Multi-armed Bandits (MAB)

Mohammad **Dehghani**

Mechanical and Industrial Engineering Department

Course Structure

Part I : Tabular Solutions

- **Core ideas of RL algorithms**
- **State and action states are small enough for the approximate value functions to be represented as arrays, or tables.**
- **Methods can often find the *exact* solution**
  - Optimal value functions
  - Optimal policy

Part II : Approximate Solutions

- **Problems with large state space**
  - Tabular Solutions become insufficient and unsuitable

- **Adopt a new approach based on the features of each state**
  - use these set of features to generalize the estimation of the value at states

Part I : Tabular Solutions

Dynamic Programming

- DP is efficient, it finds optimal policies in polynomial time for most cases.
- DP requires a complete and accurate knowledge of the model of the environment

Monte Carlo Methods

- MC consists of playing as many episodes as possible and compute the values of the states that we have passed through, then average those results.
- It does not require a model of the environment's dynamics.
- It does not work with environment with no terminating states.

Temporal Difference Learning

- TD can be seen as the fusion between DP and MC methods.
- It plays episodes but does not have to wait until the end to know the return. It computes the value of the current state based on estimates of other states.
- Require no model and are fully incremental
- More complex to analyze
- The methods also differ in several ways
  - Efficiency
  - Speed of convergence

Other Methods

- Combination of DP, MC, and TD
- Multi-step bootstrapping

Feedback Types

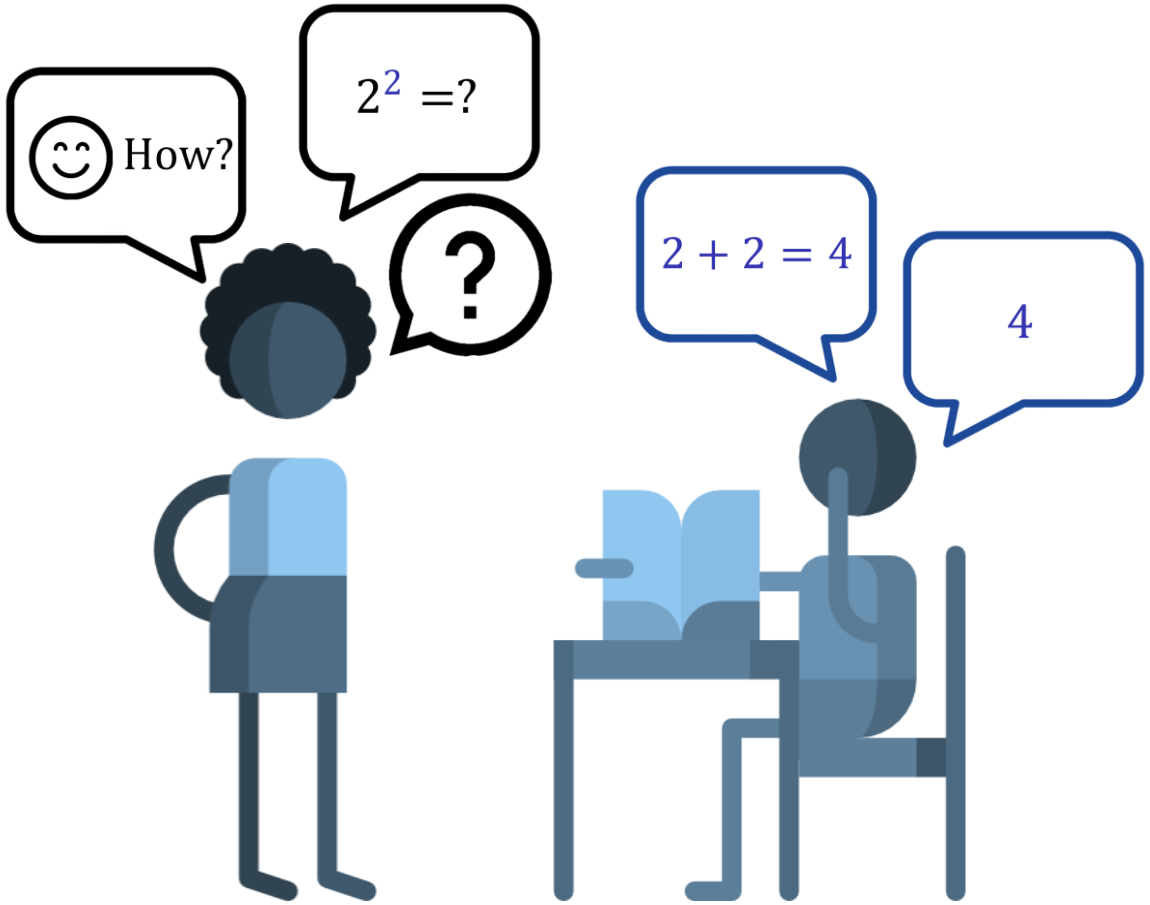
Evaluative feedback

- The reward received after each action gives some information about how good the action was.
- It is unknown that
  - If the result is the best, or
  - you achieved a good result from providing correct input, or by coincidence
- The Evaluative “feedback” depends entirely on the action selected
- The problem is inherently one requiring explicit search among the alternative actions. **(Exploration)**

Instructive feedback

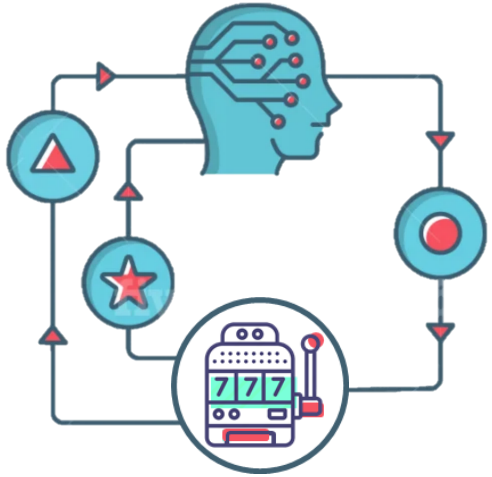
- The feedback from the environment directly indicates what the correct action should have been.
- There is no need to search: whatever action you try, you will be told what the right one would have been.
- The instructive "feedback" is typically independent of the action selected
- Instead of trying to make its environment behave in a certain way, it tries to make itself behave as instructed by its environment.

Feedback types



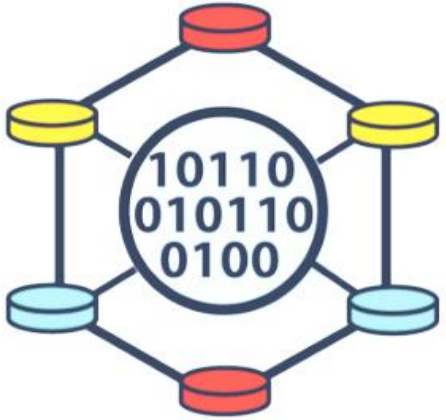
Feedback types

Reinforcement Learning



- **Evaluative feedback**

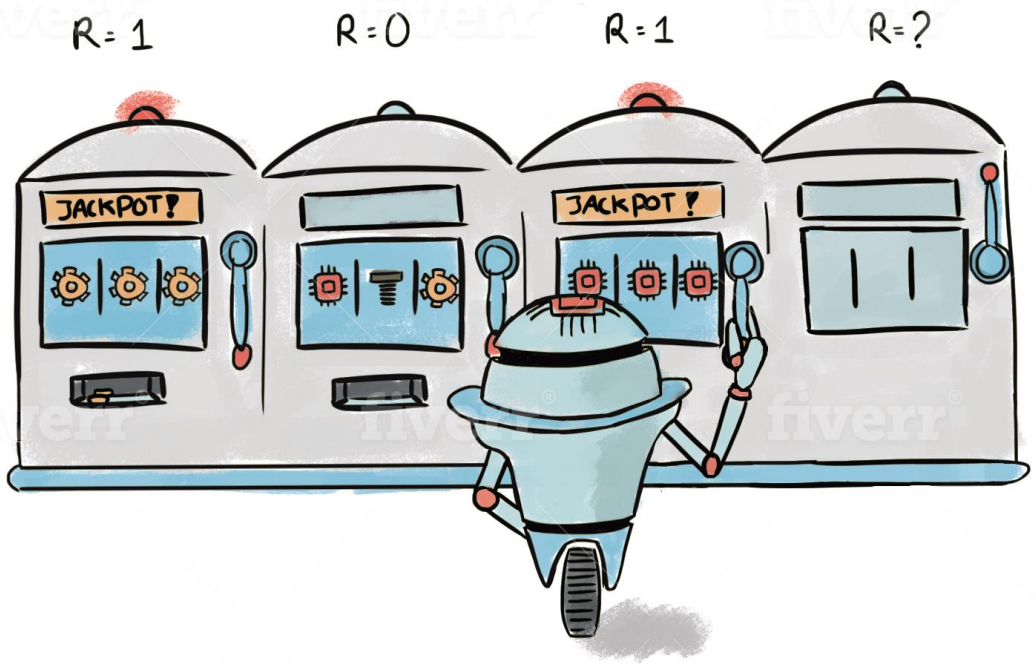
Supervised Learning



- **Instructive feedback**

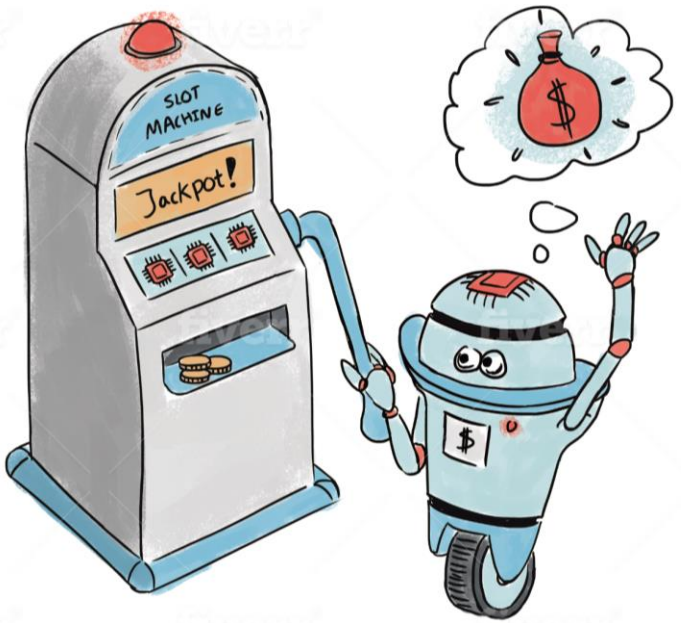
» **RL uses training information that *evaluates* the actions taken rather than *instructs* by giving correct actions.** «

Multi-armed Bandits (MAB)



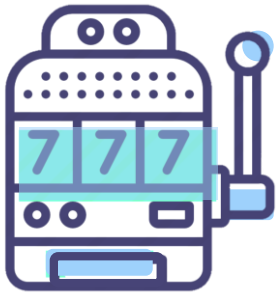
MULTI-ARMED BANDITS
BANDITS PROBLEM

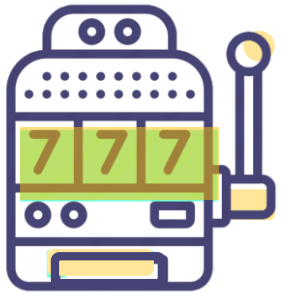
Make Money in Casino

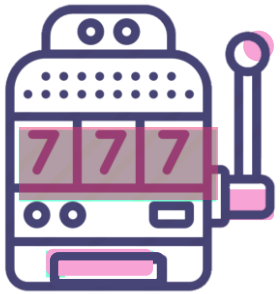


an agent facing arms in a Multi-Armed Bandits

The Multi-Armed Bandits


v_1


v_2


v_3


v_4

- At time-step t
  - **Select an Action:** Choose an Arm A_t
  - **Receive a reward:** $R_t = X_{A_t,t}$
    - Hit Jackpot: $\boldsymbol{R = +1}$
    - Otherwise: $\boldsymbol{R = 0}$

- **Goal:** Maximize the expected total reward over some time steps

Historical Motivation

- **Medical Trials** [Thompson, 1933; Gittins, 1979]


μ_1


μ_2


μ_3


μ_4

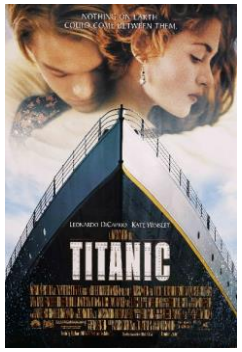
- **For the $t-th$ patient in the clinical study,**
  - Choose a treatment A_t
  - Observe a response $R_t \in \{0,1\} : P(R_t = 1 \,|R_t = \mu_a)$

- **Goal:** Maximize the expected number of patients healed.

