Introduction to Reinforcement Learning

ML on quantum and classical data

Gorka Muñoz-Gil (**ICFO**)

Master in Photonics (UPC-UAB-UB-ICFO), 2018-2019

Recap Supervised vs. Unsupervised Learning

Supervised learning

Data is labeled: photo + some information of what is on it (cat or dog?)

Examples:

- Regression methods (SVM,...)
- Image classification with NN

Unsupervised learning

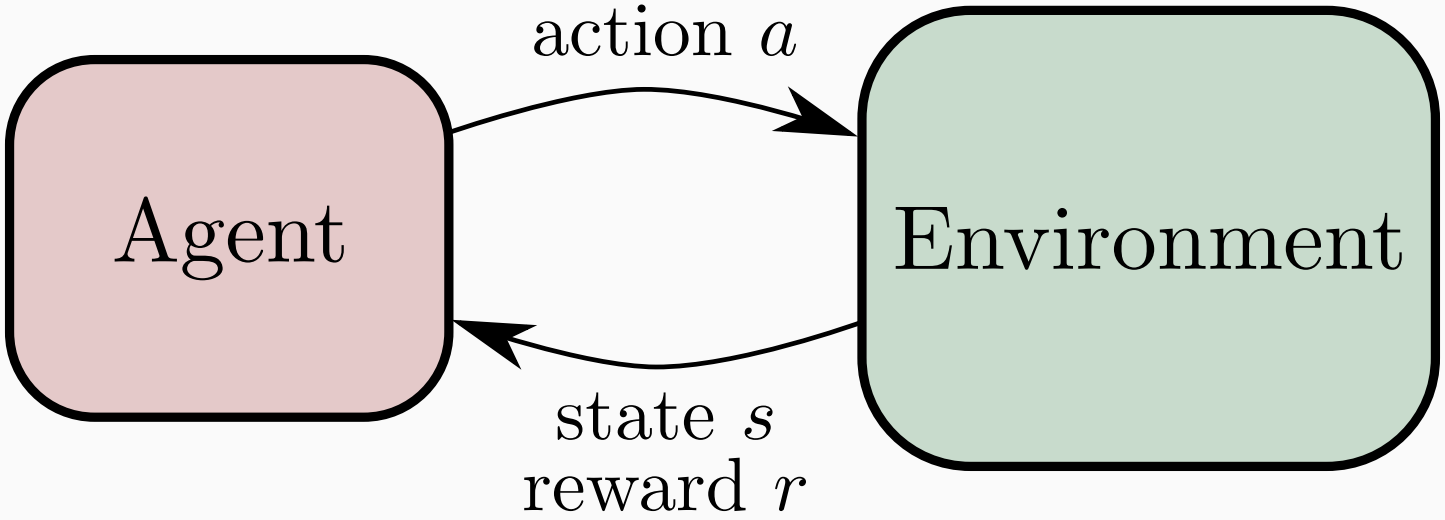
Data is not labeled, we want to find patterns or reconstruct some samples without extra information.

Examples:

- Autoencoders
- Boltzmann machines (energy models)
- Clusterization algorithms (tSNE,...)

Reinforcement Learning

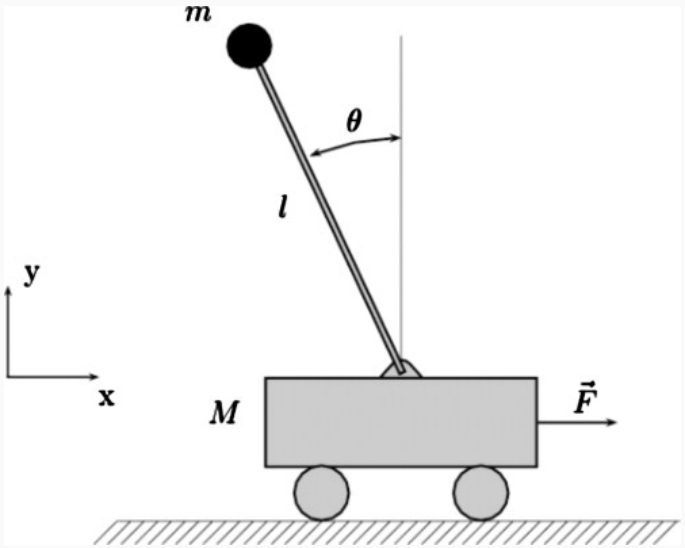
Nor supervised nor unsupervised: a mix of both!



