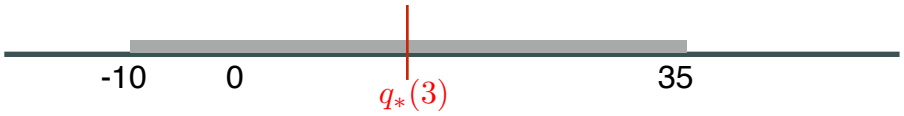
Multi-arm Bandits

Based on
Reinforcement Learning, Sutton and Barto, Chapter 2

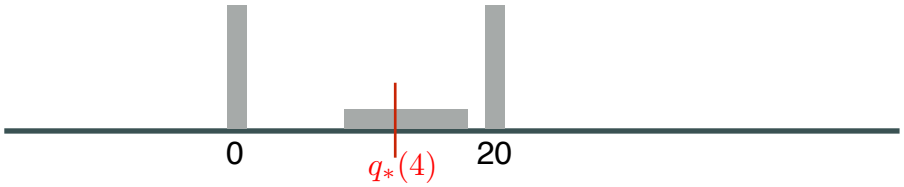
You are the algorithm!

- 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 $q_*(2) =$
- Action 3 — Randomly between -10 and 35, equiprobable



$q_*(3) =$

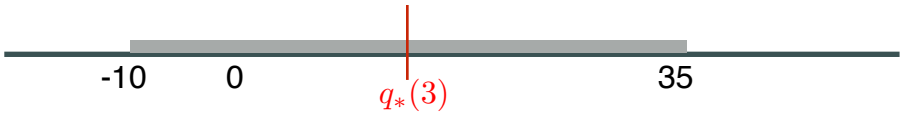
- Action 4 — a third 0, a third 20, and a third from {8,9,…, 18}



$q_*(4) =$

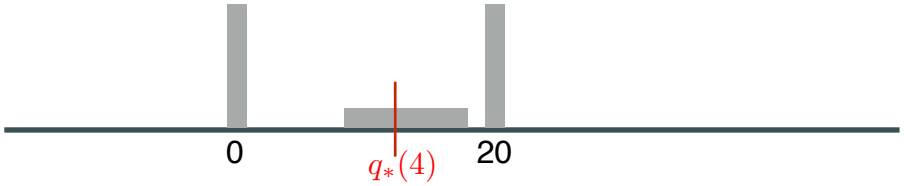
You are the algorithm! (bandit1)

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



$q_*(3) = 12.5$

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



$$q_*(4) = \frac{1}{3} \times 0 + \frac{1}{3} \times 20 + \frac{1}{3} \times 13 = 0 + \frac{20}{3} + \frac{13}{3} = \frac{33}{3} = 11$$

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)

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.

Action-Value Methods

- Methods that learn action-value estimates and nothing else
- 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

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:

