

---

# Value Iteration and Q-learning in Grid World - project report

---

Jayaram Kuchibhotla  
Department of Computer Science  
University at Buffalo  
[jayaramk@buffalo.edu](mailto:jayaramk@buffalo.edu)

## Abstract

This document summarizes the results of two problems 1.Value Iteration and 2.Q learning which are part of the project 'Reinforcement Learning' in the course 'CS188 Introduction to AI' offered by UC Berkeley in the year 2014.

## 1 Introduction

Before providing the above-mentioned summary of results, the basics of path planning are briefly described as follows:

Path planning is about finding the path from a given point to destination, preferably in an optimal manner, by an agent such as a robot in the simulated environment or real world. The agent can do the planning by 'Search', which consists of

- a. State Space: It is collection of states i.e. locations. A path is formed by few states, which have to be determined from this collection.
- b. Successor Function: This helps to know about the possible state transitions and costs associated with such transitions.
- c. Start state and Goal state for an agent. The solution to this search problem is a sequence of actions, which can lead an agent from Start state to Goal state.

Search trees are used to represent the search problem where the nodes are states. The child nodes represent the future possible states while the current node represents the current state. The path is determined using these search trees and the techniques. If the search technique is able to find the path from start state to goal state, then it is called complete. It is called optimal only if it is able to provide the best path among all the available paths

1. Deterministic Search: The actions for state transition are certain and do not have probability values associated with them. In this category, there are two sub categories:

a. Uninformed: Search happens in all the directions. The information of the goal is not present

- 1. Depth First Search: It finds the deepest node first
- 2. Breadth First Search: It finds the shallowest node first
- 3. Uniform Cost Search: It finds cheapest node (in terms of cost of the link to the node) first

b. Informed: Search happens in all the directions but directed towards the goal as its location information is known by the algorithm. Heuristics, a function is used in this search to estimate how close the goal is. E.g. Manhattan distance, Euclidean distance for pathing.

- 1. Greedy Search: It finds the node, which could be closest to a goal state
- 2. A-star search: This is a combination of Uniform cost search and Greedy search

2. Non Deterministic Search: The actions for state transition are uncertain. They are specified using the probability values. There are 2 categories of problems in this area

a. Markov Decision Processes(MDP): These are used for modelling decision making in an uncertain environment. The processes consist of states, state transition actions, state transition probability values ,utility values(the value which indicates how much the state could be part of the decision)for each state are incremented (positive reward) if the final state is goal state. The utility values are decremented (negative reward) if the final state is undesired. Also, in MDP, the transition to future state is dependent only on the current state and current action.

There could be multiple decisions, which can lead from start state to goal state. Among them, the decision, which has maximum average utility value (the average of the utility values for each state), is the optimal decision which is also called as ‘Policy’.

The grid world problem is modelled using MDP to determine the optimal path between start and goal state.

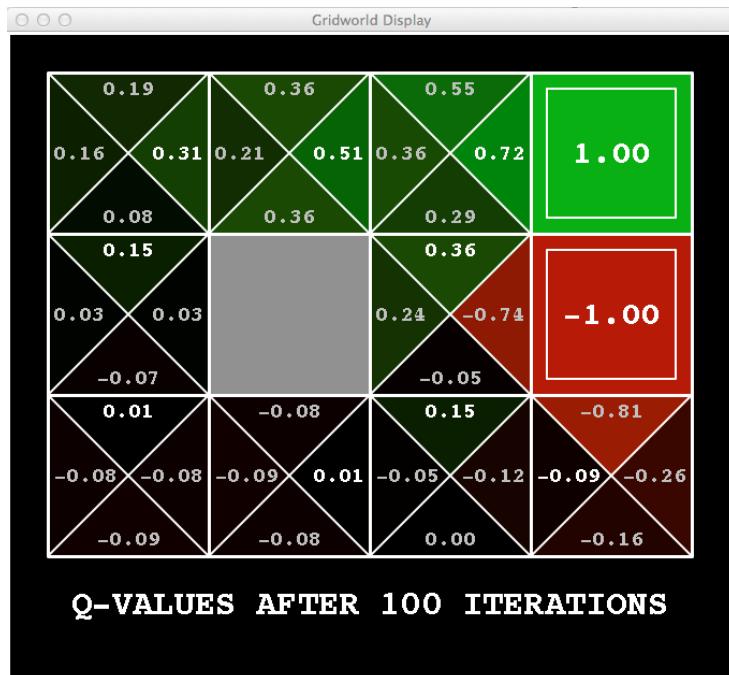
Following is an example of grid world. It has

