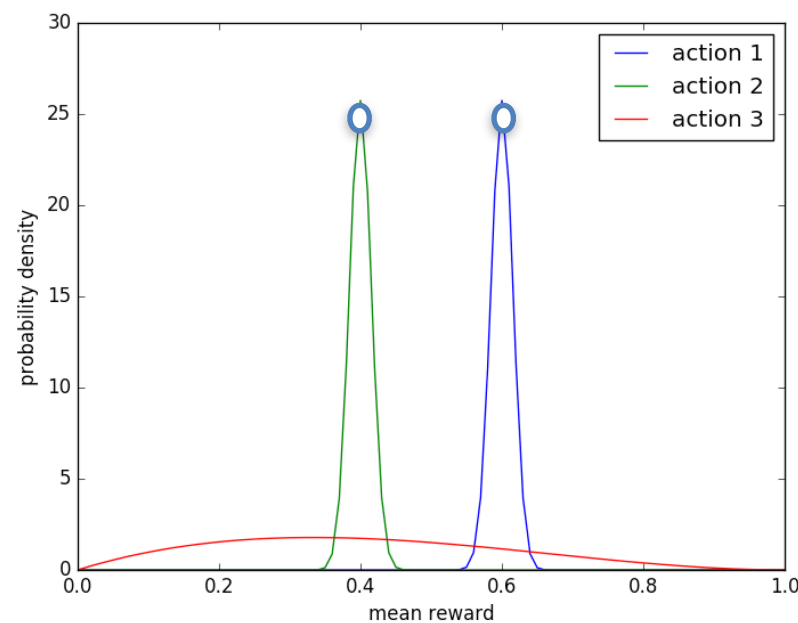
a: success  
b: failure

---

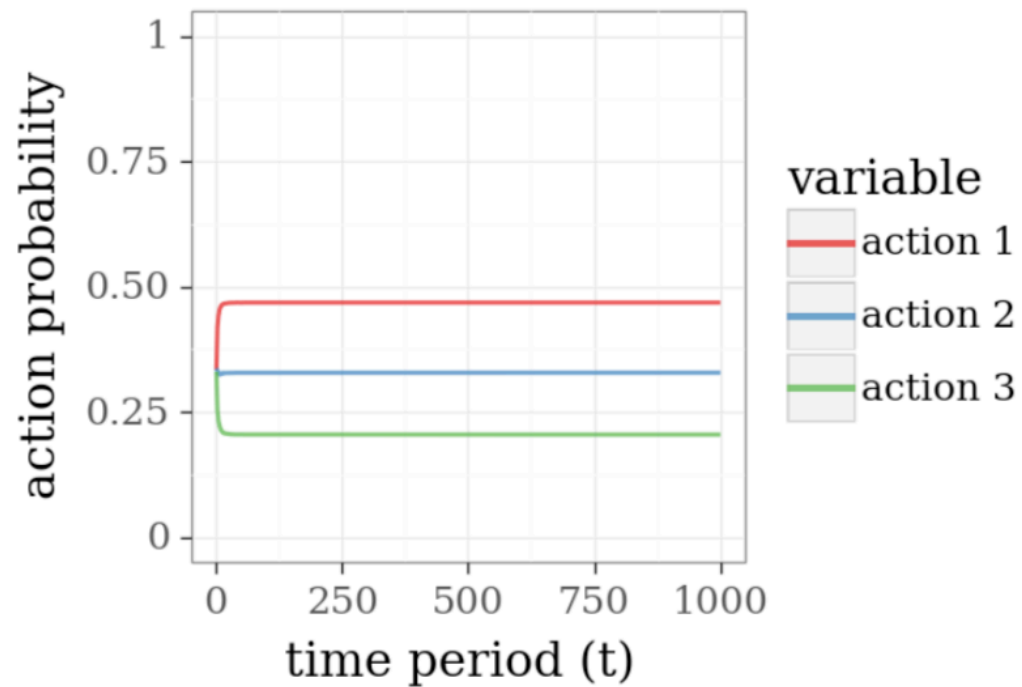
## Algorithm 2 BernThompson( $K, \alpha, \beta$ )

