CE4046 Intelligent Agent Assignment 1

Tianrun Hu

February 23, 2024

1 Method of value iteration

1.1 Description of Implemented Solution

Initialization: The environment is pass into the function by a class contains all the walls and rewards. A SummaryWriter object from tensorboardX is created for logging purposes. A utility values array V if initialized with zeros for each state in the environment.

Iteration: The update step in the value iteration algorithm is:

$$V_{k+1}(s) = max_a \sum_{s'} P(s'|s,a)[R(s,a,s'), +\gamma V_k(s')]$$

where $V_k(s)$ is the utility of state s at iteration k, max_a denotes the maximum value over all possible actions a, $P(s'|s,a)$ is the probability of transitioning to state s' from state s to state s' via action a, γ is the discount factor which prioritizes immediate rewards over distant rewards, $V_k(s')$ is the utility of state s' at iteration k.

Run for 50 iterations, and keep tracking the maximum change in utility values δ. If it falls below a specified threshold, indicating convergence.

Plotting: Plot the values and policies calculated from the previous algorithm.

1.2 Plot of Optimal Policy

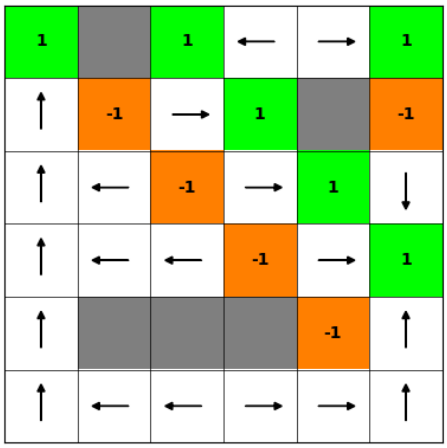


Figure 1: Optimal policy for the gridworld environment. The arrows indicate the action to be taken in each state. The color of the arrows represent the value of the action.

1.3 Utility of all states

1		1	0.99	0.99	1
0.77	-1	0.80	1		-1
0.69	0.46	-1	0.80	1	0.99
0.62	0.55	0.33	-1	0.80	1
0.56				-1	0.77
0.50	0.44	0.39	0.39	0.45	0.68

Figure 2: Utility of all states in the gridworld environment. The color of the cells represent the value of the state.

The more detailed value for the final utility is:

0.0	0.0	0.0	0.99445061	0.98753071	0.0
0.77247503	0.0	0.8	0.0	0.0	0.0
0.68529708	0.4611854	0.0	0.8	0.0	0.99445061
0.61840233	0.54986211	0.33239009	0.0	0.8	0.0
0.56078941	0.0	0.0	0.0	0.0	0.77247503
0.49686767	0.4406518	0.38504299	0.3947778	0.45040336	0.68411722

1.4 Convergence of value iteration

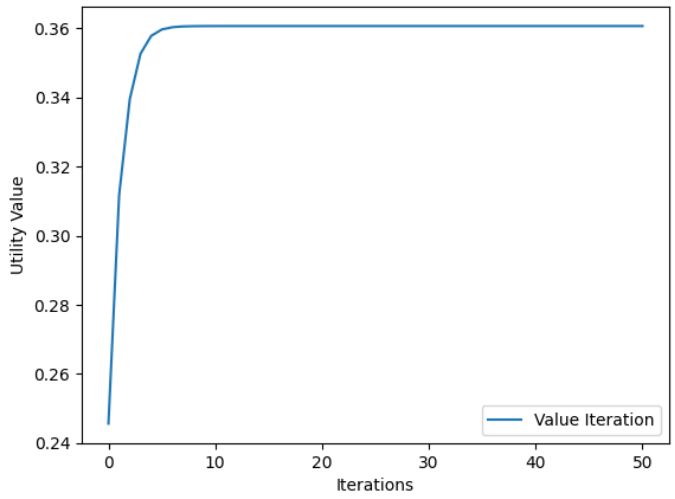


Figure 3: Convergence of value iteration. The y-axis represents the maximum change in utility values across all states. The x-axis represents the number of iterations.

2 Method of policy iteration

2.1 Description of Implemented Solution

Initialization: The setting is basically same with value iteration. Except here we initialize the policy as all actions are set to move right (0, 1).

Policy Evaluation: For each state, the expected utility is computed based on the current policy's action, considering the probability of transitions to all possible next states and the corresponding rewards.

Policy iteration: The iteration consists of two steps: evaluation and improvement.

