REINFORCEMENT LEARNING
(CS3316)

Assignment Report

Assignment 1
SMALL GRIDWORLD

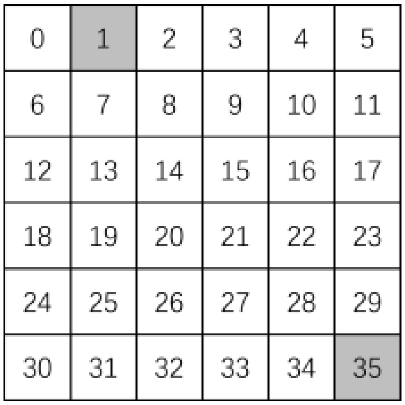
Name: Kezhi Li ID: 520021911013

Date: 16 March 2023

Contents

1 Introduction

The goal of this assignment is to apply Dynamic Programming (DP) on a Gridworld to find the shortest way from any state to the terminal state in the grid. As shown in Fig. 1, the Gridworld has 6 × 6 grids and hence the state space can be denoted as $S = \{s_t | t \in 0, \ldots, 35\}$, with S_1 and S_{35} the terminal states. The action space is $A = \{n, e, s, w\}$, which represent moving north, east, south and west respectively. Moving out of the Gridworld will not leave the grid. The probabilities of state transition are always 1 from any state to another. Each movement get a reward of -1 until the terminal state is reached.



0	1	2	3	4	5
6	7	8	9	10	11
12	13	14	15	16	17
18	19	20	21	22	23
24	25	26	27	28	29
30	31	32	33	34	35

Figure 1: Gridworld

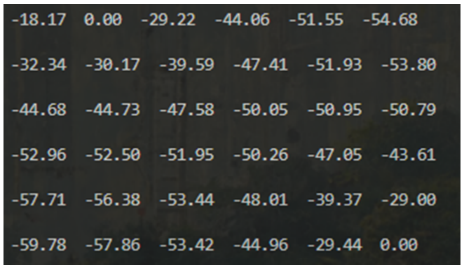
2 Experiment

For a given uniform random policy $\pi(n|\cdot) = \pi(n|\cdot) = \pi(n|\cdot) = \pi(n|\cdot) = 0.25$, the iterative policy evaluation should be used to evaluate the policy and the policy iteration and value iteration should be used to improve policy.

3 Results

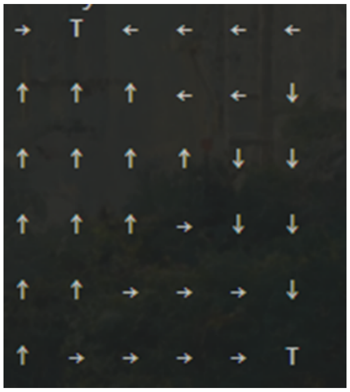
3.1 Iterative policy evaluation

The state value functions for the uniform random policy being evaluated by iterative policy evaluation is shown in Fig. 2. Besides, the optimal policy of this state function is shown in Fig. 3. In Fig. 3, the arrow represents the optimal moving direction in a single grid, and the $'T'$ token represents the terminal state. It can be concluded that the random policy is far from the best policy, since for some grids in Fig. 3, the routes from the grid to terminal are not the shortest one.



```
-18.17  0.00  -29.22  -44.06  -51.55  -54.68

-32.34  -30.17  -39.59  -47.41  -51.93  -53.80

-44.68  -44.73  -47.58  -50.05  -50.95  -50.79

-52.96  -52.50  -51.95  -50.26  -47.05  -43.61

-57.71  -56.38  -53.44  -48.01  -39.37  -29.00

-59.78  -57.86  -53.42  -44.96  -29.44  0.00
```

Figure 2: State function of uniform random policy



```
→  T  ←  ←  ←  ←

↑  ↑  ↑  ←  ←  ↓

↑  ↑  ↑  ↑  ↓  ↓

↑  ↑  ↑  →  ↓  ↓

↑  ↑  →  →  →  ↓

↑  →  →  →  →  T
```

Figure 3: Optimal policy for the state function of random policy

3.2 Policy iteration and value iteration

To improve the uniform random policy and finally find the best policy for this Gridworld, we can use the policy iteration or value iteration. For policy iteration, we use the current policy to iterate the state value function until it converges, and then replace the old policy with a greedy policy derived from the state value function of the old policy.