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

$R \leftarrow bandit(A)$

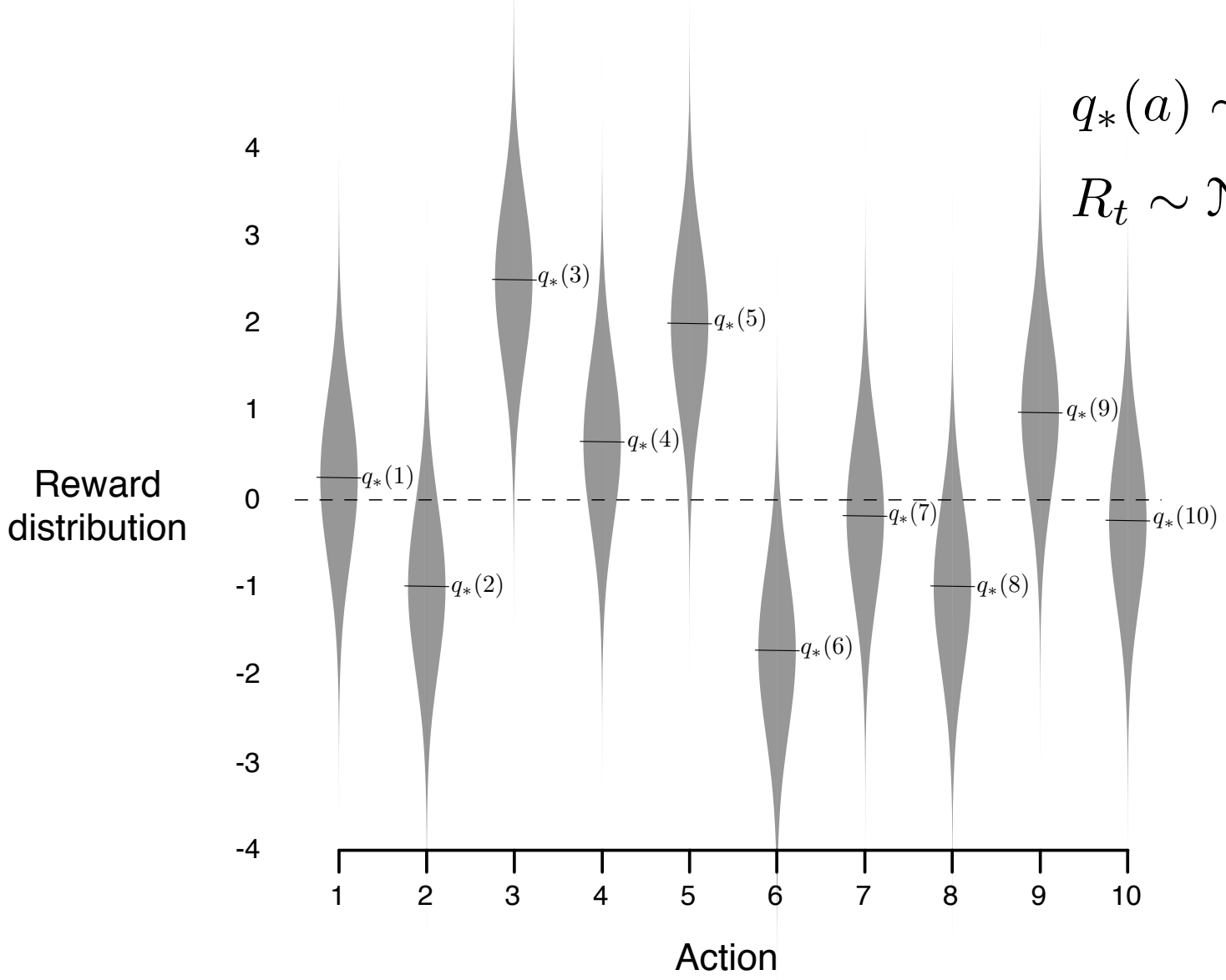
$N(A) \leftarrow N(A) + 1$

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

The 10-armed Testbed

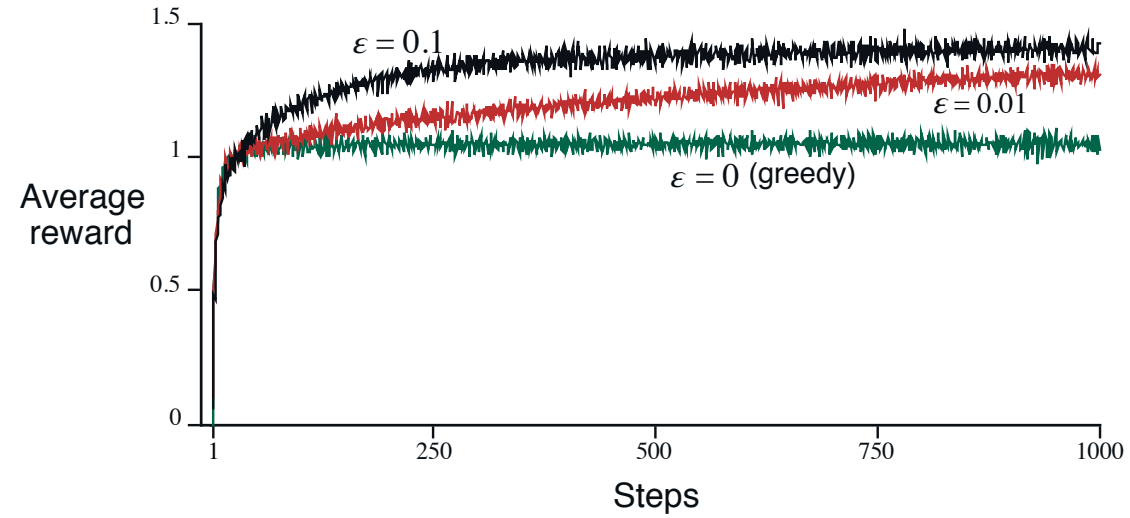
$$q_*(a) \sim \mathcal{N}(0, 1)$$

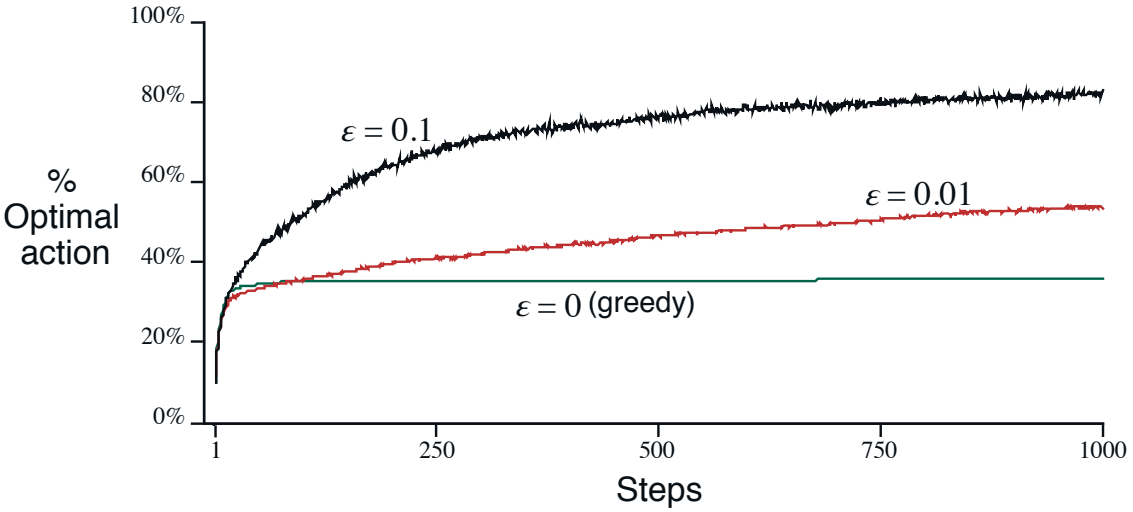
$$R_t \sim \mathcal{N}(q_*(a), 1)$$



2000 independent bandit tasks

ε-Greedy Methods on the 10-Armed Testbed



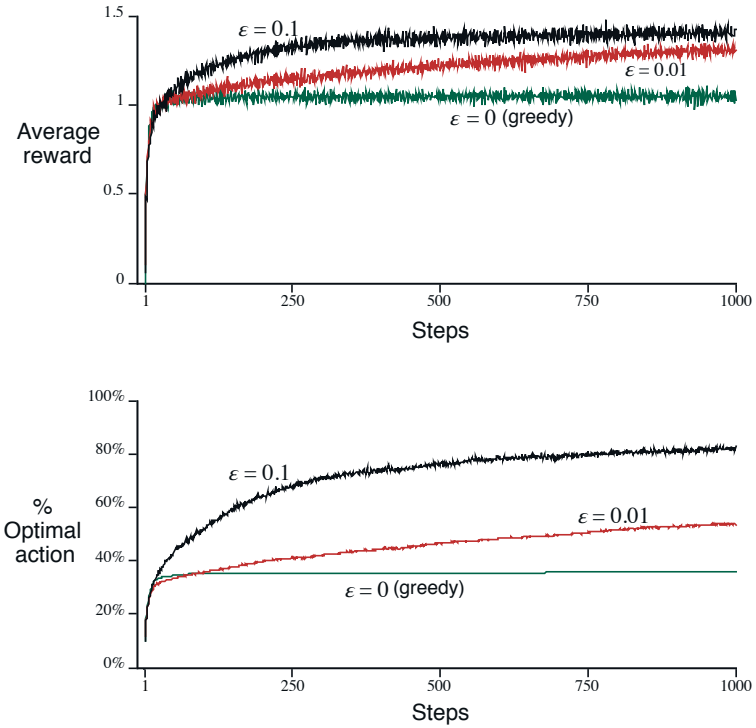


Quiz

