Multi-armed Bandits

Reinforcement learning – LM Artificial Iintelligence
(2022-23)

Alberto Castellini
University of Verona

Summary

- Introduction
- K-armed Bandit Problem
- Action-value Methods
- The 10-armed Testbed
- Incremental Implementation
- Optimistic Initial Values
- Upper Confidence Bound (UCB) action selection
- Gradient Bandit Algorithms
- Associative Search (Contextual Bandits)

Introduction

Introduction

Main feature **distinguishing RL** from other types of learning:

- **RL** uses training information to **evaluate** agent's **actions**
- **Other learning methods instruct** the agent providing examples of **correct actions**

- **Evaluative feedback: how good** the action taken was (no info about **best/worst** actions)
- **Instructive feedback:** indicates the **correct action** independently of actions actually taken

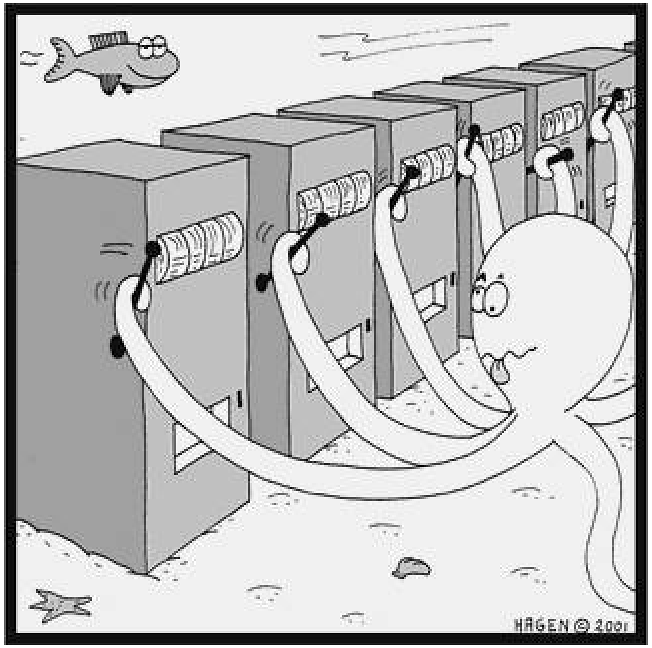
- Active **exploration** is also needed by RL to search good behaviors

- **This lecture:** evaluative aspect of RL in the **simplified** setting of **single state** (**nonassociative setting**)
  - **k-armed bandit problem**
  - related learning methods (extended in next lectures to RL setting)

K-armed Bandit Problem

K-armed Bandit Problem

1-armed bandit



k-armed bandit

Problem:

- Repeatedly **choose among k** different **options** (actions)

- After each choice you **receive** a numerical **reward** chosen from a **stationary probability distribution** depending on the **action selected**

- **Objective:** maximize the expected total reward over some time period (e.g., 1000 action selections)

K-armed Bandit Problem

- Each of the k actions has an **expected (mean) reward**, called **value of the action**

- If A_t is the **action** selected at step *t* and R_t is the corresponding **reward** then the **expected reward** given that action *a* was selected is:

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

- If you **know the value of each action** then it is trivial to solve the k-armed bandit problem: **always select the action with the highest value**

- We assume not to know the action values but to estimate them

- The **estimated value of action a at time t** is $\mathbf{Q_t(a)}$
- We would like $Q_t(a)$ to be as close as possible to $q_*(a)$

Exploration-exploitation dilemma

- Given estimates of all action values, we call **greedy action** the action with the **largest estimated value**

- When you **choose the greedy action** you **exploit** your current knowledge of action values
- When you **choose nongreedy actions** you **explore** action values to get new knowledge on them

- **Exploitation** is the **best** thing to do to maximize the expected reward on a **one step horizon**
- **Exploration** may produce **greater** total reward in the **long run**

- **Dilemma:** should I explore or exploit? There is a **conflict**
- There are sophisticated **methods** for **balancing** exploration and exploitation but most of them make strong assumptions

Action-value Methods

Action-value methods

- Methods for **estimating** action values and **selecting** optimal actions

- Since the **value** of an action is the **mean reward** obtained when the action is selected, a natural way to estimate it is by **averaging** the rewards actually received:

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

where $\mathbb{1}_{predicate}$ is 1 if predicate is true, 0 otherwise.

- When the denominator goes to infinity, by the law of large numbers, $Q_t(a)$ converges to $q_*(a)$.

- We call this the **Sample-Average Method**

Action-value methods

- **How the estimate provided by the sample-average method might be used to select actions?**

  1. Simplest rule: select one of the actions with the **highest estimated value** (**greedy action selection**)

$$A_t \doteq \underset{a}{\arg\max}\, Q_t(a)$$

- Greedy action selection **always exploits** current knowledge, hence it **maximizes immediate reward**

  2. Alternative rule: behave greedily most of the times but **with small probability** ε select randomly from nongreedy actions (ε **-greedy selection**)