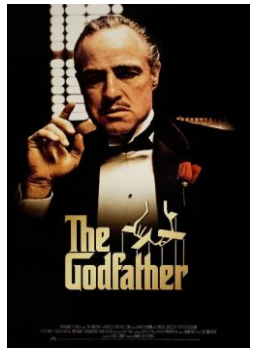
Modern Motivation

- **Recommender systems / Online advertisement** [Bresler et al., 2016]

i_1



i_2



i_3

i_4

i_5

- **As the user u logs into a virtual store**
  - The Recommender System selects an item i from the set of available items ($\mathcal{I}$), and recommends it to the user
    - $L_{u,i}$equals to $+1$ (like) or -1 (dislike).
    - Observe a rating L_{u_t,I_t}
  - The recommendation must depend only on previous feedback
- **Goal:** Minimize the number of bad recommendations per user *(Regret Function)*

Modern Motivation - extended

Watch this movie

Drive this route

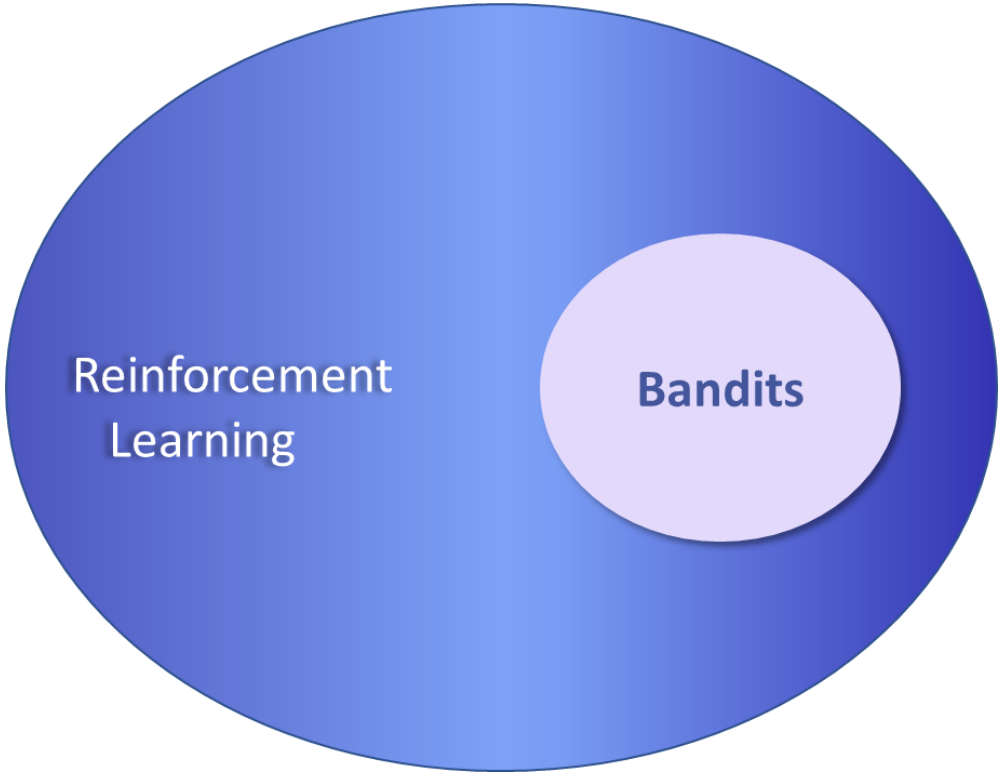
Dine in this restaurant

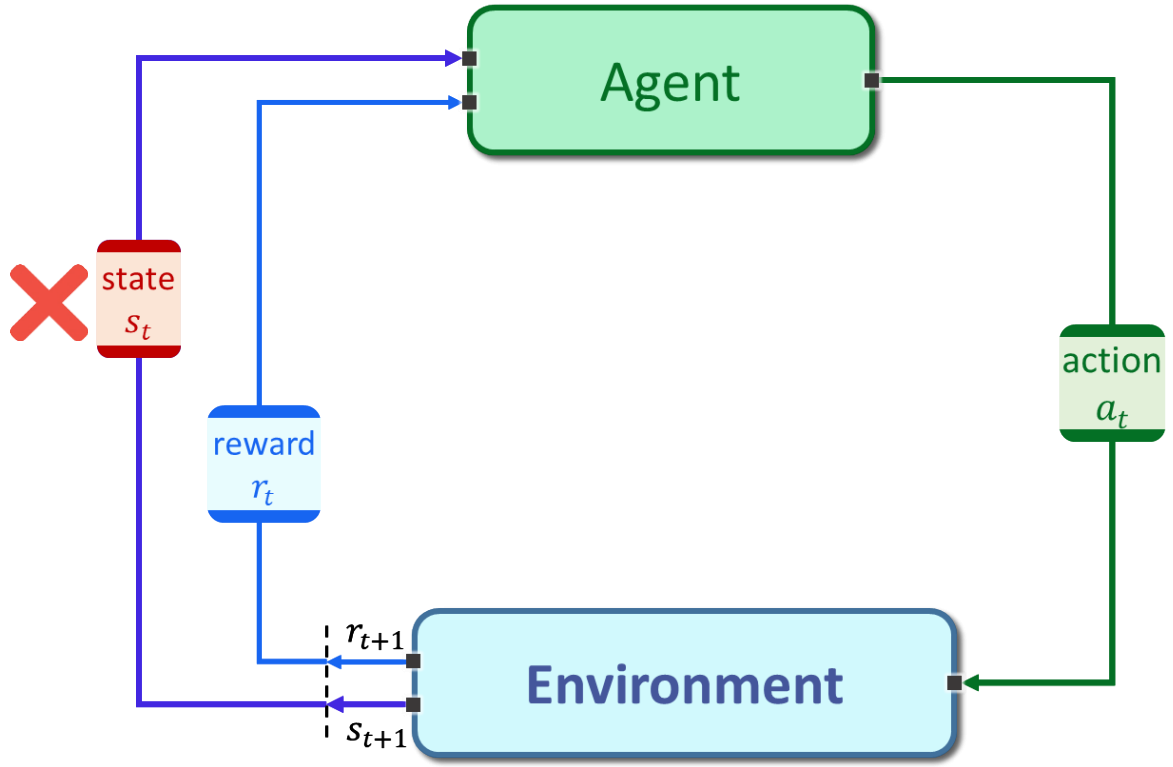
Do this exercise

Vacation in this resort

Buy this product

Why Bandits



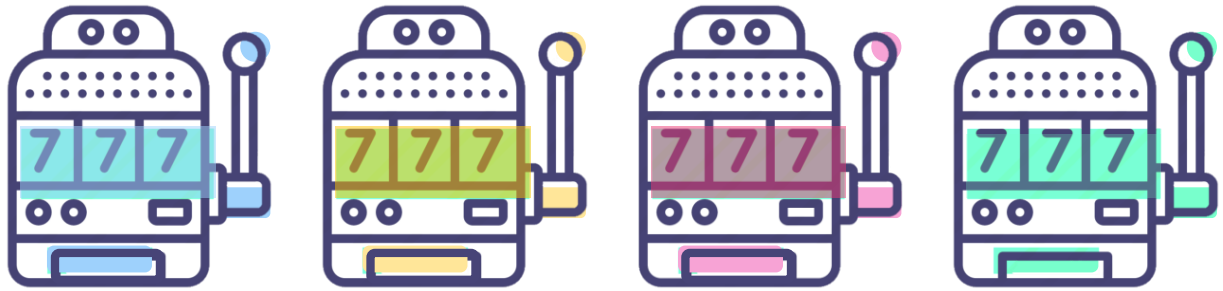


- Bandits have no different states, but only one
- The reward taken under consideration is only the immediate one

Why Bandits

- Best to consider issues and algorithm
- Rewards maximization in a stochastic bandit model
  - the simplest RL problem (one state)
- Bandits showcase the important exploration/exploitation dilemma
- A *rich literature* to tackle many specific applications

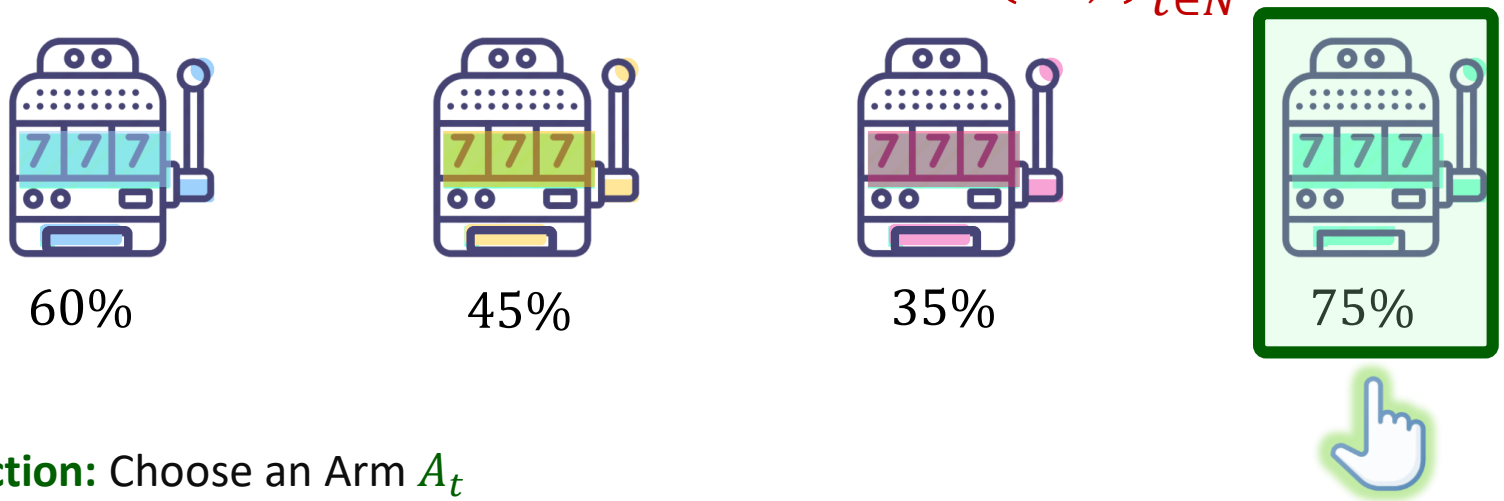
MABs
Settings



The Multi-Armed Bandits Setup

- **Know values/Certain values (Trivial)**

K **arms** ↔ K rewards streams $(X_{a,t})_{t \in N}$

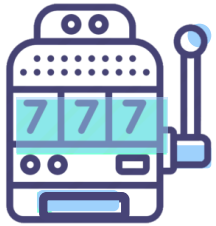


- **At time-step t**
  - **Select an Action:** Choose an Arm A_t
  - **Receive a reward:** $R_t = X_{A_t,t}$
- ***True Value* of Action a:** $\boldsymbol{q_*(a) \doteq [R_t | A_t = a]}$
- **Goal:** Maximize $\left[\sum_{t=1}^{N} R_t\right]$

