CSE 574: Introduction to Machine Learning, Section C Spring 2022

Assignment 4:
Defining and Solving Reinforcement Learning Task

Final on May 8th

Yvonne Huang (Yon-Chien Huang), Eric Chou (PoHan Chou)

ACADEMIC INTERGITY STATEMENT

"We certify that the code and data in this assignment were generated independently, using only the tools and resources defined in the course and that we did not receive any external help, coaching or contributions during the production of this work."

Part I: Define an RL Environment

Environment Description

- Theme:

  Grid World with light green states as positive rewards, purple states as the negative rewards, and yellow states as the terminal state. Giving penalty when taking any action when not receiving other reward.
- States:

  $\{S_1 = (0,0), S_2 = (0,1), S_3 = (0,2), S_4 = (0,3), S_5 = (1,0), S_6 = (1,1), S_7 = (1,2), S_8 = (1,3), S_9 = (2,0), S_{10} = (2,1), S_{11} = (2,2), S_{12} = (2,3), S_{13} = (3,0), S_{14} = (3,1), S_{15} = (3,2), S_{16} = (3,3)\}$
- Actions:

  $\{Left, Up, Right, Down\}$
- Reward:

	Left	Up	Right	Down
S1	-10	-10	+1	-1.5
S2	-10	-1.5	-1.5	-1.5
S3	-10	-1.5	-3	-1.5
S4	-10	-1.5	-1.5	-10
S5	-1.5	-10	-1.5	-1.5
S6	-1.5	+1	-1.5	-3
S7	-1.5	-1.5	-1.5	-1.5
S8	-1.5	-3	-1.5	-10
S9	+1	-10	-3	-1.5
S10	-1.5	-1.5	+1	-1.5
S11	-3	-1.5	-1.5	-1.5
S12	-1.5	-1.5	+10	-10
S13	-1.5	-10	-10	+1
S14	-1.5	-3	-10	-1.5
S15	-1.5	+1	-10	+10
S16	-1.5	-1.5	-10	-10

A. Get -9 when hitting the walls
B. Get -2 when arriving S_7 or S_{13}
C. Get +2 when arriving S_5 or S_{14}
D. Get +11 when arriving the goal S_{16}
E. Get -1.5 every time the agent moves when not receiving reward

- Objective:
  Reach the goal state S_{16} with maximum reward

Visualization

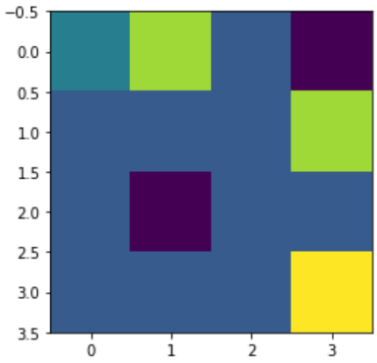
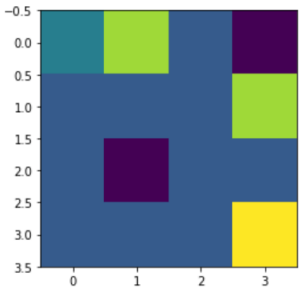
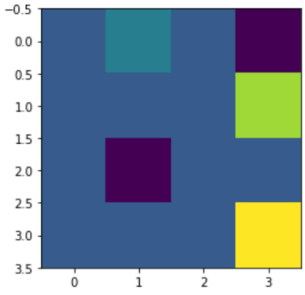
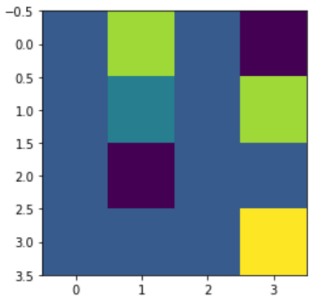
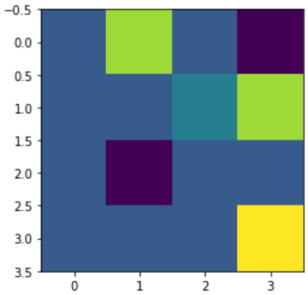


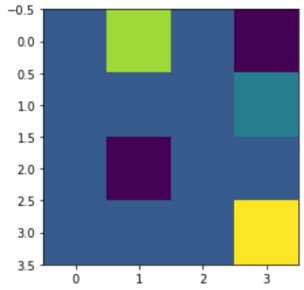
figure 1 environment visualization

figure 1 shows the environment in our RL model. The initial status of agent is $S_1(0,0)$. If agent hits the walls, it would be given a reward as -9. The Light color grids represent S_5 and S_{14}, agent receives reward as +11 when arriving these states. Also, when agent arrives purple grids S_7 and S_{14}, it gets -2. Finally, the agent receives -1 every time it moves.