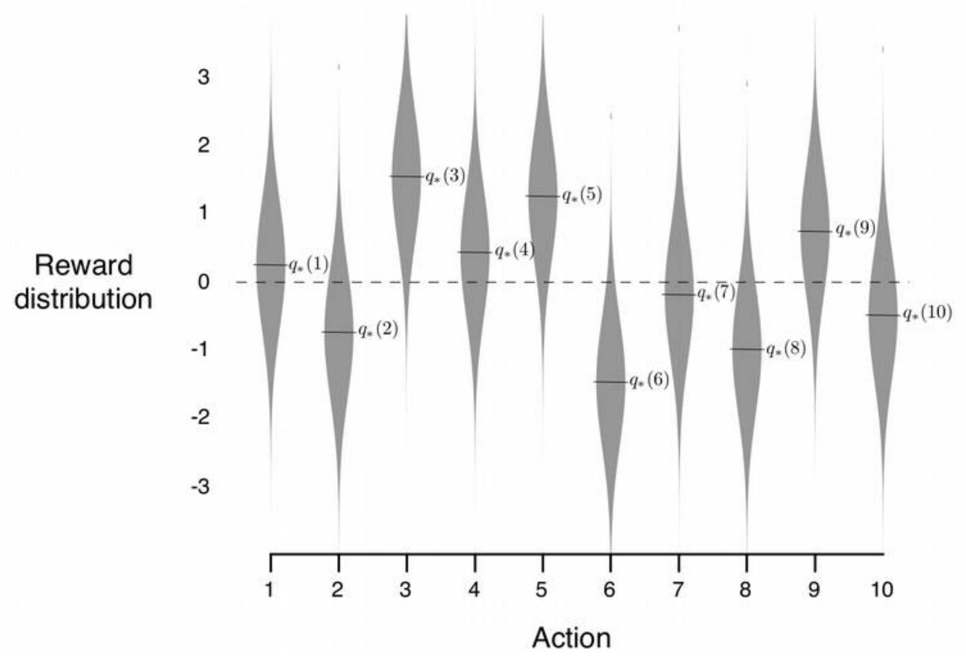
- It performs **exploration**. As the number of steps increases, every action will be **sampled infinite number of times** ensuring that **Q_t(a) converges to q_*(a)**

The 10-armed Testbed

The 10-armed testbed

Goal: to compare performance of different learning methods

- 2000 randomly generated instances of the k-armed bandit problem
- Number of actions: k=10
- For **each** bandit problem **action values** $q_*(a)$ are selected according to a **normal** (Gaussian) **distribution** with mean **0** and variance **1**

