Université libre de Bruxelles

INFO-F-409 - Learning Dynamics

Assignment Three

Multi-Armed Bandits

Raymond Lochner - 000443637

raymond.lochner@ulb.ac.be

Contents

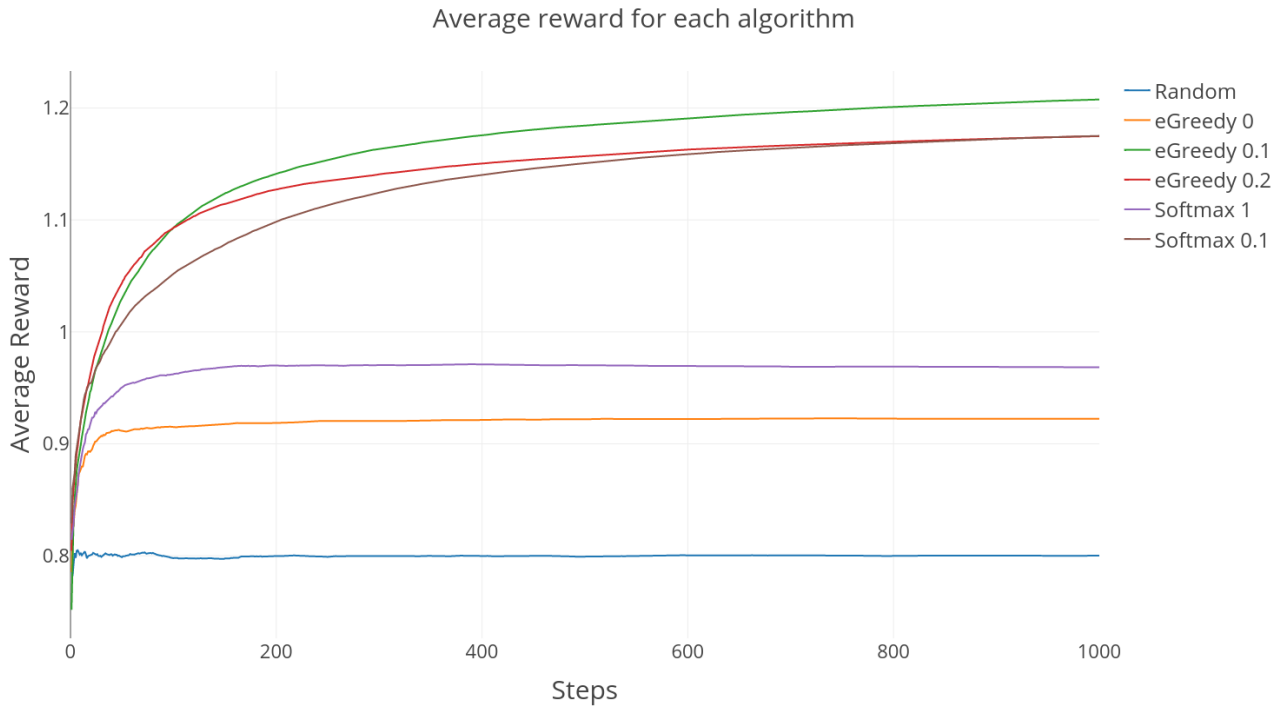
December 4, 2017

Preliminary information

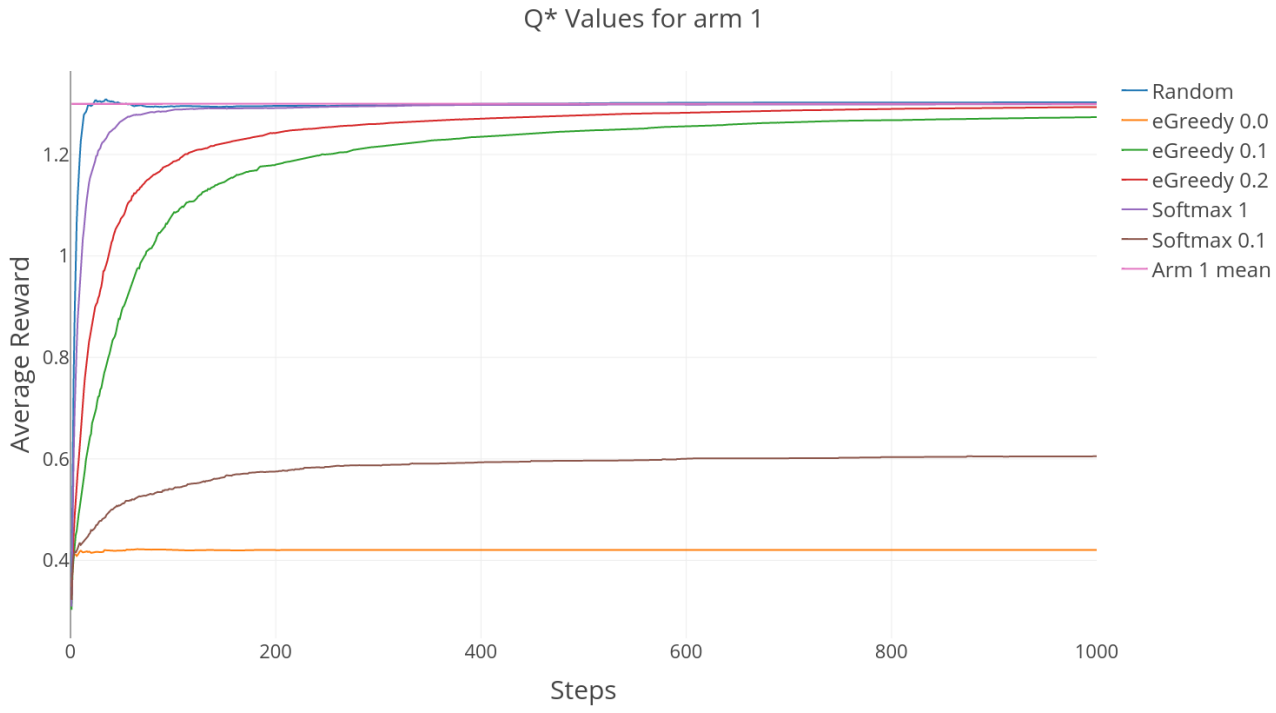
Every experiment was run 2000 times and the results averaged

