Polytechnique School of ULB
Alexandre Balon-Perin
First Master in Computational Intelligence

Learning Dynamics

Assignment 1

N-Armed Bandit problem

Academic Year 2010-2011

Table of Contents

Introduction **3**

Specifications **3**

Quality **3**

Actions **3**

ε-Greedy **3**

Softmax **3**

Selection Methods **4**

Random **4**

Greedy **4**

ε-Greedy **4**

Softmax **4**

Simulations **5**

Question 1: Plotting Results **5**

Plots 5

Qa over time 5

Average Reward 6

Histogram of Action Counter 6

New simulation to check differences 7

Analysis of the results 7

Question 2: Softmax Action Selection **8**

Plots 8

τ = 3 8

τ = 5* (1000 – t)/1000 9

Analysis of the results 10

Question 3: Over/Underestimating Q-values **11**

Introduction 11

Plots 11

Overestimating 11

Underestimating 13

Analysis of the results 15

Conclusion **16**

Introduction

In the context of the course “Learning Dynamics”, we were asked to implement the “N-Armed Bandit” problem. This is a very good way to introduce the concept of machine learning because we can fairly understand how a machine learns in a step-by-step process. On each step, the agent has the choice between several actions using one of the four different selection methods that we will implement in the system: Random, Greedy, ε-Greedy and Softmax. Each action has a quality (Qa_i) and an expected quality (Qa_i*).

In this assignment, we actually implemented a 6-Armed Bandit and a 3-Armed Bandit. We will start by writing the code of the selection methods then we’ll go through 3 different simulations. Those simulations will show us the development of the learning process by plotting graphs of the important variables such as the quality $Qa_{,i}$, the average reward, the average quality,… over time. We will eventually analyse the diagrams to understand the behavior of the selection methods.

Specifications

Quality

Qa are initialized to 0.

Actions

Action #1 : Qa = 2.7
Action #2: Qa = 2.1
Action #3: Qa = 1.5
Action #4: Qa = 0.9
Action #5: Qa = -1.5
Action #6: Qa = -2.7

ε-Greedy

ε = 0.2

Softmax

τ = 3

Selection Methods

Random

The agent chooses randomly between the 6 actions. A random number between 1 and 6 is assigned to a variable that will be used as the index of a vector of actions. This way, a unique action is selected randomly at each step. Afterwards, we calculate the reward with a normal probability distribution with mean the quality expected and variance 1. Finally, we calculation the new quality for the step:

$$Qa_i = \frac{\sum Qa_{i-1} + \text{reward}}{iteration}$$

Greedy

In the Greedy selection method, the agent always chooses the action with the higher quality. This means that the agent exploits but never explores new possibilities. Once a “good” action has been selected, it doesn’t step back to try another path. Then, we do the same calculation as the random method.

ε-Greedy

The Epsilon-Greedy selection method, we mix the two first methods. The agent will choose a random method with probability ε and a Greedy method with probability 1-ε. Eventually, we follow the same steps than the Random method. We use a coefficient ε = 0.2 which means that 20% of the time the agent explores new paths.

Softmax

The Softmax selection method is more complex because it uses a probability to choose an action a bit more elaborated:

$$p(x) = \frac{e^{Q(a)/\tau}}{\sum_{b=1}^{n} (e^{Q(b)/\tau})}$$

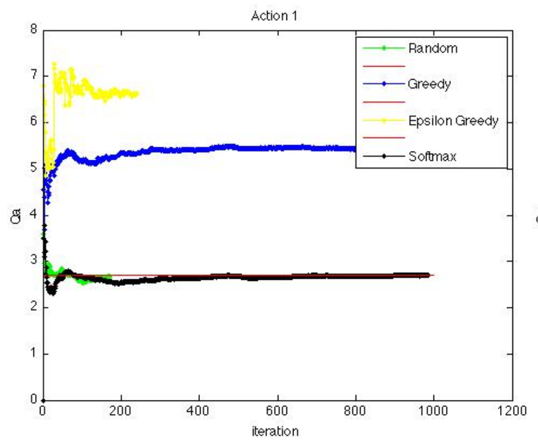
The calculus is then the same as before.