The goal of the agent is to maximize the reward over long interactions with the environment.

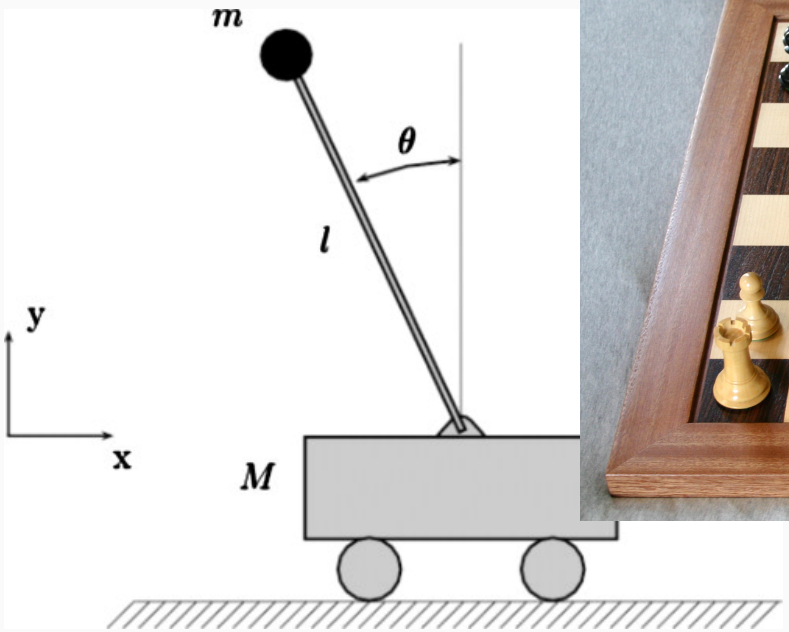
What is an environment?

As simple as



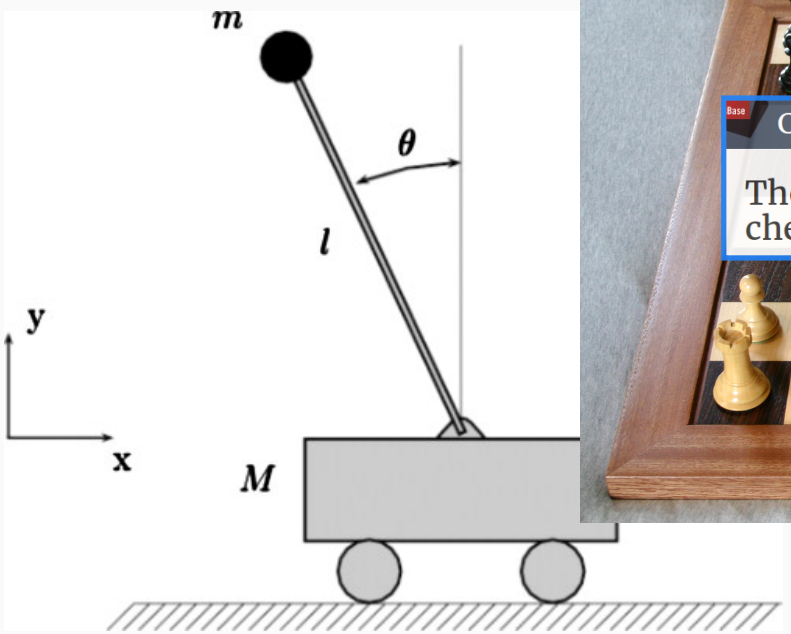
What is an environment?

More complex



What is an environment?

