ME 5404 Neuron Networks Homework #2

A0263252L Shen Xiaoting

Q1

a) Considering the Rosenbrock's Valley function has a global minimum at $(x, y) = (1,1)$ where $f(x, y) = 0$

$$f(x, y) = (1 - x)^2 + 100(y - x^2)^2$$

We can get the gradient vector $g(n) = \nabla E(w(n))$

$$\frac{\partial f(x, y)}{\partial x} = 2(x - 1) - 400x(y - x^2)$$

$$\frac{\partial f(x, y)}{\partial y} = 200(y - x^2)$$

Through the steepest descent method, we can get the iteration equation as

$$w(n + 1) = w(n) - \eta g(n)$$

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \end{bmatrix} - \eta \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

If the $f(x, y) \leq 0.0001$, we consider that it converges. Then we can get the iteration times for gradient descent method. The figures are shown below for the 3 dimension and 2 dimension. The trajectory is plotted by red line and the black dots are the steps for each iteration as shown in Figure 1.1.

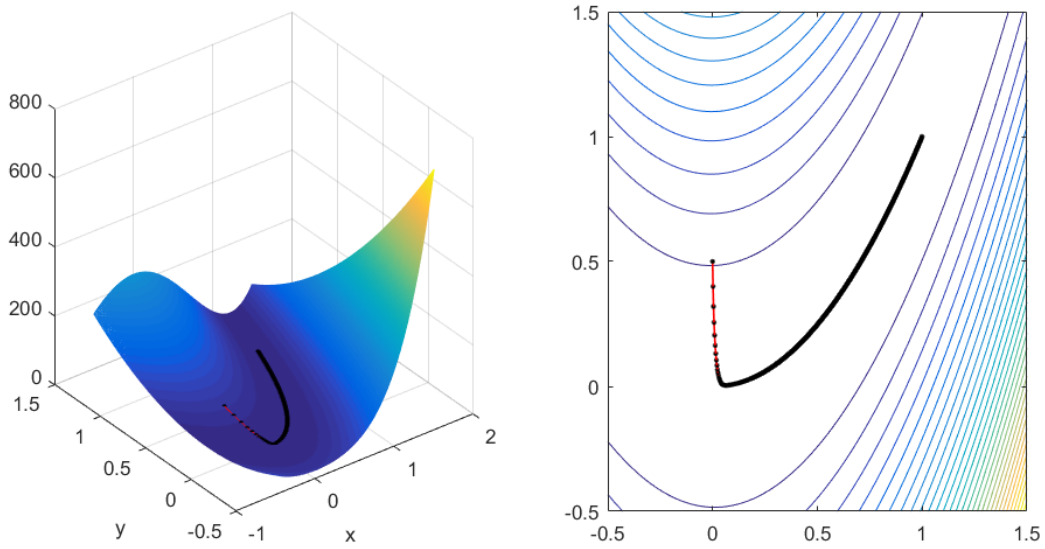


Figure 1.1 Trajectory of iterations in 3 dimension space and contour graph with learning rate = 0.001

If $f(x, y)$ is close to 0 (less than 0.0001), it will do 8564 iterations with learning rate 0.001. We change the learning rate to 0.0002, and the trajectory is similar to Figure 1.1 and it will converge to 0 with 87551 iterations. If we change the learning rate to 0.005, it will not converge in 100000 iterations and have large oscillation and the figure is shown in Figure 1.2. If the learning rate is large, it will converge at a lower speed and even not converge because of the large oscillation.