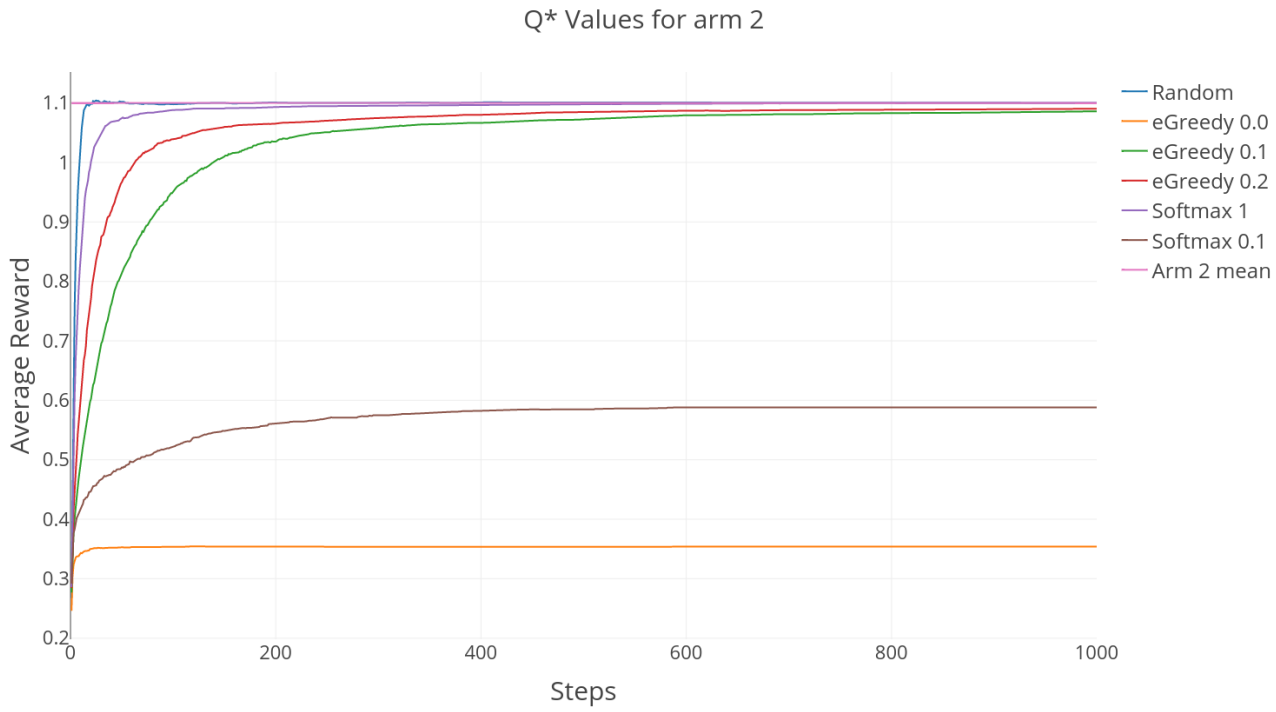
1 Exercise 1

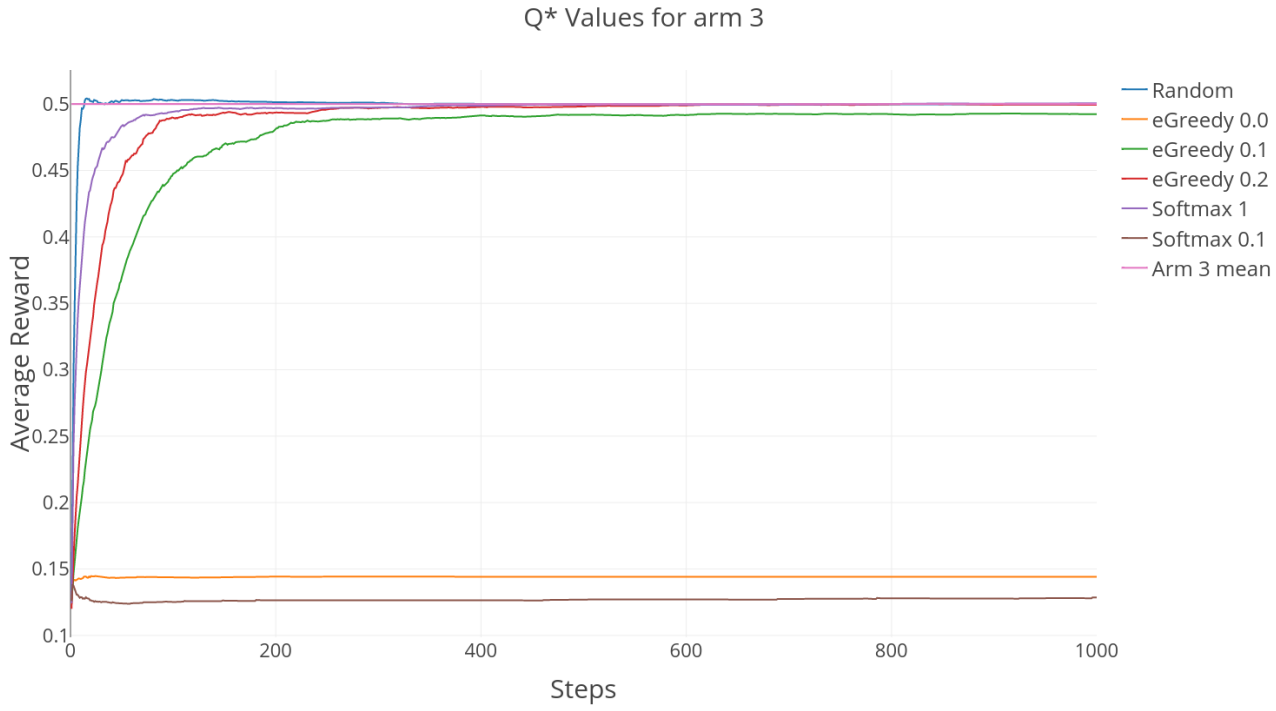
1.1 Combined Average Reward

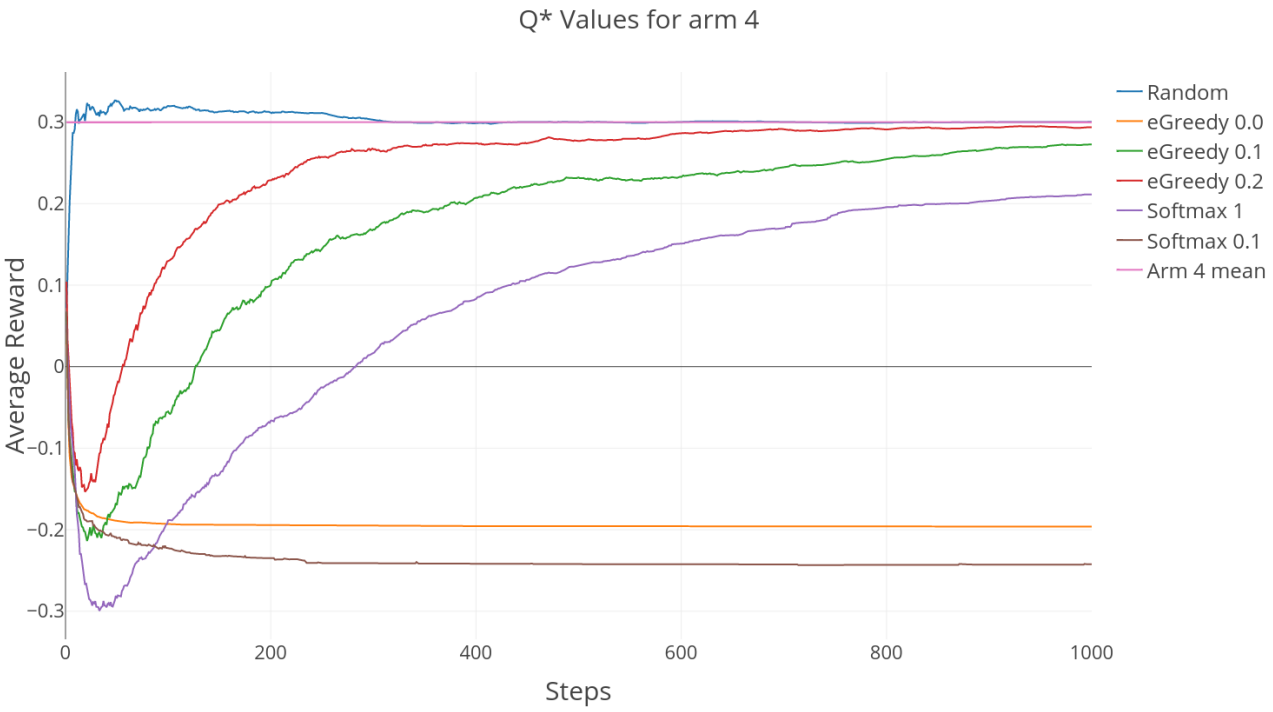


1.2 Q* values for each arm

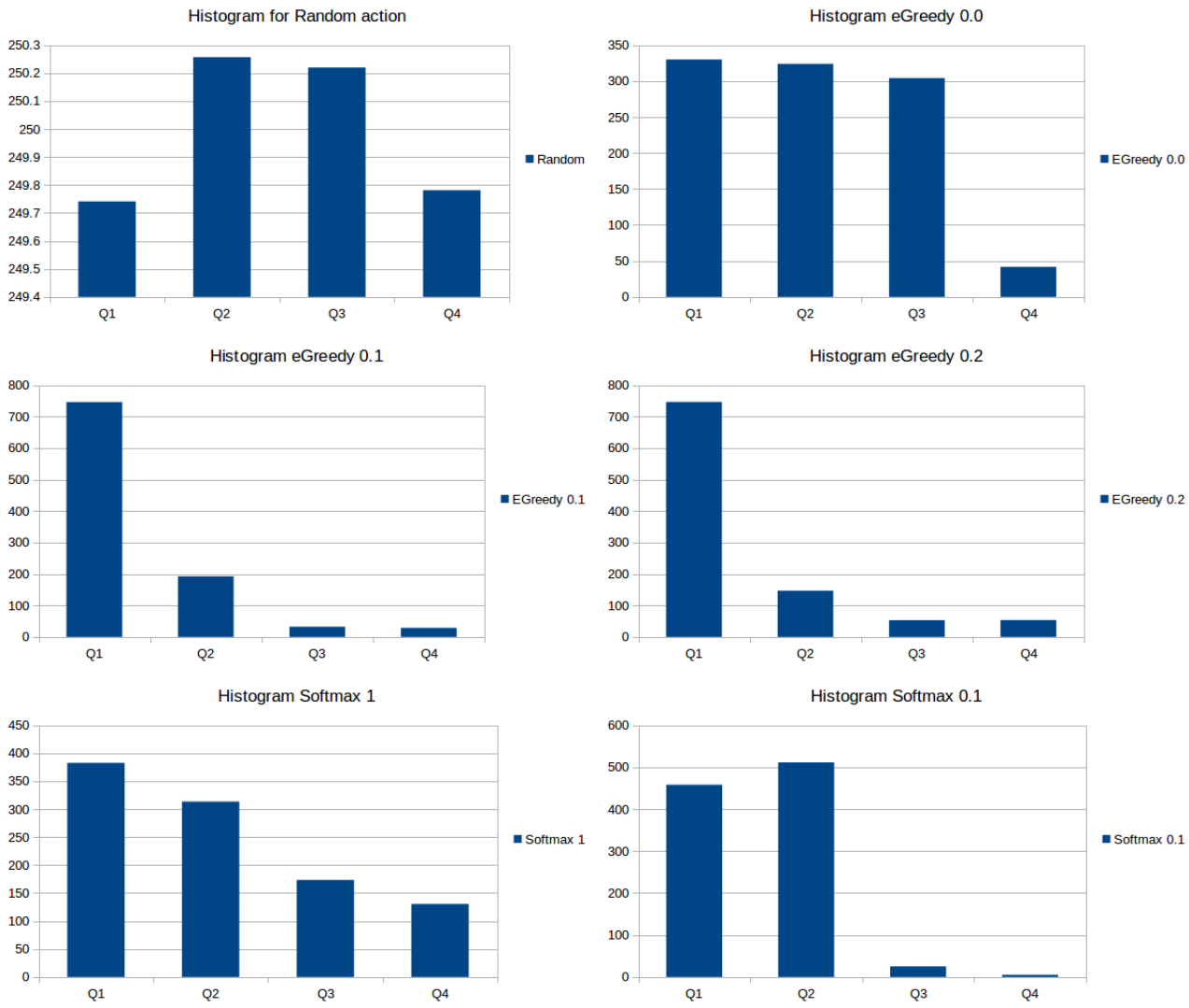



Q* Values for arm 2
Average Reward
Steps
Random
eGreedy 0.0
eGreedy 0.1
eGreedy 0.2
Softmax 1
Softmax 0.1
Arm 2 mean
1.1
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
200
400
600
800
1000


Q* Values for arm 3
Average Reward
Steps
Random
eGreedy 0.0
eGreedy 0.1
eGreedy 0.2
Softmax 1
Softmax 0.1
Arm 3 mean
