RL Lab Assignment - 3 (Value Iteration and Policy Iteration)

Name : Arvind Pandit

Roll No. : 211022001

Value Iteration Algorithm

Initialize V arbitrarily, e.g.,$V(s) = 0$, for all $s \in \mathcal{S}^+$

Repeat
    $\Delta \leftarrow 0$
    For each $s \in \mathcal{S}$:
        $v \leftarrow V(s)$
        $V(s) \leftarrow \max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)

Output a deterministic policy, π, such that
    $\pi(s) = \arg\max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$

Policy Iteration Algorithm

1. Initialization
   $V(s) \in \Re$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Repeat
   $\quad \Delta \leftarrow 0$
   $\quad$ For each $s \in \mathcal{S}$:
   $\qquad v \leftarrow V(s)$
   $\qquad V(s) \leftarrow \sum_{s'} \mathcal{P}_{ss'}^{\pi(s)} \left[\mathcal{R}_{ss'}^{\pi(s)} + \gamma V(s')\right]$
   $\qquad \Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   $\quad b \leftarrow \pi(s)$
   $\quad \pi(s) \leftarrow \arg\max_a \sum_{s'} \mathcal{P}_{ss'}^{a} \left[\mathcal{R}_{ss'}^{a} + \gamma V(s')\right]$
   $\quad$ If $b \neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop; else go to 2

Reference: Value iteration Policy Iteration

Problem 1.

Here we have taken a very simple problem of Markov Decision Process.

-1	**-1**	**10**	**-1**	**-1**

There are five 2D tiles which represent the state of MDP.

State = {0,1,2,3,4,5}

The reward of each state is given by [-1, -1, 10, -1, -1] and the goal is to reach terminal state 3rd which have reward of 10.

Reward = {-1, -1, 10, -1, -1}

The allowed action are left and right

Action = {0,1} where 0=left and 1=right

Here the agent have to collect the maximum reward and reach to the terminal state 3rd.

```
Probability Transition matrix for action a1-left

[[0.9 0.1 0.  0.  0. ]
 [0.9 0.  0.1 0.  0. ]
 [0.  0.  0.  0.  0. ]
 [0.  0.  0.9 0.  0.1]
 [0.  0.  0.  0.9 0.1]]
```

```
Probability Transition matrix for action a2-right

[[0.1 0.9 0.  0.  0. ]
 [0.1 0.  0.9 0.  0. ]
 [0.  0.  0.  0.  0. ]
 [0.  0.  0.1 0.  0.9]
 [0.  0.  0.  0.1 0.9]]
```

1. Value Iteration

```
No. of iterations in Value Iterions = 10
For Value iteration
Optimal Value function V* = [ 5.67496513  7.61064654 10.
7.61064654  5.67496513]
Optimal policy mu* = [1, 1, 0, 0, 0]
```

2. Policy Iterations

```
No. of iterations in Policy Iterions = 2
For Policy iteration
Optimal Value function V* = [ 5.67450141  7.61052229 10.
7.61079919  5.67554653]
Optimal policy mu* = [1. 1. 0. 0. 0.]
```

The Optimal Policy is mu=[1 1 0 0 0] s.t [right right left left left]

3. Verification by choosing Random Policy

```
Average collected reward with optimal policy mu [1. 1. 0. 0. 0.] is
7.314273885011818
```

```
Average collected reward with random policy mu_0 [0 1 1 0 1] is
3.9819028817869593
```

```
Average collected reward with random policy mu_1 [1 1 0 0 1] is
5.645207726534246
```

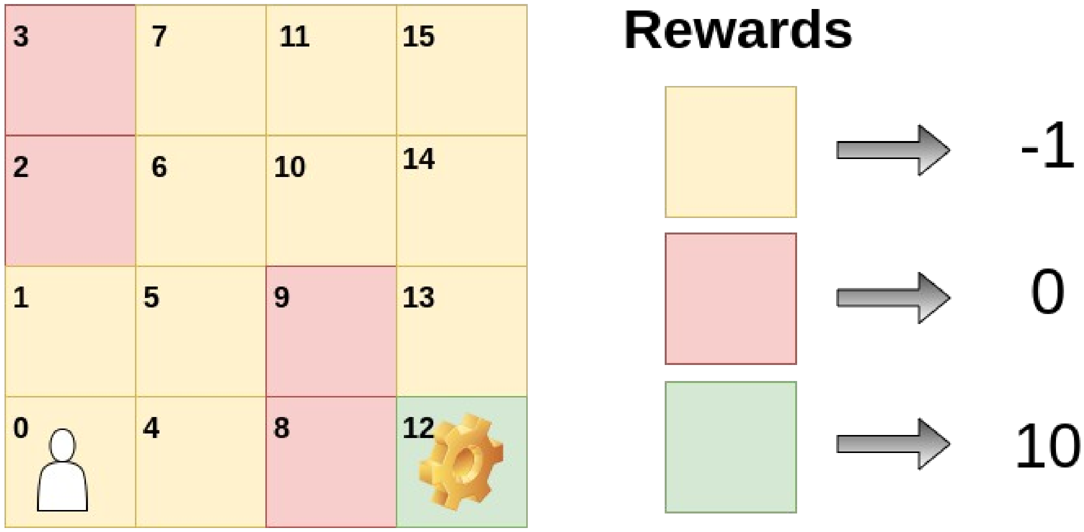
```
Average collected reward with random policy mu_2 [1 1 1 1 0] is
3.104047648468936
```

