## Markov Decision Processes and Reinforcement Learning

Machine learning is not only about function approximation, nor about data description only. Another interesting topic is about estimating the action that agent should act in a certain environment in order to maximum some kind of reward. Reinforcement learning, instead of supervised or unsupervised learning, is a very useful tool for this.

A general framework for this type of action-choosing (policy) optimization issue is the Markov decision processes. Under the framework, the environment is described in a Markovian process, which involve the current state (s), the next state (s'), and the transitional probability (P), which only depends on the current state, and the action chosen by the agent (a). The agent can get a reward (R) from the move.

The ultimate goal of reinforcement learning is to estimate the optimal policy that the agent should take in a given environment, where the transition probability and the reward are both unknown, from limited given data (previous experience). This is a very useful skill as this can be generalized as systemic way of making decision based on previous experience and is applicable in many real-world situation and practical applications.

In this assignment, two environments, an easier and a harder one are setup. First, the transition probability and the reward function are given, and the Markov Decision Processes are solved. Secondly, the transition function and the reward function are assumed to be not given, and the Q-learning (the most commonly used reinforcement learning method) is applied to estimate the optimal policy. Different approach and parameters setting are used in the process to test the effect of that and investigate which is the suitable one.

## Robot in a Maze

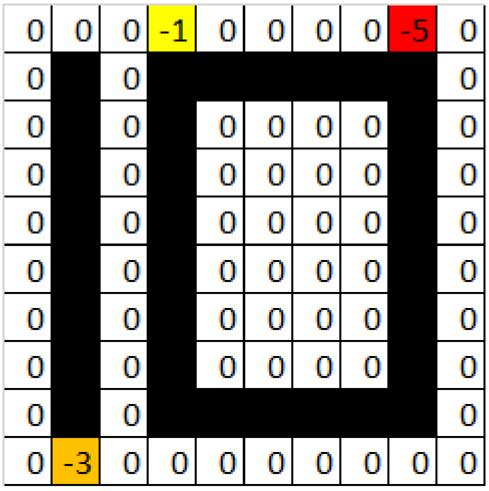
The environment and the agent under concern in this assignment will be a robot in a maze. The goal of the task is to design a policy to guide the robot from the starting to the ending point with the maximum reward gained on the way.

This is a typical and interesting problem in reinforcement learning world as in this problem, the states are clearly well defined, and also the goal. It can also be easily visualized and easier understood and can be generalized to other environments easily. This type of problem are also very similar to planning issues in artificial intelligence, and thus also the related techniques.

The easier setup is a smaller maze with 10 X 10 (100 states in total), while the larger maze is 20 x 20 (400 states in total). The starting point is at the lower-left corner, and ending point is at the upper-right hand corner. By reaching the goal, a reward of 100 is gained, and the reward (and penalties which are to be mentioned later) are discounted by a factor of 0.99 each step. For each step taken, a reward of -1 (i.e. penalty of 1) is added, which encourage a shorter path than a longer one.

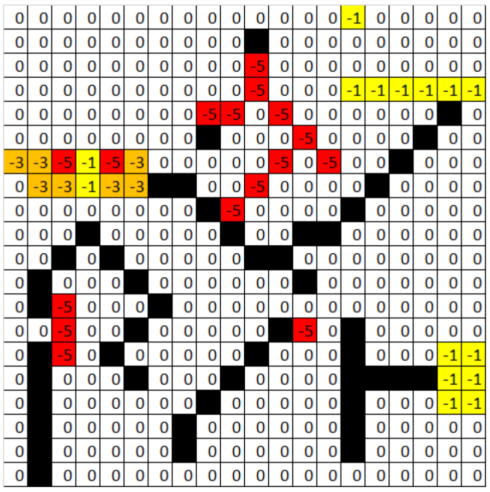
The exact reward of the two mazes (in additional to the penalty of 1 per step) are as follows:

Easy Maze



| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | -5 | 0 |
| 0 | ■ | 0 | ■ | ■ | ■ | ■ | ■ | ■ | 0 |
| 0 | ■ | 0 | ■ | 0 | 0 | 0 | 0 | ■ | 0 |
| 0 | ■ | 0 | ■ | 0 | 0 | 0 | 0 | ■ | 0 |
| 0 | ■ | 0 | ■ | 0 | 0 | 0 | 0 | ■ | 0 |
| 0 | ■ | 0 | ■ | 0 | 0 | 0 | 0 | ■ | 0 |
| 0 | ■ | 0 | ■ | 0 | 0 | 0 | 0 | ■ | 0 |
| 0 | ■ | 0 | ■ | 0 | 0 | 0 | 0 | ■ | 0 |
| 0 | ■ | 0 | ■ | ■ | ■ | ■ | ■ | ■ | 0 |
| 0 | -3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Hard Maze



| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ■ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -5 | 0 | 0 | 0 | -1 | -1 | -1 | -1 | -1 | -1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -5 | -5 | 0 | -5 | 0 | 0 | 0 | 0 | 0 | 0 | ■ | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ■ | 0 | 0 | 0 | -5 | 0 | 0 | 0 | 0 | ■ | 0 | 0 |
| -3 | -3 | -5 | -1 | -5 | -3 | 0 | 0 | 0 | 0 | 0 | -5 | 0 | -5 | 0 | 0 | ■ | 0 | 0 | 0 |
| 0 | -3 | -3 | -1 | -3 | -3 | ■ | ■ | 0 | 0 | -5 | 0 | 0 | 0 | 0 | ■ | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ■ | -5 | 0 | 0 | 0 | 0 | ■ | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | ■ | 0 | 0 | 0 | 0 | 0 | ■ | 0 | 0 | ■ | ■ | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | ■ | 0 | ■ | 0 | 0 | 0 | 0 | 0 | ■ | ■ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | ■ | 0 | 0 | 0 | ■ | 0 | 0 | 0 | 0 | 0 | 0 | ■ | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | ■ | -5 | 0 | 0 | 0 | ■ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | -5 | 0 | 0 | ■ | 0 | 0 | 0 | 0 | 0 | ■ | -5 | 0 | ■ | 0 | 0 | 0 | 0 | 0 |
| 0 | ■ | -5 | 0 | ■ | 0 | 0 | 0 | 0 | 0 | ■ | 0 | 0 | 0 | ■ | 0 | 0 | 0 | -1 | -1 |
| 0 | ■ | 0 | 0 | 0 | ■ | 0 | 0 | 0 | ■ | 0 | 0 | 0 | 0 | ■ | ■ | ■ | ■ | -1 | -1 |
| 0 | ■ | 0 | 0 | 0 | 0 | 0 | 0 | ■ | 0 | 0 | 0 | 0 | 0 | ■ | 0 | 0 | 0 | -1 | -1 |
| 0 | ■ | 0 | 0 | 0 | 0 | 0 | ■ | 0 | 0 | 0 | 0 | 0 | 0 | ■ | 0 | 0 | 0 | 0 | 0 |
| 0 | ■ | 0 | 0 | 0 | 0 | 0 | ■ | 0 | 0 | 0 | 0 | 0 | 0 | ■ | 0 | 0 | 0 | 0 | 0 |
| 0 | ■ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

