
Assignment 1 - Policy iteration and Value iteration for grid example

Author: Supreeth Suresh

Submission Date: 9/19/2025

Course: AI for Multiagent system

How to run the code

The code is located on Github <https://github.com/supreethms1809/multiagent.git>.

```
1 python assignment1_main.py [-h] \\  
2   [--task {policy_iteration,value_iteration}] \\  
3   [--gamma GAMMA] [--epsilon EPSILON] \\  
4   [--max_iterations MAX_ITERATIONS] \\  
5   [--grid_size grid_size] [--stepReward STEPREWARD] \\  
6   [--goalReward GOALREWARD] \\  
7   [--valueFunctionInit {V,Q}] \\  
8   [--randomValueFunctionInit] \\  
9   [--uniformPolicyInit] \\  
10  [--problem {1,2,3,4}] [--plotTable] \\  
11  [--goalStates GOALSTATES] \\  
12  [--splStates SPECIALSTATES] \\  
13  [--splReward SPLREWARD]  
14  
15  options:  
16  -h, --help            show this help message and exit  
17  --task {policy_iteration,value_iteration}  
18  --gamma GAMMA          Gamma for the value  
19  iteration              Epsilon for the value  
20  --epsilon EPSILON      Epsilon for the value  
21  iteration              Maximum number of  
22  --max_iterations MAX_ITERATIONS  
23  iterations             for the value iteration and  
24  policy iteration      Size of the grid N  
25  --grid_size grid_size Step reward  
26  --stepReward STEPREWARD Goal reward  
27  --goalReward GOALREWARD Type of value function used  
28  --valueFunctionInit {V,Q} V or Q  
29  --randomValueFunctionInit Initialize the value  
30  function              with random values  
31  --uniformPolicyInit    Initialize the policy  
32  with uniform distribution  
33  --problem {1,2,3,4}    Problem number  
34  --plotTable            Plot the value function
```

```
34                                     and policy
35     --goalStates GOALSTATES        Goal states list. Format
36                                     list of tuples
37                                     [(x, y), (x, y), ...]
38     --splStates SPECIALSTATES      Spl states list. Format
39                                     list of tuples
40                                     [(x, y), (x, y), ...]
41     --splReward SPLREWARD           Special state reward
```

The configurations for the four problems given in the assignment is hardcoded in the source code for convinience. Alternatively, you can also pass the configurations as the command line options. The usage is shown above. The code is modular and well commented for description. The code should be able to handle bigger square grid as well.

To run problem 1 use

```
1  python assignment1_main.py --problem 1
2
3  # Problem 1 sets the following options
4  # config.stepReward = -1
5  # config.goalReward = 0
6  # config.gamma = 0.9
7  # config.epsilon = 1e-6
8  # config.max_iterations = 150
9  # config.grid_size = 4
10 # config.valueFunctionInit = "V"
11 # config.randomValueFunctionInit = False
12 # config.uniformPolicyInit = True
13 # config.task = "policy_iteration"
14 # config.plotTable = True
15 # config.goalStates = [(0, 0), (3, 3)]
16 # config.splStates = None
17 # config.splReward = None
```

To run problem 2,3,4

```
1  python assignment1_main.py --problem 2
2  python assignment1_main.py --problem 3
3  python assignment1_main.py --problem 4
```

Problem description

- Grid map problem, one agent moves on the grid map.
- The terminal states are (0,0) --> 0 and (3,3) --> 15.
- Reward for going to the terminal state 0.

Problem 1 - Policy Iteration

- Policy iteration
- Policy is uniform distribution policy
- every step generates reward of -1
- Goal reward is 0
- gamma $\gamma = 0.9$
- Goal state (0,0) and (3,3)
- Evaluate the policy iteratively
- Plot the value of each state after the policy evaluation is complete(One plot)
- Tips: 1. run more than 150 iterations. 2. set the convergence threshold less than $1e-6$

Policy iteration consists of two parts. First is the policy evaluation and the policy improvement. In this phase, we calculate the value function using the Bellman expectation equation under the current policy

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

We run this update for all states until the values converge (i.e., they stop changing between successive iterations) or until a maximum number of iterations is reached. During this process, the policy remains fixed.