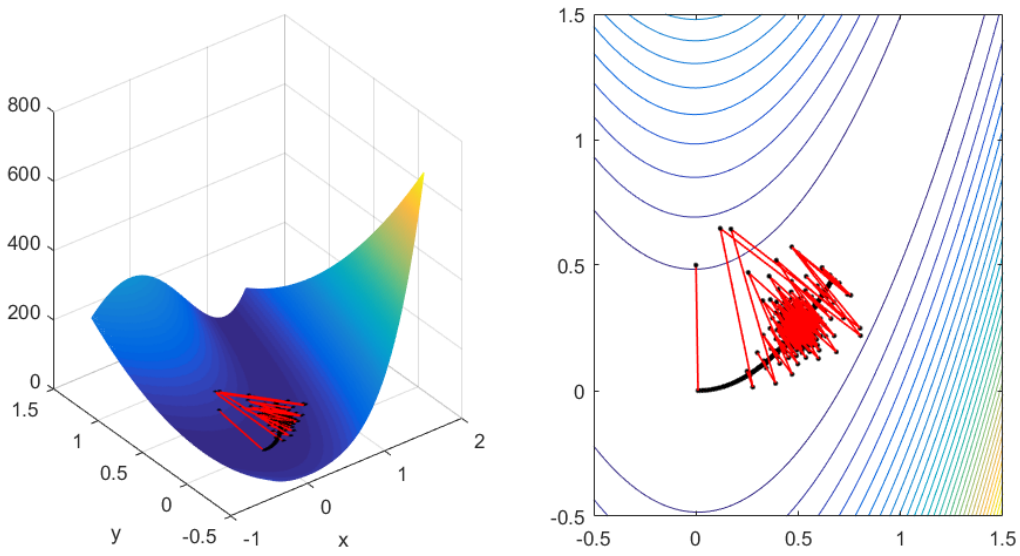


Figure 1.2 Trajectory of iterations in 3 dimension space and contour graph with learning rate = 0.005

b) We use the Newton's method to get the minimum of Rosenbrock's Valley function. The equation is

$$\Delta w(n) = -H^{-1}(n)g(n)$$

The Hessian matrix:

$$H(f) = \begin{bmatrix} \frac{\partial f(x,y)}{\partial x^2} & \frac{\partial f(x,y)}{\partial x \partial y} \\ \frac{\partial f(x,y)}{\partial y \partial x} & \frac{\partial f(x,y)}{\partial y^2} \end{bmatrix} = \begin{bmatrix} 1200x^2 - 400y + 2 & -400x \\ -400x & 200 \end{bmatrix}$$

Then we can get the iteration equation

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \end{bmatrix} - H(f)g(n)$$

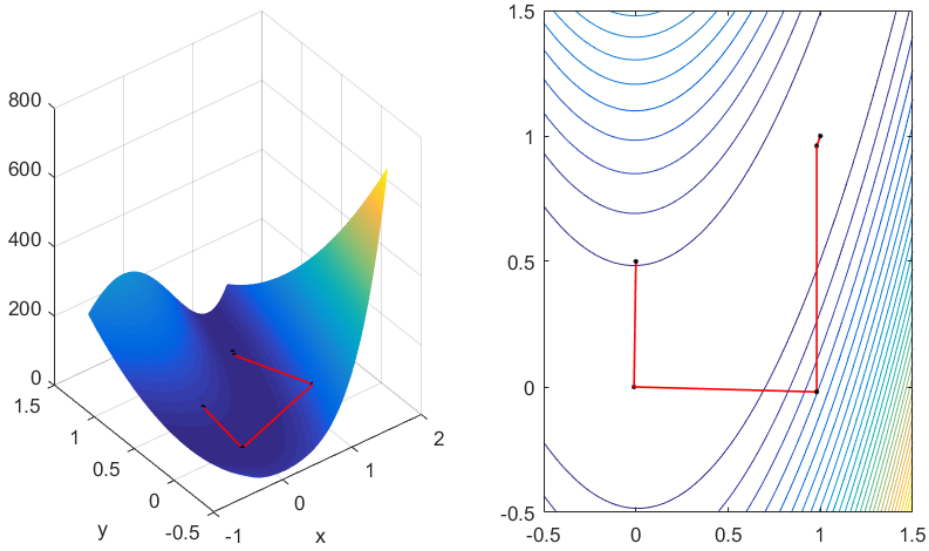


Figure 1.2 Trajectory of iterations in 3 dimension space and contour graph with Newton method

If $f(x,y)$ is close to 0 (less than 0.0001), it will do 5 iterations. Newton method is much faster than the gradient method to approach global minimum.

Q2:

We need to use MLP to approximate the following function with sequential mode and batch mode

$$y = 1.2sin(\pi x) - cos(2.4\pi x)$$

a) We use the sequential mode with BP algorithm and experiment with different structures of MLP 1-n-1, n=1,2,…,10,20,50.

