In this homework, I implement a Q-learning algorithm to solve the MountainCar-v0 game.

I tried two experiments, one is `how learning rate change affect the results`, and the other is `how state number of q-matrix affect the results`.

And I provided three diagrams to illustrate the rusults, which are `fail times over training process` (fail meant the car didn't catch the flag), `rewards over traing process` and `step number in each episode over traing process`.

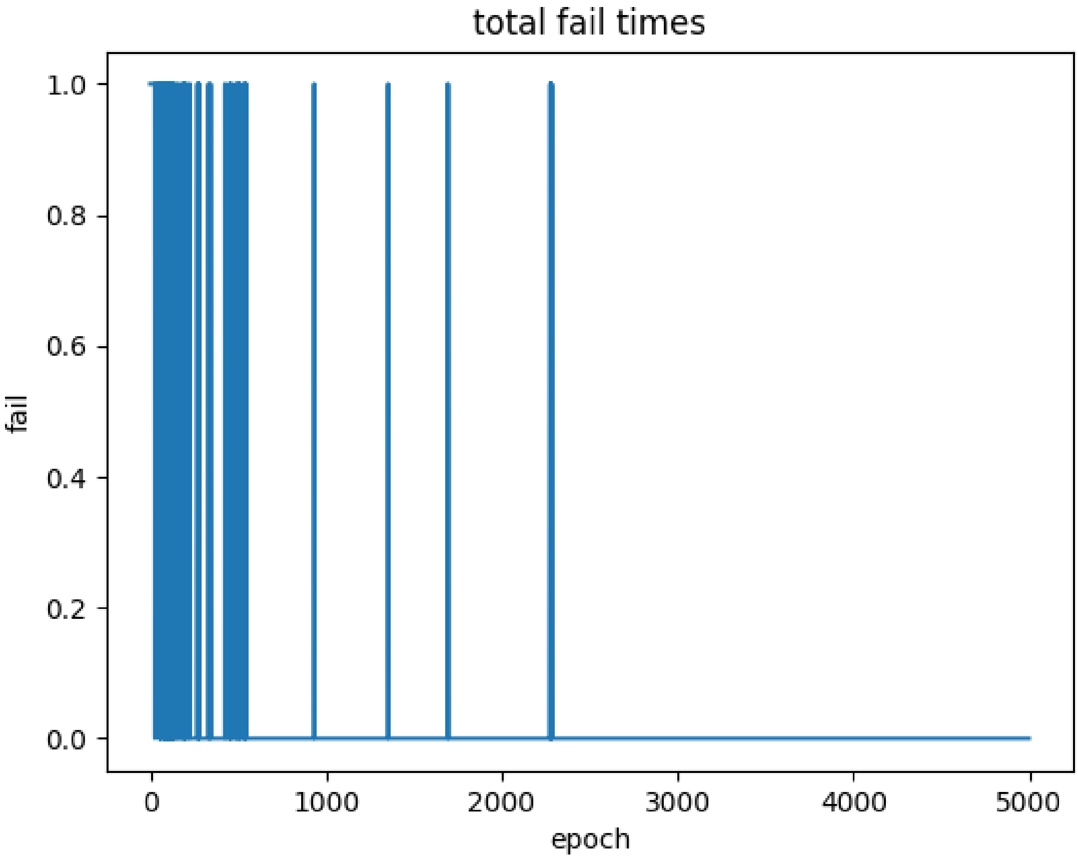
Result and Analysis

the basic environments is learning rate in 0.01, state number is 15, maximum step number is 1500 and training episode is 5000.

