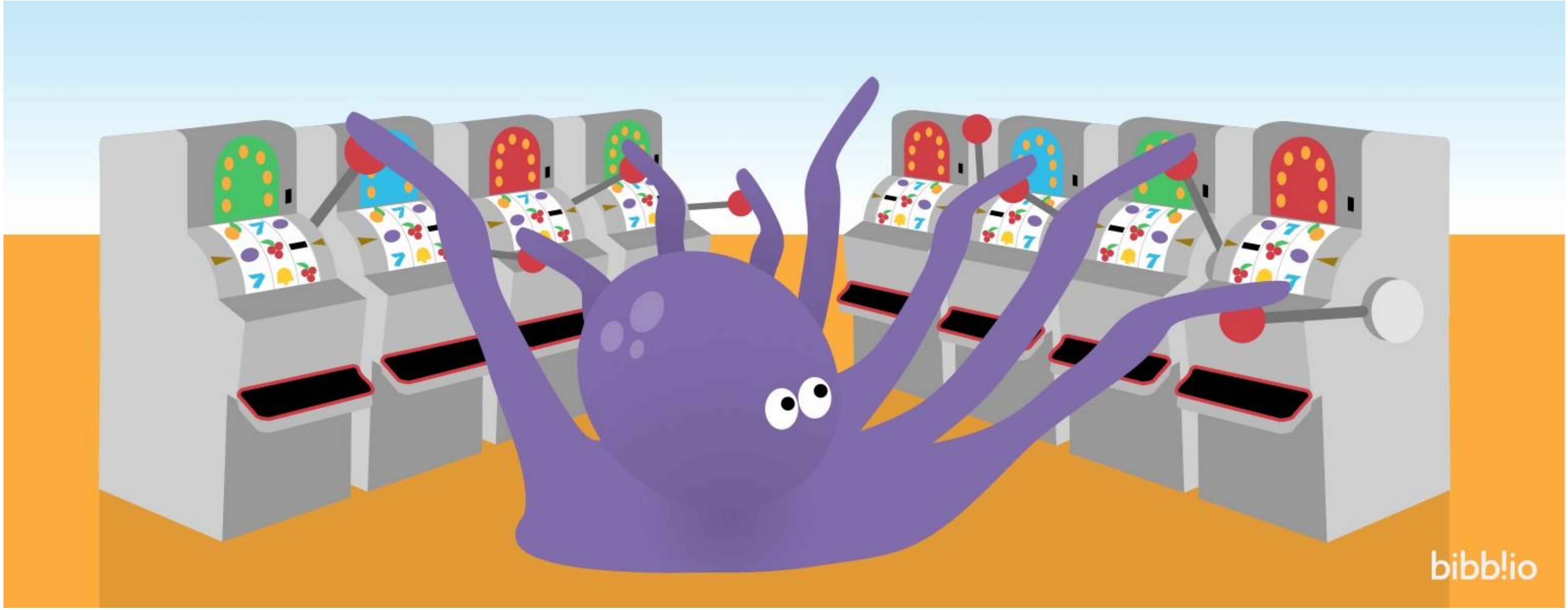
Multi-armed Bandits

Prof. Kuan-Ting Lai

2020/3/12

k-armed Bandit Problem

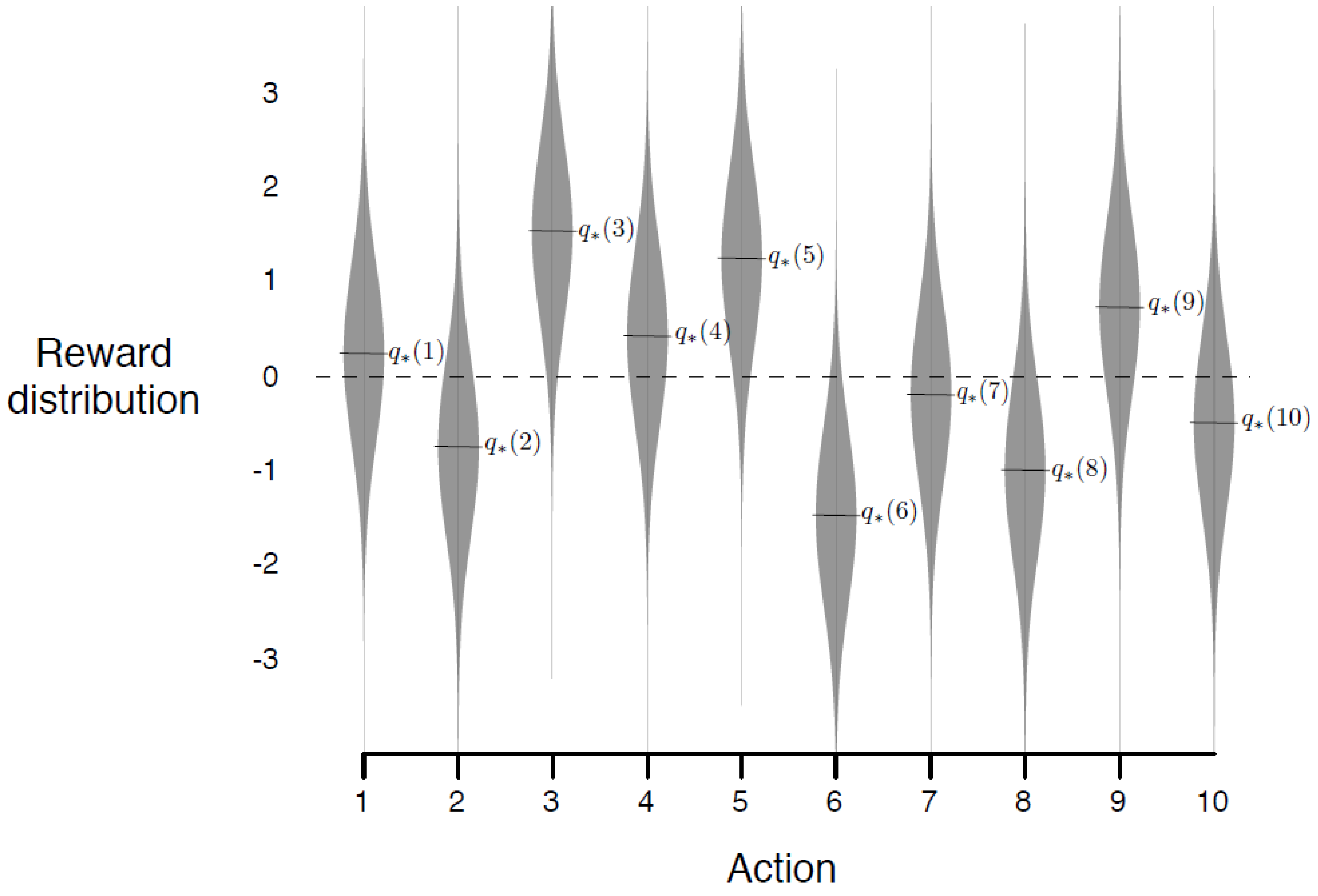
- Playing k armed bandit machines and find a way to win most money!
- Note: assume you have unlimited money and never go bankrupt!



https://towardsdatascience.com/reinforcement-learning-multi-arm-bandit-implementation-5399ef67b24b