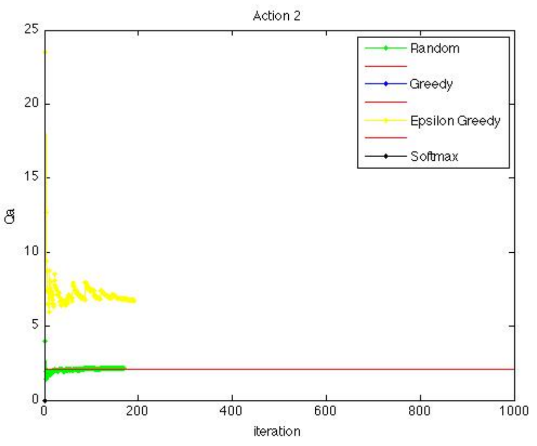
Simulations

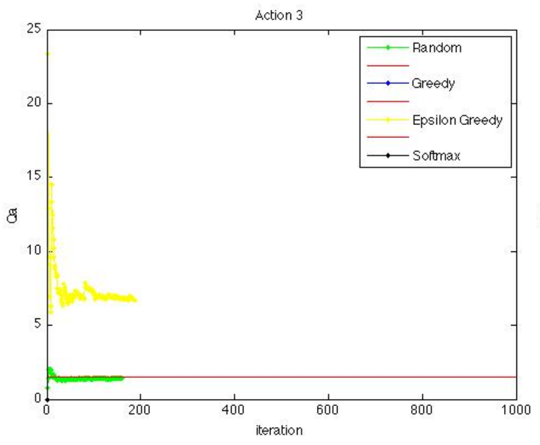
Question 1: Plotting Results

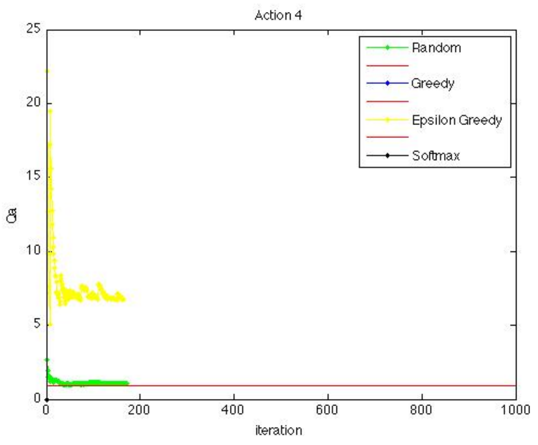
Plots

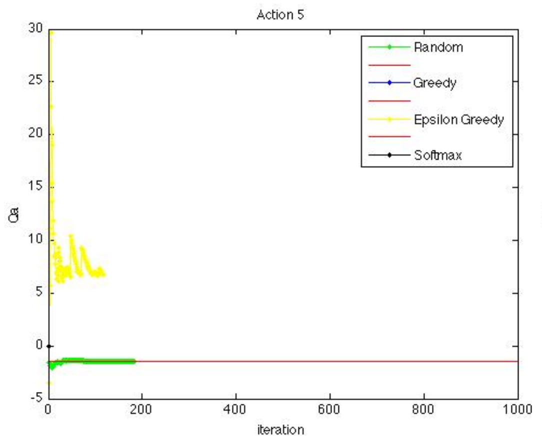
Qa over time

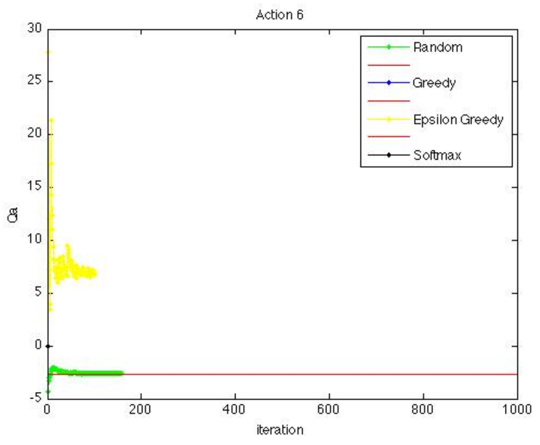




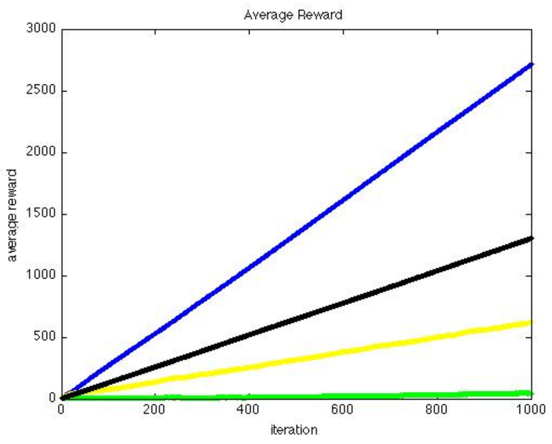




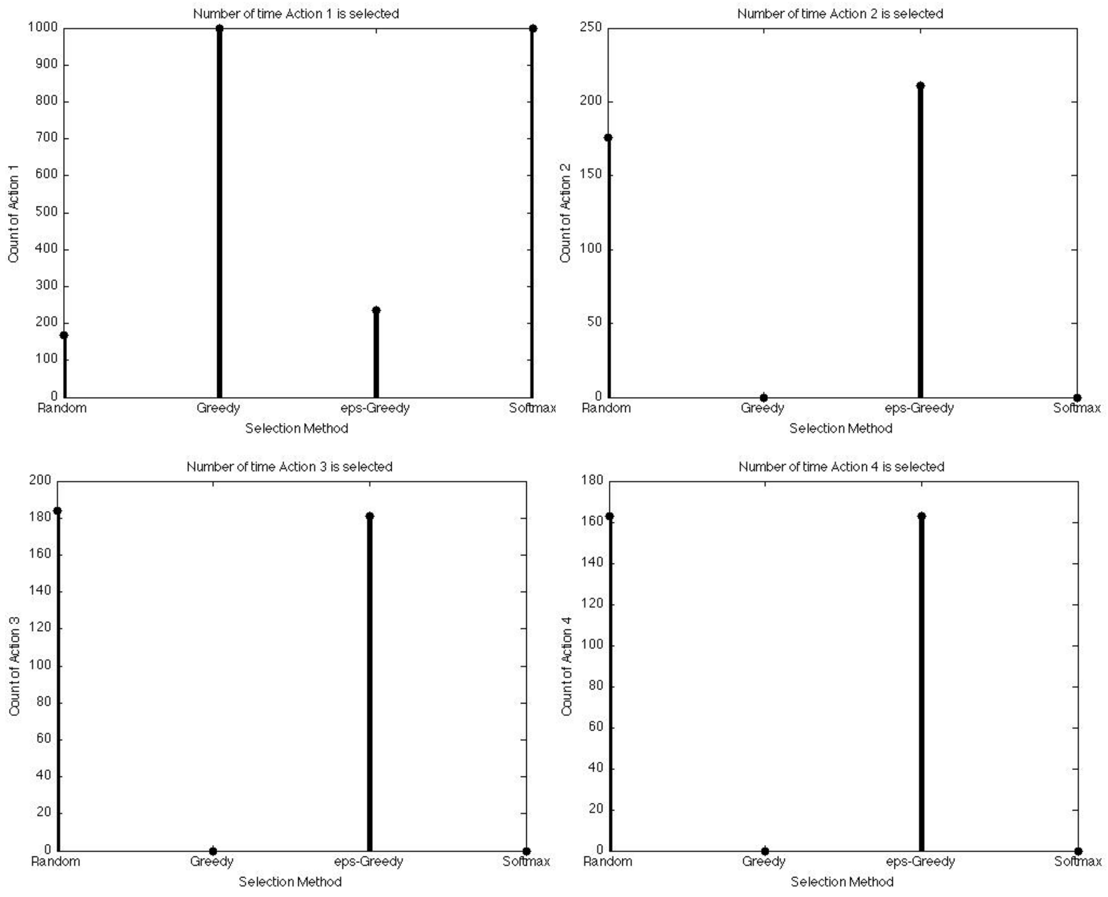


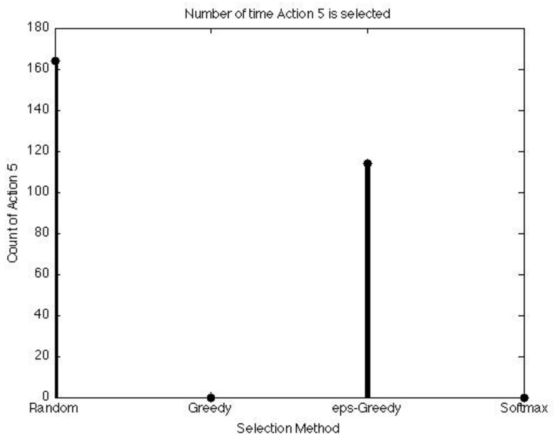


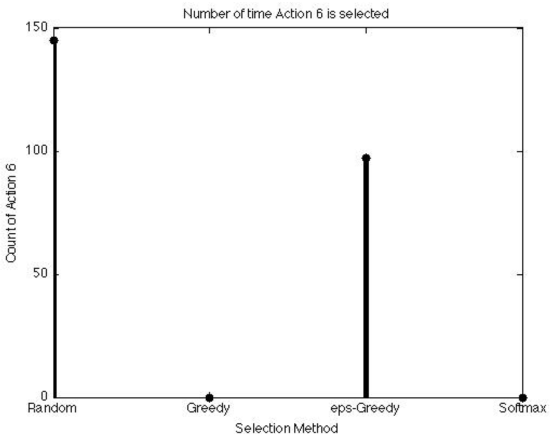
Average Reward



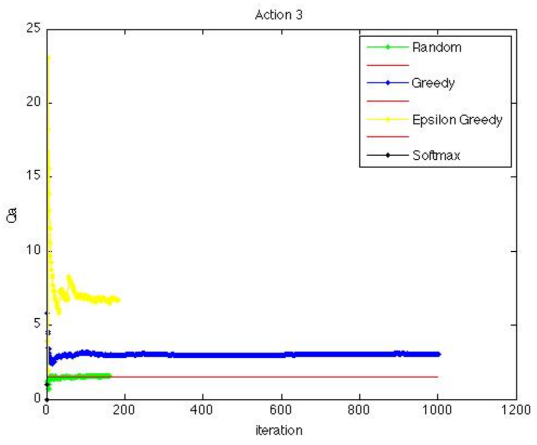
Histogram of Action Counter







New simulation to check differences



Analysis of the results

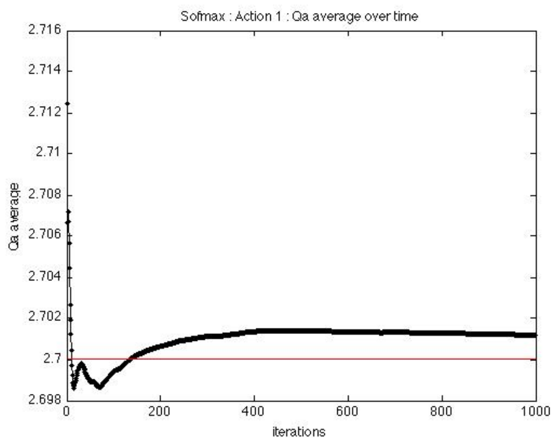
We can see on these diagrams that apart from the Random method all the others have found the best solution which is action 1. However there are differences between the methods. The Greedy finds a “good” solution and exploits it without ever exploring again whereas the ε-Greedy explores 20% of the time new path. As we can see on the last plot, the Greedy algorithm exploited action 3 without stepping back. As a result, the algorithm finds a “good” solution but definitely not the best. The Softmax method also finds the solution but does not explore as much as the ε-Greedy moreover the algorithm finds the expected quality much more accurately. There is no way to identify easily the best method between ε-Greedy and Softmax. It would require an analysis of the probability distribution used in the latest.