---

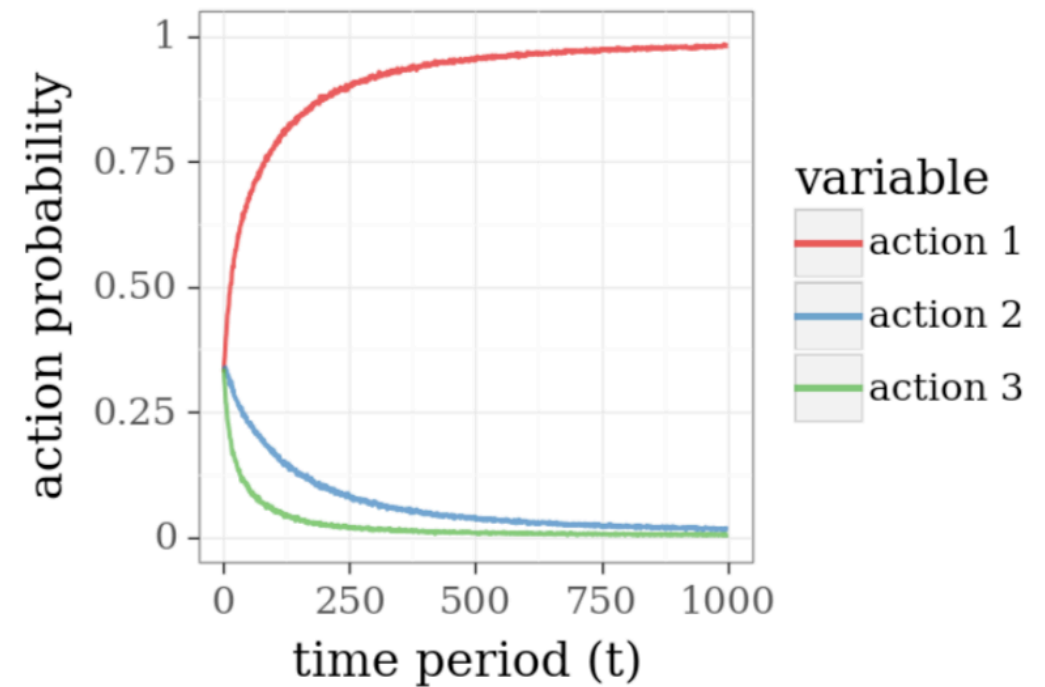
1: **for**  $t = 1, 2, \dots$  **do**  
2:   #sample model:  
3:   **for**  $k = 1, \dots, K$  **do**  
4:     Sample  $\hat{\theta}_k \sim \text{beta}(\alpha_k, \beta_k)$   
5:   **end for**  
6:  
7:   #select and apply action:  
8:    $x_t \leftarrow \operatorname{argmax}_k \hat{\theta}_k$   
9:   Apply  $x_t$  and observe  $r_t$   
10:  
11:   #update distribution:  
12:    $(\alpha_{x_t}, \beta_{x_t}) \leftarrow (\alpha_{x_t}, \beta_{x_t}) + (r_t, 1 - r_t)$   
13: **end for**