Problem 2.

We have taken a grid problem of Markov Decision Process.

Here the robot starts from the state 0 and there are walls in state (2,3,8,9) and the final destination of the robot is state 12.

The movement of the robot have the randomness such that it will go to the commanded direction with probability 0.7 and other 3 direction with each having probability 0.1.



There are 16 grids which represent the state of MDP.

State = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15}

The reward of each state is given by {-1,-1,0,0,-1,-1,-1,-1,0,0,-1,-1,10,-1,-1,-1} and the goal is to reach terminal state 12th which have reward of 10.

Each step will cost 1.

Reward = {-1,-1,0,0,-1,-1,-1,-1,0,0,-1,-1,10,-1,-1,-1}

The allowed action are up,down,left and right

Action = {0,1,2,3} where 0=up,1=down,2=left,3=right

Here the agent have to collect the maximum reward and reach to the terminal state 12 avoiding the collision with walls.

```
Probability Transition matrix for action a1-up

[[0.2 0.7 0.  0.  0.1 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]
 [0.1 0.8 0.  0.  0.  0.1 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.  1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]
 [0.1 0.  0.  0.  0.1 0.7 0.  0.  0.1 0.  0.  0.  0.  0.  0.  0. ]
 [0.  0.1 0.  0.  0.1 0.  0.7 0.  0.  0.1 0.  0.  0.  0.  0.  0. ]
 [0.  0.  0.1 0.  0.  0.1 0.  0.7 0.  0.  0.1 0.  0.  0.  0.  0. ]
 [0.  0.  0.  0.1 0.  0.  0.1 0.7 0.  0.  0.  0.1 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.  1.  0.  0.  0.  0.  0.  0. ]
 [0.  0.  0.  0.  0.  0.  0.1 0.  0.  0.1 0.  0.7 0.  0.  0.1 0. ]
 [0.  0.  0.  0.  0.  0.  0.  0.1 0.  0.  0.1 0.7 0.  0.  0.  0.1]
 [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.  0.1 0.  0.  0.1 0.1 0.7 0. ]
 [0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.1 0.  0.  0.1 0.1 0.7]
 [0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.1 0.  0.  0.1 0.8]]
```

Probability Transition matrix for action a2-down

```
[[0.8 0.1 0.  0.  0.1 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]
 [0.7 0.1 0.1 0.  0.  0.1 0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]
 [0.  0.7 0.1 0.1 0.  0.  0.1 0.  0.  0.  0.  0.  0.  0.  0.  0. ]
 [0.  0.  0.  0.9 0.  0.  0.  0.1 0.  0.  0.  0.  0.  0.  0.  0. ]
 [0.1 0.  0.  0.  0.7 0.1 0.  0.  0.1 0.  0.  0.  0.  0.  0.  0. ]
 [0.  0.1 0.  0.  0.7 0.  0.1 0.  0.  0.1 0.  0.  0.  0.  0.  0. ]
 [0.  0.  0.1 0.  0.  0.7 0.  0.1 0.  0.  0.1 0.  0.  0.  0.  0. ]
 [0.  0.  0.  0.1 0.  0.  0.7 0.1 0.  0.  0.  0.1 0.  0.  0.  0. ]
 [0.  0.  0.  0.  0.1 0.  0.  0.  0.7 0.1 0.  0.  0.1 0.  0.  0. ]
 [0.  0.  0.  0.  0.  0.1 0.  0.  0.  0.7 0.1 0.  0.  0.1 0.  0. ]
 [0.  0.  0.  0.  0.  0.  0.1 0.  0.  0.  0.7 0.1 0.  0.  0.1 0. ]
 [0.  0.  0.  0.  0.  0.  0.  0.1 0.  0.  0.7 0.1 0.  0.  0.  0.1]
 [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.  0.1 0.  0.  0.7 0.1 0.1 0. ]
 [0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.1 0.  0.  0.7 0.1 0.1]
 [0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.1 0.  0.  0.7 0.2]]
```

