Univ Gustave Eiffel - Cosys/Grettia

Reinforcement learning and optimal control

Master 2 : SIA

Nadir Farhi
chargé de recherche, UGE - Cosys/Grettia
nadir.farhi@univ-eiffel.fr

UGE - 16 spetember 2024

Presentation

- Nadir Farhi, researcher, Université Gustave Eiffel
- Laboratory : Grettia

- Habilitation in Mathematics, 2018, University Paris Est
- PhD in Mathematics, 2008, University Paris 1 Sorbonne

- Topics of research : Mathematical modeling and control
- Applications : Transportation : Traffic modeling and control

What about you ?

The program

1. Introduction
2. Markov decision processes
3. Dynamic programming
4. Monte Carlo methods
5. Q-learning and Deep Q-learnig + Complements

Evaluation

The evaluation will be based on

- Participation during the classes.
- Contribution to the project.
- Final report of the project.
- The defense.

What I will know from this course ?

- What is RL and why should I consider it ?
- Relationship with optimal control.
- How do I set it up and solve it ?
- What are some benefits and drawbacks ?

Disciplines related to reinforcement learning

- Dynamical systems
- Optimal control theory
- Markov decision processes
- Monte Carlo simulation
- TD (temporal differences) methods
- Optimization in general (policy-based algorithms
- Artificial Intelligence (AI)
- Machine learning
- Statistics
- Psychology
- neuroscience, etc.

Machine learning

Three classes of ML :

- Supervised learning ?
- Unspervised learning ?
- **Reinforcement learning** ?

The process of learning

Examples of learning (humans) :

- Look (observe)
- Move (arms, feet, head, etc.)
- Walk
- Speak
- Run
- Play
- Write, read, calculate
- Memorize
- Hold conversation
- Present (give a presentation)
- Lie
- Drive, etc.

RL - Learn from interaction

modules.ilas.uw.ed

- Learning is to be aware of how our environment responds to what we do.
- Learning from interaction is one of the foundational ideas of RL.
- In RL, we explore a computational approach to learning from interaction.
- Evaluation through mathematical analysis or computational experiments.
- RL is focused on goal-directed learning from interactions.

Trial-and-error search & delayed reward

- In Reinforcement learning, one learns :
  - what to do ?
  - how to map situations to actions ?
  - $\rightarrow$ to maximize a numerical reward signal.
- The learner is not told which actions to take.
  Instead, he must discover which actions yield the most reward by trying them.
- Actions may affect not only the immediate reward,
  But also the next situation and, through that, all subsequent rewards.
- Two characteristics of RL :
  - Trial-and-error search
  - Delayed reward

RL & Markov Decision Processes

Markov decision processes are intended to include three aspects :

- sensation (observation)
- action,
- goal

Supervised & unsupervised learning

Supervised learning :

- Learn from a training set of labeled examples
- Provided by a knowledgable external supervisor.
- Each example is a description of a situation
- Label : a specification of the correct action to take in that situation.
- Learn : extrapolate, or generalize, its responses to new situations.

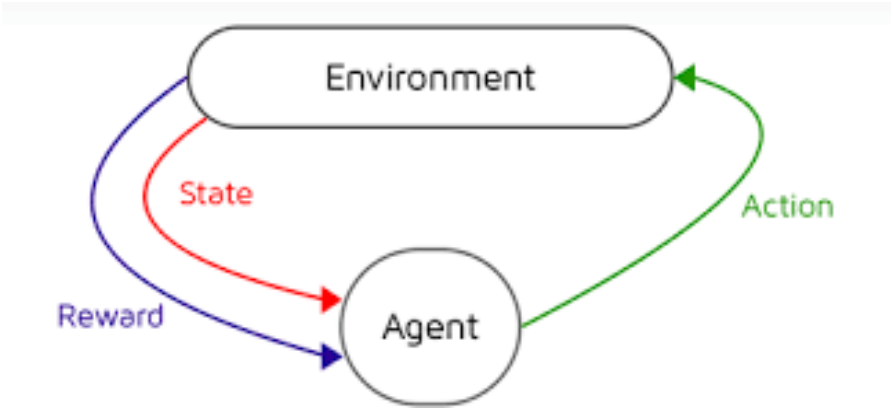
Unsupervised learning :

- Find structure hidden in collections of unlabeled data.

Agent

- By a complete, interactive, goal-seeking agent we do not always mean something like a complete organism or robot.
- These are clearly examples, but a complete, interactive, goal-seeking agent can also be a component of a larger behaving system.
- In this case, the agent directly interacts with the rest of the larger system and indirectly interacts with the larger system ?s environment.
- Examples :
  - Robot,
  - autonomous vehicle,
  - an agent that monitors the charge level of robot's battery and sends commands to the robot.

Elements of RL (1/2)



DeepAI

- **Agent :** the controller
- **Environment :** local and global neighbor of the agent
- **Policy :** the way the agent behaves at a given time
- **Reward :** the goal function of a RL problem
- **Model :** it mimics the behavior of the environment and gives the transitions.

