

UMassAmherst

Project Assignment Report

Course ECE 697AA – Artificial Intelligence Based Wireless
Network Design

Teacher Beatriz Lorenzo

Student Xiaohao Xia, Yinxun Wu

Introduction

This report will introduce our completion of the ECE697AA project assignment and the analysis of the experimental results.

Following reference paper [ref2], assigned for presentation at seminars 2 and 3, enhance the network slicing model by considering that tenants share their subscribers' resources (whenever idle) as fog nodes with the infrastructure provider (InP) to augment its infrastructure. The fog nodes are equipped with communication, computing, and storage capabilities. The InP will slice the augmented infrastructure that contains its own resources and fog resources to serve service requests from multiple tenants. Using this enhanced network slicing model, answer the following questions:

[ref2] N. Van Huynh, D. Thai Hoang, D. N. Nguyen and E. Dutkiewicz, "Optimal and Fast Real-Time Resource Slicing With Deep Dueling Neural Networks," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1455-1470, June 2019.

Contents

1. Project Introduction.....	1
1.1 Completed Content.....	1
1.2 Technology.....	1
1.3 Codes.....	1
2. Question 1.....	2
2.1 Problem description for Question 1.....	2
2.2 Answer to Question 1.....	2
3. Question 2.....	3
3.1 Problem description for Question 2.....	3
3.2 Answer to Question 2.....	3
4. Question 3.....	5
4.1 Problem description for Question 3.....	5
4.2 Answer to Question 3.....	5
5. Question 4.....	7
5.1 Problem description for Question 4.....	7
5.2 Answer to Question 4.....	7
6. Question 5.....	9
6.1 Problem description for Question 5.....	9
6.2 Answer to Question 5.....	9
7. Discussion.....	10
7.1 Impact of Iteration.....	10
7.2 Limitations of Matlab.....	11
8. Conclusions.....	12
References.....	13

1. Project Introduction

1.1 Completed Content

We use Q-learning, Double Deep Q-learning to solve the optimization problem and compare the convergence of these two algorithms according to the parameters in the reference paper [ref2]. Besides, we answer all the questions in Project Assignment.

As in the paper [ref2], we also consider three common classes of slices, i.e., utilities (class-1), automotive (class-2), and manufacturing (class-3) as evaluation parameters. The immediate reward r_c for each accepted request from class-1, class-2, and class-3 are 1, 2, and 4, respectively. The arrival rates are 8, 10, 12 for each class, and we set hidden layers as 64 to the neural network. In addition, we also set the resources in the fog node and InP as 100 and 500 respectively.

In addition, we boosted the iterations to 10,000 to compare and analyze with the 1000 in the paper.

1.2 Technology

The development tool we used is Matlab, version 2018b. As shown in Figure 1-1, we built the network architecture in the paper [ref2] through the *deep learning toolbox* and realized the training function of the algorithms used.

The Deep Learning Toolbox provides a framework for designing and implementing deep neural networks through algorithms, pre-trained models and applications.

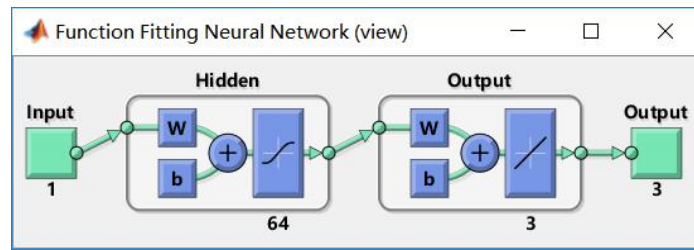


Figure 1-1 Simple network architecture diagram in Matlab

1.3 Codes

By running the Matlab code corresponding to the problem, we can get the corresponding output results.

The codes for the our project assignment are publicly available at

<https://github.com/XiaohaoXia/ECE697AA-Course-Project>

2. Question 1

2.1 Problem description for Question 1

Assume the network provider aims to maximize the immediate reward defined in [equation (16), ref2]. Formulate the network provider's resource allocation problem as a centralized optimization problem. This problem includes the allocation of radio, computing and storage resources to network slices (by the network provider) to meet the slice requests from tenants.