- 12 states
- 4 possible actions in a state (North, South, West, East) and the arrow in each square indicates optimal action (one out of four values)
- Double bordered green square which is the goal state and
- Double bordered red square which is the undesired terminal state
- The numerical values inside each square is the utility value
- Start state, which is the bottom most and left most square

The blue colored circle is the agent, which moves from one square to another square in the grid world. The agent is visible while running the program. It is not shown in the below snapshot



In the below snapshot, each square has four Q-values, one for each action. The utility value is the maximum Q-value among all Q-values in a square. Q-value means quadrant value. In the below snapshot ,it can be seen that there are 4 quadrants ,one for each action , in a single square



The MDP is solved using the 'Expectimax' search tree. In this tree, the computation is done bottom up.

In the bottom most level 'V0', there are default values. In the level E1, which is on top of V0, the expected values are calculated using values from V0. In level V1, which is on top of E1, the maximum among the expected values from E1 are determined. This calculation proceeds up to the level Vn, which is the top most level in the tree and n is the number of iterations

When this tree is used for solving MDP, it is called 'Value Iteration'.

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V_k(s')]$$

This equation has two computations, one is the calculation of the expected value/Q-value and the other is max operation. In Value Iteration, the default values in all the states are zeroes

$V_{k+1}(s)$  is the utility value of the state 's' in the current iteration.

$T(s,a,s')$  is the probability value for transiting to state  $s'$  given that the current states is  $s$  and current action is 'a'  
 $R(s,a,s')$  is the reward value for transiting to state  $s'$  given that the current states is  $s$  and current action is 'a'  
The above two values are present beforehand in a table obtained using the prior experiences in the grid world

' $\gamma$ ' is the discount factor which could be in the range (0,1).

$V_k(s')$  is the utility value of the state  $s'$  in the previous iteration

Discount factor is used to weigh the utility value  $V_k(s')$ . By using this discount factor, the utility values in earlier iterations are given more importance compared to the utility values in earlier iterations. This is done to obtain the convergence – the stopping condition of value iteration where the utility values do not change significantly

There are 'k+1' iterations before the convergence.

#### b. Reinforcement learning:

Unlike MDP, in this methodology, there is no prior knowledge of transition probability values and reward function values. These values are computed from the training episodes of agent in 2 ways

##### 1. Model based learning

In this approach, the agent explores actions and states during a few training episodes. Based on these state utility values, the transition probability value say from state  $s$  to state  $s'$  is calculated as  $T(s,a,s') = \text{number of transitions from } s \text{ to } s' / \text{number of transitions from state } s \text{ to any other state}$ . The reward value is  $R(s,a,s') = \text{most common utility value among all the episodes for the transition from } s \text{ to } s'$

After learning these values, value iteration is used to compute the utility value and best action in each state during further iterations.

2. Model free learning: It does not estimate the  $T(s,a,s')$  and  $R(s,a,s')$  values to calculate the utility values. It is further divided into two categories

a. Passive Reinforcement Learning: In this category, a fixed policy (map of utility values and best action for each state) is given as input and the state values are further learnt by acting in the world. Every time, a state is visited a utility value is calculated, a few more samples are calculated one for each visit to this state. All these sample values are averaged to get a utility value for the corresponding state. The actions are taken as per the input policy alone and there is no chance for further exploration

b. Active Reinforcement Learning: There is no policy or any other input. All Q-values are zeros in this technique, which means the utility values are also zeroes. The Q-values are computed by a technique called Q-learning which is given by the below equations.

Equation 1:

$$sample = R(s, a, s') + \gamma \max_{a'} Q(s', a')$$

' $\gamma$ ' is the discount factor which could be in the range (0,1).

$R(s,a,s')$  is the reward value for transiting to state  $s'$  given that the current states is  $s$  and current action is 'a'

The computation after the discount factor is done to get the utility value, which is the maximum Q-value among all the Q-values calculated for all possible future actions that can result in a future state transition  $s'$ .

Equation 2:

$$Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + (\alpha) [sample]$$

$\alpha$  is the learning rate.