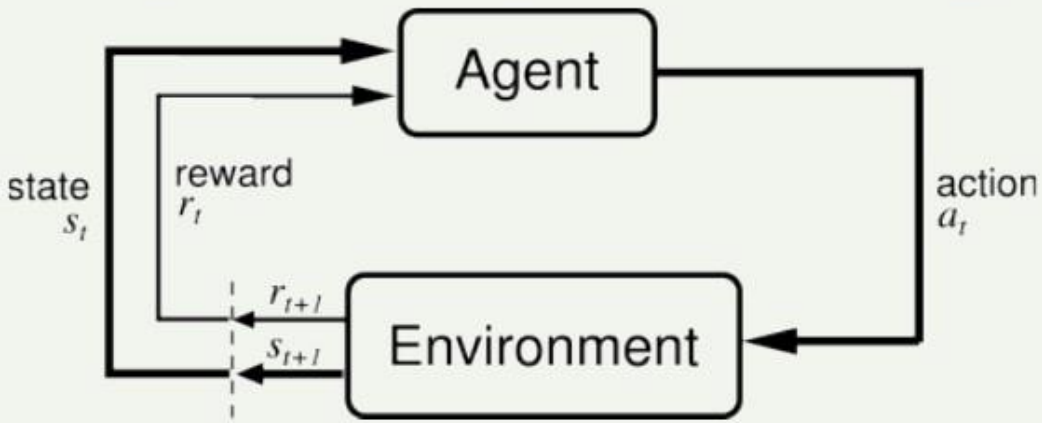
Elements of RL (2/2)

- RL framework:

Environment | Agent | Step | State | Action | Reward | Episode

- Markov Decision Process:

1. Observation > 2. Action > 3. Transition > 4. Reward



- Goal: Find a policy that maximizes discounted cumulative reward over time.

Exploitation & Exploration

- Exploitation & Exploration dilemma is specific to RL.
- A reinforcement learning agent can prefer :
  1. Already tried actions effective in producing reward (exploit)
  2. New actions not selected before (explore)
- Dilemma : Neither exploration nor exploitation can be pursued exclusively.
- The agent must try a variety of actions and progressively favor the best ones.
- Stochastic tasks : Each action must be tried many times to estimate its reward.
- This dilemma has been intensively studied, yet remains unresolved.

TD-Gammon, AlphaGo & Alpha Zero

- TD-Gammon (1990) : plays backgammon
- Alpha Go (2016, Deep Mind Inc.) : plays Go
- Alpha Zero (2017, Deep Mind Inc.) : plays chess, Go, etc.

Alpha Go & Alpha Zero :

- Offline training is used to teach how to evaluate states (positions) and to generate good actions (moves) at any given state (position).
- On-line play is used to play in real time (against humans or computers).
- Both programs play better than all competitor computer programs and much better than all humans.

Bandit Problem

A k-armed Bandit Problem

- We have to take a choice among k different options, or actions.
- After each choice we receive a numerical reward.
- The reward is chosen from a stationary probability distribution that depends on the action we selected.
- Our objective is to maximize the expected total reward over some time period.
- For example, over 1000 action selections, or time steps.

The value of an action

$q_*(a) := \mathrm{E}(R_t \mid A_t = a)$.

- If we know this value, it would be trivial.
- We assume that we do not know it.
- We would like that the estimate $Q_t(a)$ approaches $q_*(a)$.
- Greedy action : the action with the maximum estimate.
- **Exploitation :** Selection of greedy actions.
- **Exploration :** Selection of non-greedy actions.

Action value methods

- One natural way to estimate the value of an action is by averaging the rewards actually received :

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

- If the denominator is zero, we define : $Q_t(a) = 0$.
- The simplest action selection is to take the (or one of the) greedy action(s) :

