# Machine Learning Assessment 1

180770429 Jingxiong Li

# 1.Linear regression

a)

Modify the function *calculate\_hypothesis.m* to return the predicted value for a single specified training example. Include in the report the corresponding lines from your code.

[5 points]

To implement the , write down like below:



figure 1 code for *calculate\_hypothesis.m*

b)

Modify it to use the *calculate\_hypothesis* function. Include the corresponding lines of the code in your report.

[5 points]

Comment the original code and write like this.

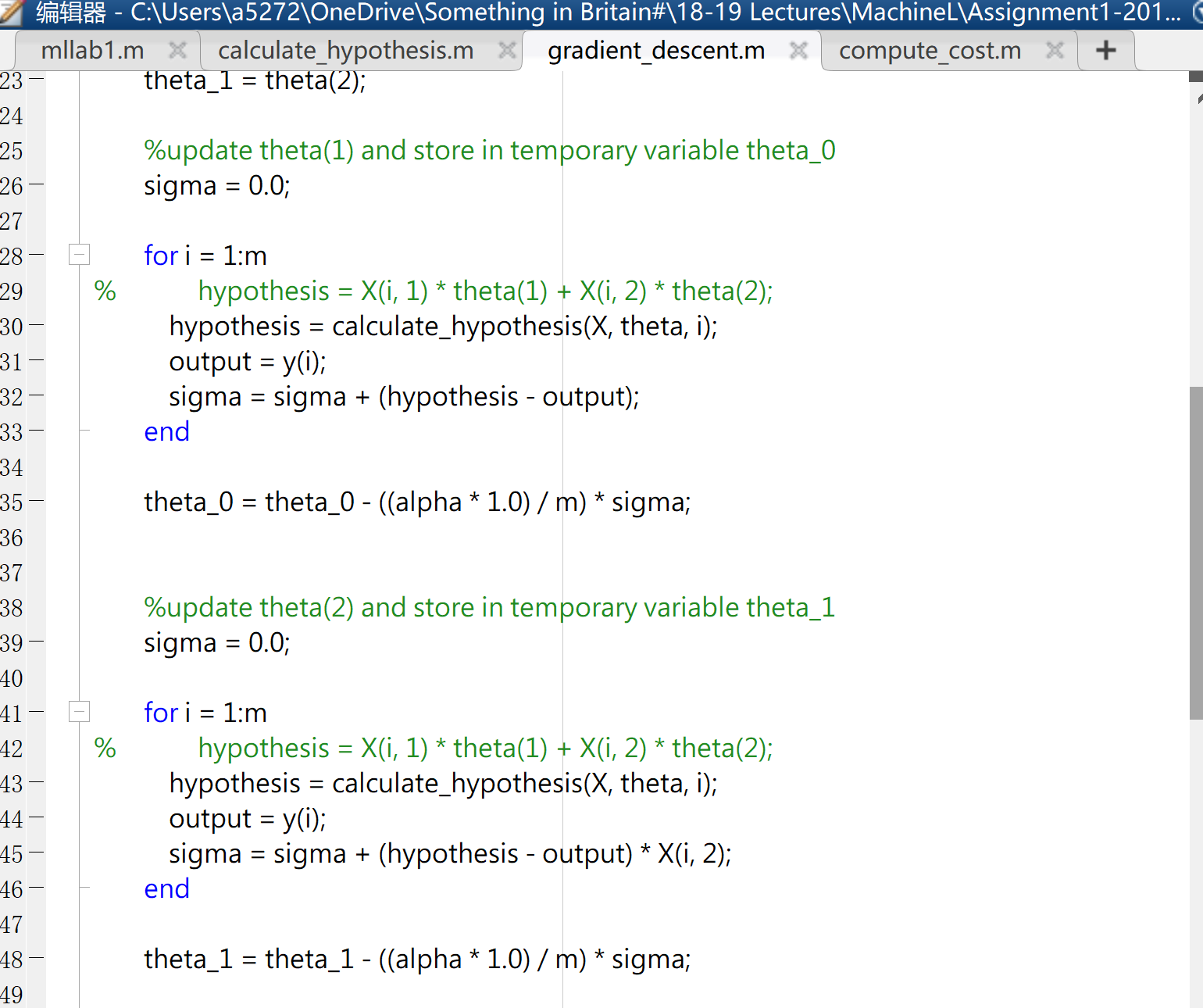


figure 2 modified *gradient\_descent*

c)

Observe what happens when you use a very high or very low learning rate. Document and comment on your findings in the report.

[5 points]

Hi-learning rate (α = 0.01):

When the learning rate is high, the curve of cost is rough.

The learning speed is fast (interaction time is low), but it may miss the pole.

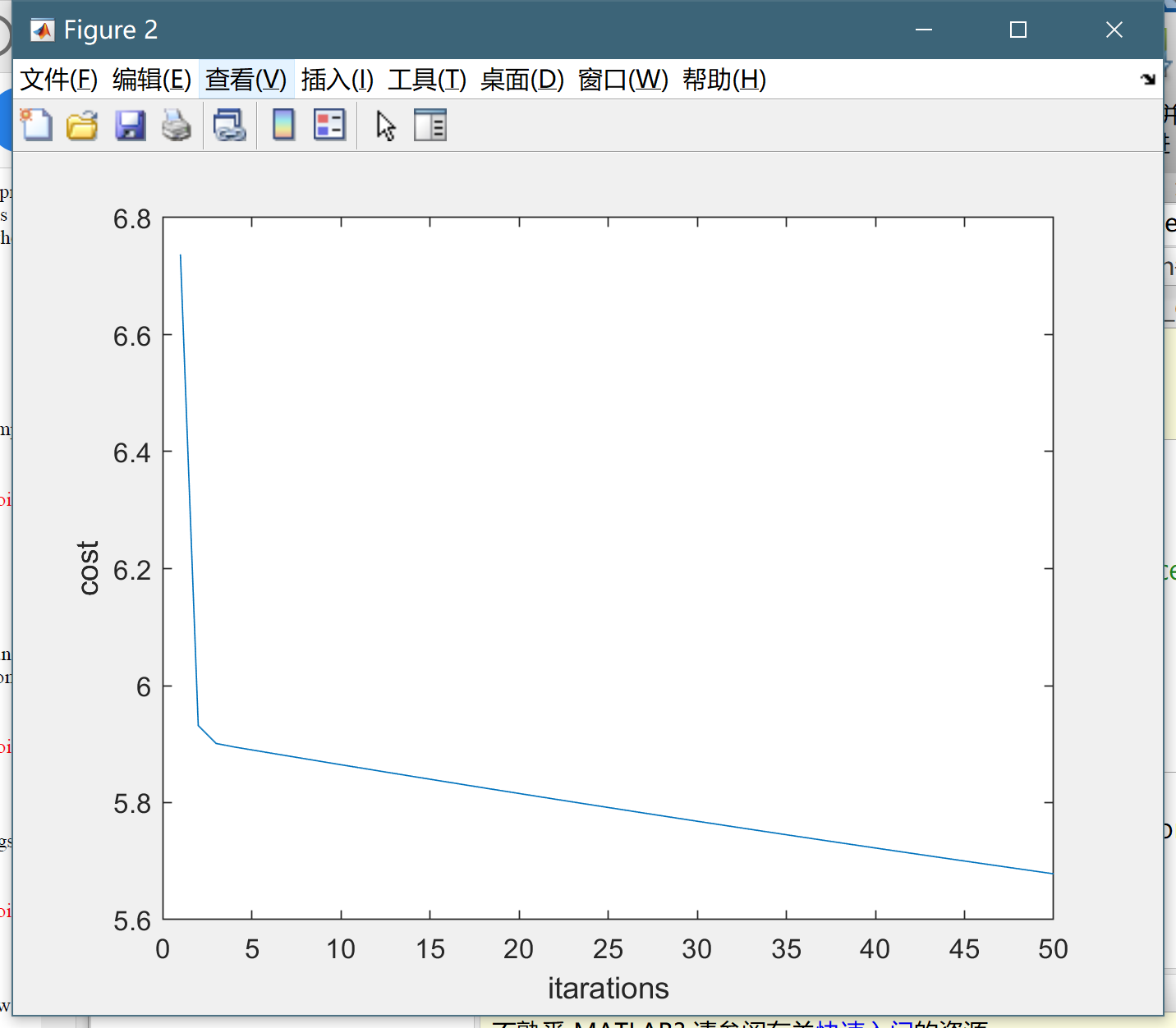


figure 3 Hi-learning rate (α = 0.01)

Low-Learning Rate (α = 0.001):

When the learning rate is high, the curve of cost is smoother.

When using a lower learning rate, the curve we got is smoother and it is more likely to reach the local optimum.

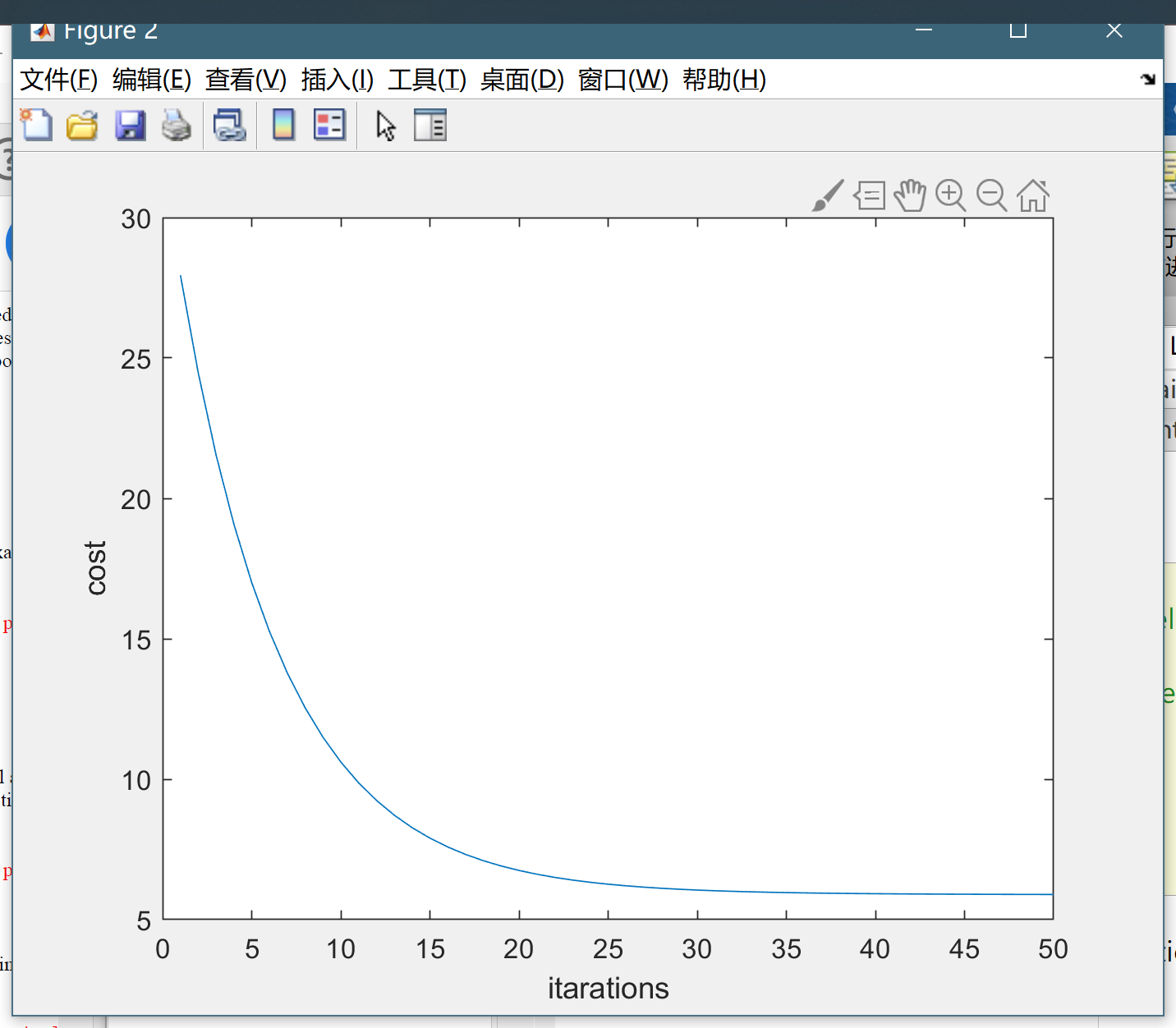


figure 4 Low-Learning Rate (α = 0.001)

