Multi-arm Bandits

Sutton and Barto, Chapter 2

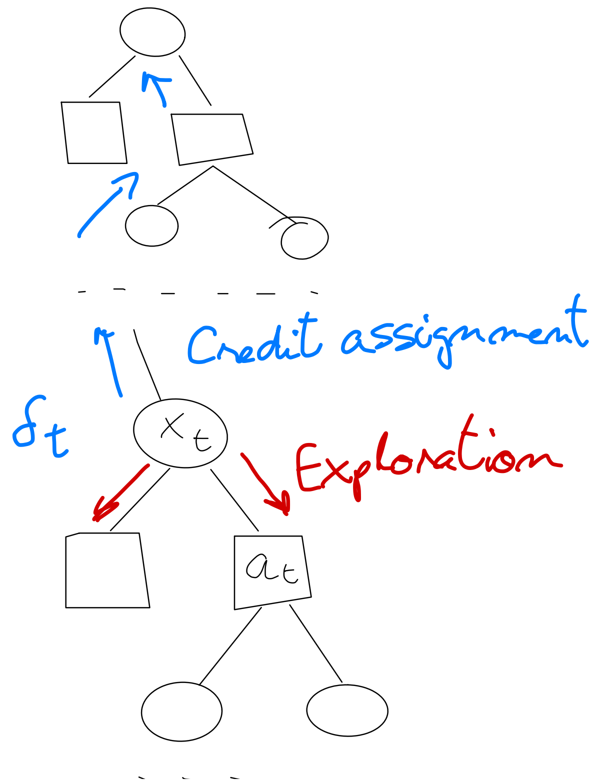
The simplest reinforcement learning problem

Recap: What is Reinforcement Learning?

- Agent-oriented learning—learning by interacting with an environment to achieve a goal
  - more realistic and ambitious than other kinds of machine learning
- Learning by trial and error, with only evaluative feedback (reward)
  - the kind of machine learning most like natural learning
  - learning that can tell for itself when it is right or wrong
- The beginnings of a *science of mind*

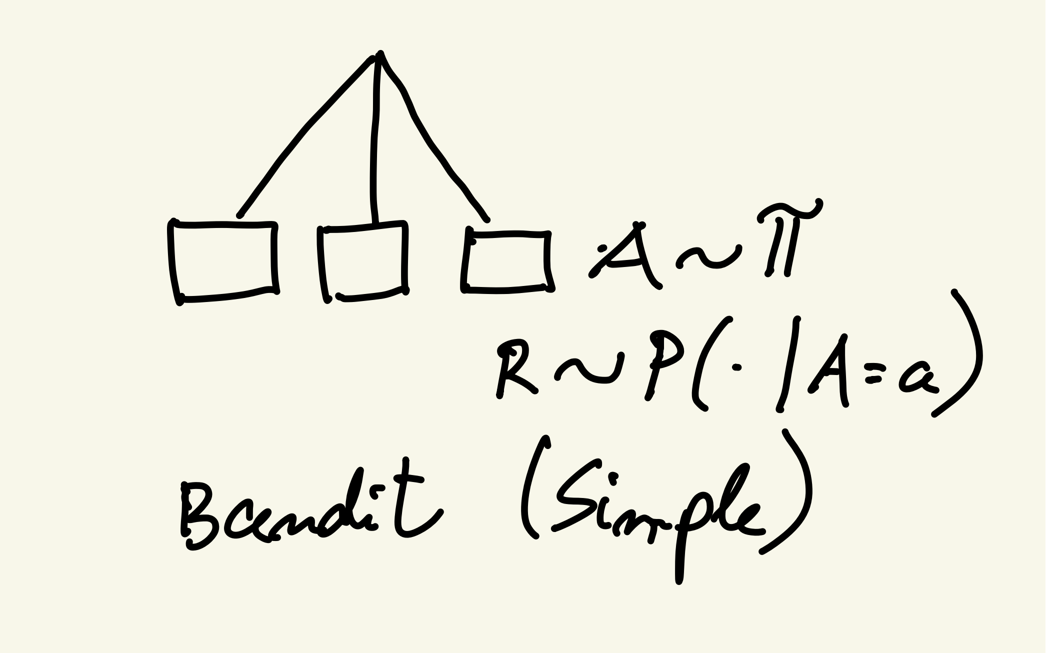
Recall: How to think about RL more formally?