Probability Transition matrix for action a3-left

```
[[0.8 0.1 0.  0.  0.1 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]
 [0.1 0.7 0.1 0.  0.  0.1 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.  1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]
 [0.7 0.  0.  0.  0.1 0.1 0.  0.  0.1 0.  0.  0.  0.  0.  0.  0. ]
 [0.  0.7 0.  0.  0.1 0.  0.1 0.  0.  0.1 0.  0.  0.  0.  0.  0. ]
 [0.  0.  0.  0.  0.  0.1 0.7 0.1 0.  0.  0.1 0.  0.  0.  0.  0. ]
 [0.  0.  0.  0.  0.  0.  0.1 0.8 0.  0.  0.  0.1 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.  1.  0.  0.  0.  0.  0.  0. ]
 [0.  0.  0.  0.  0.  0.  0.7 0.  0.  0.1 0.  0.1 0.  0.  0.1 0. ]
 [0.  0.  0.  0.  0.  0.  0.  0.7 0.  0.  0.1 0.1 0.  0.  0.  0.1]
 [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.  0.1 0.8 0.1 0. ]
 [0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.7 0.  0.  0.1 0.1 0.1]
 [0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.7 0.  0.  0.1 0.2]]
```

Probability Transition matrix for action a4-right

```
[[0.2 0.1 0.  0.  0.7 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]
 [0.1 0.1 0.1 0.  0.  0.7 0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]
```

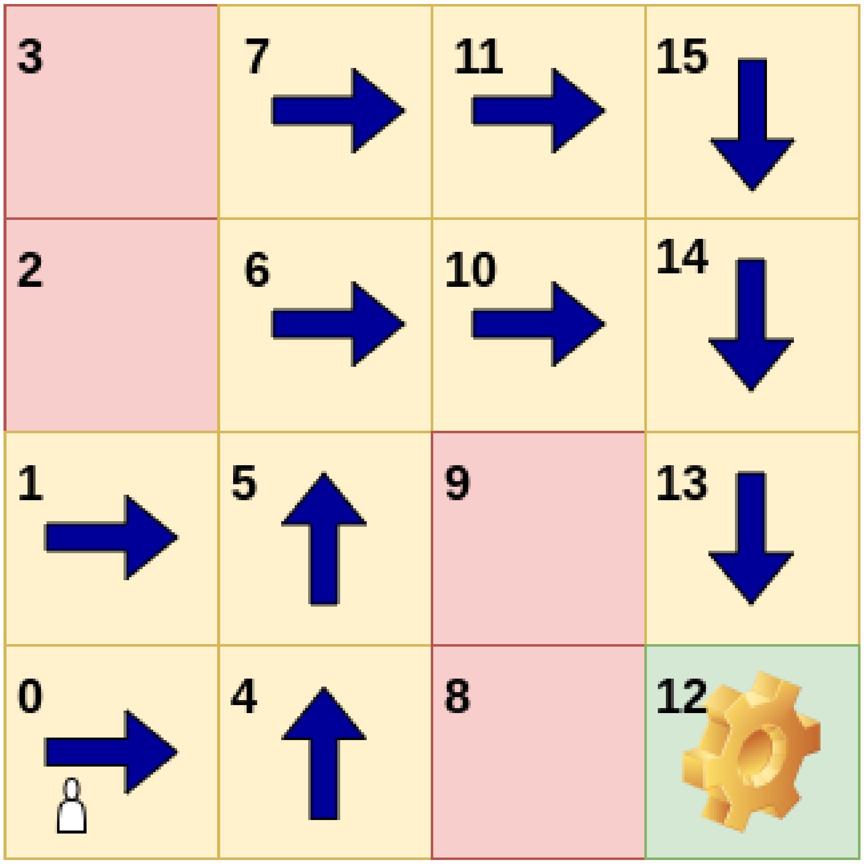
```
 [0.  0.  1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]
 [0.  0.  0.  1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]
 [0.1 0.  0.  0.  0.8 0.1 0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]
 [0.  0.1 0.  0.  0.1 0.7 0.1 0.  0.  0.  0.  0.  0.  0.  0.  0. ]
 [0.  0.  0.1 0.  0.  0.1 0.  0.1 0.  0.  0.7 0.  0.  0.  0.  0. ]
 [0.  0.  0.  0.1 0.  0.  0.1 0.1 0.  0.  0.  0.7 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.  1.  0.  0.  0.  0.  0.  0. ]
 [0.  0.  0.  0.  0.  0.  0.1 0.  0.  0.1 0.  0.1 0.  0.  0.7 0. ]
 [0.  0.  0.  0.  0.  0.  0.  0.1 0.  0.  0.1 0.1 0.  0.  0.  0.7]
 [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.  0.1 0.  0.  0.1 0.7 0.1 0. ]
 [0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.1 0.  0.  0.1 0.7 0.1]
 [0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.1 0.  0.  0.1 0.8]]
```