- In ε-greedy action selection, for the case of two actions and $\varepsilon = 0.5$, what is the probability that the greedy action is selected?

Quiz

- Based on these Figures, which method performs better in the long run if the reward variance had been larger, say 10 instead of 1? Which one likely performs better if reward variance was 0?



Learning Action Values

- 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:

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

Derivation of Incremental Update

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

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

Tracking a Non-stationary Problem

- Suppose the true action values change slowly over time, then we say that the problem is *non-stationary*
  - *Stationary:* reward probabilities do not change over time.
- In this case, sample averages are not a good idea. It makes sense to give more weight to recent rewards than to long-past rewards.
- Better is an "exponential, recency-weighted average":

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

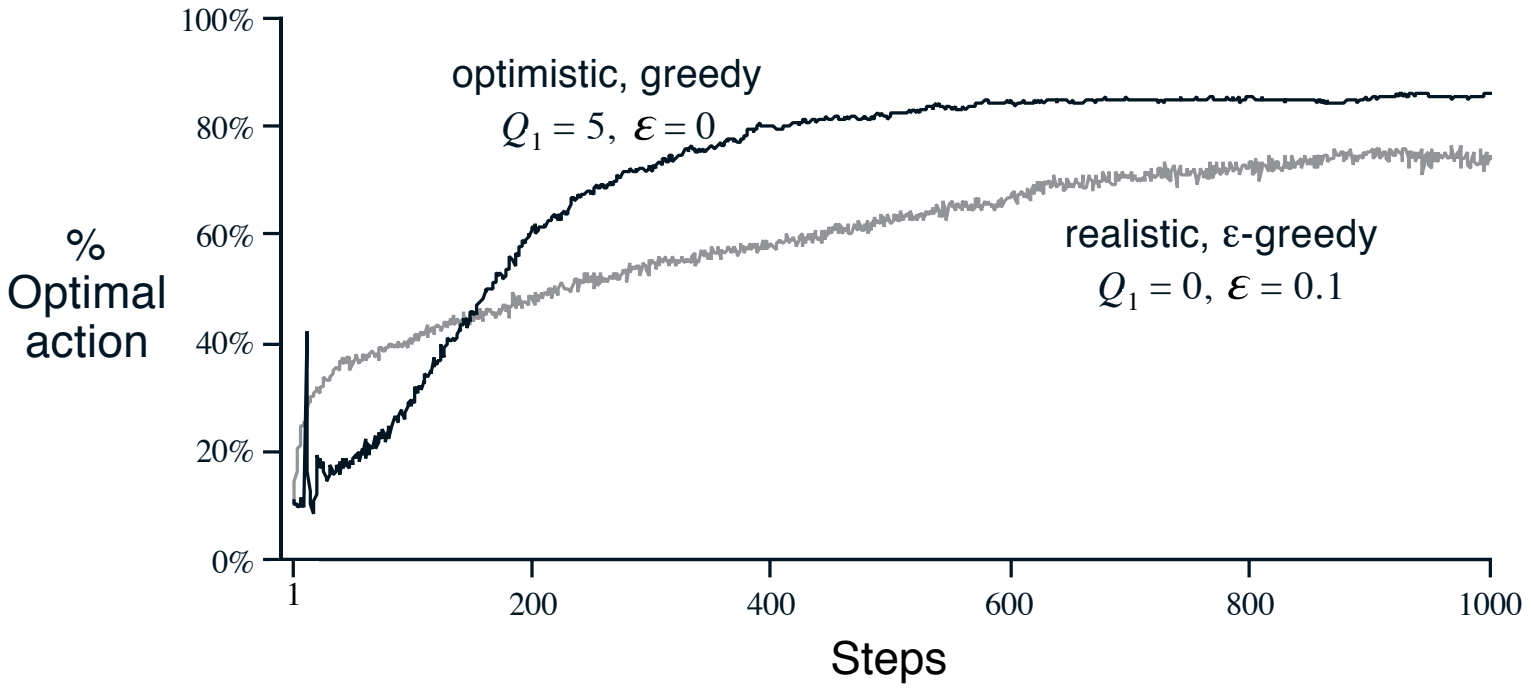
where α is a constant *step-size parameter*, $\alpha \in (0, 1]$