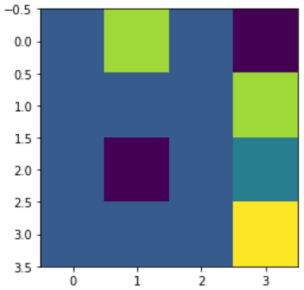
table 1 visualization of steps

This is step: 1 The action of this step: up The current state: [0, 0] The reward get by this step: -10 The total reward: -10	This is step: 2 The action of this step: right The current state: [0, 1] The reward get by this step: 1 The total reward: -9	This is step: 3 The action of this step: down The current state: [1, 1] The reward get by this step: -1 The total reward: -10	This is step: 4 The action of this step: right The current state: [1, 2] The reward get by this step: -1 The total reward: -11
			

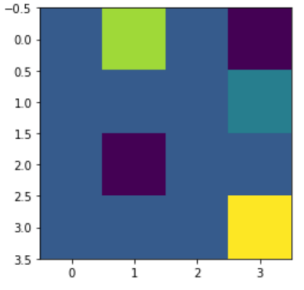
This is step: 5
The action of this step: right
The current state: [1, 3]
The reward get by this step: 1
The total reward: -10



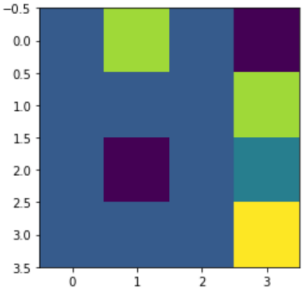
This is step: 6
The action of this step: down
The current state: [2, 3]
The reward get by this step: -1
The total reward: -11



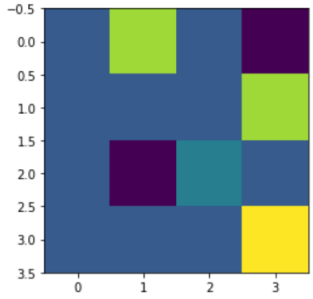
This is step: 7
The action of this step: up
The current state: [1, 3]
The reward get by this step: 1
The total reward: -10



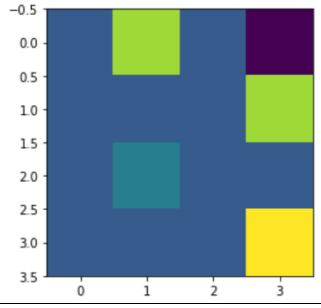
This is step: 8
The action of this step: down
The current state: [2, 3]
The reward get by this step: -1
The total reward: -11



This is step: 9
The action of this step: left
The current state: [2, 2]
The reward get by this step: -1
The total reward: -12



This is step: 10
The action of this step: left
The current state: [2, 1]
The reward get by this step: -3
The total reward: -15



Safety in AI

To prevent the agent from going out of boundaries, the agent is forced to stay in its original state when it decides to move outward. Also, when the decision of moving outward is made, the agent would receive a relatively large magnitude of negative reward, therefore, the agent would prefer to move within this 4 by 4 grid environment. In addition, we expected the agent to find the shortest path to reach the terminal state with maximum reward. Therefore, the agent receives -1 whenever it moves.