10-armed Testbed

- Each bandit machine has its own reward distribution



Action-value Function

- $Q_t(a)$: The estimated value (reward) of action *a* at time *t*

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

- Let $q_*(a)$ be the true (optimal) action-value function

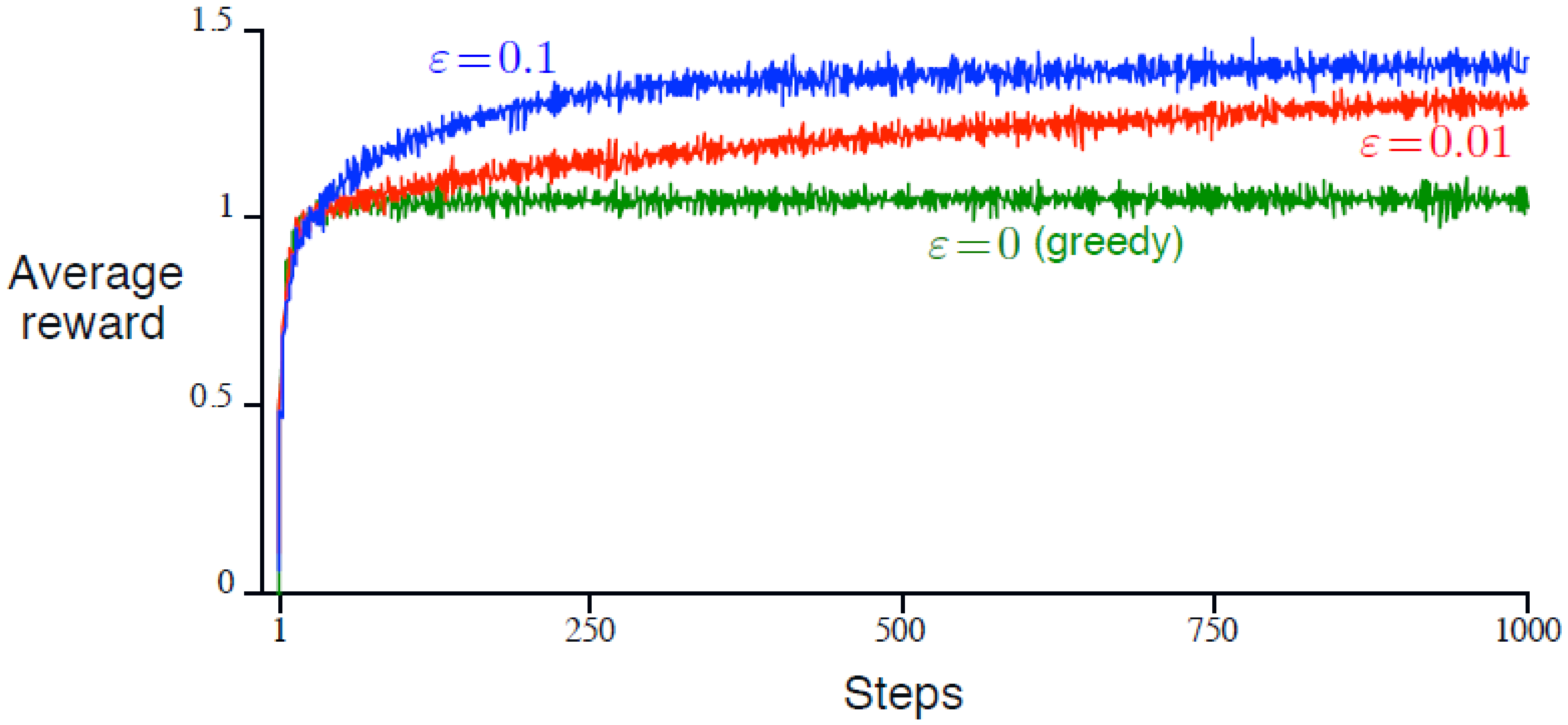
$$q_*(a) \leftarrow E[R_t | A_t = a]$$

ε-greedy

- Greedy action
  - Always select the action with max value
  - $A_t \leftarrow \underset{a}{\operatorname{argmax}}\, Q_t(a)$
- ε-greedy
  - Select the greedy action (1- ε) of the time, select random actions ε of the time