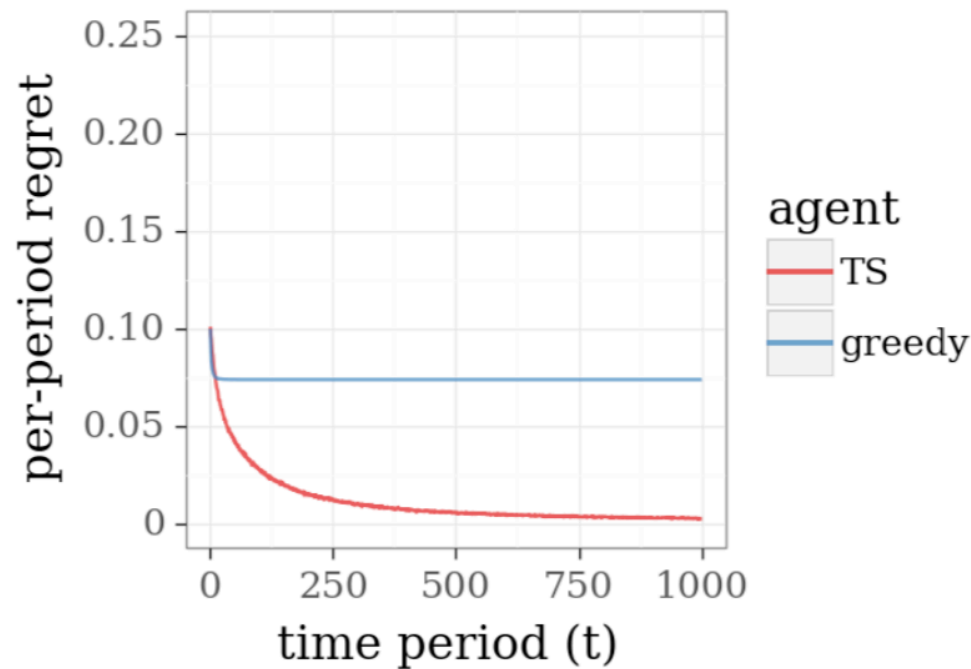
Using uniform prior in  $[0,1]$  for the success probabilities