More complex

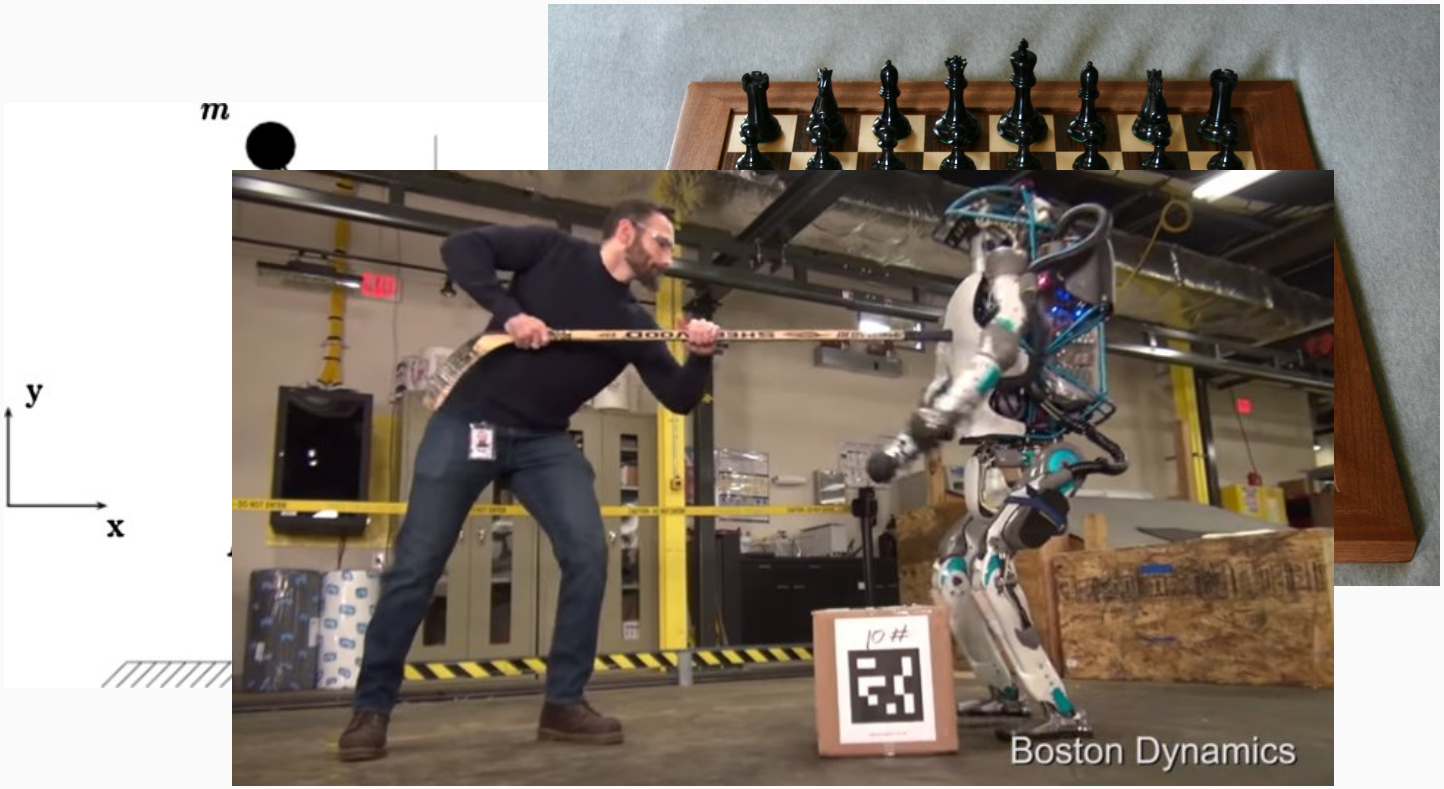


What is an environment?