Fig. 4 shows all iterations results for the policy iteration. After three iterations, the optimal policy for the whole problem is found.

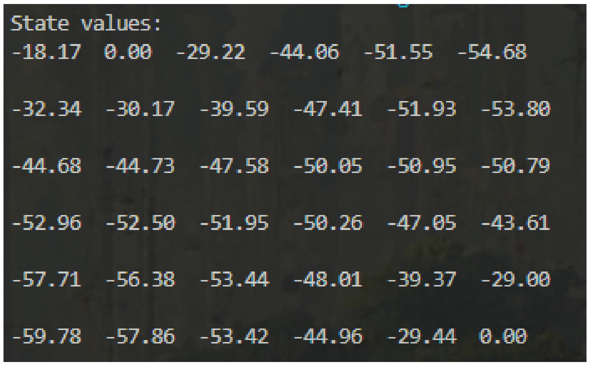
For the value iteration, we do not wait the state value function to converge at every policy, but only calculate the maximal reward of actions for one step and update the policy. This method is equivalent to change the step of policy iteration to be 1.

As shown in Fig. 5. the value iteration ran 5 iterations to get the final best result. The final optimal policy is the same as the policy derived from the policy iteration.

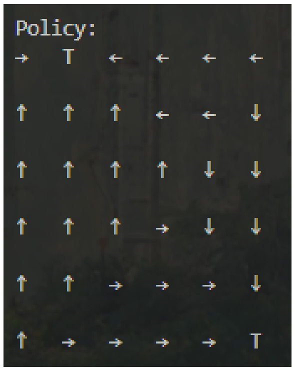
4 Conclusion

The iterative policy evaluation can evaluate the goodness of a given policy. Both the policy iteration and the value iteration can find the improve the random policy and find the optimal policy. Confirmed with the principles, the value iteration uses more steps to find the optimal policy, since it does not wait a single policy during the process to converge. However, for each iteration, the policy iteration will consume more time to find the convergent result for the policy. In conclusion, it depends when choosing the value iteration or the policy iteration to find the optimal policy in reinforcement learning.

Iteration 1

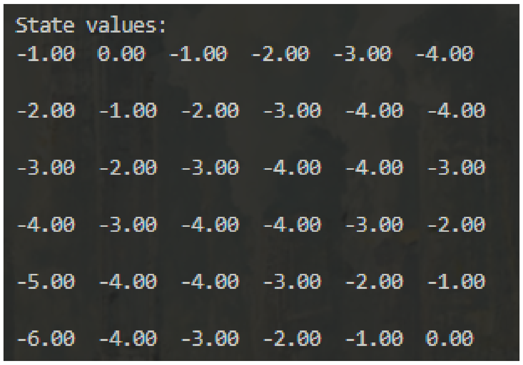


```
State values:
-18.17  0.00  -29.22  -44.06  -51.55  -54.68

-32.34  -30.17  -39.59  -47.41  -51.93  -53.80

-44.68  -44.73  -47.58  -50.05  -50.95  -50.79

-52.96  -52.50  -51.95  -50.26  -47.05  -43.61

-57.71  -56.38  -53.44  -48.01  -39.37  -29.00

-59.78  -57.86  -53.42  -44.96  -29.44  0.00
```