# 2.Linear Regression with Multiple Variables

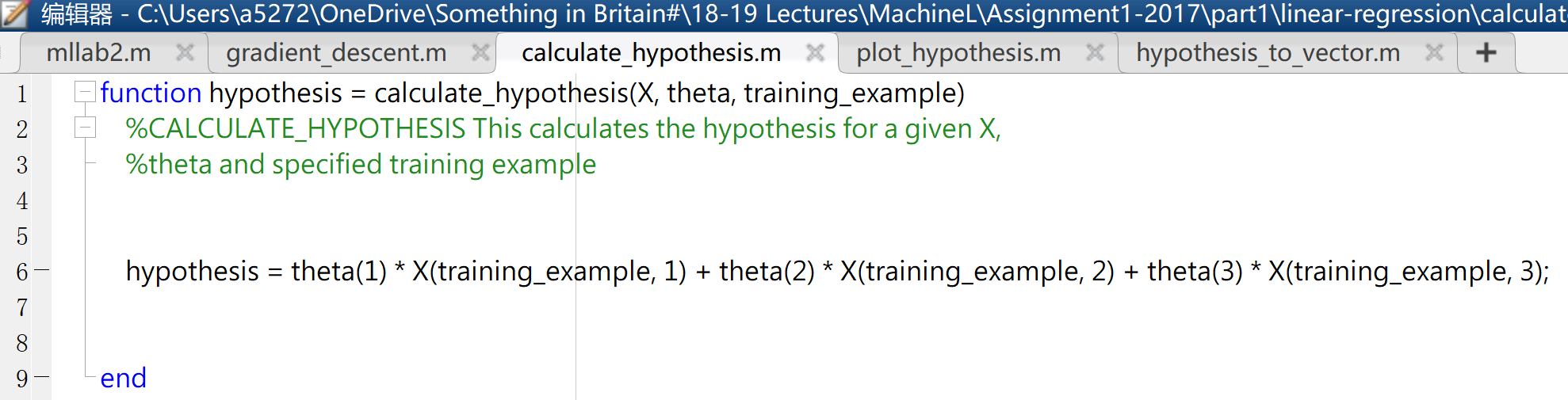
a)

Modify the functions *calculate\_hypothesis* and *gradient\_descent* to support the new hypothesis function. This should be sufficiently general so that we can have any number of extra variables. Include the relevant lines of the code in your report.

[5 points]

Add on the third theta and x

figure Linear Regression with Multiple Variables: code



b)

Run mllab2.m and see how different values of alpha affect the convergence of the algorithm. Print the theta values found at the end of the optimization. Include the values of theta and your observations in your report.

[5 points]

α = 5 θ value

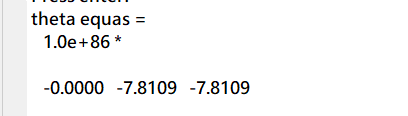
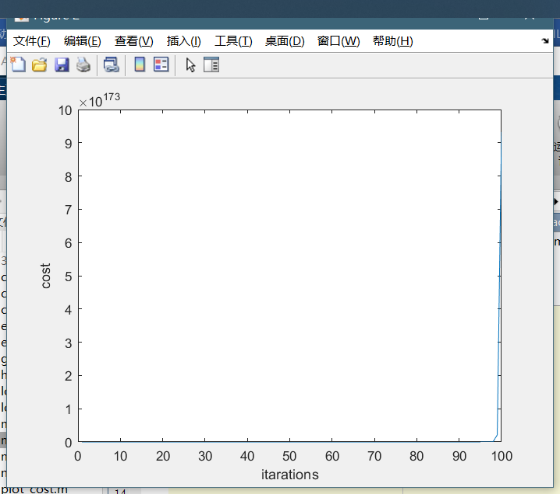


figure α= 5 (overshooting)

α = 0.1 θ value equal to

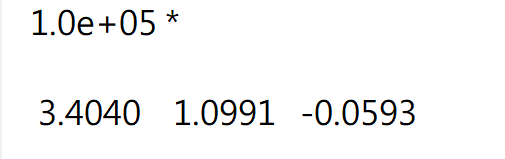
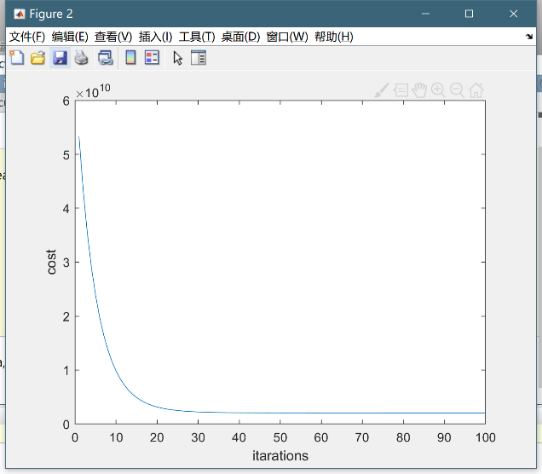
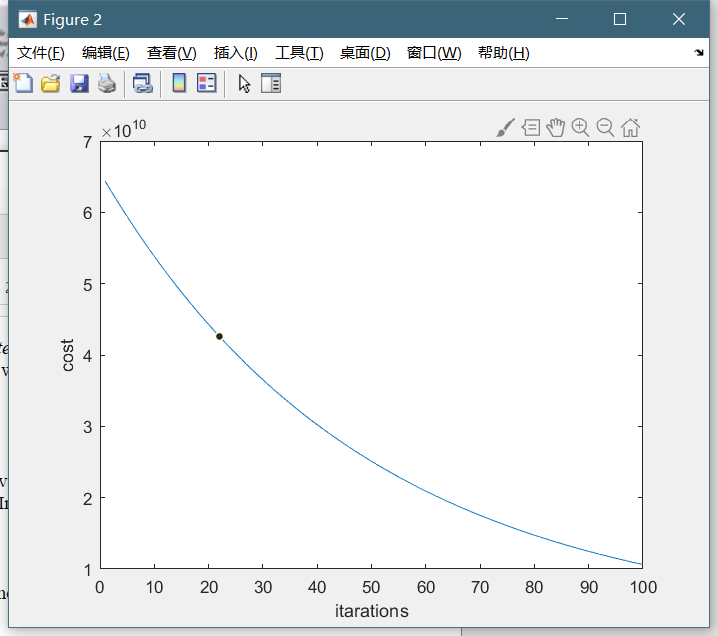
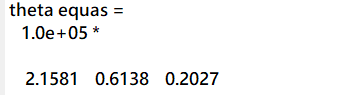


figure α = 0.1

α = 0.01 θ value equal to

figure α = 0.01



Observations: α has a strong effect to the interaction.

When α is large (as α = 5 ), converge interaction the speed of convergence is faster, however, it has the risk to miss the optimal.

When α is small (as α = 0.01 ), the speed may maybe too slow.

Modify the function as below

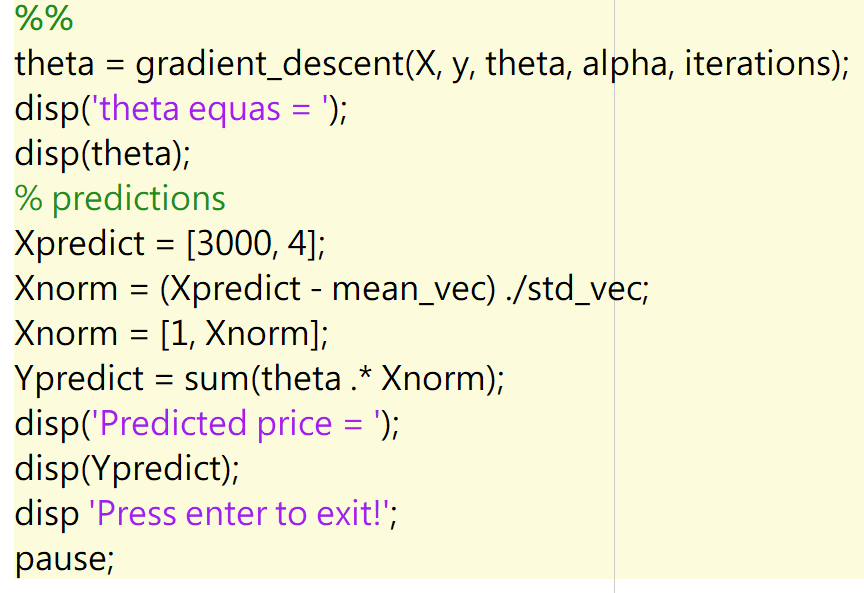


figure 9 house prediction code

When 1650 sq.ft. 3 bedrooms

The prediction value is



When 3000 sq.ft. 4 bedrooms

The prediction value is

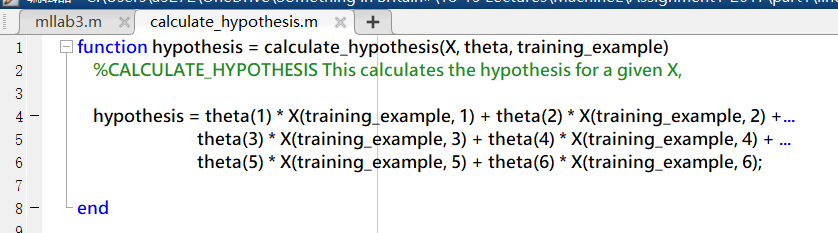


# 3.Regularzed Linear regression

a)

Note that the punishment for having more terms is not applied to the bias. This cost function has been implemented already in the function *compute\_cost\_regularised*. Modify *gradient\_descent* to use the *compute\_cost\_regularised* method instead of *compute\_cost.* Include the relevant lines of the code in your report and a brief explanation.

[5 points]



Following the requirements, modify the function with the punishment coefficient I.

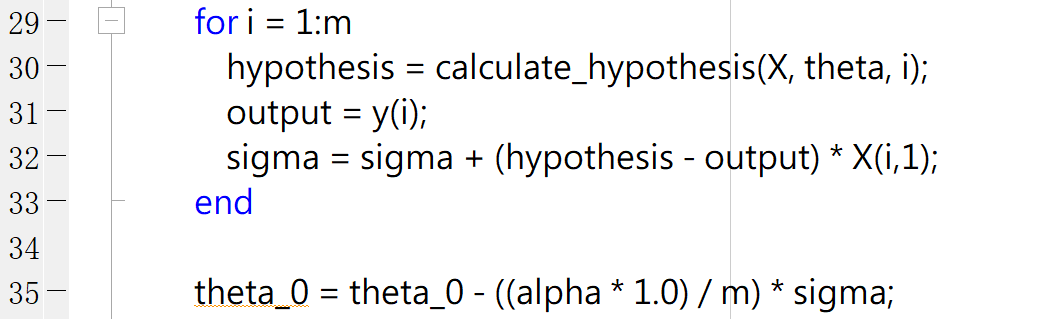
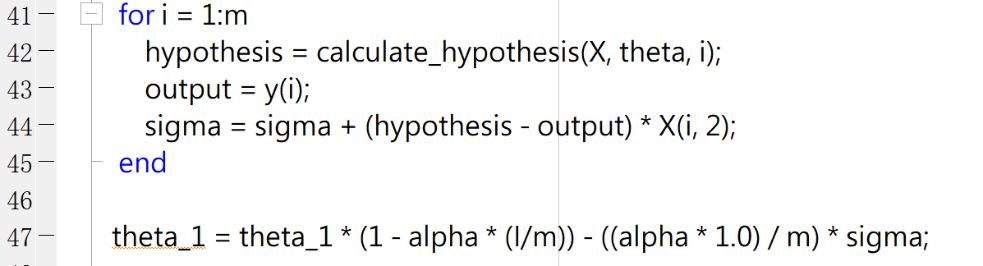
Modify the cost\_vetor to the regularized cost vetor.

As X has been extended, to so the *calculate\_hypothesis* is modified as the screenshot.

Next, modify *gradient\_descent* to incorporate the new cost function. Include the relevant lines of the code in your report.

[5 points]

figure modified *gradient\_descent* (up, j=0), (down, j>0)



First of all, find the best value of alpha to use in order to optimize best. Report the value of alpha that you found in your report.