2.2 Answer to Question 1

According to *equation (16)* and all notations are the same as the reference paper [ref2]. We set the resources in the fog node and InP as 100 and 500 respectively.

$$r(\mathbf{s}, a_{\mathbf{s}}) = \begin{cases} r_c, & \text{if } e_c = 1, \quad a_s = 1, \text{ and } \mathbf{s}' \in \mathcal{S}, \\ 0, & \text{otherwise.} \end{cases} \quad (16)$$

$$\mathcal{R}_{\pi}(\mathbf{s}) = \lim_{K \rightarrow \infty} \frac{\mathbb{E}\{\sum_{k=0}^K r(\mathbf{s}_k, \pi(\mathbf{s}_k)) | \mathbf{s}_0 = \mathbf{s}\}}{\mathbb{E}\{\sum_{k=0}^K \tau_k | \mathbf{s}_0 = \mathbf{s}\}}, \quad \forall \mathbf{s} \in \mathcal{S}, \quad (17)$$

$$\begin{aligned} \mathcal{R}_{\pi}(\mathbf{s}) &= \lim_{K \rightarrow \infty} \frac{\mathbb{E}\{\sum_{k=0}^K r(\mathbf{s}_k, \pi(\mathbf{s}_k)) | \mathbf{s}_0 = \mathbf{s}\}}{\mathbb{E}\{\sum_{k=0}^K \tau_k | \mathbf{s}_0 = \mathbf{s}\}} \\ &= \frac{\bar{\mathcal{L}}_{\pi} r(\mathbf{s}, \pi(\mathbf{s}))}{\bar{\mathcal{L}}_{\pi} y(\mathbf{s}, \pi(\mathbf{s}))}, \quad \forall \mathbf{s} \in \mathcal{S}, \end{aligned} \quad (18)$$

From the *equation (17), (18)* described in the paper [ref2]. We can formulate the resources problem as the followings:

$$\begin{aligned} &\text{Max} \sum_{\mathbf{s}} L(\mathbf{s} | \mathbf{s}') r_c 1(e_c = 1) 1(a_s = 1) \\ &\text{subject to } Q_s(s, a_s, \theta_i) * \nabla_s Q_s(s, a_s, \theta_i) > q \\ &\quad \mathbf{s} = 0, 1, \dots, p \\ &\quad a_s, r_c > 0 \text{ for any } s \text{ and } c \end{aligned}$$

And we can find that the \mathbf{s} is the variable in the optimization. From the formula, the objective function is to maximize the objective reward function of resource allocation, and the limiting condition is to achieve the optimal value of the resource supply of each node.

3. Question 2

3.1 Problem description for Question 2

Solve the optimization problem and represent the number of requests versus the optimum reward for different numbers of available fog nodes. *You can use Matlab and the simulation parameters described in [ref2].*

3.2 Answer to Question 2

In this question, all parameters are set the same as those in the reference paper [ref2]. That is, the arrival rates are 8, 10, 12 for each class, and the completion rates are 3, with the reward set 1, 2, 4 for each class respectively. We set hidden layers as 64 to the neural network. In addition, we also set the resources in the fog node and InP as 100 and 500 respectively. And in this part, we consider that using the simple LSTM-neural network to solve the optimization problem.

And the output is shown below:

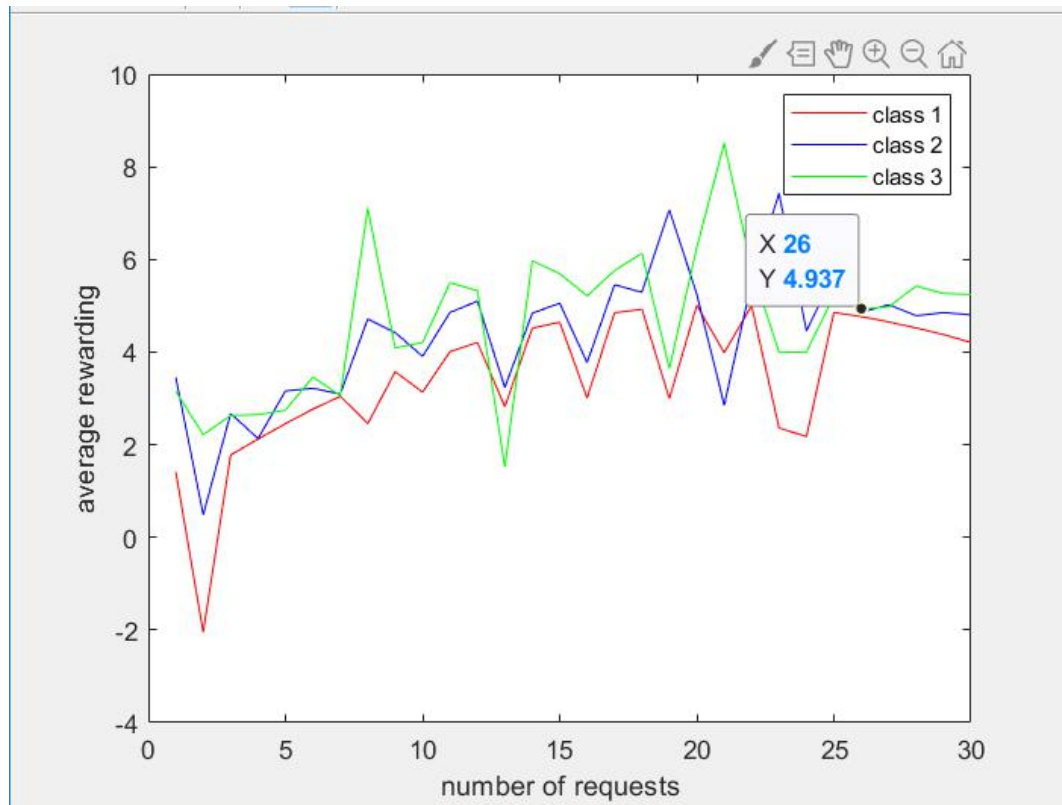
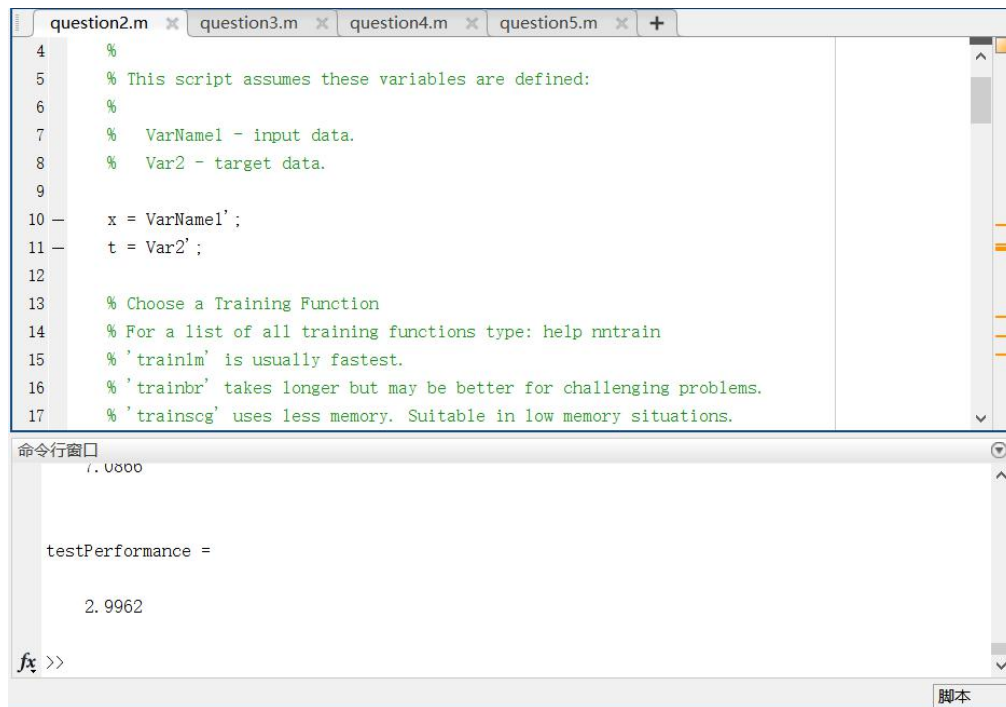


Figure 3-1 The output of the LSTM-neural network



```
question2.m question3.m question4.m question5.m +
4 %
5 % This script assumes these variables are defined:
6 %
7 % VarName1 - input data.
8 % Var2 - target data.
9
10 x = VarName1';
11 t = Var2';
12
13 % Choose a Training Function
14 % For a list of all training functions type: help nntrain
15 % 'trainlm' is usually fastest.
16 % 'trainbr' takes longer but may be better for challenging problems.
17 % 'trainscg' uses less memory. Suitable in low memory situations.
```

```
命令行窗口
./, 0000

testPerformance =

    2.9962

fx >>
```

脚本

Figure 3-2 The test performance of the LSTM-neural network

We can find that class-1, class-2, and class-3 will fluctuate greatly before convergence. This may be affected by the number of iterations. At this time, the test performance of this algorithm is 2.9962. We will analyze this phenomenon in Chapter 7. Discussion.

4. Question 3

4.1 Problem description for Question 3

Implement the Q-learning algorithm described in [Section IV, ref2] for the enhanced network slicing model. *You can use Matlab or Tensorflow and the simulation parameters described in [ref2].* Compare the performance with the optimum solution obtained in question 2.

4.2 Answer to Question 3

We use the Q-learning method to solve the problem with the parameters the same as in **Question 2** (that is, using the same parameters as in reference paper *ref[2]*) with the hidden layer of 64.