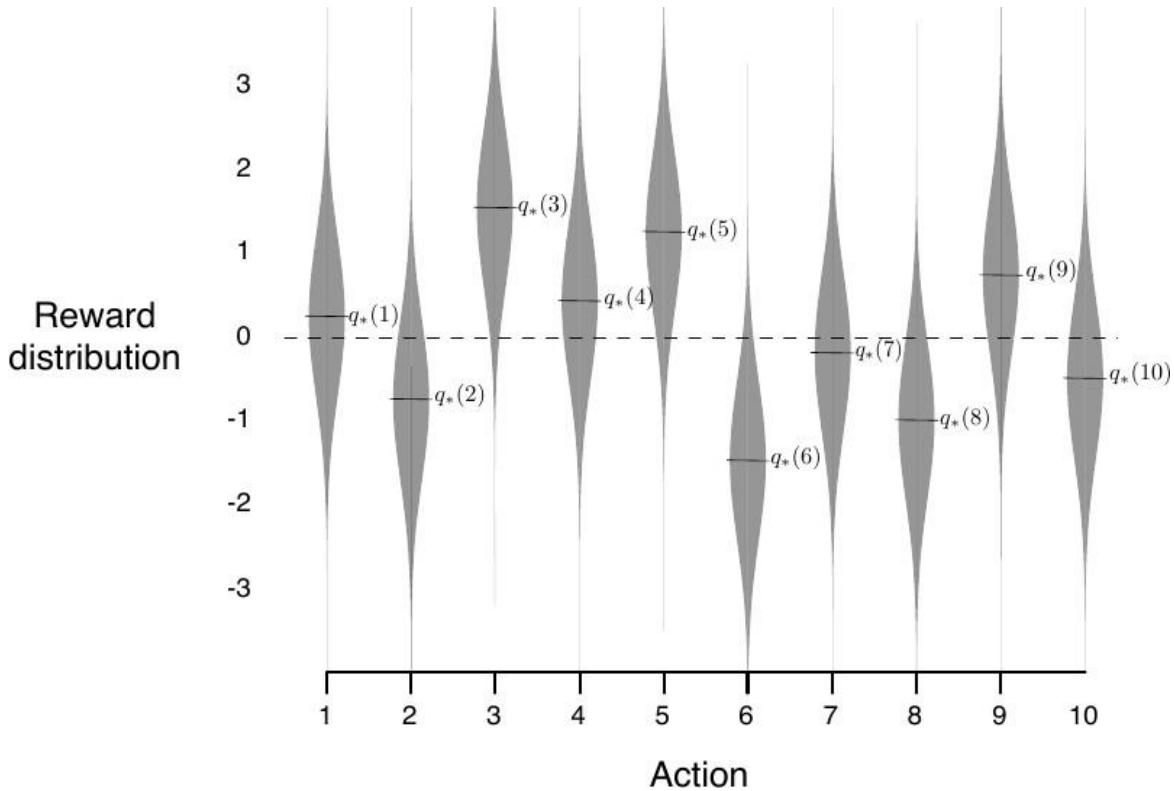
$$A(t) = \arg\max_a Q_t(a).$$

- ε-greedy action :

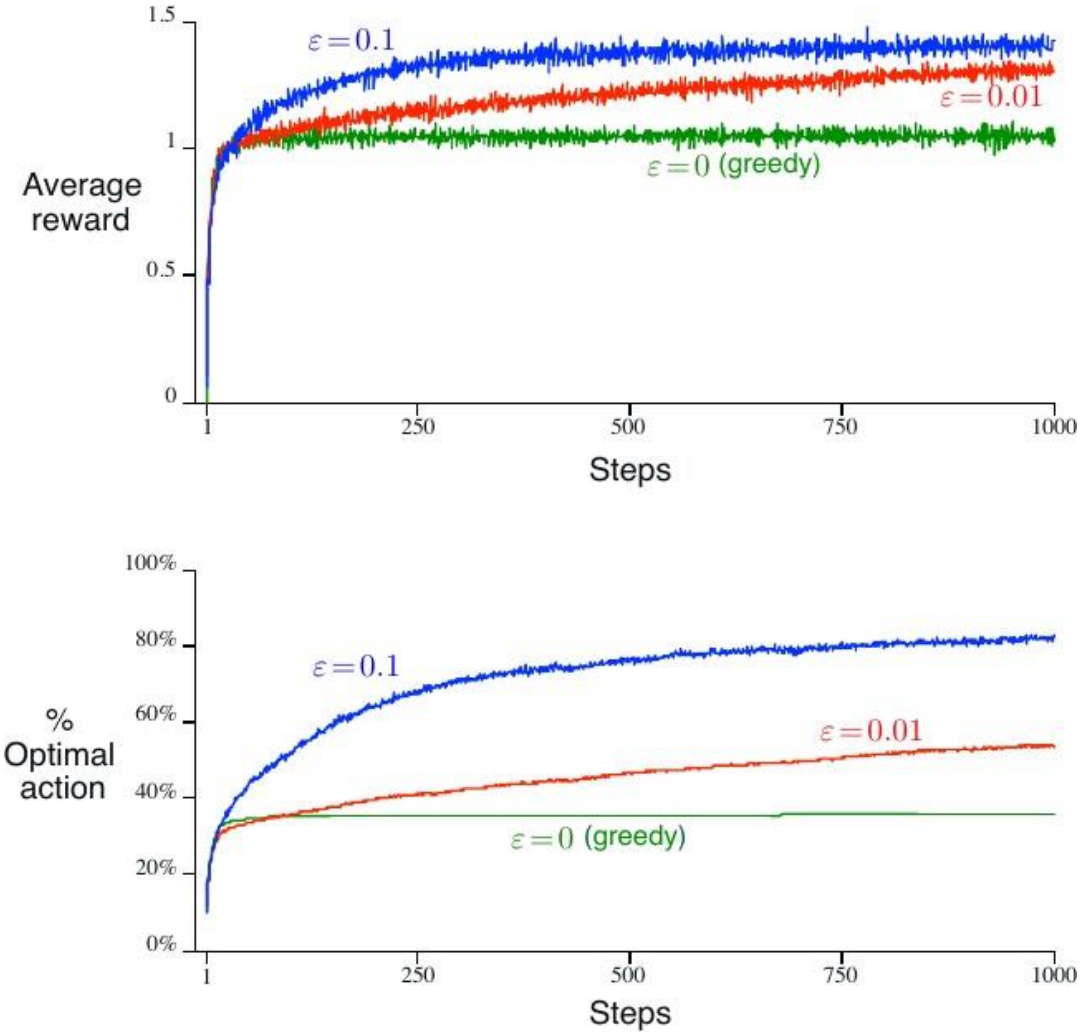
$$A(t) = \begin{cases} \arg\max_a Q_t(a) & \text{with probability } 1 - \varepsilon \\ \text{randomly selected action} & \text{with probability } \varepsilon \end{cases}$$

The 10-armed Testbed

- We take a set of 2000 randomly generated k -armed bandit problems with k = 10.
- The action values, $q_*(a), a = 1, 2, \ldots, 10$, were selected according to a normal (Gaussian) distribution with mean 0 and variance 1.
- The actual reward, R_t , was selected from a normal distribution with mean $q_*(A_t)$ and variance 1.