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0
200
400
600
800
1000



1.3 Histogram on actions chosen



1.4 Results

We observe the combined average reward figure: The *eGreedy 0.1* method has the highest average reward after 100 steps until the end of the simulation of 1000 steps. *eGreedy 0.2* and *Softmax 0.1* share the second place together. *Softmax 1* comes in fourth place, *eGreedy 0.0* on fifth and the random algorithm last.

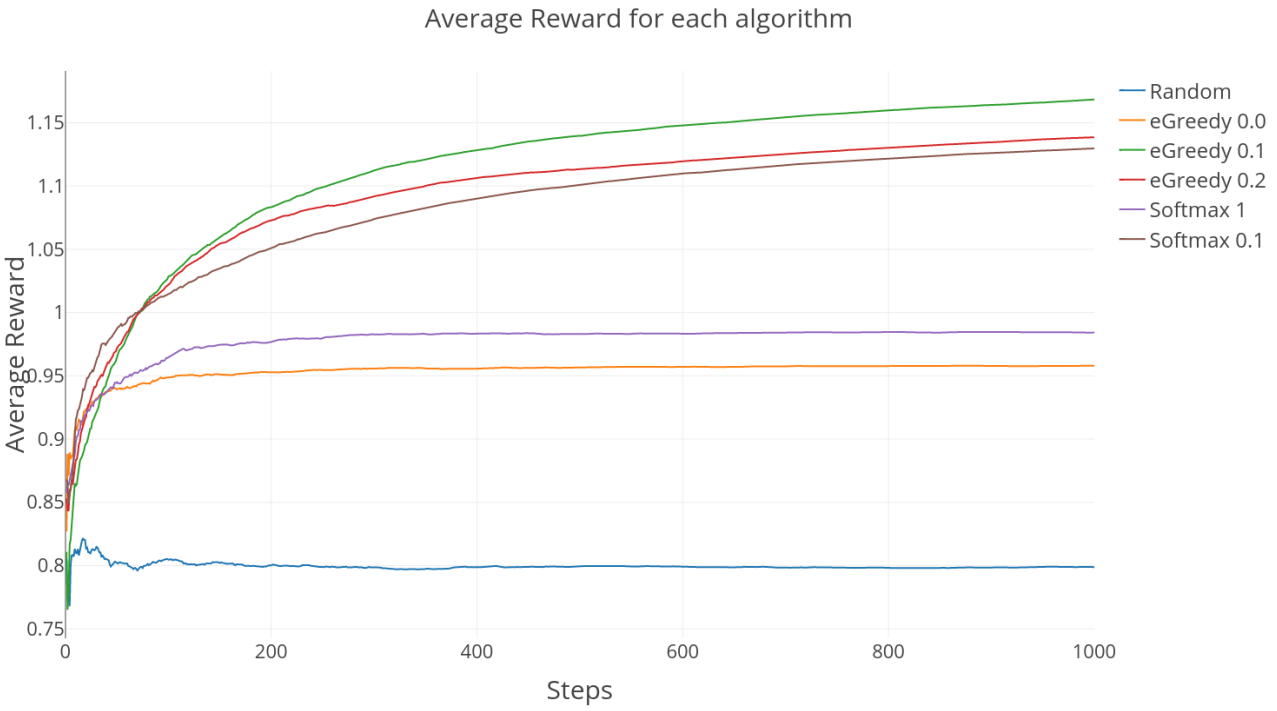
These results can also be observed by looking at the histograms where *eGreedy 0.1* has the highest arm 1 selection.