The cell in black is a “wall”, i.e. somewhere that the robot cannot enter, and the -1, -3, -5 cells are in yellow, orange and red, which are cells that the robot are not expected to pass through.

In the first maze (the easier one), it will be interesting to see if the robot will be able to avoid the -3 at the lower-left hand corner and the -1 and -5 at the top. The -1 state is actually making the task easier as it is making the reward grids denser and serve as a warning of the deferred penalty of -5 on the right.

In the second maze (the hard one), it will be interesting to see if the robot will be able to avoid the path on the left with two -1 penalties and choose the right one with only one -1 penalty without any guideline or warning in advance, but by just sparse and deferred reward/penalty alone.

## Solving MDP – value iteration vs policy Iteration

The first goal is the solve this Markov Decision Processes (MDP). The transition probability and the reward function are known, and thus no specify empirical experience is needed as the underlying physics of the world is already known. The only things need to do is actually about designing the right policy for that.

There are generally two approaches to solving a Markov Decision Process problem: either through value iteration or policy iteration. The first one (value iteration) solves the MDP by first starting at arbitrary utilities of each state, then update the utilities based on neighbourhoods using the Bellman equation and iterate until the utility function converges. The optimal policies are then determined based on the utility of each states. The second one (policy iteration) solves the MDP by first starting at arbitrary policy of each state, then calculate the utility with the given policy, and improve the policy with the calculated utilities, and iterate the utility calculation and the policy improvement steps until it converges. The key equation of the two iteration methods are shown as below:

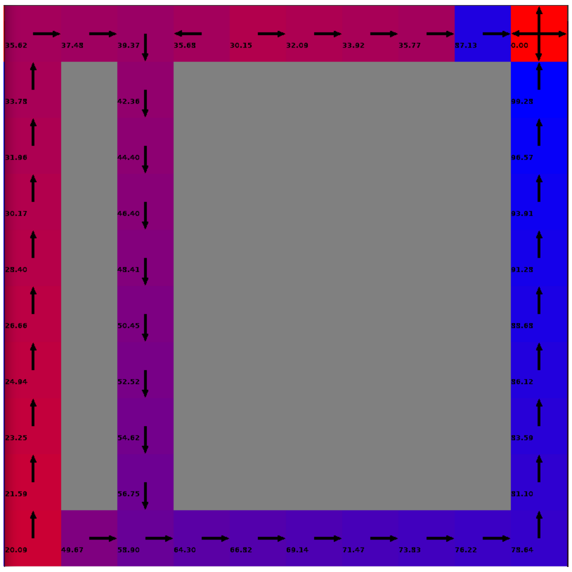
Value Iteration: $U_{t+1}(s) = R(s) + \gamma \max_a \{\sum_{s'} [T(s, a, s')\, U_t(s')\}$

Policy Iteration: $U_t(s) = R(s) + \gamma \sum_{s'} [T(s, \pi_t(s), s')\, U_t(s')]$

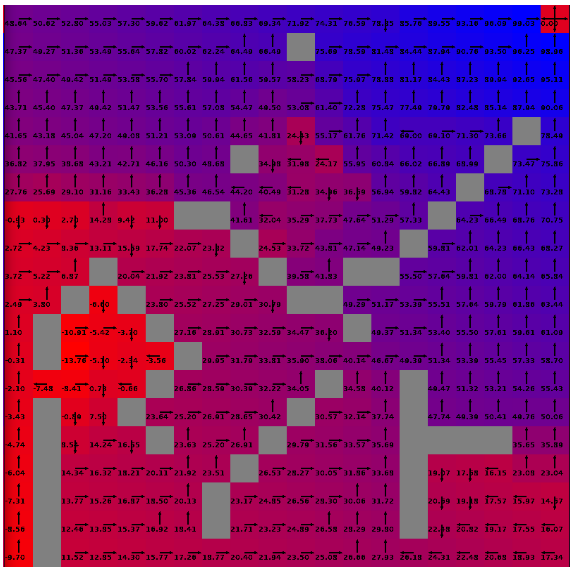
The utility function (U) is simply the sum of current reward, and all discounted future discounted reward under the optimal policy. R(s) is the reward of reaching a certain state s, and γ is the discount rate.

The results (utility and policy) of the two methods are shown as below:

Easy Maze



Hard Maze

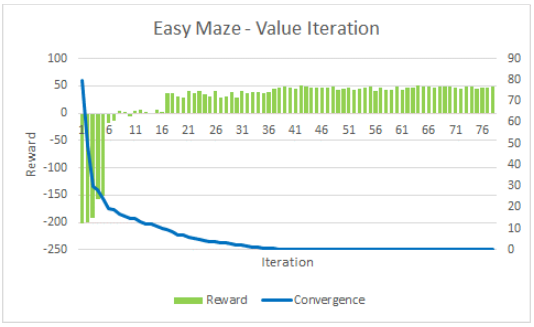


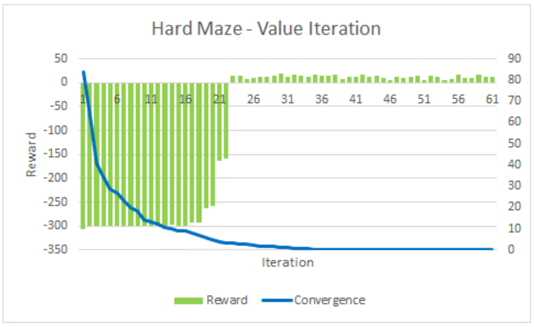
The graph shows the utility and policy of each states at convergence. The policy at the states mostly match our expectation.