Once we have an updated value function, we improve the policy by making it greedy with respect to the current value estimates:

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

If the policy changes, we repeat the evaluation and improvement steps. If the policy remains unchanged (i.e., stable), then we have reached convergence.

Run `python assignment1_main.py --problem 1`

```
1 # Problem 1 sets the following options
2 # config.stepReward = -1
3 # config.goalReward = 0
4 # config.gamma = 0.9
5 # config.epsilon = 1e-6
6 # config.max_iterations = 200
7 # config.grid_size = 4
8 # config.valueFunctionInit = "V"
9 # config.randomValueFunctionInit = True
10 # config.uniformPolicyInit = True
11 # config.task = "policy_iteration"
12 # config.plotTable = True
13 # config.goalStates = [(0, 0), (3, 3)]
14 # config.splStates = None
15 # config.splReward = None
```

Running problem 1

```
1 INFO:__main__:Performing calculations for Prob 1: policy_iteration with
  V value function and uniform distribution for policy initialization
2 INFO:__main__:Using V value function with random initialization
3 INFO:__main__:Using uniform policy initialization
4 INFO:__main__:Starting policy iteration with 200 max iterations
5 INFO:__main__:Value function converged after 62 evaluation iterations
6 INFO:__main__:Plotting value function
7 INFO:__main__:Value function converged after 4 evaluation iterations
8 INFO:__main__:Value function converged after 1 evaluation iterations
9 INFO:__main__:Policy converged after 3 iterations
10 INFO:__main__:Policy iteration converged successfully
```

Plot the value of each state after the policy evaluation is complete: Value and action of each state after the policy evaluation is complete

Values for each state after policy evaluation is complete

0.000 State 0 - TERM	-4.75 State 1	-6.81 State 2	-7.39 State 3
-4.75 State 4	-6.23 State 5	-6.87 State 6	-6.81 State 7
-6.81 State 8	-6.87 State 9	-6.23 State 10	-4.75 State 11
-7.39 State 12	-6.81 State 13	-4.75 State 14	0.000 State 15 - TERM

Terminal States Regular States

Problem 2 - Policy Iteration

- Policy iteration
- Policy is uniform distribution policy

-
- every step generates reward of -4
 - Goal state reward 0
 - gamma $\gamma = 0.9$
 - Special state reward -1
 - Special state (2,2) --> state 10,
 - Goal state (0,0) and (3,3)
 - Evaluate the policy iteratively
 - Plot the value of each state after the policy evaluation is complete(One plot)

Run `python assignment1_main.py --problem 2`

```
1 # Problem 2 sets the following options
2 # config.stepReward = -4
3 # config.goalReward = 0
4 # config.gamma = 0.9
5 # config.epsilon = 1e-6
6 # config.max_iterations = 200
7 # config.grid_size = 4
8 # config.valueFunctionInit = "V"
9 # config.randomValueFunctionInit = True
10 # config.uniformPolicyInit = True
11 # config.task = "policy_iteration"
12 # config.plotTable = True
13 # config.goalStates = [(0, 0), (3, 3)]
14 # config.splStates = [(2,2)]
15 # config.splReward = -1
```

Running problem 2

```
1 INFO:__main__:Performing calculations for Prob 2: policy_iteration with
  V value function and uniform distribution for policy initialization
2 INFO:__main__:Using V value function with random initialization
3 INFO:__main__:Using uniform policy initialization
4 INFO:__main__:Starting policy iteration with 200 max iterations
5 INFO:__main__:Value function converged after 62 evaluation iterations
6 INFO:__main__:Plotting value function
7 INFO:__main__:Value function converged after 4 evaluation iterations
8 INFO:__main__:Value function converged after 1 evaluation iterations
9 INFO:__main__:Policy converged after 3 iterations
10 INFO:__main__:Policy iteration converged successfully
```

Plot the value of each state after the policy evaluation is complete: Value and action of each state after the policy evaluation is complete

Values for each state after policy evaluation is complete

0.000 State 0 - TERM	-18.37 State 1	-26.19 State 2	State 3
-18.37 State 4	-23.75 State 5	-25.52 State 6	-25.84 State 7
-26.19 State 8	-25.52 State 9	-23.20 State 10 - SPL	-17.14 State 11
-28.56 State 12	-25.84 State 13	-17.14 State 14	0.000 State 15 - TERM