1. Value Iteration

```
No. of iterations in Value Iterions = 76
For Value iteration
Optimal Value function V* = [21.89782805 24.80772769 22.53287879
16.14270632 26.55055866 31.07872535
 38.0726069  34.11590598 40.15483901 38.60073338 49.49279352
43.12606991
 99.96670104 78.15737539 63.03837479 51.94169849]
Optimal policy mu* = [3, 3, 1, 1, 0, 0, 3, 3, 1, 1, 3, 3, 0, 1, 1, 1]
Optimal policy['right', 'right', 'nan', 'nan', 'up', 'up', 'right',
'right', 'nan', 'nan', 'right', 'right', 'nan', 'down', 'down',
'down']
```

2. Policy Iteration

```
[3. 3. 0. 0. 0. 0. 1. 1. 1. 0. 2. 2. 0. 1. 1. 1.]
[3. 1. 0. 0. 1. 1. 3. 3. 1. 1. 3. 3. 0. 1. 1. 1.]
[3. 3. 1. 1. 0. 0. 3. 3. 1. 1. 3. 3. 0. 1. 1. 1.]
[3. 3. 1. 1. 0. 0. 3. 3. 1. 1. 3. 3. 0. 1. 1. 1.]
No. of iterations in Policy Iterions = 4
For Value iteration
Optimal Value function V* = [ 21.92829068  24.83823287  22.5612402
16.15729915  26.58141653
  31.10992607  38.10416901  34.14586757  40.18609368  38.63246463
  49.52532705  43.15841139 100.          78.19043061  63.07129906
  51.9744841 ]
Optimal policy mu* = [3. 3. 1. 1. 0. 0. 3. 3. 1. 1. 3. 3. 0. 1. 1. 1.]
Optimal policy['right', 'right', 'nan', 'nan', 'up', 'up', 'right',
'right', 'nan', 'nan', 'right', 'right', 'nan', 'down', 'down',
'down']
```



3. Verification by choosing Random Policy

```
Verification by choosing Random Policy
Average collected reward with optimal policy mu [3. 3. 1. 1. 0. 0. 3.
3. 1. 1. 3. 3. 0. 1. 1. 1.] is 42.51030953722727


Average collected reward with random policy mu [2 3 2 1 0 0 2 2 0 3 0
1 0 2 1 3] is 7.855080063370282


Average collected reward with random policy mu [2 3 0 1 0 2 2 3 0 1 0
0 1 2 1 3] is 8.715658344714956


Average collected reward with random policy mu [3 0 2 1 3 3 0 3 2 0 3
3 3 2 2 3] is 3.951926359712211


Average collected reward with random policy mu [3 1 1 3 0 2 2 0 0 1 0
0 2 2 3 3] is 4.679206897793618
```

```
Average collected reward with random policy mu [3 1 3 0 3 2 2 1 3 1 1
1 2 2 3 2] is 4.390354843189319
```

4. Observations

Problem-1

Optimal Value function V* = [5.67 7.61 10. 7.61 5.67]

Optimal Policy mu* = ['right', 'right', 'nan', 'left', 'left']

Value Iteration

No. of iterations in Value Iterions = 10

Policy Iteration

No. of iterations for Vmu0 = 49

No. of iterations for Vmu1 = 9

No. of iterations in Policy Iterions = 2

Problem-2

Optimal Value function V* = [21.89 24.80 22.53 16.14 26.55 31.07 38.07 34.11 40.15 38.60 49.49 43.12 99.96 78.15 63.03 51.94]

Optimal policy mu* = ['right', 'right', 'nan', 'nan', 'up', 'up', 'right', 'right', 'nan', 'nan', 'right', 'right', 'nan', 'down', 'down', 'down']

Value Iteration

No. of iterations in Value Iterions = 76

Policy Iteration

No. of iterations for Vmu0 = 67

No. of iterations for Vmu1 = 28

No. of iterations for Vmu2 = 45

No. of iterations for Vmu3 = 36

No. of iterations in Policy Iterions = 4