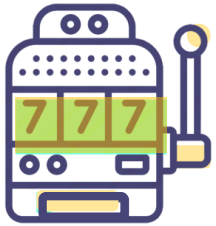
The Stochastic Multi-Armed Bandit Setup

- **Stochastic values/Uncertain values**

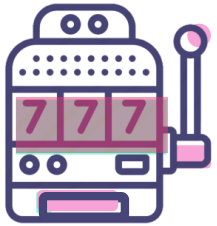
K **arms** ↔ K probability distributions : v_a has mean μ_a



v_1



v_2



v_3

v_4

- **At time-step t**
  - **Select an Action:** Choose an Arm A_t
  - **Receive a reward:** $R_t = X_{A_t,t} \sim v_{A_t}$
- ***Expected Value*** of Action a: $\boldsymbol{q_*(a) \doteq \mathbb{E}[R_t | A_t = a]}$

- **Goal:** Maximize $\mathbb{E}\left[\sum_{t=1}^{N} R_t\right]$

The Stochastic Multi-Armed Bandit Setup

Definition

- **Expected Value of Action a:**

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

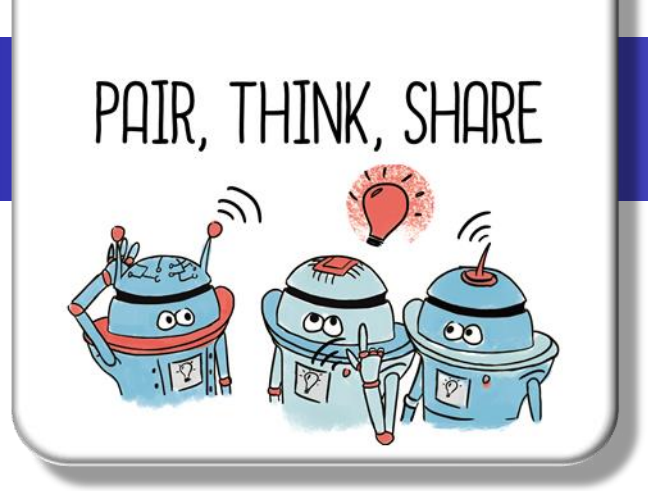
Definition: Q-value

- **Estimated Value of Action a at time t :**

$$\boxed{Q_t(a)}$$

- **Infinite sampling:**
  - As we take the action more, this estimate becomes more accurate (law of large numbers) and in the limit:

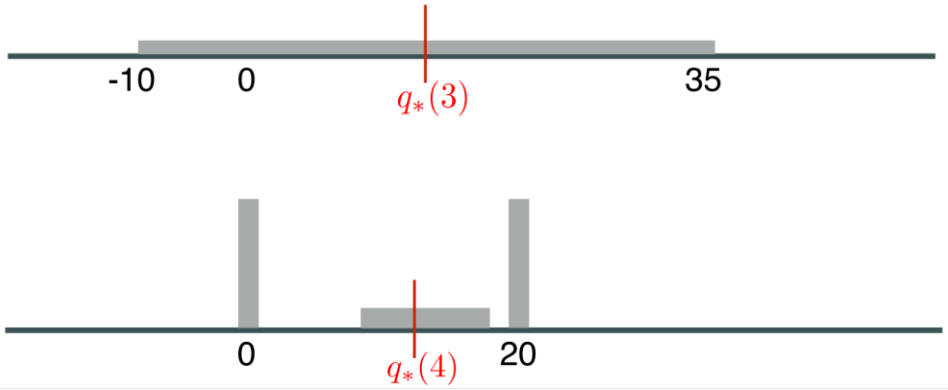
$$\boxed{\lim_{n \to \infty} Q_t(a) = q_*(a)}$$



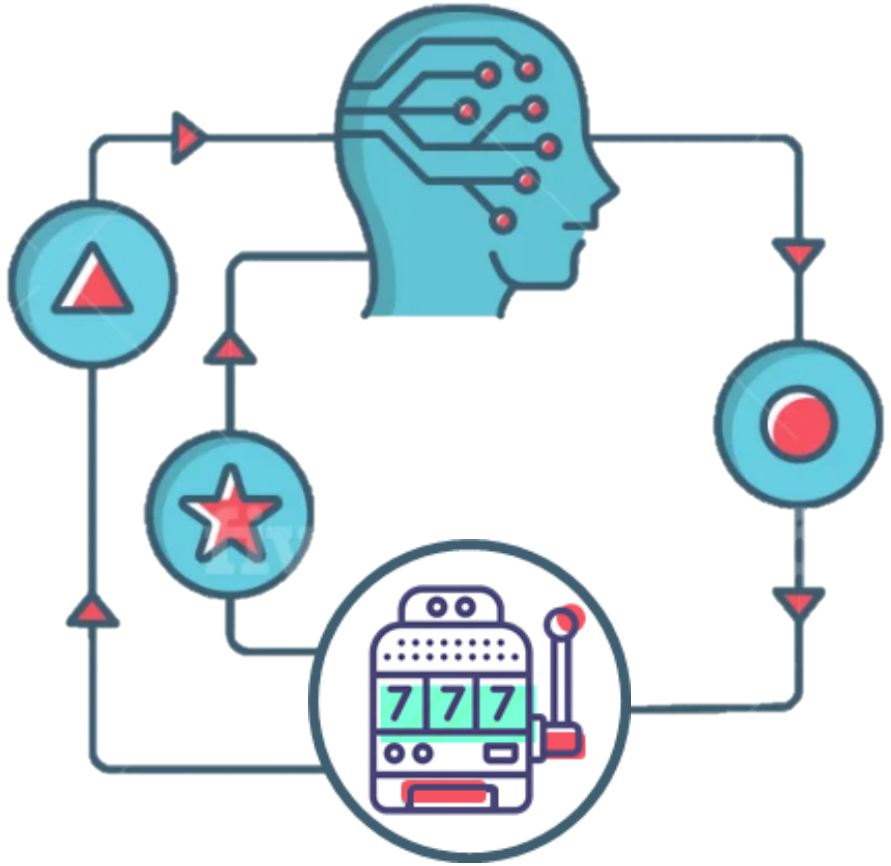
Example of a 4-armed bandit:

How you should act?

- **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 $q_*(2) = 12$
- **Action 3:** randomly between -10 and 35, equiprobable
  - Value of action 3 is $q_*(3) = 12.5$
- **Action 4:** a third 0, a third 20, and a third from {8,9,…, 18}
  - Value of action 4 is $q_*(4) = 11$



MAB Strategies



Two naive strategies

Idea 1: Greedy policy

- Always trust the empirical ***best*** arm

Exploitation

- Exploiting your current knowledge of the values of the actions
- maximize the expected reward on the one step, but not in the long run

Idea 2: Non-Greedy policy

- Draw each arm T/k times

Exploration

- Exploring the value of nongreedy action's
- Exploration may produce the greater total reward in the long run

- If you have many time steps ahead on which to make action selections?

A new Idea

Idea 3: ε – greedy Algortihm

- **Behave greedily most of the time: $(1-\varepsilon)$**
- **Select randomly from among all the actions (ε)**
  - equal probability, independently of the action-value estimates.

- **As the number of steps increases:**
  - every action will be sampled an infinite number of times, t
  - $Q_t(\boldsymbol{a})$ converges to $\boldsymbol{q}_*(\boldsymbol{a})$

How to estimate the values of actions?

- **Action-value Methods (sample-average method)**
  - In this method, the true value of an action is the **mean reward** when that action is selected

Action is selected

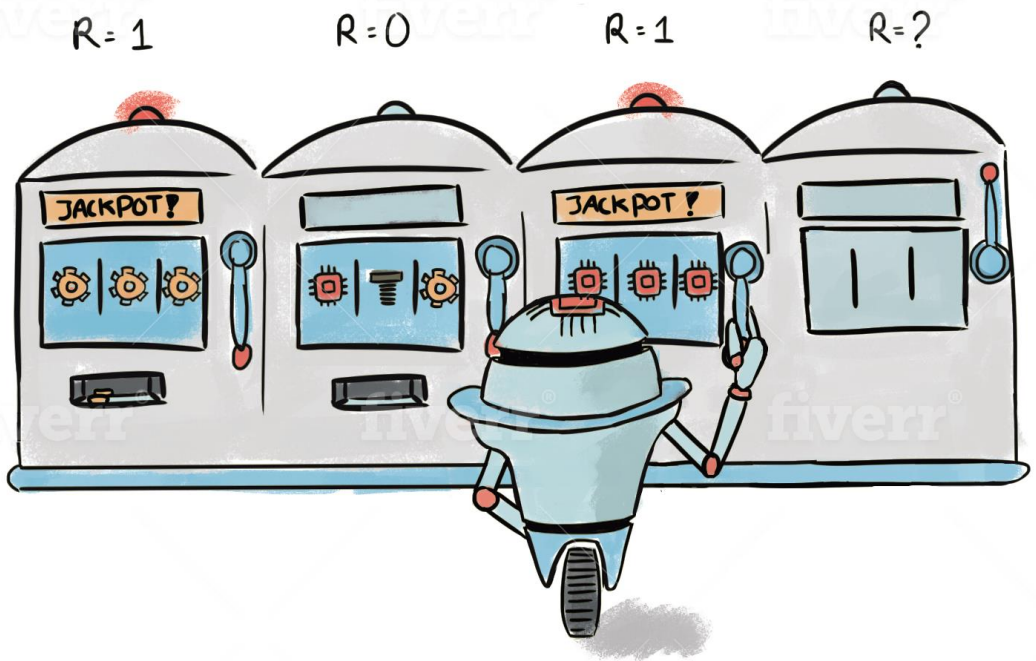
$$Q_t(\boldsymbol{a}) \doteq \frac{\sum_{i-1}^{t-1} R_i . \mathbb{1}_{A_i=a}}{\sum_{i-1}^{t-1} . \mathbb{1}_{A_i=a}}$$

where:

- $\mathbb{1}_{predicate}$ denotes the random variable that is 1 if predicate is true

$$Q_t(\boldsymbol{a}) \doteq \frac{\text{sum of rewards when } a \text{ taken prior to } t}{\text{number of times } a \text{ taken prior to t}}$$