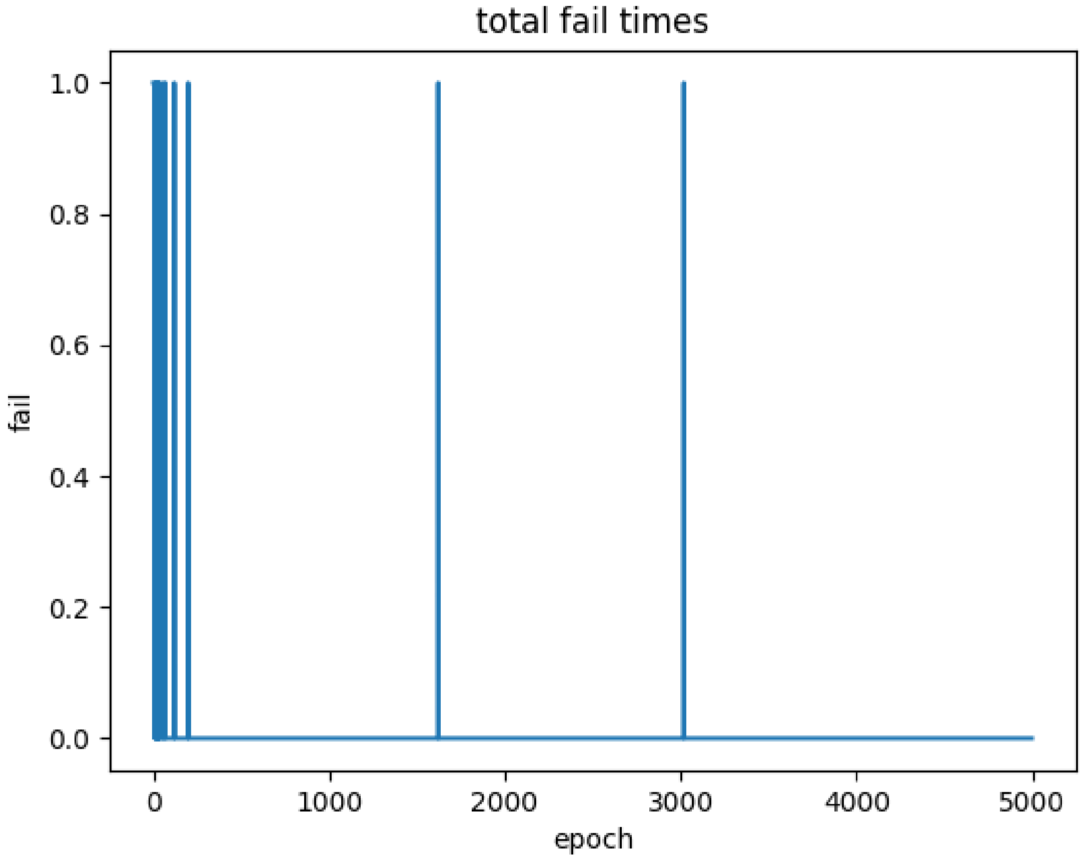
with different learning rate

fail times

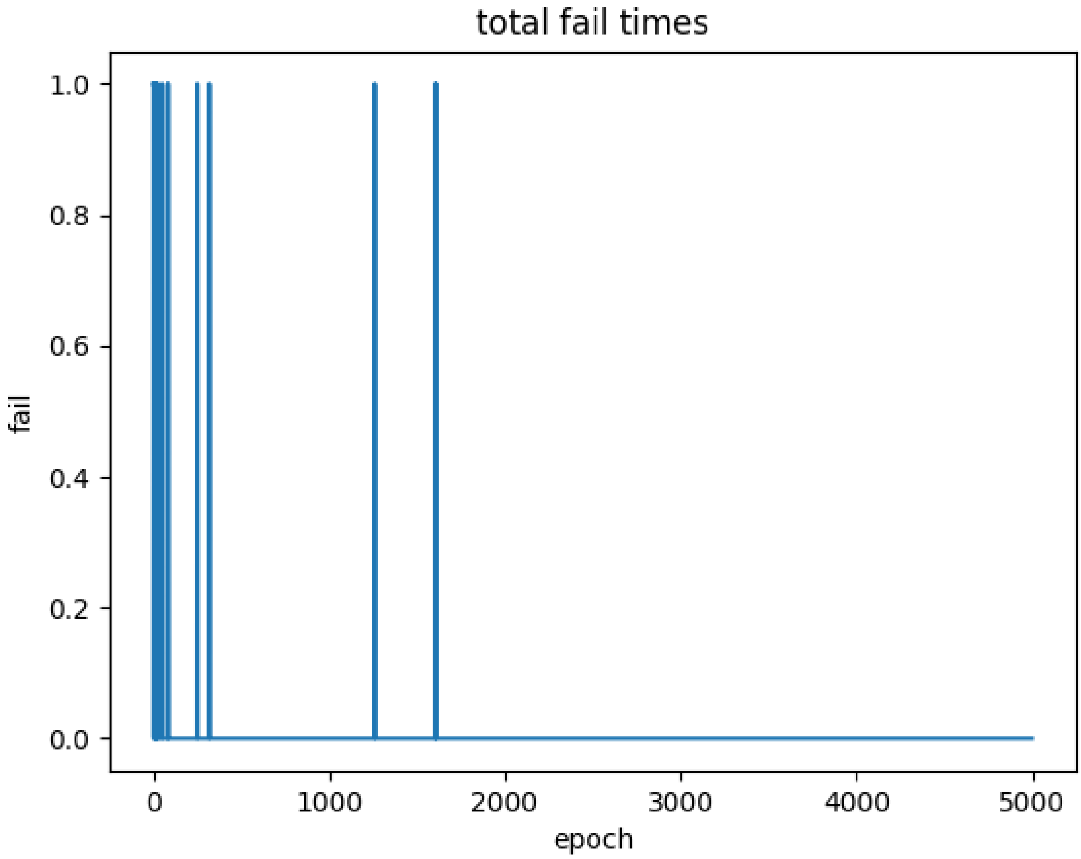
- learning rate = 0.01



- learning rate = 0.05



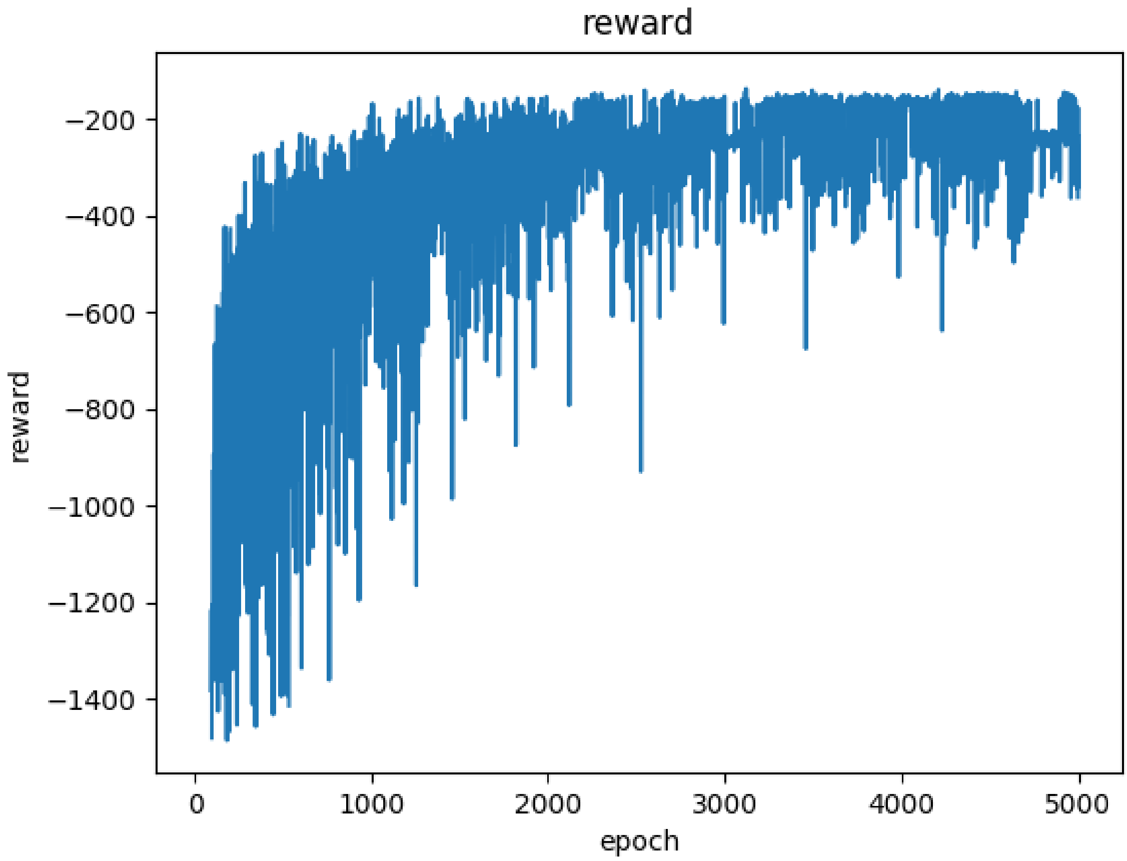
- learning rate = 0.1



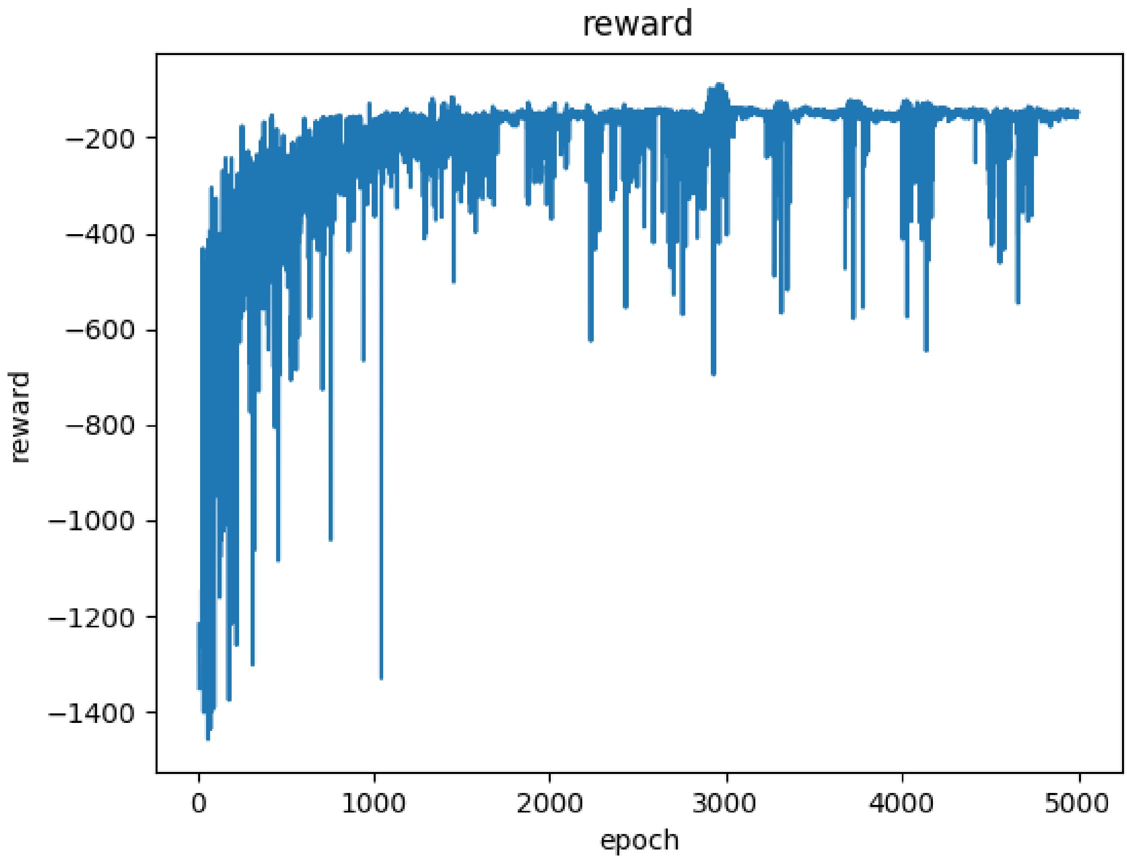
as you can see, the fail times as first reduce as learning rate increase, but with a long training period they are no different.