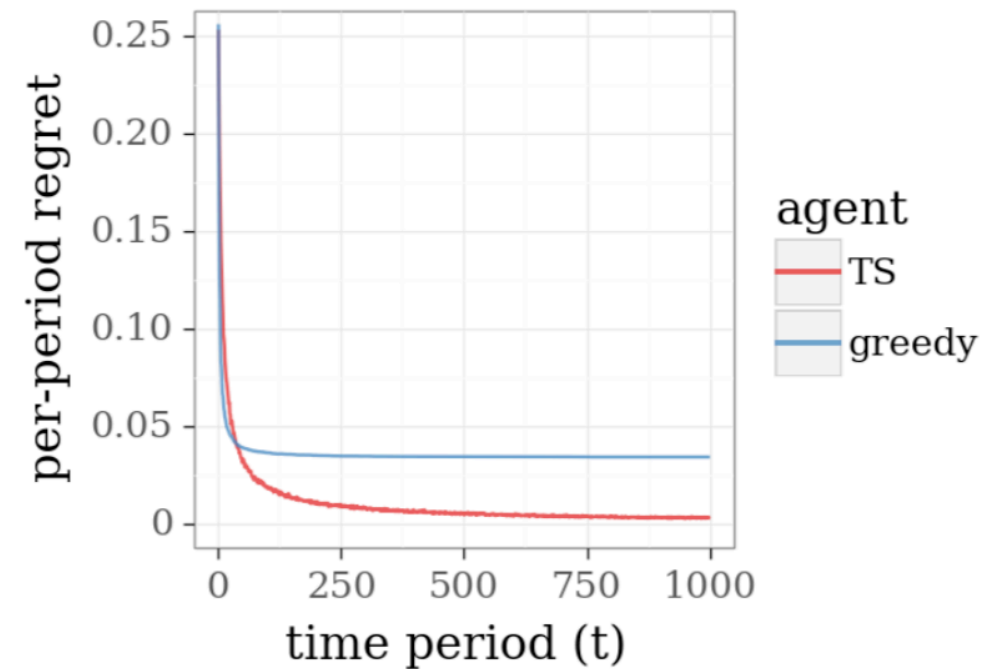
(a) greedy algorithm



(b) Thompson sampling



(a)  $\theta = (0.9, 0.8, 0.7)$

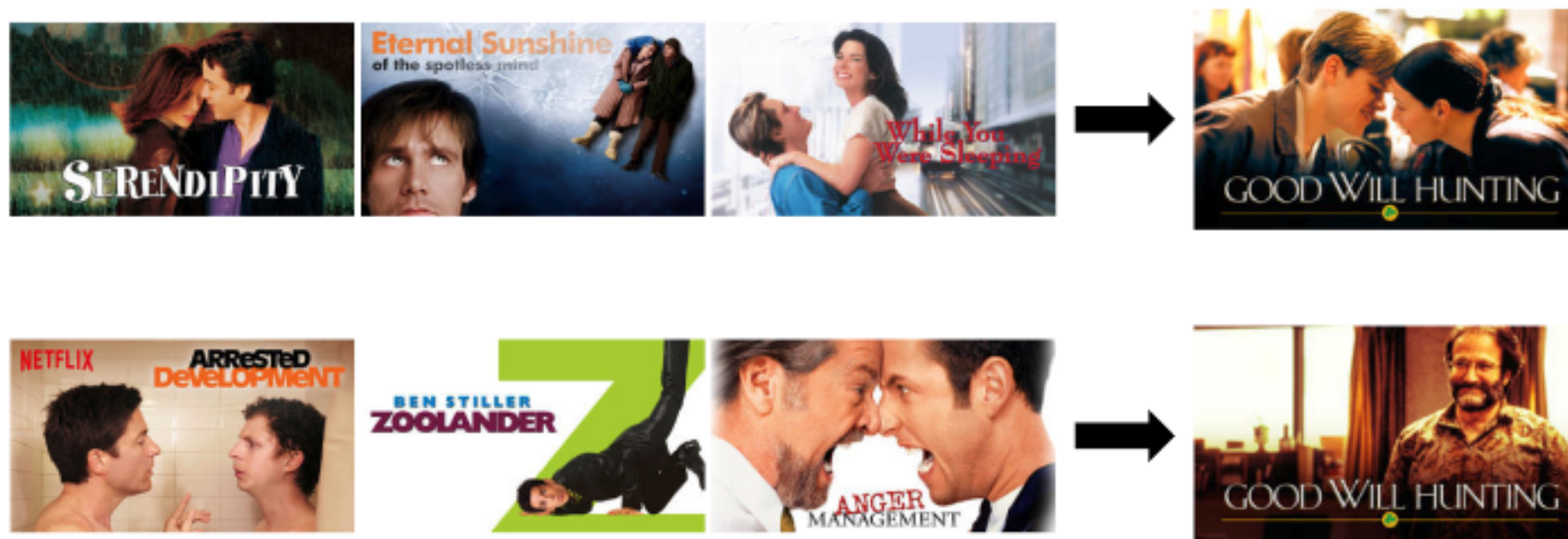


(b) average over random  $\theta$

# Contextual Bandits (a.k.a Associative Search)

- A contextual bandit is a tuple  $(A, S, R)$
- $A$  is a known set of  $k$  actions (or “arms”)
- $\mathcal{S} = \mathbb{P}[s]$  is an unknown distribution over states (or “contexts”)
- $\mathcal{R}_s^a(r) = \mathbb{P}[r | s, a]$  is an unknown probability distribution over rewards
- At each time  $t$ 
  - Environment generates state  $s_t \sim \mathcal{S}$
  - Agent selects action  $a_t \in \mathcal{A}$
  - Environment generates reward  $r_t \sim \mathcal{R}_{s_t}^{a_t}$
- **The goal** is to maximize cumulative reward  $\sum_{\tau=1}^t r_\tau$

# Real world motivation: Personalized NETFLIX artwork



For a particular title **and a particular user**, we will use the **contextual** multi-armed bandit formulation to decide what image to show per title **per user**

- Actions: uploading an image (available for this movie title) to a user's home screen
- Mean rewards (unknown): the % of NETFLIX users that will click on the title and watch the movie
- Estimated mean rewards: the average click rate (+quality engagement, not clickbait)
- **Context** (s) : user attributes, e.g., language preferences, gender of films she has watched, time and day of the week, etc.

Netflix Artwork: <https://medium.com/netflix-techblog/artwork-personalization-c589f074ad76>

# Question: was there a train and test phase on our multi-armed bandit algorithms?

No, the setup we explored was: given a set of  $K$  arms, how do we select actions to minimize our cumulative regret.

Q: what would be the learning based equivalent of the multi-armed bandit problem?

A:

- We have a training set of  $N$  multi-armed bandit instantiations.
- Each  $K$ -armed bandit is one training example.
- The agent gets  $n$  number of interactions, and obtains a final reward (-regret).
- The agent learns a policy –mapping from its set of actions taken thus far **and** their outcomes, to a probability over what actions to try next

We will visit this setup in the meta-learning lecture.