The output is shown below:

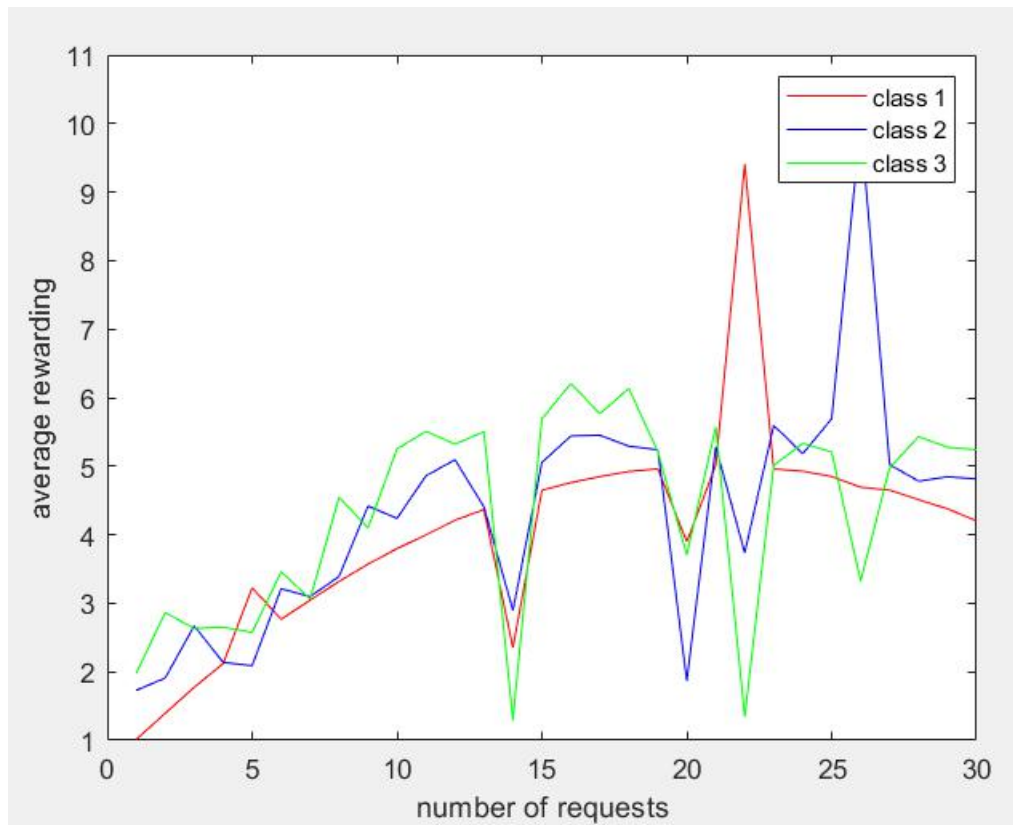
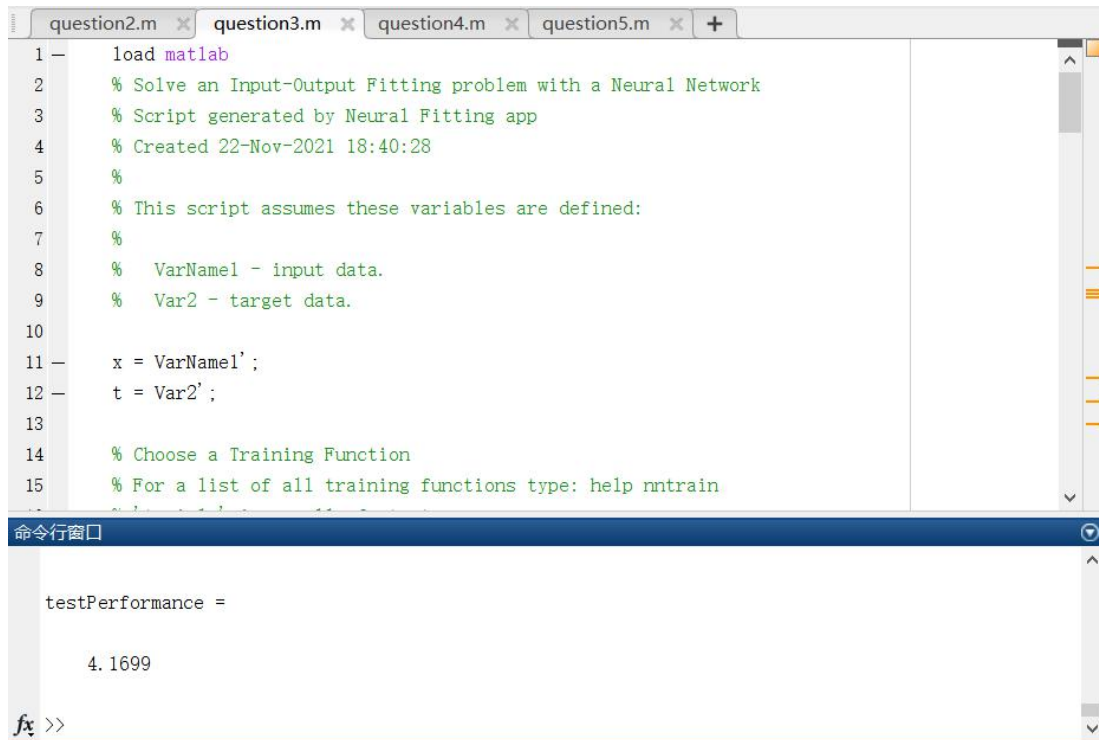