The random algorithm arrives the fastest at determining the Q* value for every arm but does not yield a good average. This could indicate that random exploration is a good method in the beginning but has to stop after some steps to select a single arm afterwards to earn the maximum award.

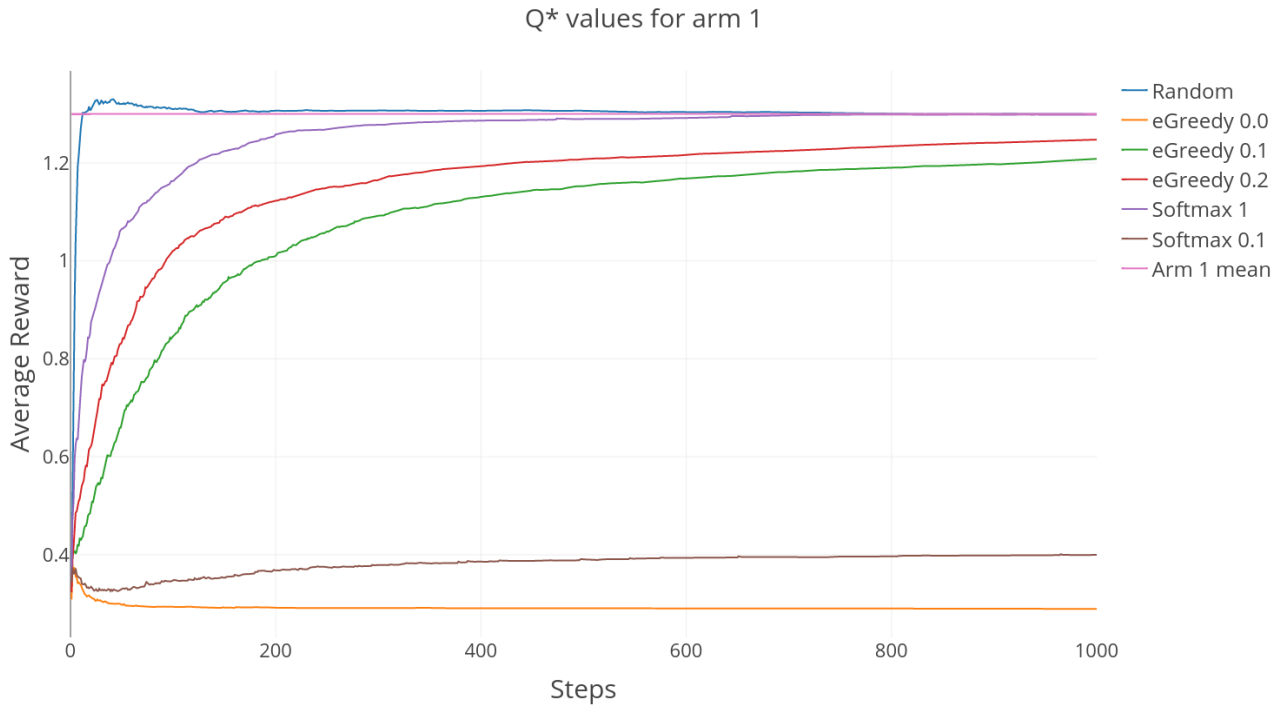
The graphs that show the Q* values for each arm show us that the two methods *eGreedy 0.0* and *Softmax 0.1* are not exploring enough in the beginning as they do not arrive at the actual Q* value and are stuck with a action they selected in the very beginning. This is due to the fact that we are initializing all Q* values to 0 at step 0 which results in the algorithm sticking to one action with a very high percentage without having explored the other actions.