- There is bias due to Q_1 that becomes smaller over time.
- Initial reward values become an easy values to provide prior knowledge.

Optimistic Initial Values

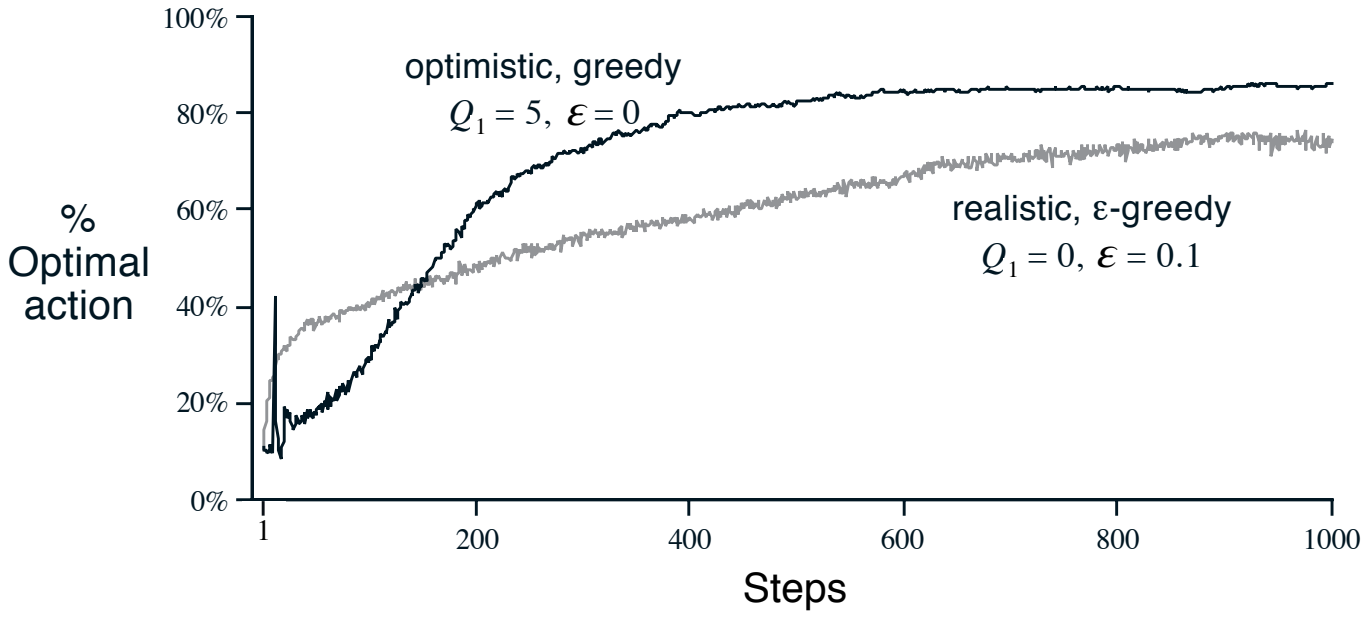
- All methods so far depend on $Q_1(a)$, i.e., they are biased.
  So far we have used $Q_1(a) = 0$