A new Q-value is obtained by using the sample and the old Q-value

The Q-values are learnt after multiple iterations. From them, utility values and the best action for each state are determined which is nothing but the policy representing the best path from start state to goal state

## 2 Results Summary

### Problem 1: Value Iteration

a. Value Iteration for five iterations and one episode

- Input Command: `python gridworld.py -a value -i 5`

- Output:

Grid World snapshot:



Command line Window snapshot

```
C:\WINDOWS\system32\cmd.exe - python gridworld.py -a value -i 5
Started in state: <3, 2>
Took action: exit
Ended in state: TERMINAL_STATE
Got reward: 1
EPISODE 1 COMPLETE: RETURN WAS 0.387420489
AVERAGE RETURNS FROM START STATE: 0.387420489

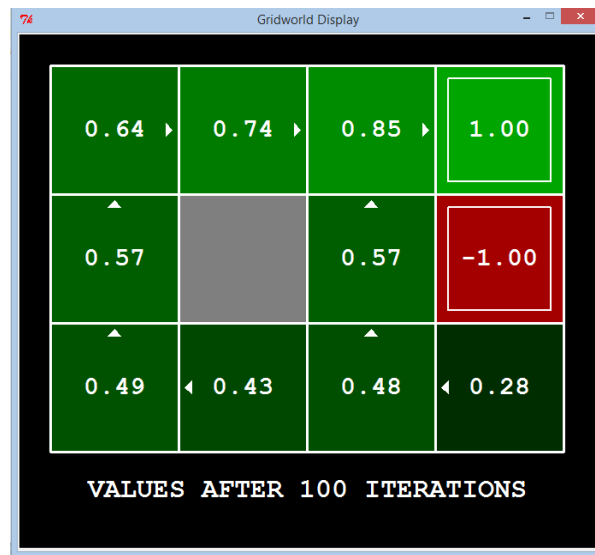
C:\Python27\reinforcement>python gridworld.py -a value -i 5
RUNNING 1 EPISODES
BEGINNING EPISODE: 1
Started in state: <0, 0>
Took action: north
Ended in state: <0, 1>
Got reward: 0.0
Started in state: <0, 1>
Took action: north
Ended in state: <0, 2>
Got reward: 0.0
Started in state: <0, 2>
Took action: east
Ended in state: <1, 2>
Got reward: 0.0
Started in state: <1, 2>
Took action: east
Ended in state: <2, 2>
Got reward: 0.0
Started in state: <2, 2>
Took action: east
Ended in state: <3, 2>
Got reward: 0.0
Started in state: <3, 2>
Took action: exit
Ended in state: TERMINAL_STATE
Got reward: 1
EPISODE 1 COMPLETE: RETURN WAS 0.59049
AVERAGE RETURNS FROM START STATE: 0.59049

C:\Python27\reinforcement>python gridworld.py -a value -i 5
```

b. Value Iteration for 100 iterations and 10 episodes  
- Input Command: `python gridworld.py -a value -i 100 -k 10`

- Output

Grid World snapshot:



Command line Window snapshot

```
C:\WINDOWS\system32\cmd.exe

Started in state: <3, 2>
Took action: exit
Ended in state: TERMINAL_STATE
Got reward: 1

EPISODE 9 COMPLETE: RETURN WAS 0.59049

BEGINNING EPISODE: 10

Started in state: <0, 0>
Took action: north
Ended in state: <0, 1>
Got reward: 0.0

Started in state: <0, 1>
Took action: north
Ended in state: <0, 2>
Got reward: 0.0

Started in state: <0, 2>
Took action: east
Ended in state: <1, 2>
Got reward: 0.0

Started in state: <1, 2>
Took action: east
Ended in state: <2, 2>
Got reward: 0.0

Started in state: <2, 2>
Took action: east
Ended in state: <2, 2>
Got reward: 0.0

Started in state: <2, 2>
Took action: east
Ended in state: <2, 2>
Got reward: 0.0

Started in state: <2, 2>
Took action: east
Ended in state: <3, 2>
Got reward: 0.0

Started in state: <3, 2>
Took action: exit
Ended in state: TERMINAL_STATE
Got reward: 1

EPISODE 10 COMPLETE: RETURN WAS 0.4782969

AVERAGE RETURNS FROM START STATE: 0.510476260609

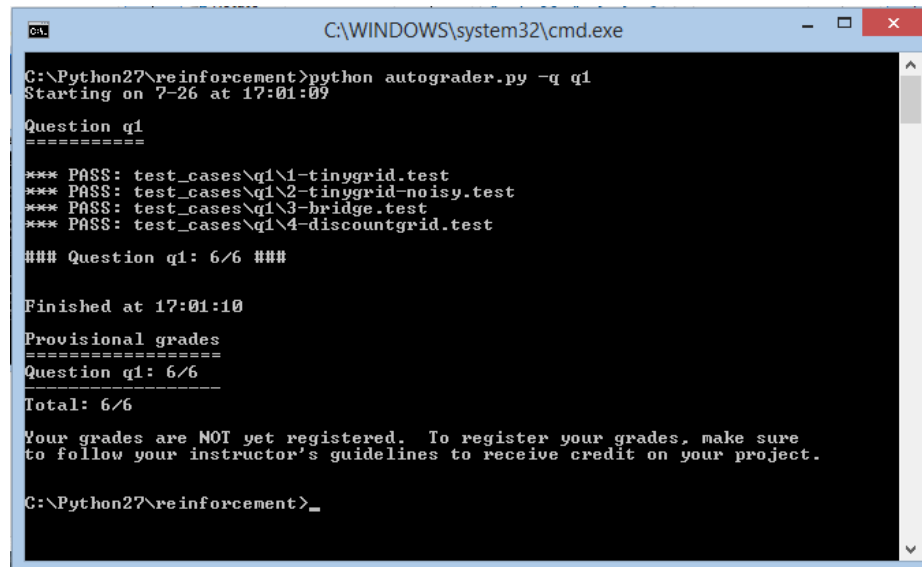
C:\Python27\reinforcement>
```