Part II:

Solve your environment using a tabular method – SARSA

SARSA result

```
[[ 4. 10.  9.  3.]
 [ 6. 10.  8. 10.]
 [ 2. 10.  1.  8.]
 [ 5.  9.  4.  4.]
 [ 8.  9.  9.  8.]
 [ 7.  3.  4.  3.]
 [ 9.  7.  1.  2.]
 [ 3. 10.  1.  5.]
 [ 1.  5.  8. 10.]
 [ 7.  7.  7. 10.]
 [ 8.  3.  6.  2.]
 [ 1.  3.  8.  4.]
 [ 5.  7.  5.  7.]
 [ 9.  7. 10.  6.]
 [ 9. 10.  7. 10.]
 [ 0.  0.  0.  0.]]
```

figure 2 Q matrix before SARSA

```
[[-6.47328249  0.77253274  0.98598076 -5.89388097]
 [-8.0915684   0.84253192  0.86373547 -0.61193323]
 [-4.02741928  2.31962477  1.67393134  1.54376979]
 [ 1.21188869  6.3625479   2.2812361   3.71864014]
 [ 0.93198442  1.60818374  1.65074055  0.88756854]
 [ 1.81734305  2.05845967  2.06441775  1.87563021]
 [ 2.60588671  2.90137318  4.43421673  1.72517754]
 [ 2.86971443  6.10274613 -2.33221979  3.95145368]
 [ 2.03397128  2.66827301  2.52164329  2.50617394]
 [ 4.73839341  4.76040359  4.32663827  3.07655543]
 [ 3.94192129  4.02944551  4.28568199  2.07763054]
 [ 3.01692538  9.4416349   2.06361272  3.77404299]
 [ 3.99233987  1.19500416  4.63142737  0.47691166]
 [ 5.7366547   3.59303149  6.89356165  4.78761373]
 [ 7.164273    5.58420183  9.80616754  7.04790821]
 [ 0.          0.          0.          0.        ]]
```

figure 3 Q matrix after SARSA

After applying SARSA, the Q matrix in figure 2 changed to figure 3 and provided the expected reward in each location. Each row represents the expected reward in each state while each column represents one direction (up, down, right, left) respectively.

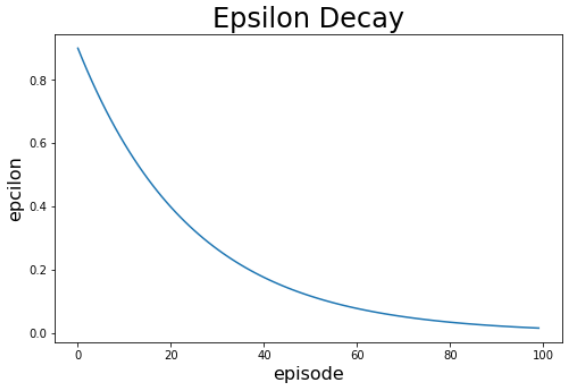


figure 4 epsilon decay in SARSA

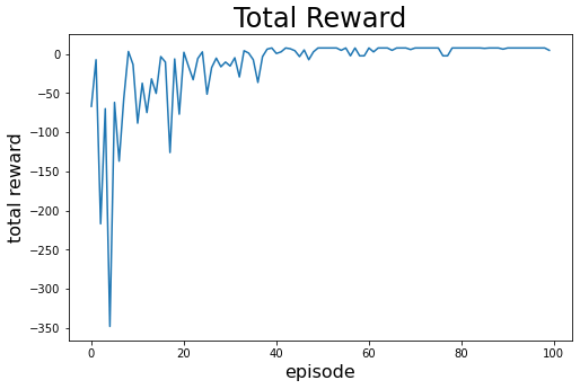


figure 5 total reward per episode

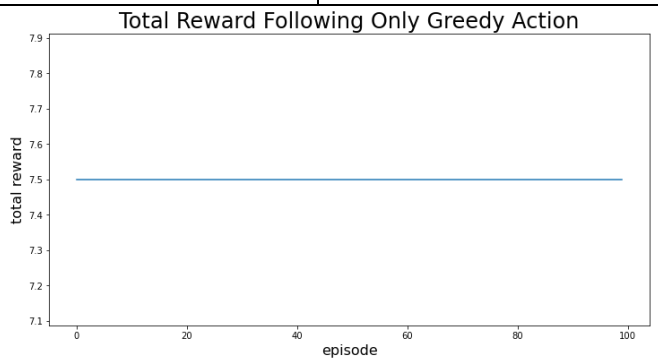


figure 6 only greedy action after SARSA

In the process of SARSA, the Q matrix was updated in each episode. At the beginning stage of SARSA process, the agent preferred to take random action the while episode was smaller, and the epsilon was greater. The epsilon decreased while episode increased, showed as figure 4. When the epsilon became smaller, the agent tended to take greedy action. Therefore, figure 5 shows that the total reward became more stable along the episode. After sufficient training, total reward shows the trend of constant in figure 6 because the agent only chooses greedy action, meaning that the route it chooses each time is the same.