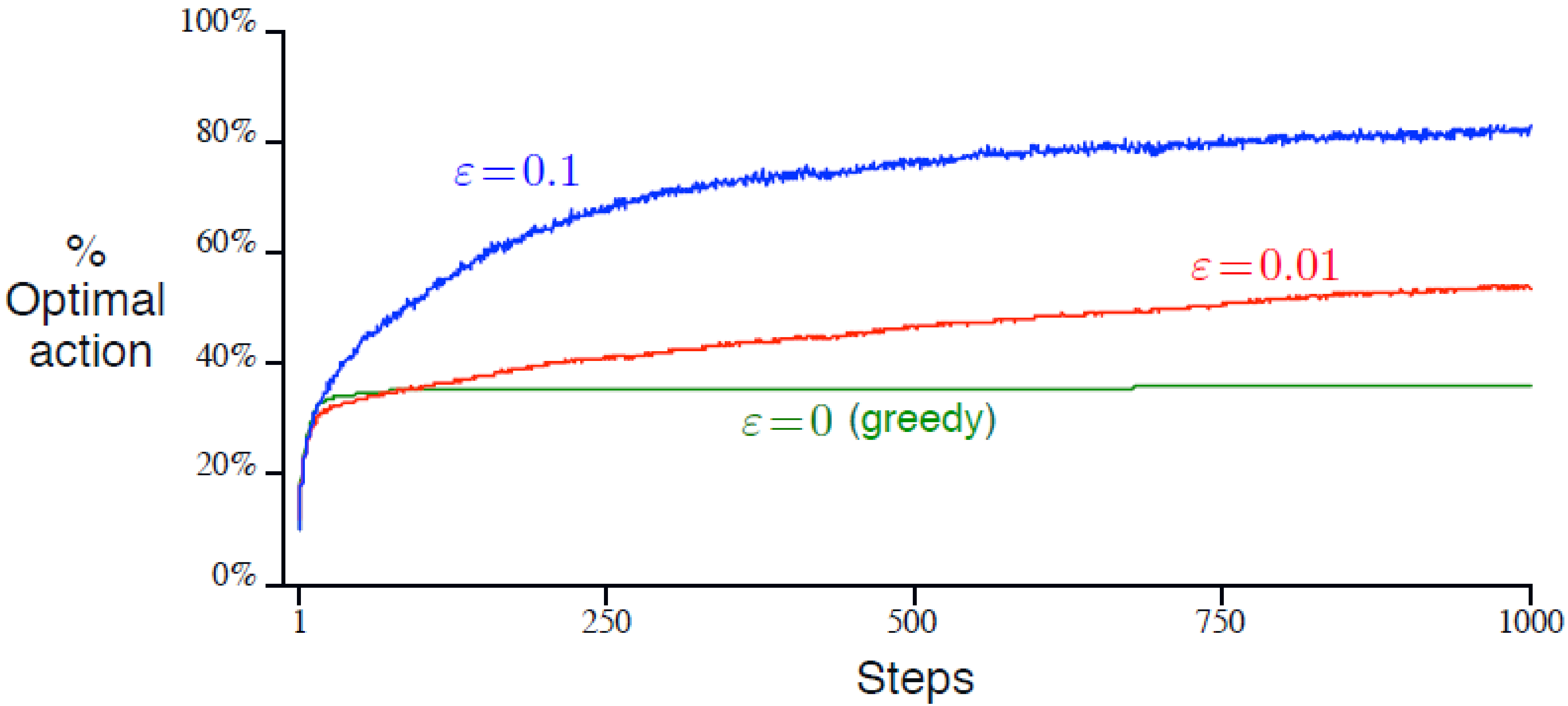
Performance of ε-greedy

- Average rewards over 2000 runs with ε=0, 0.1, 0.01



Optimal Actions Selected by ε-greedy

- Optimal actions selected over 2000 runs with ε=0, 0.1, 0.01



Update $Q_t(a)$

- let Q_n denote the estimate of its action value after it has been selected n − 1 times

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

Deriving Update Rule

- Require only memory of Qn and Rn

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

Tracking a Nonstationary Problem

- Using constant step-size $\alpha \in (0,1]$
- Constant step-size doesn't converge

$$
\begin{aligned}
Q_{n+1} &= Q_n + \alpha\Big[R_n - Q_n\Big] \\
&= \alpha R_n + (1-\alpha)Q_n \\
&= \alpha R_n + (1-\alpha)\left[\alpha R_{n-1} + (1-\alpha)Q_{n-1}\right] \\
&= \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2 Q_{n-1} \\
&= \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2 \alpha R_{n-2} + \\
&\qquad \cdots + (1-\alpha)^{n-1}\alpha R_1 + (1-\alpha)^n Q_1 \\
&= (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i.
\end{aligned}
$$