[5 points]

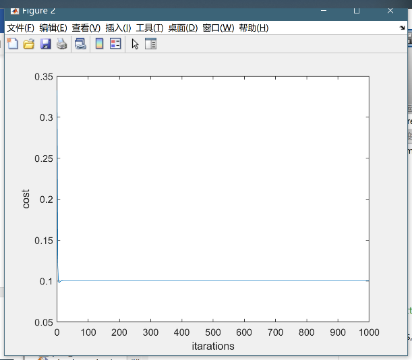
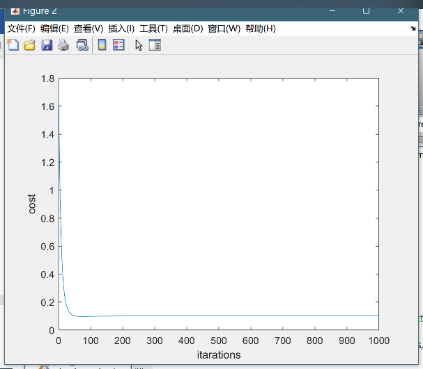
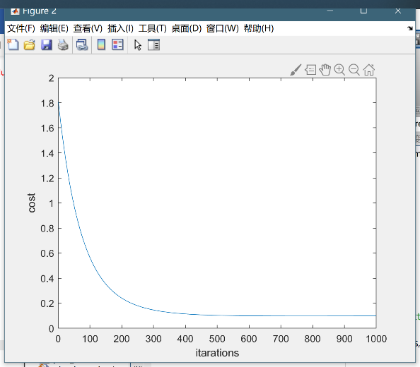


figure 11 α= 0.01，α= 0.1，α= 1

the best value tested is 0.1

Next, experiment with different values of 𝜆 and see how this affects the shape of the hypothesis. Note that *gradient\_descent* will have to be modified to take an extra parameter, l (which represents 𝜆). Include in your report the plots for a few different values of 𝜆 and comment.

[5 points]

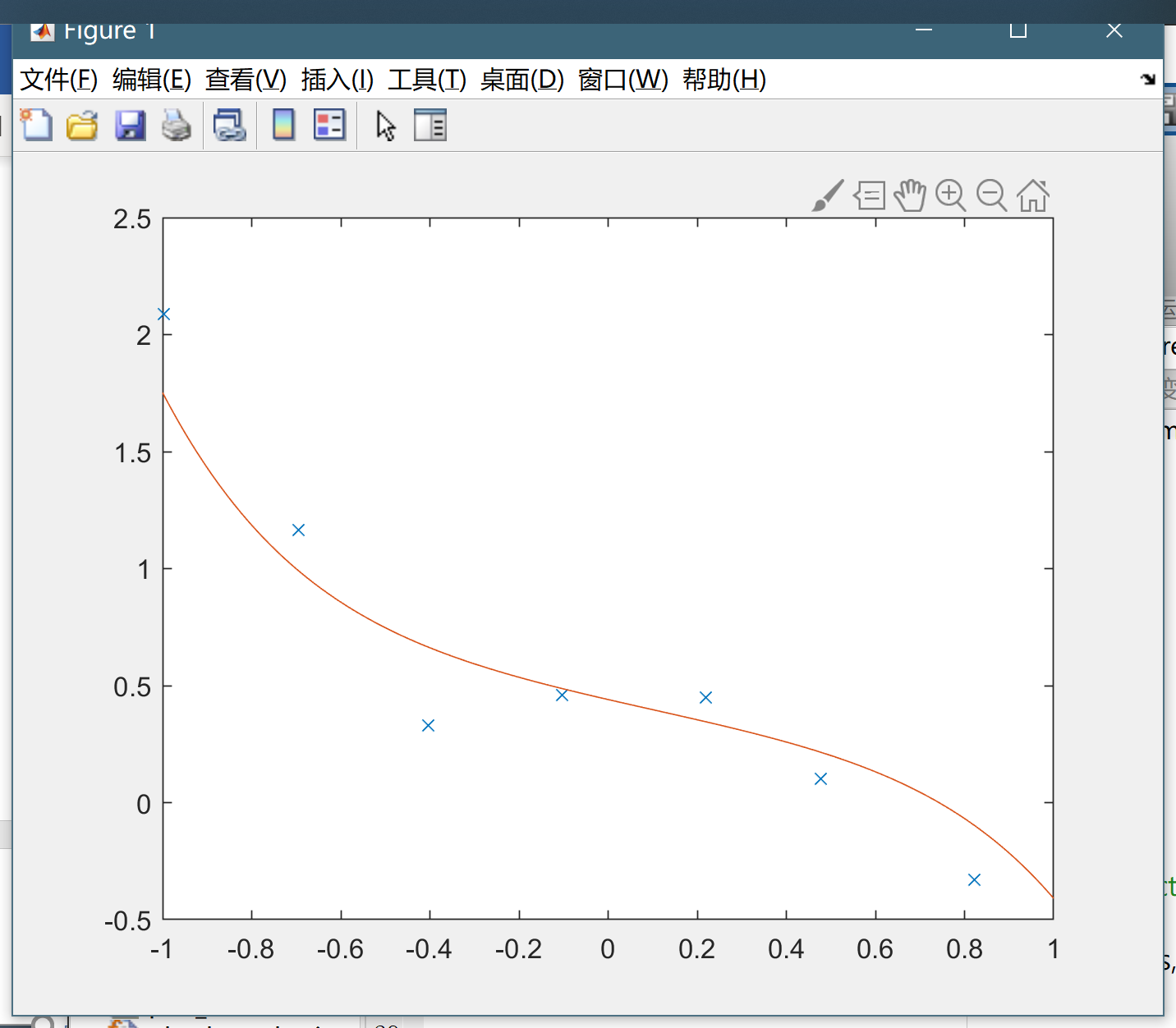
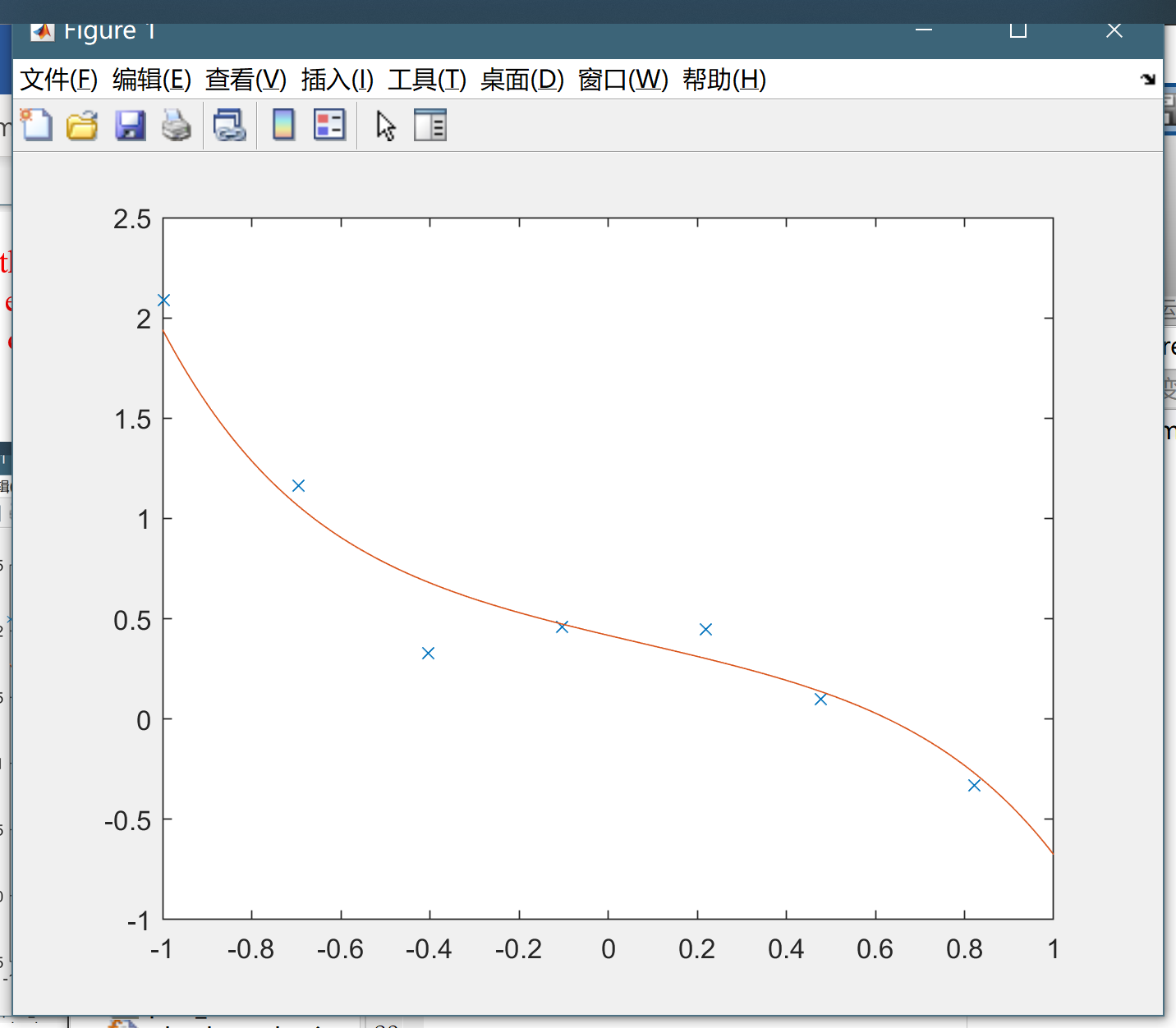
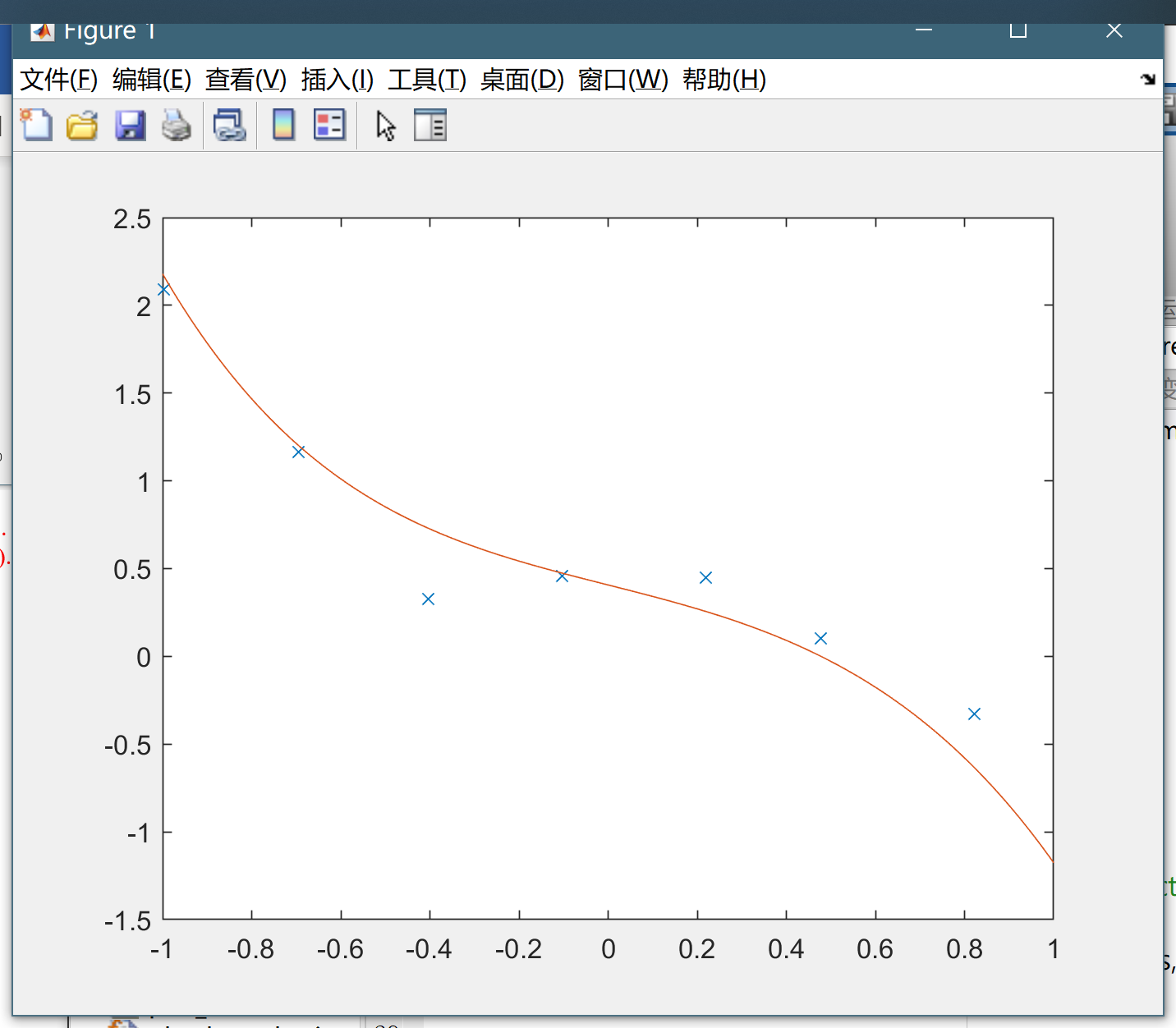


figure 12 I = 0，I = 1，I = 2

Investigating the fitted figures, when I = 0 or I = 1, almost all the points are on the plot. When I = 2, the overfitting rate is lower and this number is better.