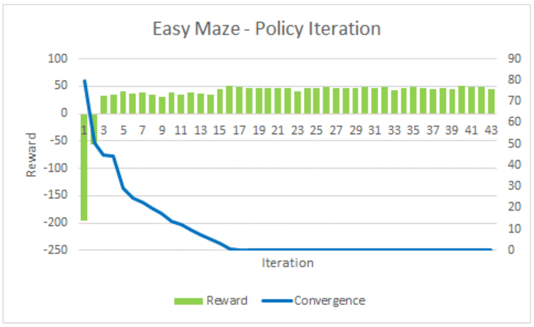
This is something as expected from just visualization of the maze. The robot avoids the penalties well, and this shows that dynamic programming is doing an excellent job in solving this Bellman equation.

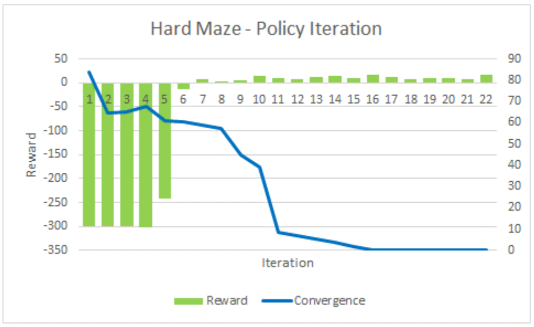
The convergences of both algorithm are good, and are shown as below:

<u>Easy Maze</u> <u>Hard Maze</u>









The policy iteration is found to be converging faster (in a few iteration) then value iteration. This is because by iterating in policy, the jump is in states which is just a few of them, instead of jumping in the continuous utility, and thus a bigger jump is allowed and converges faster.

| **Iteration required** | <u>Easy Maze</u> | <u>Hard Maze</u> |
|---|---|---|
| Value Iteration | 78 | 61 |
| Policy iteration | 43 | 22 |

However, if the speed is measured in terms of time spend instead of iteration, the policy iteration is found to be slower than value iteration.

| **Time Spent (second)** | <u>Easy Maze</u> | <u>Hard Maze</u> |
|---|---|---|
| Value Iteration | 1.54 | 2.47 |
| Policy iteration | 1.56 | 4.11 |

Notes: In the graph, the reward is the sum of all discounted reward at the initial sate by following the estimate optima policy, i.e. the utility at the initial state. The convergence is the average of the difference of the last ten revision, which measure how well the estimates are converging.

## **Reinforcement Learning**

Now, the transition probability and the reward function are assumed to be unknown, and thus dynamic programming (either value or policy iteration) can be applied to solve the MDP.

### An alternative Form of Utility Function, Q(s, a)

Q-Learning is applied to solve the problem under this case. It aims at estimating the Q function, which is the utility function but a certain action is forced taken, and is denoted by $Q(s,a) = R(s) + \gamma \sum_{s'} \{T(s, a, s') \max_{a'} [Q(s',a')]\}$.

This alternative form of utility function can be estimated easier, and then as long as the Q function is estimated, the utility function can be calculated by $\max_{a} [Q(s,a)]$, and the optimal policy is $\operatorname{argmax}_{a} [Q(s,a)]$

### Estimate Q(s,a) from Transition Probability

By updating the $Q(s, a) \leftarrow (1 - \alpha) Q(s, a) + \alpha (r + \gamma \max_{a'} [Q(s', a')])$ iteratively, where r denote one observed reward, (which is a form of value iteration as Q is just an alternative form of utility function), the Q function can be estimate by observed data (also known as previous experience), which consist of the transition probability and reward.

### Greedy Exploring

Just like many other machine learning algorithm, Q-learning possibly may face local optimum issue while estimation. Therefore, it will be good to explore somehow in the process, in additional to just doing iteratively update of Q(s, a) mentioned above (exploitation of existing information).

The most common way of doing so is to introduction a probability $\varepsilon$ that the policy is not chosen as the optimal one estimated by the Q function and takes a random action instead. By setting $\varepsilon$ as a decreasing function as iteration increase, the estimate of Q(s,a) and the $\pi(s)$ is guaranteed to convergence to the true one, given the underlying nature of the environment remain unchanged.

### Choosing the Parameters

$\alpha$ and $\varepsilon$ are parameters that to be chosen by the user. As discussed, they control the learning rate, and the exploration rate in the learning process.

In this assignment, the learning algorithms are run multiply number of times with starting $\alpha$ and $\varepsilon$ equal to value of 0.1, 0.2, 0.3, 0.4, and 0.5, i.e. 5 X 5 = 25 times. The parameter pair with the best rewards are chosen.

## Results

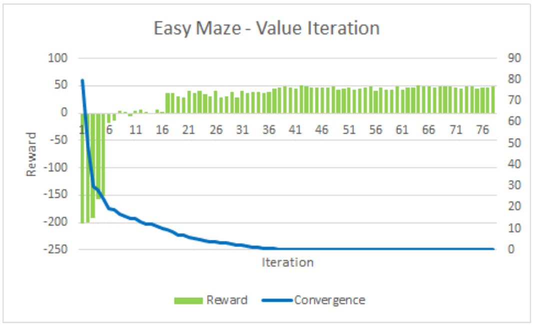
In order to have an idea about the performance of the Q-learning algorithm, the results of it are compared to that from value iteration (which is considered as the true answer as the transition probability and reward function are known in this case).

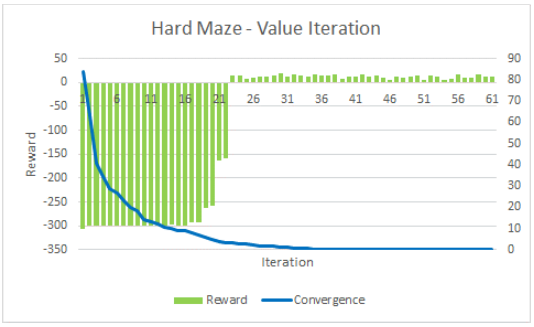
The difference between them can be considered as the estimate error of the Q-learning algorithm.

