

# Chapter 2: Multi-armed Bandits

Ryan Lee

# One-armed Bandit

- Slot machine
- Each spin (action) is independent



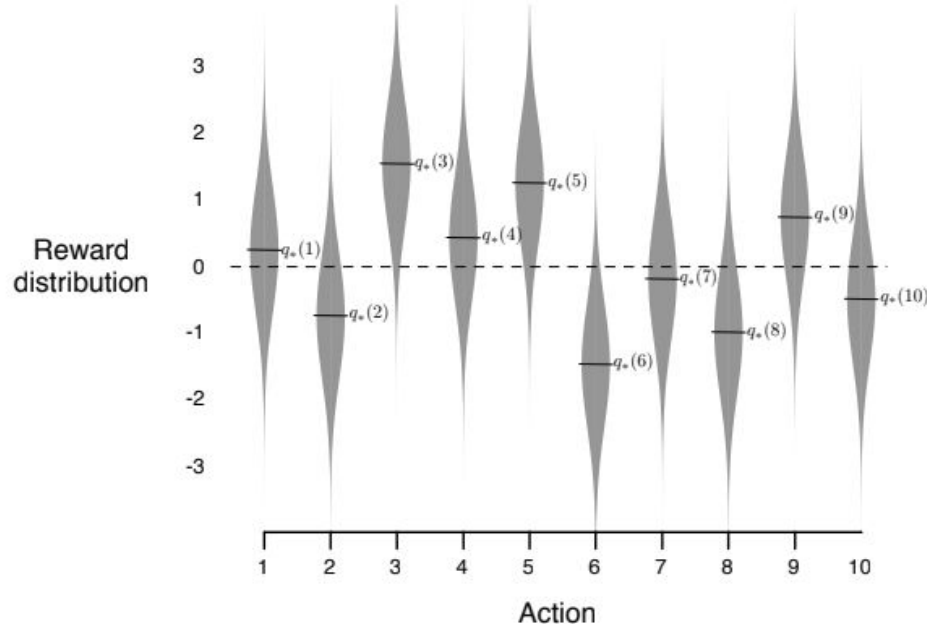
# Multi-armed Bandit problem

- Multiple slot machines to choose from
- Simplified setting to avoid complexities of RL problems
  - No observation
  - Action does not have delayed effect



# 10-armed Testbed

- 10 actions, 10 reward distributions
- Reward  $R_t$  chosen from stationary probability distributions



# Expected Reward

- Knowing expected reward trivializes the problem
- Estimate  $q_*(a)$  with  $Q_t(a)$

$$q_*(a) \doteq \mathbb{E}(R_t \mid A_t = a)$$

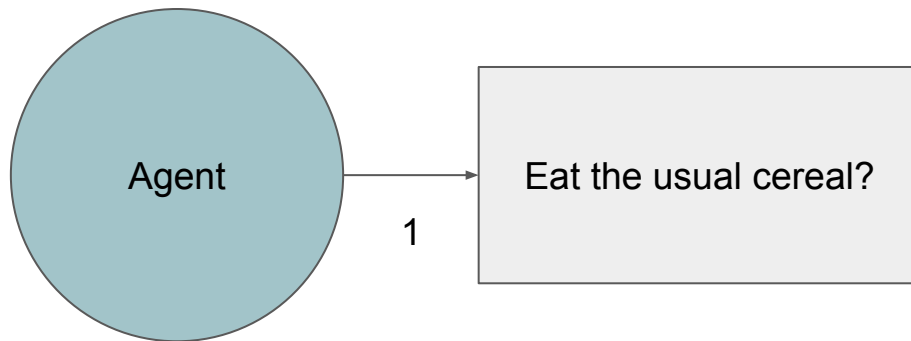
# Sample-average

- Estimate  $q_*(a)$  by averaging received rewards
- Default value (ex. 0) if action was never selected
- $Q_t(a)$  converges to  $q_*(a)$  as denominator goes to infinity