- At time t, agent receives an observation from set $\mathcal{X}$ and can choose an action from set $\mathcal{A}$ (think finite for now)
- Goal of the agent is to maximize long-term return



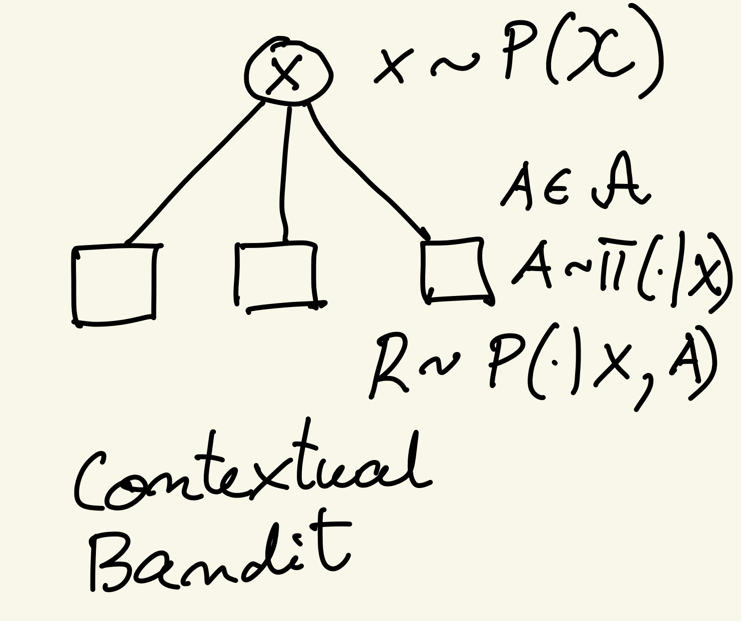
Simple case: Bandits

- No x, take an action, observe a reward immediately
- So, a degenerate tree (not truly sequential)
- This is what we call a simple (multi-arm) bandit problem
- Focus on exploration, not credit assignment



Contextual bandits

- There is an observation x (context) followed by action and immediate rewards
- Focus still on exploration
- Lots of applications in ad placement, more recently in large language models



Recall: Play a bandit exercise

- Imagine you have two actions
- You play action 1 and get a reward of 0
- You play action 2 and get a reward of 1
- Which action should you prefer?
- Which action should you try next?

Main Principles

- Optimize Expected Value
- Other criteria are possible, eg conditional value at risk (CVaR)
- Need to balance exploration (trying all actions) vs exploitation
- We cannot stop exploring!
- More data reduces uncertainty in the mean reward of each action

The *k*-armed Bandit Problem

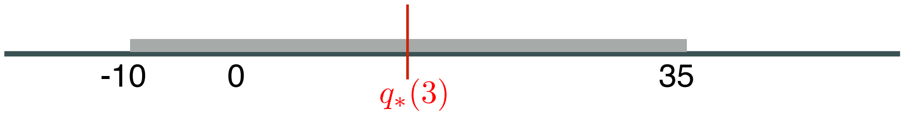
- On each of an infinite sequence of *time steps*, t=1, 2, 3, …, you choose an action A_t from k possibilities, and receive a real-valued *reward* R_t
- The reward depends only on the action taken; it is identically, independently distributed (i.i.d.):

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

- These true values are *unknown*. The distribution is unknown
- Nevertheless, you must maximize your total reward
- You must both try actions to learn their values (explore), and prefer those that appear best (exploit)

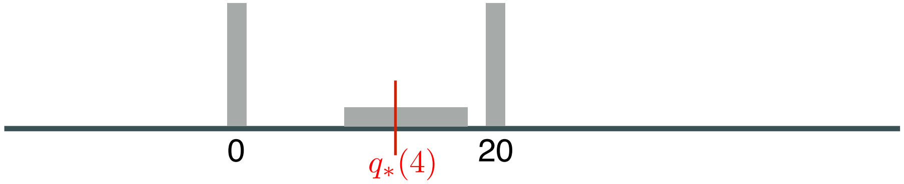
Action Values

- Action 1 — Reward is always 8
  - value of action 1 is $q_*(1) =$
- Action 2 — 88% chance of 0, 12% chance of 100!
  - value of action 2 is $q_*(2) = .88 \times 0 + .12 \times 100 =$
- Action 3 — Randomly between -10 and 35, equiprobable



$q_*(3) =$

- Action 4 — a third 0, a third 20, and a third from {8,9,…, 18}



$q_*(4) =$

Action-Value Estimation

- Learn and action-value estimate from sequence of rewards
- For example, estimate action values as *sample averages*:

$$Q_t(a) \doteq \frac{\text{sum of rewards when } a \text{ taken prior to } t}{\text{number of times } a \text{ taken prior to } t} = \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbf{1}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbf{1}_{A_i=a}}$$

- The sample-average estimates converge to the true values *If* the action is taken an infinite number of times

$$\lim_{N_t(a)\to\infty} Q_t(a) \;=\; q_*(a)$$

The number of times action a has been taken by time t

Averaging $\longrightarrow$ learning rule

- 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, then divide
- Anything more elegant?