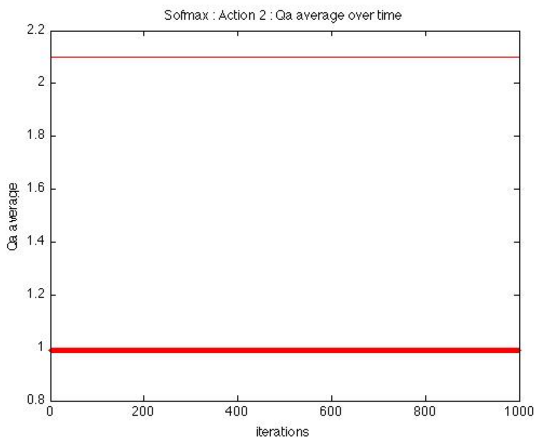
Eventually the average reward over time grows much faster with the Greedy algorithm as expected. Obviously the average reward for a random selection oscillates around the mean of the expected quality’s values.

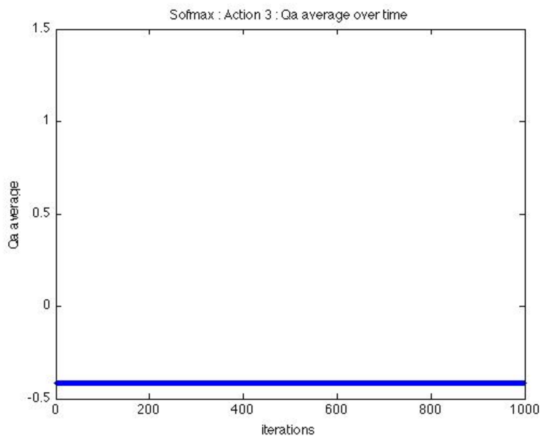
Question 2: Softmax Action Selection

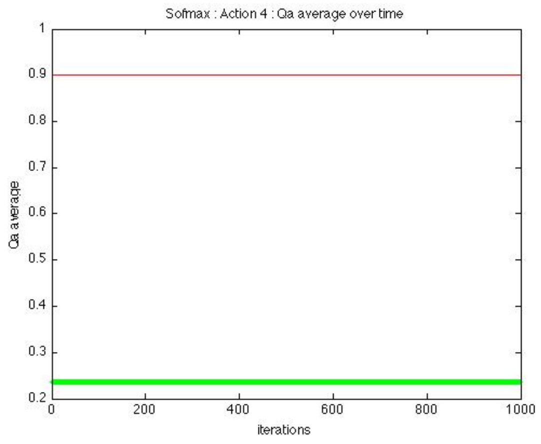
Plots

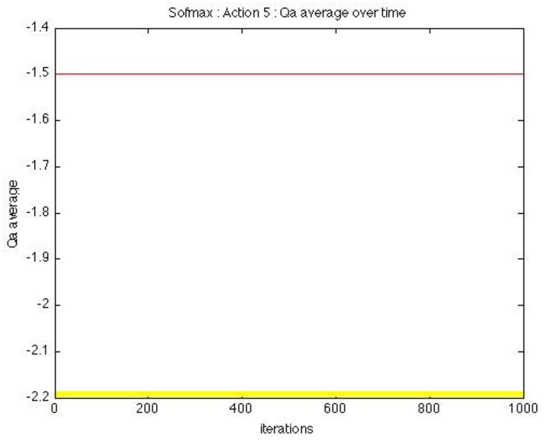
τ = 3

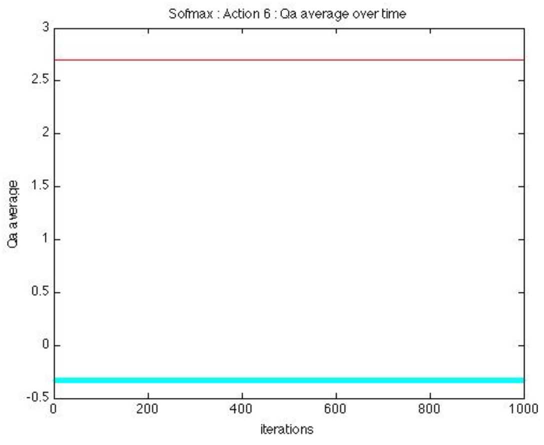




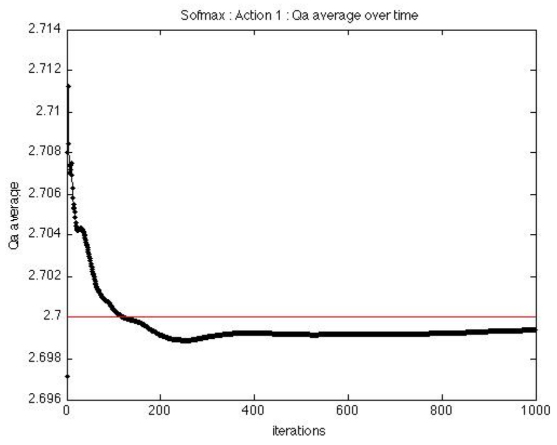


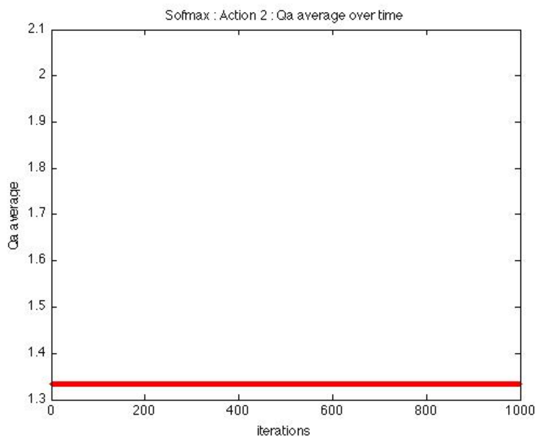


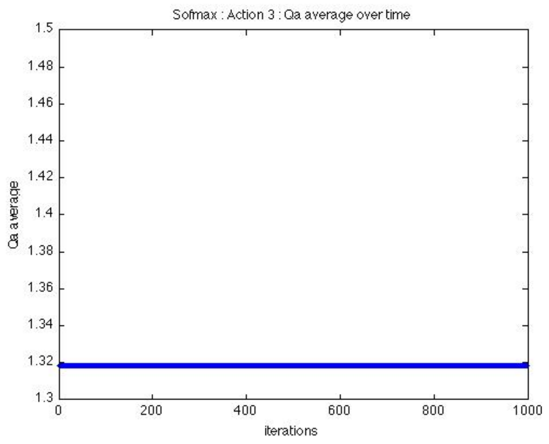


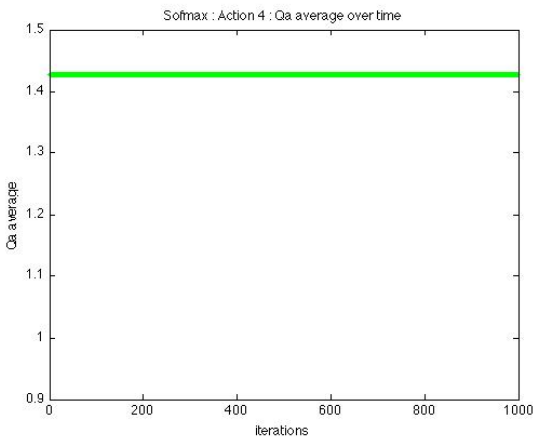


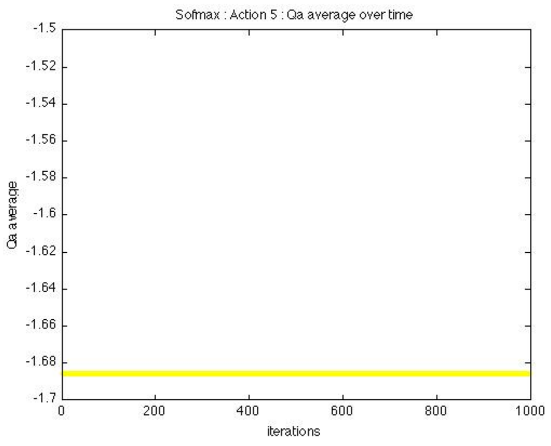
$\tau = 5 * (1000 - t)/1000$

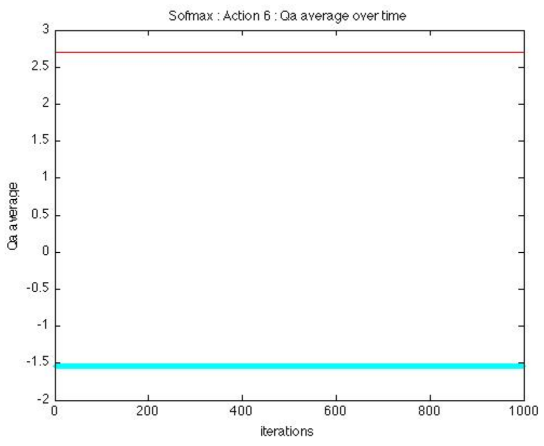












Analysis of the results

We observe that the best action's average quality Qa_i converges to the expected quality of this action when the simulation is launched a large amount of times.

