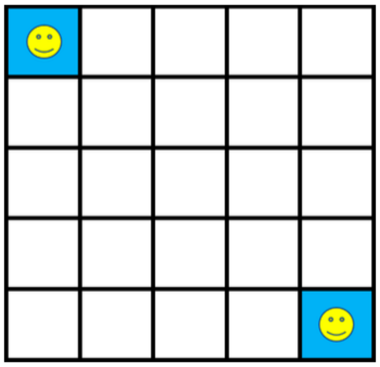
Name: Chaoran Li
Student ID: (UTD ID) 2021489307; (NET ID) cxl190012
Course Number: CS 6364.002

Homework 6 Writeup Part

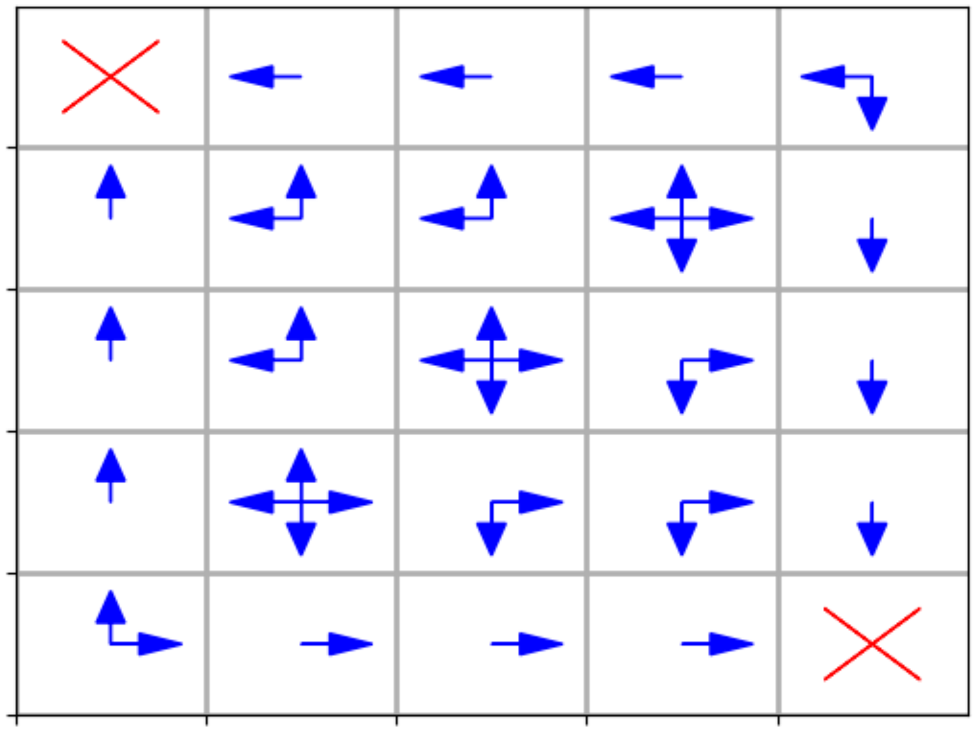
A Markov decision Process is defined by (S, A, R, P, γ), where S denotes the set of possible states, A denotes the set of possible actions, R denotes distribution of reward given (state, action) pair, P denotes transition probability, and γ is a discount factor. In this homework, given a simple 5 × 5 grid game (see below), the upper left (i.e., state 1-1) and lower right (i.e., state 5-5) corners are terminal states. Therefore, there are 5×5 = 25 states (i.e., |S| = 25) and each is denoted as s(i, j) where i and j represent i-th row and j-th column, respectively. Four possible actions are {right,left,up,down}. We set a negative reward r = −5 for each transition, a discount factor γ = 0.7, and the probability of an initial state p(s0) equals 1. Our goal is to reach one of the terminal states (smiling states) 25 in least number of actions. In other words, it aims to find the optimized policy π∗ that maximized cumulative discounted reward. The initial Q table can be randomly defined by you.



Q1: [Value iteration algorithm] Using the Bellman’s equation, implement value iteration algorithm to iteratively update the Q table until convergence.
After 6 rounds, the Q table converges:

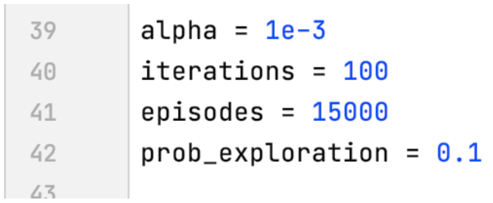
```
Round 6:
[[  0.        0.        0.        0.     ]
 [-10.95    -5.       -8.5     -10.95   ]
 [-12.665   -8.5     -10.95    -12.665  ]
 [-13.8655 -10.95    -12.665   -13.8655]
 [-13.8655 -12.665   -13.8655 -12.665  ]
 [-10.95    -8.5      -5.      -10.95   ]
 [-12.665   -8.5      -8.5     -12.665  ]
 [-13.8655 -10.95    -10.95    -13.8655]
 [-12.665  -12.665   -12.665   -12.665  ]
 [-12.665  -13.8655 -13.8655 -10.95   ]
 [-12.665  -10.95     -8.5     -12.665  ]
 [-13.8655 -10.95    -10.95    -13.8655]
 [-12.665  -12.665   -12.665   -12.665  ]
 [-10.95   -13.8655 -13.8655 -10.95   ]
 [-10.95   -12.665   -12.665    -8.5    ]
 [-13.8655 -12.665   -10.95    -13.8655]
 [-12.665  -12.665   -12.665   -12.665  ]
 [-10.95   -13.8655 -13.8655 -10.95   ]
 [ -8.5    -12.665   -12.665    -8.5    ]
 [ -8.5    -10.95    -10.95     -5.     ]
 [-12.665  -13.8655 -12.665   -13.8655]
 [-10.95   -13.8655 -13.8655 -12.665  ]
 [ -8.5    -12.665   -12.665   -10.95   ]
 [ -5.     -10.95    -10.95     -8.5    ]
 [  0.        0.        0.        0.     ]]
End after 6 rounds.

Process finished with exit code 0
```



Q2: [Deep Q-learning] This question functions the same as question 1. The only difference is to replace with a simple four-layer (i.e., three hidden layers and one output layer) fully-connected deep neural network to approximate the Q table. The three hidden layers contain 32, 8, and 16 neurons respectively and all applied ReLU activation functions. The output function is linear.

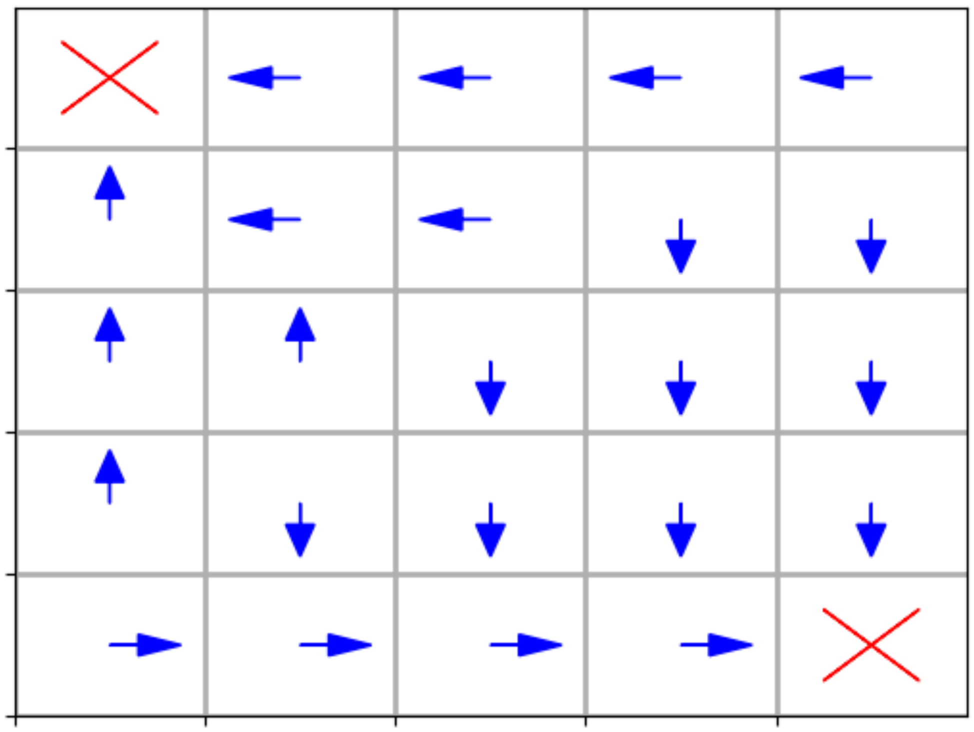
Use RMSProp as optimizer and MSELoss as loss function.
Parameters:



```
alpha = 1e-3
iterations = 100
episodes = 15000
prob_exploration = 0.1
```

After convergence, I print the path for each position:

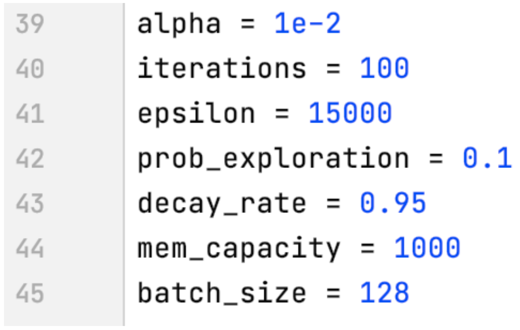
```
Evaluating...
Paths:
[0]
[1, 0]
[2, 1, 0]
[3, 2, 1, 0]
[4, 3, 2, 1, 0]
[5, 0]
[6, 5, 0]
[7, 6, 5, 0]
[8, 13, 18, 23, 24]
[9, 14, 19, 24]
[10, 5, 0]
[11, 6, 5, 0]
[12, 17, 22, 23, 24]
[13, 18, 23, 24]
[14, 19, 24]
[15, 10, 5, 0]
[16, 21, 22, 23, 24]
[17, 22, 23, 24]
[18, 23, 24]
[19, 24]
[20, 21, 22, 23, 24]
[21, 22, 23, 24]
[22, 23, 24]
[23, 24]
[24]
```



Compared with Q1, the result has only one solution for on position. Maybe some optimization should be implemented to my algorithm.