The 10-armed Testbed



Averages over 2000 runs with different bandit problems. Initial estimates : zero.

Incremental Implementation

- Estimate of the value of an action in *n* steps :

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

- Incremental estimation :

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

- General principle :

$$NewEstimate \leftarrow OldEstimate \; + \; StepSize \; \Big[Target - OldEstimate\Big].$$

A simple Bandit algorithm

A simple bandit algorithm

Initialize, for $a = 1$ to k:

$Q(a) \leftarrow 0$

$N(a) \leftarrow 0$

Loop forever:

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

$R \leftarrow bandit(A)$

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

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

Nonstationary problems

- **Stationary problem :** the reward probabilities do not change over time.
- Most of encountred RL problems are nonstationary.
- In such cases it makes sense to give more weight to recent rewards than to long-past rewards.
- One of the most popular ways of doing this is to use a constant

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

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

Nonstationary problems

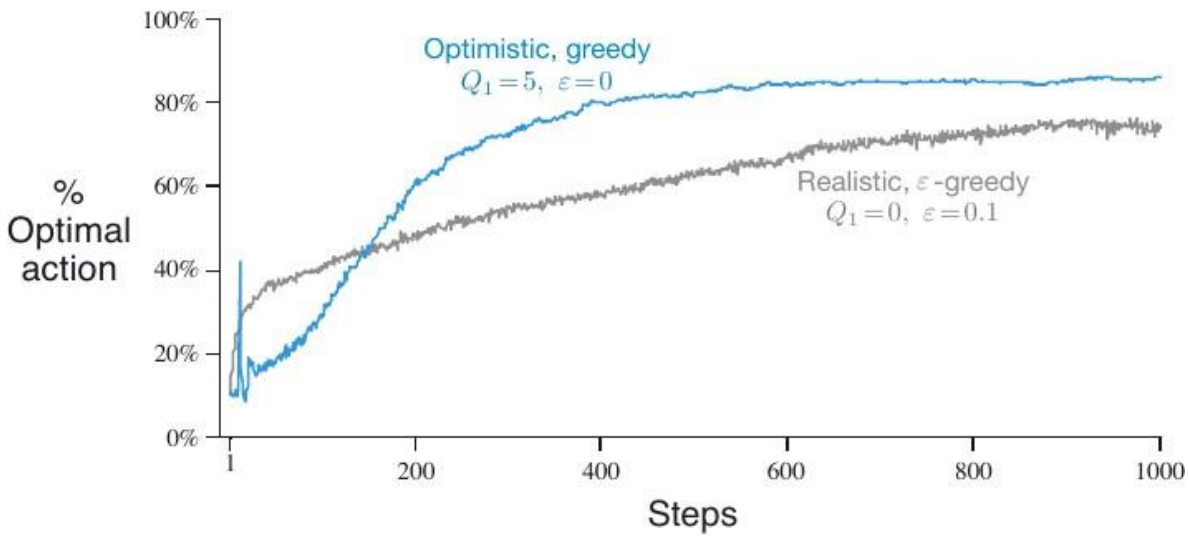
- Sample average method : $\alpha_n(a) = 1/n$, converges to $q_*(a)$.
- Weighted average method : $\alpha_n(a) = \alpha$, no convergence.

The conditions required to assure convergence with probability 1 :

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

Optimistic Initial Values

- The methods above are dependent to some extent on the initial action-value estimates, $Q_1(a)$.
- For the sample-average methods ($\alpha_n(a) = 1/n$), the bias disappears once all actions have been selected at least once.
- For methods with constant α, the bias is permanent.
- The downside of this bias is that the initial estimates become a set of parameters that must be picked by the user, if only to set them all to zero.
- The upside is that they provide an easy way to supply some prior knowledge about what level of rewards can be expected.
- Initial action values can also be used as a simple way to encourage exploration.