In this example the best action is the first one. The diagram of that action shows that when:

$\tau = 3$, Qa has a higher value and decreases to reach 2.7 at infinity:

$$\lim_{i \to \infty^+} Qa_i = 2.7$$

$\tau = 5 * \frac{1000-t}{1000}$, Qa has a lower value and increases to reach 2.7 at infinity:

$$\lim_{i \to \infty^-} Qa_i = 2.7$$

The other diagrams show that the average quality of "bad" actions tends to keep the same value over simulations. This is because Softmax algorithm finds the solution pretty fast so the Qa value of the "bad" actions has no time to diverge so much from one simulation to the next. The value of the quality average for "bad" actions should be close to the first value that Qa gets randomly on each simulation.

Question 3: Over/Underestimating Q-values

Introduction

In this simulation, we use only 3 actions that are action #2, #4 and #6 from the specifications. So on our diagrams,

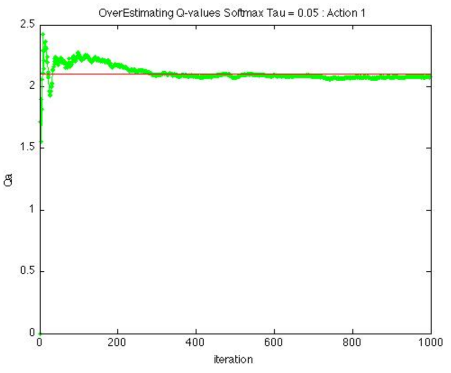
action #1 = action #2,
action #2 = action #4,
action #3 = action #6.