reward

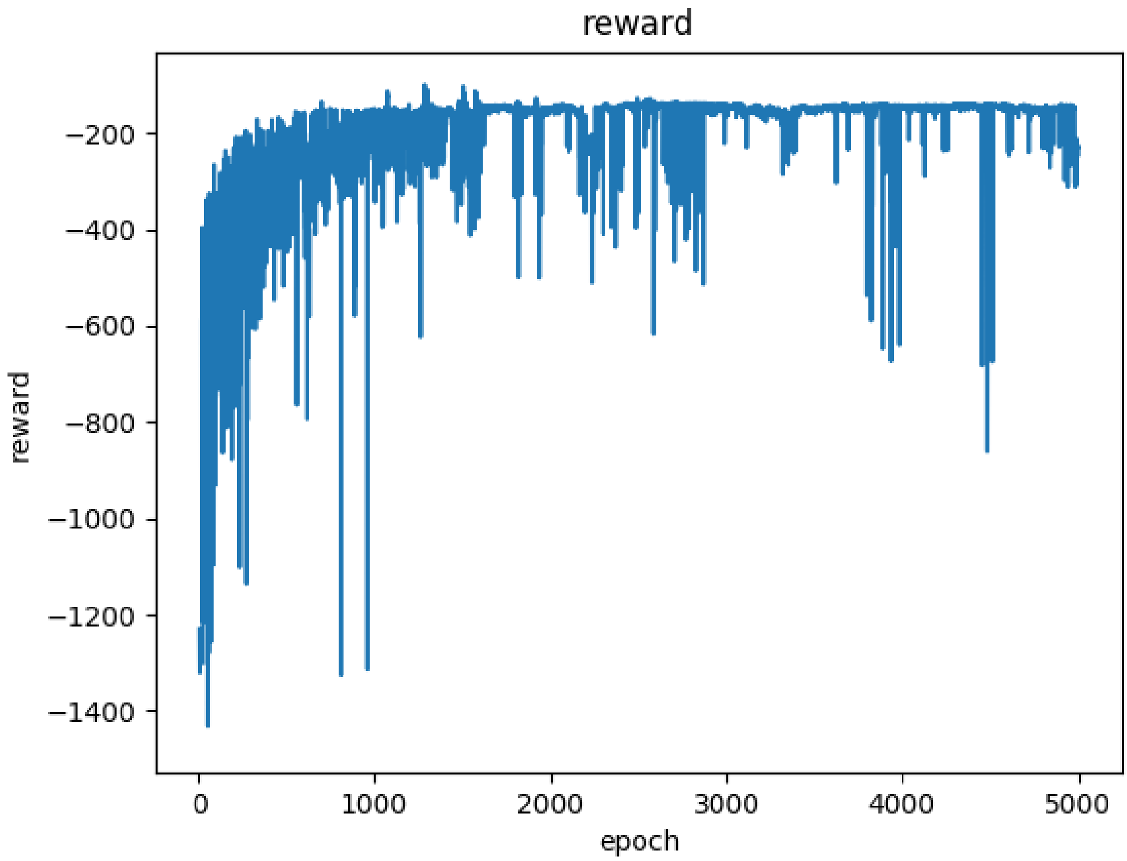
- learning rate = 0.01



- learning rate = 0.05



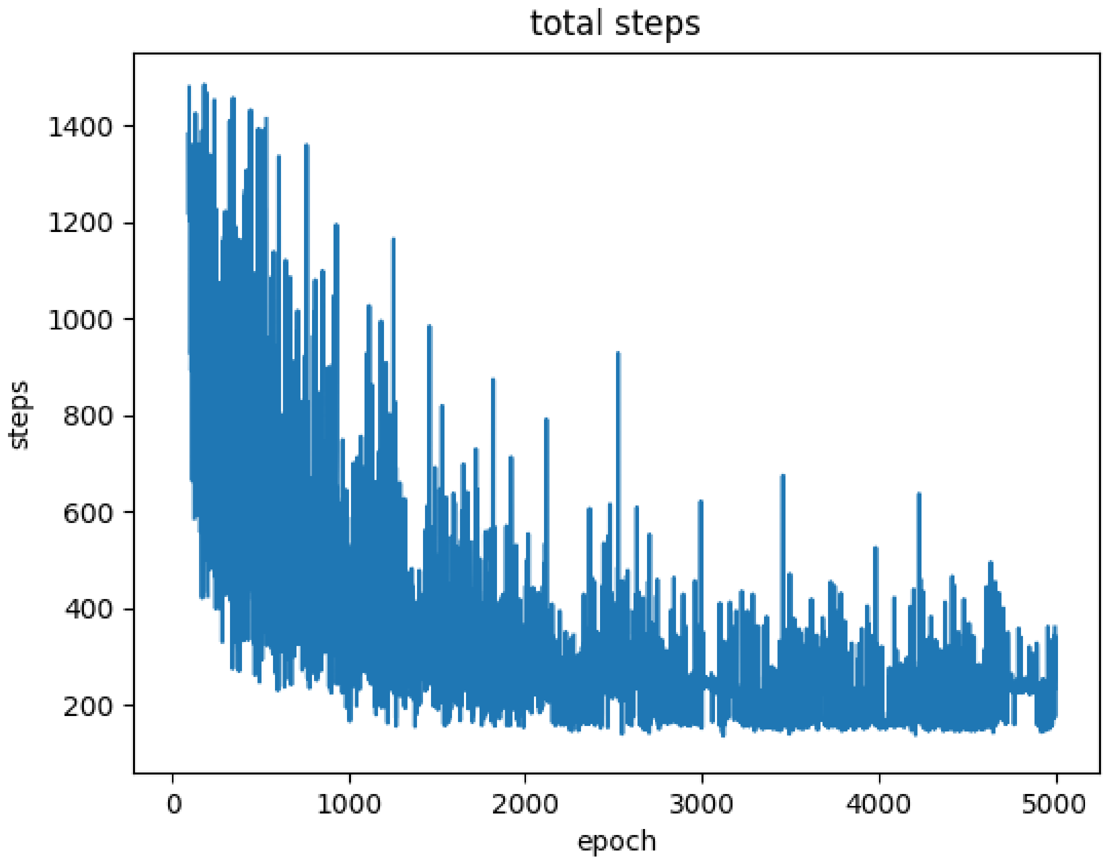
- learning rate = 0.1



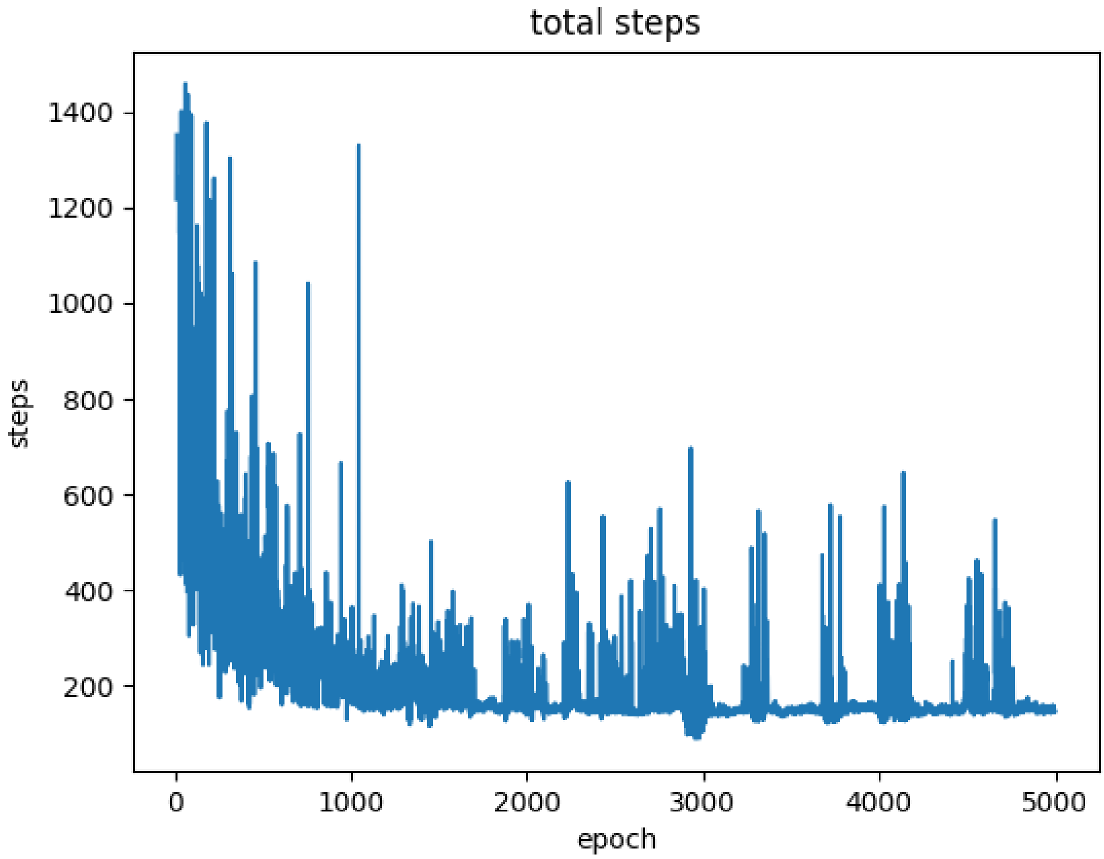
as you can see, the reward reduce as learning rate increase, and seems learning rate = 0.05 would be a magic number of this game.