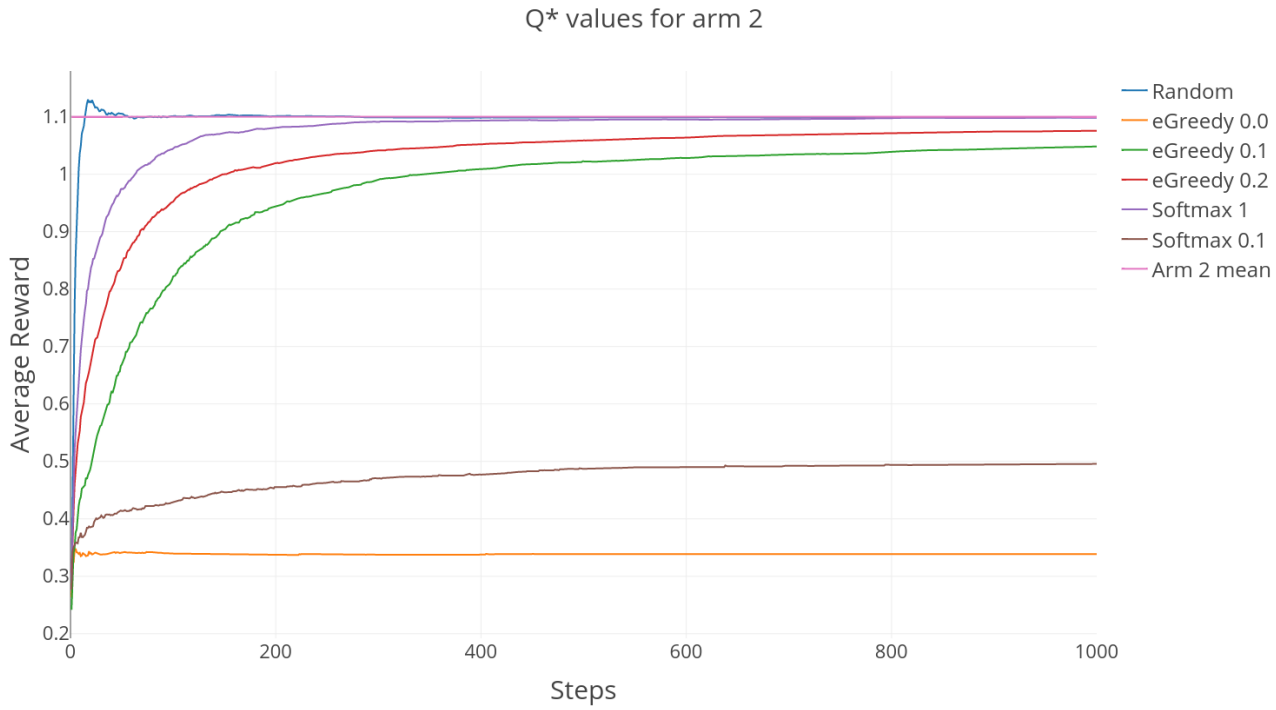
2 Exercise 2

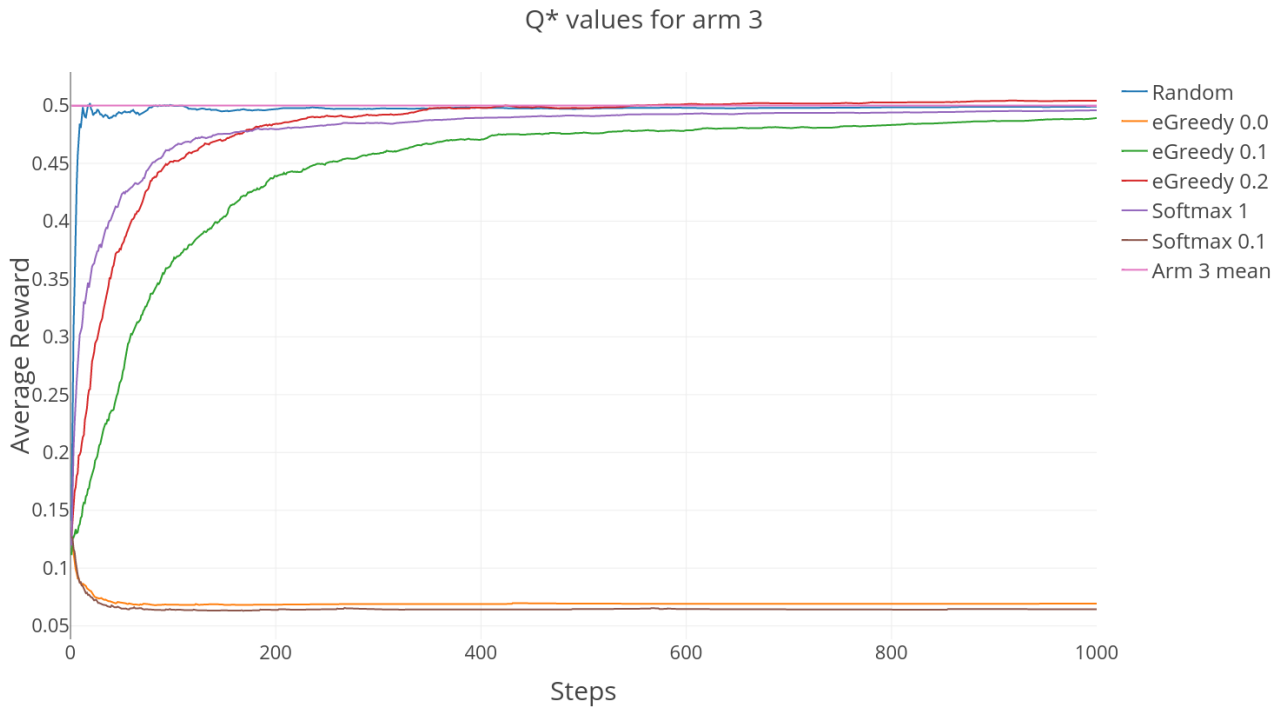
2.1 Combined Average Reward

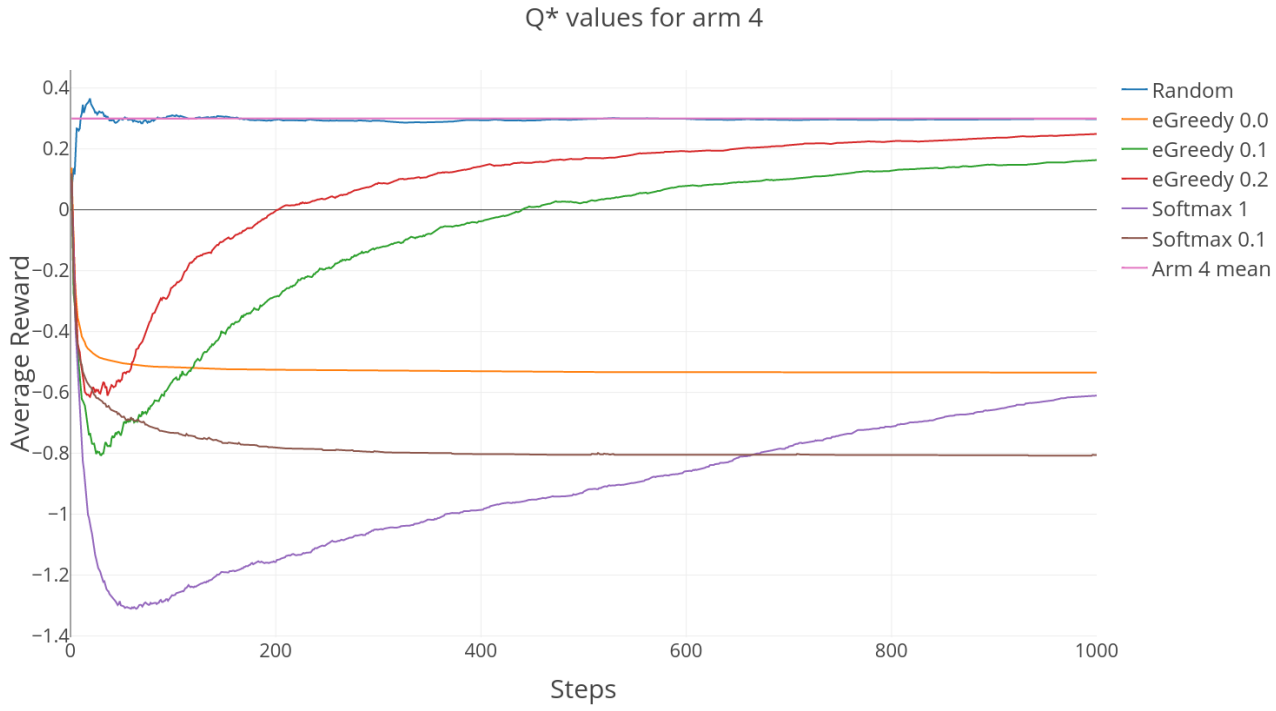


2.2 Q* values for each arm

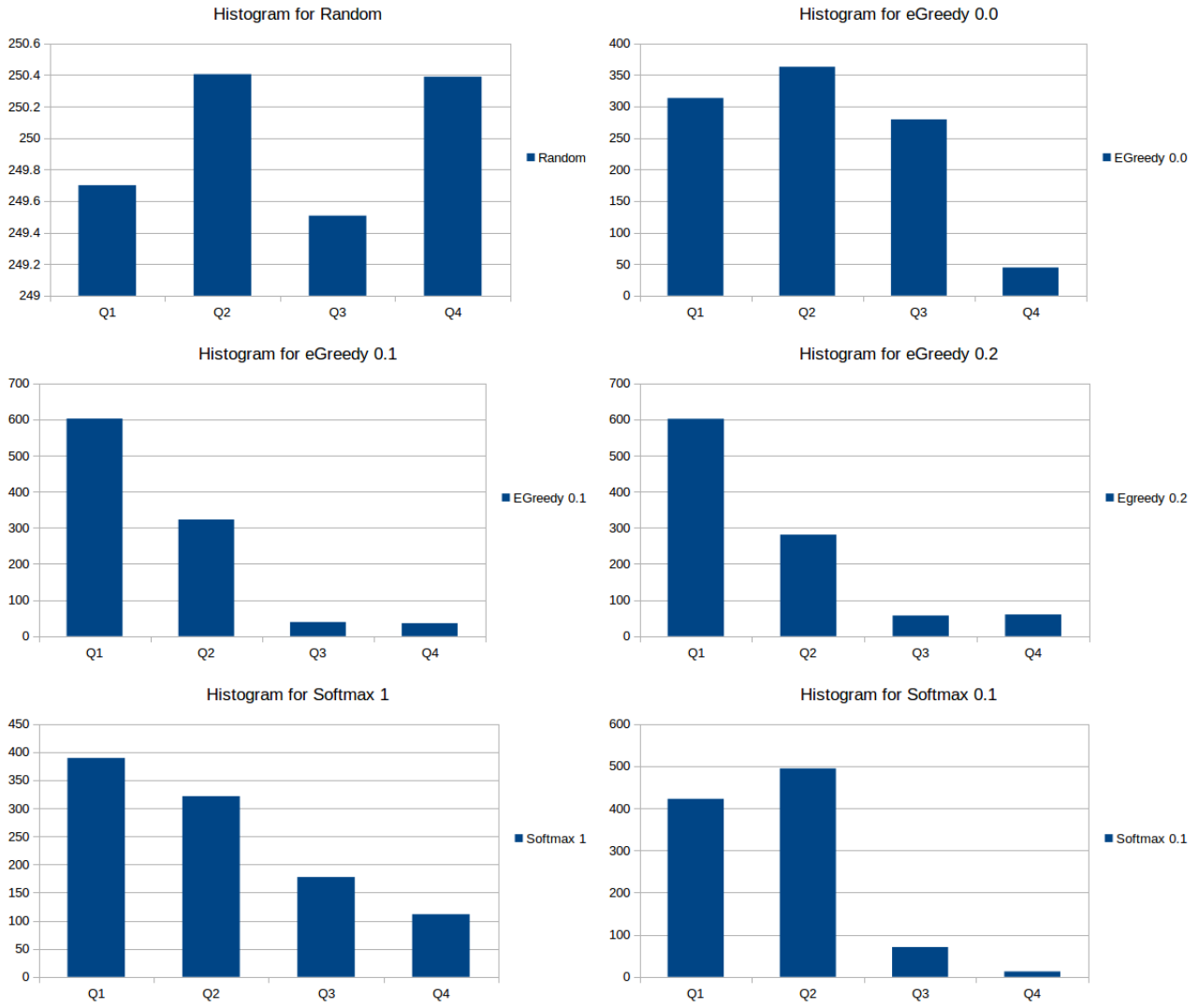





Q* values for arm 3
Average Reward
Steps
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
200
400
600
800
1000
Random
eGreedy 0.0
eGreedy 0.1
eGreedy 0.2
Softmax 1
Softmax 0.1
Arm 3 mean


Q* values for arm 4
Average Reward
Steps
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1
−1.2
−1.4
0
200
400
600
800
1000
Random
eGreedy 0.0