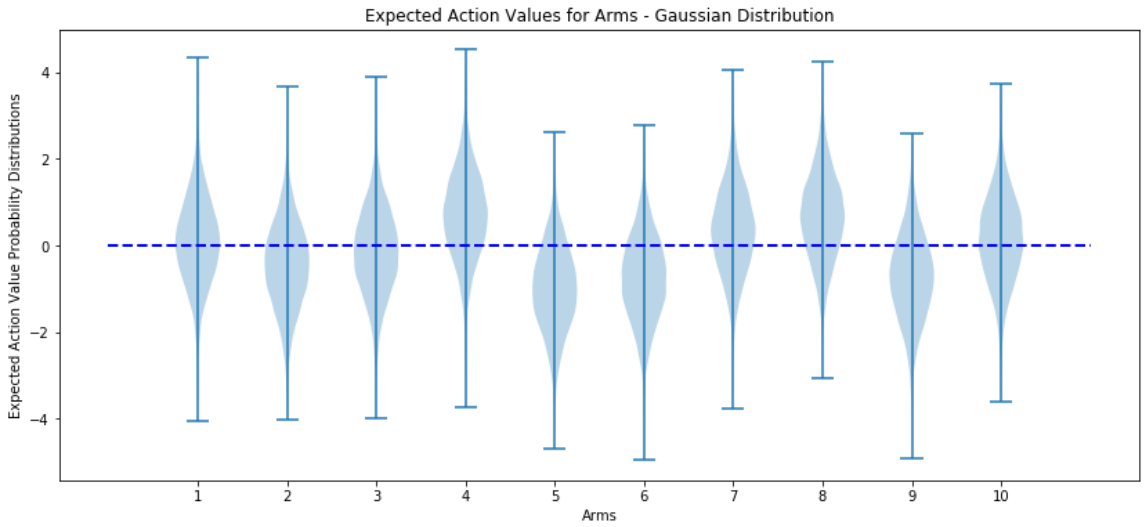
Multi-armed Bandits
Testbeds



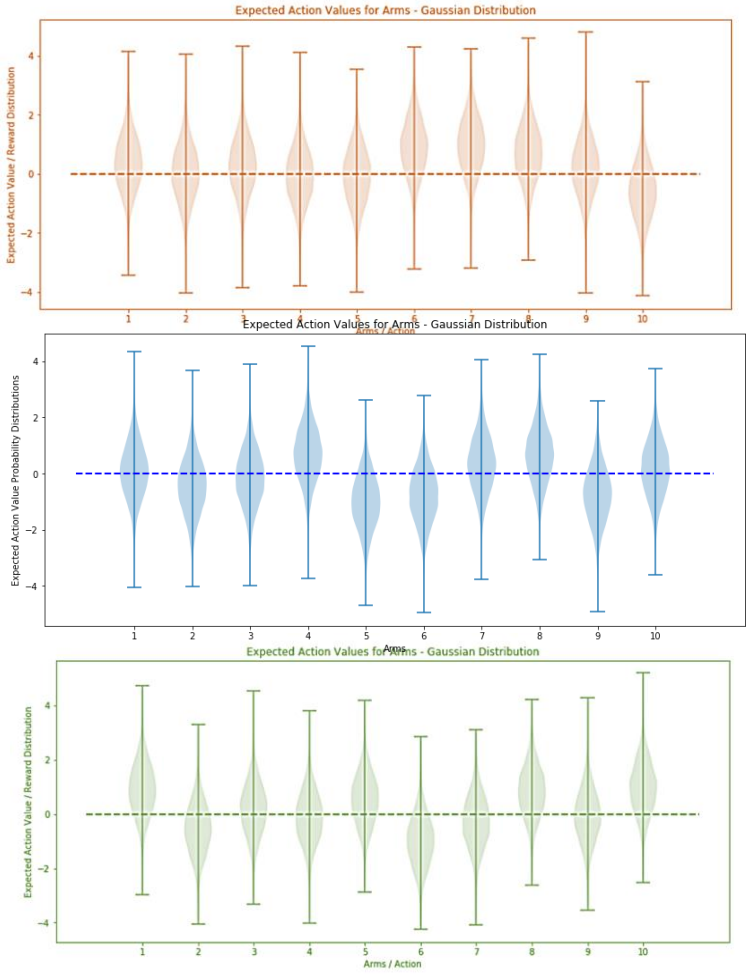
MULTI-ARMED BANDITS
BANDITS PROBLEM

Stationary vs Nonstationary

Stationary



Nonstationary



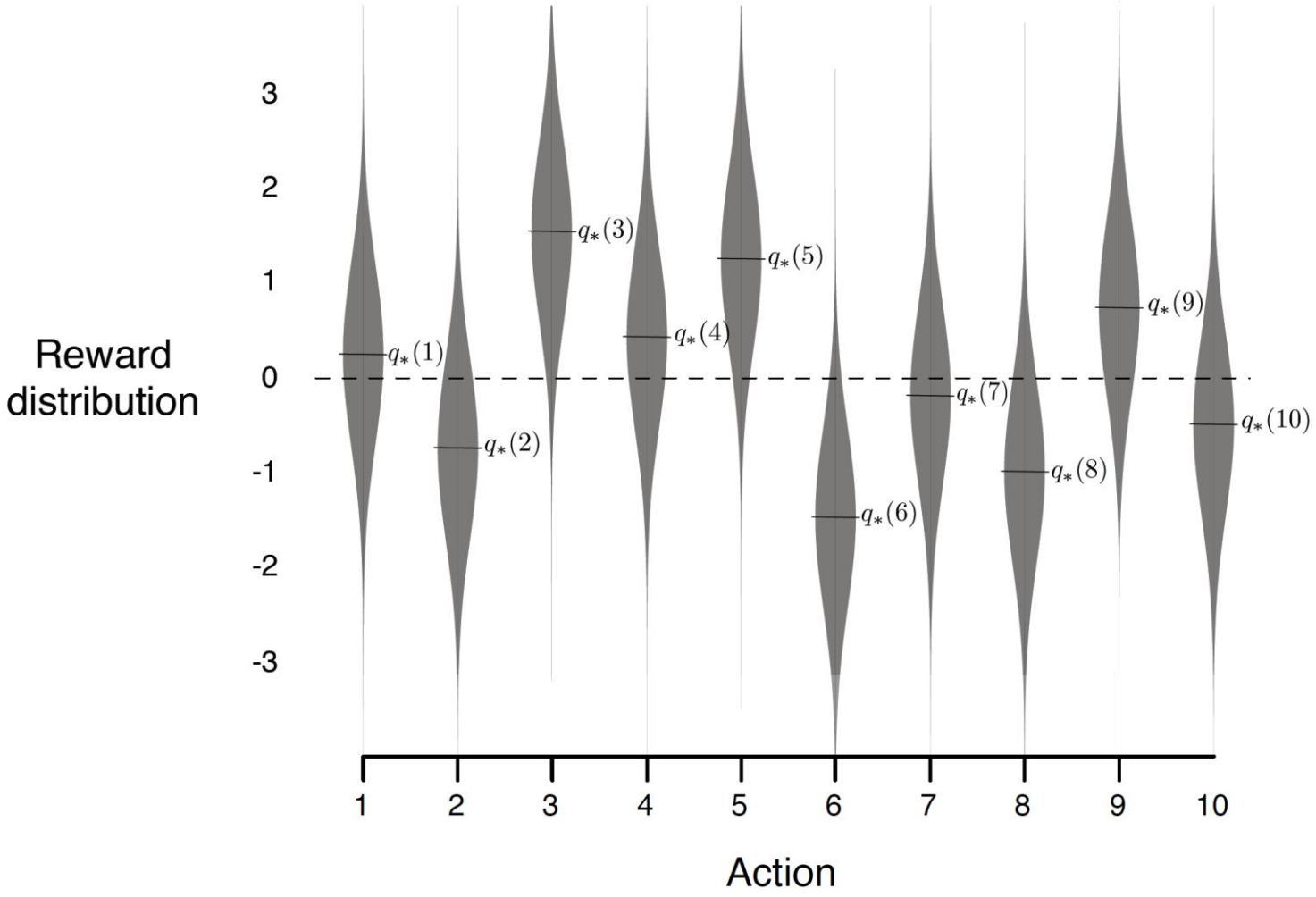
- **Stationary**
  - underlying model does not change over time
- **Nonstationary**
  - underlying model can change during play

The 10-armed Testbed (settings)

- **Actual rewards**
  - Normal (Gaussian) distribution
  - $N(0,1)$
- **Experiment Settings**
  - **Steps:** 1000 steps
  - **Runs:** 2000 replications
- **Scenarios**
  - greedy:

$$A_t \doteq \text{argmax}_a \, Q_t(a)$$

  - $\boldsymbol{\varepsilon}$−greedy
    - $\varepsilon = 0.01$
    - $\varepsilon = 0.1$



Incremental Implementation (sample-average)

Sample-average method

- R_i: the ***reward*** received after i_{th} selection of the action

$$Q_t(\boldsymbol{a}) \doteq \frac{\text{sum of rewards when } a \text{ taken prior to } t}{\text{number of times } a \text{ taken prior to t}}$$

- Q_{n+1}: the ***estimate of the action value*** after it has been selected n times

$$\boxed{Q_{n+1} \doteq \frac{R_1 + R_2 + \cdots + R_n}{n}}$$

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

Incremental Implementation (sample-average)

- R_i: the *reward* received after i_{th} selection of the action

- Q_n: the *estimate of the action value* after it has been selected $n - 1$ times

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

$$NewEstiamte \leftarrow OldEstimate + StepSize\ [Target - OldEstiamte]$$

α

error in the estimate

Incremental Implementation (Pseudocode)

A simple bandit algorithm

Initialize, for $a = 1$ to k:

$Q(a) \leftarrow 0$

$N(a) \leftarrow 0$

Loop forever:

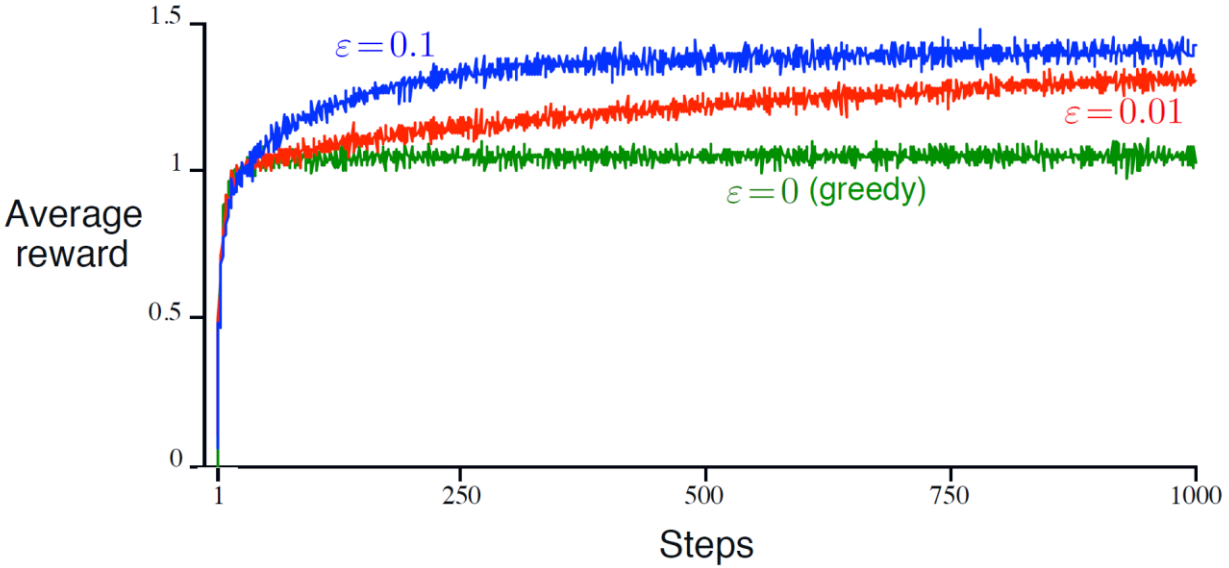
$$A \leftarrow \begin{cases} \text{argmax}_a\, Q(a) & \text{with probability } 1-\varepsilon \quad \text{(breaking ties randomly)} \\ \text{a random action} & \text{with probability } \varepsilon \end{cases}$$