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

# Greedy method

- Always select *greedily*:  $A_t \doteq \operatorname{argmax}_a Q_t(a)$
- No exploration
- Often stuck in suboptimal actions



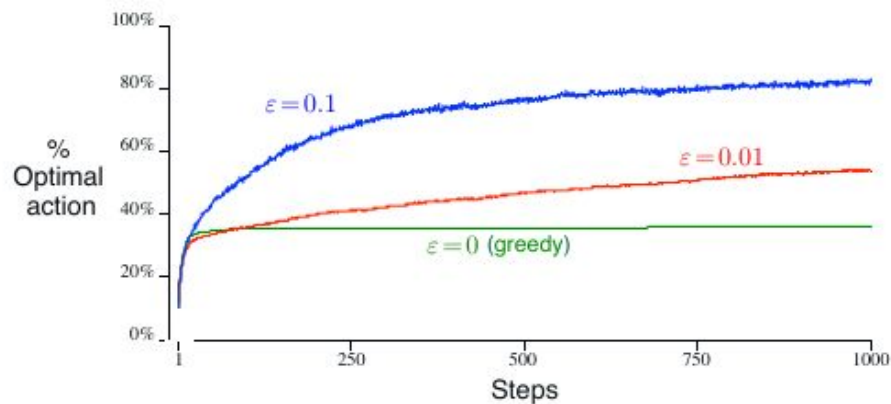
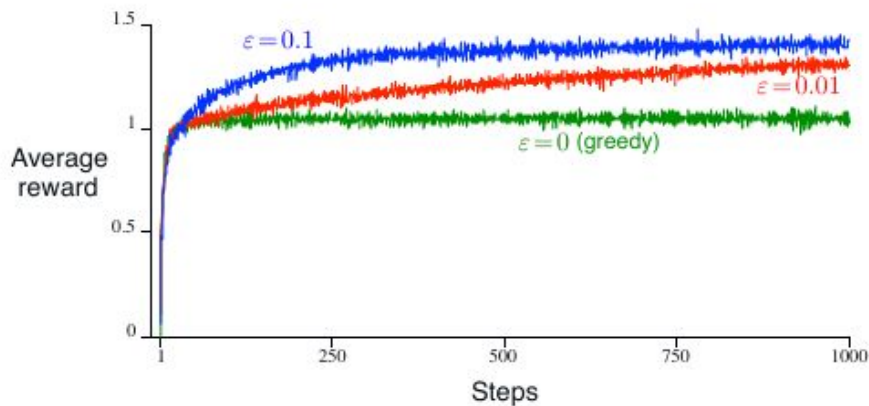
# $\epsilon$ -greedy method

- Select random action with probability  $\epsilon$
- All  $Q_t(a)$  converges to  $q_*(a)$  as denominator goes to infinity





# Greedy vs. $\epsilon$ -greedy



# Incremental Implementation

- Don't store reward for each step

$$Q_{n+1} = \frac{R_1 + R_2 + \dots + R_n}{n}$$

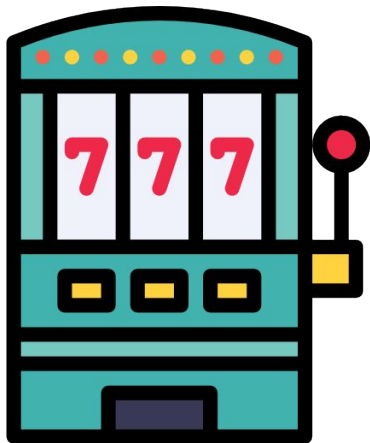
- Compute incrementally

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

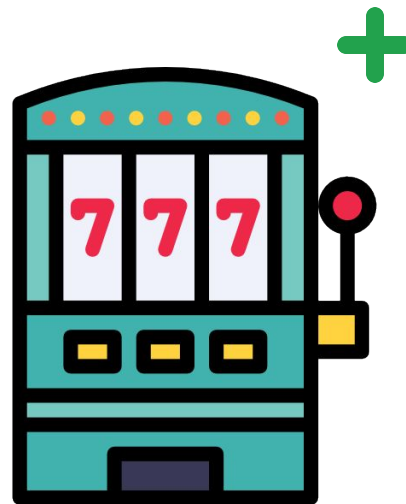
$$NewEstimate \leftarrow OldEstimate + StepSize [Target - OldEstimate]$$