# Assssment1 (part2)

## 1. Logistic Regression

**Task1:** Include in your report the relevant lines of code and the result of the running the plot\_sigmoid\_function.m.

[3 points]

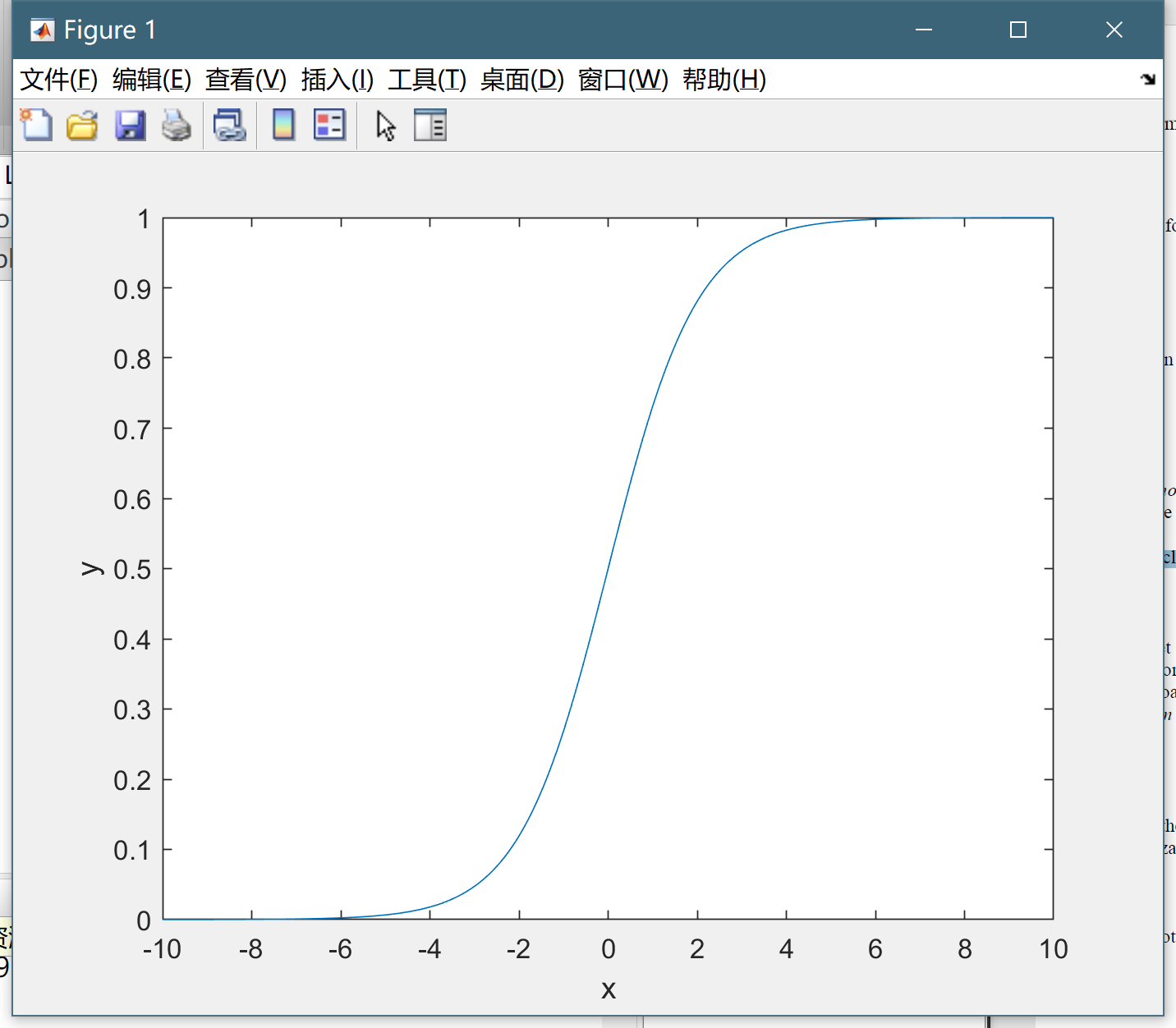
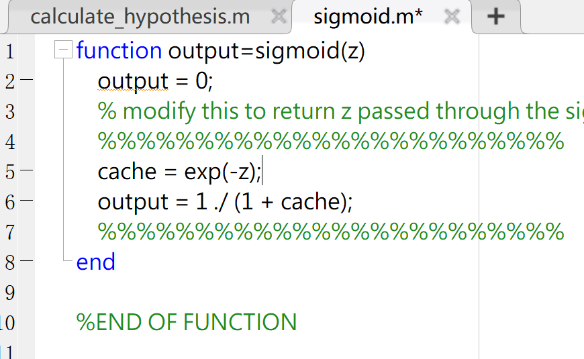


figure 13 *sigmoid.m* and the result for **Task 1**

**Task 2**. Plot the data again to see what it looks like in this new format. Enclose this in your report.

[2 points]

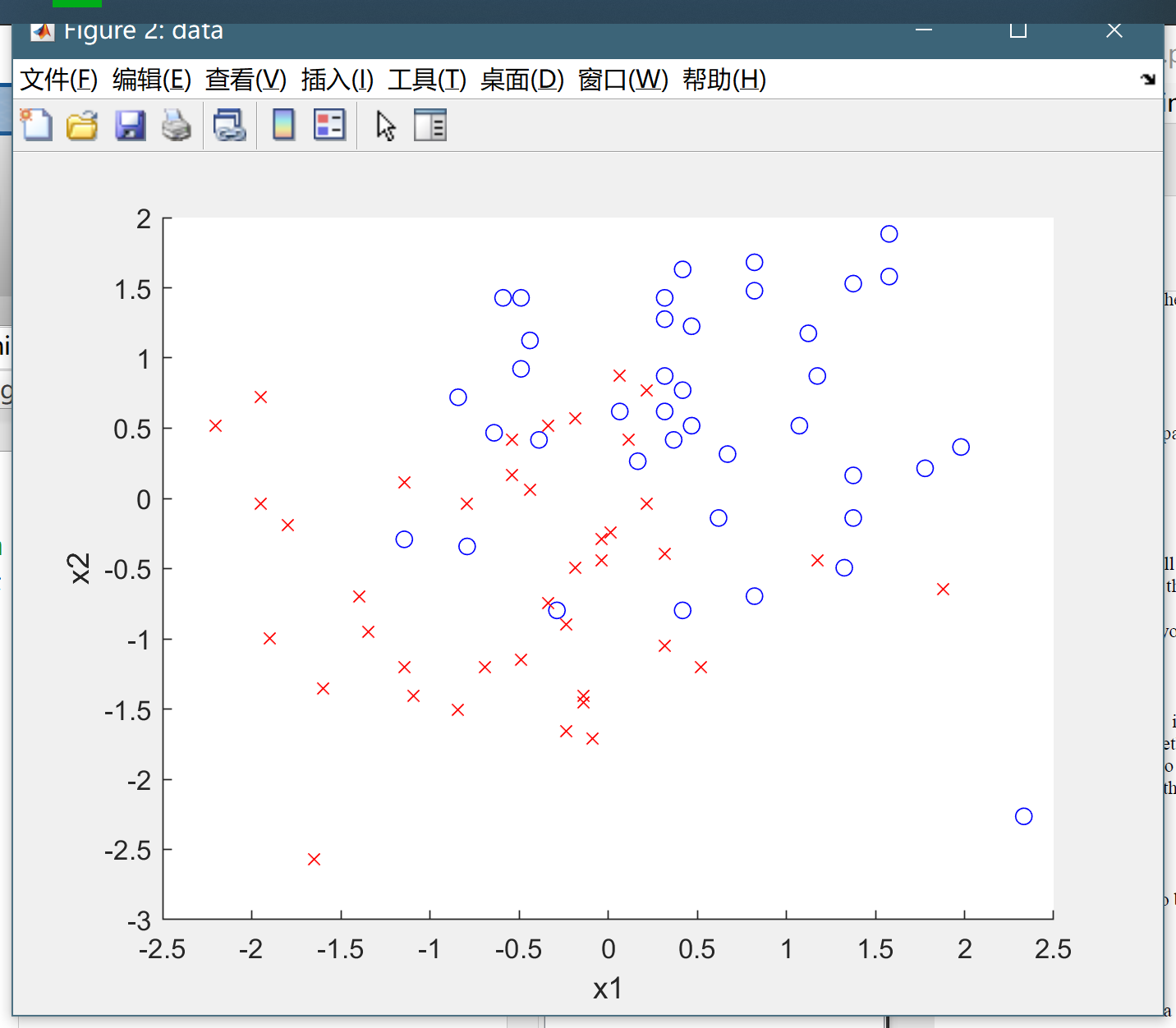
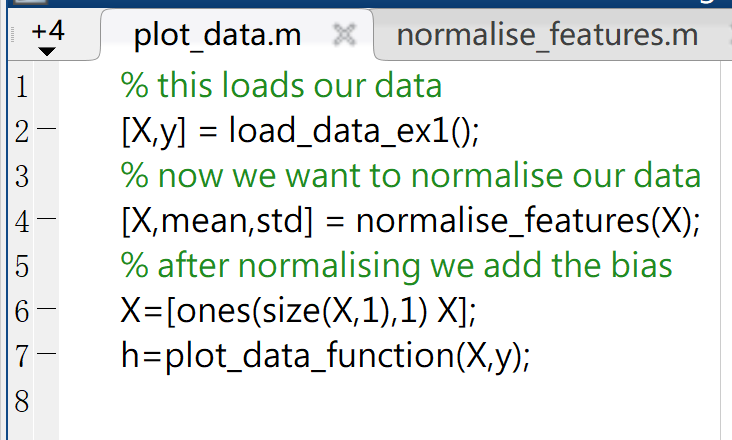
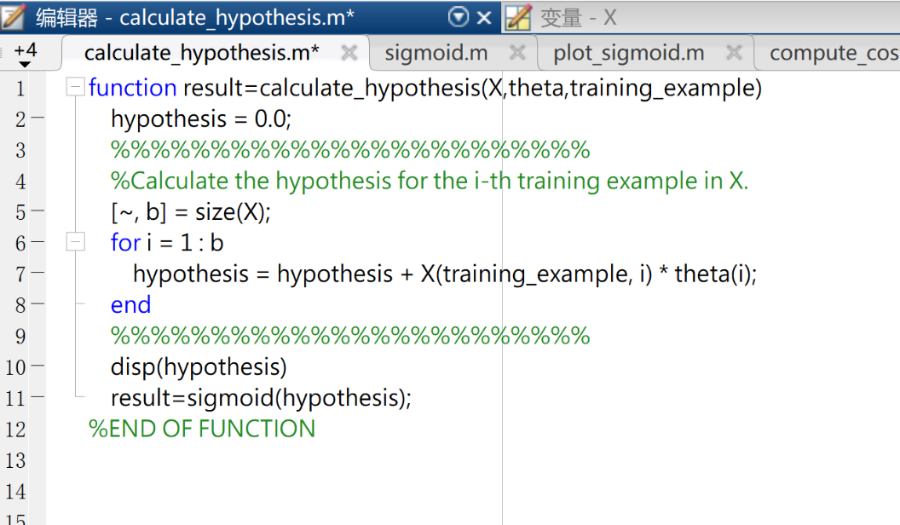
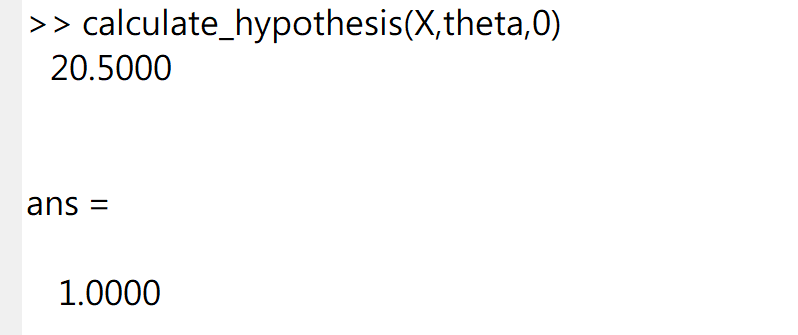


figure 14 regularized data and plot for **Task 2**

**Task 3.** Modify the *calculate\_hypothesis.m.* The function should be able to handle datasets of any size. Enclose in your report the relevant lines of code.