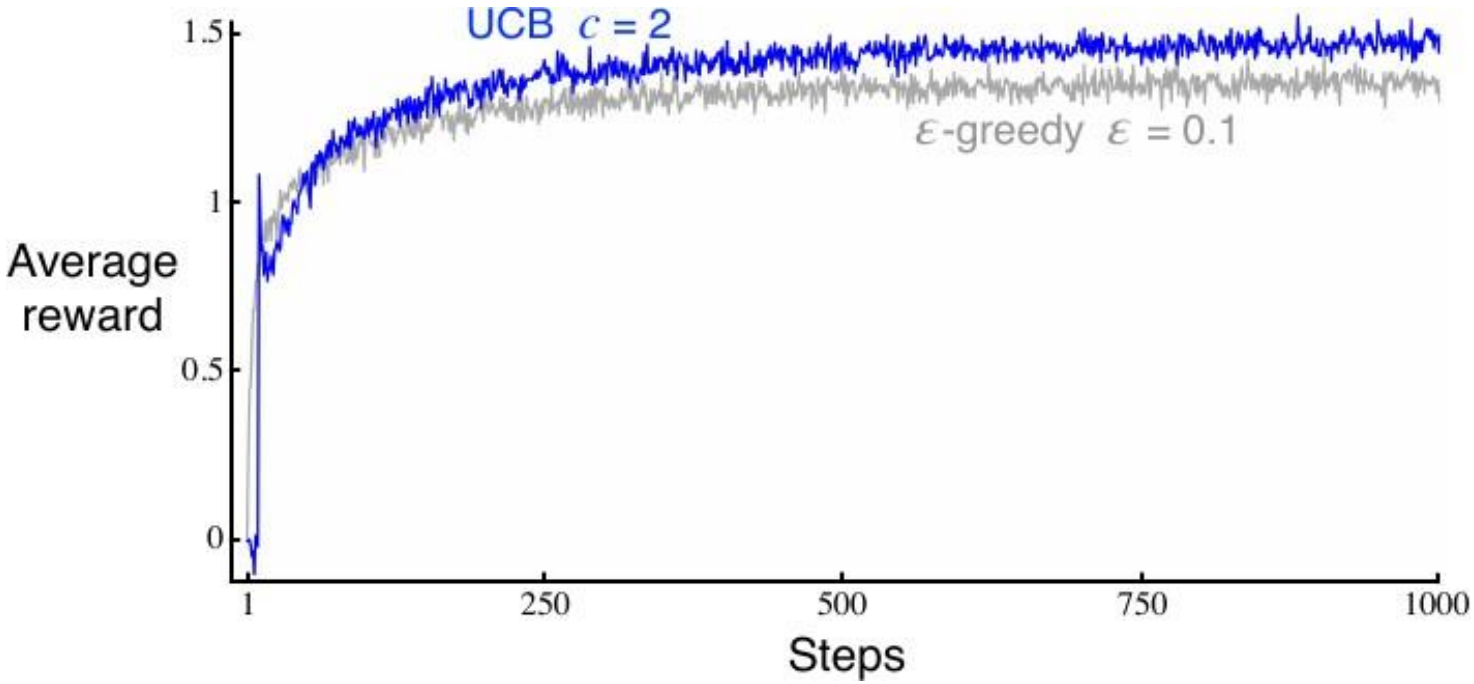
Upper-Confidence-Bound (UCB) Action Selection

- ε-greedy action selection forces the non-greedy actions to be tried, but indiscriminately.
- It would be better to select among the non-greedy actions according to their potential for actually being optimal.
- One effective way of doing this is to select actions according to :

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

- $N_t(a)$ denotes the number of times that action a has been selected prior to time t.
- $c > 0$ controls the degree of exploration.
- Difficulty 1 : non-stationary problems.
- Diffuculty 2 : large state spaces.

UCB action selection



Gradient Bandit Algorithms

- We define a preference $H_t(a)$ for each action a.
- The larger the preference, the more often that action is taken (with no interpretation in terms of reward).
- We determine the action probabilities, according to the Soft-max distribution :

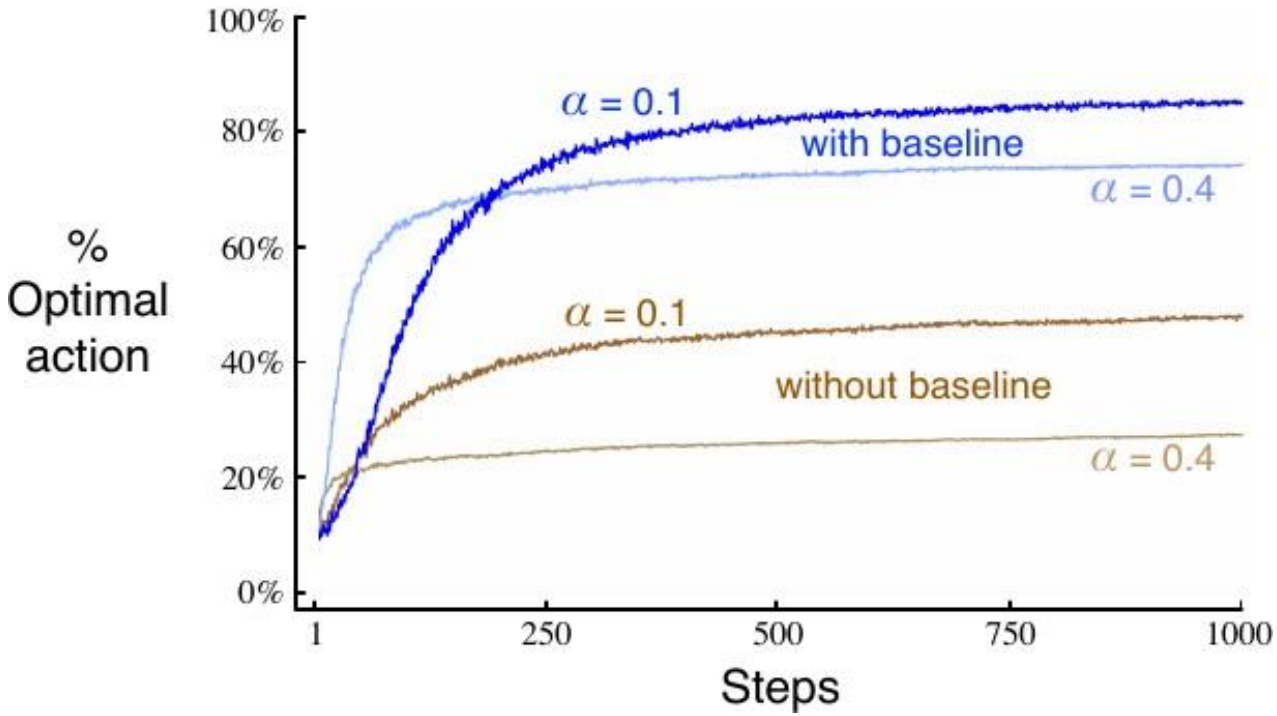
$$\Pr\{A_t = a\} \doteq \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} \doteq \pi_t(a),$$