Figure 4-1 The output of the Q-learning



```
1 load matlab
2 % Solve an Input-Output Fitting problem with a Neural Network
3 % Script generated by Neural Fitting app
4 % Created 22-Nov-2021 18:40:28
5 %
6 % This script assumes these variables are defined:
7 %
8 % VarName1 - input data.
9 % Var2 - target data.
10
11 x = VarName1';
12 t = Var2';
13
14 % Choose a Training Function
15 % For a list of all training functions type: help nntrain
```

命令窗口

```
testPerformance =
4.1699
```

fx >>

Figure 4-2 The test performance of the Q-learning

By observing Figure 3-1 and Figure 4-1, we can see that when the number of the request is small (before 13), the Q-learning algorithm is significantly better than the algorithm we gave in **Question2**. When the number of request is large (after 13), both have a certain degree of volatility, and the effect is not very good.

Besides, we compare Figure 3-1 and Figure 4-1, taking the obvious class-1 as an example, we find that the class-1 curve of **Question 3** fluctuates less frequently than that of **Question 2** (Q-learning has only 4 fluctuations), which can indirectly indicate that the convergence of Q-learning is more stable compared to LSTM-neural network.

In addition, according to Figure 3-2 and Figure 4-2, from a numerical point of view, the test performance of the new Q-learning algorithm is 4.1699, which is also higher than the 2.9962 of the method in **Question 2**. This also shows that the new Q-learning algorithm is better.

5. Question 4

5.1 Problem description for Question 4

Implement the Double Deep Q-learning algorithm described in [Section V.A, ref2] for the enhanced network slicing model. *You can use Matlab or Tensorflow and the simulation parameters described in [ref2].* Compare the performance with the optimum solution obtained in question 2 and Q-learning in question 3.

5.2 Answer to Question 4

We use the Double Deep Q-learning method to solve the problem with the parameter the same in **Question 2** with hidden layer of 64.

And the output is shown below:

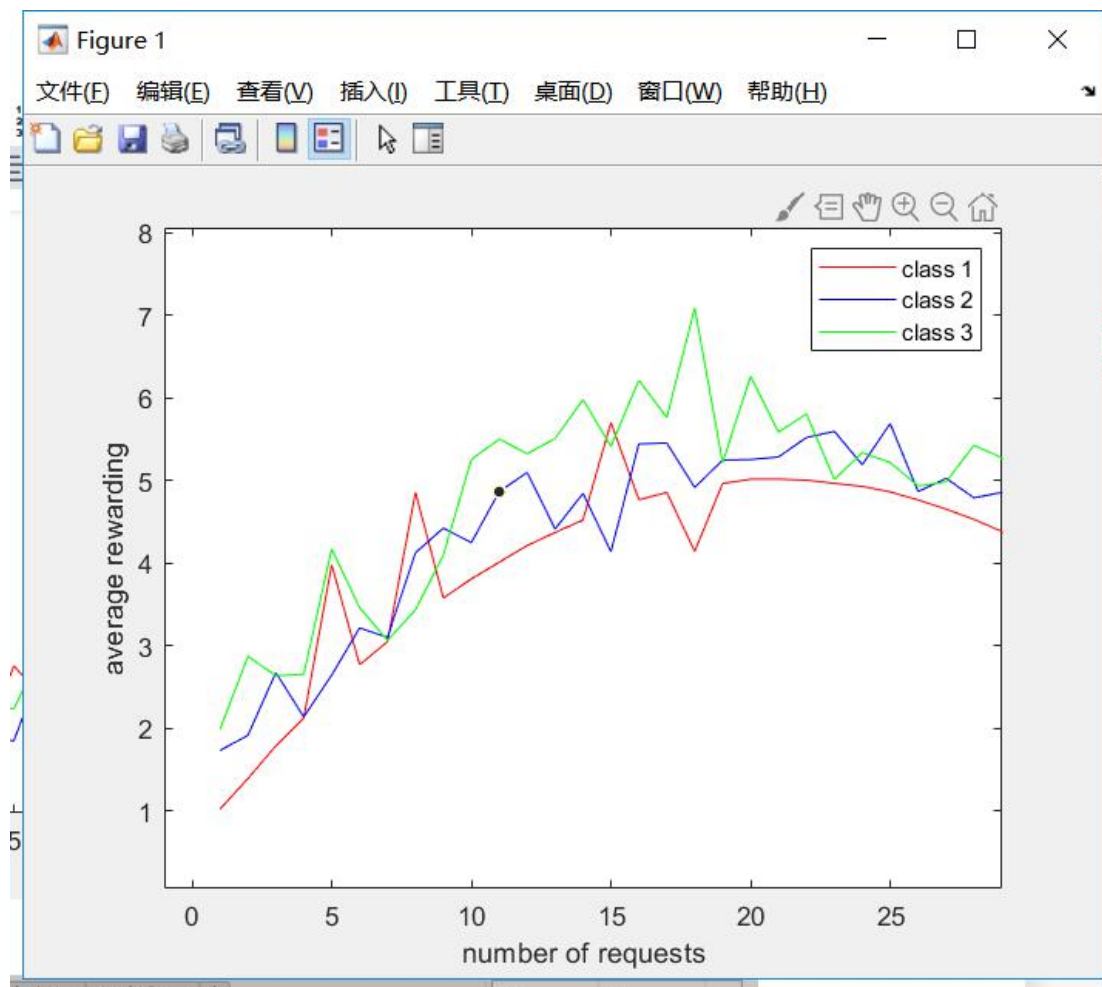
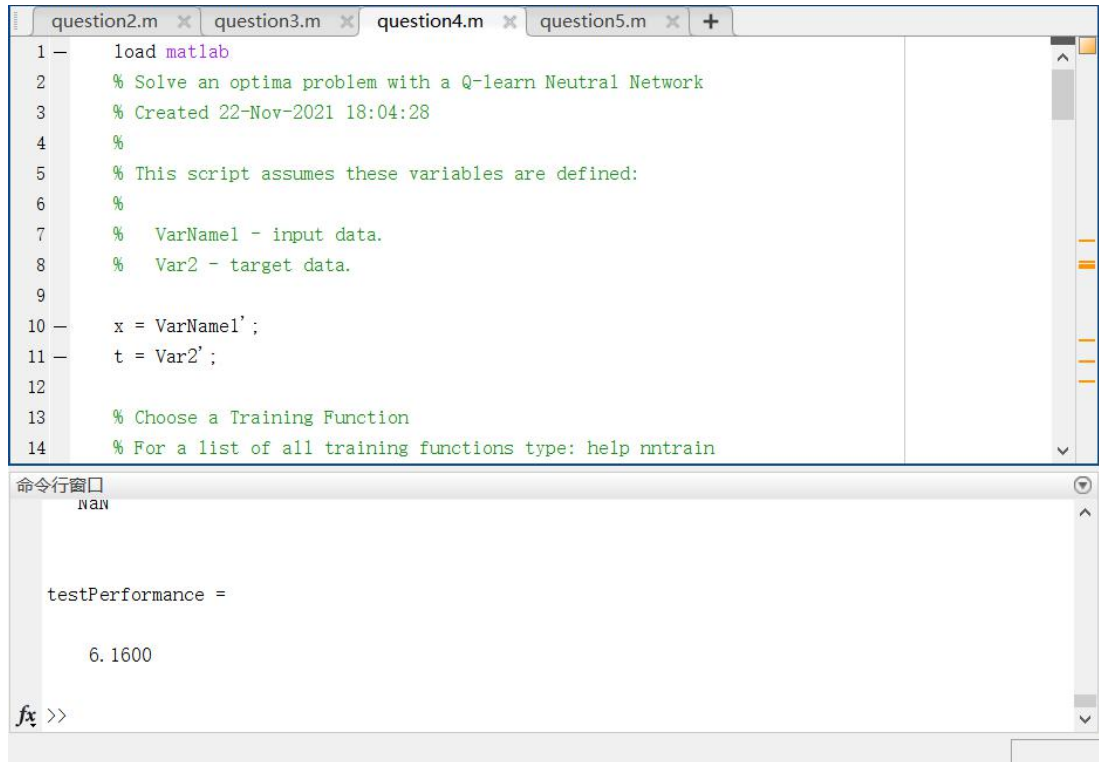


Figure 5-1 The output of the Double Deep Q-learning



```
1 - load matlab
2 % Solve an optima problem with a Q-learn Neural Network
3 % Created 22-Nov-2021 18:04:28
4 %
5 % This script assumes these variables are defined:
6 %
7 % VarName1 - input data.
8 % Var2 - target data.
9
10 - x = VarName1';
11 - t = Var2';
12
13 % Choose a Training Function
14 % For a list of all training functions type: help nntrain
```

命令窗口

```
NAN

testPerformance =

    6.1600

fx >>
```

Figure 5-2 The test performance of the Double Deep Q-learning

By observing Figure 3-1, Figure 4-1, and Figure 5-1, we can find that due to the number of iterations and computer CPU limitations, the output of **Question 4** still fluctuates, but we observe that the fluctuation of **Question 4** is significantly lower than that of **Question 2**, which proves that Double Deep Q-learning has better convergence efficiency and better stability than LSTM-neural network.