[5 points]

figure *calculate\_hypothesis.m* and test result for **Task 3**



**Task 4**. Modify the line. To calculate a logarithm, you can use log(x). Now run the file *lab2\_lr\_ex1.m* What is the final cost found by the gradient descent algorithm? In your report include the modified code and the cost graph.

[5 points]

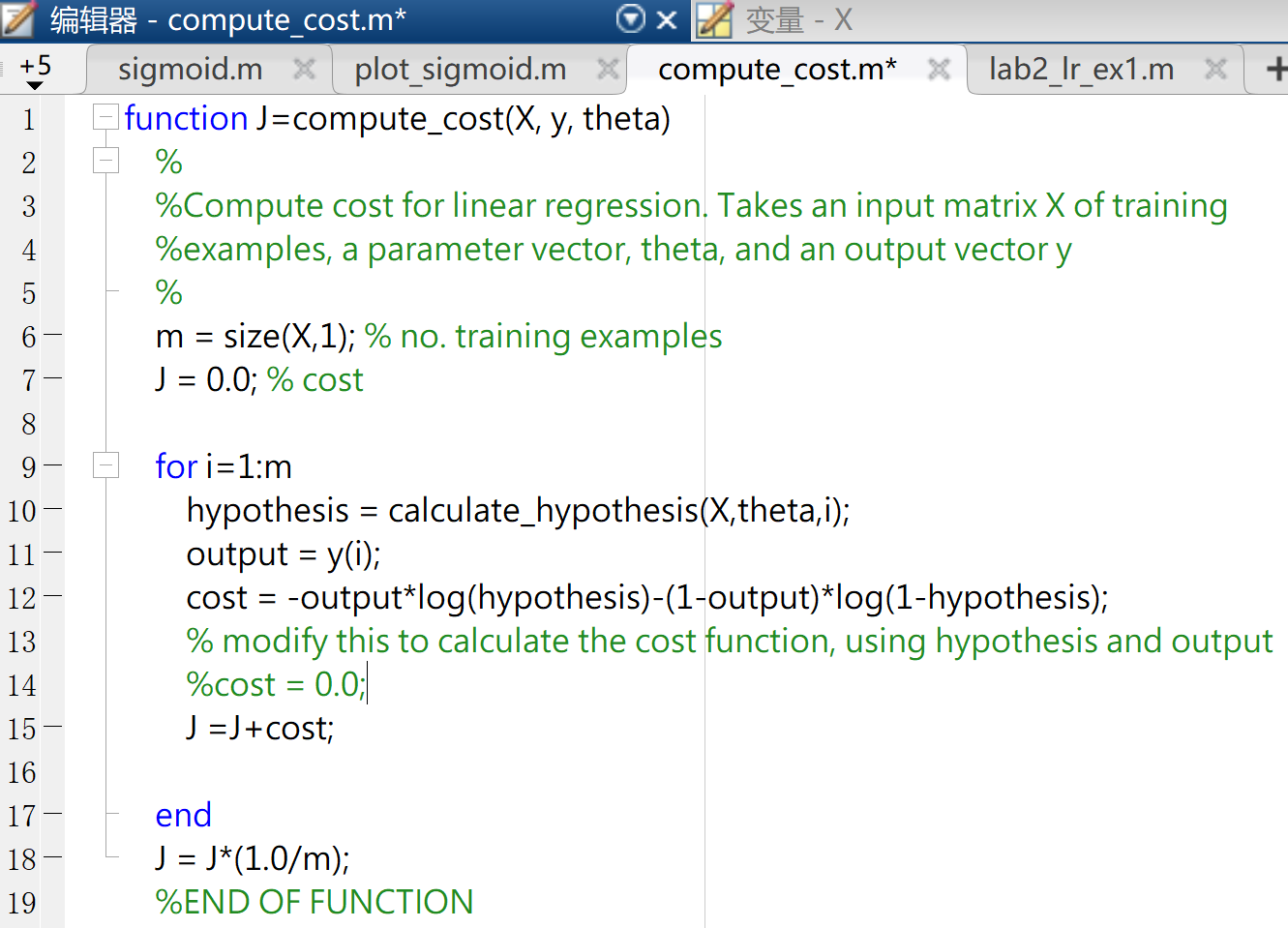


figure 16 the modified code for **Task 4**

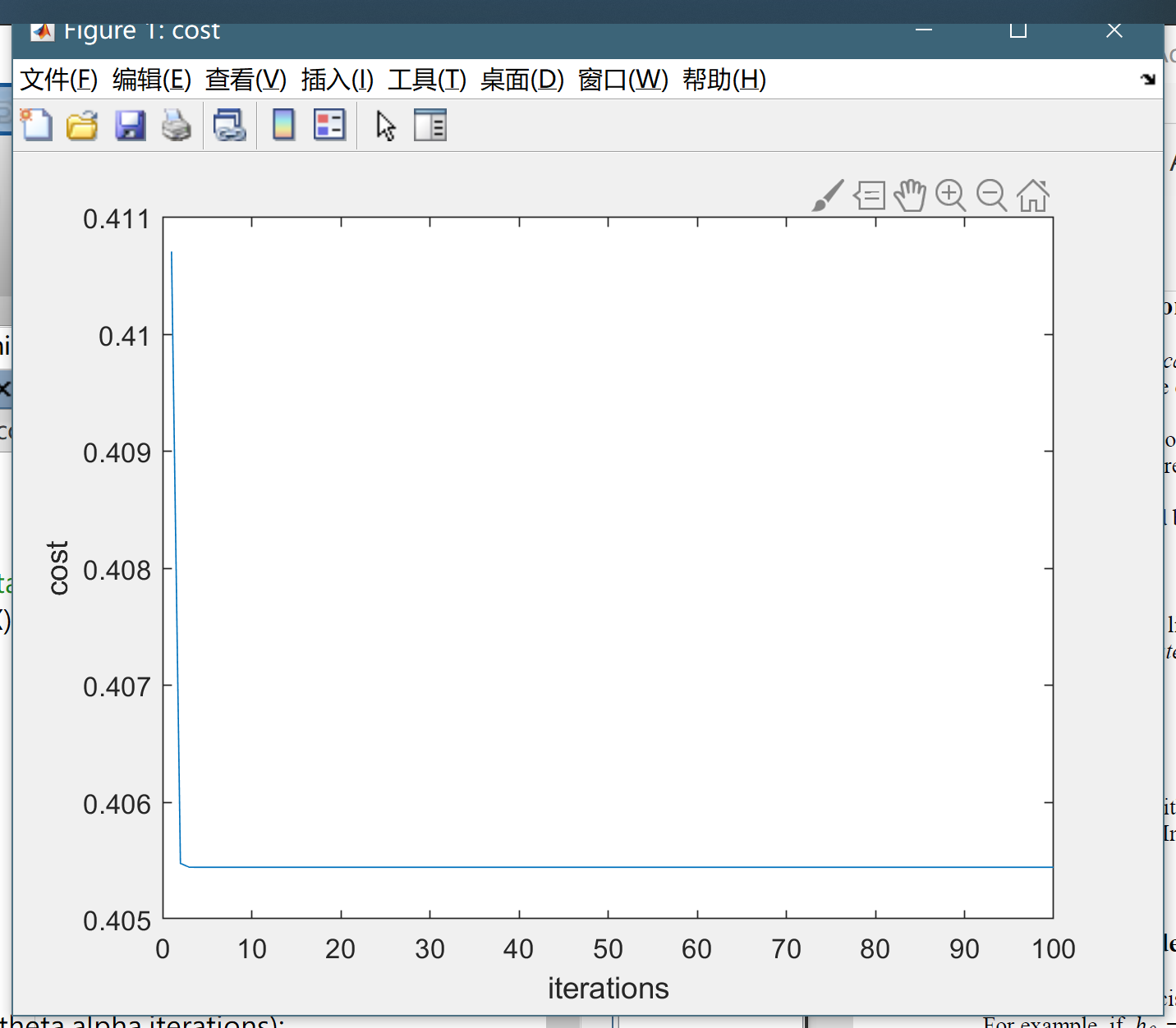
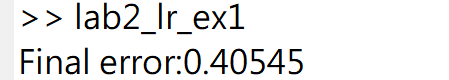
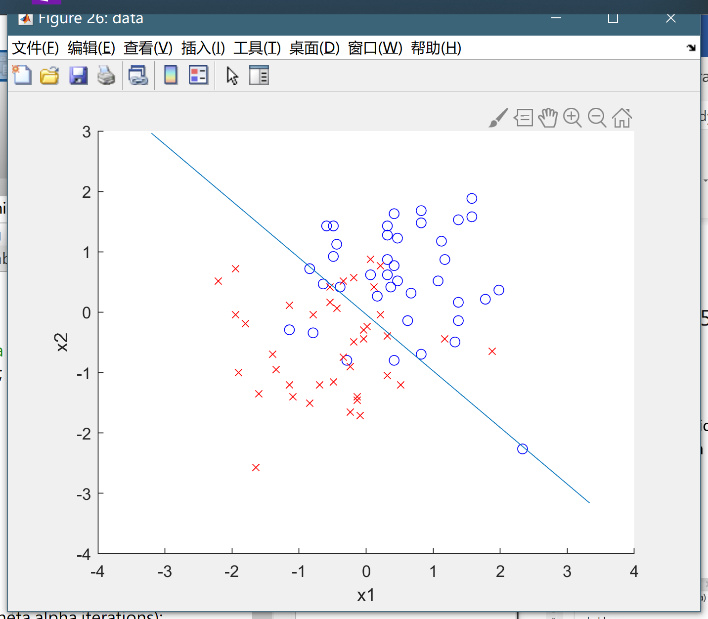


figure 17 the cost graph for **Task 4**

**Task 5.** Plot the decision boundary. Uncomment the relevant plot function in ***lab2\_lr\_ex1.m*** and include the graph in your report.

[5 points]



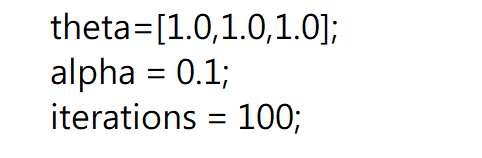


figure 18 code and graph for **Task 5**

**Task 6.** Run the code in ***lab2\_lr\_ex2.m*** several times. What is the general difference between the training and test error? When does the training set generalize well? Demonstrate two splits with good and bad generalisation and put both graphs in your report.

[2 points]

Because of the decision boundary is linear and the training set is small, the training error might be small. And the test error might be very large because of the lack of training examples. From example 1, it can be inferred the test error is far more than the training error. Which is a bad split.

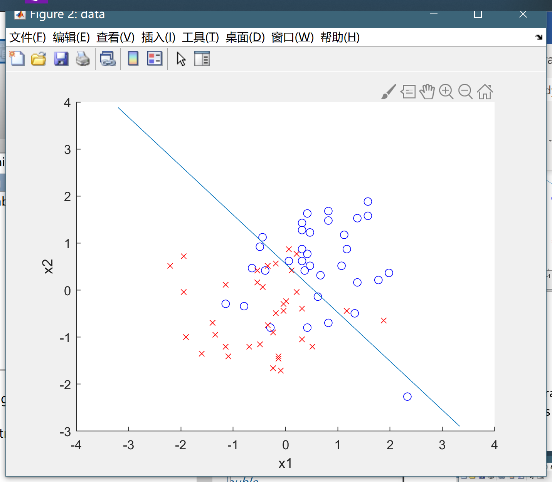
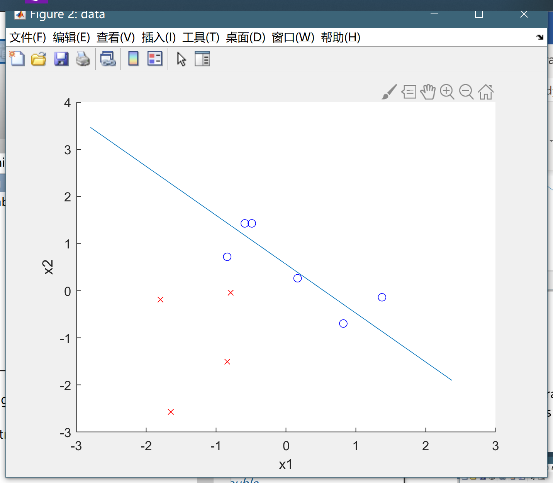
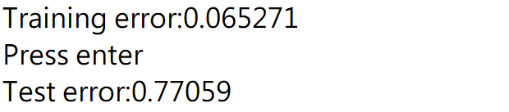
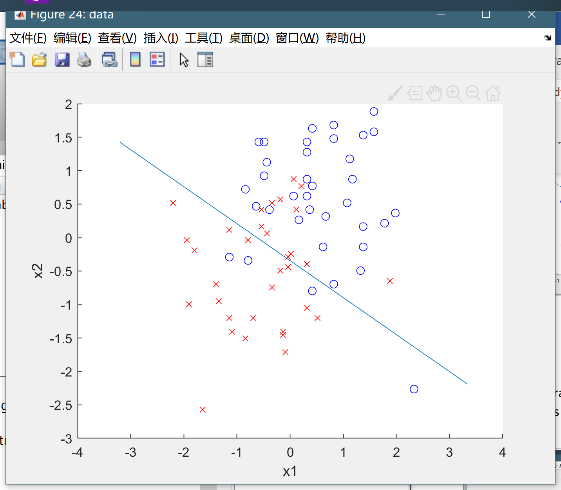
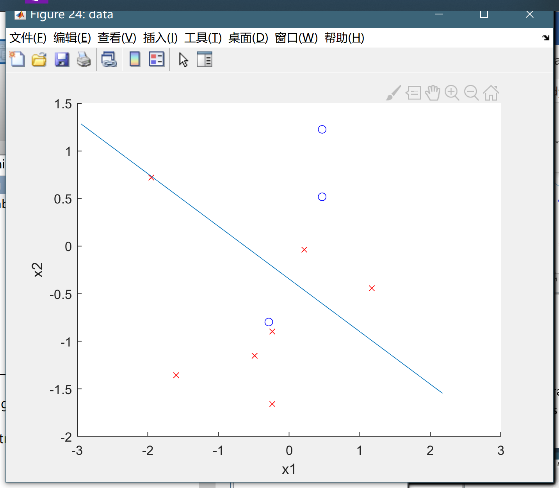
  