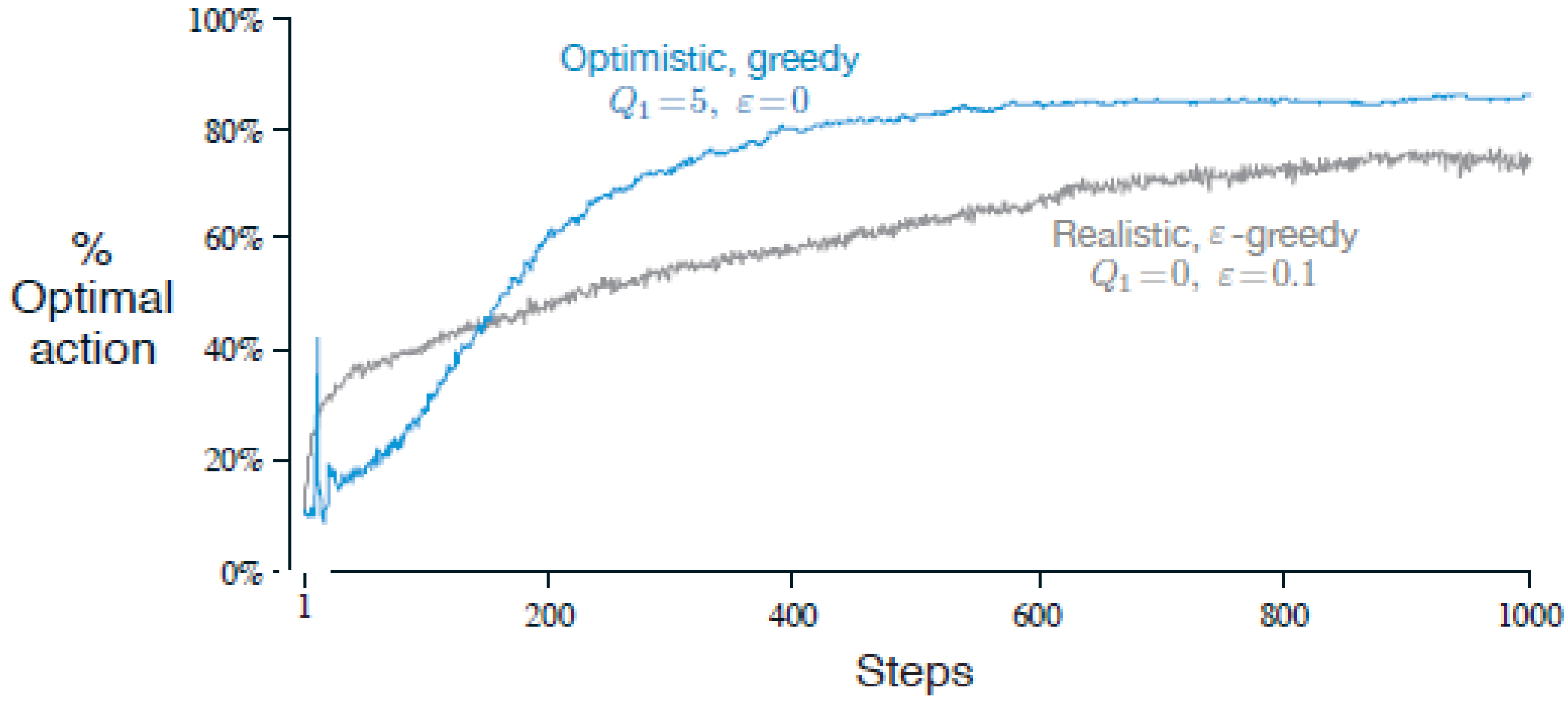
Exponential Recency-weighted Average

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

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

Optimistic Initial Values

- We should not care about initial value too much in practice



Upper-Confidence-Bound Action Selection

- $N_t(a)$: Number of times that action a has been selected prior to time t
- Not practical for large state spaces

$$A \leftarrow \arg\max_a \left[Q_t(a) + c \sqrt{\frac{\ln t}{N_t(a)}} \right]$$

Gradient Bandit Algorithms

- Soft-max function
- $\pi_t(a)$ is the probability of taking action *a* at time *t*

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

Selecting Actions based on $\pi_t(a)$

$$
\begin{aligned}
H_{t+1}(A_t) &\doteq H_t(A_t) + \alpha\big(R_t - \bar{R}_t\big)\big(1 - \pi_t(A_t)\big), && \text{and} \\
H_{t+1}(a) &\doteq H_t(a) - \alpha\big(R_t - \bar{R}_t\big)\pi_t(a), && \text{for all } a \neq A_t,
\end{aligned}
$$