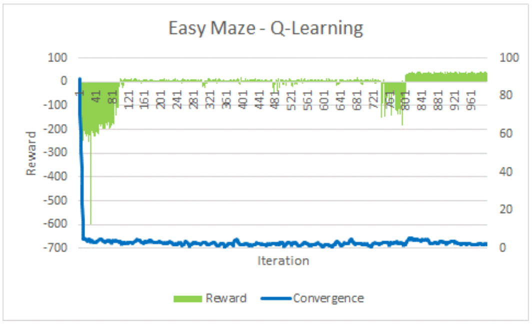
| **Easy Maze** | Q-Learning | Value Iteration |
|---|---|---|
| (α, ε) | (0.1, 0.1) | N/A |
| Utility | 40.72 | 48.54 |
| Step | 25.9 | 48.06 |
| Convergence | 2.04 | 0.00 |
| **Hard Maze** | Q-Learning | Value Iteration |
| (α, ε) | (0.2, 0.4) | N/A |
| Utility | 14.02 | 11.45 |
| Step | 64.1 | 63.58 |
| Convergence | 6.41 | 0.00 |

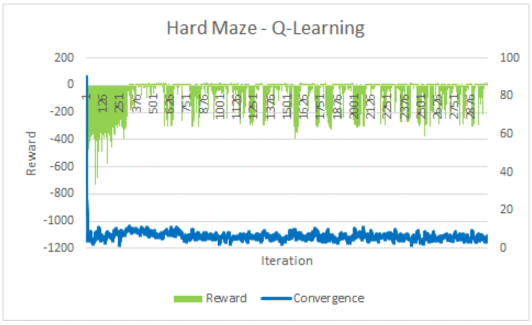
Easy Maze

Hard Maze







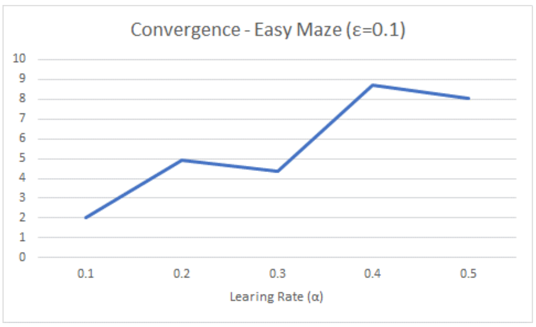


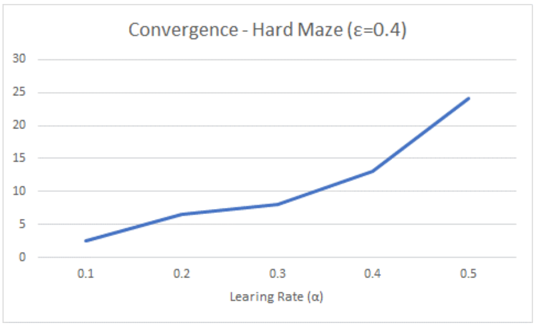
It is clear that the results, including rewards and number of steps, which mean also the optimal policy chosen from Q-learning is not exactly the same as the true value – just like all statistical method / machine learning algorithm.

The convergence of Q-learning are also worse that that of the true answer due to the stochastic nature of the algorithm.

## Effect of parameters on the convergence

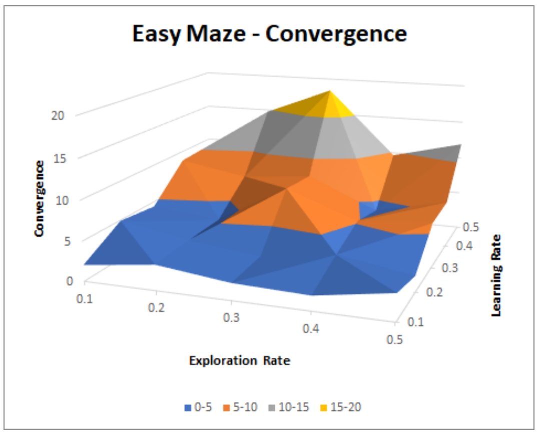
An increase trend of the convergence index, i.e. worse convergence, is observed when the learning rate is becoming higher.

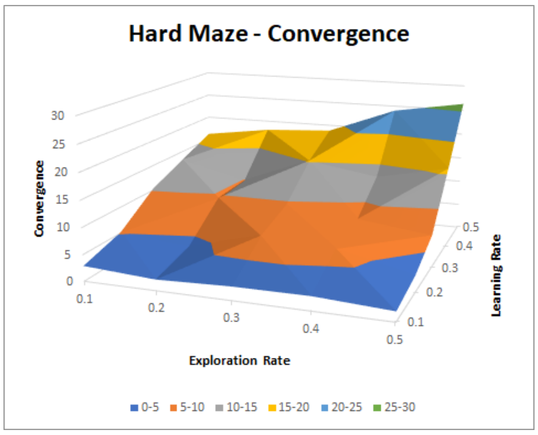




This is likely because of the instability introduced by giving too much weight to new observation, and this makes the results oscillating too much.

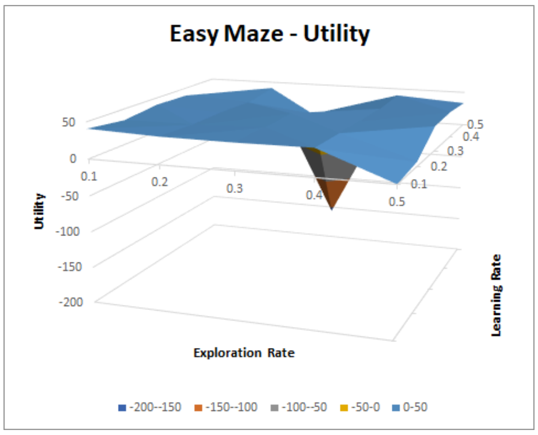
However, no trend of the convergence is found on the exploration side.