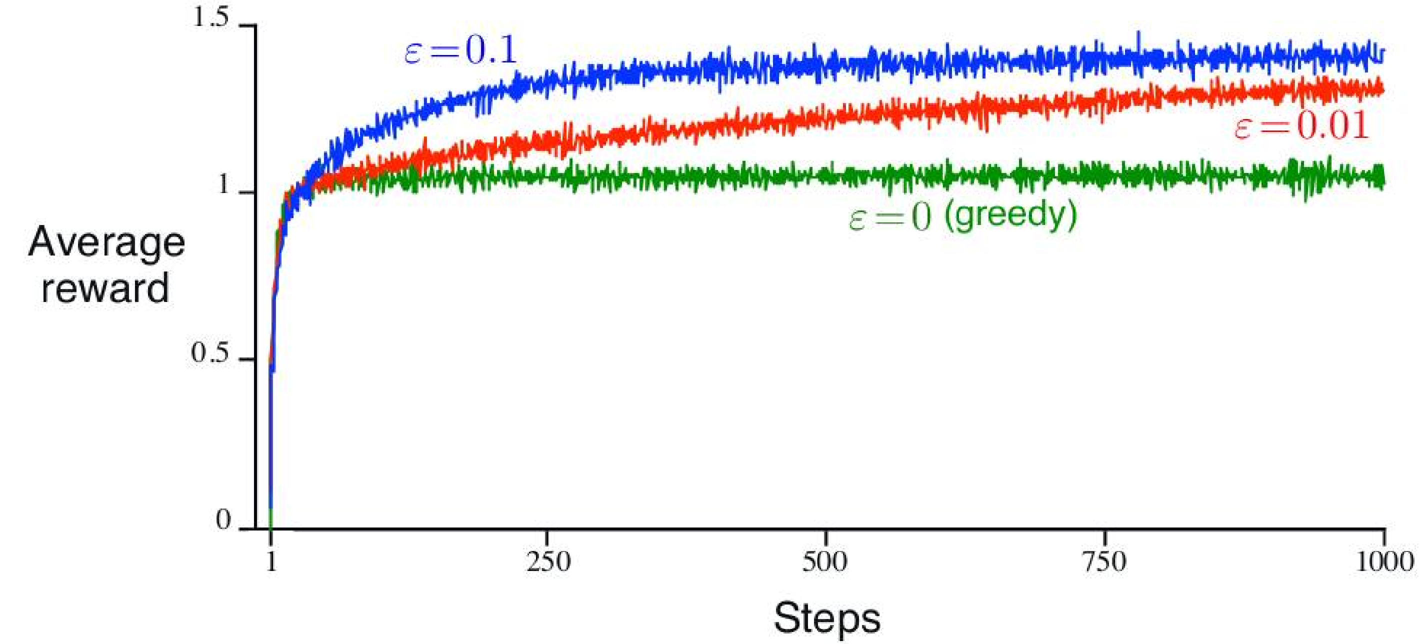


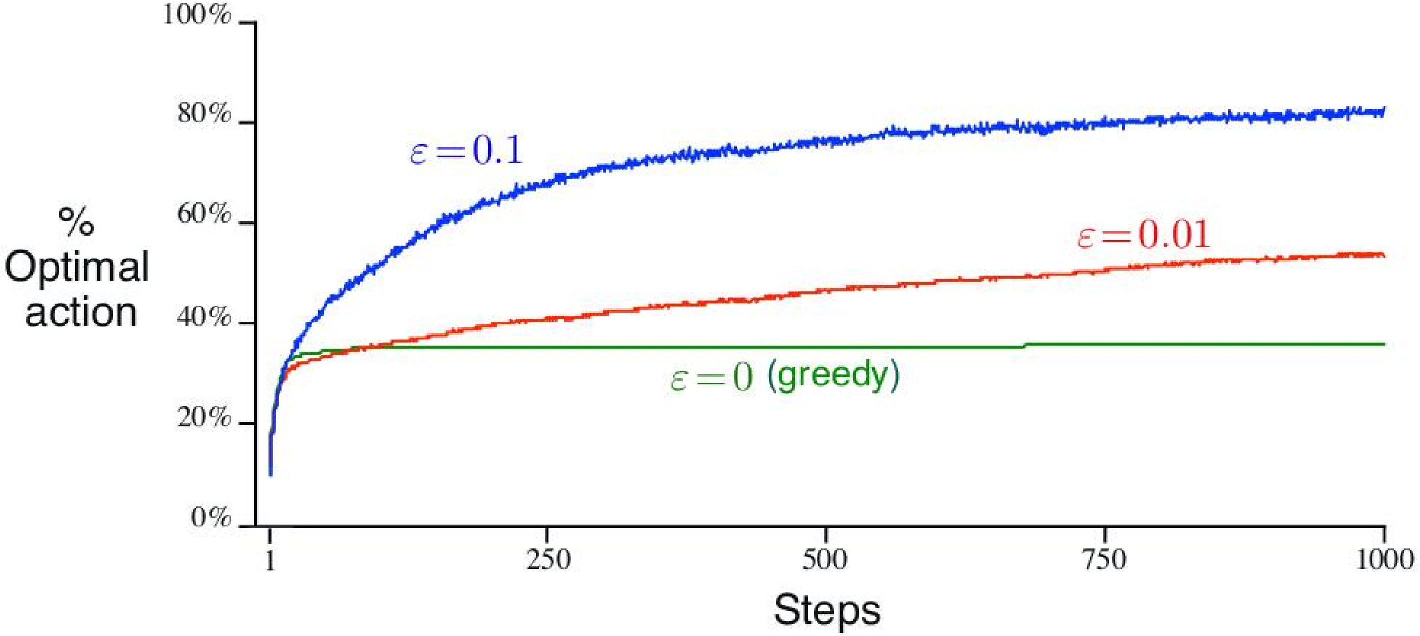
- When an **action A_t is selected** at time *t* the reward R_t is selected from a **normal distribution** with mean $\mathbf{q_*(A_t)}$ and variance **1** (grey plots)

The 10-armed testbed

- To **test a learning method** we store its rewards over **1000 steps (run)**
- Then we repeat this for **2000 independent runs** (each run refers to a different instance of the 10-armed bandit problem (i.e., different $q_*(a)$ values)
- Finally, we **average** rewards of all runs at the same time *t*

Comparing greedy, 0.01-greedy and 0.1-greedy methods





Comparing greedy, 0.01-greedy and 0.1-greedy methods

- With **larger reward variance** (e.g., 10 instead of 1) ε-greedy methods should perform even better than the greedy method

- If the **reward variances are zero** then a single try is enough to discover action values. In this case **greedy methods perform best** because they soon find the best action and then never explore

- If the **bandit tasks** were **nonstationary** (i.e., true values $q_*(a)$ change over time) then exploration is needed also in the deterministic case (zero variance)

- **Nonstationarity is the case most commonly encountered in RL**

Incremental Implementation

Incremental Implementation

- Action value **estimations $Q_t(a)$** are computed by **averaging** observed rewards in **action-value methods**

- **Question: how can we compute/update these averages efficiently (i.e., constant memory and constant per-time-step computation)?**

- Let's focus on a **single action *a***. Let R_i be the reward received at step i selecting action a, and Q_n the estimated value of action a after n-1 selections of this action

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

- By maintaining a **record of rewards** we can sum them up and **divide** by the current number of selections, **at each update**
- **Memory** and **computational** requirements grow **linearly** with the number of rewards

Incremental Implementation

- It is easy to devise more efficient **incremental formulas**

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

- It requires **memory** only for Q_n and *n*, and only a **small computation** (three mathematical operations) at each step

Incremental Implementation

- This **update rule** is of a form that **occurs frequently in RL**. The general form is:

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

- ***Target-OldEstimate*** is an **error in the estimate** which is reduced taking a **step** toward the *Target*