c. Execution of grading script for value iteration

- Input Command: `python autograder.py -q q1`

- Output

Command Line Window snapshot:



```
C:\WINDOWS\system32\cmd.exe
C:\Python27\reinforcement>python autograder.py -q q1
Starting on 7-26 at 17:01:09

Question q1
=====
*** PASS: test_cases\q1\1-tinygrid.test
*** PASS: test_cases\q1\2-tinygrid-noisy.test
*** PASS: test_cases\q1\3-bridge.test
*** PASS: test_cases\q1\4-discountgrid.test

### Question q1: 6/6 ###

Finished at 17:01:10

Provisional grades
=====
Question q1: 6/6
-----
Total: 6/6

Your grades are NOT yet registered. To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.

C:\Python27\reinforcement>
```

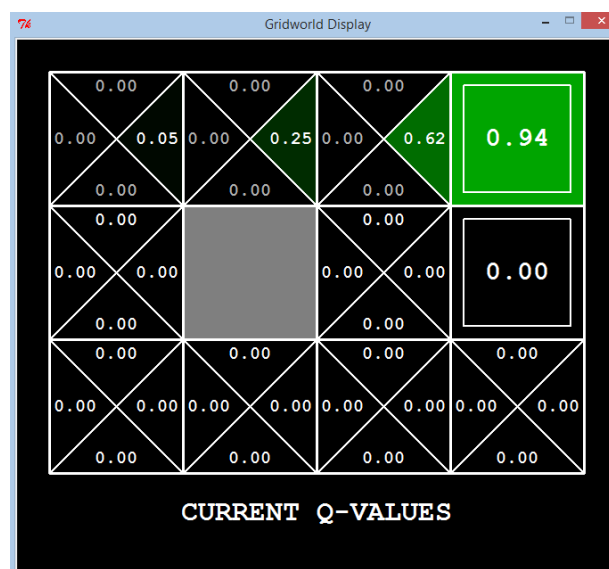
## Problem 2: Q-Learning

a. Q Learning for agent for a maximum of 5 episodes

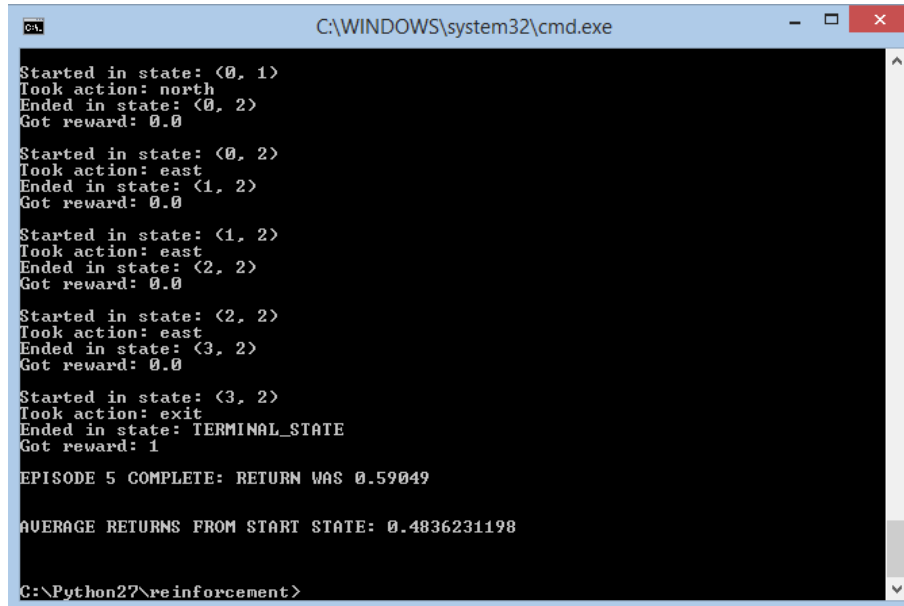
- Input Command: `python gridworld.py -a q -k 5 -m`