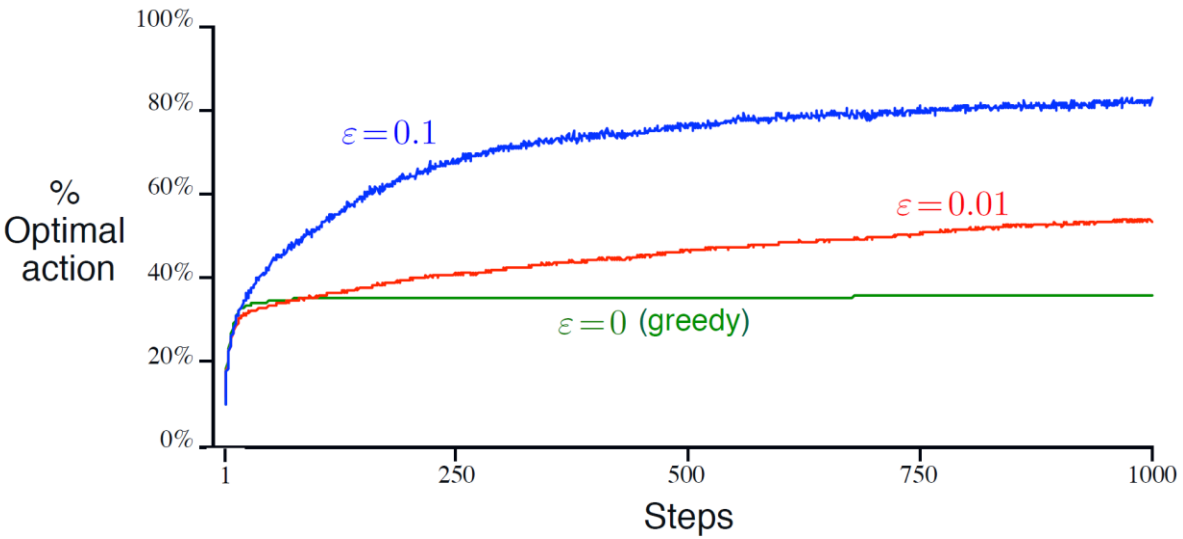
$R \leftarrow bandit(A)$

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

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

The 10-armed Testbed (results)



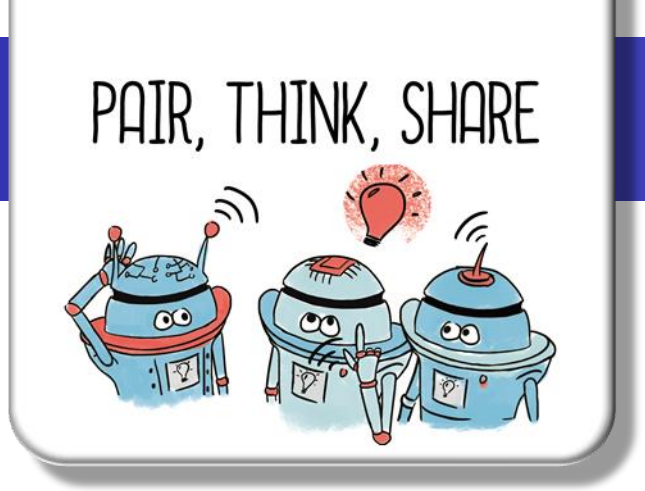


Average performance of "-greedy action-value methods on the 10-armed testbed. These data are averages over 2000 runs with different bandit problems. All methods used sample averages as their action-value estimates.

Observations and Weaknesses

- **Observations**:
  - Greedy method improved faster at the very beginning, but level off at a lower level.
  - ε –greedy methods continue to Explore and eventually perform better.
  - The $\varepsilon = 0.01$ method improves slowly, but eventually performs better than the $\varepsilon = 0.1$ method.

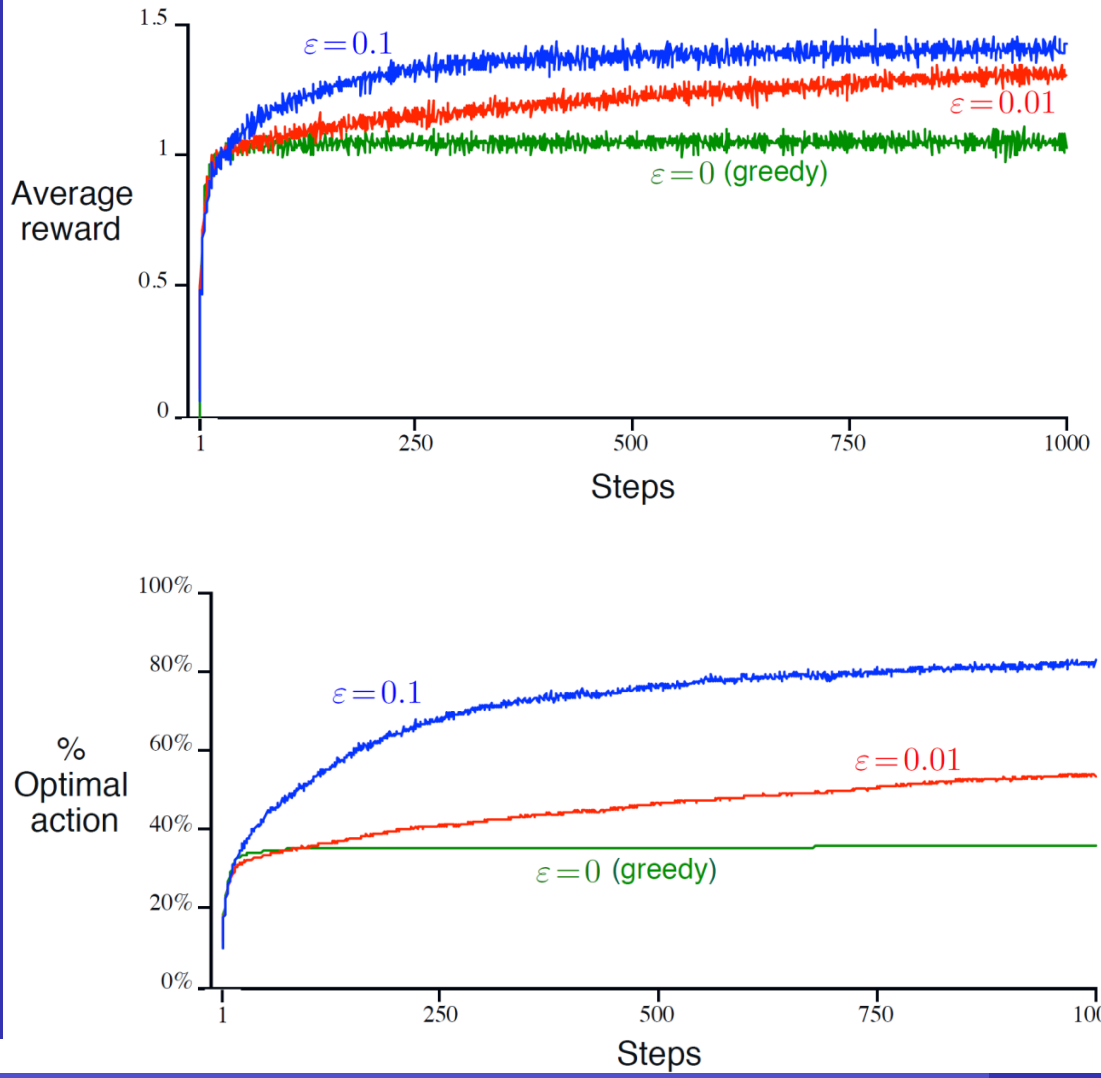
- **Weaknesses**
  - Randomly selects an action to explore, does not explore more "promising" actions.
  - Does not consider confidence interval. If an action has been taken many times, no need to explore it.



1. In the ε–greedy method, with ε=0.1, what is the probability of selecting the optimal action?

2. Which method will perform best in the long run in terms of cumulative reward and probability of selecting the best action?

3. Suppose the reward variance had been larger, say 10 instead of 1. Then which approach is more desirable?



Tracking a Nonstationary Problem

- The ***averaging methods*** are appropriate for ***stationary*** bandit problems

- In ***nonstationary*** cases, it makes sense to give more ***weight*** to ***recent rewards*** than to long-past rewards

- Therefore, we can use a constant step-size parameter (α)

$$Q_{n+1} \doteq Q_n + \alpha \left[R_n - Q_n\right]$$

where the step-size parameter $\alpha \epsilon (0,1]$ is constant

Tracking a Nonstationary Problem (weighted-average)

- This results in Q_{n+1} being a weighted average of past rewards and the initial estimate Q_1:

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

Tracking a Nonstationary Problem (weighted-average)

Definition

In the weighted average method, Q_{n+1} being a weighted average of the ***initial estimate*** (Q_1) and the ***past rewards***:

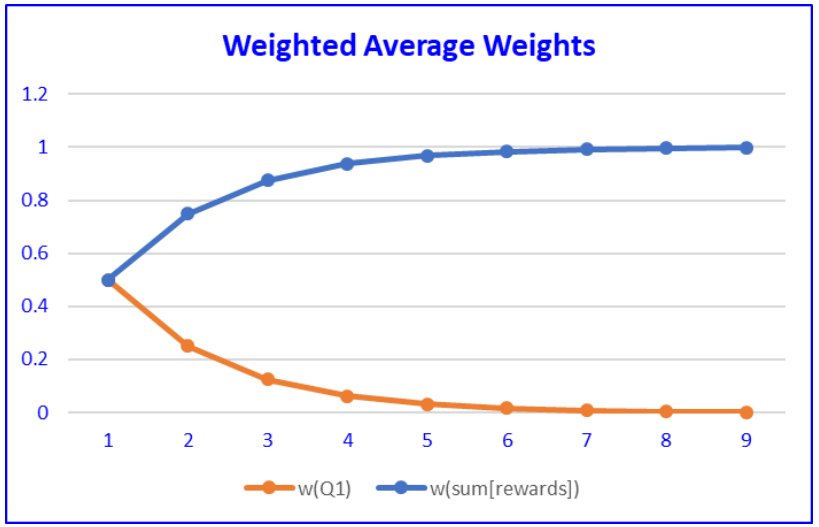
$$Q_{n+1} \doteq (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i$$

- The weight given to R_i decreases as the number of intervening rewards increases.
- In fact, the weight decays exponentially according to the exponent on $1-\alpha$
  - If $1-\alpha = 0$, then all the weight goes on the very last reward, R_n
- Accordingly, this is sometimes called an ***exponential recency-weighted average***.

Tracking a Nonstationary Problem (weighted-average)

alpha	1-alpha
0.5	0.5

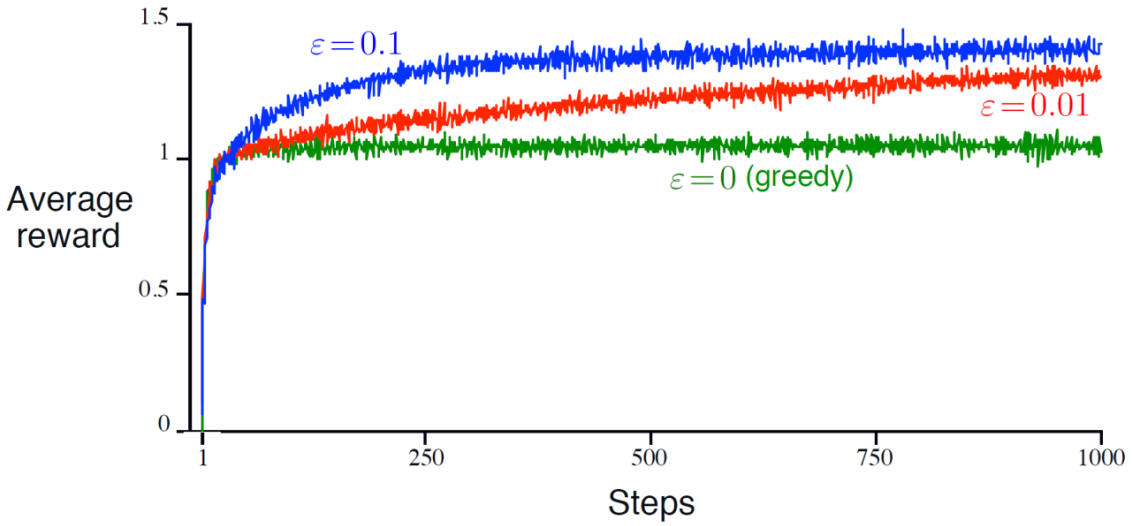
	n=1	n=2	n=3	n=4	n=5	n=6	n=7	n=8	n=9
i	1	2	3	4	5	6	7	8	9
1	0.5	0.25	0.125	0.0625	0.03125	0.015625	0.007813	0.003906	0.001953
2		0.5	0.25	0.125	0.0625	0.03125	0.015625	0.007813	0.003906
3			0.5	0.25	0.125	0.0625	0.03125	0.015625	0.007813
4				0.5	0.25	0.125	0.0625	0.03125	0.015625
5					0.5	0.25	0.125	0.0625	0.03125
6						0.5	0.25	0.125	0.0625
7							0.5	0.25	0.125
8								0.5	0.25
9									0.5
w(sum[rewards])	0.5	0.75	0.875	0.9375	0.96875	0.984375	0.992188	0.996094	0.998047
w(Q1)	0.5	0.25	0.125	0.0625	0.03125	0.015625	0.007813	0.003906	0.001953
sum	1	1	1	1	1	1	1	1	1