Terminal States

Special States

Regular States

Problem 3 - Policy Iteration

- Policy iteration
- Policy is uniform distribution policy

-
- every step generates reward -4
 - Goal state reward 0
 - gamma $\gamma = 0.9$
 - Special state reward -1
 - Special state (2,2) --> state 10
 - Goal state (0,0) and (3,3)
 - Evaluate the policy iteratively

Run `python assignment1_main.py --problem 3`

```
1 # Problem 3 sets the following options
2 # config.stepReward = -4
3 # config.goalReward = 0
4 # config.gamma = 0.9
5 # config.epsilon = 1e-6
6 # config.max_iterations = 200
7 # config.grid_size = 4
8 # config.valueFunctionInit = "V"
9 # config.randomValueFunctionInit = True
10 # config.uniformPolicyInit = True
11 # config.task = "policy_iteration"
12 # config.plotTable = True
13 # config.goalStates = [(0, 0), (3, 3)]
14 # config.splStates = [(2,2)]
15 # config.splReward = -1
```

Running problem 3

```
1 INFO:__main__:Performing calculations for Prob 3: policy_iteration with
  V value function and uniform distribution for policy initialization
2 INFO:__main__:Using V value function with random initialization
3 INFO:__main__:Using uniform policy initialization
4 INFO:__main__:Starting policy iteration with 200 max iterations
5 INFO:__main__:Value function converged after 62 evaluation iterations
6 INFO:__main__:Plotting optimal policy with values and actions
7 INFO:__main__:Value function converged after 4 evaluation iterations
8 INFO:__main__:Plotting optimal policy with values and actions
9 INFO:__main__:Value function converged after 1 evaluation iterations
10 INFO:__main__:Plotting optimal policy with values and actions
11 INFO:__main__:Policy converged after 3 iterations
12 INFO:__main__:Policy iteration converged successfully
13 INFO:__main__:Plotting value function
14 INFO:__main__:Plotting optimal policy with values and actions
```


Plot the comparison of value and optimal action of each state after each policy improvement (similar to the slides). As many plots as the number of policy improvement goes

Value and action of each state after the 1st policy improvement

Values and action for each state after 1 policy improvement

<div>0.00 TERM</div> <div>State 0 - TERM</div>	<div>-18.37 ←</div> <div>State 1</div>	<div>-26.19 ←</div> <div>State 2</div>	<div>-28.56 ↓</div> <div>State 3</div>
<div>-18.37 ↑</div> <div>State 4</div>	<div>-23.75 ↑</div> <div>State 5</div>	<div>-25.52 ↓</div> <div>State 6</div>	<div>-25.84 ↓</div> <div>State 7</div>
<div>-26.19 ↑</div> <div>State 8</div>	<div>-25.52 →</div> <div>State 9</div>	<div>-23.20 →</div> <div>State 10 - SPL</div>	<div>-17.14 ↓</div> <div>State 11</div>
<div>-28.56 →</div> <div>State 12</div>	<div>-25.84 →</div> <div>State 13</div>	<div>-17.14 →</div> <div>State 14</div>	<div>0.00 TERM</div> <div>State 15 - TERM</div>

Value and action of each state after the 2nd policy improvement

Values and action for each state after 2 policy improvement

0.00 TERM State 0 - TERM	0.00 ← State 1	-4.00 ← State 2	<div>Terminal States</div> <div>Special States</div> <div>-7.60 ↓ Regular States</div> <div>State 3</div>
0.00 ↑ State 4	-4.00 ↑ State 5	-4.60 ↓ State 6	-4.00 ↓ State 7
-4.00 ↑ State 8	-4.60 → State 9	-4.00 ↓ State 10 - SPL	0.00 ↓ State 11
-7.60 ↑ State 12	-4.00 → State 13	0.00 → State 14	0.00 TERM State 15 - TERM