SARSA and Q-Learning comparison

SARSA

SARSA is an on-policy update method, it updates the Q matrix by following ϵ-greedy policy. The agent is allowed to explore the enviroment while advoiding dangerous routes.

$$update\ function: Q(S,A) \leftarrow \mathrm{Q(S,A)} + \alpha[\mathrm{R} + \gamma \mathrm{Q}(\mathrm{S}', \mathrm{A}') - \mathrm{Q(S,A)}]$$

Q-Learning

Q-learning is a off-policy update method, it uses the maximum Q in all actions to update. There is no exploration in Q-learning.

$$update\ function: Q(S,A) \leftarrow \mathrm{Q(S,A)} + \alpha[\mathrm{R} + \gamma \max_{a} Q(S', a) \ - \mathrm{Q(S,A)}]$$

$S: state;\ A: action;\ \ Q: Q\ matrix;\ \ R: immediate\ reward;\ \ \alpha: learning\ rate;\ \ \gamma: discount\ rate$

Tuning hyperparameters

table 2 tuning min epsilon

	Setup 1	Setup 2	Setup 3
Epsilon	0.9	0.9	0.9
Epsilon decay	0.96	0.96	0.96
Alpha	0.1	0.1	0.1
Gamma	0.9	0.9	0.9
Episode	100	100	100
Max timestep	200	200	200
Min epsilon	0.3	0.05	0.001