step number

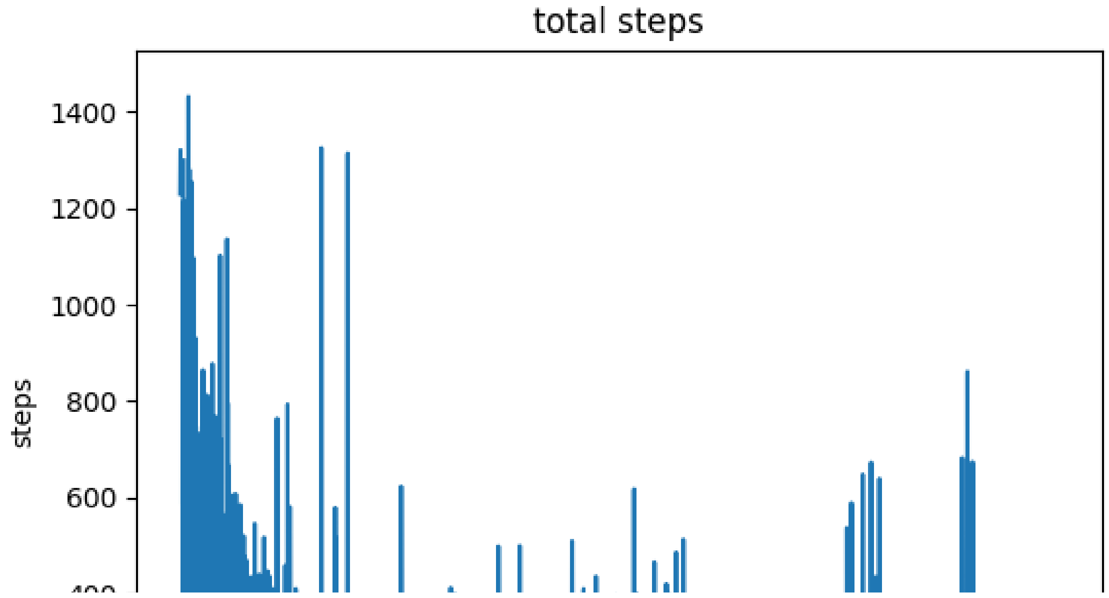
- learning rate = 0.01



- learning rate = 0.05



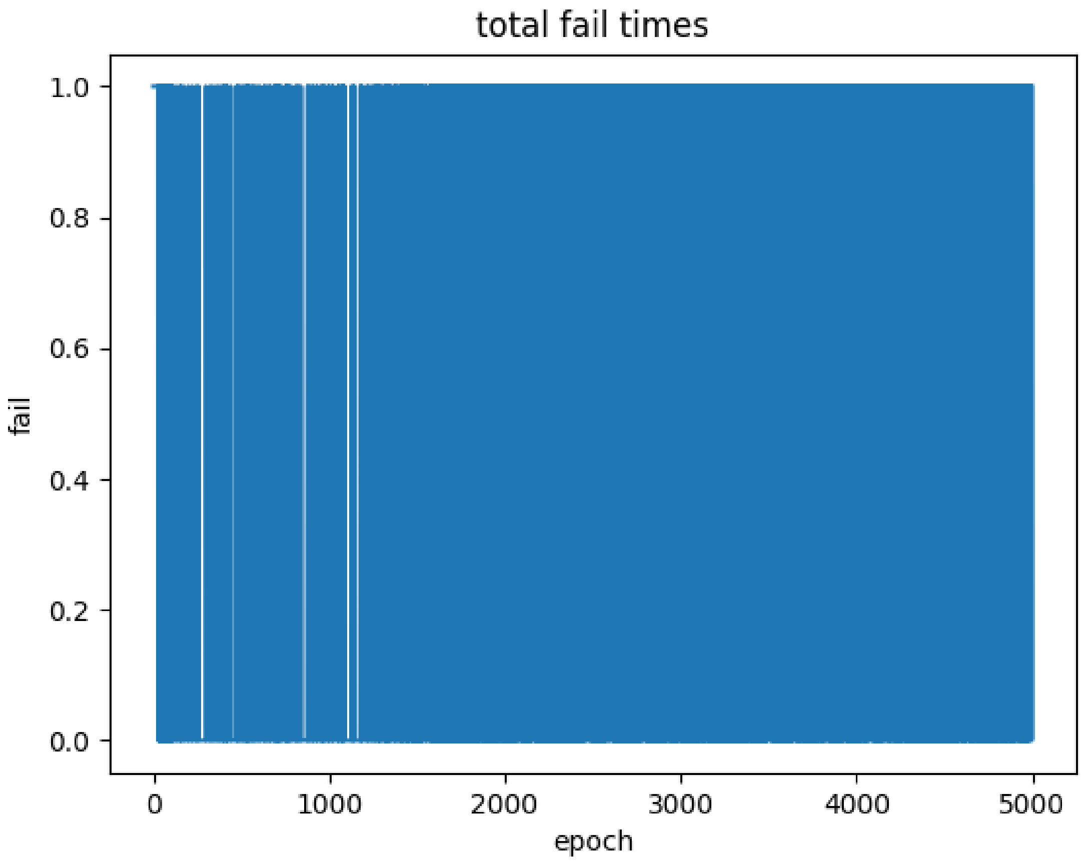
- learning rate = 0.1


total steps
steps
1400
1200
1000
800
600
400