Besides, by comparing Figure 4-1 and Figure 5-1, we can find that curve of **Question 4** fluctuation is lower than **Question 3** (especially the class-1 and class-3, the class-3 fluctuation of **Question 4** is not as severe as those of **Question 2** and **Question 3**).

And take the class-1 curve as an example, we can also find that the class-1 curve of **Question 4** converges faster than the class-1 curve of **Question 3** (the class-1 of **Question 4** has converged before 20, and the class-1 of **Question 3** started to converge after 20), which proves that Double Deep Q-learning has better convergence efficiency than Q-learning.

According to Figure 3-2, Figure 4-2, and Figure 5-2, from a numerical point of view, the test performance of the Double Deep Q-learning algorithm is 6.1600, which is also higher than the 2.9962 of the method in **Question 2** and 4.1699 in method 3 in **Question 3**. This also shows that the new Double Deep Q-learning algorithm is better.

6. Question 5

6.1 Problem description for Question 5

Illustrate the convergence of the Q-learning and Double Deep Q-learning algorithms developed in Questions 3 and 4, respectively.

6.2 Answer to Question 5

From the comparison of the two algorithms, although the two curves are close to each other due to the same parameters, we can see that the Q-learning algorithm has a slower convergence and needs to run to the maximum number of iterations, while the Double Deep Q-learning algorithm has a stronger convergence and can get results soon.

Below is the reward versus iteration times, our main observation is the moment when the two algorithms start to converge. And from the figure 6-1, We can find that the convergence time of the Double Deep Q-learning algorithm is earlier than the Q-learning algorithm. Besides, we can find that the double deep Q-learning algorithm has better convergence than Q-learning algorithm under the same parameters.

And based on the test performance comparison between Question 3 and Question 4 (the value of Double Deep Q-learning is greater than that of Q-learning), and the class-1 convergence comparison in Figure 4-1 and Figure 5-1, we can find that Double Deep Q-learning converges faster.

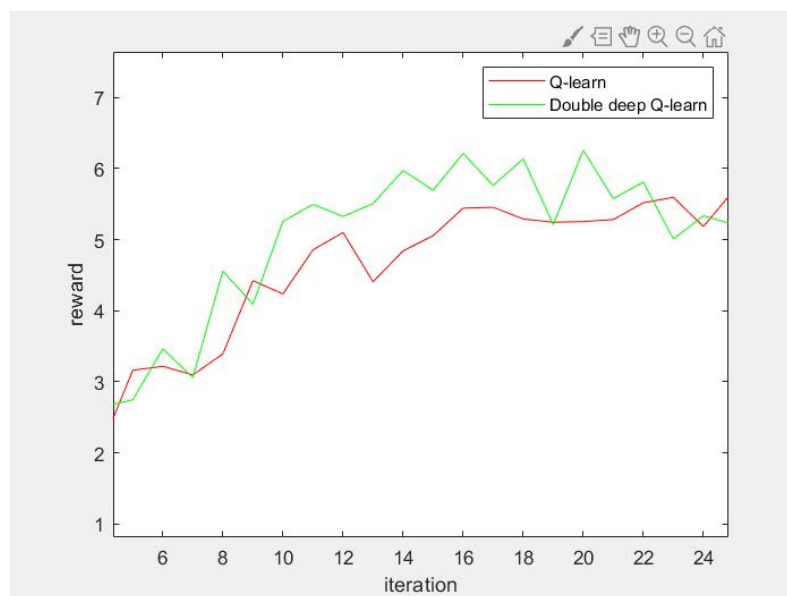


Figure 6-1 Comparison of Q-learning and Double Deep Q-learning

7. Discussion

7.1 Impact of Iteration

We can find that in Figure 3-1, Figure 4-1, and Figure 5-1, the curves of class-1,class-2,class-3 fluctuate more significantly before they start to converge. We guess that this may be limited by the number of iterations.

Therefore, we try to change the 1000 iterations following the paper [ref 2] to 10000 iterations with other parameters unchanged, and the outputs of Q-learning and Double Deep Q-learning algorithm are shown in Figure 7-1, and Figure 7-2.