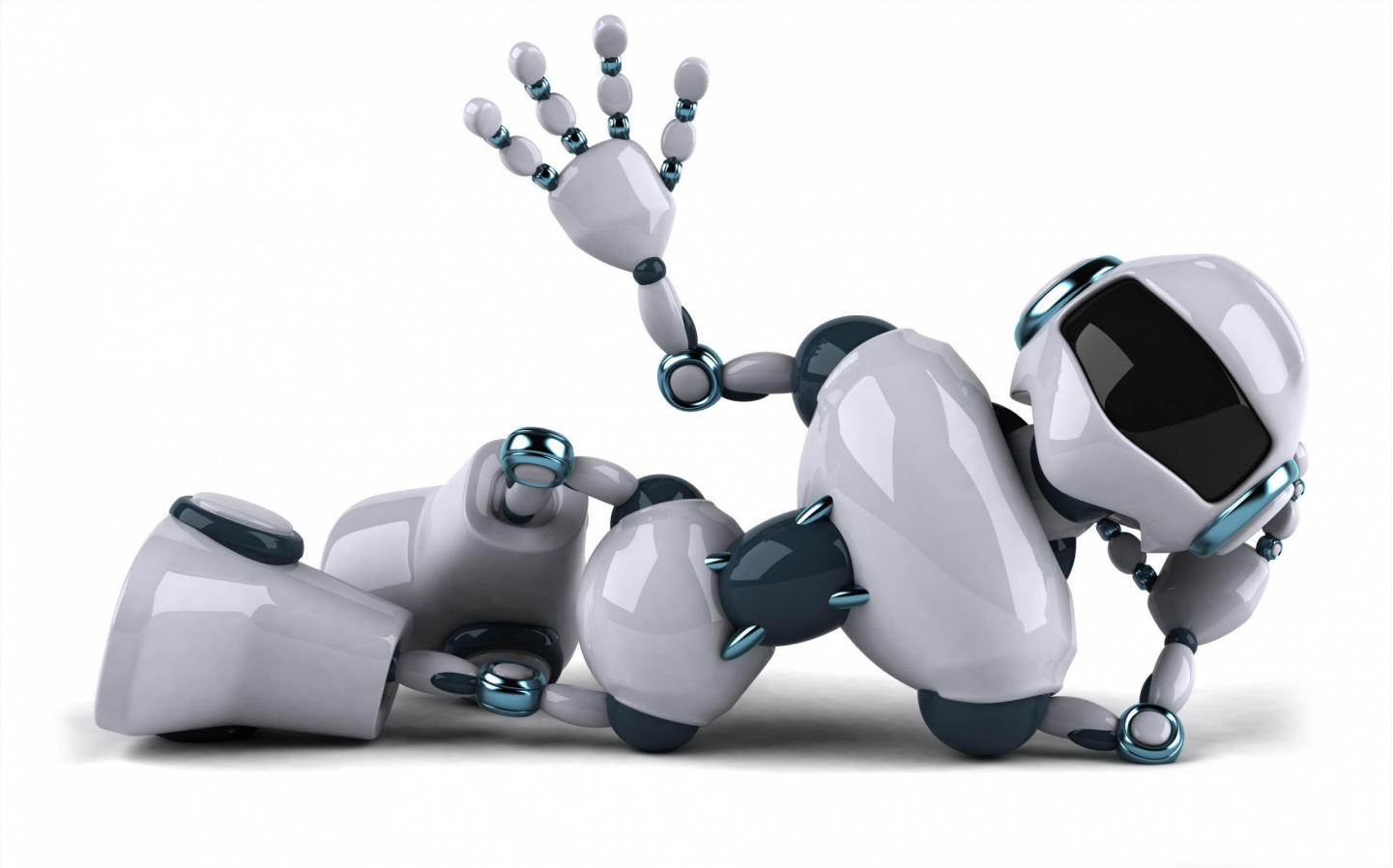
More complex

What is an environment?