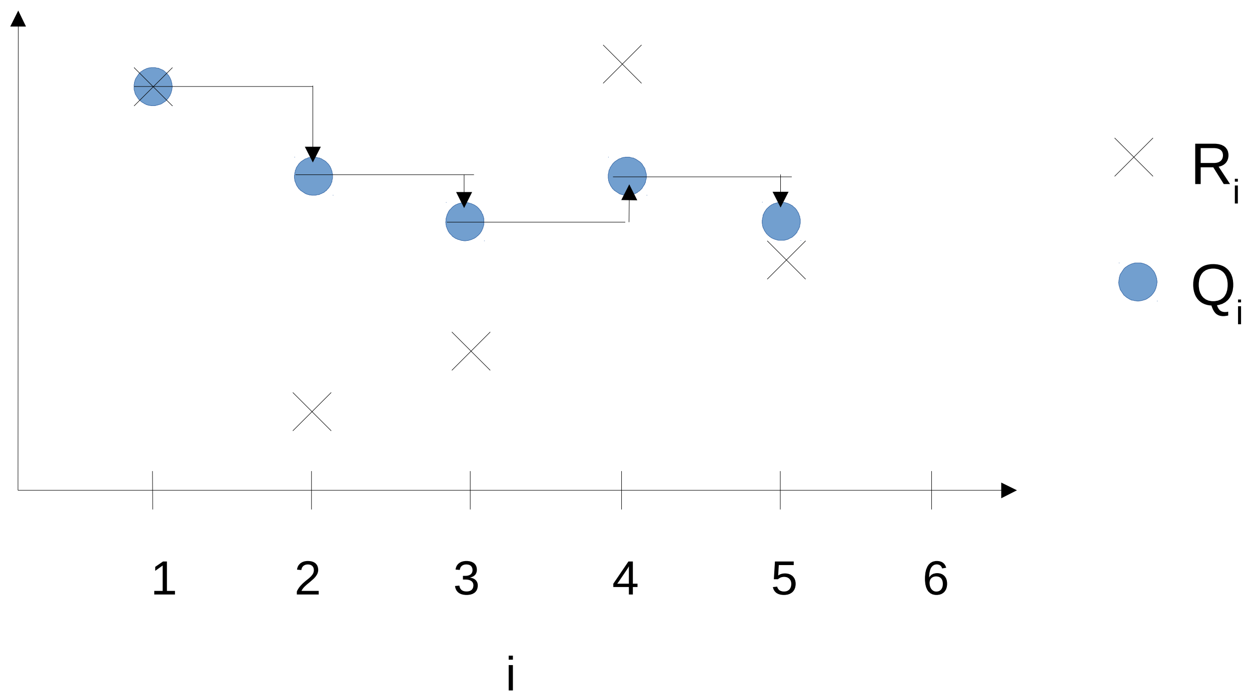
- The *Target* is presumed to indicate a **desirable direction** in which to move

- But the *Target* is a **noisy signal** (e.g., *n*th reward)

- The ***StepSize*** parameter in the incremental average computation is **1/n**, hence it **decreases** at each step. This parameter, called α in general, can get also **other values**

Incremental Implementation

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



Complete bandit algorithm

A simple bandit algorithm

Initialize, for $a = 1$ to k:

$Q(a) \leftarrow 0$

$N(a) \leftarrow 0$

Loop forever:

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

$R \leftarrow bandit(A)$

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

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

Tracking a Nonstationary Problem

Tracking a Nonstationary Problem

- The **averaging methods** discussed above are appropriate for **stationary** bandit problems (reward probabilities fixed over time)

- **RL** problems are always **nonstationary**

- In these cases it makes sense to give **more weight** to **recent rewards** than to long-past rewards → **Constant size parameter**

- The previous **incremental update rule** for estimating value Q_n from the n-1 last rewards becomes

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

- with $\alpha \in (0, 1]$ and **constant**

Tracking a Nonstationary Problem

- This recursive formula can be rewritten as

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

- This is a weighted average because

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

- $(1-\alpha)$ is less than 1, thus the weight given to R_i **decreases** as *n* increases (weight decreases exponentially according to the exp of 1-α)

Tracking a Nonstationary Problem

- Sometimes it is convenient to **vary the step-size parameter** from step to step.

- Let $\alpha_n(a)$ be the parameter used to process the reward received **after the *n*th selection of action *a***.

- E.g., in the **sample-average method** $\alpha_n(a) = \frac{1}{n}$ and the value Q_n is **guaranteed to converge to** the **true action value** by the law of large numbers

- **Problem: convergence is not guaranteed for all sequences of the step-size parameter**

Tracking a Nonstationary Problem

- Stochastic approximation theory provides **conditions required to assure convergence with probability 1**:

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

- **First condition:** guarantees that the steps are large enough to eventually overcome any initial condition or random fluctuations

- **Second condition:** guarantees that eventually the steps become small enough to assure convergence

- **Both conditions are met by** $\boxed{\alpha_n(a) = \frac{1}{n}}$ ($\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$ Euler's proof of the Basel problem)

- For $\boxed{\alpha_n(a) = \alpha}$ with constant α the **second condition is not met** → The estimate **never completely converges** but continue to vary in response to most recently received rewards (**desirable in nonstationary environments**)

Optimistic Initial Values

Optimistic Initial Values

- All methods discussed so far depend on (i.e., they are **biased** by) **initial action-value estimates**, $Q_1(a)$