eGreedy 0.1
eGreedy 0.2
Softmax 1
Softmax 0.1
Arm 4 mean

2.3 Histogram on actions chosen



2.4 Results

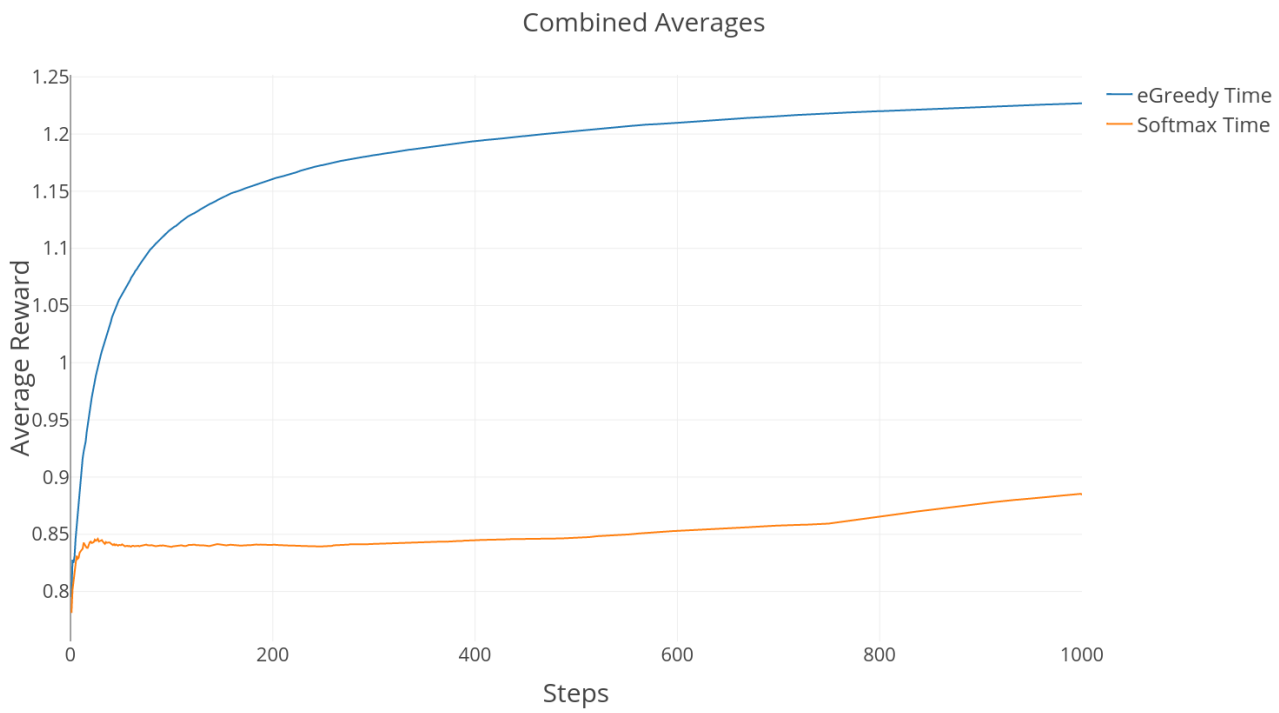
Doubling the deviation does not change the order of best algorithms with 1000 steps. The problem is harder to learn as the Q* values are similar to exercise one, but they are doing so with more steps.

The performance, or total reward after 1000 steps influences the used algorithms differently. It decreases the reward for the *eGreedy 0.1* and *eGreedy 0.2* methods but increases the reward slightly for the *Softmax 0.1* method.