- Suppose we initialize the action values *optimistically* ($Q_1(a) = 5$),
  e.g., on the 10-armed testbed (with $\alpha = 0.1$)



Quiz

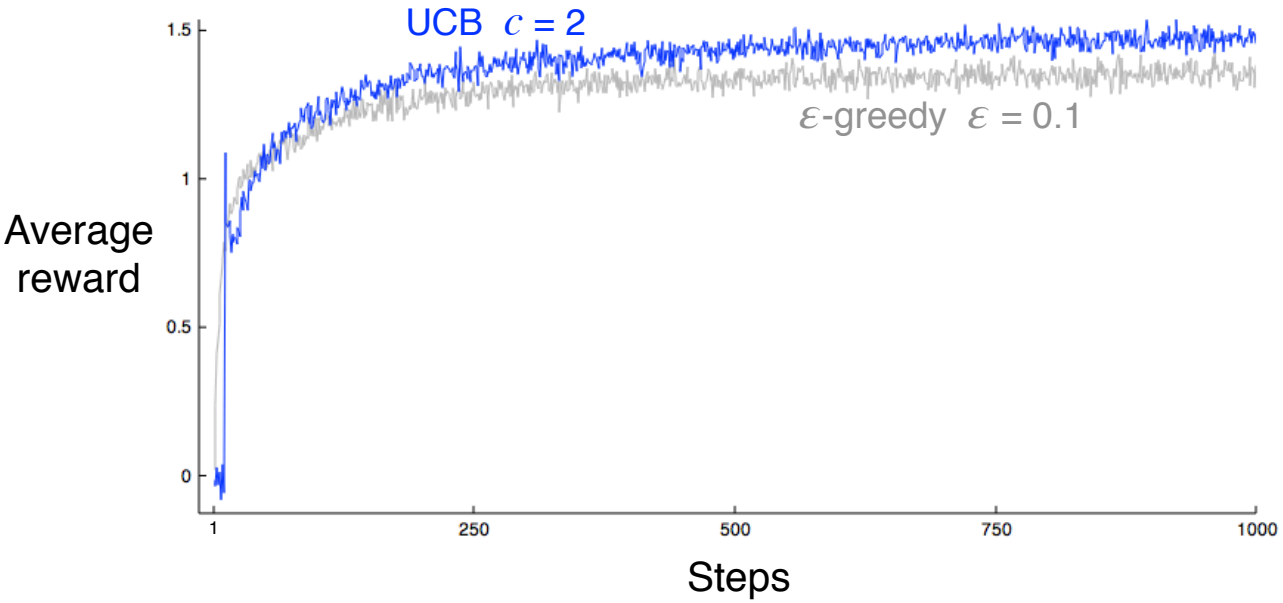
- Initially, the optimistic method performs worse. Why?



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: look at potential of actions for actually being optimal.

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



Gradient-Bandit Algorithms

- Let $H_t(a)$ be a learned *preference* for taking action a. The larger the preference, the more often that action is taken, but the preference has no interpretation in terms of reward.

$$\Pr\{A_t = a\} \doteq \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} \doteq \pi_t(a)$$

$$H_{t+1}(a) \doteq H_t(a) + \alpha\big(R_t - \bar{R}_t\big)\big(\mathbf{1}_{a=A_t} - \pi_t(a)\big), \qquad \forall a,$$

$$\bar{R}_t = \frac{1}{t-1}\Sigma_{i=1}^{t-1} R_i$$

- If the reward is higher than the baseline, then the probability of taking a in the future is increased, and if the reward is below baseline, then the probability is decreased. The non-selected actions move in the opposite direction.