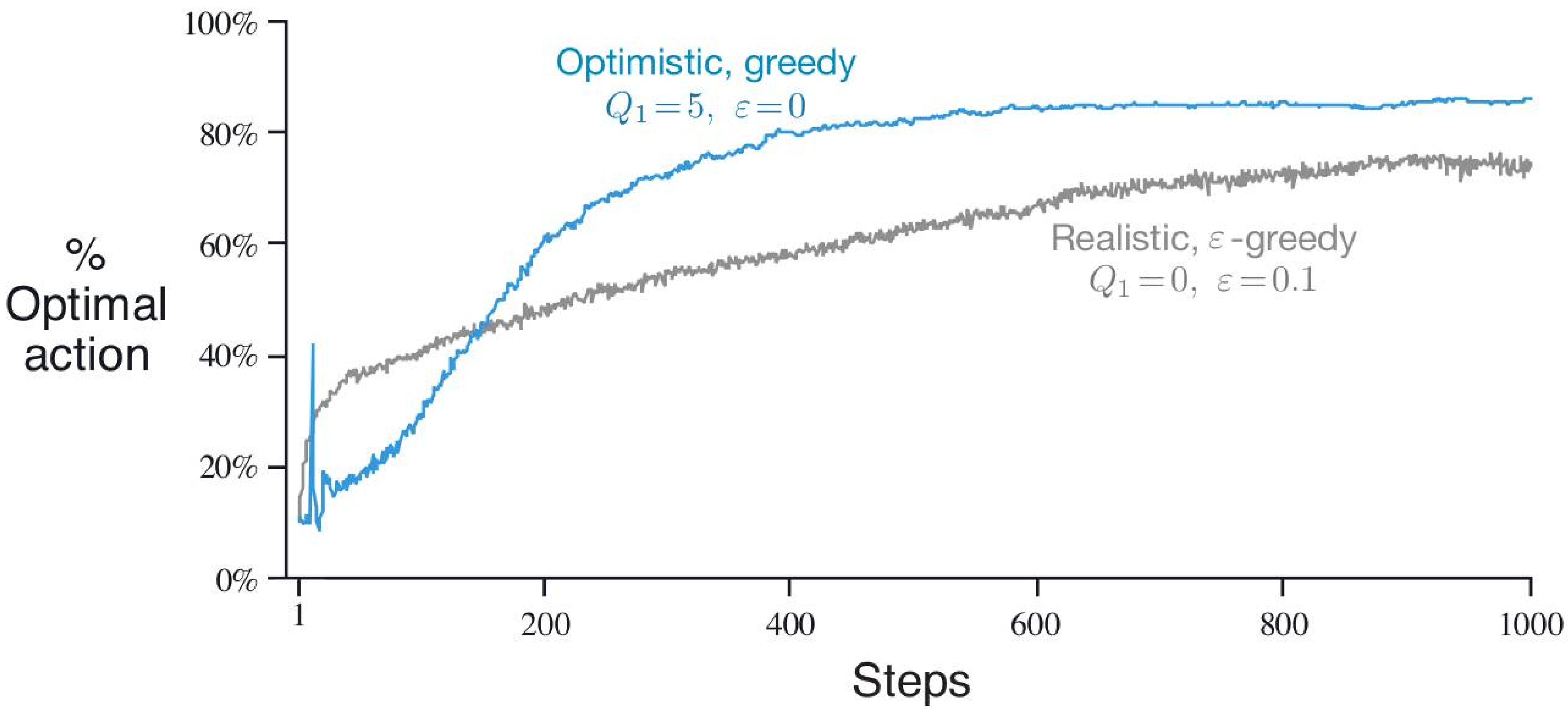
- In practice, this **bias** is not a problem and can sometimes be helpful
  - **Downside:** biases need extra parameters
  - **Upside:** biases provide an easy way to provide **prior knowledge** about **expected levels of reward** from different actions

- **Optimistic initial values (i.e., $Q_1(a)$ values larger than expected) are a simple way to encourage exploration**

Example: 10-armed testbed with $\mathbf{Q_1(a)=5}$ instead of $Q_1(a)=0$

- Whichever action is selected, the reward is less than the starting estimate → **The greedy learner switches to other actions thus performing exploration**

Optimistic Initial Values: application to 10-armed bandit



Optimistic initial values:

- simple trick
- effective on stationary problems
- **Not suited for nonstationary problems** because its drive for **exploration** is **inherently temporary**

Upper Confidence Bound (UCB) action selection

Upper-Confidence-Bound (UCB) Action Selection

- Exploration is needed: because of the uncertainty about action-value estimates

- ε **-greedy** action selection forces the non-greedy actions to be tried **indiscriminately** (no preference for actions that are nearly greedy or particularly uncertain)

- **Better** to **select** among **non-greedy actions** according to their **potential** for actually being **optimal**
  - **How close** their estimates are to being **maximal**
  - **Uncertainty** in those **estimates**