# Nonstationary problem

- $q_*(a)$  changes over time
- Want to give new experience more weight



$$q_*(A_1 = a) \sim N(0, 1)$$



$$q_*(A_2 = a) \sim N(3, 1)$$

# Exponentially weighted average

- Constant step-size parameter  $\alpha$
- Give more weight to recent rewards

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

## Sample-average

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

- Guaranteed convergence
- Converge slowly: need tuning
- Seldom used in applications

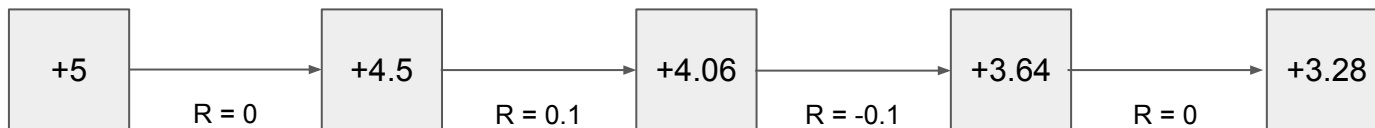
## Weighted average

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

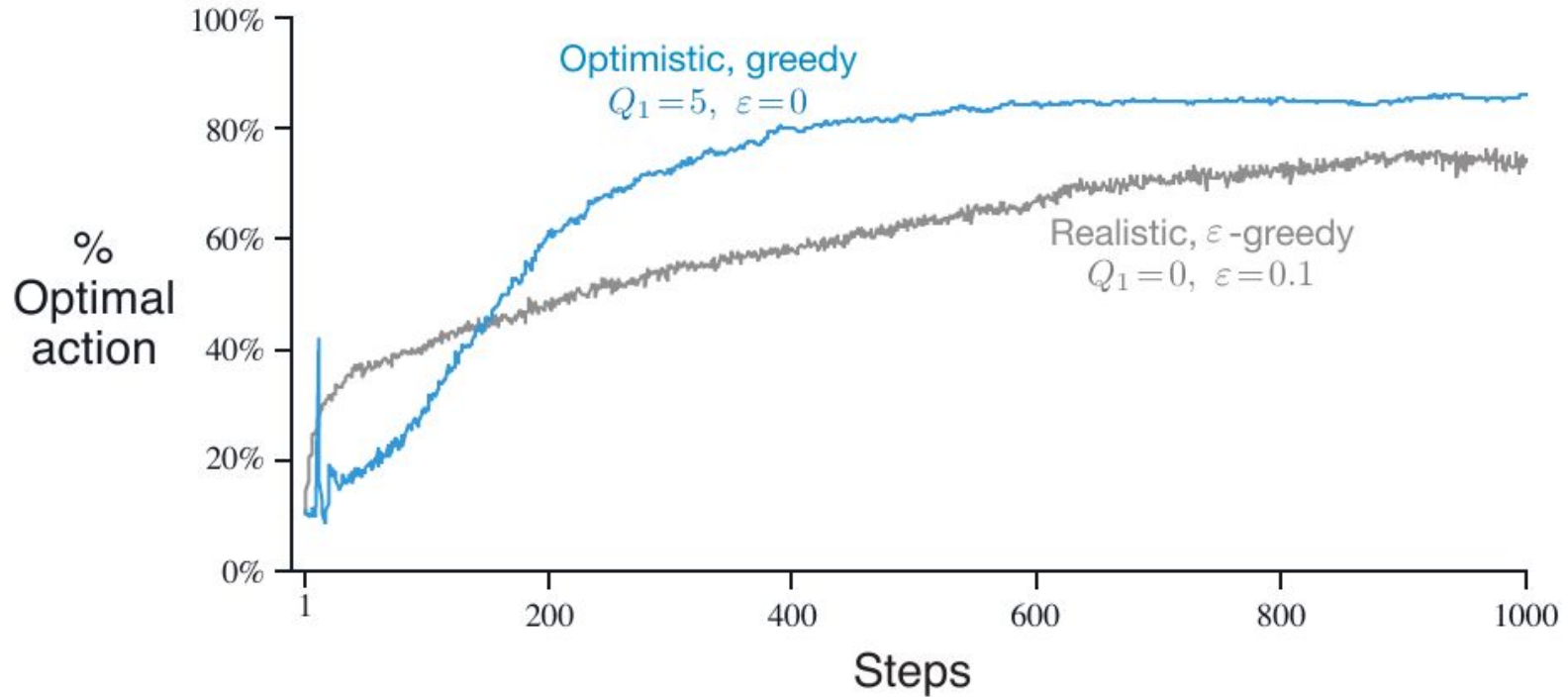
- Never completely converges
- Desirable in nonstationary problems