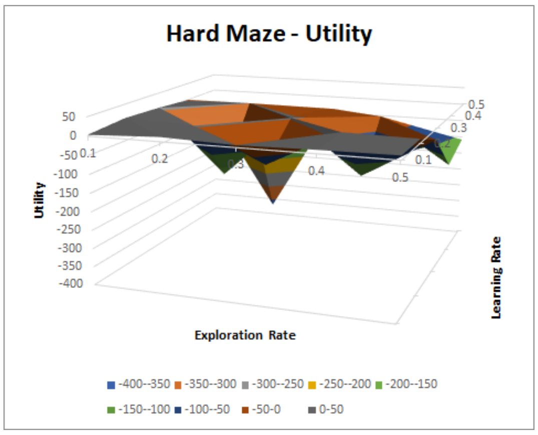




## Effect of parameters on the utility

From the graph, it is trivial that the easy maze (with less states) is really a easier one to solve. The utility obtained are way less affected by the choice of the parameters when compared with that of the hard maze (with more states).





In additional, it is also found in the hard maze that as the exploration rate increase, the utility become worse, but also smoother in general. This is the consequence of spending more time on exploration instead of exploitation.

Changing the initial utility

By changing the initial utility, the performance of the algorithm may change. For example, if the initial utility is now -10,000 instead of 0, the final reward, the intermediate penalty will become too small make the algorithm to be sensitive to that, and this will affect the ability of finding the optimal policy.

Initial utility of -100, and 100 are tested in additional, and the performance are poor:

| Initial utility | -100 | 0 | 100 |
|---|---|---|---|
| Performance | All Crashed | Base Case | Unstable Results |

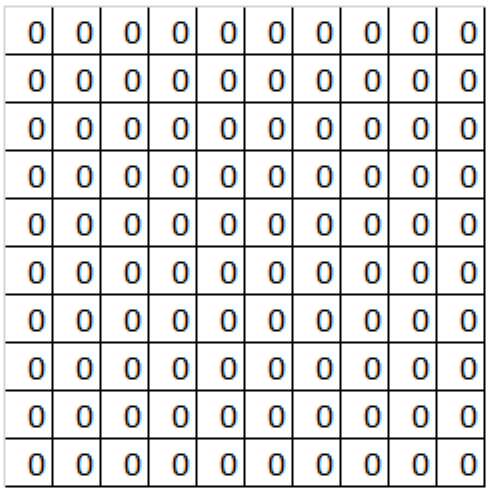
This shows that it is important to set a reasonable initial values in order to get a good results, and to avoid instability and local optimal of the Q-learning algorithm.

**Performance at different size (number of states)**

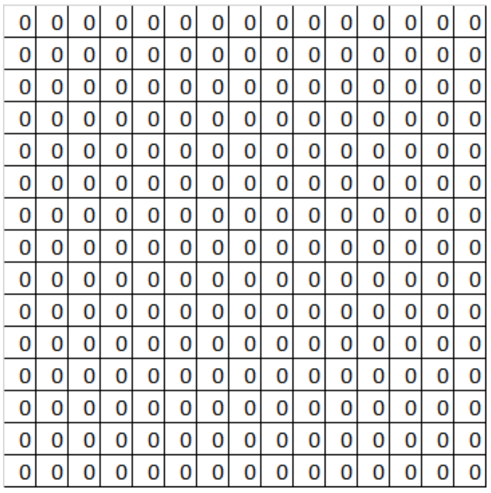
One of the important concerns of MDP and reinforcement learning is about the number of states. In these two examples, the number of states is directly (and only, as number of states = size$^2$) related to the size of the maze. Performance of different algorithm under different number of states are to be discussed.

In this session, the maze will be assumed to be a matrix with all zero, i.e. all states are assessible, and there are no reward or penalty within between. The rewards are sparse as the only reward is at the terminal point, which is a challenge for the algorithms are all the rewards are deferred one.

Sample Maze:



| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |



| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## Time Spent

It is found that no matter how many states are there, value iteration takes more iteration than policy iteration to finish the task. It is also found that vale iteration is slower than policy iteration when there is just a few number of states, and vice versa when there are more states. As expected, Q-learning is always the slowest.

| Size | Iteration | | |
|---|---|---|---|
| | Value Iteration | Policy Iteration | Q-Learning |
| 2 | 15 | 2 | 104 |
| 3 | 18 | 2 | 232 |
| 4 | 22 | 3 | 415 |
| 5 | 26 | 3 | 676 |
| 7 | 30 | 3 | 1,009 |
| 10 | 35 | 4 | 1,416 |
| 15 | 43 | 6 | 2,516 |
| 20 | 51 | 7 | 3,465 |
| 25 | 65 | 8 | 5,058 |
| 30 | 69 | 8 | 4,107 |
| 35 | 78 | 10 | 5,074 |
| 40 | 90 | 11 | 4,839 |
| 50 | 107 | 13 | 10 |

| Size | Time (second) | | |
|---|---|---|---|
| | Value Iteration | Policy Iteration | Q-Learning |
| 2 | 0.18 | 0.05 | 6.19 |
| 3 | 0.17 | 0.13 | 8.90 |
| 4 | 0.34 | 0.25 | 12.24 |
| 5 | 0.31 | 0.26 | 15.26 |
| 7 | 0.41 | 0.38 | 44.07 |
| 10 | 0.88 | 0.70 | 35.75 |
| 15 | 1.33 | 1.19 | 84.72 |
| 20 | 1.87 | 2.18 | 157.94 |
| 25 | 3.79 | 3.95 | 170.21 |
| 30 | 6.00 | 5.80 | 209.38 |
| 35 | 8.84 | 8.43 | 296.28 |
| 40 | 12.45 | 15.27 | 353.98 |
| 50 | 23.96 | 26.54 | 21.00 |

## Results

The value iteration and policy always return really similar results (utility and number of steps), while Q-learning returns similar results as the other two when the size is small only. When the size become bigger and bigger, the Q-learning algorithm are performing worse and worse, and finally doesn't work at all when the size is as large as 50.