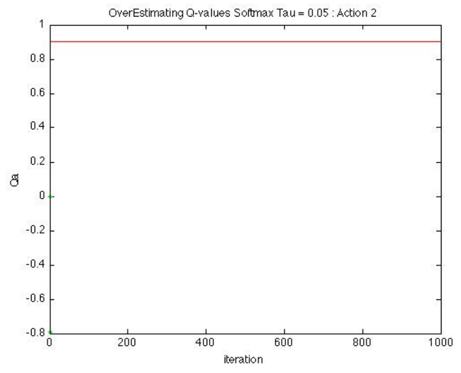
For the first simulation, we will underestimate Q-values so that they are initialized to -10. The second simulation will do the contrary the Q-values will be initialized to 10.

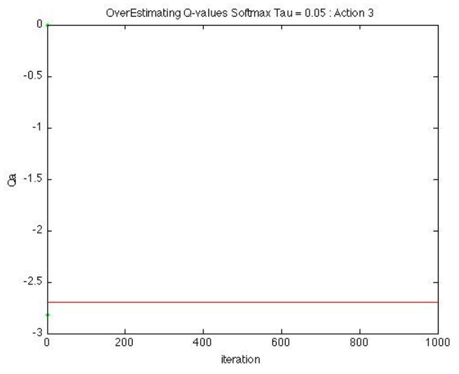
Plots

Overestimating

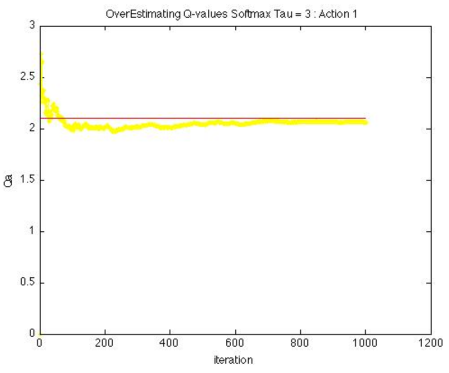
τ = 0.05

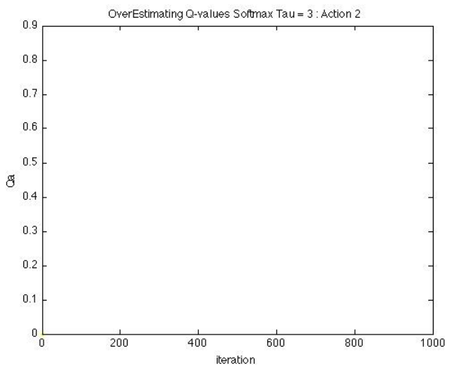


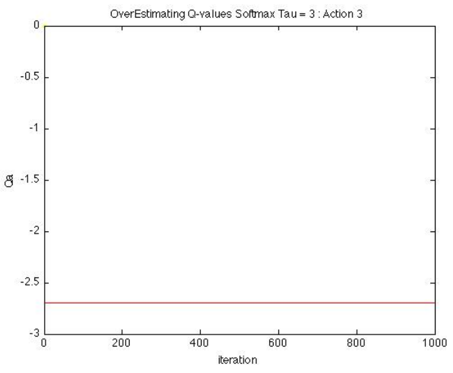




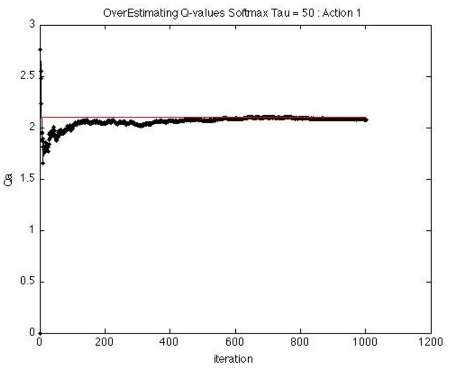
τ = 3

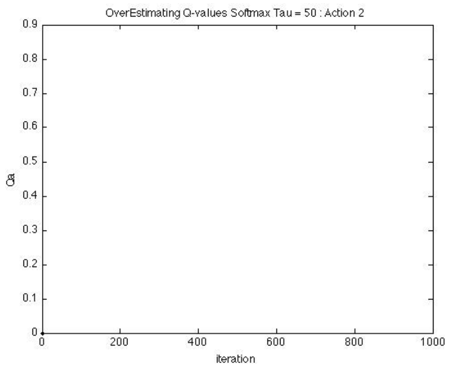

OverEstimating Q-values Softmax Tau = 3 : Action 1
Qa
iteration

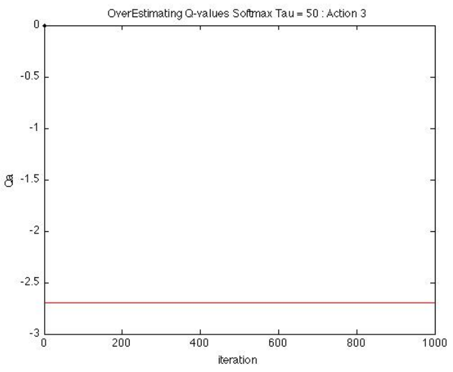

OverEstimating Q-values Softmax Tau = 3 : Action 2
Qa
iteration


OverEstimating Q-values Softmax Tau = 3 : Action 3
Qa
iteration

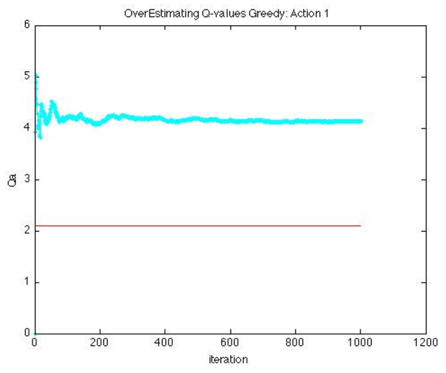
τ =50

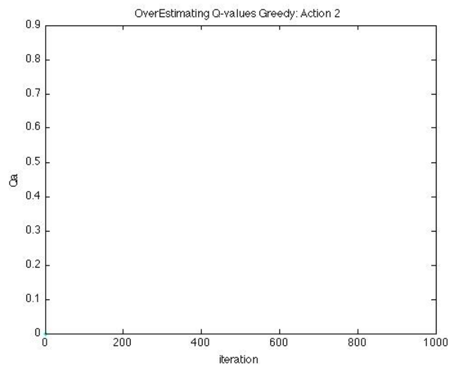

OverEstimating Q-values Softmax Tau = 50 : Action 1
Qa
iteration

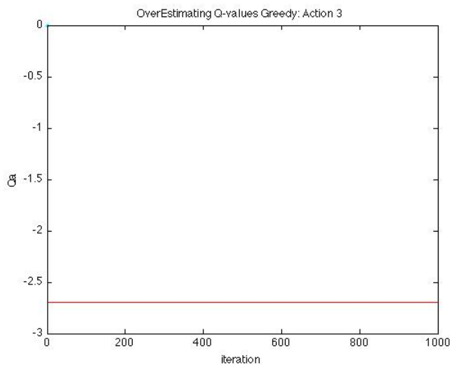

OverEstimating Q-values Softmax Tau = 50 : Action 2
Qa
iteration


OverEstimating Q-values Softmax Tau = 50 : Action 3
Qa
iteration

Greedy