We plot the outputs of the MLP for the test samples after training and plot the true function from the range of -3 to 3 of Figure 3.1.

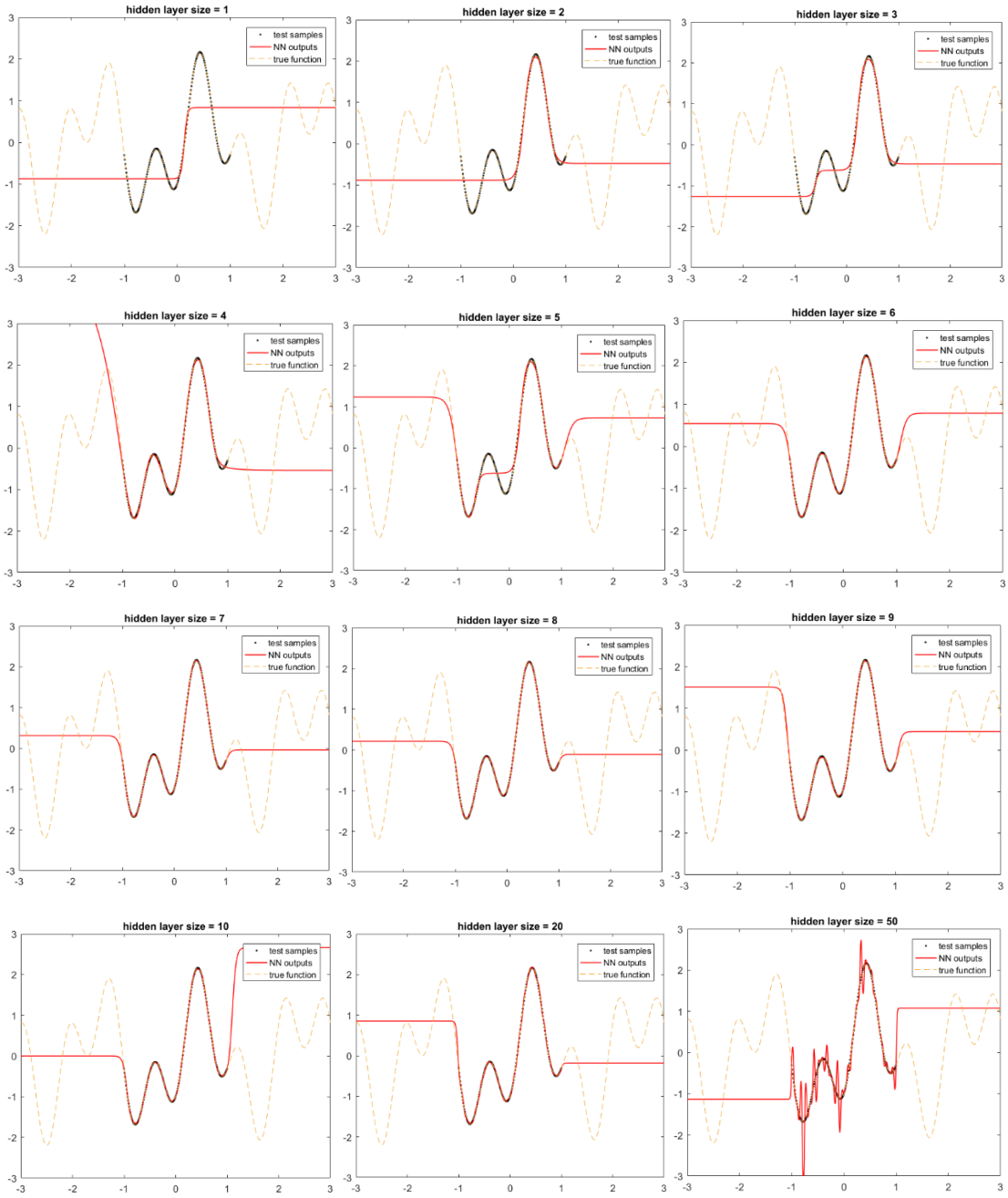


Figure2.1 Sequential mode to approximate the function results

We can observe that if the hidden layer size less than 6, the figure is under-fitting. MLP with 6 hidden layers is the most proper fitting to the function. We also calculate the MSE of the testing