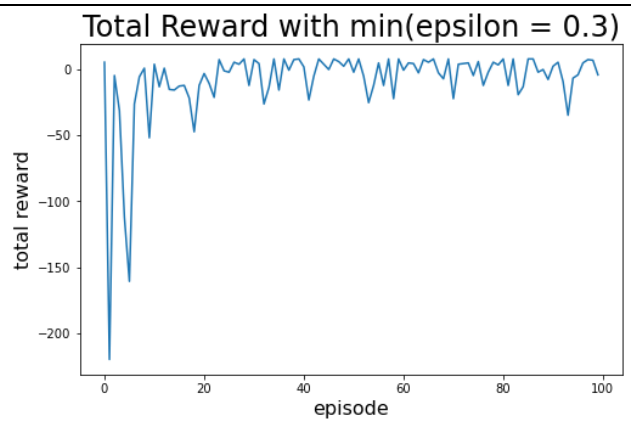


figure 7 total reward with min epsilon 0.3

figure 8 epsilon decay with min epsilon 0.3

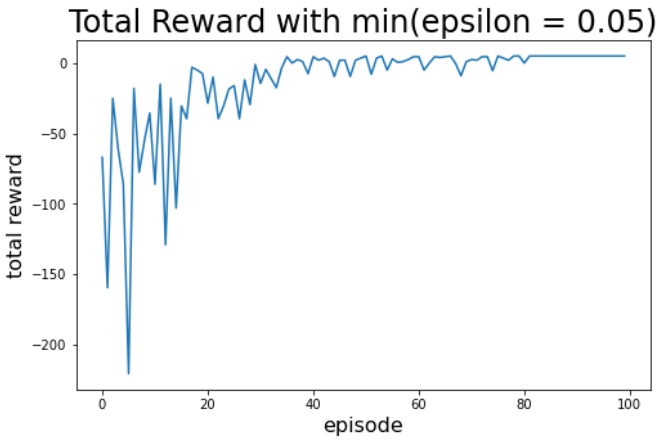


figure 9 total reward with min epsilon 0.05

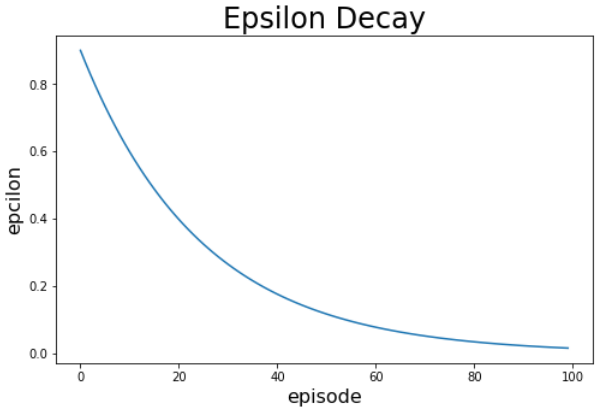


figure 10 epsilon decay with min epsilon 0.05

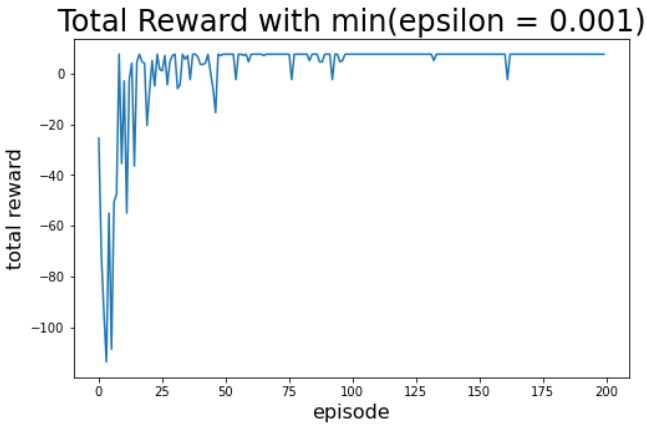


figure 11 total reward with min epsilon 0.001

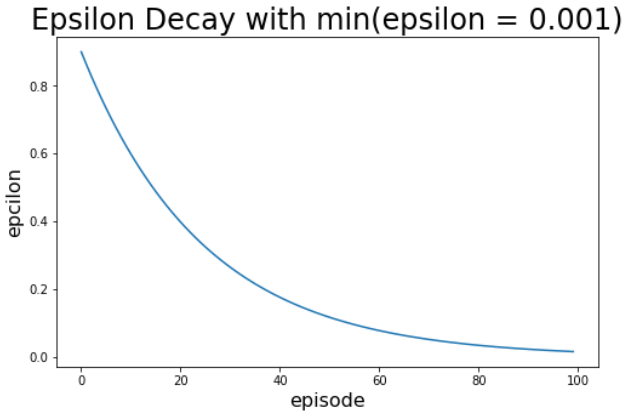


figure 12 epsilon decay with min epsilon 0.001

figure 8, figure 10, figure 12 show the epsilon decay plot when minimum of epsilon is set as 0.3,0.05, and 0.001 respectively. According to figure 7, when the minimum of epsilon is set as 0.5, the total reward is more unstable. figure 9 and figure 11 show the total reward become more stable along the episode. However, when the minimum of epsilon is too small, the total reward shows more trends of constant while episode increase, represents that the agent is not allowed to explore the environment. In our opinion, the minimum of epsilon should be set as 0.5.

table 3 tuning alpha

	Setup 1	Setup 2	Setup 3
Epsilon	0.9	0.9	0.9
Epsilon decay	0.96	0.96	0.96
Alpha	0.8	0.1	0.02
Gamma	0.9	0.9	0.9
Episode	100	100	100
Max timestep	200	200	200
Min epsilon	0.05	0.05	0.05