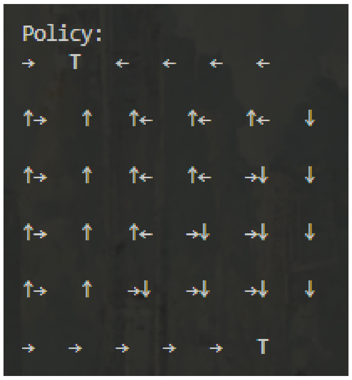


```
Policy:
→   T   ←   ←   ←   ←

↑   ↑   ↑   ←   ←   ↓

↑   ↑   ↑   ↑   ↓   ↓

↑   ↑   ↑   →   ↓   ↓

↑   ↑   →   →   →   ↓

↑   →   →   →   →   T
```

Iteration 2

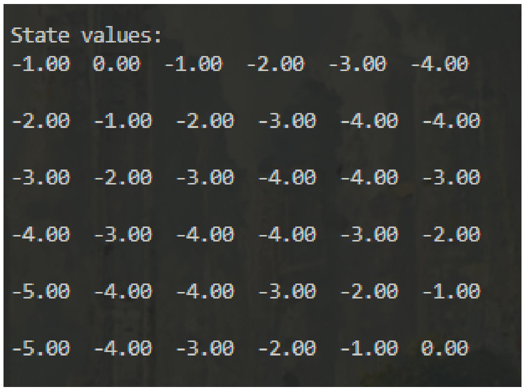


```
State values:
-1.00  0.00  -1.00  -2.00  -3.00  -4.00

-2.00  -1.00  -2.00  -3.00  -4.00  -4.00

-3.00  -2.00  -3.00  -4.00  -4.00  -3.00

-4.00  -3.00  -4.00  -4.00  -3.00  -2.00

-5.00  -4.00  -4.00  -3.00  -2.00  -1.00

-6.00  -4.00  -3.00  -2.00  -1.00  0.00
```

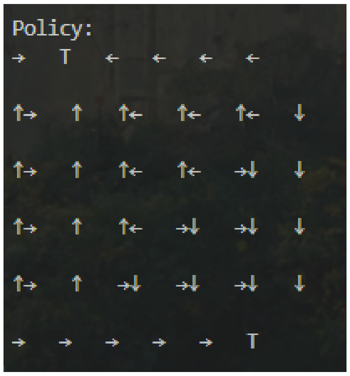


```
Policy:
→    T    ←    ←    ←    ←

↑→   ↑    ↑←   ↑←   ↑←   ↓

↑→   ↑    ↑←   ↑←   →↓   ↓

↑→   ↑    ↑←   →↓   →↓   ↓

↑→   ↑    →↓   →↓   →↓   ↓

→    →    →    →    →    T
```