figure 19 task 6 example 1

from example2, the training set decided a boundary has familiar test error as the training error. The test error is familiar as the final error in **task4.** This could be considered as a good split.



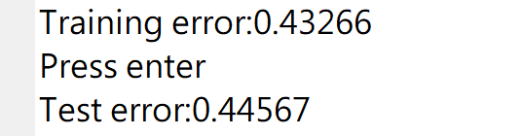


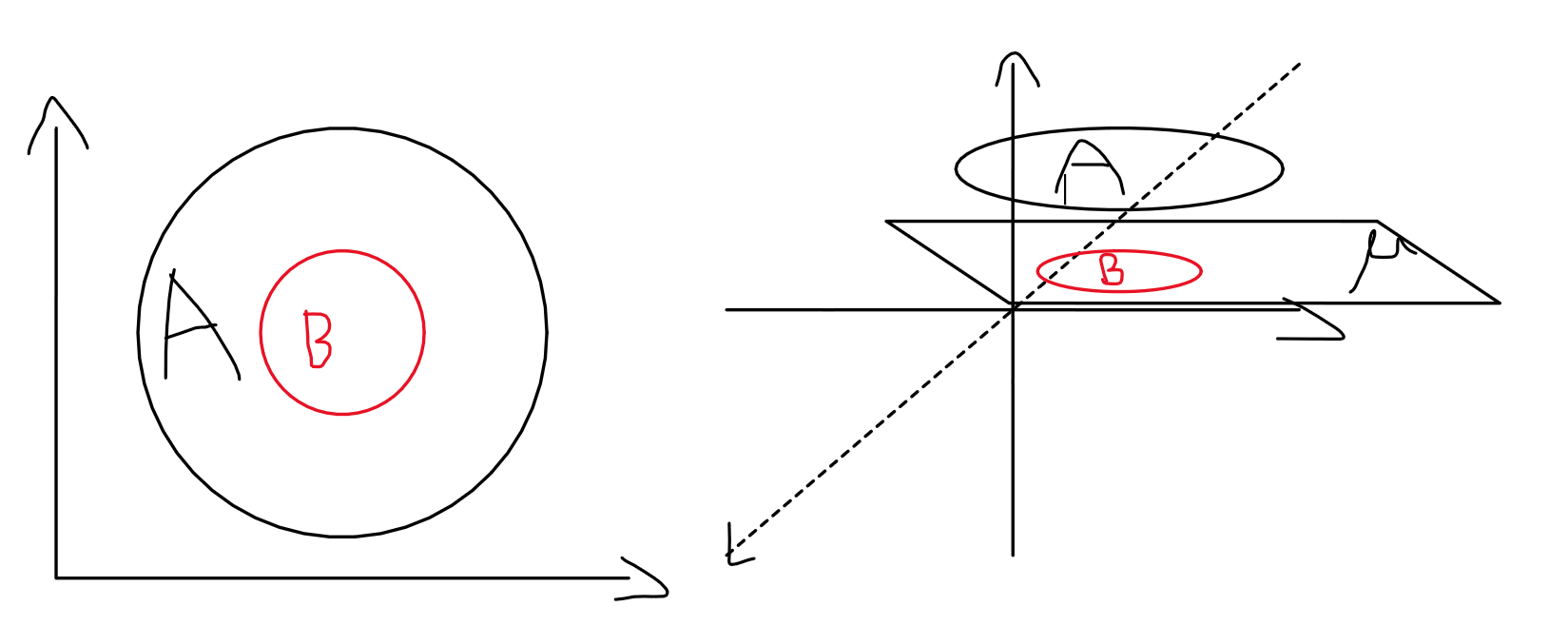
figure 20 task 6 example 2

**Task 7.** Run logistic regression on this dataset. How does the error compare to using the original features (i.e. the error found in Task 4)? Include in your report the error and an explanation on what happens.

[5 points]



Before task7, more dimensions are added into X. For some of the dataset, adding more dimensions could be effective to divide the dataset. The result has been shown below.



Imagine there are dataset which could be divided into A and B, in a 2-D plain, the decision boundary is a circle, which is nonlinear. However, when we expand it into a 3-D space, it could be easily divided by a plain.

However, for some of the dataset like the one using in **task7**, it may not have effect on the error.

**Task 8**. Add extra features (e.g. a third order polynomial) and analyse the effect. What happens when the cost function of the training set goes down but that of the test set goes up?

[5 points]

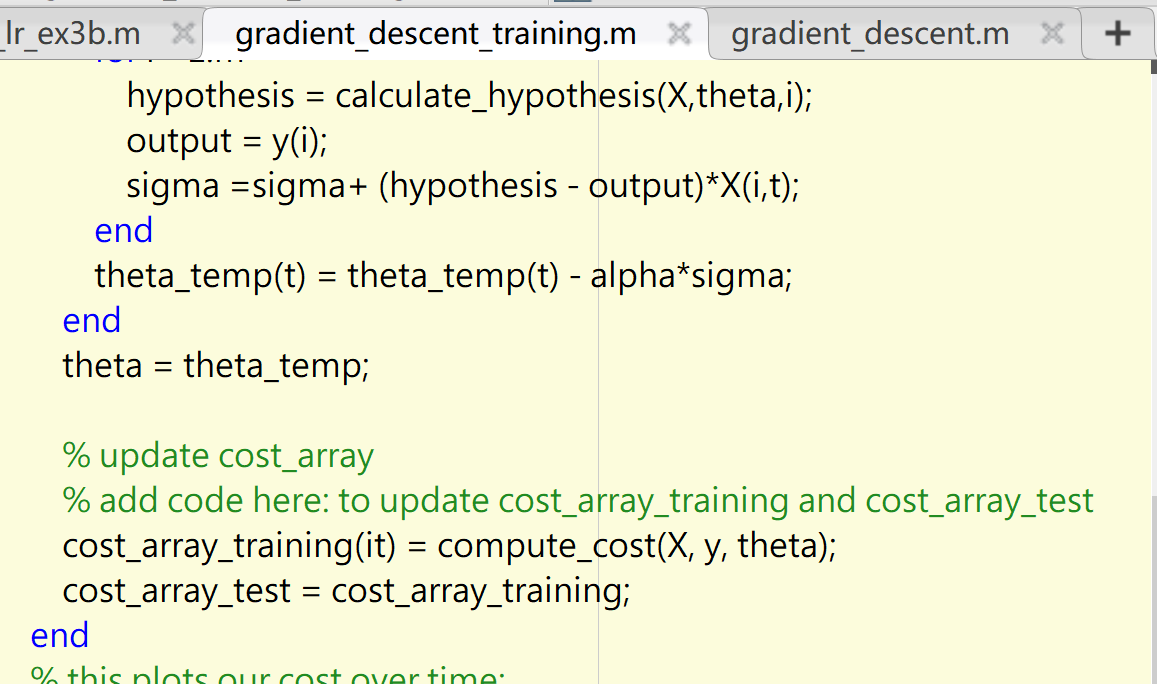


figure 21 modified code for gradient\_descent\_training()

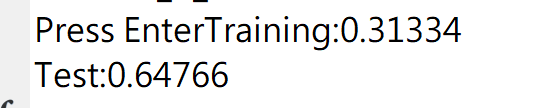
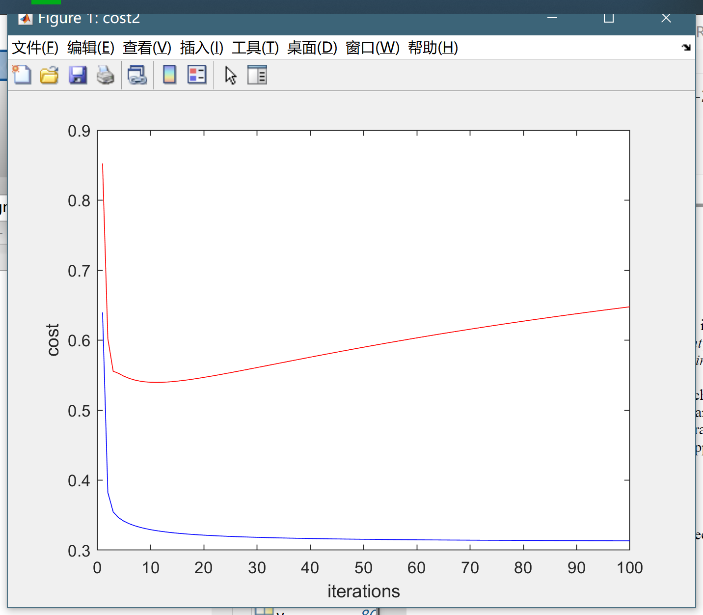


figure 22 figure for 20 training points

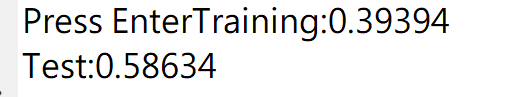
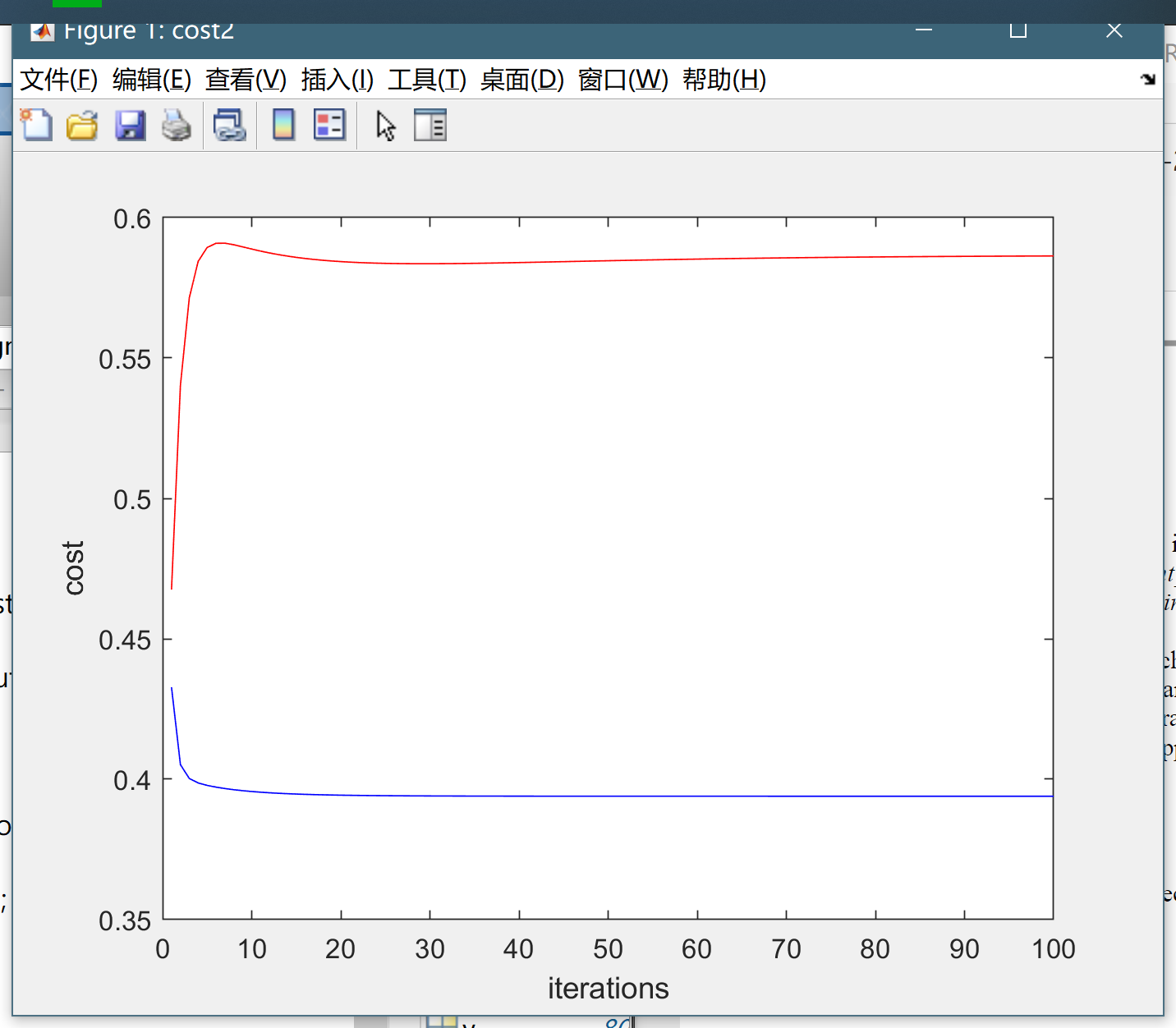