- A natural learning algorithm for soft-max action preferences based on the idea of stochastic gradient ascent :
- On each step, after selecting action A_t and receiving the reward R_t, the action preferences are updated :

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

- α : step size parameter,
- $\bar{R}_t$: the average of the rewards up to but not including t (baseline).

Gradient Bandit Algorithms

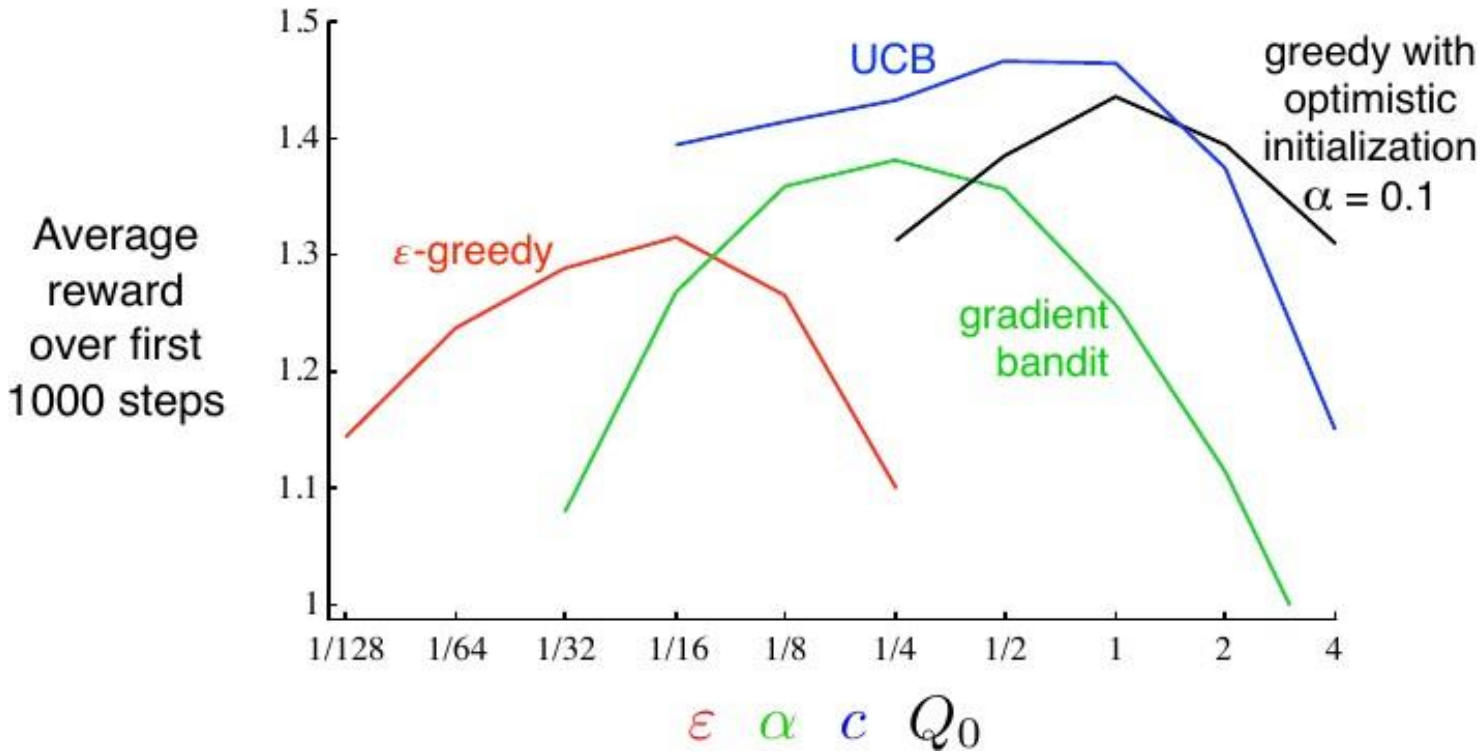


Average performance of the gradient bandit algorithm with and without a reward baseline ($\bar{R}_t$ /= or $= 0$) on the 10-armed testbed when the $q_*(a)$ are chosen to be near +4 rather than near zero.