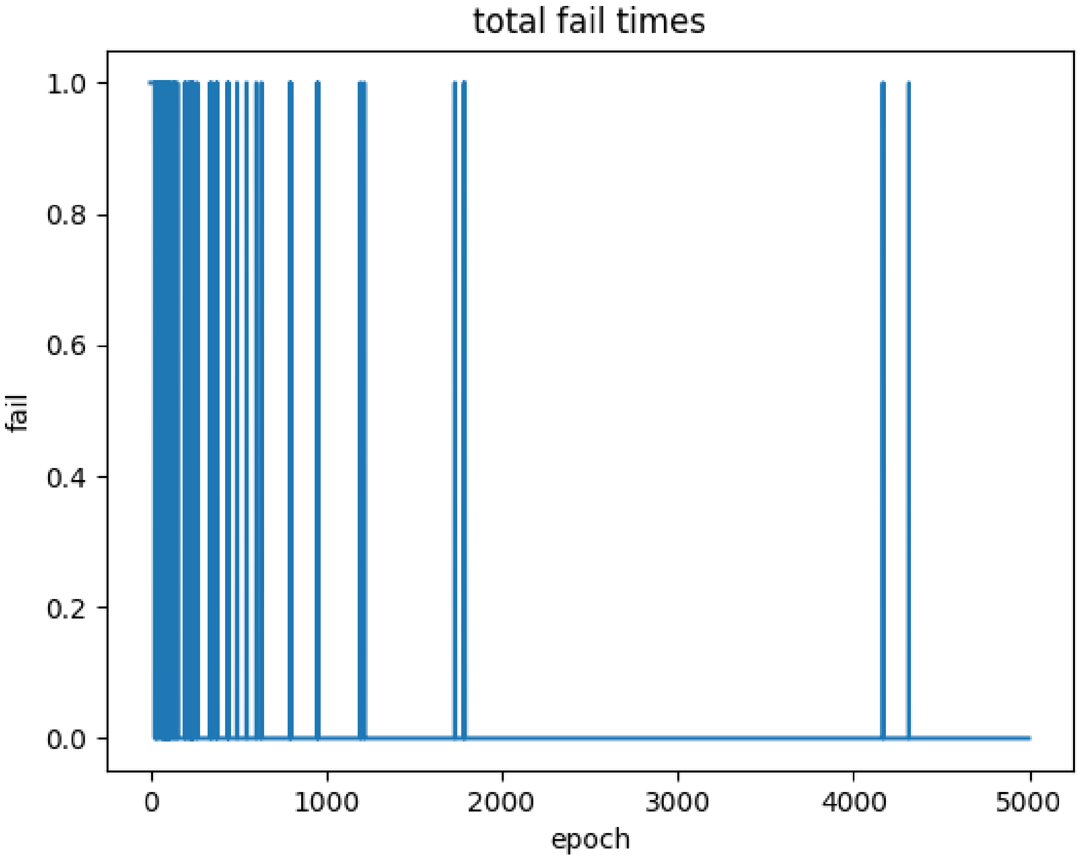
with different state number of q-matrix

fail times

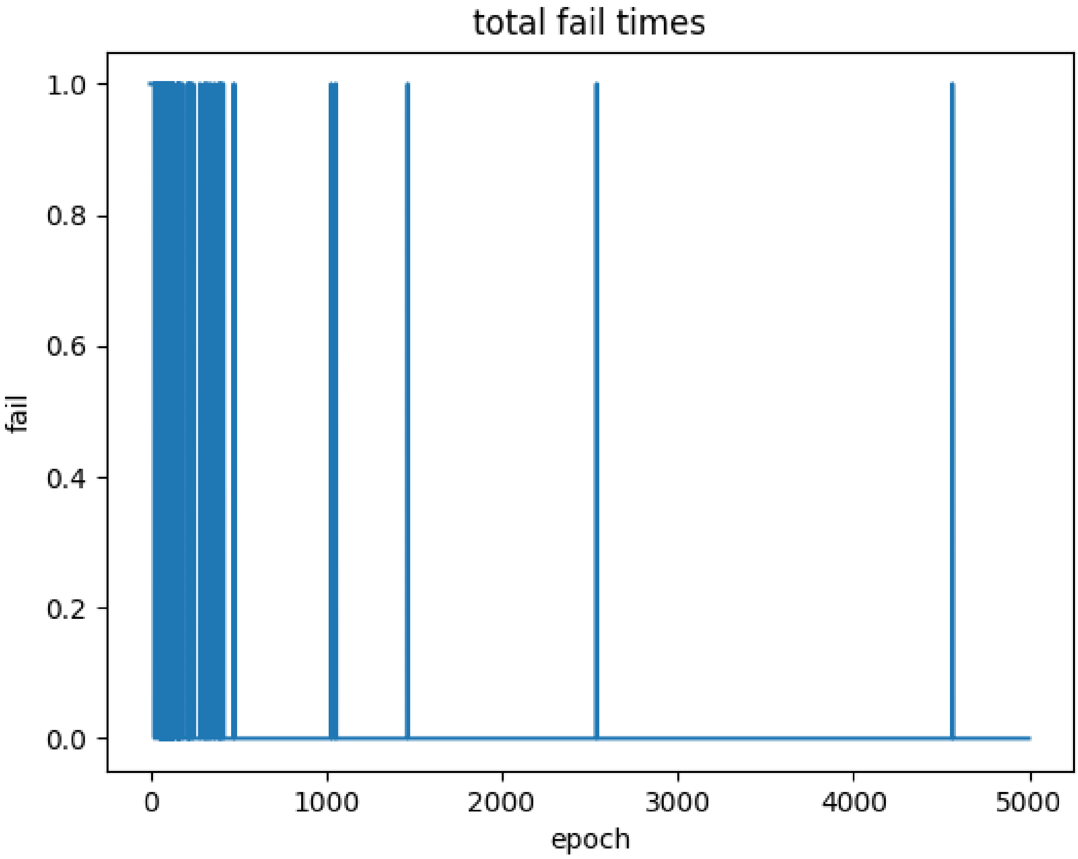
- state number = 5 * 5



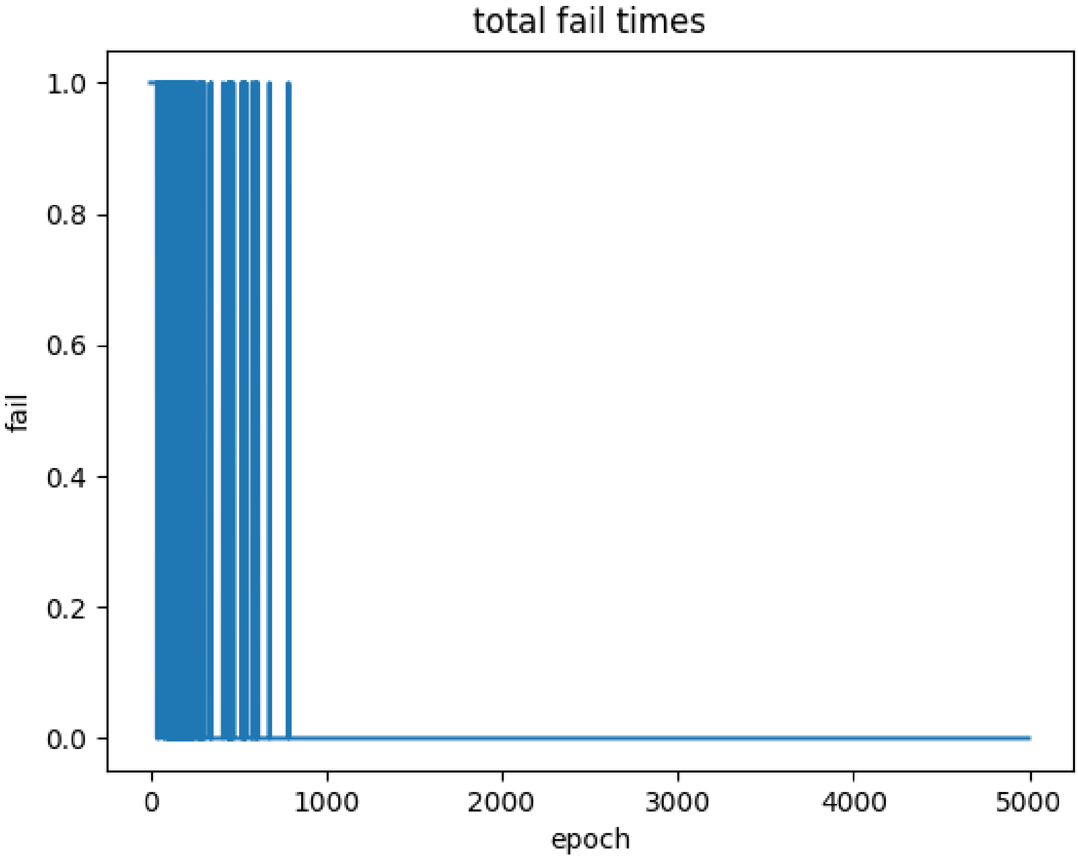
- learning rate = 10 * 10