| Size | Utility | | |
|---|---|---|---|
| | Value Iteration | Policy Iteration | Q-Learning |
| 2 | 98.5 | 98.3 | 98.5 |
| 3 | 95.9 | 96.0 | 95.5 |
| 4 | 93.4 | 93.1 | 92.8 |
| 5 | 90.7 | 90.7 | 90.1 |
| 7 | 84.6 | 85.5 | 83.6 |
| 10 | 77.4 | 76.8 | 75.9 |
| 15 | 64.8 | 64.1 | 64.1 |
| 20 | 51.1 | 50.1 | 49.0 |
| 25 | 37.0 | 37.1 | 35.7 |
| 30 | 24.4 | 25.2 | 21.3 |
| 35 | 11.4 | 10.2 | 6.2 |
| 40 | -3.2 | -3.0 | -7.6 |
| 50 | -29.0 | -29.7 | -3,995.7 |

| Size | Number of Steps | | |
|---|---|---|---|
| | Value Iteration | Policy Iteration | Q-Learning |
| 2 | 3.5 | 3.7 | 3.5 |
| 3 | 6.1 | 6.0 | 6.5 |
| 4 | 8.6 | 8.9 | 9.2 |
| 5 | 11.3 | 11.3 | 11.9 |
| 7 | 17.4 | 16.5 | 18.4 |
| 10 | 24.6 | 25.2 | 26.1 |
| 15 | 37.3 | 37.9 | 37.9 |
| 20 | 51.0 | 51.9 | 53.0 |
| 25 | 65.0 | 65.0 | 66.3 |
| 30 | 77.6 | 76.8 | 80.7 |
| 35 | 90.7 | 91.8 | 95.8 |
| 40 | 105.2 | 105.0 | 109.6 |
| 50 | 131.0 | 131.7 | 3,998.7 |

## Appendix – Results

### Easy Maze under different parameter setting:

| Iteration | | Elipson | | | | |
|---|---|---|---|---|---|---|
| Initial Q = | 0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
| Learning Rate | 0.1 | 1000 | 1000 | 1000 | 1000 | 1000 |
| | 0.2 | 1000 | 1000 | 1000 | 1000 | 1000 |
| | 0.3 | 1000 | 1000 | 1000 | 1000 | 1000 |
| | 0.4 | 1000 | 1000 | 1000 | 1000 | 1000 |
| | 0.5 | 1000 | 1000 | 1000 | 1000 | 1000 |

| Iteration | | Elipson | | | | |
|---|---|---|---|---|---|---|
| Initial Q = | 100 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
| Learning Rate | 0.1 | 150 | 327 | 142 | 273 | 256 |
| | 0.2 | 1000 | 1000 | 1000 | 1000 | 1000 |
| | 0.3 | 1000 | 1000 | 1000 | 1000 | 1000 |
| | 0.4 | 1000 | 1000 | 1000 | 1000 | 1000 |
| | 0.5 | 1000 | 1000 | 1000 | 1000 | 1000 |

| Iteration | | Elipson | | | | |
|---|---|---|---|---|---|---|
| Initial Q = | -100 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
| Learning Rate | 0.1 | 25 | 23 | 16 | 31 | 16 |
| | 0.2 | 84 | 18 | 46 | 48 | 37 |
| | 0.3 | 130 | 118 | 130 | 29 | 12 |
| | 0.4 | 60 | 382 | 69 | 80 | 134 |
| | 0.5 | 58 | 92 | 26 | 1000 | 72 |

| Time | | Elipson | | | | |
|---|---|---|---|---|---|---|
| Initial Q = | 0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
| Learning Rate | 0.1 | 0.248 | 0.243 | 0.235 | 0.253 | 0.237 |
| | 0.2 | 0.247 | 0.219 | 0.223 | 0.297 | 0.237 |
| | 0.3 | 0.39 | 0.334 | 0.476 | 0.317 | 0.231 |
| | 0.4 | 0.242 | 0.25 | 0.246 | 0.227 | 0.292 |
| | 0.5 | 0.299 | 0.265 | 0.257 | 0.236 | 0.25 |

| Time | | Elipson | | | | |
|---|---|---|---|---|---|---|
| Initial Q = | 100 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
| Learning Rate | 0.1 | 0.078 | 0.184 | 0.102 | 0.156 | 0.157 |
| | 0.2 | 0.336 | 0.461 | 0.447 | 0.339 | 0.341 |
| | 0.3 | 0.414 | 0.457 | 0.375 | 0.357 | 0.433 |
| | 0.4 | 0.403 | 0.427 | 0.361 | 0.323 | 0.344 |
| | 0.5 | 0.285 | 0.376 | 0.398 | 0.363 | 0.394 |

| Time | | Elipson | | | | |
|---|---|---|---|---|---|---|
| Initial Q = | -100 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
| Learning Rate | 0.1 | 0.132 | 0.049 | 0.08 | 0.076 | 0.042 |
| | 0.2 | 0.151 | 0.034 | 0.076 | 0.081 | 0.037 |
| | 0.3 | 0.175 | 0.165 | 0.213 | 0.06 | 0.028 |
| | 0.4 | 0.102 | 0.68 | 0.101 | 0.112 | 0.19 |
| | 0.5 | 0.092 | 0.135 | 0.048 | 0.338 | 0.079 |

| Utility | | Elipson | | | | |
|---|---|---|---|---|---|---|
| Initial Q = | 0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
| Learning Rate | 0.1 | 40.72 | 37.08 | 36.48 | 37.28 | 0.16 |
| | 0.2 | 34.2 | 34.52 | 37.44 | 36.4 | 6.18 |
| | 0.3 | 39.9 | 35.04 | 39.44 | 35 | 34.14 |
| | 0.4 | 38.18 | 32.98 | 21.32 | 35.52 | 36.64 |
| | 0.5 | -18.26 | 40.54 | -156.3 | 39.82 | 33.4 |

| Utility | | Elipson | | | | |
|---|---|---|---|---|---|---|
| Initial Q = | 100 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
| Learning Rate | 0.1 | -38.24 | 45.7 | -93.36 | 41.74 | 38.88 |
| | 0.2 | 44.1 | -145.1 | 40.78 | 35.44 | 50.08 |
| | 0.3 | 39.74 | 47.48 | 40.1 | 37.96 | -102.6 |
| | 0.4 | -208.6 | 38.28 | 36.22 | 49.22 | -74.18 |
| | 0.5 | -20.7 | 20.06 | 43.86 | 41.48 | 32.5 |

| Utility | | Elipson | | | | |
|---|---|---|---|---|---|---|
| Initial Q = | -100 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
| Learning Rate | 0.1 | -194 | -201.9 | -203.1 | -201.9 | -201.4 |
| | 0.2 | -186.5 | -202.5 | -201.9 | -201.9 | -205.4 |
| | 0.3 | -201.3 | -201.9 | -201.3 | -192.5 | -200.3 |
| | 0.4 | -203.1 | -199.6 | -200.2 | -207.1 | -201.3 |
| | 0.5 | -203.6 | -199.6 | -201.9 | 33.2 | -201.9 |