figure 23 figure for 40 training points

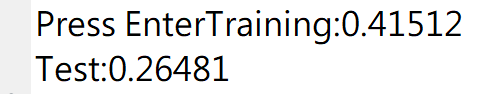
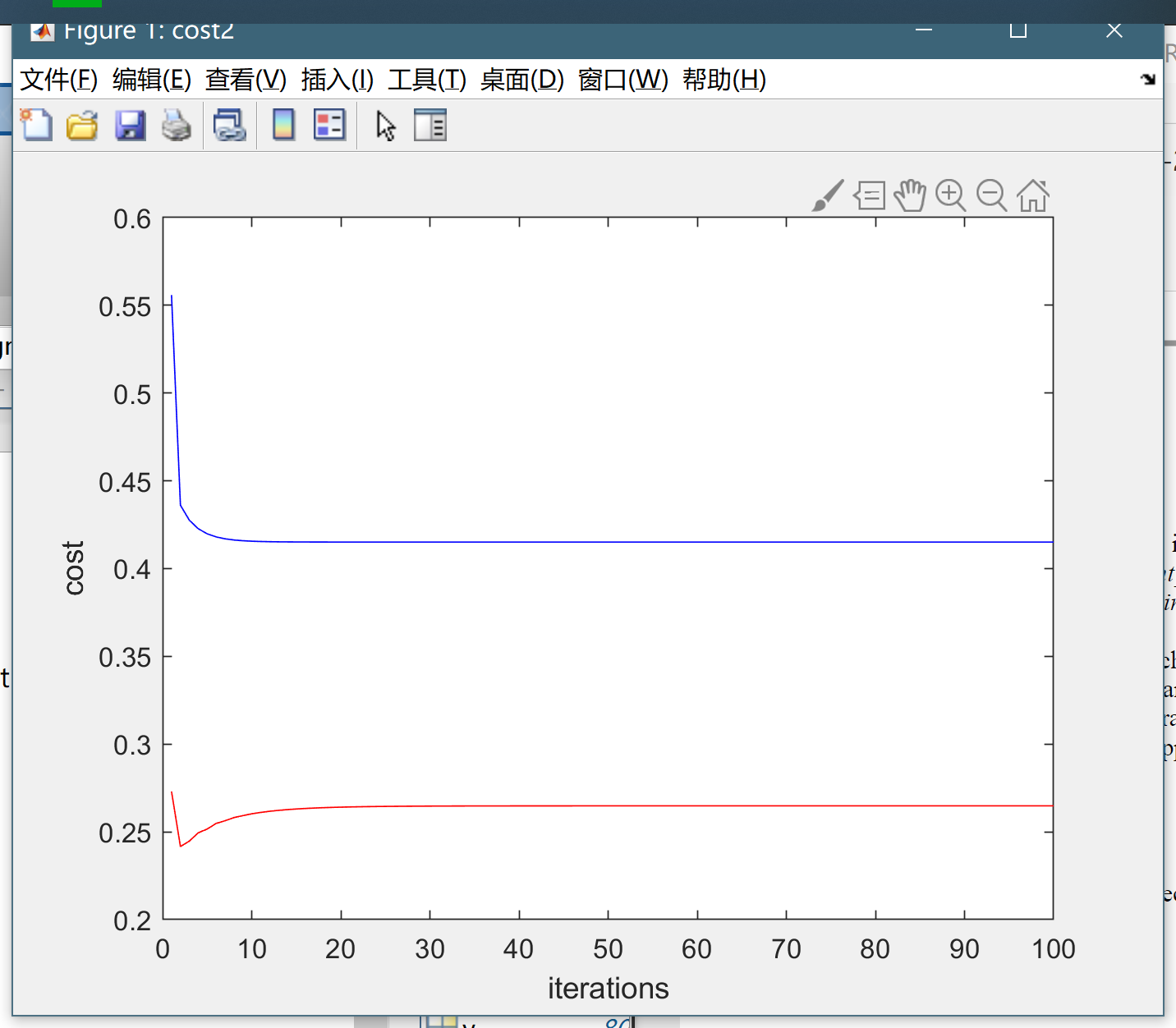


figure 24 figure for 70 training points

It can be decided that overfitting is occurred when training set goes down but that of the test set goes up.

**Task 9.** With the aid of a diagram of the decision space, explain why a logistic regression unit cannot solve the XOR classification problem.

[3 points]

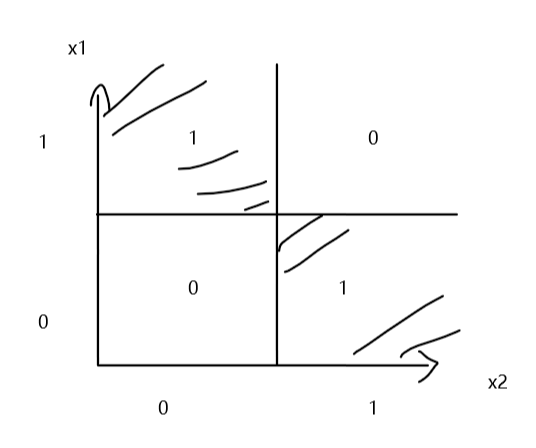


figure 25 visualized XOR

The decision space of XOR classification is divided by two decision boundaries. However, one logistic regression process could only decide one continuous decision boundary. As a result, it’s impossible for it to solve XOR problem.

## 2. Neural Network

**Task 10**. Implement backpropagation. Although XOR only has one output, this should support outputs of any size. Do this following the steps below:

[5 points]

**Step 2.**

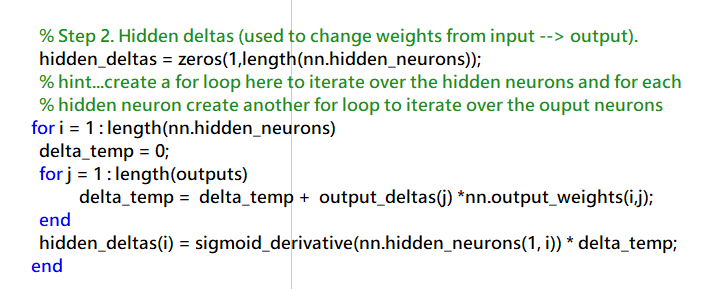


figure 26 task10 step2

**Step 4.**

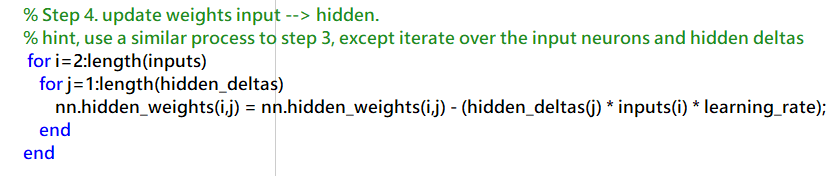
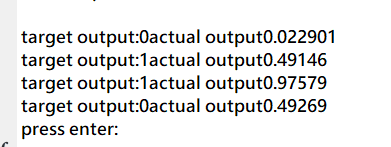
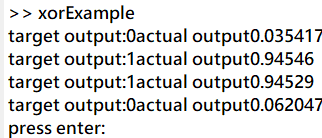
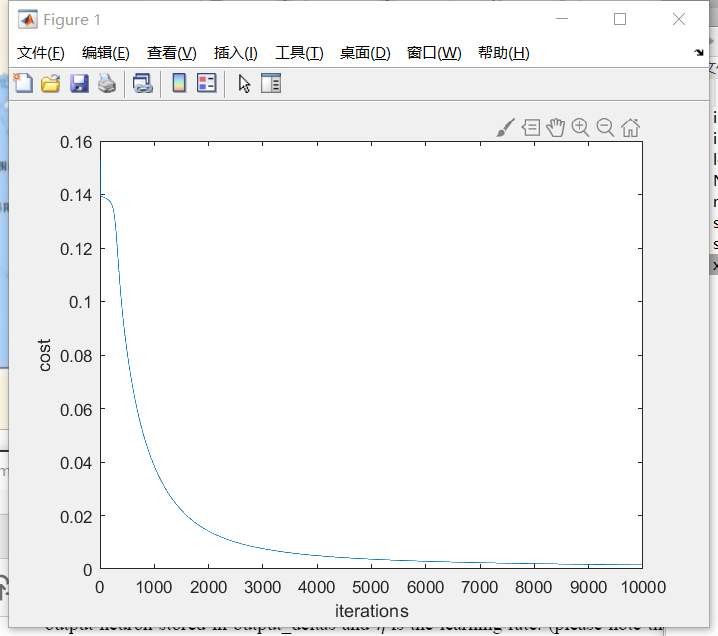


figure 27 task10 step4

the best learning rate is 1. When laring rate is 0.5, it may stuck into a local optimal and one of the target output cannot reach 1 or 0.

figure learning rate = 0.5 (local optimal)

figure learning rate = 1



**Task 11**. Change the training data in ***xor.m*** to implement a different logical function, such as NOR or AND. Plot the error function of a successful trial.

[5 points]

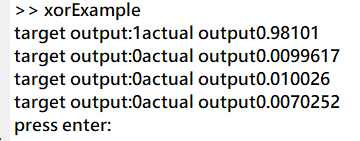
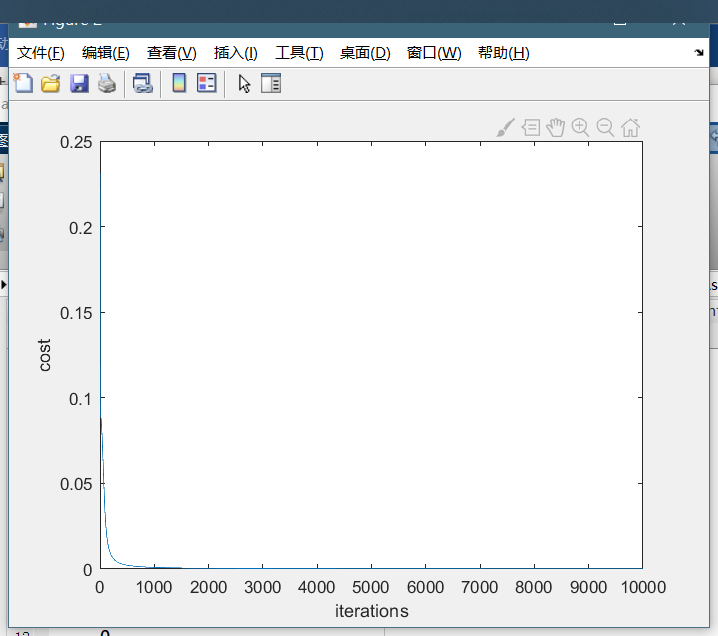


figure NOR, learning rate = 1

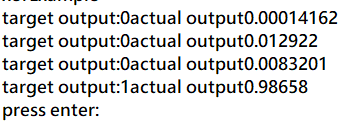
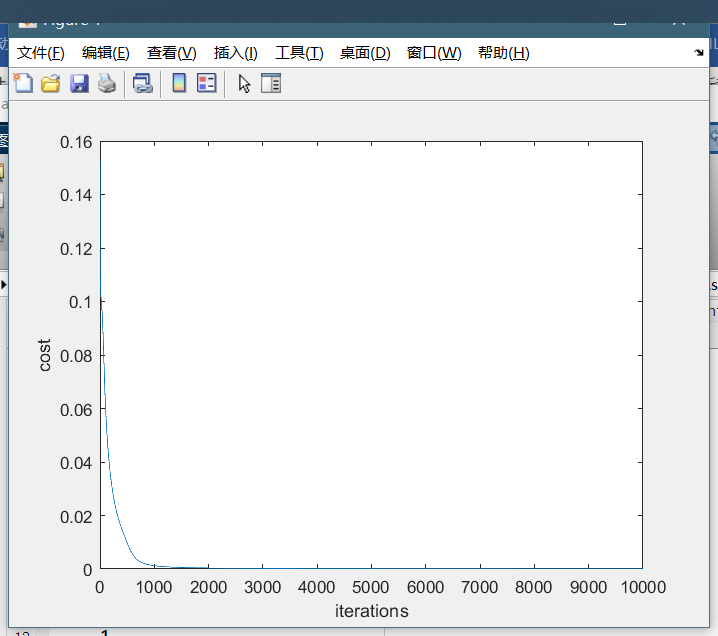


figure AND, learning rate = 1

**Task 12.** The Iris data set contains three different classes of data that we need to discriminate between. How would accomplish this if we used a logistic regression unit? How is it different using a neural network?