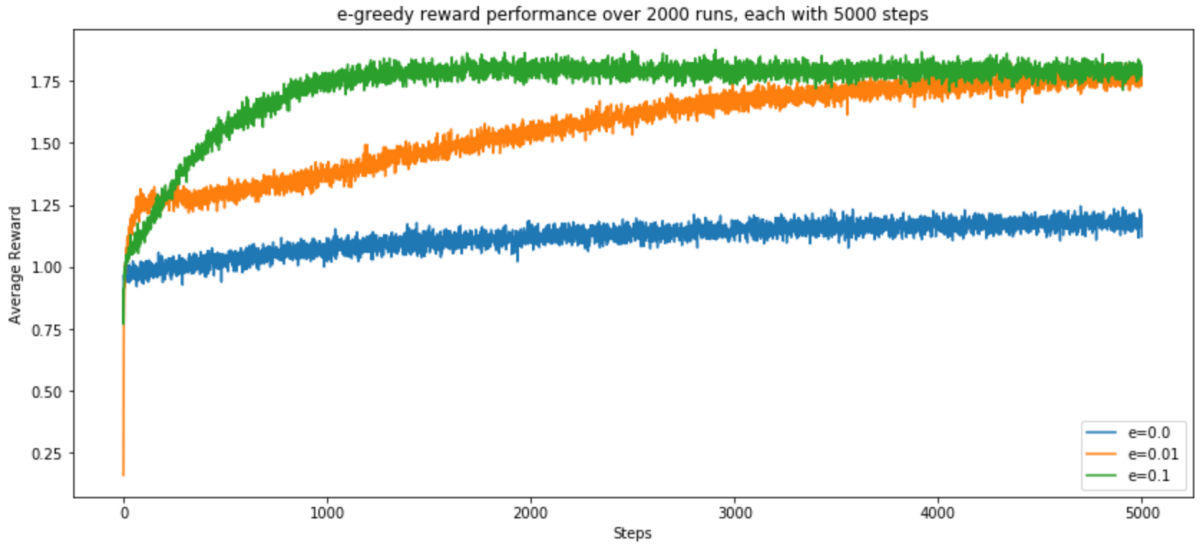
Tracking a Nonstationary Problem (results)

Tracking a Nonstationary Problem (results)

Sample Average (1000 steps)



Weighted Average (5000 steps)



Tracking a Nonstationary Problem

Convergence Conditions (Robbins-Monro conditions)

There are two conditions required to assure convergence with probability

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

- The first condition is required to guarantee that the steps are large enough to eventually overcome any initial conditions or random fluctuations.
- The second condition guarantees that eventually the steps become small enough to assure convergence.
- the choice $\alpha_n(a) = \frac{1}{n}$ results in the sample-average method, which is guaranteed to converge to the true action values by the law of large numbers.

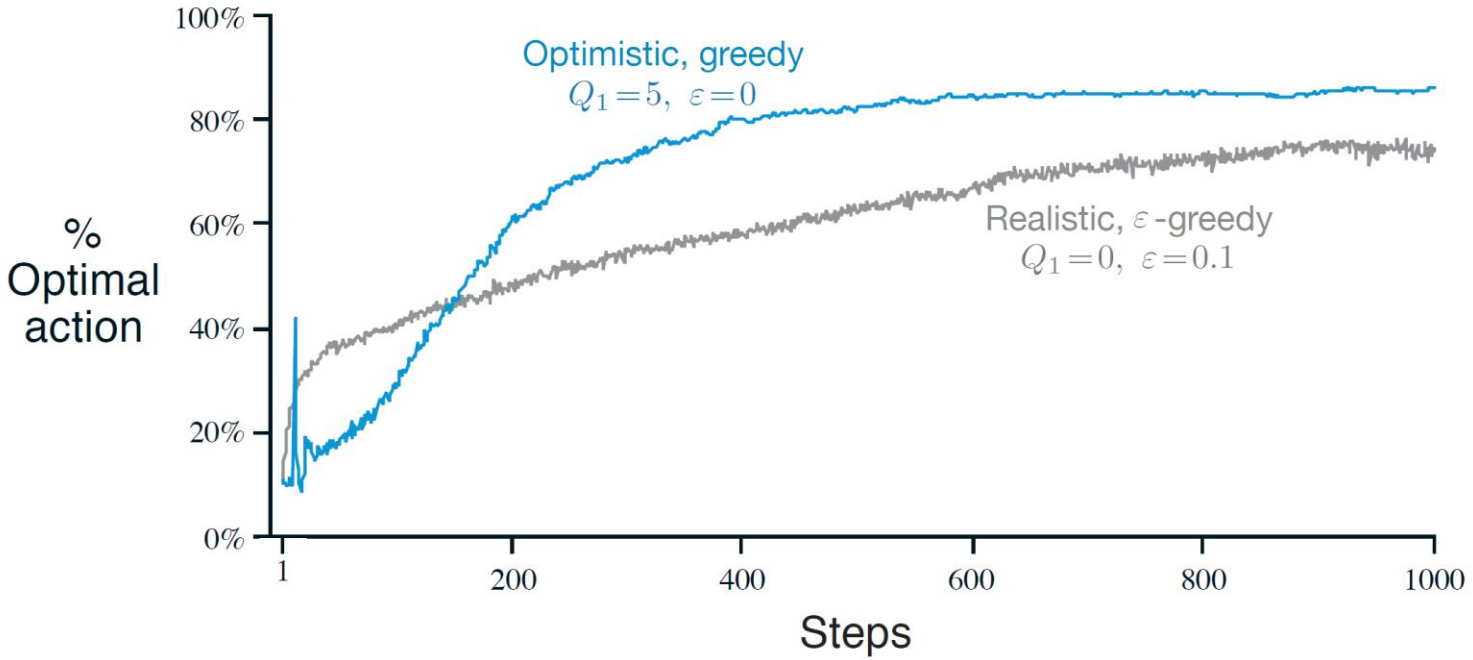
Optimistic Initial Values (a simple trick)

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

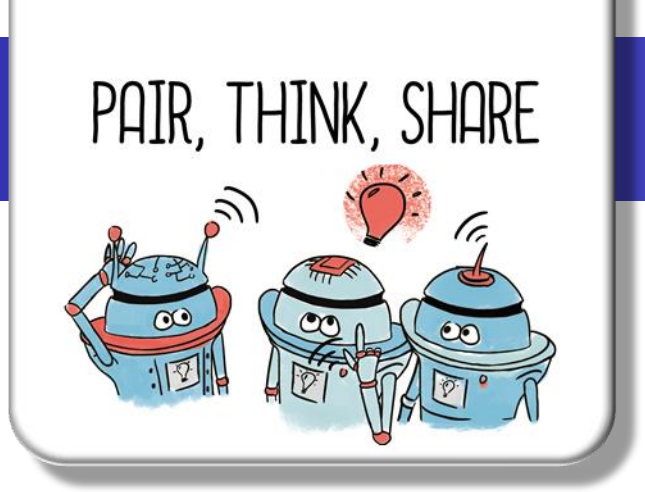
$$Q_{n+1} \doteq (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^n R_i$$

- All the methods we have discussed so far are dependent to some extent on the initial action-value estimates, $Q_1(a)$
  - These methods are ***biased*** by their initial estimates
- For the sample-average methods, the bias disappears once all actions have been selected at least once
- For methods with ***constant α***, the bias is ***permanent***, though decreasing over time

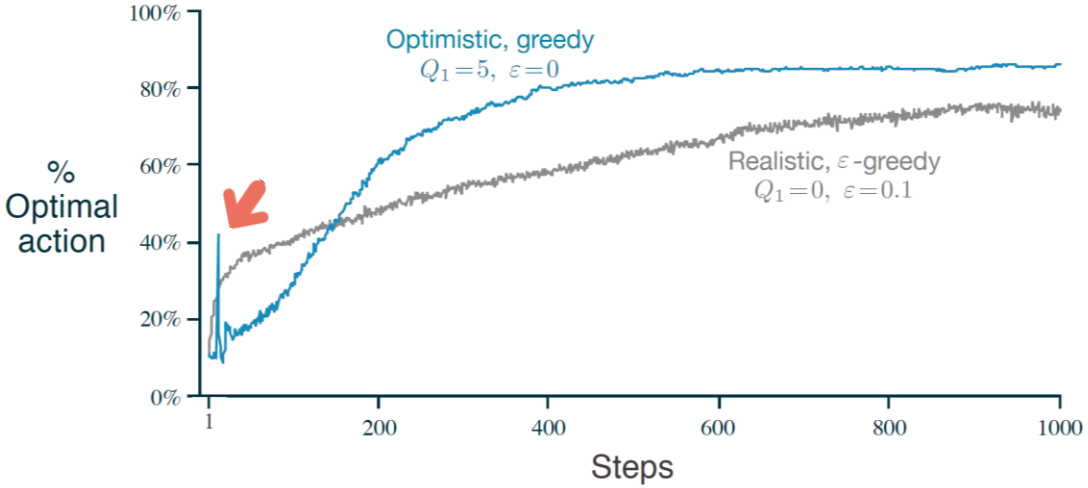
Optimistic Initial Values (results)



The effect of optimistic initial action-value estimates on the 10-armed testbed. Both methods used a constant step-size parameter, $\alpha = 0.1$



1. Why *optimistic greedy approach* outperforms *ε-greedy* method?
2. What is the reason for oscillations and spikes in the early part of the curve?
3. Does the *optimistic initial conditions* help with the general *nonstationary* cases?