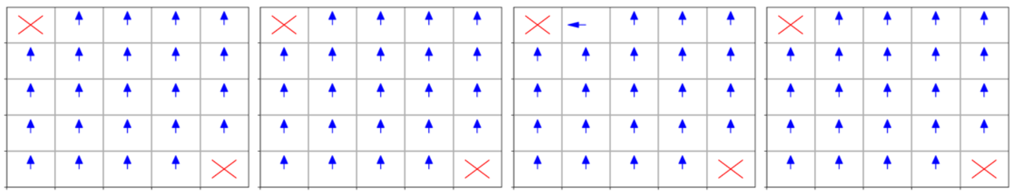
Q3: [Deep Q-learning with experience replay, bonus question] This question functions the same as question 2, except that you need to implement the Deep Q-learning with experience replay. Note that if you finish this bonus question, the points that you obtained in this HW will be doubled.

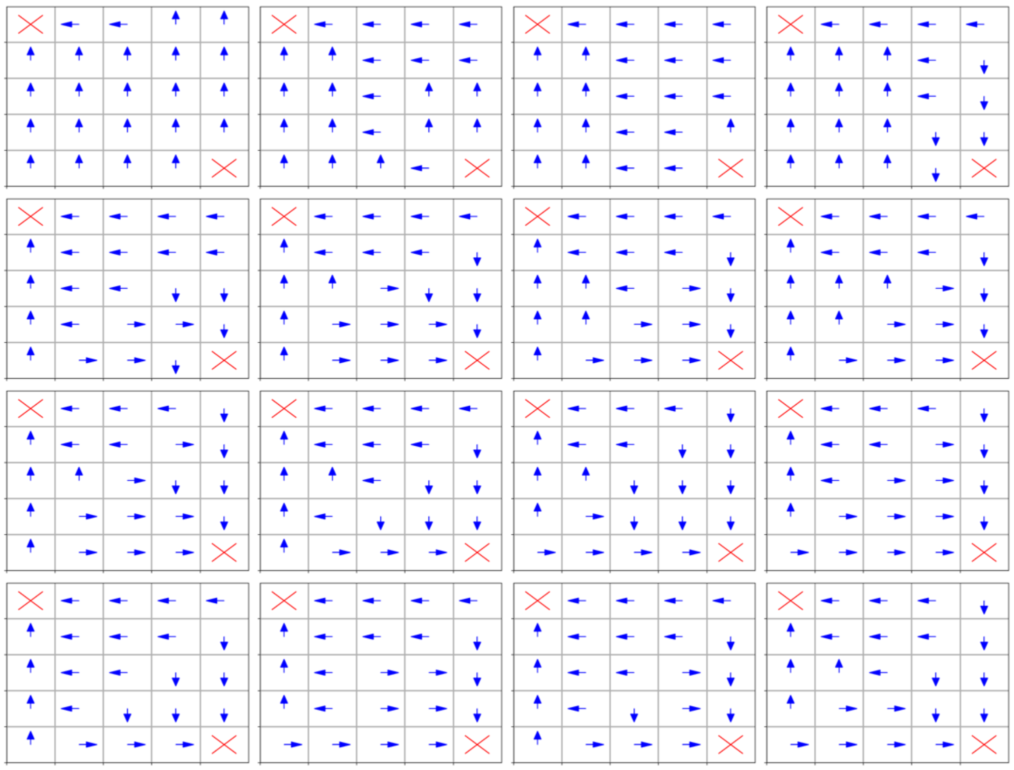
Use RMSProp as optimizer and MSELoss as loss function.
Parameters:



```
alpha = 1e-2
iterations = 100
epsilon = 15000
prob_exploration = 0.1
decay_rate = 0.95
mem_capacity = 1000
batch_size = 128
```

I also print the result for 20 rounds:





After convergence, I print the path for each position:

```
Evaluating...
Paths:
[0]
[1, 0]
[2, 1, 0]
[3, 2, 1, 0]
[4, 9, 14, 19, 24]
[5, 0]
[6, 5, 0]
[7, 6, 5, 0]
[8, 7, 6, 5, 0]
[9, 14, 19, 24]
[10, 5, 0]
[11, 6, 5, 0]
[12, 11, 6, 5, 0]
[13, 18, 23, 24]
[14, 19, 24]
[15, 10, 5, 0]
[16, 17, 18, 23, 24]
[17, 18, 23, 24]
[18, 23, 24]
[19, 24]
[20, 21, 22, 23, 24]
[21, 22, 23, 24]
[22, 23, 24]
[23, 24]
[24]
```