| Steps | | Elipson | | | | |
|---|---|---|---|---|---|---|
| Initial Q = | 0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
| Learning Rate | 0.1 | 25.9 | 24.9 | 25.5 | 25.86 | 25.1 |
| | 0.2 | 26.62 | 26.88 | 25.7 | 26.16 | 25 |
| | 0.3 | 24.98 | 25.78 | 24.86 | 25.82 | 26.68 |
| | 0.4 | 24.96 | 25.52 | 42.4 | 26.46 | 25.92 |
| | 0.5 | 53.56 | 24.92 | 170.7 | 26.22 | 26.26 |

| Steps | | Elipson | | | | |
|---|---|---|---|---|---|---|
| Initial Q = | 100 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
| Learning Rate | 0.1 | 125.6 | 47.46 | 150.1 | 51.8 | 50.84 |
| | 0.2 | 48.08 | 168 | 25.26 | 62.4 | 48.48 |
| | 0.3 | 26.88 | 48.54 | 27.68 | 24.6 | 159.7 |
| | 0.4 | 200 | 26.02 | 27.5 | 48.94 | 130.1 |
| | 0.5 | 43.24 | 42.5 | 46.36 | 26.88 | 28.32 |

| Steps | | Elipson | | | | |
|---|---|---|---|---|---|---|
| Initial Q = | -100 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
| Learning Rate | 0.1 | 197.9 | 200 | 200 | 200 | 196.9 |
| | 0.2 | 190.7 | 200 | 200 | 200 | 200 |
| | 0.3 | 200 | 200 | 200 | 196.4 | 199.9 |
| | 0.4 | 200 | 200 | 200 | 200 | 200 |
| | 0.5 | 200 | 200 | 200 | 25.88 | 200 |

| Convergence | | Elipson | | | | |
|---|---|---|---|---|---|---|
| Initial Q = | 0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
| Learning Rate | 0.1 | 2.043 | 3.176 | 2.108 | 1.813 | 3.308 |
| | 0.2 | 4.913 | 3.408 | 2.375 | 3.589 | 2.098 |
| | 0.3 | 4.355 | 3.436 | 8.515 | 5.14 | 5.82 |
| | 0.4 | 8.704 | 7.153 | 7.963 | 4.004 | 6.024 |
| | 0.5 | 8.047 | 14.75 | 18.38 | 9.649 | 11.91 |

| Convergence | | Elipson | | | | |
|---|---|---|---|---|---|---|
| Initial Q = | 100 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
| Learning Rate | 0.1 | 0.447 | 0.475 | 0.479 | 0.454 | 0.465 |
| | 0.2 | 2.073 | 2.205 | 2.866 | 1.888 | 2.449 |
| | 0.3 | 4.305 | 6.15 | 5.901 | 8.002 | 5.493 |
| | 0.4 | 13.28 | 8.345 | 4.741 | 5.278 | 13.17 |
| | 0.5 | 8.348 | 4.901 | 8.27 | 9.491 | 9.283 |

| Convergence | | Elipson | | | | |
|---|---|---|---|---|---|---|
| Initial Q = | -100 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
| Learning Rate | 0.1 | 0.397 | 0.495 | 0.349 | 0.33 | 0.366 |
| | 0.2 | 0.417 | 0.494 | 0.361 | 0.336 | 0.3 |
| | 0.3 | 0.348 | 0.454 | 0.309 | 0.292 | 0.124 |
| | 0.4 | 0.271 | 0.474 | 0.307 | 0.155 | 0.287 |
| | 0.5 | 0.497 | 0.319 | 0.157 | 7.933 | 0.173 |

### Hard Maze under different parameter setting:

| Iteration | | Elipson | | | | |
|---|---|---|---|---|---|---|
| Initial Q = | 0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
| Learning Rate | 0.1 | 3000 | 3000 | 3000 | 3000 | 3000 |
| | 0.2 | 3000 | 3000 | 3000 | 3000 | 3000 |
| | 0.3 | 3000 | 3000 | 3000 | 3000 | 3000 |
| | 0.4 | 3000 | 3000 | 3000 | 3000 | 3000 |
| | 0.5 | 3000 | 3000 | 3000 | 3000 | 3000 |

| Iteration | | Elipson | | | | |
|---|---|---|---|---|---|---|
| Initial Q = | 100 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
| Learning Rate | 0.1 | 3000 | 3000 | 3000 | 3000 | 3000 |
| | 0.2 | 3000 | 3000 | 3000 | 3000 | 3000 |
| | 0.3 | 3000 | 3000 | 3000 | 3000 | 3000 |
| | 0.4 | 3000 | 3000 | 3000 | 3000 | 3000 |
| | 0.5 | 3000 | 3000 | 3000 | 3000 | 3000 |

| Iteration | | Elipson | | | | |
|---|---|---|---|---|---|---|
| Initial Q = | -100 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
| Learning Rate | 0.1 | 17 | 20 | 15 | 17 | 13 |
| | 0.2 | 19 | 17 | 15 | 18 | 16 |
| | 0.3 | 17 | 15 | 14 | 15 | 19 |
| | 0.4 | 16 | 19 | 15 | 15 | 18 |
| | 0.5 | 16 | 15 | 17 | 14 | 18 |

| Time | | Elipson | | | | |
|---|---|---|---|---|---|---|
| Initial Q = | 0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
| Learning Rate | 0.1 | 1.725 | 1.514 | 1.8 | 1.798 | 1.47 |
| | 0.2 | 1.565 | 1.809 | 1.75 | 1.983 | 1.322 |
| | 0.3 | 1.356 | 1.818 | 1.732 | 1.762 | 1.597 |
| | 0.4 | 1.418 | 1.706 | 1.504 | 1.627 | 1.578 |
| | 0.5 | 1.733 | 1.709 | 1.633 | 1.614 | 1.53 |