figure 13 total reward with alpha 0.8

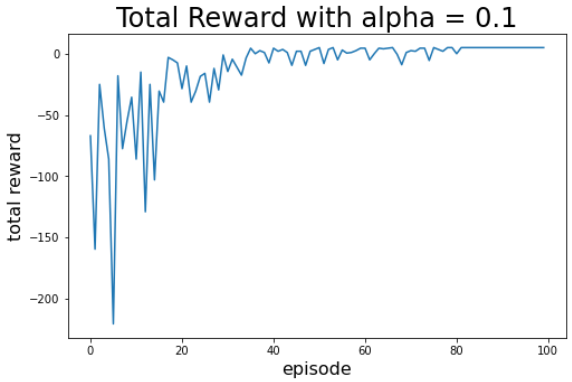


figure 14 total reward with alpha 0.1

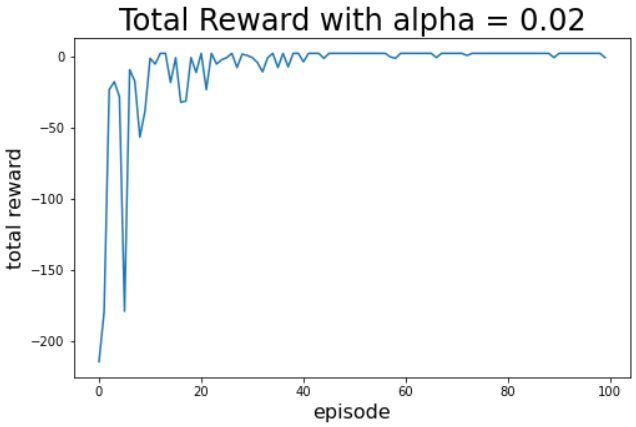


figure 15 total reward with alpha 0.02

When episode is small, the agent is allowed to explore the environment, meaning that the Q matrix is updated in each episode. figure 13 represents that when α is too large, the Q is adjusted aggressively, which might cause the result not to converge. In this plot, the total reward is unstable when episode is small. figure 15 shows the result of small learning rate, according to this plot, the total reward does not change significantly comparing to figure 14. When learning rate is small, it may take more time to converge, or it may only find the local minimum. Therefore, we think the best hyperparameter setting is 0.1.

Bonus points:

Double Q-learning

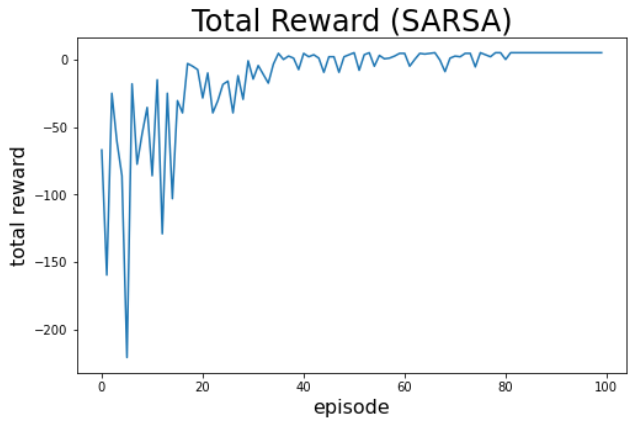


figure 16 total reward with SARSA

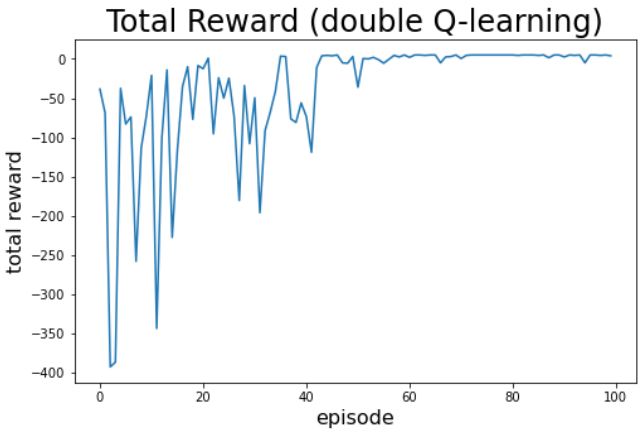


figure 17 total reward with double Q-learning

Double Q-learning is used to prevent the results from overestimating, but it may underestimate the action values at times. Comparing to double Q-learning, SARSA only has one Q matrix to make decision and back propagation, which makes the learning speed faster. In double Q-learning, the second Q matrix helps to increase the stability of total reward along episodes, but it also leads to the slower calculating speed.

Contribution

Team member	Assignment part	Contribution (%)
Yon-Chien Huang	Part 1, part 2, bonus reports Part 2, bonus code	50%
Po Han Chou	Bonus report Part 1, part2, bonus code	50%