Associative Search (Contextual Bandits)

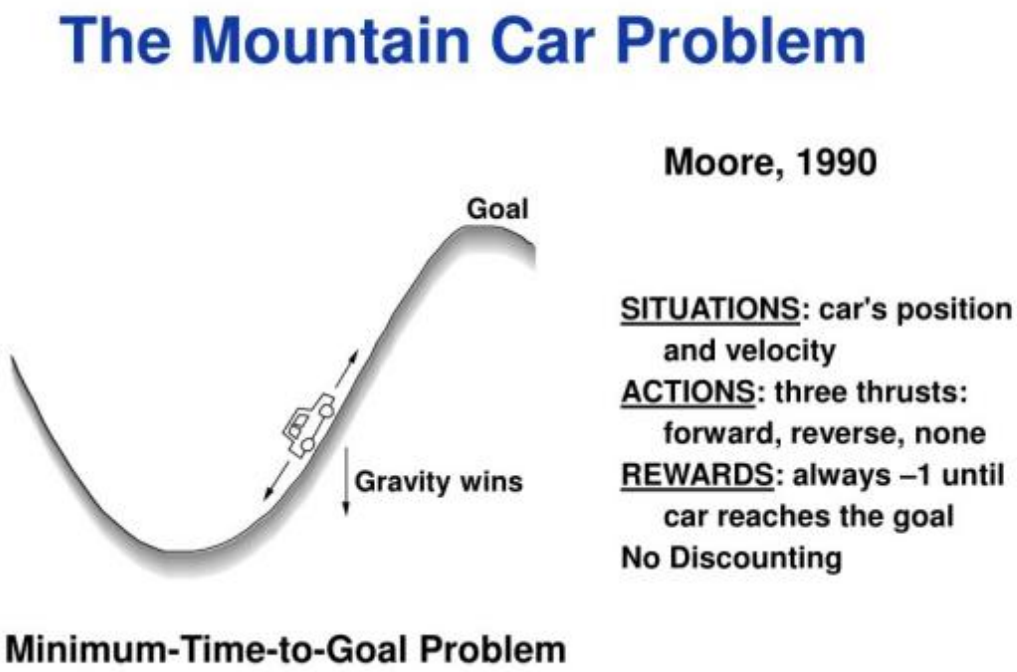
- Actions are associated with the situations in which they are best.
- Associative research affect only the intermediate reward, like in the k-armed bandit.
- They involve learning a policy, as in the full reinforcement lerning problem. So they are intermediate between the k-armed bandit problem and the full reinforcement learning problem.
- A policy maps obervation of the environment to actions (deterministic).
- A policy maps obervation of the environment to probability distribution on the set of actions (stochastic).

Summary



Each point is the average reward obtained over 1000 steps with a particular algorithm at a particular setting of its parameter.

Project 2020 - 2021 - Mountain car



- A standard testing domain in Reinforcement learning.
- Under-powered car must drive up a steep hill.
- Since gravity is stronger than the car's engine, even at full throttle, the car cannot simply accelerate up the steep slope.
- The car is situated in a valley and must learn to leverage potential energy by driving up the opposite hill before the car is able to make it to the goal at the top of the rightmost hill.

Project 2021 - 2022 - Lane change

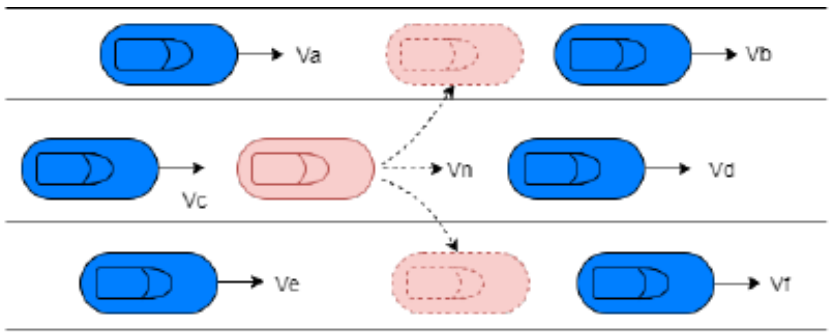
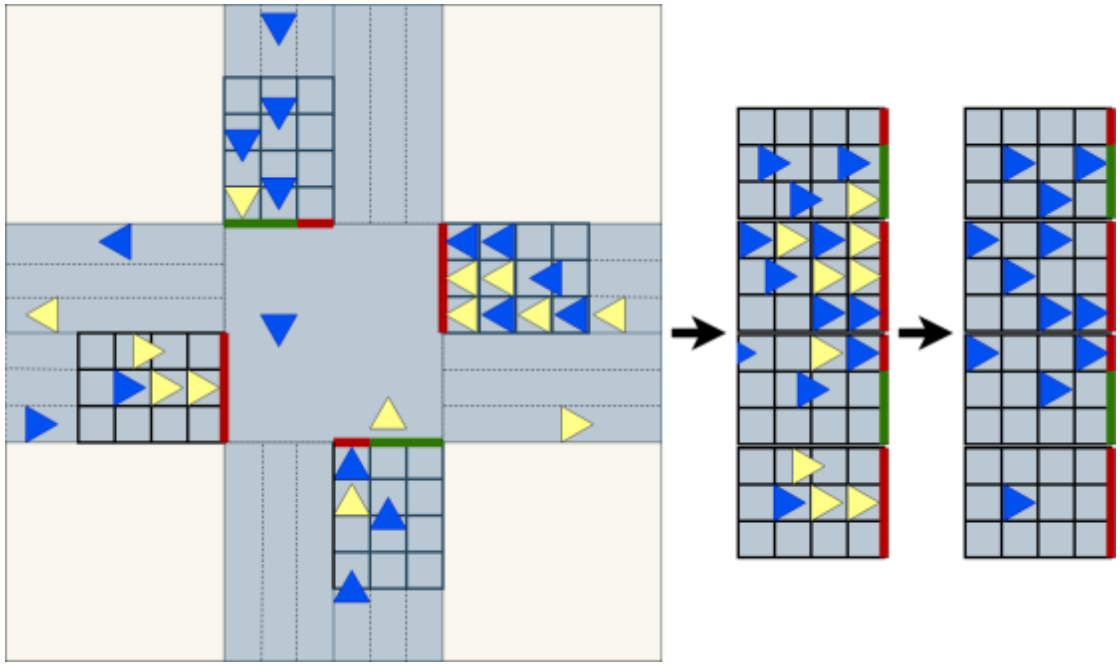