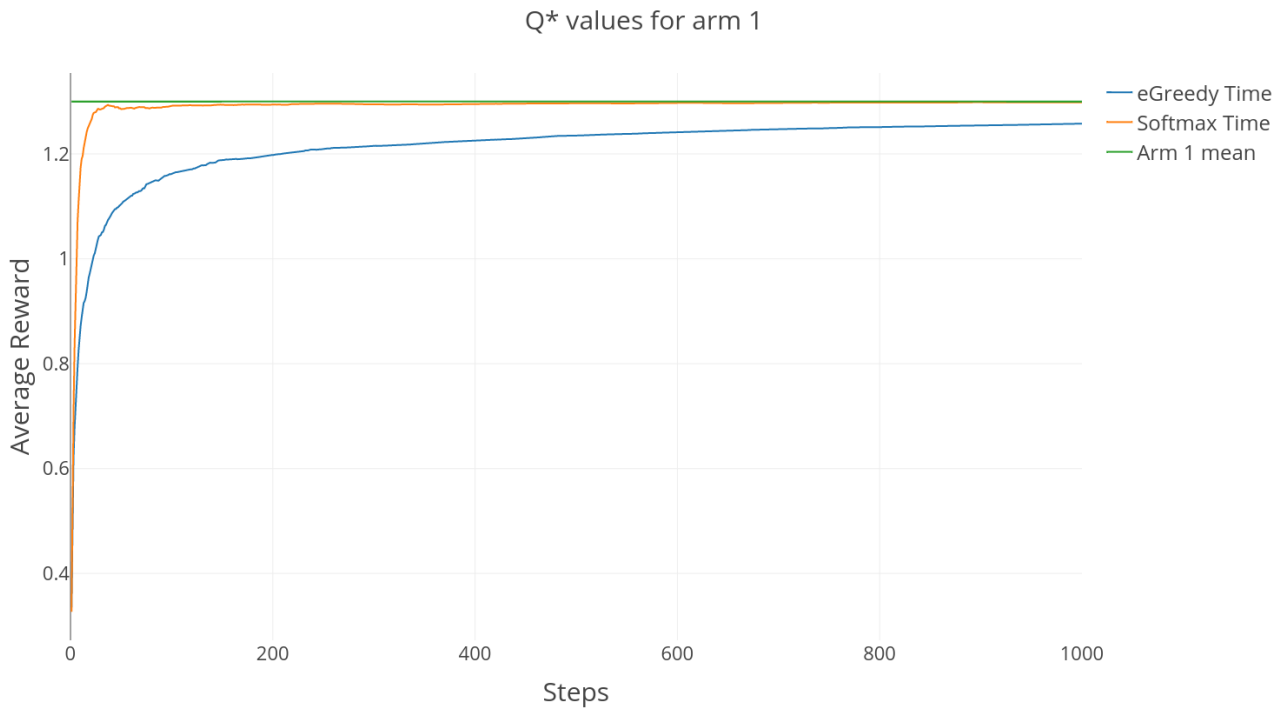
The average reward per algorithm is lower than compared to the first exercise. This can be explained due to the fact that the doubling of the deviation makes the Q* values more *random* and hence influences the probability of lower performing arms to be chosen in the beginning.