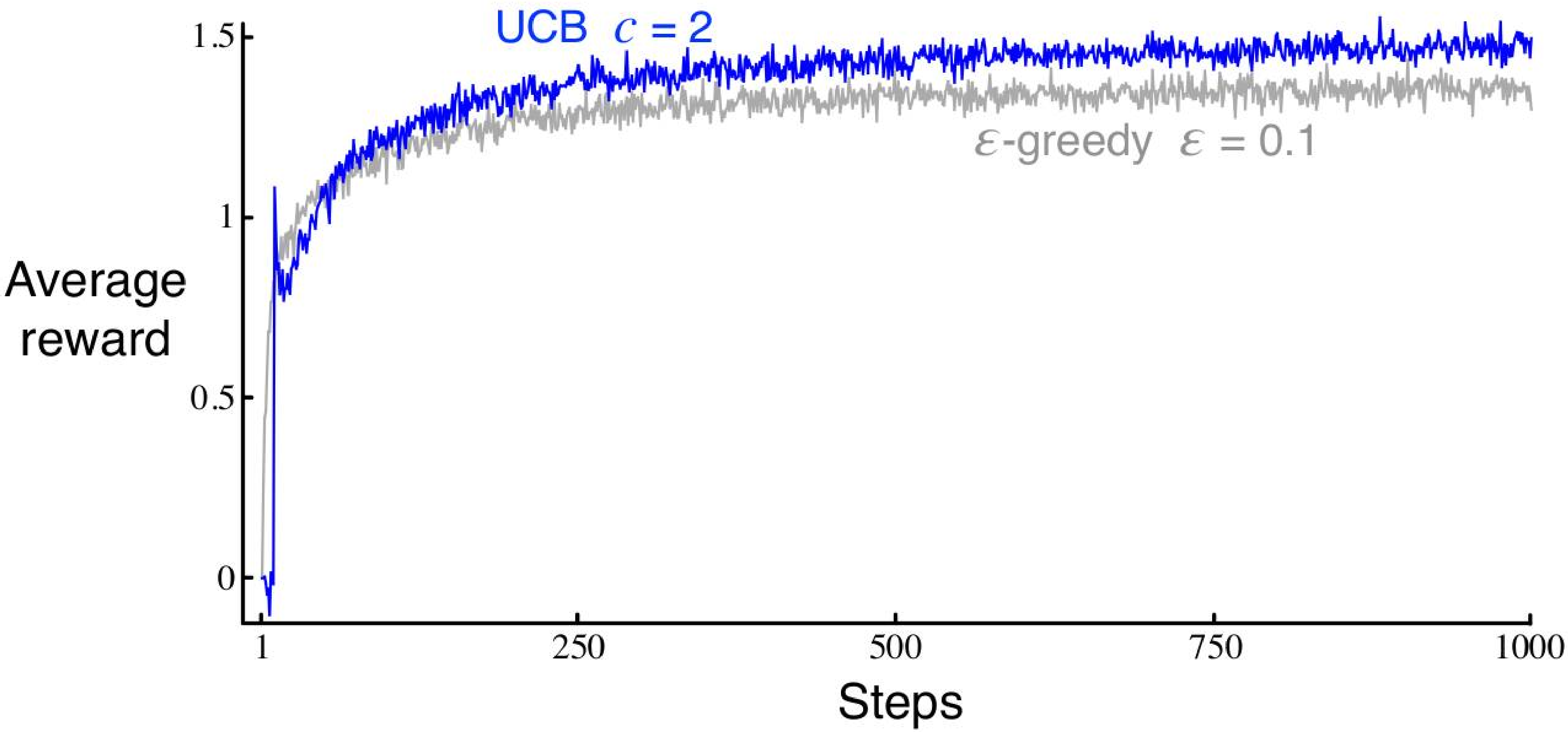
Upper-Confidence-Bound (UCB) Action Selection

- **UCB action selection:**

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

- where
  - t is the total number of action-selections performed so far
  - $N_t(a)$ is the number of times action a has been selected prior to time t
  - $c>0$ controls the degree of exploration
  - for $N_t(a)=0$, a is considered an action with maximal reward (i.e., to be tested)

UCB: application to 10-armed bandit



- UCB performs well
- UCB is **more difficult than ε-greedy** to **extend** beyond bandits to the more general **RL setting** (see lecture about model-based RL).

Gradient Bandit Algorithms

Gradient Bandit Algorithms

- **Idea: learn a numerical preference $H_t(a)$ for each action *a* instead of estimating action values**

- The **larger** the preference, the **more often** the action is taken

- Only the **relative preference** of one action over another is **important**

- Action **probabilities** are determined according to a **soft-max distribution**:

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

- **Notation**: $\pi_t(a)$ probability of taking action *a* at time *t*

Learning algorithm:

- **Initially** all action preferences are the **same** (e.g., $H_1(a)=0$): all actions have equal probability of being selected

- **At each step**, after selecting action A_t and receiving the reward R_t, the **action preferences** are **updated** by the following **rule** based on **stochastic gradient ascent**:

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

where

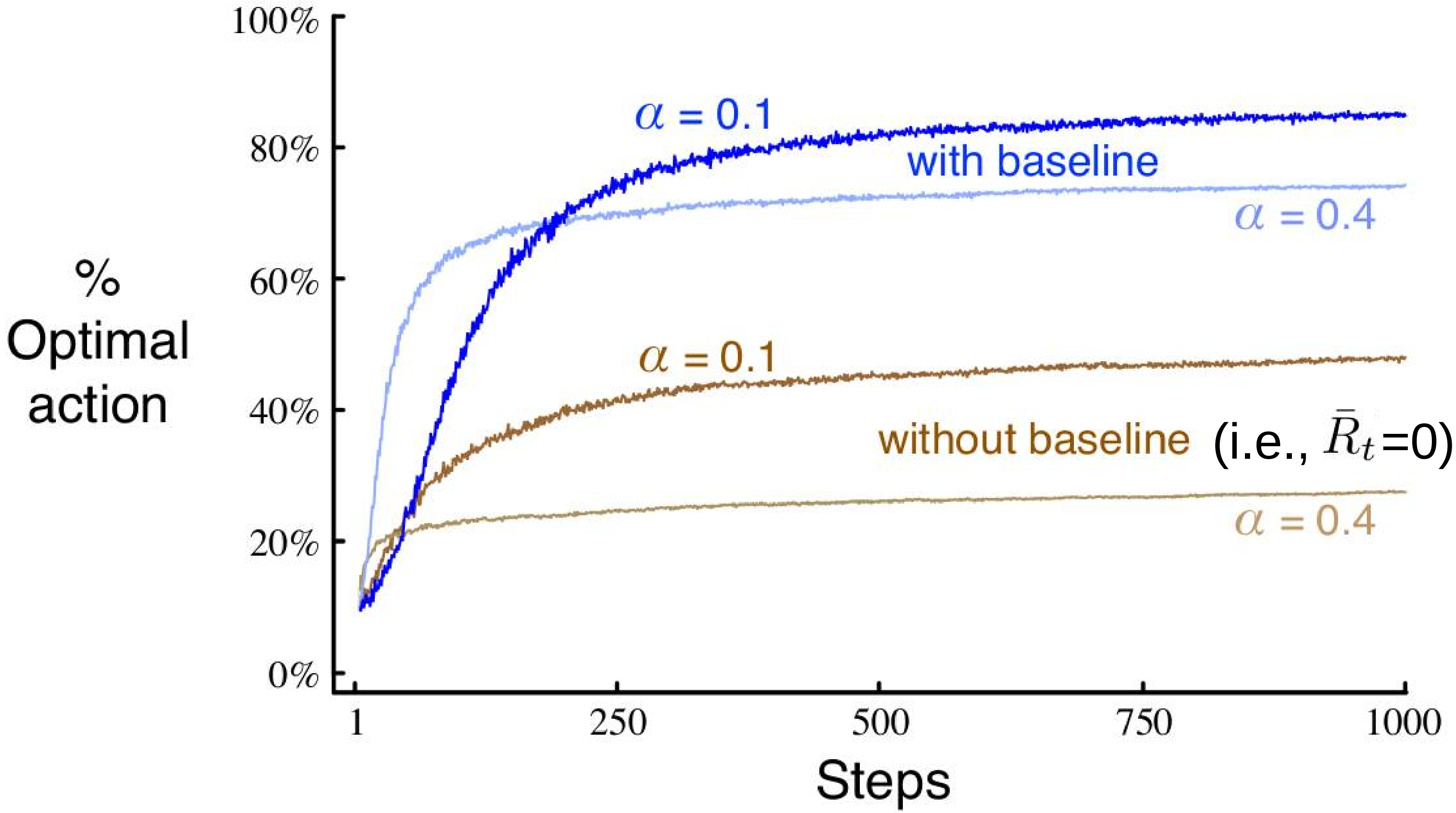
- $\alpha > 0$: step-size parameter
- $\bar{R}_t$: **average** of all the **rewards** received so far (from all actions) and including time *t* (which can be computed incrementally as seen before). This term serves as a **baseline** with which the reward is compared

Gradient Bandit Algorithm

Idea:

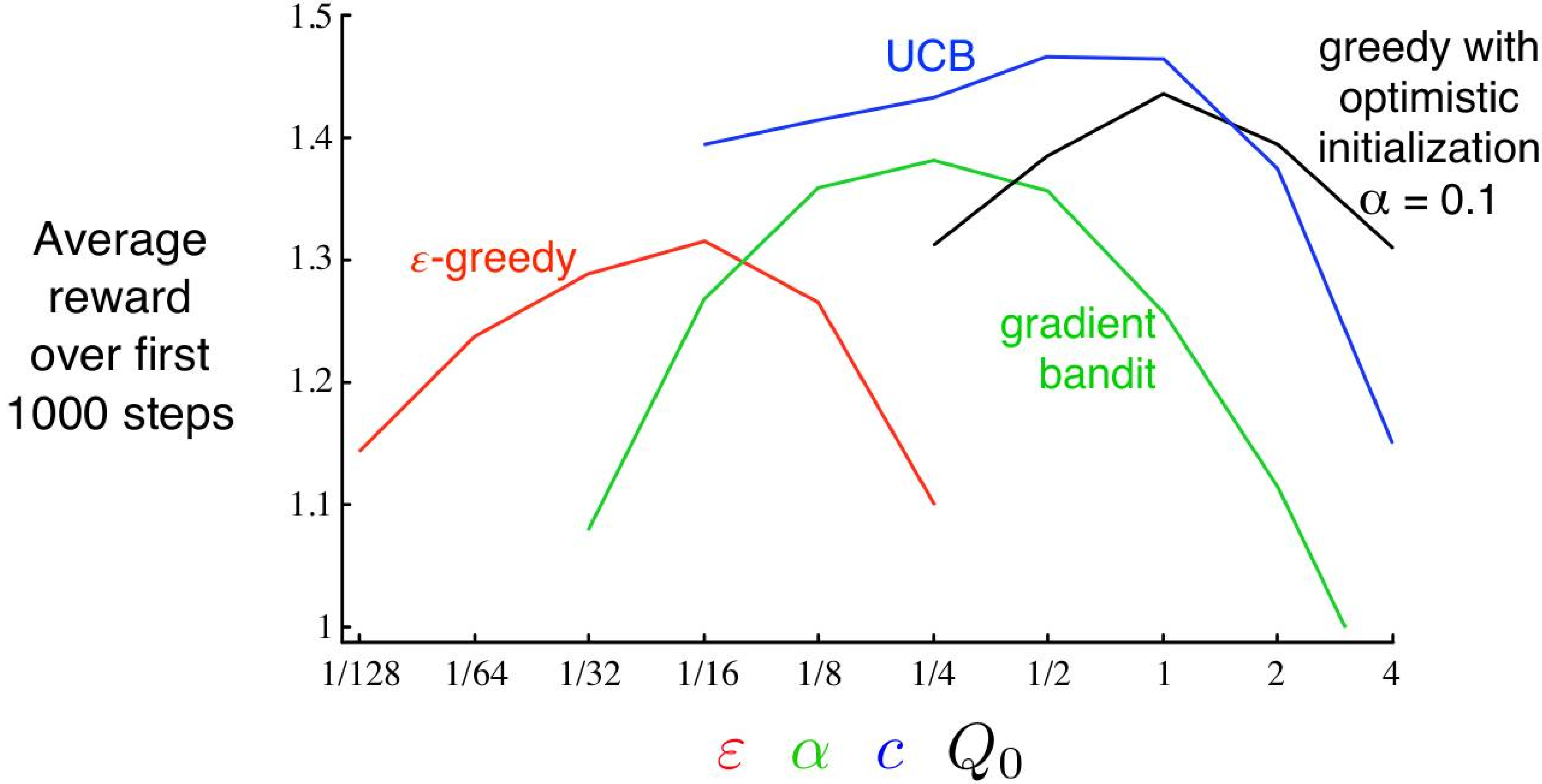
- If the **reward** is **higher than the baseline** then the **probability** of taking A_t in the future is **increased**
- If the **reward** is **below baseline** then the **probability** is **decreased**
- The probabilities of **non-selected** actions move in the **opposite direction**