# Optimistic Initial Values

- Set optimistic initial action values to (ex. +5)
- Temporarily encourage exploration
- Doesn't work in nonstationary problems



# Optimistic Greedy vs. Realistic $\epsilon$ -greedy



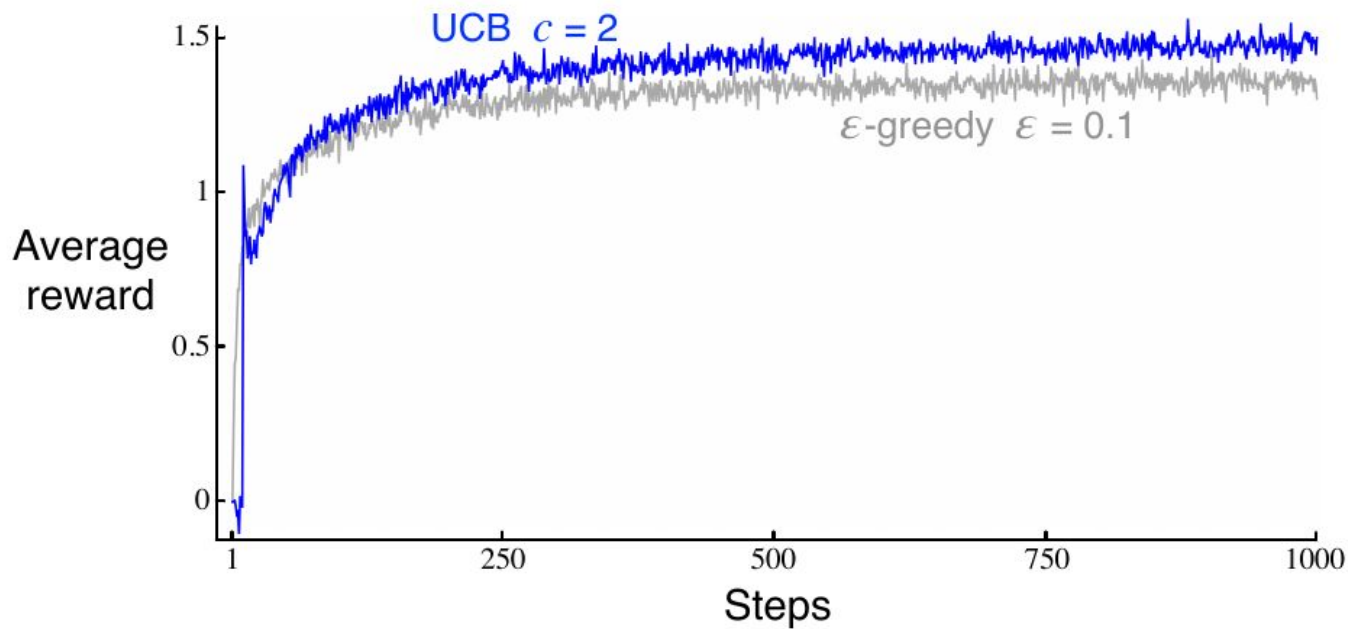
# Upper Confidence Bound (UCB)