Policy Evaluation:

$$V^{\pi}(s) = \sum_{s',r} P(s', r|s, \pi(s))[r + \gamma V^{\pi}(s')]$$

where $V^{\pi}(s)$ is the utility of state s under policy π, $P(s', r|s, \pi(s))$ is the probability of transitioning to state s' with reward r from state s when following policy π. γ is the discount factor and $\pi(s)$ is the action prescribed by policy π in state s.

Policy Improvement:

$$\pi'(s) = argmax_a \sum_{s',r} P(s', r|s, a)[r + \gamma V^{\pi}(s')]$$

where $pi'(s)$ is the updated action for state s under the improved policy π', $argmax_a$ denotes the action that maximizes the expected utility.

2.2 Plot of Optimal Policy

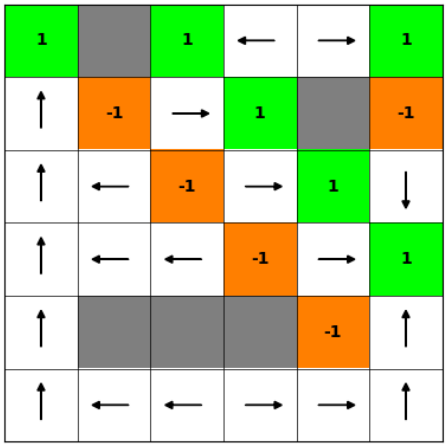


Figure 4: Optimal policy for the gridworld environment. The arrows indicate the action to be taken in each state. The color of the arrows represent the value of the action.

2.3 Utility of all states

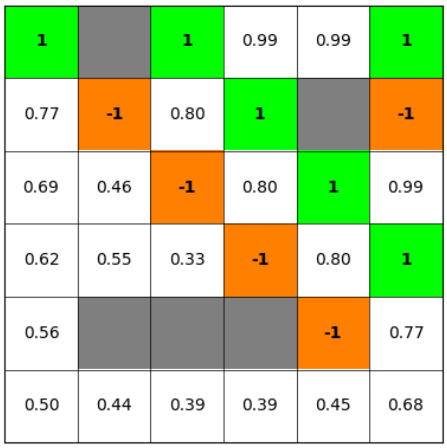


Figure 5: Utility of all states in the gridworld environment. The color of the cells represent the value of the state.

0.0	0.0	0.0	0.99445061	0.98753071	0.0
0.77247503	0.0	0.8	0.0	0.0	0.0
0.68529708	0.4611854	0.0	0.8	0.0	0.99445061
0.61840233	0.54986211	0.33239009	0.0	0.8	0.0
0.56078941	0.0	0.0	0.0	0.0	0.77247503
0.49686767	0.4406518	0.38504299	0.3947778	0.45040336	0.68411722

It shows that in the default maze setting, the policy iteration and value iteration will converge to the same utility values.

2.4 Convergence of policy iteration

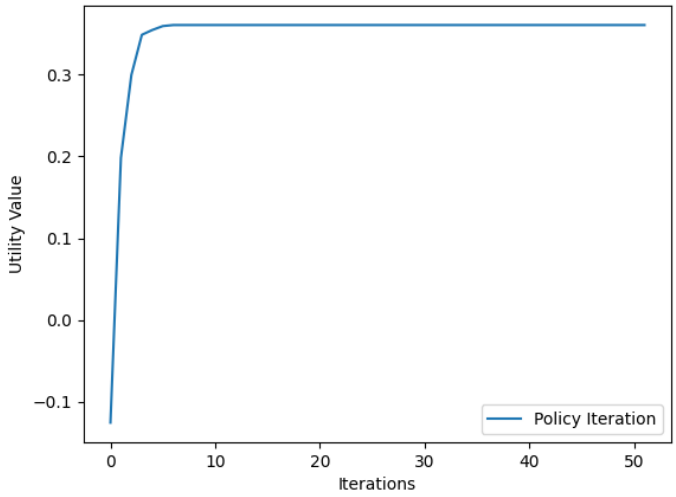
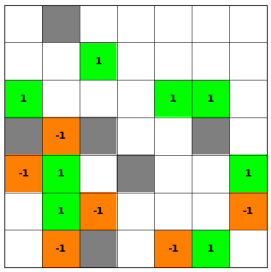


Figure 6: Convergence of policy iteration. The y-axis represents the maximum change in utility values across all states. The x-axis represents the number of iterations.