Upper-Confidence-Bound (UCB) Action Selection

- ε -greedy action selection forces the non-greedy actions to be tried, but indiscriminately, with ***no preference*** for those that are nearly greedy or particularly uncertain.
- It would be better to select among the non-greedy actions according to their ***potential for actually being optimal*** by taking into account:
  - how close their estimates are ***(i) to being maximal*** and ***(ii) the uncertainties*** in those estimates

UCB action selection

In the UCB method, actions are selected based on :

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

Where:

- $N_t(a)$ denotes the number of times that action a has been selected prior to time t
- number $c > 0$ controls the degree of exploration (confidence level)
  - If $N_t(a) = 0$ then a is considered to be a maximizing action

Upper-Confidence-Bound (UCB) Action Selection

UCB action selection

In the UCB method, actions are selected based on :

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

uncertainty or variance in the estimate of a's value

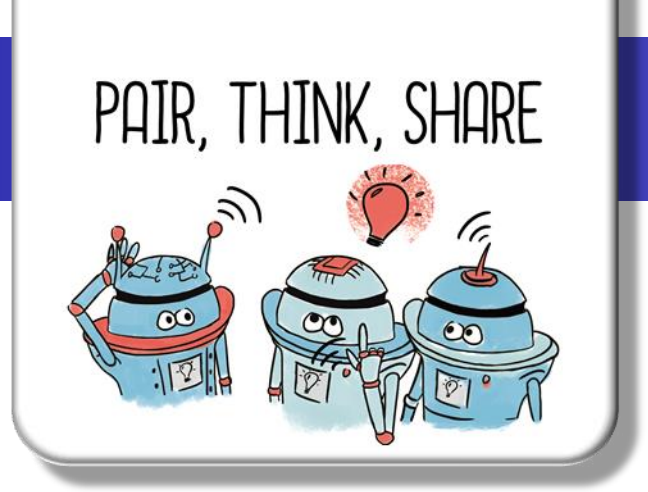
upper bound on the likely value of this action based on our uncertainty

select a $(A_t = a)$

- $N_t(a)$ ↑
- **Uncertainty** ↓

do not select a $(A_t \neq a)$

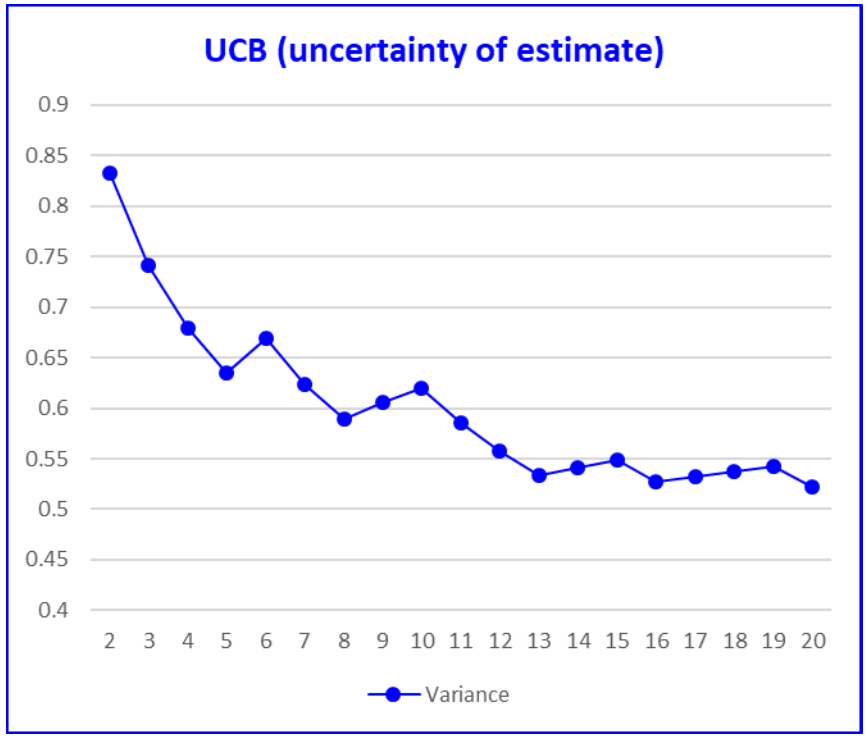
- $N_t(a)$ ↓
- **Uncertainty** ↑



The figures shows the variance estimate of UCB with 20 time steps (t) for a particular action a

- In which time steps, action a is selected? $(A_t = a)$

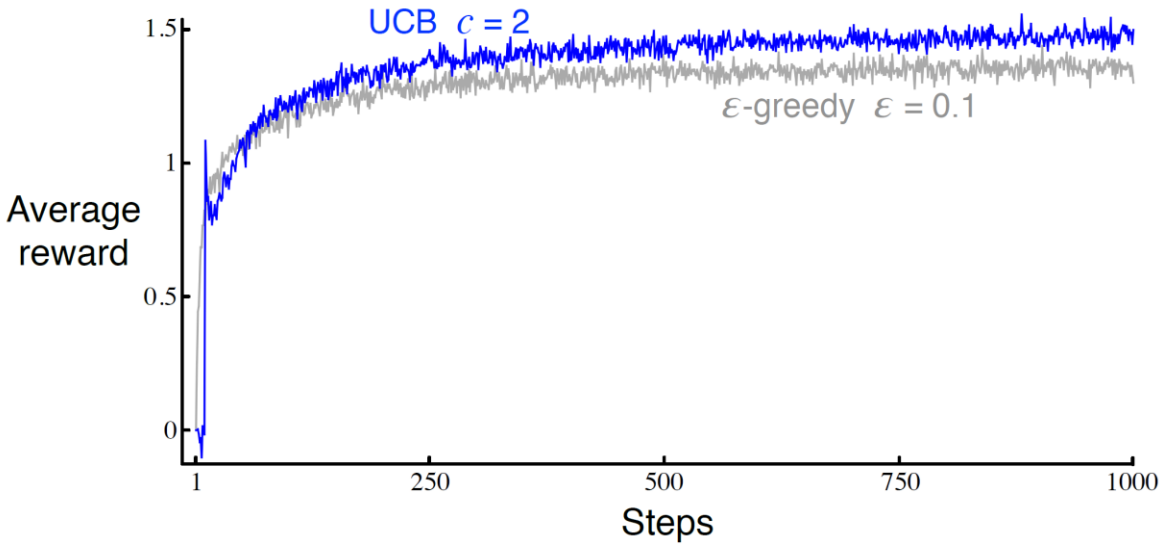
$$uncertainty \doteq \sqrt{\frac{\ln t}{N_t(a)}}$$

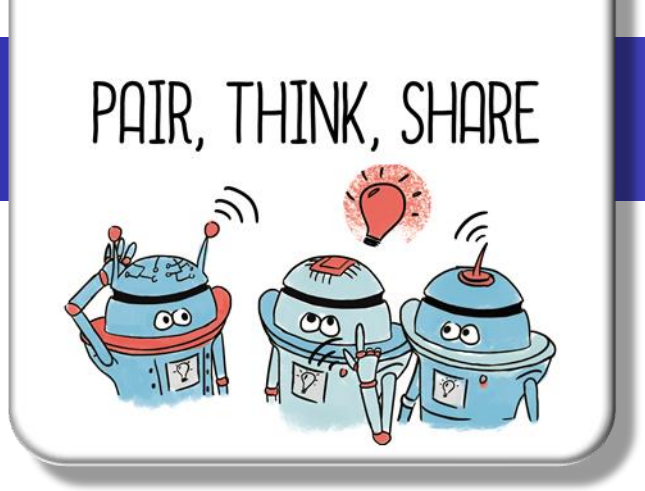


Upper-Confidence-Bound (results)

- **UCB generally performs better than ε -greedy action selection**
  - except in the first k steps, when it selects randomly among the as-yet-untried actions.

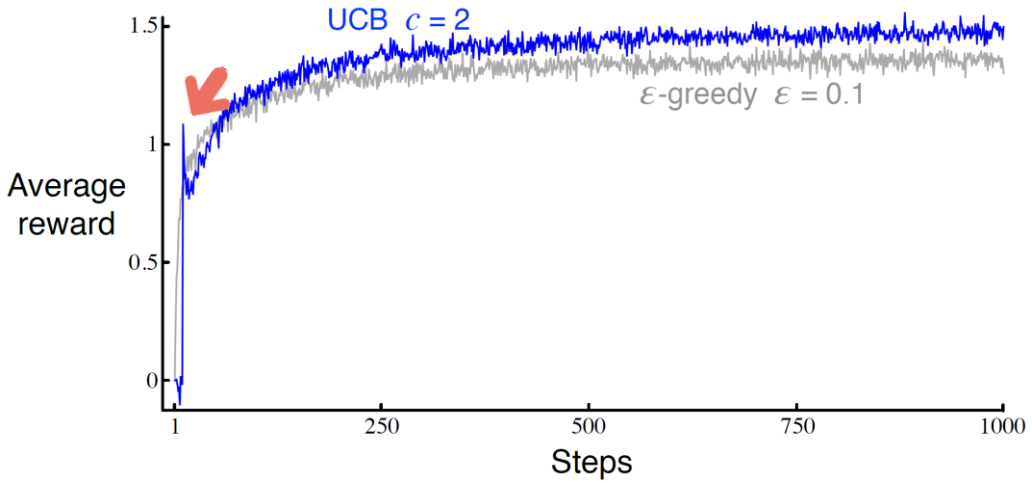
- **It is difficult to extend UCB beyond bandits to the more general RL settings**
  - More complex nonstationary problems
  - Another difficulty is dealing with large state spaces, particularly when using function approximation





1. The UCB algorithm shows a distinct spike in performance on the 11th step. Why is this?

2. Explain both why the reward increases on the 11th step and why it decreases on the subsequent



Gradient Bandit Algorithm

- ***Gradient Bandit Algorithms*** is an action selection approach based on ***learning a numerical preference*** for each action
- The ***larger*** the preference, the ***more often*** that action is taken
- Only the relative preference of one action over another is important

Definition: soft—max distribution (Gibbs or Boltzmann distribution)

If we denote the a numerical preference for each action by $H_t(a)$, then, the action probabilities is determined using the soft—max distribution as follows:

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

Where:

- $\pi_t(a)$ is the probability of taking action a at time t.
- Initially all action preferences are the same $(H_1(a) = 0, \text{a} \in \mathcal{A})$

Gradient Bandit Algorithm

Gradient Bandit Algorithm

- *action selection approach based on learning a numerical preference for each action*

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

where $H_t(A_t)$ demonstrates the preference values and updates as follows

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

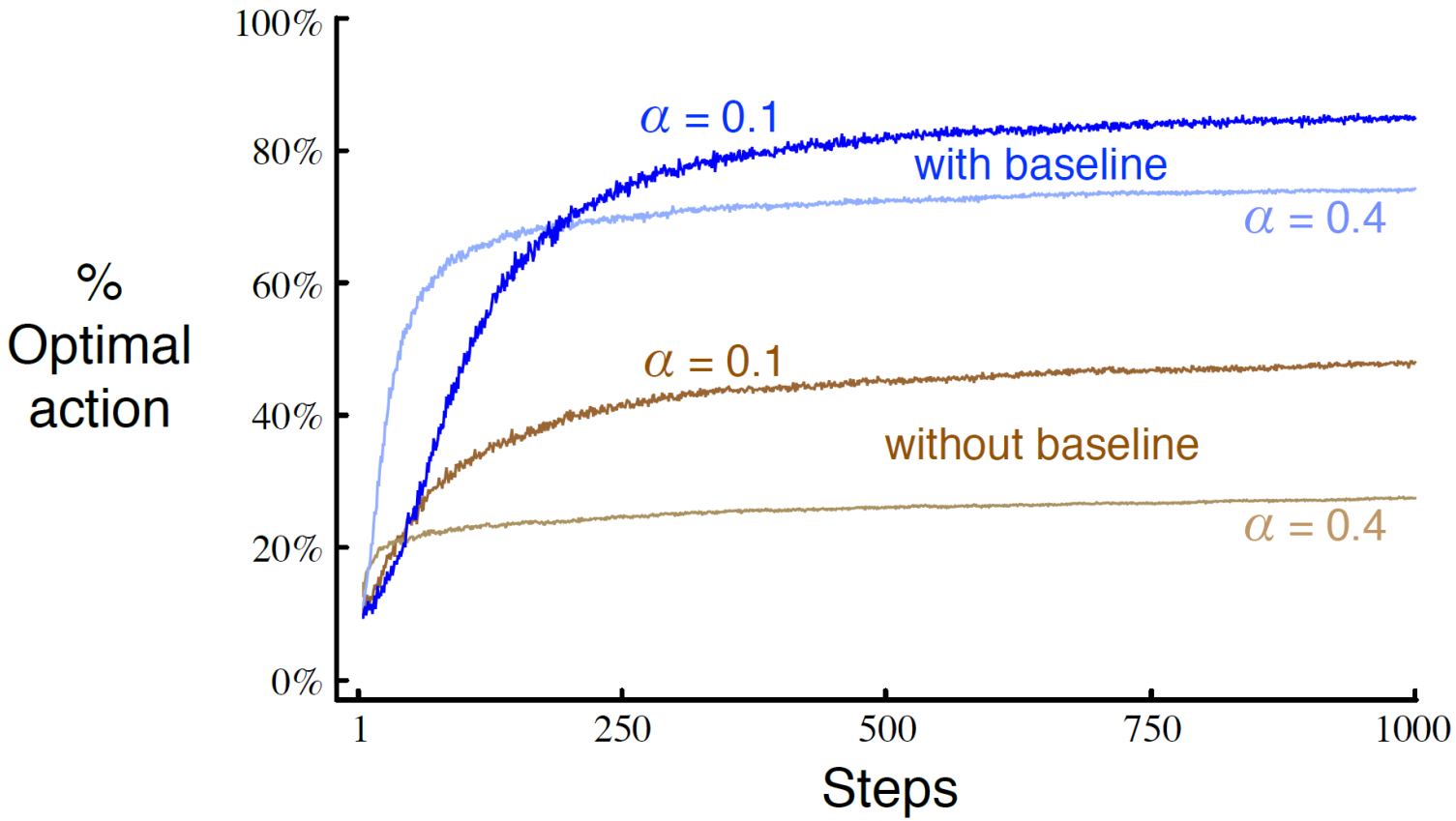
$$H_{t+1}(A_t) \doteq H_t(a) + \alpha(R_t - \bar{R}_t)\pi_t(a), \qquad \text{for all } a \neq A_t$$