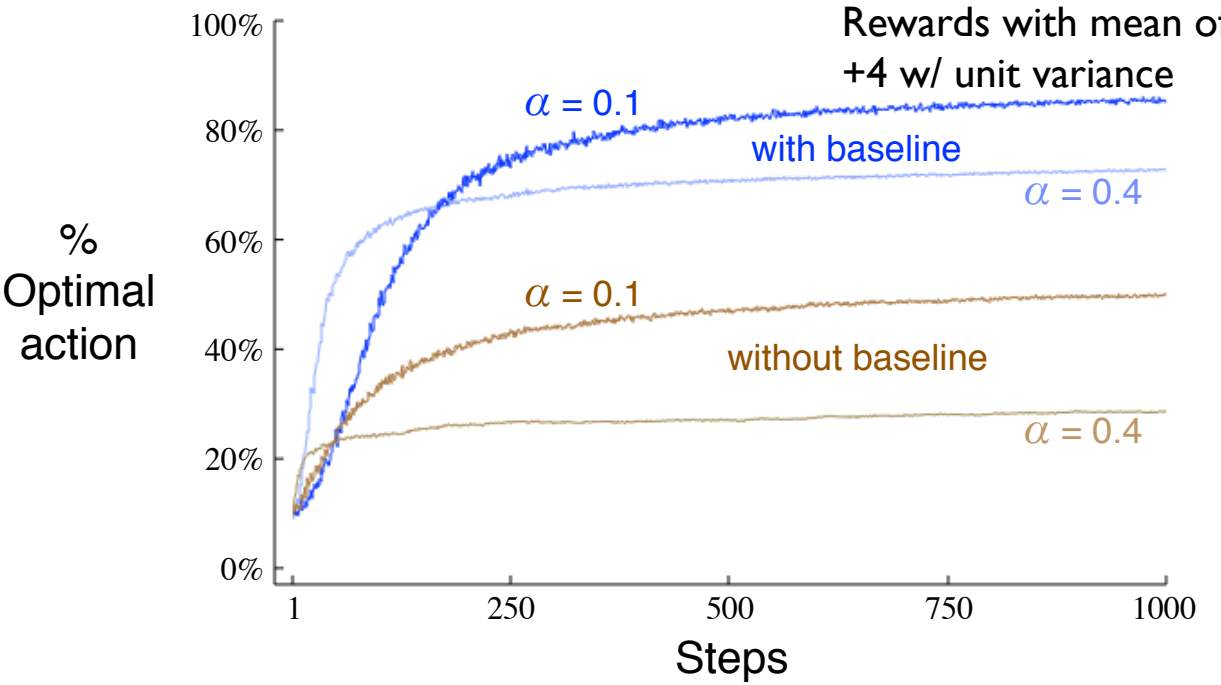
Gradient-Bandit Algorithms

- Let $H_t(a)$ be a learned *preference* for taking action a. The larger the preference, the more often that action is taken, but the preference has no interpretation in terms of reward.

$$\Pr\{A_t = a\} \doteq \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} \doteq \pi_t(a)$$

$$H_{t+1}(a) \doteq H_t(a) + \alpha\big(R_t - \bar{R}_t\big)\big(\mathbf{1}_{a=A_t} - \pi_t(a)\big), \qquad \forall a,$$

$$\bar{R}_t = \frac{1}{t-1}\Sigma_{i=1}^{t-1} R_i$$



Contextual Bandit

- So far, no need to associate different actions with different situations.
    - find a single best action when stationary or track the best action as it changes over time when the task is non-stationary.
- Contextual bandit: find a mapping from situations to the actions.
    - There are several different k-armed bandit tasks, and on each step you confront one of these chosen at random.

Summary Comparison of Bandit Algorithms