- learning rate = 15 * 15



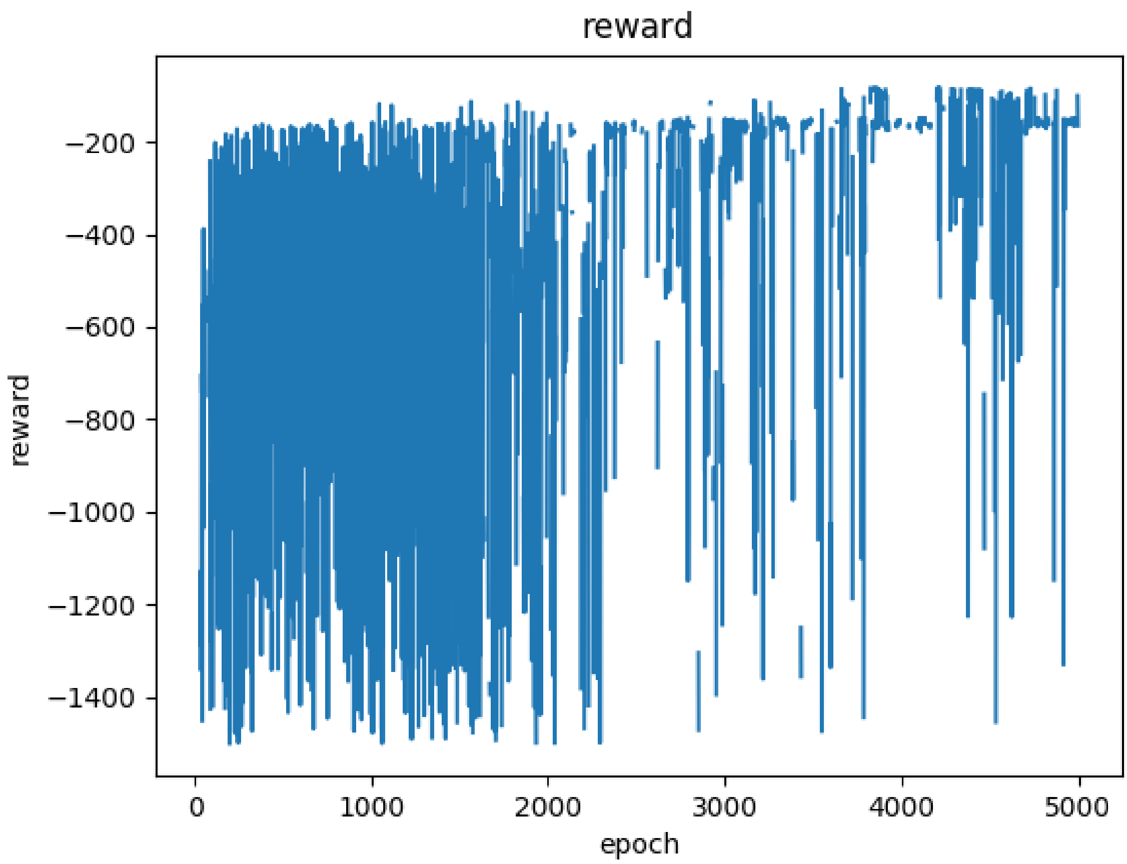
- learning rate = 20 * 20



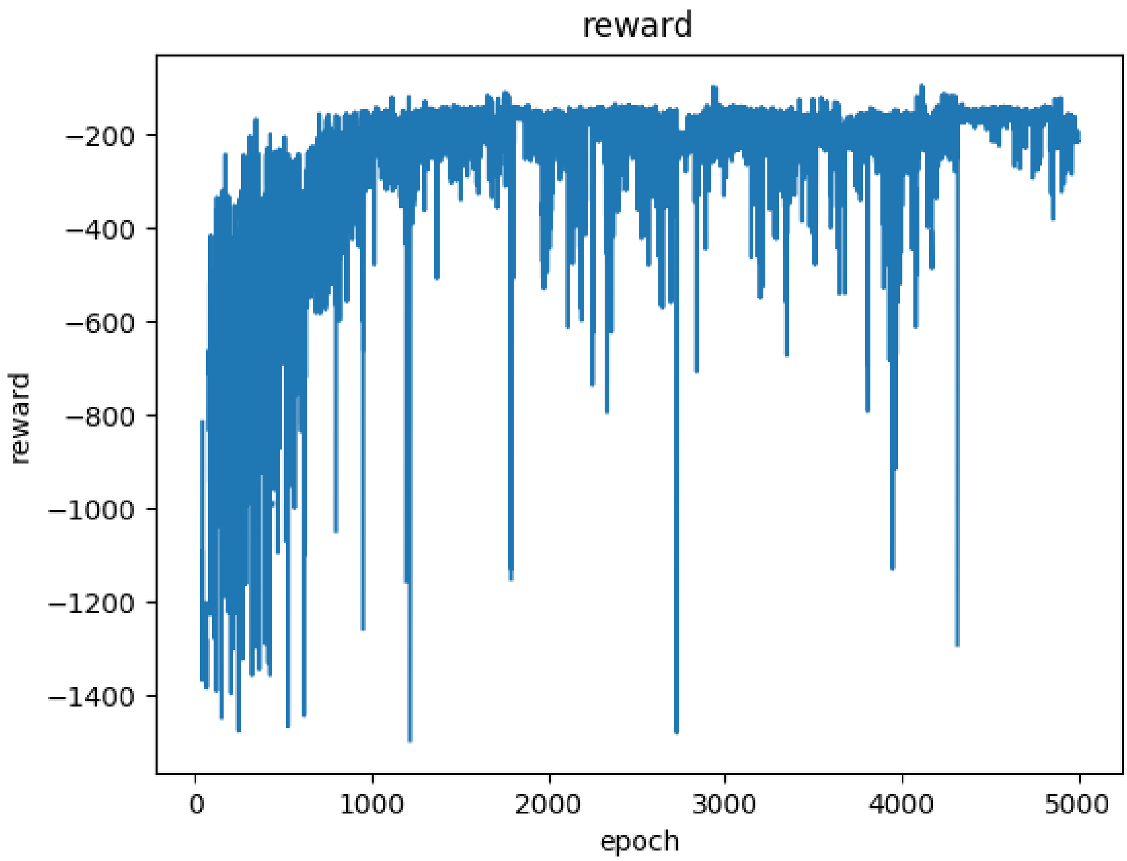
the state number of 5 5 *q-matrix was not working, and 20* 20 can perfore well relatively.