Iteration 3

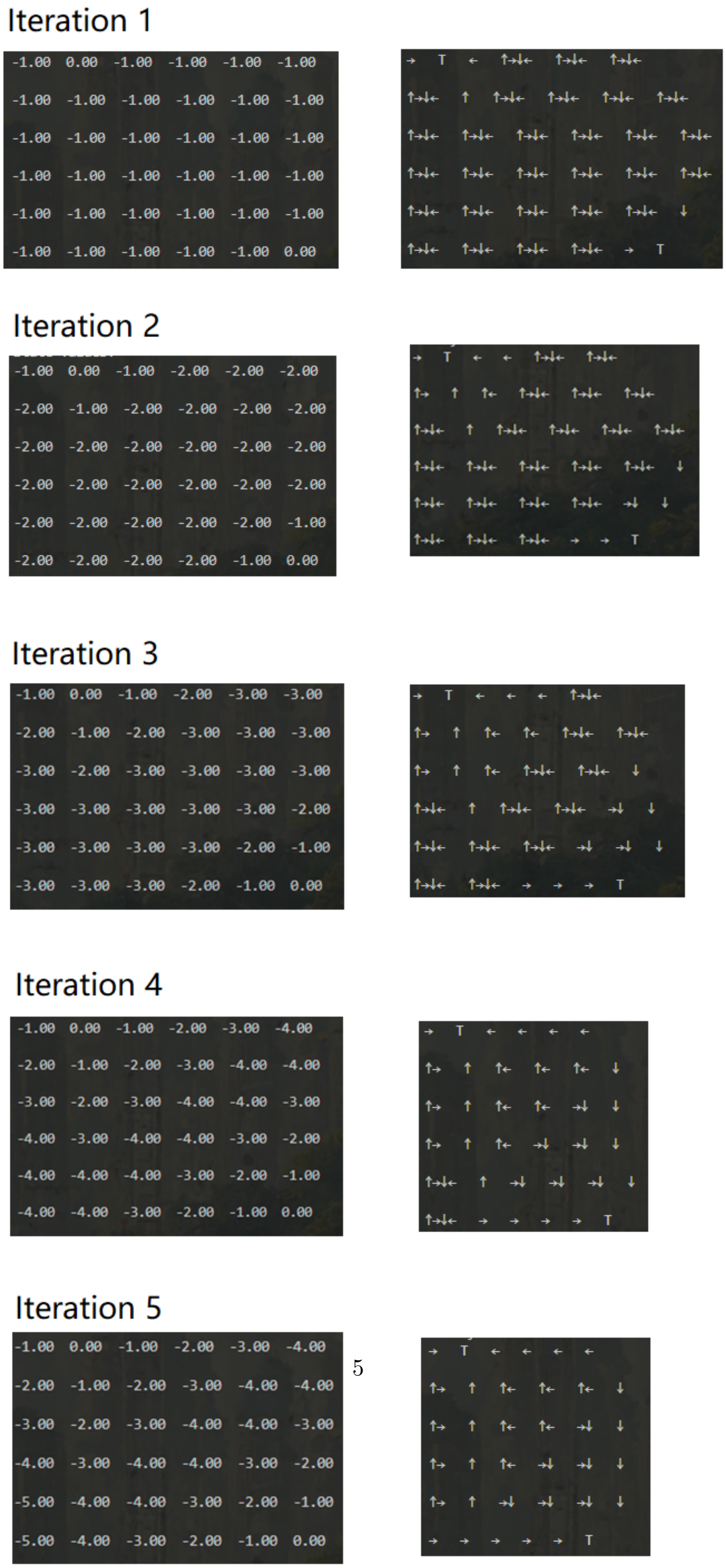


```
State values:
-1.00  0.00  -1.00  -2.00  -3.00  -4.00

-2.00  -1.00  -2.00  -3.00  -4.00  -4.00

-3.00  -2.00  -3.00  -4.00  -4.00  -3.00

-4.00  -3.00  -4.00  -4.00  -3.00  -2.00

-5.00  -4.00  -4.00  -3.00  -2.00  -1.00

-5.00  -4.00  -3.00  -2.00  -1.00  0.00
```



```
Policy:
→    T    ←    ←    ←    ←

↑→   ↑    ↑←   ↑←   ↑←   ↓

↑→   ↑    ↑←   ↑←   →↓   ↓

↑→   ↑    ↑←   →↓   →↓   ↓

↑→   ↑    →↓   →↓   →↓   ↓

→    →    →    →    →    T
```

Figure 4: The state value functions and respective optimal policy for all iterations



Iteration 1

```
-1.00  0.00 -1.00 -1.00 -1.00 -1.00
-1.00 -1.00 -1.00 -1.00 -1.00 -1.00
-1.00 -1.00 -1.00 -1.00 -1.00 -1.00
-1.00 -1.00 -1.00 -1.00 -1.00 -1.00
-1.00 -1.00 -1.00 -1.00 -1.00 -1.00
-1.00 -1.00 -1.00 -1.00 -1.00  0.00
```

```
→     T     ←     ↑→↓←  ↑→↓←  ↑→↓←
↑→↓←  ↑     ↑→↓←  ↑→↓←  ↑→↓←  ↑→↓←
↑→↓←  ↑→↓←  ↑→↓←  ↑→↓←  ↑→↓←  ↑→↓←
↑→↓←  ↑→↓←  ↑→↓←  ↑→↓←  ↑→↓←  ↑→↓←
↑→↓←  ↑→↓←  ↑→↓←  ↑→↓←  ↑→↓←  ↓
↑→↓←  ↑→↓←  ↑→↓←  ↑→↓←  →     T
```