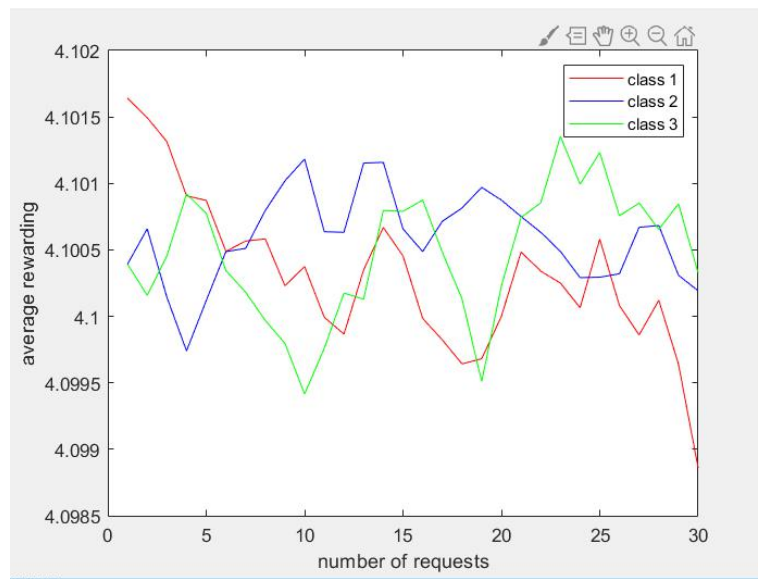


Figure 7-1 The output of Q-learning with 10,000 iterations

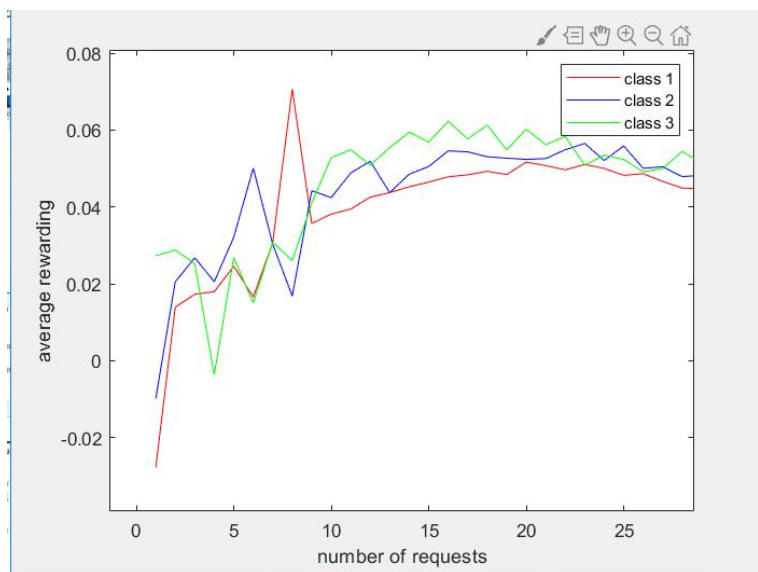


Figure 7-2 The output of Double Deep Q-learning with 10,000 iterations

By observing Figure 7-1 and Figure 7-2, we can first find that class-1, class-2, and class-3 have obvious convergence trends through the ordinate value and the curve trend.

And comparing the output of different iterations of the same algorithm (compare Figure 7-1 with Figure 4-1, compare Figure 7-2 with Figure 5-1), we can find that by increasing the number of iterations, the stability of the algorithm convergence is better. Especially the output of Double Deep Q-learning is the most obvious. We can find that when iterates 10,000 times, the convergence of the class-1, class-2, and class-3 curves of Double Deep Q-learning is very ideal.

In summary, we can find that Double Deep Q-learning converges faster than Q-learning, and the convergence curve is more stable.

7.2 Limitations of Matlab

The core technology of Matlab to implement machine learning lies in *Deep Learning Toolbox*, which enables Matlab to quickly implement Double Deep Q-learning for project operations.

However, due to the limitations of the tool, Matlab does not guarantee that the randomization process at the beginning of each time uses a fixed random number like tensorflow, which will affect the convergence results and test performance.

In addition, because Matlab occupies a large amount of computer resources when running, and more computing resources of the computer are needed to train the network, this is also a huge challenge to the performance of the computer.

If possible, in future work, we plan to use tensorflow to test again to ensure the stability of the results.

8. Conclusions

We use Matlab to solve the optimization problem through three algorithms (LSTM-neural network, Q-learning, Double Deep Q-learning). And by comparing the three algorithms, we found that Double Deep Q-learning has the best convergence effect.

In addition, we analyzed the number of iterations and the limitations of Matlab. When we changed the number of iterations from 1000 to 10000, Double Deep Q-learning had ideal output results.

References

[1] N. Van Huynh, D. Thai Hoang, D. N. Nguyen and E. Dutkiewicz, "Optimal and Fast Real-Time Resource Slicing With Deep Dueling Neural Networks," IEEE Journal on Selected Areas in Communications, vol. 37, no. 6, pp. 1455-1470, June 2019.

[2] Deep Learning Toolbox in Matlab:

<https://www.mathworks.com/solutions/deep-learning.html>