reward

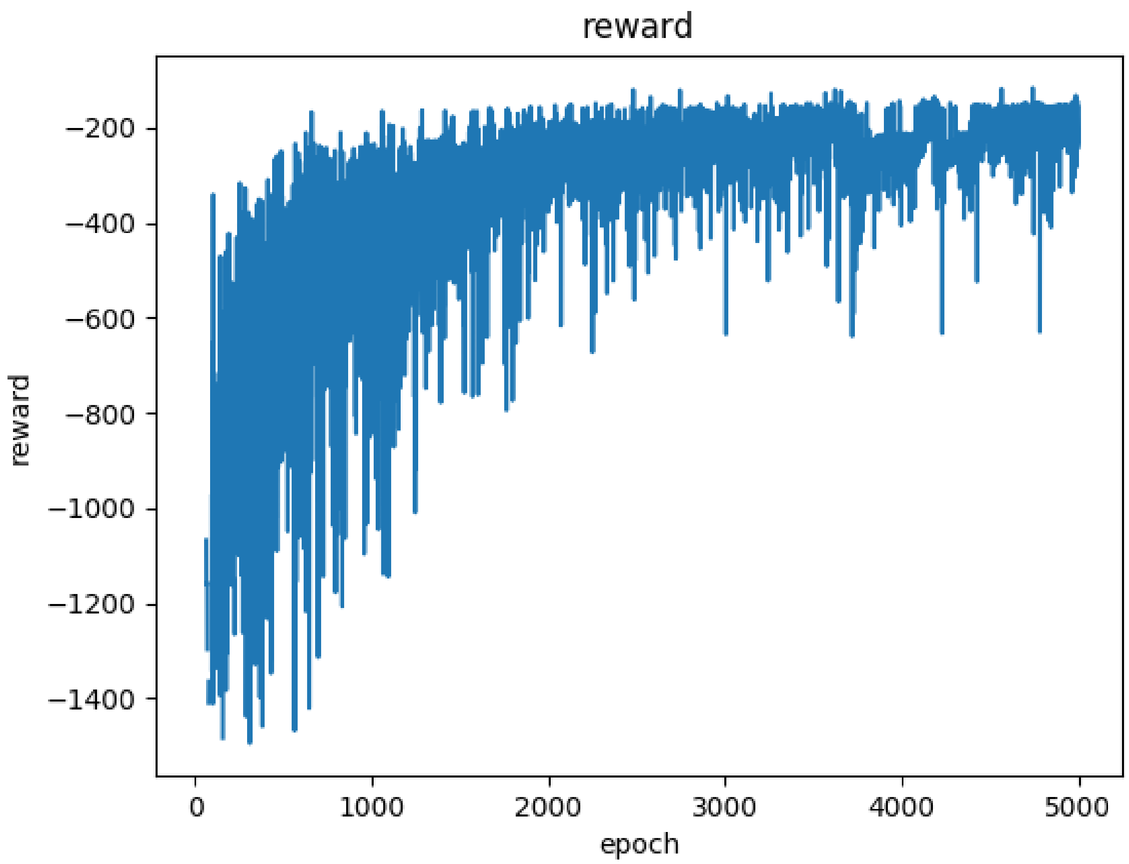
- state number = 5 * 5



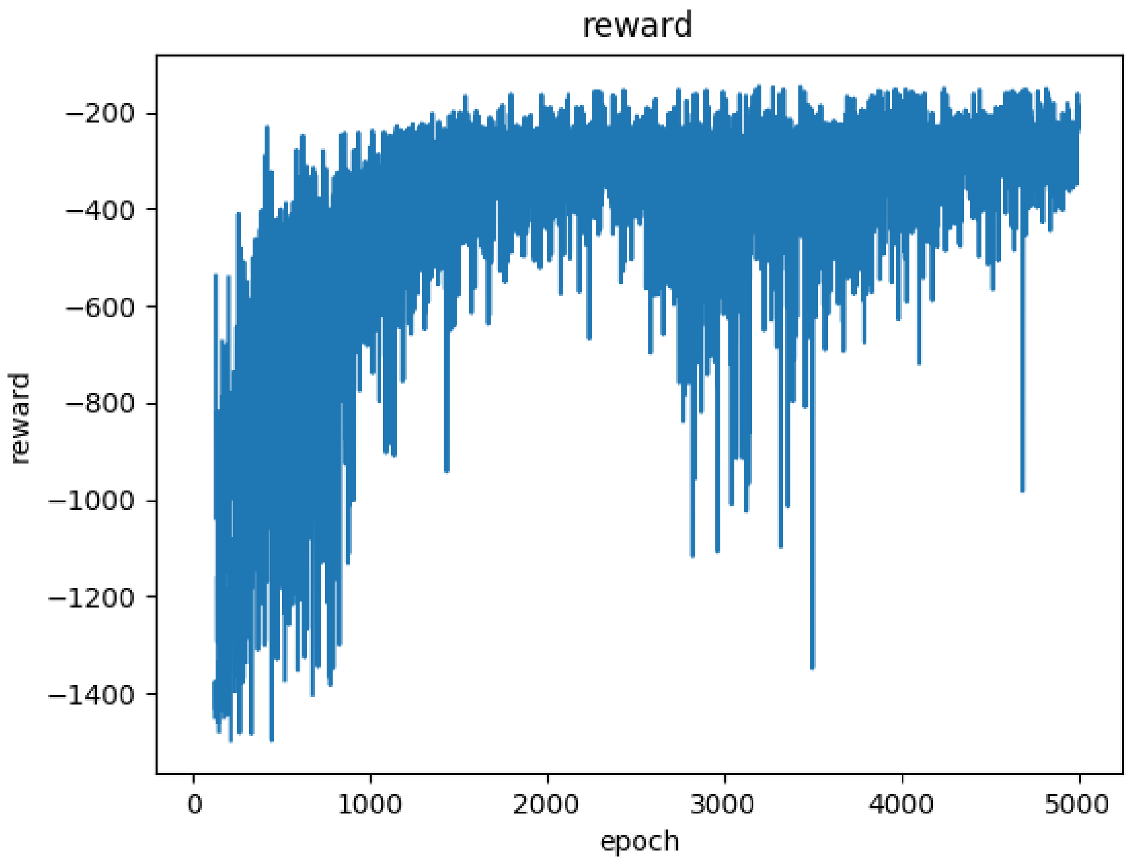
- learning rate = 10 * 10



- learning rate = 15 * 15



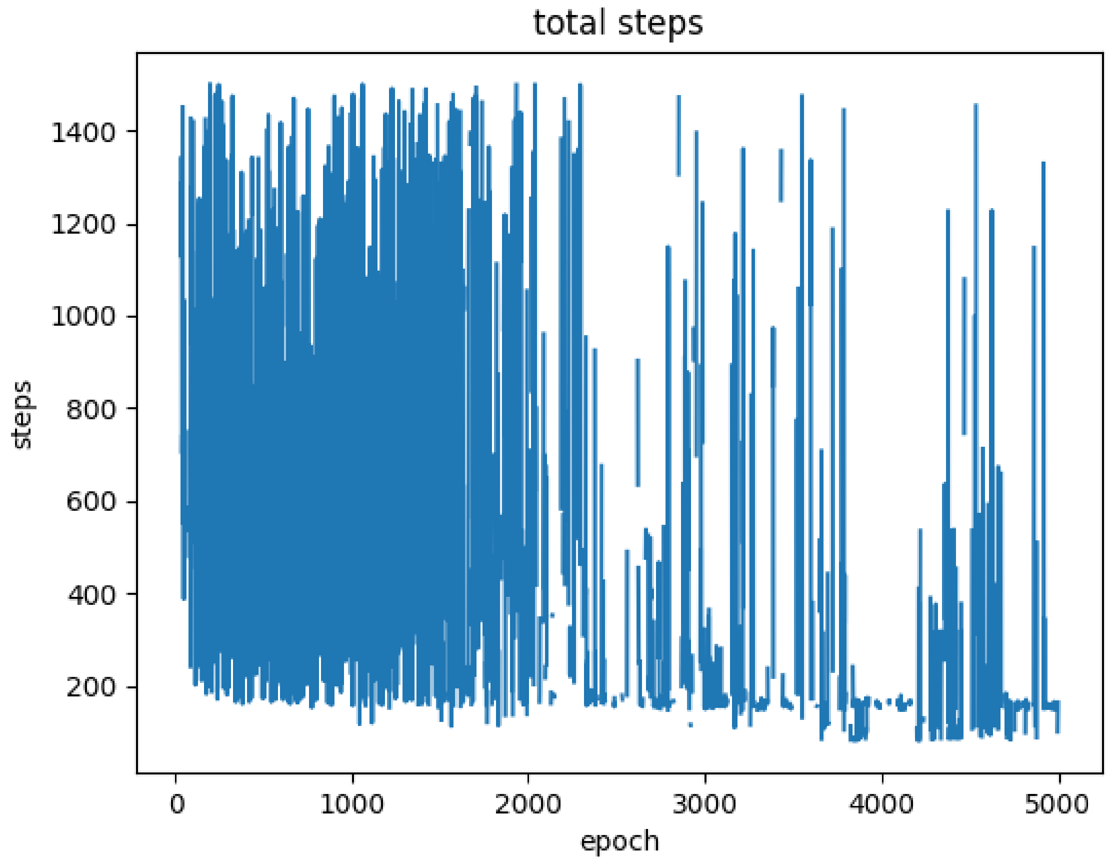
- learning rate = 20 * 20



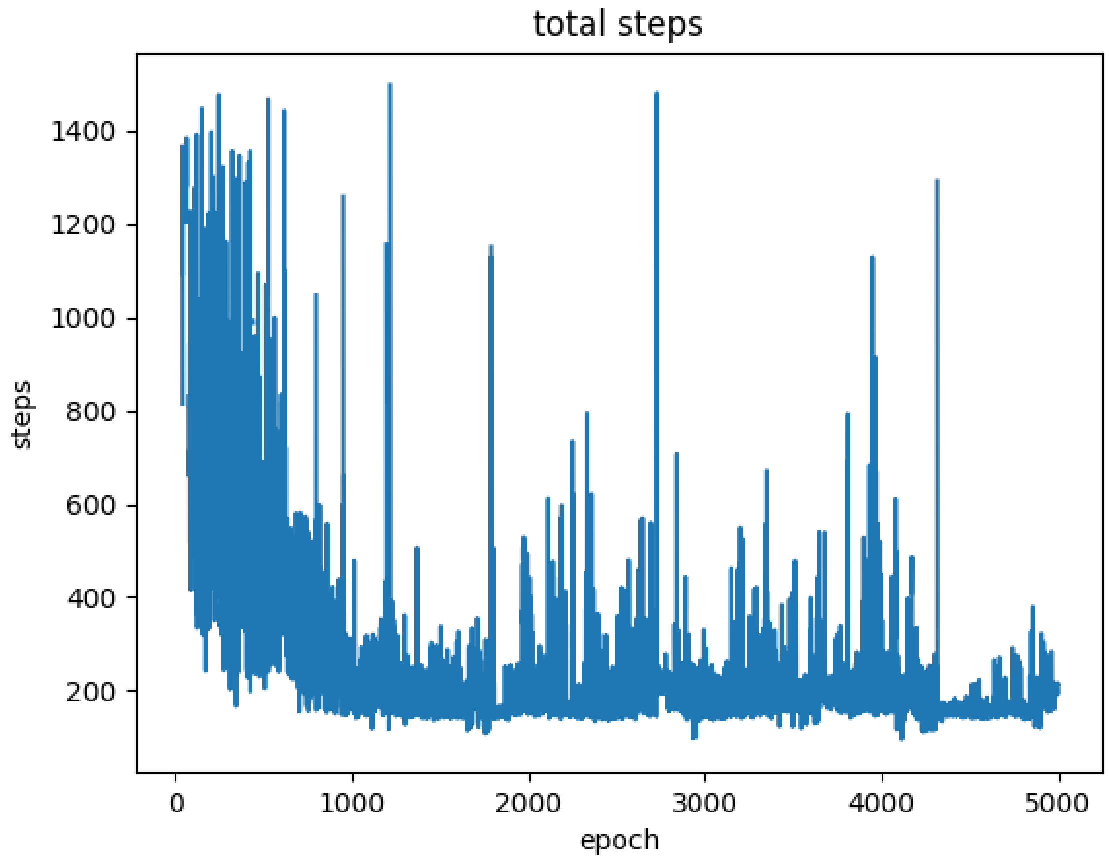
though the 5 *5 q-matrix still not working, but performance of 20* 20 q-matrix is worse than 15 *15 or 10* 10. A intuited reason is we have so much states so the agent just doesn't know how to select well at first, so it may need more training episode to solve.

step number

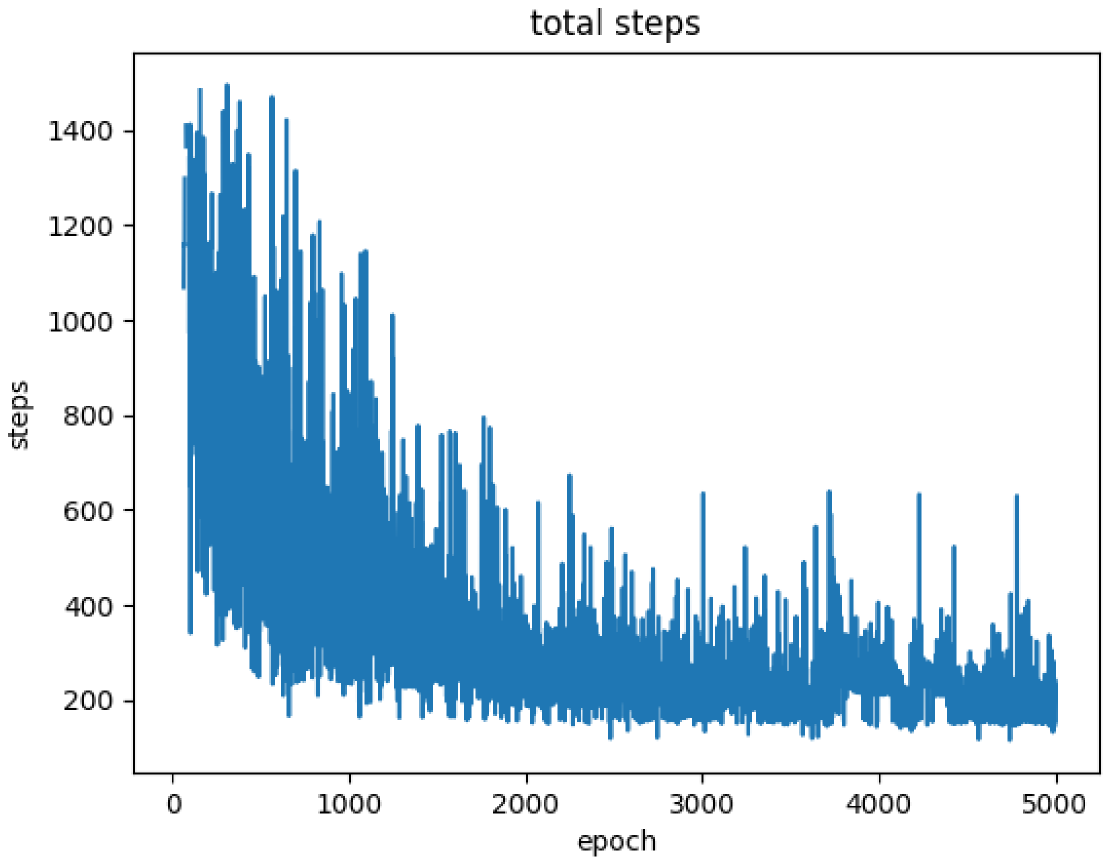
- state number = 5 * 5



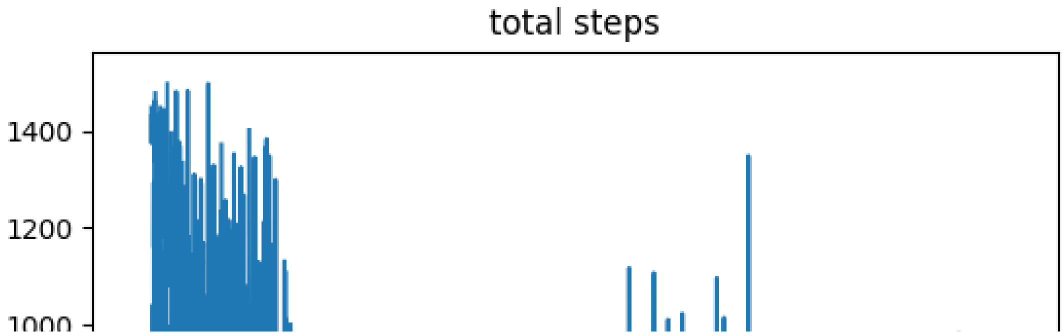
- learning rate = 10 * 10



- learning rate = 15 * 15



- learning rate = 20 * 20



Answers about questions

After analyzing the experiment, please answer the following questions in your report (30%)

1. What kind of RL algorithms did you use? value-based, policy-based, model-based? why? (10%)

   ```
   ANS:
   I used Q-learning, a value-based reinforcement learning algorithm, which meant it didn't any policy selection function to play the game, and I choosed Q-learning algorithm just because it was easy to implement.
   ```

2. This algorithms is off-policy or on-policy? why? (10%)

   ```
   ANS:
   Q-learning is a off-policy reinforcement algorithm, because it use maximum Q-value which meant it was not independent.
   ```

3. How does your algorithm solve the correlation problem in the same MDP? (10%)

   ```
   ANS:
   Q-matrix is simply equal to MDP. and due to lack of state number of q-matrix compared to NN, the Q-learning have no urgently demand to solving overfitting problem of correlation.
   And because of our game environment is a continal problem the variavant of each state will be very small.
   ```