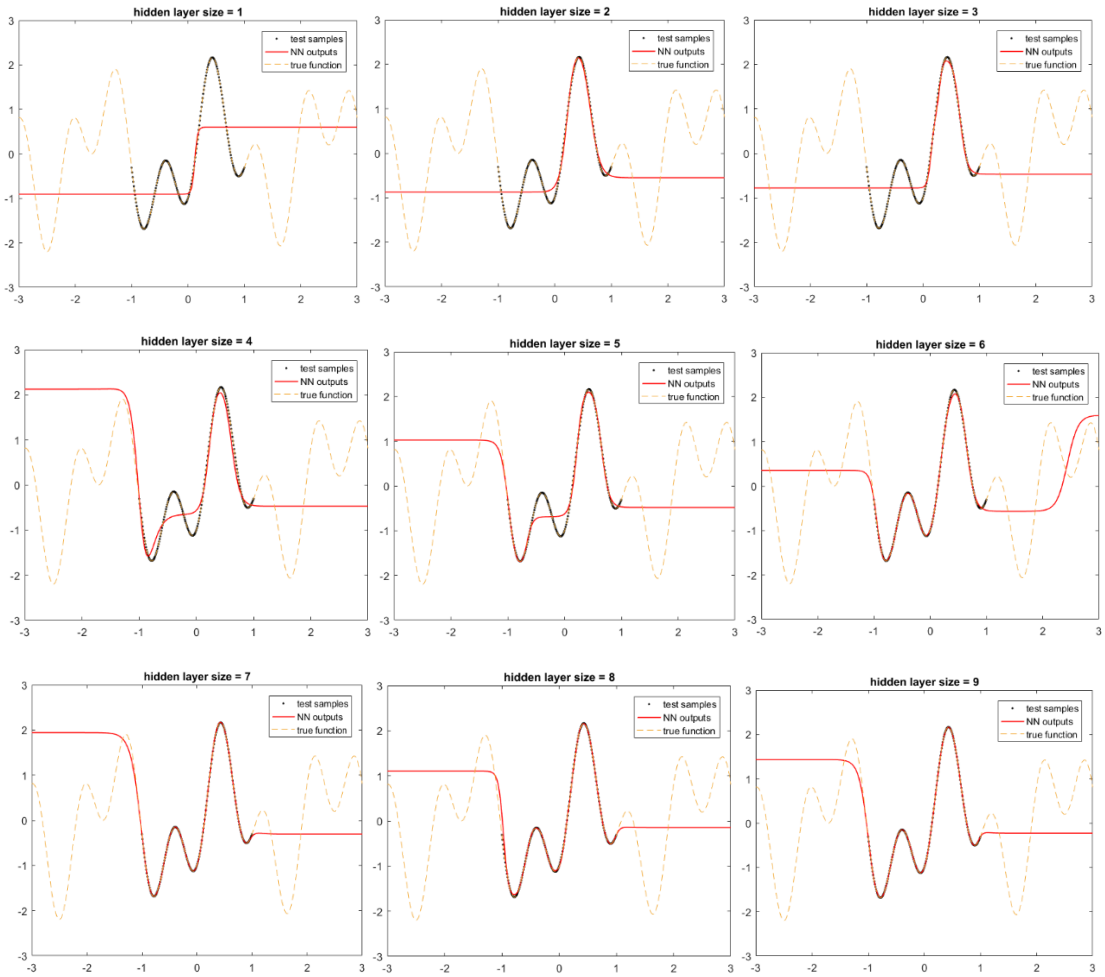
samples and get the results in table 2.1. It also indicates that the hidden layer size between 6 and 10 is proper fitting, and it has a small error. If there are too many layers to do function approximation, the outputs show that it is over-fitting. The outputs of the MLP when x=-3 and x=3 has a big difference to the original function. MLP cannot do reasonable predictions outside the domain of the input limited by the training set.