[5 points]

The logistic regression could only describe two classifications at one time. Because of the four categories in iris dataset, if classify it by logistic regression, we must classify one classification at one time. It is shown in figure 32.

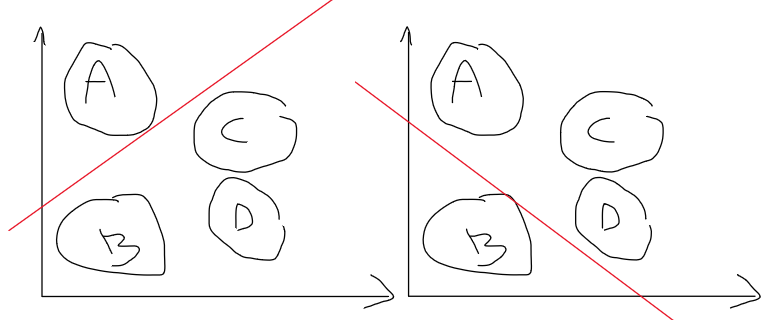


figure 32 classification example

after four classification and got 4 decision boundaries, we will obtain a joint boundary which classify all the categories.

**Task 13.** Run ***iris.m*** using the following number of hidden neurons: 1, 2, 3, 5, 7, 10. The program will plot the costs of the training set (blue) and test set (red) at each iteration. What are the differences for each number of hidden neurons? Which number do you think is the best to use? How well do you think that we have generalized?

[5 points]

Firstly, the number of neurons has a significant impact on errors. When hidden neuron = 1, the error is larger than more numbers of neuron.

Secondly, when there are too many neurons, the training error might go down, however, the testing error may not go down. The main reason of this is overfitting.

The best number of hidden neurons is 3.

As the number of training set, which is 75, isn’t large, the testing error may go down when analyzing more data.

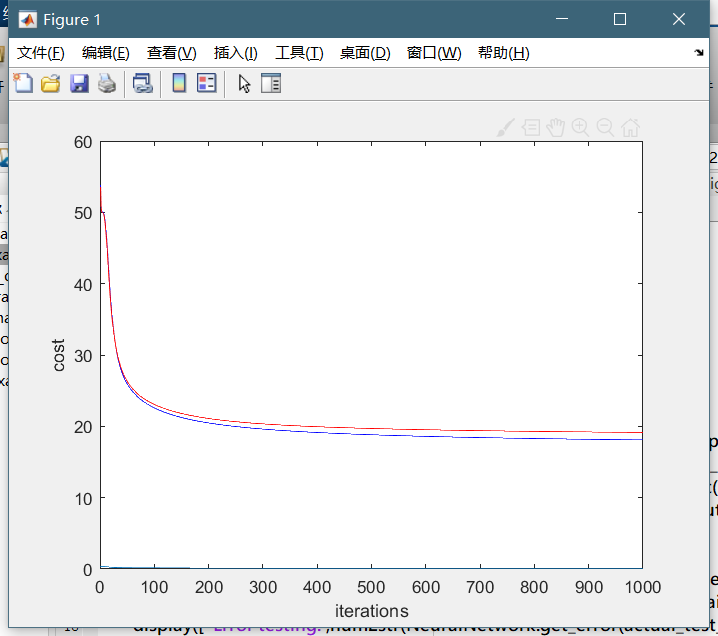


figure 33 hidden neuron = 1

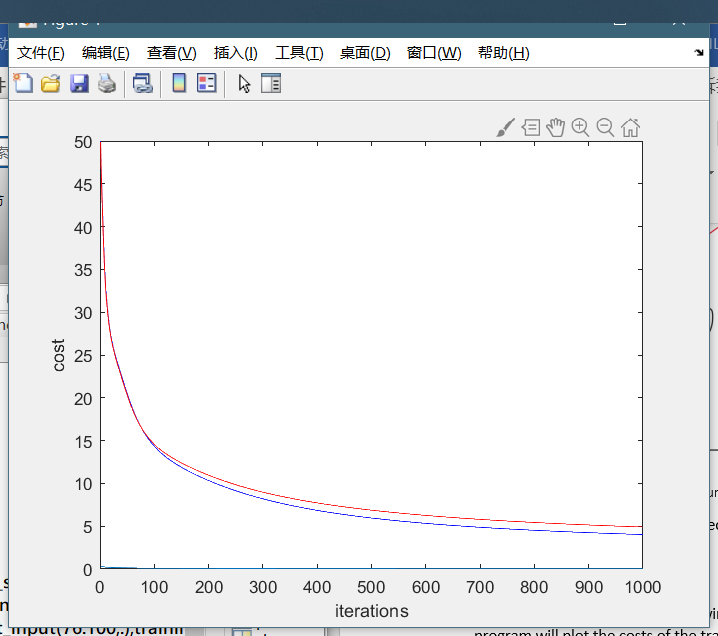
 

figure 34 hidden neuron = 2

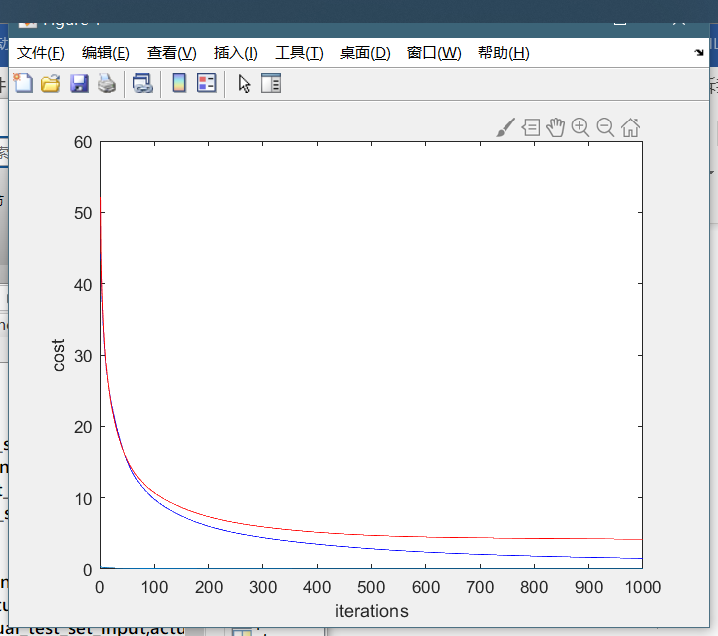
 

figure 35 hidden neuron = 3

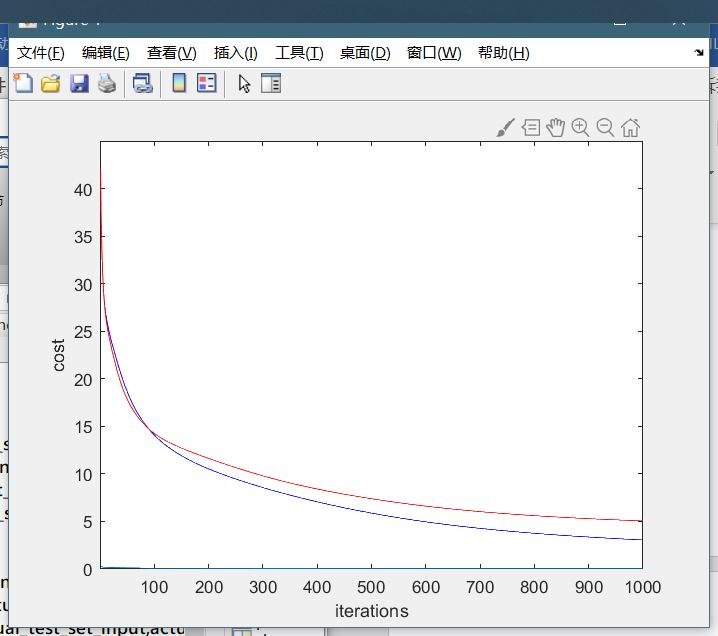
 

figure 36 hidden neuron = 5

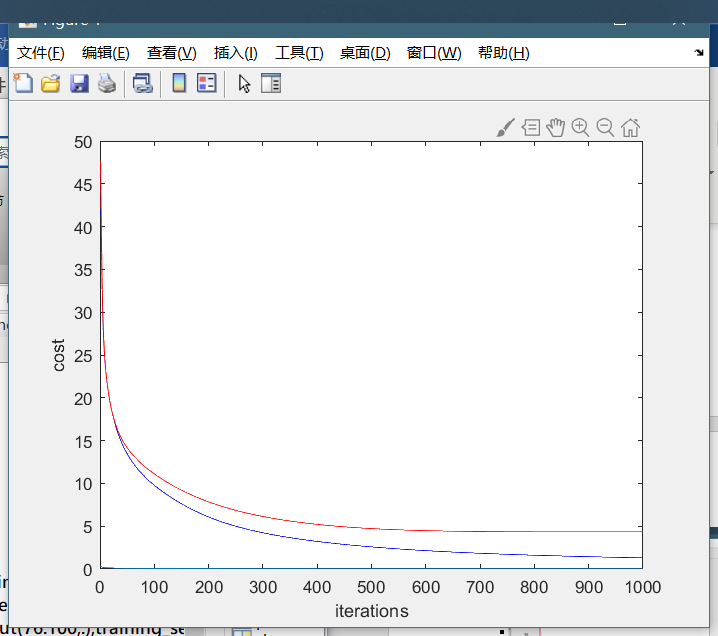
 

figure 37 hidden neuron = 7

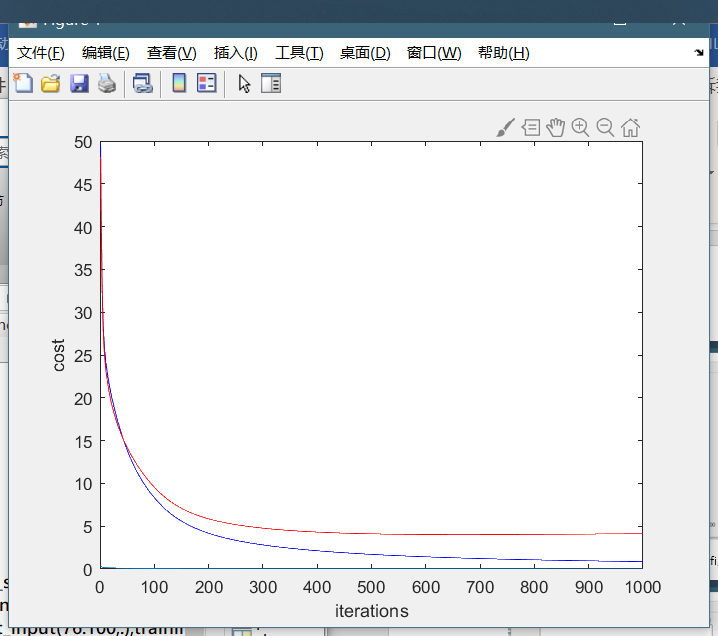
 

figure 38 hidden neuron = 10