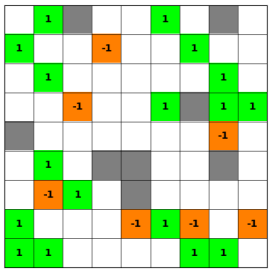
3 Bonus questions

3.1 Environment generation

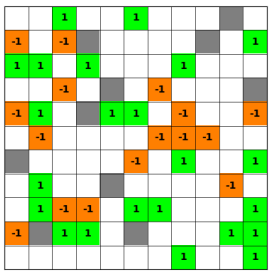
Randomly generated environment: The environment is generated by randomly placing walls and rewards in the gridworld. The number of walls and rewards are specified by the user. The rewards are placed randomly in the environment, and the walls are placed randomly in the environment. The generated environment is save as yaml file in config folder. Here are some examples of the generated environment:



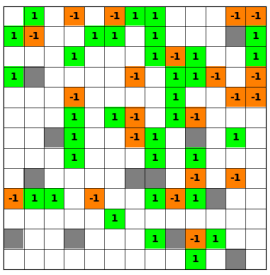
(a) Maze size 7



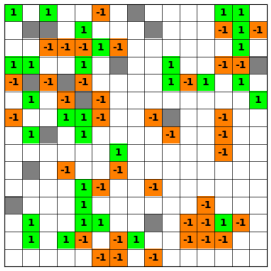
(b) Maze size 9



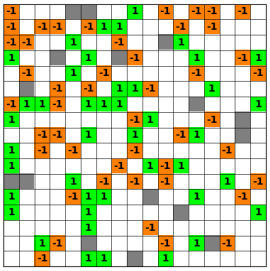
(a) Maze size 11



(b) Maze size 13



(a) Maze size 15



(b) Maze size 17

Figure 9: Randomly generated environments.

3.2 Comparison of value iteration and policy iteration

Convergence among different maze complexity: The convergence of value iteration and policy iteration for different maze size is shown in Figure 3 and Figure 6.

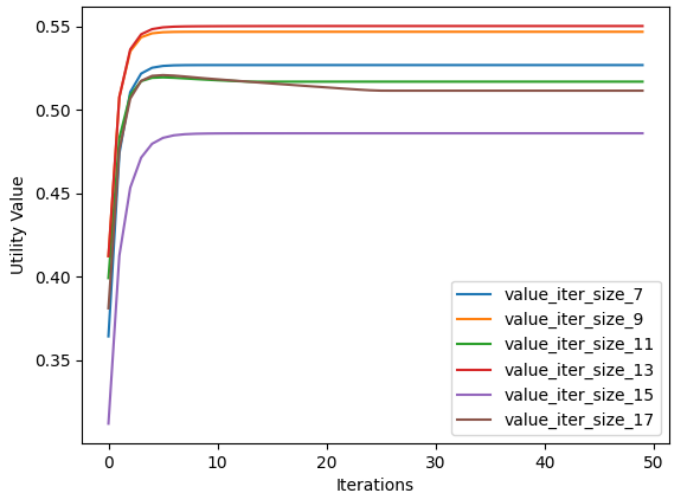


Figure 10: The value iteration curves for different maze size.

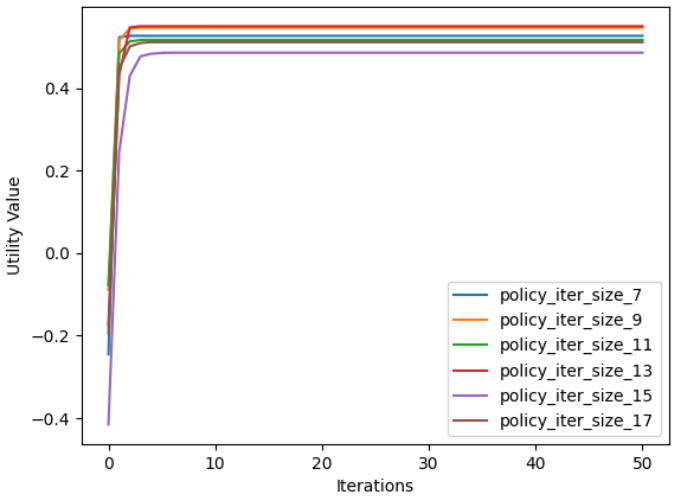


Figure 11: The policy iteration curves for different maze size.

It shows that the maze complexity will affect the convergence of value iteration and policy iteration. The larger the maze size, the slower the convergence, and the lower final utility values will be.