# Report of Reinforcement Learning Experimentation

## *This repository is forked from https://github.com/rlcode/reinforcement-learning*

## Prerequiesites

### Dependencies

1. Python 3.5
2. Tensorflow 1.0.0
3. Keras
4. numpy
5. pandas
6. matplot
7. pillow
8. Skimage
9. h5py

Before installing these, i suggest to setup a python environment:

python -m venv env

Activate it:

* Windows

env/Scripts/activate

* Linux

source /path/to/ENV/bin/activate

Install all prerequisites:

pip install -r requirements.txt

## How to run

Run python file related to name of model, e.g grid world with policy iteration:

python policy\_iteration.py

# Explanation of running program

## Grid world with policy iteration

1. First Running - set an environment

When we running the program for the first time, we will see interface like above image. Reward 1.0 is given to the blue circle as final destination. Two green triangles is placed near blue circle as penalty with given reward -1.0. The empty grid will be given reward 0 for initial value.

The main program loads environment from environment.py and apply policy iteration on that environment.

1. Policy Evaluation

Click on button Evaluate to update reward for each states. policy\_evaluation function calculate next value based on policy table with Bellman Expectation Equation. Policy are initialized with 0.25 for all actions. This function will update value that are stored in value\_table.

Value calculated with summation of action times reward of every next possible value in all possible actions. Value will be rounded to only x.xx.

1. Policy Improvement

Improve button will update probability movement of rectangle in grid environment. It’s shown with arrows inside of each grid. In every policy\_improvement run, policy will initialized with 0.0. For every actions [reward + (discount factor) \* (next state value function)] will be calculated. Multiple actions will be picked with same max values and probability of action will be stored for next policy.

1. Steps for Converge.

* Click Evaluate to update value each grid
* Click Improve to update probability of rectangle actions.
* When converge or rectangle move to blue circle, value near blue circle is 1. In my observation, this is due to near to finish value will get higher.
* Each grid that lead to goal have value that grater than 0.5. Rectangle will choose grid that maximum.
* But it’s not always the case, if you click Evaluate > Improve > Evaluate x7 > Improve you will get value in each grid like this image:

1. Change Initial Value and Initial Policy Experiment

This is experiment to observe training process when initial value of value function set to random values and initial policy set to normalized random values.

Random value set by numpy random number:

random\_value = np.random.rand()

Random policy set by numpy random dirichlet:

# return array of random distribution that sum is 1  
random\_policy = np.random.dirichlet(np.ones(4), size=1)

With this random values in initial value and initial policy there is slight change in training process. In our optimum way to converge, we get less step for Evaluate process. We achieve 6 step only for evaluate after improvement. We can see the result of our value from training process in above image.

1. Insert Additional Obstacle Experiment

In this experiment, we insert additional obstacle (triangle) in our grid world. New obstacle will be placed in 3.4 grid or state. The image of our new grid world is like below image:

Additional obstacle placed on row 3, column 4 in our grid world environment. This additional obstacle affects to our training process in terms of iteration. For optimal solution of training process, we must running Evaluate and Improve sequentially 5 times. This is due to limitation of possible path of rectangle in this grid world. While it’s possible to have a path going bottom, but the optimum path is with upper path because have a high value.

We can not use our way like grid world before that not running Evaluate and Improve sequentially.

## Grid world with value iteration

1. First Running - set an environment

When we running the program for the first time, we will see interface like above image. Reward 1.0 is given to the blue circle as final destination. Two green triangles is placed near blue circle as penalty with given reward -1.0. The empty grid will be given reward 0 for initial value.

The main program loads environment from environment.py and apply value iteration on that environment.

1. Value Iteration

Action list get from q value of each possible actions in grid world with this function:

value = (reward + self.discount\_factor \* next\_value)

action that has maximum q value will be added to action list through this function:

if value > max\_value:  
 action\_list.clear()  
 action\_list.append(action)  
 max\_value = value  
elif value == max\_value:  
 action\_list.append(action)

1. Training

We manage get optimal solution with six iterations of running Calculate. Two possible path are going straight from starting point either sideway or going bottom. These are due to same value they have in their path. These value are:

1.0 (6th grid)  
1.0 \* 0.9 = 0.9 (5th grid)  
0.9 \* 0.9 = 0.81 (4th grid)  
0.81 \* 0.9 = 0.73 (3rd grid)  
0.73 \* 0.9 = 0.66 (2nd grid)  
0.66 \* 0.9 = 0.59 (1st grid)

When grid of path is near to blue circle, the value are near 1.0. The reason why this happening is the grid is near to reward. Because we use discount factor in this experiment, we get 0.9 as discounted reward.

1. Change Initial Value Experiment

In this experiment we randomize our inital value. We obtain random value using numpy randomize:

self.random\_value = np.random.rand()

With this random value of inital value resulting like below image:

As we can see, we get inital value of 0.68 in this experiment. We use this value as starting point of training process. We manage converge of training with eight iterations. But, when we run the program again, we achieve 4-6 iterations for training. This is posibbly due to have low initial value but not zero. When initial value is above 0.5 we need more iteration of training to converge.

1. Insert Additional Obstacle Experimentation

Additional obstacle are inserted in this experimentation. Obstacle is placed to grid of 3,2 with reward -1. We can observe in above image that obstacle blockage bottom nearest path, so we don’t have any similar path like previous experimentaion.

We achieved total eight iterations to train in this grid world. Optimal path is going straight sideway like this below image.