Gradient Bandit Algorithm: application to 10-armed bandit variant



- Variant of the 10-armed testbed in which the true expected rewards were selected according to a normal distribution with a **mean of +4** instead of 0
- The **shift has no effect** on the gradient bandit algorithm because of the **reward baseline term** that instantaneously adapts to the new level

Bandit algotithms: Performance comparison



Associative Search (Contextual Bandits)

Associative Search (Contextual Bandit)

- From **bandit problems** (single state) to **RL problems** (multiple states with different action values)

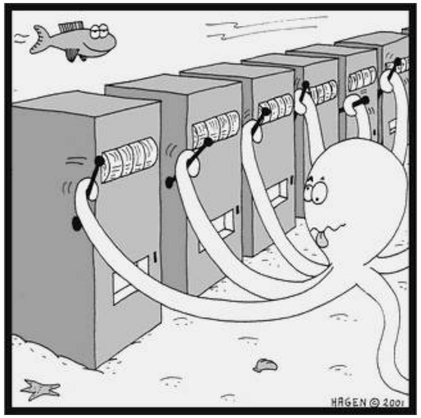
- **Bandit problems** are non-associative: no need to associate different **actions** with different **situations**

- In **general** RL **problems** there is **more than one situation**

- **Goal of RL:** learn a mapping from **situations** to **actions** that are best in those situations (policy)

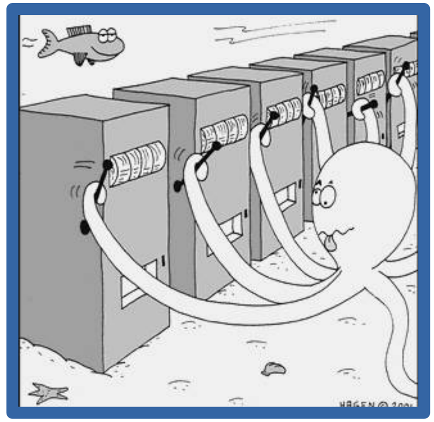
Associative Search (Contextual Bandit)

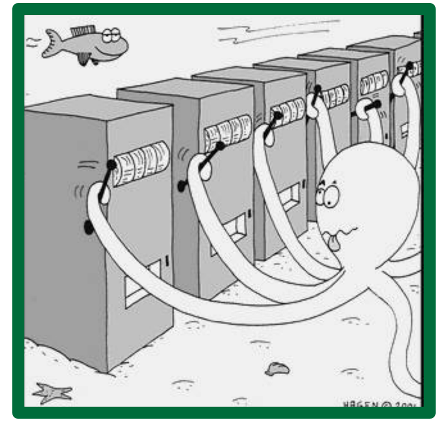
Non-associative multi-armed bandit

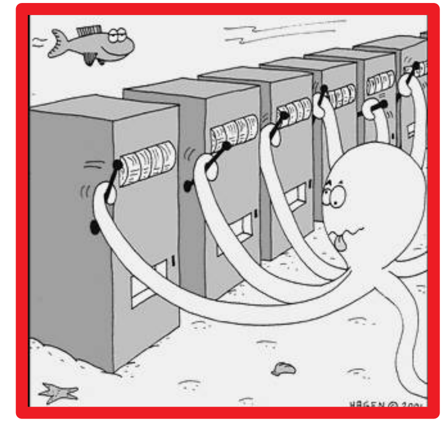


Associative multi-armed bandit (contextual bandit)

Different action values in different situations







Random transitions between different multi-armed bandits

A_1

A_2

A_3

...

$A_1 \rightarrow A_2 \rightarrow A_3 \rightarrow A_4 \rightarrow A_5 \rightarrow A_6$

... ...

Associative search task: involves both trial-and-error learning (as in nonassociative tasks) and action-situation association

References

- R. S. Sutton, A. G. Barto. Reinforcement learning, An Introduction. Second edition. Chapter 2