| Time | | Elipson | | | | |
|---|---|---|---|---|---|---|
| Initial Q = | 100 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
| Learning Rate | 0.1 | 1.566 | 1.503 | 1.597 | 1.234 | 1.21 |
| | 0.2 | 1.04 | 1.091 | 1.075 | 1.084 | 1.124 |
| | 0.3 | 1.525 | 1.607 | 1.242 | 1.424 | 1.387 |
| | 0.4 | 1.395 | 1.398 | 1.493 | 1.133 | 1.128 |
| | 0.5 | 1.236 | 1.217 | 1.256 | 1.235 | 1.24 |

| Time | | Elipson | | | | |
|---|---|---|---|---|---|---|
| Initial Q = | -100 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
| Learning Rate | 0.1 | 0.081 | 0.025 | 0.03 | 0.021 | 0.015 |
| | 0.2 | 0.042 | 0.024 | 0.025 | 0.033 | 0.016 |
| | 0.3 | 0.03 | 0.018 | 0.016 | 0.02 | 0.019 |
| | 0.4 | 0.024 | 0.019 | 0.015 | 0.016 | 0.02 |
| | 0.5 | 0.017 | 0.016 | 0.02 | 0.017 | 0.019 |

| Utility | | Elipson | | | | |
|---|---|---|---|---|---|---|
| Initial Q = | 0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
| Learning Rate | 0.1 | 3.67 | 12.02 | 5.8 | 9.54 | 11.25 |
| | 0.2 | 12.37 | 3.77 | -86.91 | 14.02 | 10.61 |
| | 0.3 | 7.38 | -174.3 | 3.16 | -146 | -30.99 |
| | 0.4 | -0.91 | 0.33 | -13.4 | -14.84 | -143.1 |
| | 0.5 | -70.46 | -369.4 | -38.17 | -167.2 | -108.5 |

| Utility | | Elipson | | | | |
|---|---|---|---|---|---|---|
| Initial Q = | 100 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
| Learning Rate | 0.1 | 12.56 | -143.7 | -0.15 | -8.36 | 6.84 |
| | 0.2 | -30.12 | -167.6 | 11.98 | -45.36 | 12.39 |
| | 0.3 | -261.2 | -21.63 | -205.5 | -60.38 | -61.65 |
| | 0.4 | -0.24 | -72.47 | -71.07 | 4.4 | -11.68 |
| | 0.5 | -97.25 | -13.22 | -285.1 | -74.03 | -59.55 |

| Utility | | Elipson | | | | |
|---|---|---|---|---|---|---|
| Initial Q = | -100 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
| Learning Rate | 0.1 | -299 | -299 | -299 | -299 | -299 |
| | 0.2 | -299 | -299 | -299 | -299 | -299 |
| | 0.3 | -299 | -299 | -299 | -299 | -299 |
| | 0.4 | -299 | -299 | -299 | -299 | -299 |
| | 0.5 | -299 | -299 | -299 | -299 | -299 |

| Steps | | Elipson | | | | |
|---|---|---|---|---|---|---|
| Initial Q = | 0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
| Learning Rate | 0.1 | 64.54 | 65.05 | 67.89 | 64.66 | 64.96 |
| | 0.2 | 65.9 | 65.8 | 153.2 | 64.1 | 63.46 |
| | 0.3 | 65.47 | 190.2 | 70.26 | 188.3 | 107.3 |
| | 0.4 | 57.35 | 68.5 | 66.78 | 82.92 | 156.9 |
| | 0.5 | 106.3 | 228.3 | 87.37 | 199.3 | 171.9 |

| Steps | | Elipson | | | | |
|---|---|---|---|---|---|---|
| Initial Q = | 100 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
| Learning Rate | 0.1 | 63.95 | 183.5 | 68.41 | 69.82 | 61.94 |
| | 0.2 | 101.4 | 205.9 | 65.02 | 75.26 | 68.53 |
| | 0.3 | 191.1 | 73.48 | 230.9 | 105.9 | 134.2 |
| | 0.4 | 74.54 | 117.1 | 115.8 | 70.5 | 75.25 |
| | 0.5 | 150.3 | 68.81 | 279 | 101.9 | 97.57 |

| Steps | | Elipson | | | | |
|---|---|---|---|---|---|---|
| Initial Q = | -100 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
| Learning Rate | 0.1 | 300 | 300 | 300 | 300 | 300 |
| | 0.2 | 300 | 300 | 300 | 300 | 300 |
| | 0.3 | 300 | 300 | 300 | 300 | 300 |
| | 0.4 | 300 | 300 | 300 | 300 | 300 |
| | 0.5 | 300 | 300 | 300 | 300 | 300 |

| Convergence | | Elipson | | | | |
|---|---|---|---|---|---|---|
| Initial Q = | 0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
| Learning Rate | 0.1 | 2.892 | 2.054 | 2.54 | 2.618 | 1.854 |
| | 0.2 | 5.174 | 3.897 | 7.431 | 6.483 | 3.215 |
| | 0.3 | 10.33 | 9.651 | 9.678 | 8.131 | 6.424 |
| | 0.4 | 13.28 | 9.925 | 15.23 | 13.07 | 15.89 |
| | 0.5 | 16.19 | 18.07 | 18.84 | 24.05 | 26.28 |

| Convergence | | Elipson | | | | |
|---|---|---|---|---|---|---|
| Initial Q = | 100 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
| Learning Rate | 0.1 | 2.238 | 3.374 | 1.899 | 2.354 | 1.618 |
| | 0.2 | 6.988 | 5.893 | 5.674 | 8.467 | 4.557 |
| | 0.3 | 12.23 | 9.582 | 7.527 | 11.92 | 5.73 |
| | 0.4 | 10.63 | 15.55 | 11.76 | 10.9 | 10.63 |
| | 0.5 | 20.83 | 18.23 | 20.58 | 26.02 | 17.97 |

| Convergence | | Elipson | | | | |
|---|---|---|---|---|---|---|
| Initial Q = | -100 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
| Learning Rate | 0.1 | 0.483 | 0.002 | 0.495 | 0.49 | 0.391 |
| | 0.2 | 0.003 | 3E-04 | 0.133 | 0.079 | 6E-04 |
| | 0.3 | 0.038 | 0.41 | 0.436 | 0.003 | 0.137 |
| | 0.4 | 0.004 | 0.314 | 0.002 | 0.077 | 0.006 |
| | 0.5 | 0.029 | 0.307 | 0.313 | 0.002 | 0.068 |