- Output:

Grid world Snapshot after 4 episodes of Q-learning



Command Line Window snapshot after completing all episodes of Q-learning



```
C:\WINDOWS\system32\cmd.exe

Started in state: <0, 1>
Took action: north
Ended in state: <0, 2>
Got reward: 0.0

Started in state: <0, 2>
Took action: east
Ended in state: <1, 2>
Got reward: 0.0

Started in state: <1, 2>
Took action: east
Ended in state: <2, 2>
Got reward: 0.0

Started in state: <2, 2>
Took action: east
Ended in state: <3, 2>
Got reward: 0.0

Started in state: <3, 2>
Took action: exit
Ended in state: TERMINAL_STATE
Got reward: 1

EPISODE 5 COMPLETE: RETURN WAS 0.59049

AVERAGE RETURNS FROM START STATE: 0.4836231198

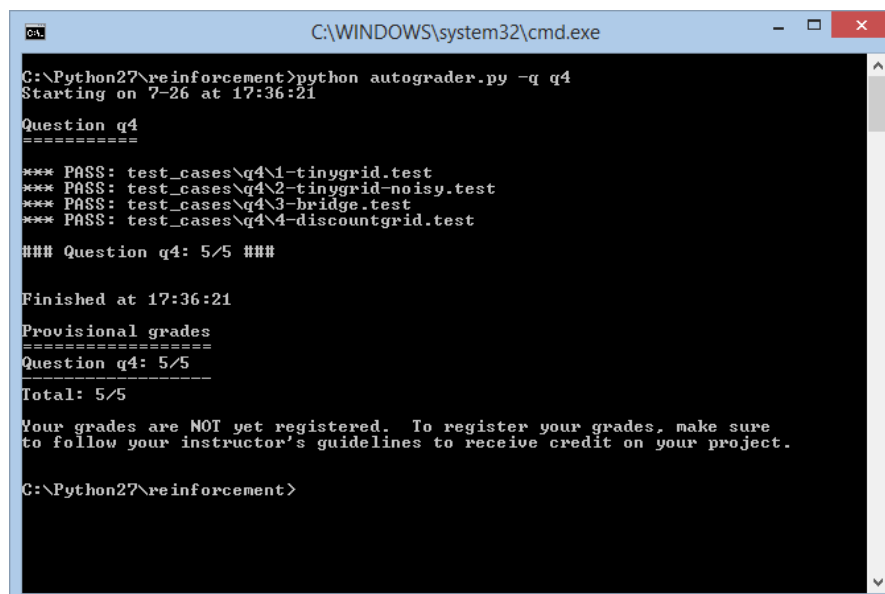
C:\Python27\reinforcement>
```

b. Execution of grading script for Q-learning

- Input Command: python autograder.py -q q4

- Output

Command Line Window snapshot:



```
C:\WINDOWS\system32\cmd.exe

C:\Python27\reinforcement>python autograder.py -q q4
Starting on 7-26 at 17:36:21

Question q4
=====

*** PASS: test_cases\q4\1-tinygrid.test
*** PASS: test_cases\q4\2-tinygrid-noisy.test
*** PASS: test_cases\q4\3-bridge.test
*** PASS: test_cases\q4\4-discountgrid.test

### Question q4: 5/5 ###

Finished at 17:36:21

Provisional grades
=====
Question q4: 5/5
-----
Total: 5/5

Your grades are NOT yet registered. To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.

C:\Python27\reinforcement>
```



### 3 Conclusion

Implementing the solutions for these two problems helped me to learn basics of path planning using Reinforcement Learning

### 4 References

- [1] Reinforcement Learning project problem statements and code for execution environment  
<http://ai.berkeley.edu/reinforcement.html>
- [2] CS 188 course videos upto Lecture 11  
[http://ai.berkeley.edu/lecture\\_videos.html](http://ai.berkeley.edu/lecture_videos.html)
- [3] CS 188 course slides upto Lecture 11  
[http://ai.berkeley.edu/lecture\\_slides.html](http://ai.berkeley.edu/lecture_slides.html)