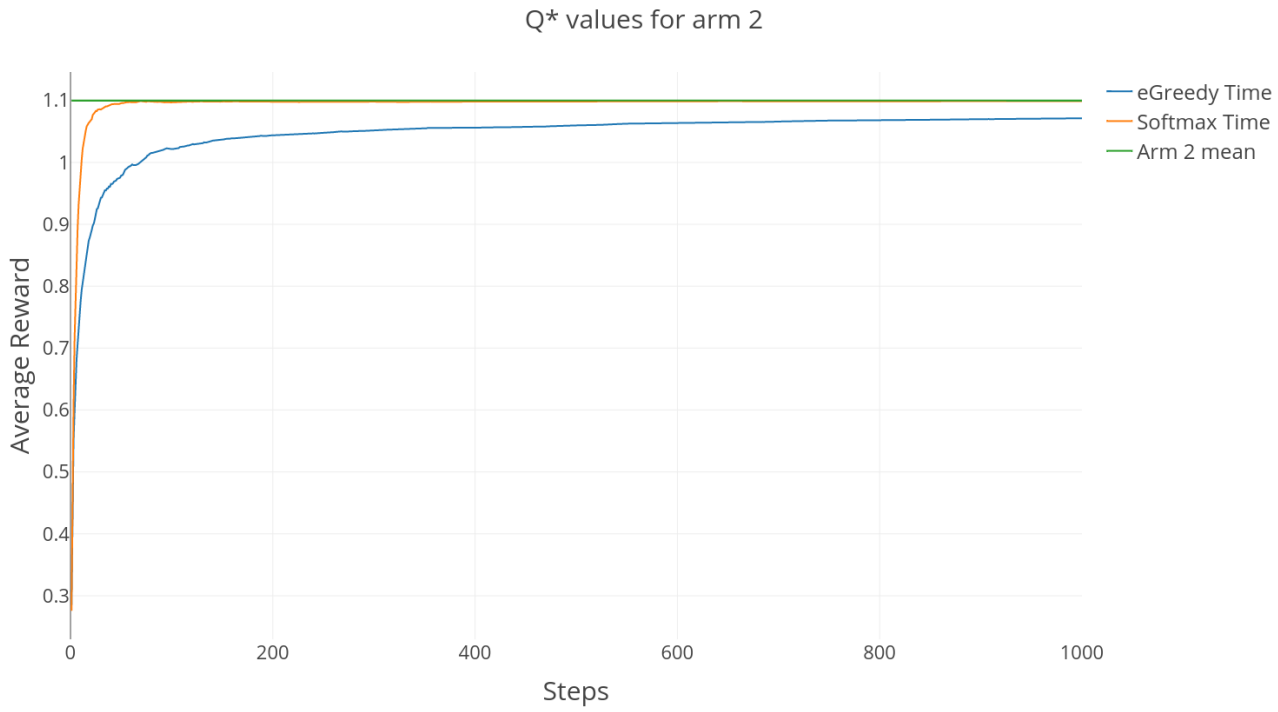
3 Exercise 3

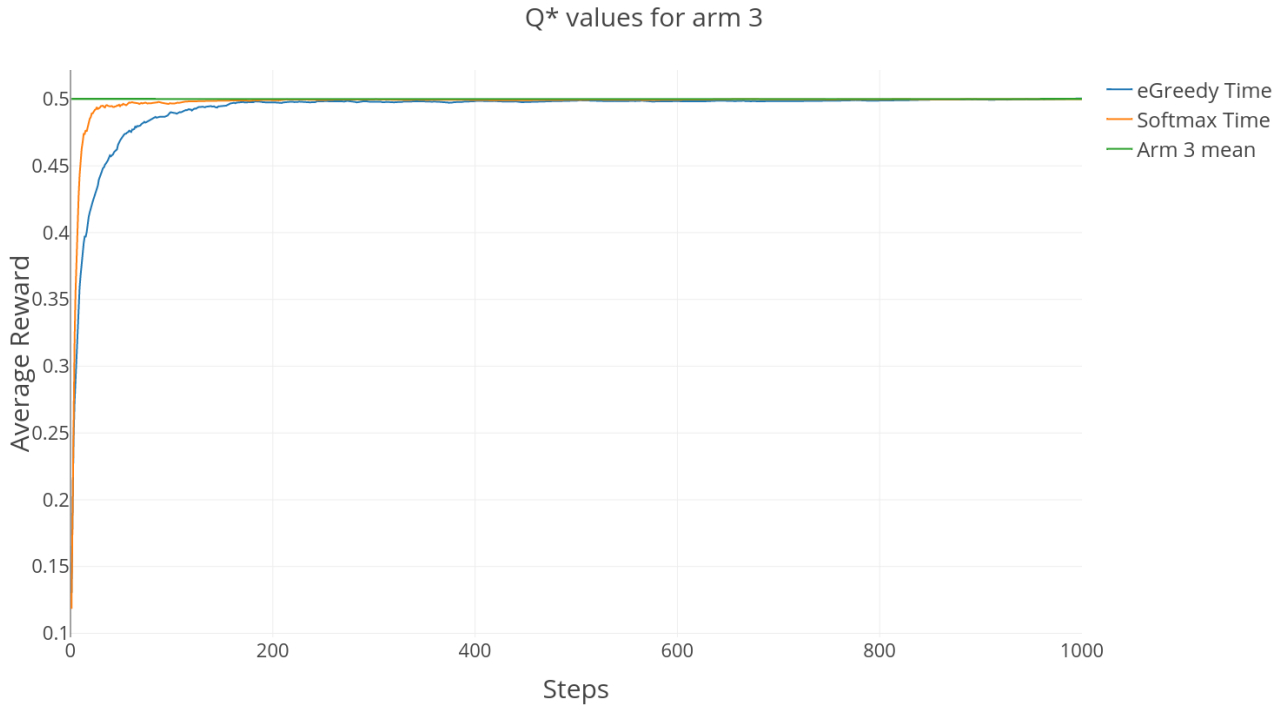
3.1 Combined Average Reward

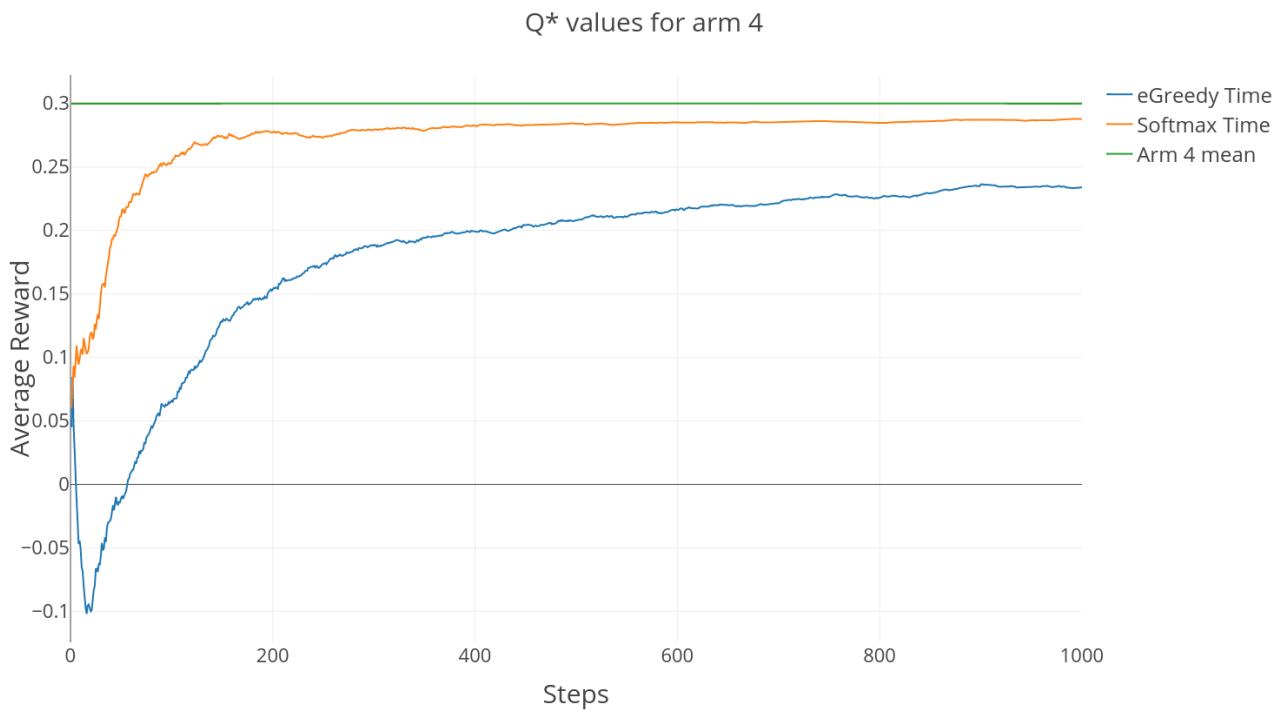


3.2 Q* values for each arm

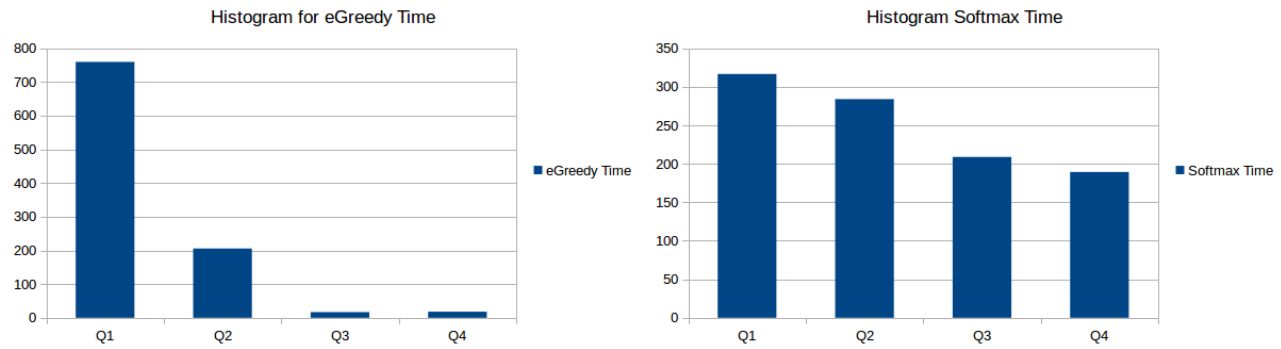



Q* values for arm 2
eGreedy Time
Softmax Time
Arm 2 mean
Average Reward
1.1
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
200
400
600
800
1000
Steps


Q* values for arm 3
eGreedy Time
Softmax Time
Arm 3 mean
Average Reward
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0
200
400
600
800
1000
Steps



3.3 Histogram on actions chosen



3.4 Results

The new *eGreedy* algorithm performs better than any other algorithm. The new *Softmax* algorithm performs only better than the *random* algorithm after 1000 time steps but has an increasing curve. Over a longer time-period, it would out-perform the bottom three algorithms - *Random*, *eGreedy 0.0* and *Softmax 1*.

From the Q* value graphs we can observe that the *Softmax Time* method is able to find the mean value of each arm faster than the *eGreedy Time* method due to its higher exploration at the beginning. It does not outperform it though, as it exploits the best arm later.

We can conclude from these experiments that the best way to achieve the highest reward over time is to explore at the beginning and then only exploit after some time. The downside of the *eGreedy* algorithms is that they keep exploring even though they already *learned* the best arms.