Gradient Ascent

$$H_{t+1}(a) \doteq H_t(a) + \alpha \frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)}.$$

$$\mathbb{E}[R_t] = \sum_x \pi_t(x) q_*(x)$$

Calculating Gradient

- Adding a baseline B

$$\frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)} = \frac{\partial}{\partial H_t(a)} \left[\sum_x \pi_t(x) q_*(x) \right]$$

$$= \sum_x q_*(x) \frac{\partial \pi_t(x)}{\partial H_t(a)}$$

$$= \sum_x \big(q_*(x) - B_t\big) \frac{\partial \pi_t(x)}{\partial H_t(a)}$$

Convert Equation into Expectation

- Multiplied by $\pi_t(x)/\pi_t(x)$
- Choose baseline $B_t = \overline{R_t}$

$$\frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)} = \sum_x \pi_t(x)\big(q_*(x) - B_t\big)\frac{\partial \pi_t(x)}{\partial H_t(a)}/\pi_t(x)$$

$$= \mathbb{E}\left[\big(q_*(A_t) - B_t\big)\frac{\partial \pi_t(A_t)}{\partial H_t(a)}/\pi_t(A_t)\right]$$

$$= \mathbb{E}\left[\big(R_t - \bar{R}_t\big)\frac{\partial \pi_t(A_t)}{\partial H_t(a)}/\pi_t(A_t)\right],$$

Calculating Gradient of Softmax

$$
\begin{aligned}
\frac{\partial \pi_t(x)}{\partial H_t(a)} &= \frac{\partial}{\partial H_t(a)} \pi_t(x) \\
&= \frac{\partial}{\partial H_t(a)} \left[\frac{e^{H_t(x)}}{\sum_{y=1}^{k} e^{H_t(y)}} \right] \\
&= \frac{\frac{\partial e^{H_t(x)}}{\partial H_t(a)} \sum_{y=1}^{k} e^{H_t(y)} - e^{H_t(x)} \frac{\partial \sum_{y=1}^{k} e^{H_t(y)}}{\partial H_t(a)}}{\left(\sum_{y=1}^{k} e^{H_t(y)}\right)^2} \qquad \text{(by the quotient rule)} \\
&= \frac{\mathbb{1}_{a=x} e^{H_t(x)} \sum_{y=1}^{k} e^{H_t(y)} - e^{H_t(x)} e^{H_t(a)}}{\left(\sum_{y=1}^{k} e^{H_t(y)}\right)^2} \qquad \text{(because } \tfrac{\partial e^x}{\partial x} = e^x\text{)} \\
&= \frac{\mathbb{1}_{a=x} e^{H_t(x)}}{\sum_{y=1}^{k} e^{H_t(y)}} - \frac{e^{H_t(x)} e^{H_t(a)}}{\left(\sum_{y=1}^{k} e^{H_t(y)}\right)^2} \\
&= \mathbb{1}_{a=x} \pi_t(x) - \pi_t(x) \pi_t(a) \\
&= \pi_t(x) \left(\mathbb{1}_{a=x} - \pi_t(a)\right). \qquad \text{Q.E.D.}
\end{aligned}
$$

Final Result

- Gradient bandit algorithm = gradient of expected reward!

$$\begin{aligned}\mathbb{E}\left[(R_t - \bar{R}_t)\frac{\partial \pi_t(A_t)}{\partial H_t(a)}/\pi_t(A_t)\right] &= \mathbb{E}\big[(R_t - \bar{R}_t)\pi_t(A_t)\big(\mathbb{1}_{a=A_t} - \pi_t(a)\big)/\pi_t(A_t)\big] \\ &= \mathbb{E}\big[(R_t - \bar{R}_t)\big(\mathbb{1}_{a=A_t} - \pi_t(a)\big)\big].\end{aligned}$$

$$\Downarrow$$

$$H_{t+1}(a) = H_t(a) + \alpha\big(R_t - \bar{R}_t\big)\big(\mathbb{1}_{a=A_t} - \pi_t(a)\big)$$

Reference

- Chapter 2, Richard S. Sutton and Andrew G. Barto, “Reinforcement Learning: An Introduction,” 2nd edition, Nov. 2018