Action is selected

Baseline

Where: $\bar{R}_t \doteq \frac{1}{t}\sum_{i=1}^{t} R_i$

Gradient Bandit Algorithm (results)



Results on a variant of the 10-arm bandit problem with mean reward=4

Some other Strategies

1: softmax

- Selects actions using a Boltzmann distribution:

$$\pi(a|s) = Pr\{A_t = a|S_t = s\} = \frac{e^{\frac{Q(s,a)}{\tau}}}{\sum_b e^{\frac{Q(s,b)}{\tau}}}$$

  - τ is a positive parameter called temperatur

2: ε-first or ε-decreasing

- Given $m \in \{1, \dots, {}^{T}/_{K}\}$
  - Draw each arm m times
  - Compute the empirical best arm

$$\hat{a} = argmax_a \, \hat{\mu}_a(K_m)$$

  - Keep playing this arm until round T

$$A_{t+1} = \hat{a} \text{ for } t \geq Km$$

  - Explore-then-Commit [O. Caelen and G. Bontempi, 2007]

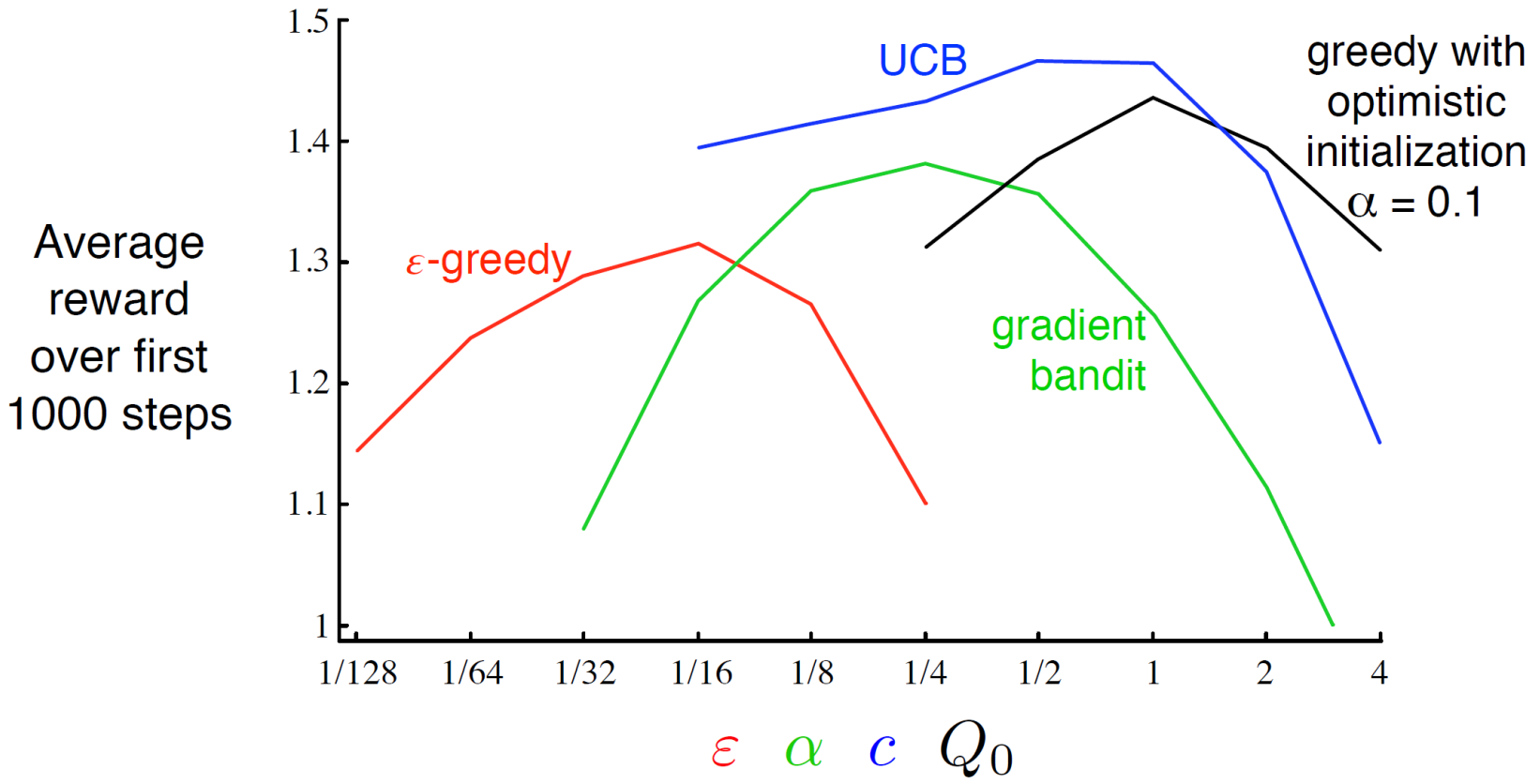
3: ε-PCSgreedy

- PCS: Probability of Correct Selection
  - Optimal exploration on the basis of PCS
  - The idea is that an effective exploration step should lead to the largest increase of the probability PCS of correctly selecting the best arm.
    - [J. Vermorel and M. Mehryar, 2005]

4: ε -greedy VDBE-Boltzmann

- Value-Difference Based Exploration (VDBE)
- Extends the ε-greedy method by controlling a state-dependent exploration probability
- $\varepsilon(s)$, in dependence of the value-function error instead of manual tuning
  - [M. Tokic,, 2010]

Comparison of Bandit Algorithms



Contextual bandits

- **Imagine you are solving a k-armed bandit problem, but the problem changes (randomly) each time step**
  - (you get one of n bandit problems)
  - You can recognize what bandit problem you are in
- **This is the associative or contextual case**

- **Note that this is still not the full RL problem (the action choice doesn't affect the state)**

Summary

- **We have investigated a very simple problem setting, with only one state and already we ran into**
  - learning from feedback rather than instruction
  - explore-exploit dilemma & different ways to select actions
  - computational efficiency issues
  - tracking and non-stationarity
  - bias and optimism in the face of uncertainty
- **Problems have significant industrial applications**
  - (online advertising, Yahoo!)

Summary

- **These are all simple methods**
  - but they are complicated enough—we will build on them
  - we should understand them completely
  - there are still open questions

- **Our first algorithms that learn from evaluative feedback**
  - and thus must balance exploration and exploitation

- **Our first algorithms that appear to have a goal**
  - that learn to maximize reward by trial and error

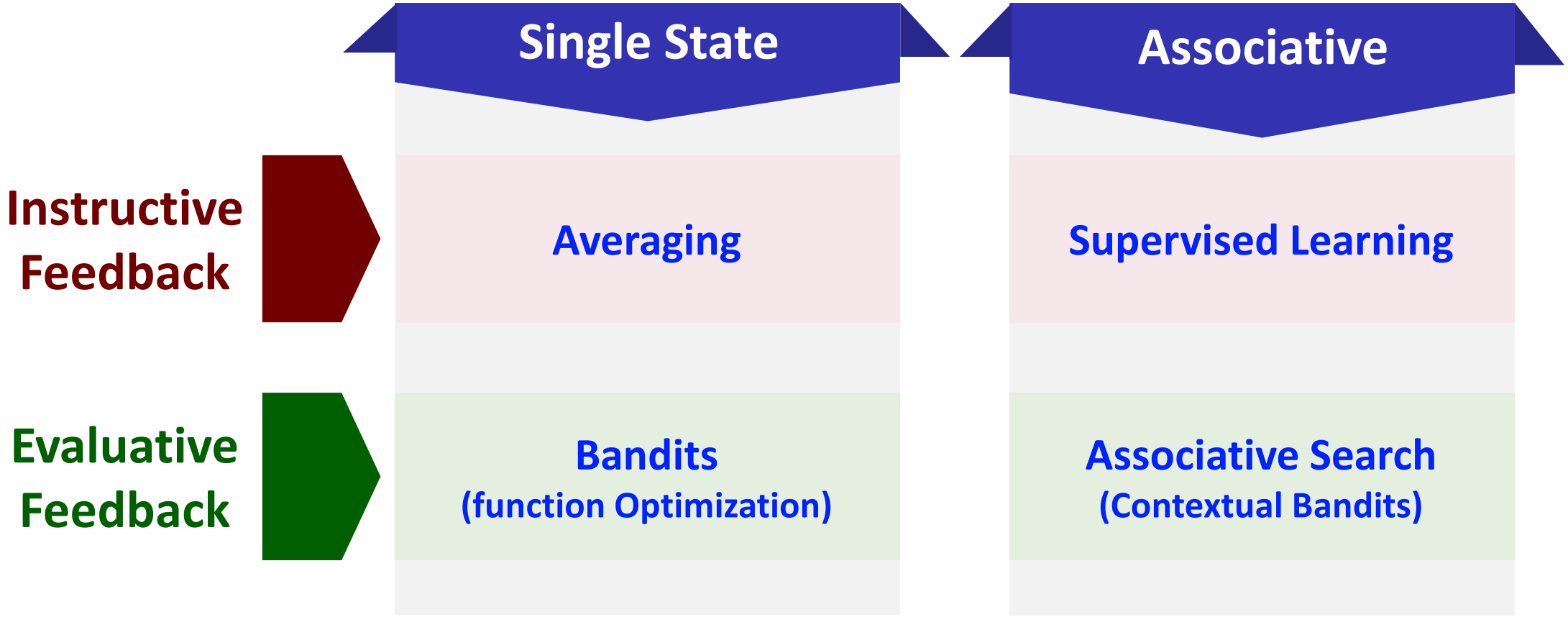
Summary

- **Action values**
  - Commonly used throughout RL (e.g., Q-learning)

- **• UCB**
  - Similar ideas used for exploration bonuses in RL and
  - planning (e.g., Monte Carlo tree search)

- **Gradient methods**
  - Commonly used throughout RL (e.g., policy gradient methods)

Problem Space



References

- 1] R. S. Sutton and A. G. Barto, Reinforcement learning: An introduction. MIT press, 2018.
- [2] A. Slivkins, “Introduction to multi-armed bandits,” arXiv preprint arXiv:1904.07272, 2019.
- [3] W. R. Thompson, “On the likelihood that one unknown probability exceeds another in view of the evidence of two samples,” Biometrika, vol. 25, no. 3/4, pp. 285–294, 1933.
- [4] G. Bresler, D. Shah, and L. F. Voloch, “Collaborative filtering with low regret,” in ACM SIGMETRICS Performance Evaluation Review, 2016, vol. 44, pp. 207–220.
- [5] M. Tokic, “Adaptive ε-greedy exploration in reinforcement learning based on value differences,” in Annual Conference on Artificial Intelligence, 2010, pp. 203–210.
- [6] J. Vermorel and M. Mehryar, “Multi-Armed Bandit Algorithms and Empirical Evaluation,” in European conference on machine learning, 2005.