- Take into account each action's **potential** to be optimal
- Selected less  $\rightarrow$  more potential
- Difficult to extend beyond multi-armed bandits

$$A_t \doteq \operatorname{argmax} \left[ Q_t(a) + c \sqrt{\frac{\ln t}{N_t(a)}} \right]$$



# UCB vs. $\epsilon$ -greedy



# Gradient Bandit Algorithms

- Learn a numerical preference  $H_t(a)$  for each action
- Convert to probability with softmax:

$$\pi_t(a) = \frac{e^{H_t(a)}}{\sum_{b \in \mathcal{A}} e^{H_t(b)}}$$

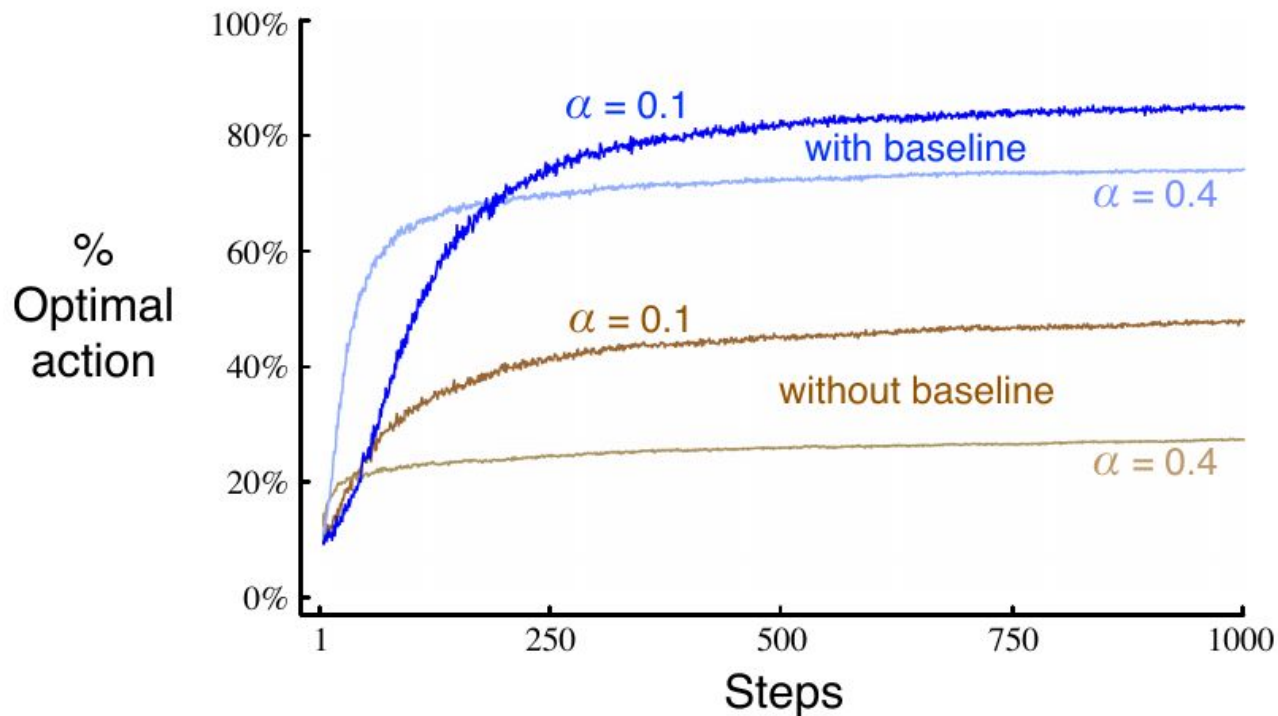
# Gradient Bandit: Stochastic Gradient Descent