Value and action of each state after the 3rd policy improvement

Values and action for each state after 3 policy improvement

0.00 TERM State 0 - TERM	0.00 ← State 1	-4.00 ← State 2	-7.60 ↓ State 3
0.00 ↑ State 4	-4.00 ↑ State 5	-4.60 ↓ State 6	-4.00 ↓ State 7
-4.00 ↑ State 8	-4.60 → State 9	-4.00 ↓ State 10 - SPL	0.00 ↓ State 11
-7.60 ↑ State 12	-4.00 → State 13	0.00 → State 14	0.00 TERM State 15 - TERM

Final Optimal policy

Final Policy after the algorithm is complete

0.00 TERM State 0 - TERM	0.00 ← State 1	-4.00 ← State 2	<div>Terminal States</div> <div>Special States</div> <div>-7.60 ↓ Regular States</div>
0.00 ↑ State 4	-4.00 ↑ State 5	-4.60 ↓ State 6	-4.00 ↓ State 7
-4.00 ↑ State 8	-4.60 → State 9	-4.00 ↓ State 10 - SPL	0.00 ↓ State 11
-7.60 ↑ State 12	-4.00 → State 13	0.00 → State 14	0.00 TERM State 15 - TERM

Comment: The states 1, 4, 11, 14 has the value 0 at the end. If the reward for going to the terminal state were to positive in the problem, these states would have some values other than 0. When the value is 0, the agent has no incentive to reach the goal. So, the decision to choose reward is important.

Problem 4 - Value Iteration

- Value iteration
- every step generates reward -4
- Goal state reward 0
- gamma $\gamma = 0.9$
- Special state reward -1
- Special state (2,2) --> state 10
- Goal state (0,0) and (3,3)
- Run value iteration and generate a policy based on the values

Here we are using value iteration. Value iteration has only one update. we directly update the value function using the Bellman optimality equation.

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

We repeat this update across all states until the value function converges (the difference between successive iterations is below some threshold ϵ)

Run `python assignment1_main.py --problem 4`

```
1 # Problem 4 sets the following options
2 # config.stepReward = -4
3 # config.goalReward = 0
4 # config.gamma = 0.9
5 # config.epsilon = 1e-6
6 # config.max_iterations = 200
7 # config.grid_size = 4
8 # config.valueFunctionInit = "V"
9 # config.randomValueFunctionInit = True
10 # config.uniformPolicyInit = True
11 # config.task = "value_iteration"
12 # config.plotTable = True
13 # config.goalStates = [(0, 0), (3, 3)]
14 # config.splStates = [(2,2)]
15 # config.splReward = -1
```

Running problem 4

```
1 INFO:__main__:Performing calculations for Prob 4: policy_iteration with
  V value function and uniform distribution for policy initialization
2 INFO:__main__:Using V value function with random initialization
3 INFO:__main__:Value iteration converged after 4 iterations
4 INFO:__main__:Value iteration converged successfully
5 INFO:__main__:Plotting value function
6 INFO:__main__:Plotting optimal policy with values and actions
```

Plot the comparison of value and optimal action of each state after the algorithm is completed (similar to the slides). One plot for value, one plot for policy.

Value after the algorithm is complete

Final Value Function after the algorithm is complete

0.000 State 0 - TERM	0.000 State 1	-4.00 State 2	0.000 State 3
0.000 State 4	-4.00 State 5	-4.60 State 6	-4.00 State 7
-4.00 State 8	-4.60 State 9	-4.00 State 10 - SPL	0.000 State 11
-7.60 State 12	-4.00 State 13	0.000 State 14	0.000 State 15 - TERM

Terminal States Special States Regular States

Policy after the algorithm is complete

Final optimal policy after the algorithm is complete

0.00 TERM State 0 - TERM	0.00 ← State 1	-4.00 ← State 2	<div>0.00 ← State 3</div> <div>Terminal States Special States -7.60 Regular States</div>
0.00 ↑ State 4	-4.00 ↑ State 5	-4.60 ↓ State 6	-4.00 ↓ State 7
-4.00 ↑ State 8	-4.60 → State 9	-4.00 ↓ State 10 - SPL	0.00 ↓ State 11
-7.60 ↑ State 12	-4.00 → State 13	0.00 → State 14	0.00 TERM State 15 - TERM