Iteration 2

```
-1.00  0.00 -1.00 -2.00 -2.00 -2.00
-2.00 -1.00 -2.00 -2.00 -2.00 -2.00
-2.00 -2.00 -2.00 -2.00 -2.00 -2.00
-2.00 -2.00 -2.00 -2.00 -2.00 -2.00
-2.00 -2.00 -2.00 -2.00 -2.00 -1.00
-2.00 -2.00 -2.00 -2.00 -1.00  0.00
```

```
→     T     ←     ←     ↑→↓←  ↑→↓←
↑→    ↑     ↑←    ↑→↓←  ↑→↓←  ↑→↓←
↑→↓←  ↑     ↑→↓←  ↑→↓←  ↑→↓←  ↑→↓←
↑→↓←  ↑→↓←  ↑→↓←  ↑→↓←  ↑→↓←  ↓
↑→↓←  ↑→↓←  ↑→↓←  ↑→↓←  →↓    ↓
↑→↓←  ↑→↓←  ↑→↓←  →     →     T
```

Iteration 3

```
-1.00  0.00 -1.00 -2.00 -3.00 -3.00
-2.00 -1.00 -2.00 -3.00 -3.00 -3.00
-3.00 -2.00 -3.00 -3.00 -3.00 -3.00
-3.00 -3.00 -3.00 -3.00 -3.00 -2.00
-3.00 -3.00 -3.00 -3.00 -2.00 -1.00
-3.00 -3.00 -3.00 -2.00 -1.00  0.00
```

```
→     T     ←     ←     ←     ↑→↓←
↑→    ↑     ↑←    ↑←    ↑→↓←  ↑→↓←
↑→    ↑     ↑←    ↑→↓←  ↑→↓←  ↓
↑→↓←  ↑     ↑→↓←  ↑→↓←  →↓    ↓
↑→↓←  ↑→↓←  ↑→↓←  →↓    →↓    ↓
↑→↓←  ↑→↓←  →     →     →     T
```

Iteration 4

```
-1.00  0.00 -1.00 -2.00 -3.00 -4.00
-2.00 -1.00 -2.00 -3.00 -4.00 -4.00
-3.00 -2.00 -3.00 -4.00 -4.00 -3.00
-4.00 -3.00 -4.00 -4.00 -3.00 -2.00
-4.00 -4.00 -4.00 -3.00 -2.00 -1.00
-4.00 -4.00 -3.00 -2.00 -1.00  0.00
```

```
→     T     ←     ←     ←     ←
↑→    ↑     ↑←    ↑←    ↑←    ↓
↑→    ↑     ↑←    ↑←    →↓    ↓
↑→    ↑     ↑←    →↓    →↓    ↓
↑→↓←  ↑     →↓    →↓    →↓    ↓
↑→↓←  →     →     →     →     T
```

Iteration 5

```
-1.00  0.00 -1.00 -2.00 -3.00 -4.00
-2.00 -1.00 -2.00 -3.00 -4.00 -4.00
-3.00 -2.00 -3.00 -4.00 -4.00 -3.00
-4.00 -3.00 -4.00 -4.00 -3.00 -2.00
-5.00 -4.00 -4.00 -3.00 -2.00 -1.00
-5.00 -4.00 -3.00 -2.00 -1.00  0.00
```

```
→     T     ←     ←     ←     ←
↑→    ↑     ↑←    ↑←    ↑←    ↓
↑→    ↑     ↑←    ↑←    →↓    ↓
↑→    ↑     ↑←    →↓    →↓    ↓
↑→    ↑     →↓    →↓    →↓    ↓
→     →     →     →     →     T
```

Figure 5: The state value functions and respective optimal policy for all iterations