Fig. 2. The presentation of traffic environment for the control of the lane changing behavior of the target vehicle (in pink).

- Apply the algorithm of Q-learning to learn by numerical simulation the lane change decisions on a stretch of highway.
- Use trafic simulator SUMO to simulate the car-following of all cars, and the lane change of all cars, except one, which we control under the Q-learning algorithm.

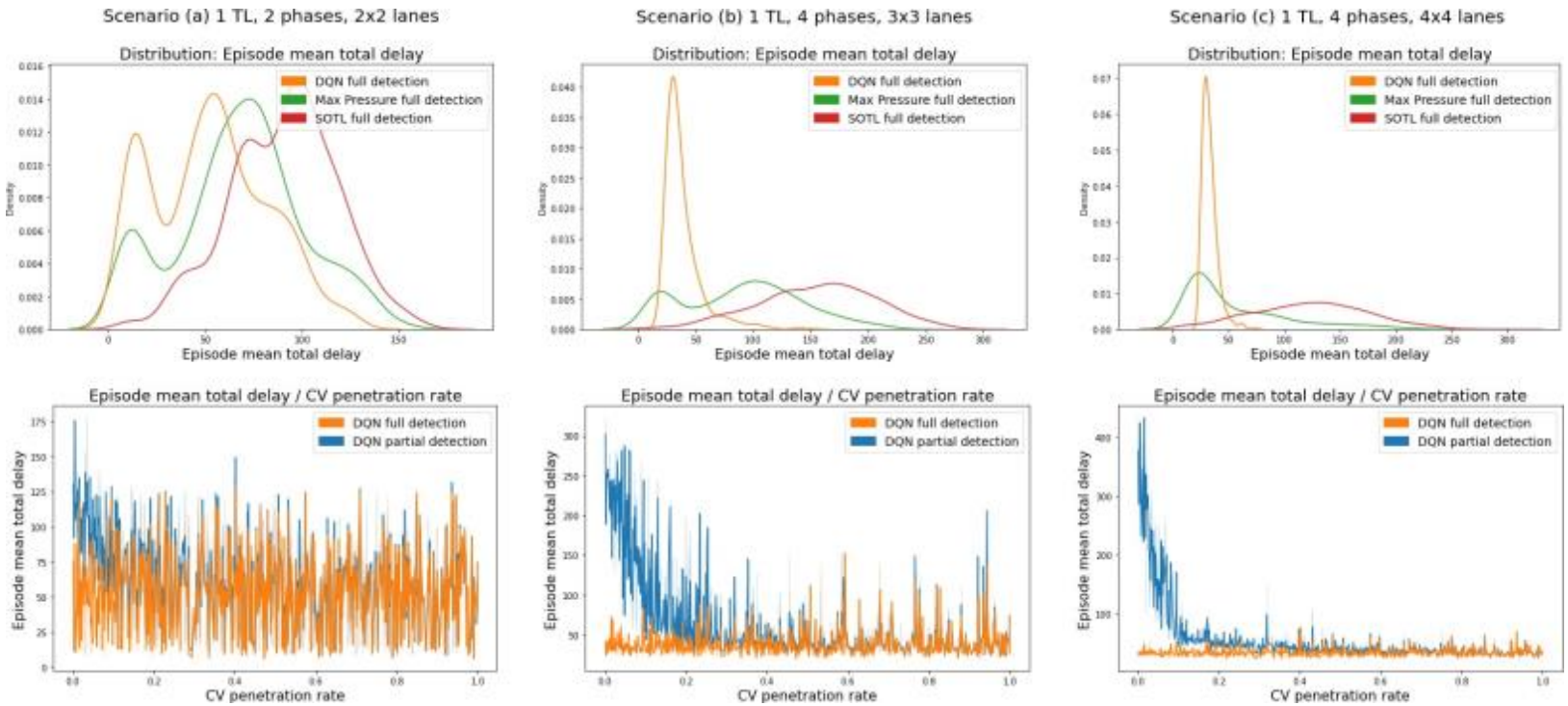
Research internship in the Laboratory

Romain Ducrocq (2021)
Deep reinforcement Q-learning for intelligent traffic signal control with partial detection



Research internship in the Laboratory

Romain Ducrocq (2021)
Deep reinforcement Q-learning for intelligent traffic signal control with partial detection



Research internship in the Laboratory

Shurok Khozam (2022)

Deep Reinforcement Q-Learning for Intelligent Traffic Control in Mass Transit