- Update preference  $H_t(a)$  with SGD

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

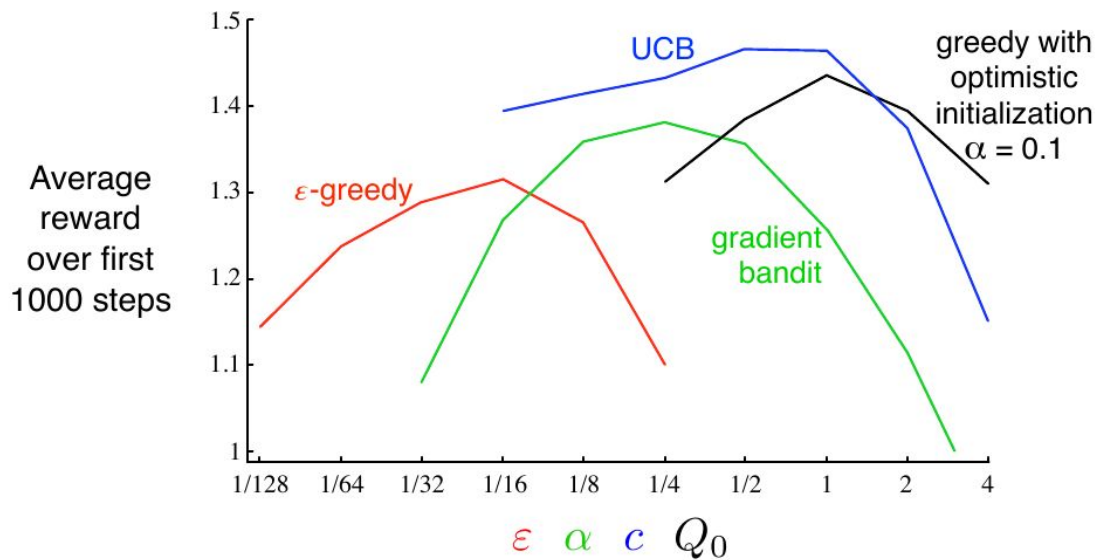
- Baseline  $\bar{R}_t$  : average of all rewards  $R_1, R_2, \dots, R_t$ 
  - Increase probability if reward is above baseline
  - Decrease probability if reward is below baseline

# Gradient Bandit: Results



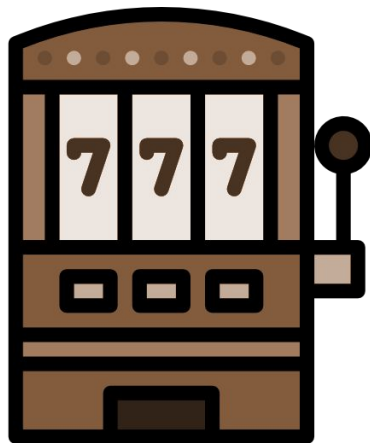
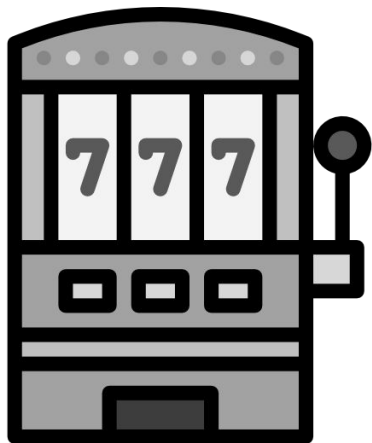
# Parameter Study

- Check performance in best setting
- Check hyperparameter sensitivity



# Associative Search (Contextual Bandit)

- Observe some *context* that can help decision
- Intermediate between multi-armed bandit and full RL problem
  - Need to learn a **policy** to *associate* observations and actions
  - Each action only affects immediate reward



# Thank you!

Original content from

- [Reinforcement Learning: An Introduction by Sutton and Barto](#)

You can find more content in

- [github.com/seungjaeryanlee](#)
- [www.endtoend.ai](#)