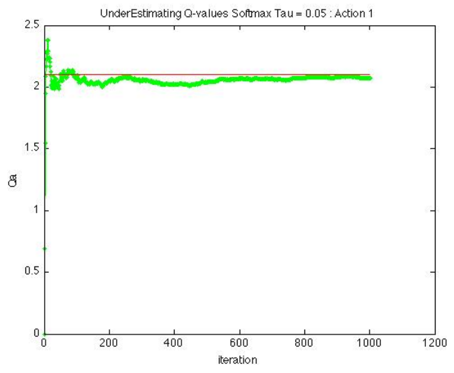


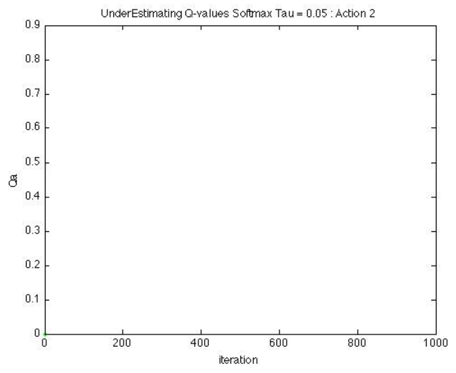


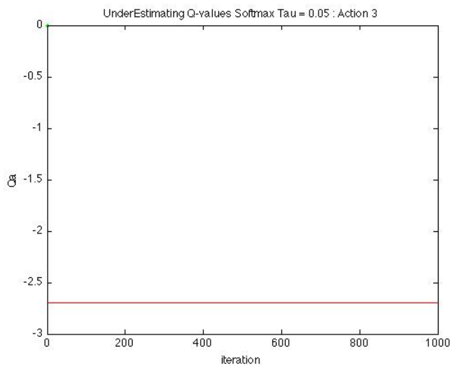


Underestimating

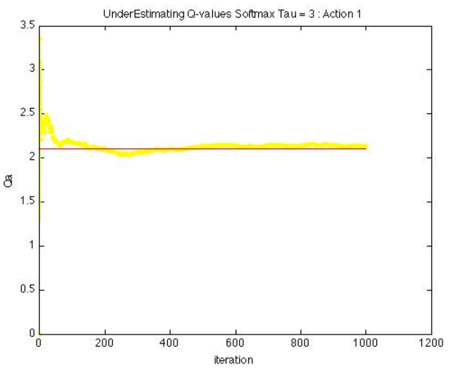
τ = 0.05

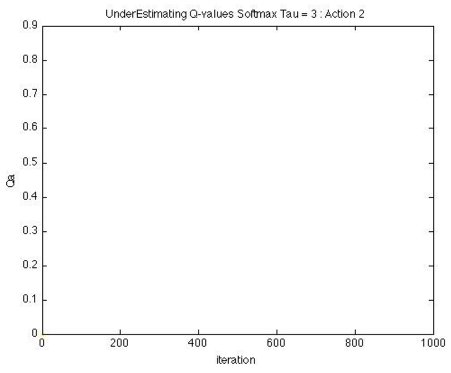


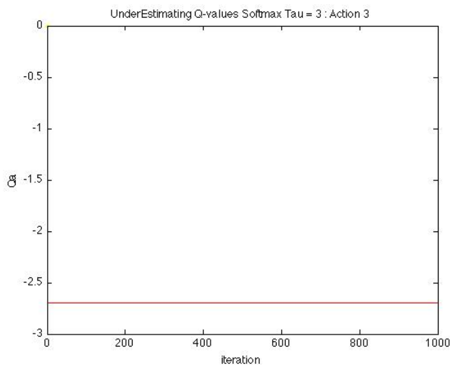




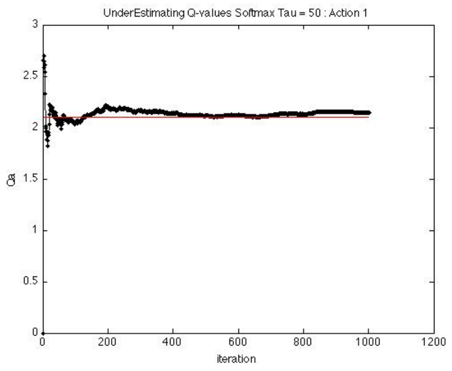
τ = 3

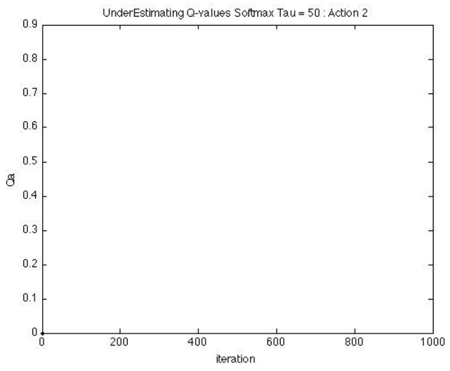

UnderEstimating Q-values Softmax Tau = 3 : Action 1
Qa
iteration

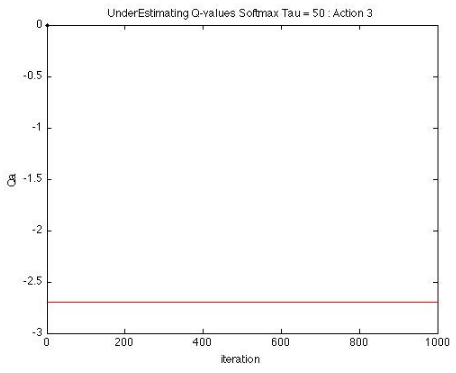

UnderEstimating Q-values Softmax Tau = 3 : Action 2
Qa
iteration


UnderEstimating Q-values Softmax Tau = 3 : Action 3
Qa
iteration

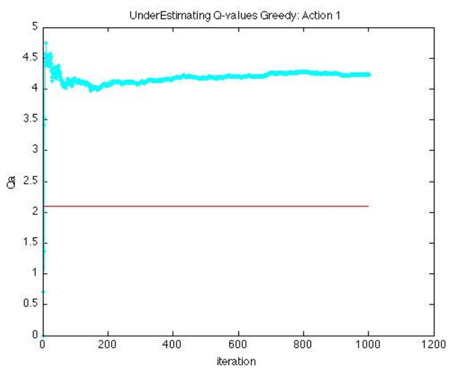
τ = 50

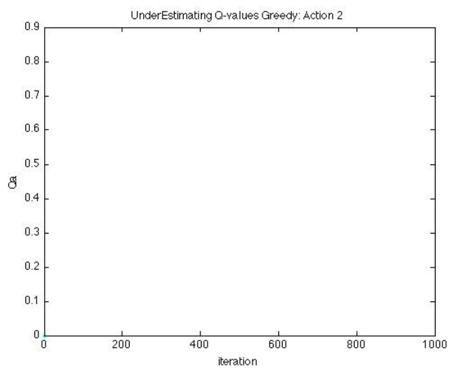

UnderEstimating Q-values Softmax Tau = 50 : Action 1
Qa
iteration

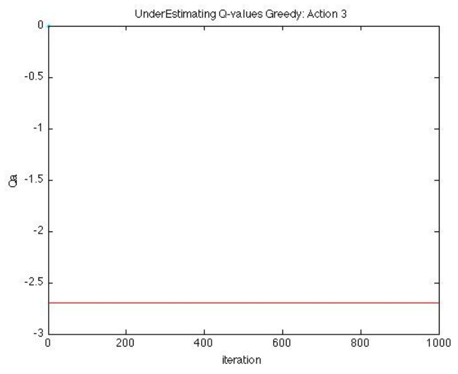

UnderEstimating Q-values Softmax Tau = 50 : Action 2
Qa
iteration


UnderEstimating Q-values Softmax Tau = 50 : Action 3
Qa
iteration

Greedy







Analysis of the results

We can easily see on the diagrams that Softmax method always gets the best solution regardless of the initialized values of Qa. The Greedy algorithm also finds the best solution but estimates poorly the expected values of the action's quality. None of the two methods explores much but Softmax is definitely better than Greedy because it always gets a very close estimation of the expected quality.

Conclusion

This assignment was a very good introduction to the world of “Learning Dynamics”. The most important thing is that no algorithm is perfect. They have all advantages and inconvenient that have to be understood and managed to produce good agents. We have seen that Greedy selection method can give good results but is likely to follow a path leading to a solution that is not the best because it exploit more than explore. The ε-Greedy partly correct this problem. However it doesn’t always find the best solution either. The Sofmax algorithm is definitely more complex; it leads to very good solution and estimation of expected quality.

This piece of work gave me the opportunity to use the knowledge acquired in class and to implement interesting algorithms on a simple problem.