RL Assignment 1

September 11, 2019

1 Computing State Value Function by solving linear system of equation

We solve the equation Ax = b to obtain the state value function for the grid problem given. This equation will have 4 variables that correspond to the number of possible actions and a total of 25 such variables so the size of A is 25x25 and b is 25 sized-vector. Therefore, we get the output state value function as a 25 dimensional vector.

By solving this linear system of equation we obtain the solution as:

```
[[ 3.3  8.8  4.4  5.3  1.5]
 [ 1.5  3.   2.3  1.9  0.5]
 [ 0.1  0.7  0.7  0.4 -0.4]
 [-1.  -0.4 -0.4 -0.6 -1.2]
 [-1.9 -1.3 -1.2 -1.4 -2. ]]
```

Figure 1: Solution

2 Computing State Action Value Function by solving non-linear system of equation

We solve the solve the equation of the form Ax ¿ b for obtaining the state action value function. For each of the possible actions, we get an update for state action value and we will have to take the maximum for all state value functions for each possible action, so we solve Ax ¿ b. Here, we will have A as 100x25 as the number of states are 25 and actions are 4. b will be 100 sized vector so we get x satisfying the equation as a 25 dimensional vector.

By solving this non-linear system of equation we obtain the solution and the optimal policy as:

```
[[22.  24.4 22.  19.4 17.5]
 [19.8 22.  19.8 17.8 16. ]
 [17.8 19.8 17.8 16.  14.4]
 [16.  17.8 16.  14.4 13. ]
 [14.4 16.  14.4 13.  11.7]]
[['R' 'DURL' 'L' 'DURL' 'L']
 ['UR' 'U' 'UL' 'L' 'L']
 ['UR' 'U' 'UL' 'UL' 'UL']
 ['UR' 'U' 'UL' 'UL' 'UL']
 ['UR' 'U' 'UL' 'UL' 'UL']]
```

Figure 2: Solution

3 Policy Iteration

Final policy obtained after policy iteration for grid example is as follows:

```
Final Policy :
[['-', 'L', 'L', 'DL'], ['U', 'UL', 'DURL', 'D'], ['U', 'DURL', 'DR', 'D'], ['UR', 'R', 'R', '-']]
```

```
State Value Function :
[[ 0. -1. -2. -3.]
 [-1. -2. -3. -2.]
 [-2. -3. -2. -1.]
 [-3. -2. -1.  0.]]
```

Also, as shown in the jupyter notebook, we can clearly see that the difference between new and previous state value function decreases as iterations increase.

4 Value Iteration

Final policy obtained after value iteration for grid example is as follows:

```
Final Policy :
[['-', 'L', 'L', 'DL'], ['U', 'UL', 'DURL', 'D'], ['U', 'DURL', 'DR', 'D'], ['UR', 'R', 'R', '-']]
```

```
State Value Function :
[[ 0. -1. -2. -3.]
 [-1. -2. -3. -2.]
 [-2. -3. -2. -1.]
 [-3. -2. -1.  0.]]
```

Also, as shown in the jupyter notebook, we can clearly see that the difference between new and previous state value function decreases as iterations increase.

5 Jack Car Rental Problem

The plot for optimal policy depicts the action that we must take at each state (21*21=441 states).

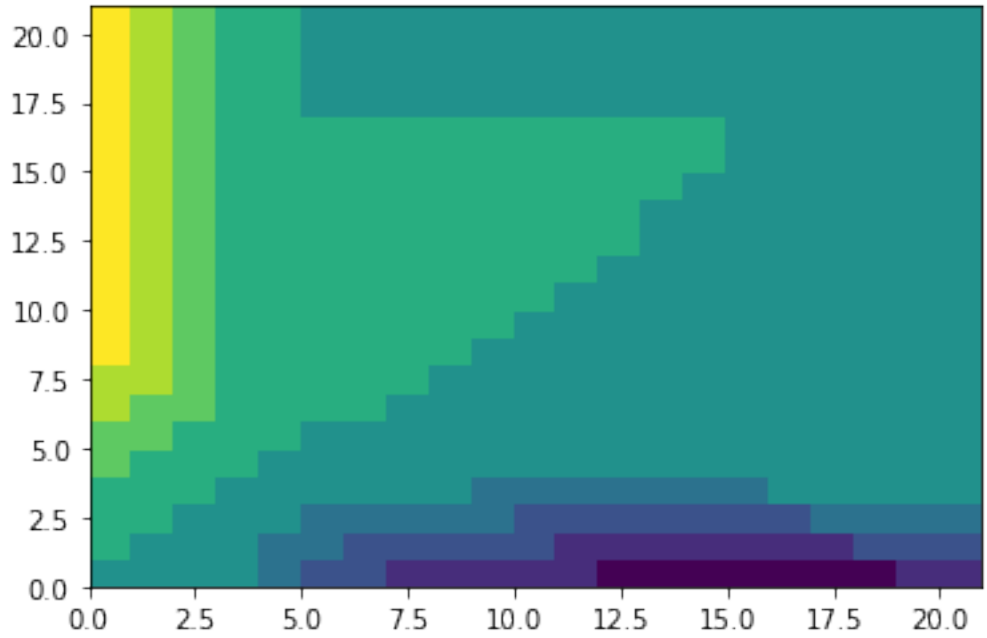

Figure 3: Optimal Policy

The plot for the state value function suggests the expected return of each state. The height for each point (state) in this plot represents its value so a higher plot is a better state to be in.

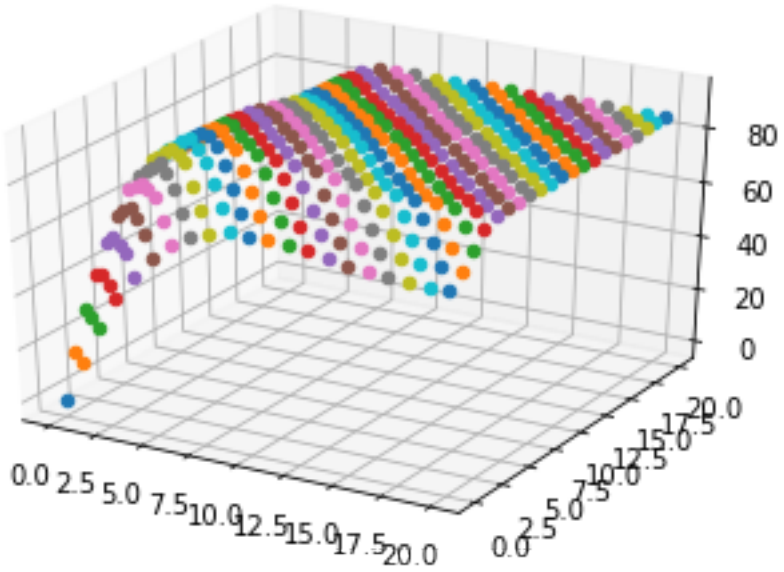

Figure 4: Final State Value Function