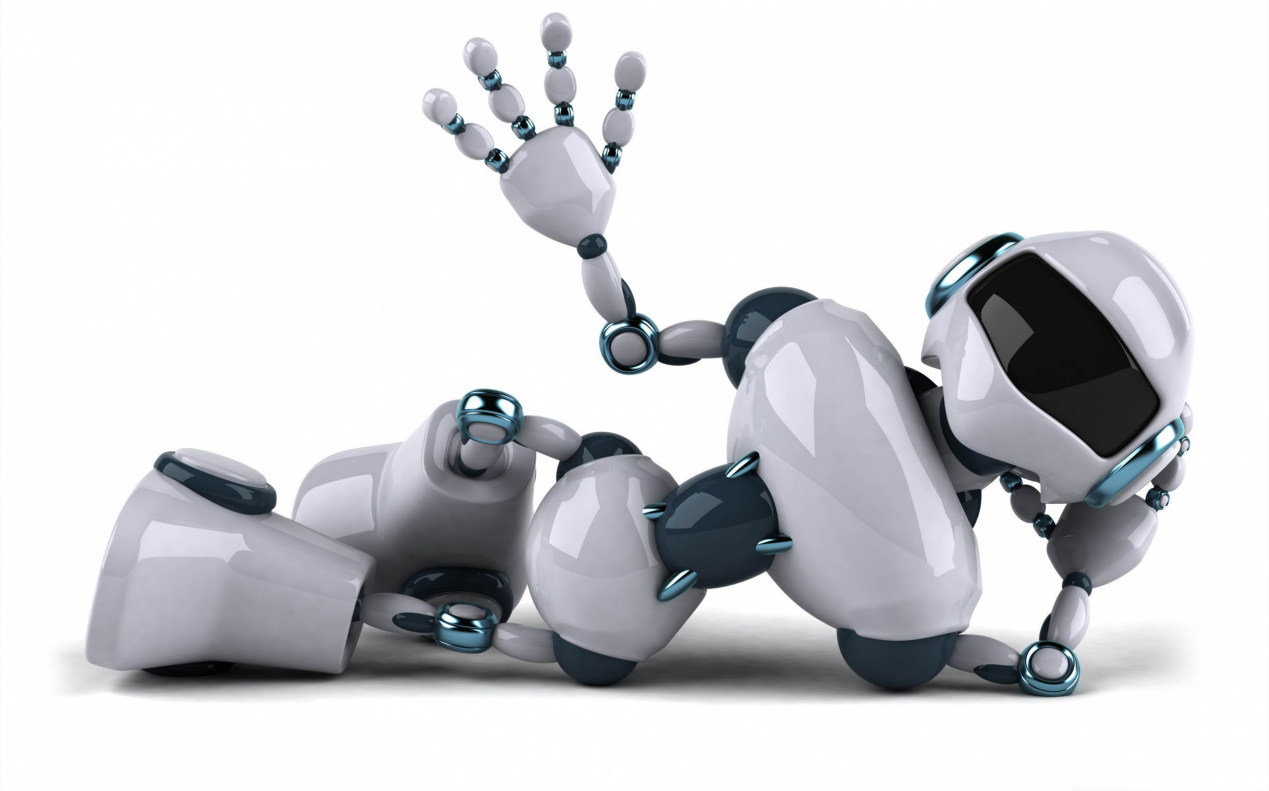
More more complex



What is an agent?



What is an agent?



Indeed, more boring... We will see that it can be from a simple table (Q-Table), a neural network (Q-learning).

Both will be used to approximate a probabilistic function (policy) that tells how to act given the state of the environment .

What problems can we solve?

Classical

- Learn how to play games from the easier (cartpole, Frozen lake) to more complex (Doom, Go, Dota,...).
- Robotics: learn to move, complete tasks, be autonomous,...
- Optimize memory control, personalized web services,...

Quantum

- Huge effort on the generalization of RL in the 'quantum realm' (see V. Dunjko, arxiv:1610.08251)
- Quantum control: prepare physical systems in desired quantum state (1705.00565v2, 1805.00654).
- Preparation of quantum experiments (projectivesimulation.org)
- Certification of many-body properties assisted by machine learning (soon!)

(Tentative) Plan of the course

1. (Day 1) Introduction to Reinforcement Learning
    - Basisc elements of RL
    - Multi-armed bandits
2. (Day 2) Markov decision processes
3. (Day 2) Q-learning: tables and networks
    - Frozen Lake example: tables and networks

Basic elements of Reinforcement Learning

Beyond the agent and the environment, we identify four main subelements:

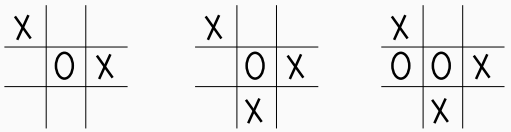
- Policy: ensemble of rules that set the behaviour of the agent.
- Reward signal: defines the goal in a RL problem.
- Value function: specifies what is good on the long run.
- Model: Mimics the responses of the environment.