Derivation of incremental update

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

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

Averaging $\longrightarrow$ learning rule

- 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:

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

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

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

Tracking a Non-stationary Problem

- Suppose the true action values change slowly over time
  - then we say that the problem is *non-stationary*
- In this case, sample averages are not a good idea (Why?)
- Better is an "exponential, recency-weighted average":

$$Q_{n+1} \doteq Q_n + \alpha\Big[R_n - Q_n\Big]$$
$$= (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i,$$

where α is a constant *step-size parameter*, $\alpha \in (0, 1]$

- There is bias due to Q_1 that becomes smaller over time

Standard stochastic approximation convergence conditions

- To assure convergence with probability 1:

$$\sum_{n=1}^{\infty} \alpha_n(a) = \infty \qquad \text{and} \qquad \sum_{n=1}^{\infty} \alpha_n^2(a) < \infty$$

- e.g., $\alpha_n \doteq \frac{1}{n}$
- not $\alpha_n \doteq \frac{1}{n^2}$

if $\alpha_n \doteq n^{-p}, \quad p \in (0, 1)$

then convergence is at the optimal rate:

$$O(1/\sqrt{n})$$

The Exploration/Exploitation Dilemma

- Suppose you form estimates

$$Q_t(a) \approx q_*(a), \quad \forall a$$ *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. Or maybe not.

ε-Greedy Action Selection

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

A simple bandit algorithm

Initialize, for $a = 1$ to k:

$Q(a) \leftarrow 0$

$N(a) \leftarrow 0$

Repeat forever:

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

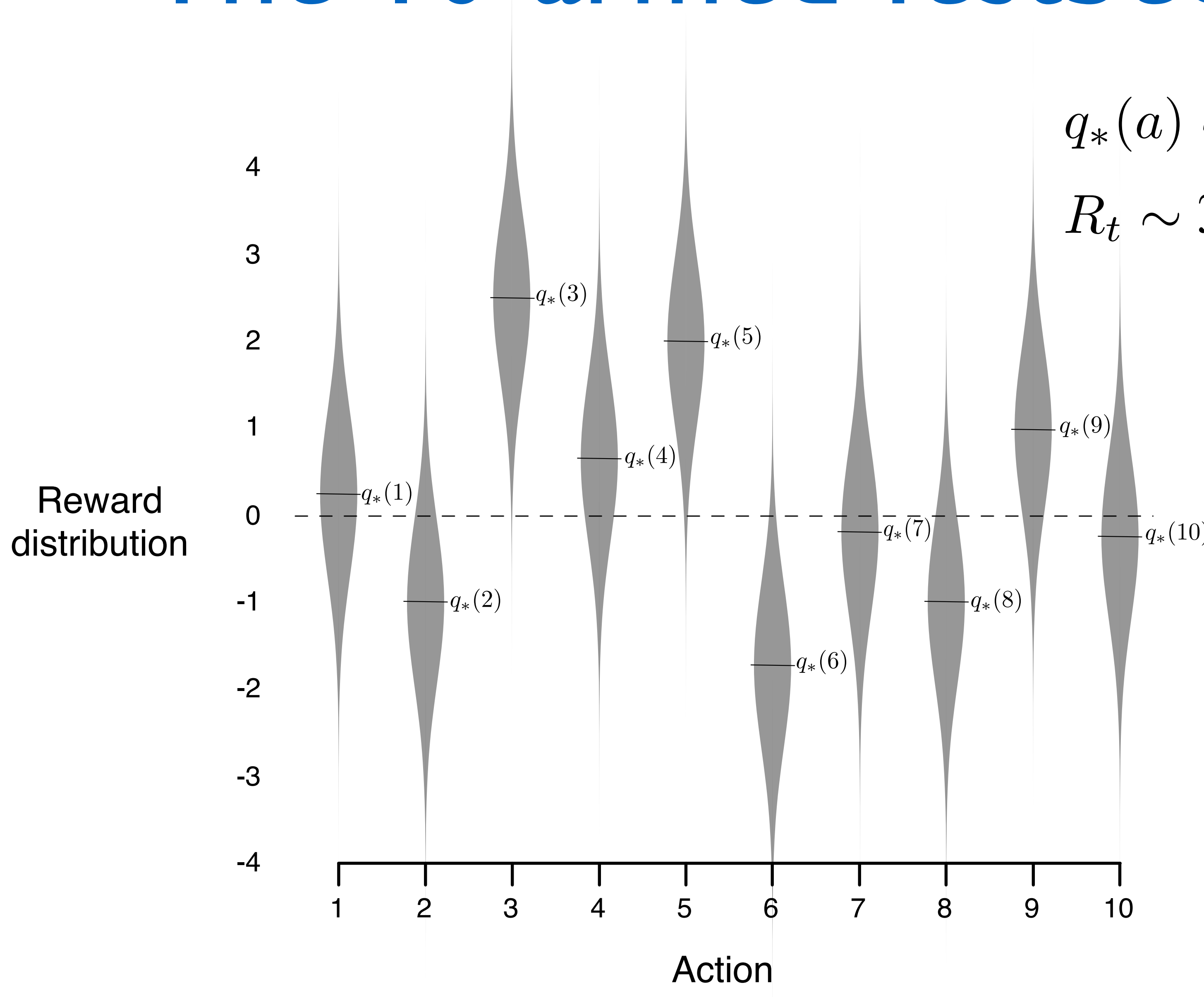
$R \leftarrow bandit(A)$

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

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

One Bandit Task from

The 10-armed Testbed



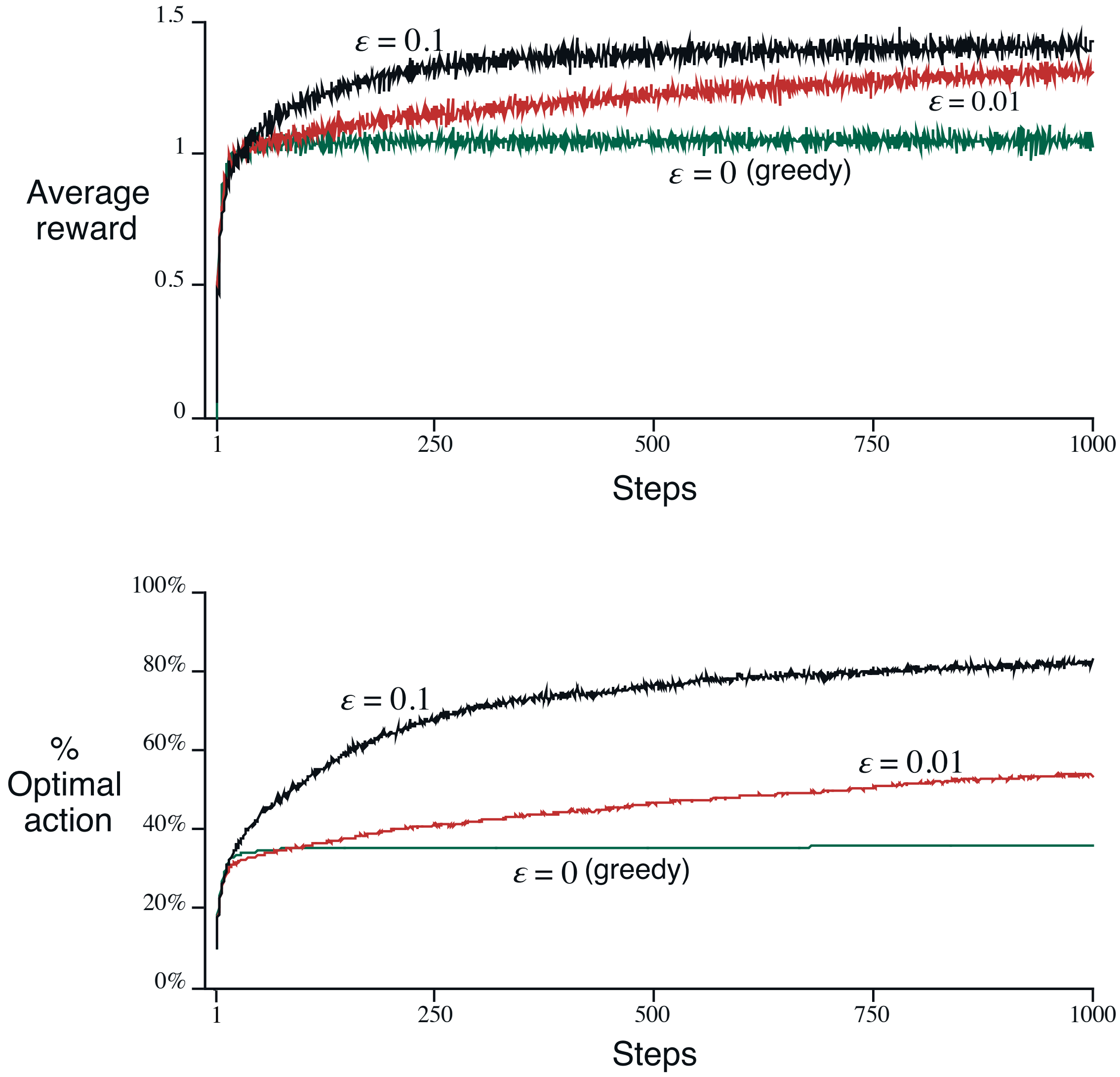
$$q_*(a) \sim \mathcal{N}(0, 1)$$

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

Run for 1000 steps

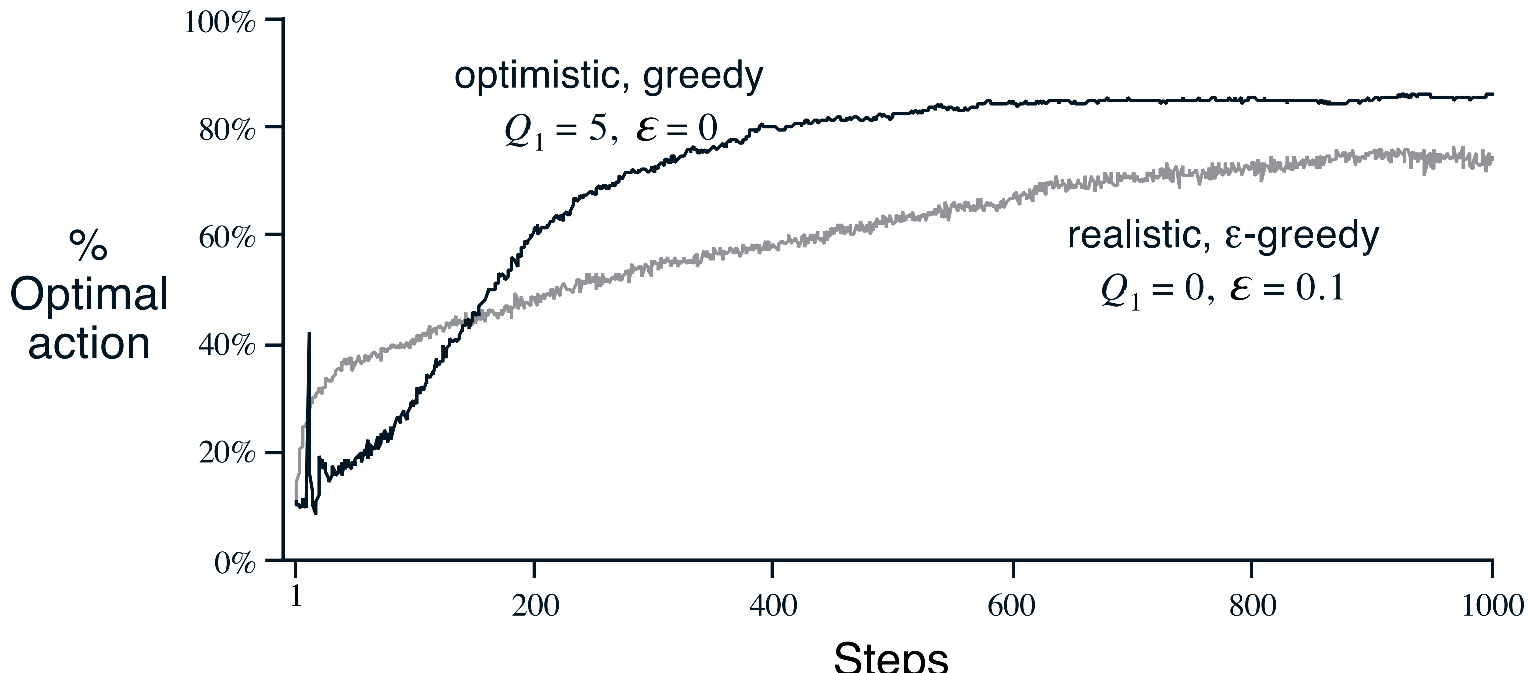
Repeat the whole thing 2000 times with different bandit tasks

ε-Greedy Methods on the 10-Armed Testbed



Optimistic Initial Values

- All methods so far depend on $Q_1(a)$, i.e., they are biased. So far we have used $Q_1(a) = 0$
- Suppose we initialize the action values *optimistically* ($Q_1(a) = 5$), e.g., on the 10-armed testbed (with $\alpha = 0.1$)



Upper Confidence Bound (UCB) action selection

- 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

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