We are not considering here action and states, which will be explored in the MDPs.

Basic elements of Reinforcement Learning

Example: Tic-tac-toe problem

We construct a vector of values, which is our *value function* V:



$$V \rightarrow \boxed{v(s_k) \mid v(s_j) \mid v(s_i) \mid \cdots}$$

Reward: $v_i \rightarrow 1$ if the current state wins, $v_i \rightarrow 0$ if the current state looses.

Policy: We choose the action that takes us to the state with higher v_i.

How to update V?

$$V(s) = V(s) + \alpha \left[V(s') - V(s) \right]$$

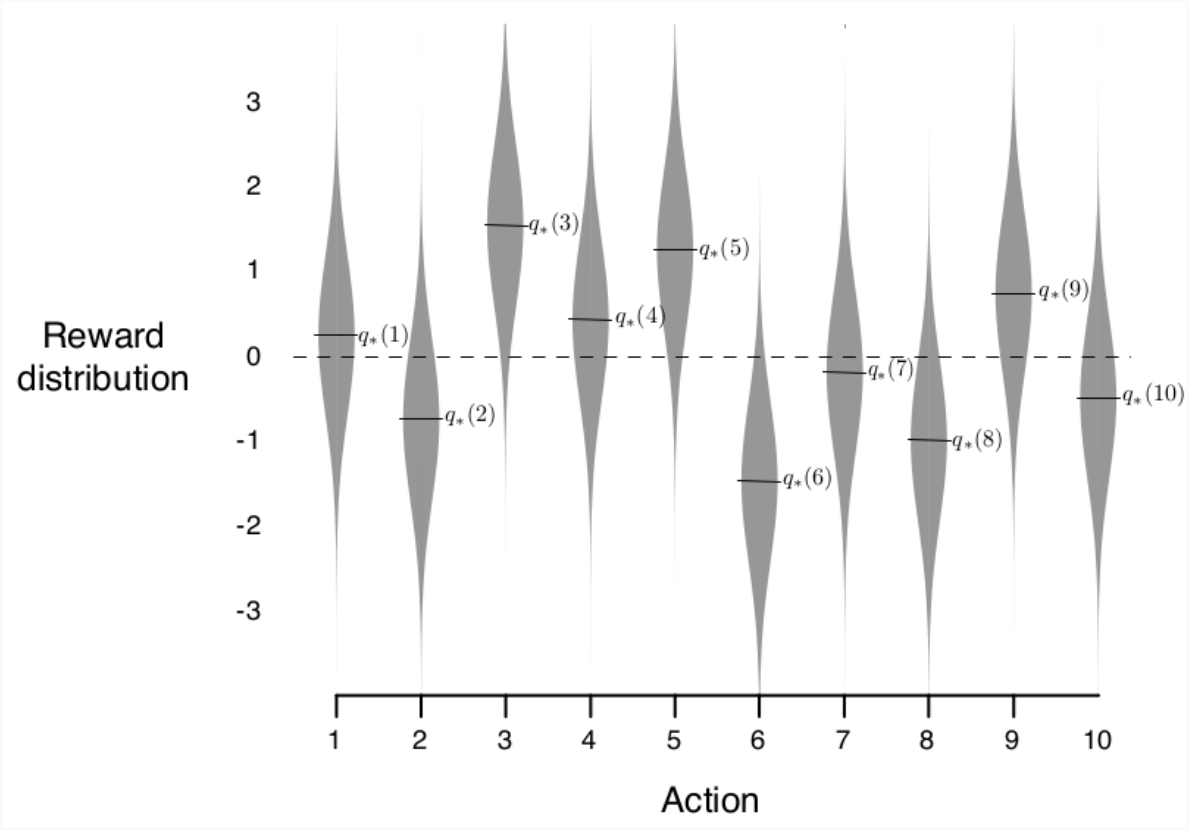
Multi-armed bandits

Multi-armed bandits

SR: 50% SR: 30% SR: 60% SR: 80% SR: 50%

Multi-armed bandits

SR: 50% SR: 30% SR: 60% SR: 80% SR: 50%



Multi-armed bandits

In a k-armed bandit, each of the k actions has an expected reward $\rightarrow$ we will call this the *value* of the action. More formally:

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

We don't know a priori $q_*(a)$. We denote the estimate as $Q(a)$

$\rightarrow$ **Goal:** $Q_t(a)$ to be close to $q_*(a)$!

A simple way of building $Q_t(a)$ would be as follows:

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

We choose actions using $\boxed{A_t = \text{argmax}_a Q_t(a)}$. To allow exploration, we use ε-greedy method.

Incremental implementation

To avoid memory blow-out, we don't want to calculate $\sum_{i=1}^{t-1} R_i \delta(A_i - a)$ at each step. Instead, we use:

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

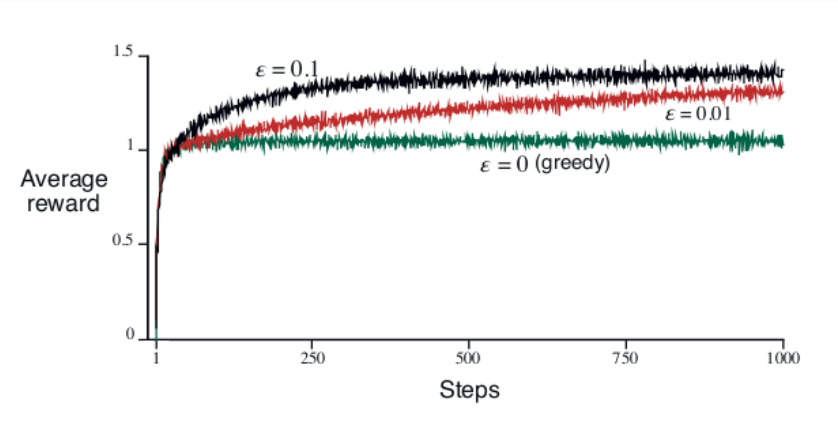
Non-stationary problems

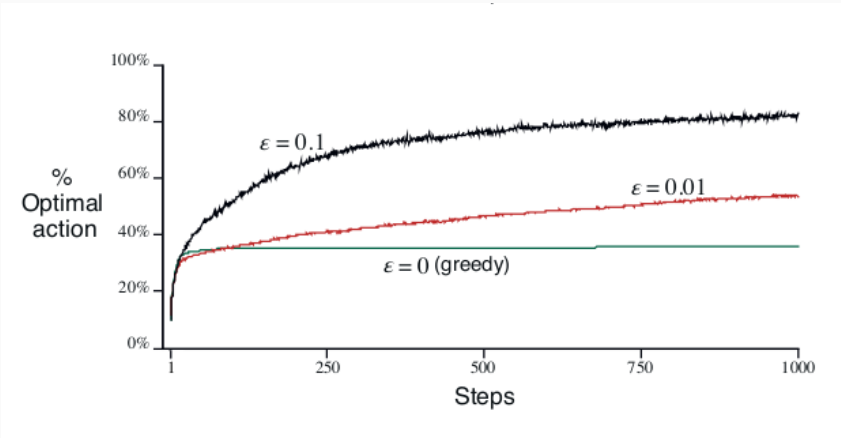
We often encounter problems that evolve in time. Then, it makes more sense to give higher weights to recent rewards than to long-past rewards.

$$Q_{n+1} = Q_n + \alpha[R_n - Q_n] = ... = (1-\alpha)^n Q_i + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i$$

Multi-armed bandits: hands-on problem

Exercise: Solve a 'Gaussian' 10-bandit problem. Compare the results with $\varepsilon = [0, 0.01, 0.1]$. Plot the average reward at each episode over $\sim$2000 runs to find similar results as





(details in github)