Table 2.1 Sequential mode for function approximation errors

Hidden layer size	1	2	3	4	5	6
Error(mse)	0.5105	0.1226	0.0712	0.0024	0.0423	4.82×10^{-5}
Hidden layer size	7	8	9	10	20	50
Error(mse)	1.43×10^{-4}	1.52×10^{-4}	3.6×10^{-4}	4.3×10^{-4}	0.0012	0.1202

b) We use the batch mode with trainlm algorithm to approximate the function

We tried to change the hidden layer size to approximate the function and get the outputs as Figure 2.2.



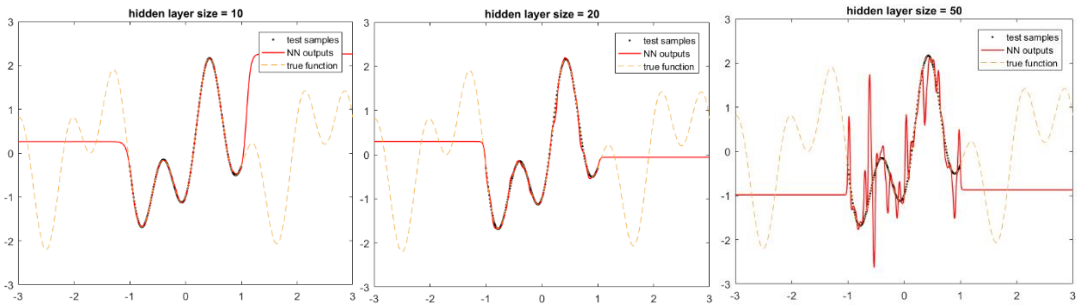


Figure 2.2 Batch mode(trainlm) to approximate the function results

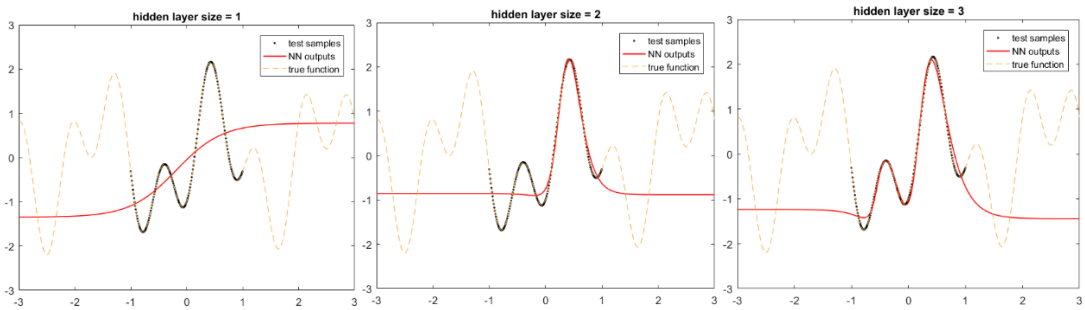
It is under-fitting if the hidden layers are less than 5. The layers between 6 to 10 is proper fitting and the layers over 10 are over-fitting. We also calculate the MSE of the testing samples and get the results in table 2.2. It also indicates that the hidden layer size between 6 and 10 is proper fitting, and it has a small error, while the others have a large error. The minimal number of line segments that can construct the basic geometrical shape of the target function is 5 which is consistent with the output of MLP. We can also get the outputs of the MLP when x=-3 and x=3. It is obvious that it has a big error with the actual value. MLP can't make reasonable predictions beyond the range of the training set.

Table 2.2 Batch mode(trainlm) for function approximation errors

Hidden layer size	1	2	3	4	5	6
error	0.5143	0.1232	0.1274	0.0670	0.0449	8.9×10^{-4}
Hidden layer size	7	8	9	10	20	50
error	1.1×10^{-4}	1.7×10^{-9}	7.97×10^{-4}	2.31×10^{-5}	0.0182	0.4534

c) We use the batch mode with trainbr algorithm to approximate the function.

We change the hidden layer size to approximate the function and get the outputs as follows. It is under-fitting if the hidden layers are less than 5. The layers between 6 to 10 is proper fitting and the layers over 10 are over-fitting. The minimal number of line segments that can construct the basic geometrical shape of the target function is 5 which is consistent with the output of MLP. We can also get the outputs of the MLP when x=-3 and x=3. It is obvious that it has a big error with the actual value. MLP can't make reasonable predictions beyond the range of the training set.



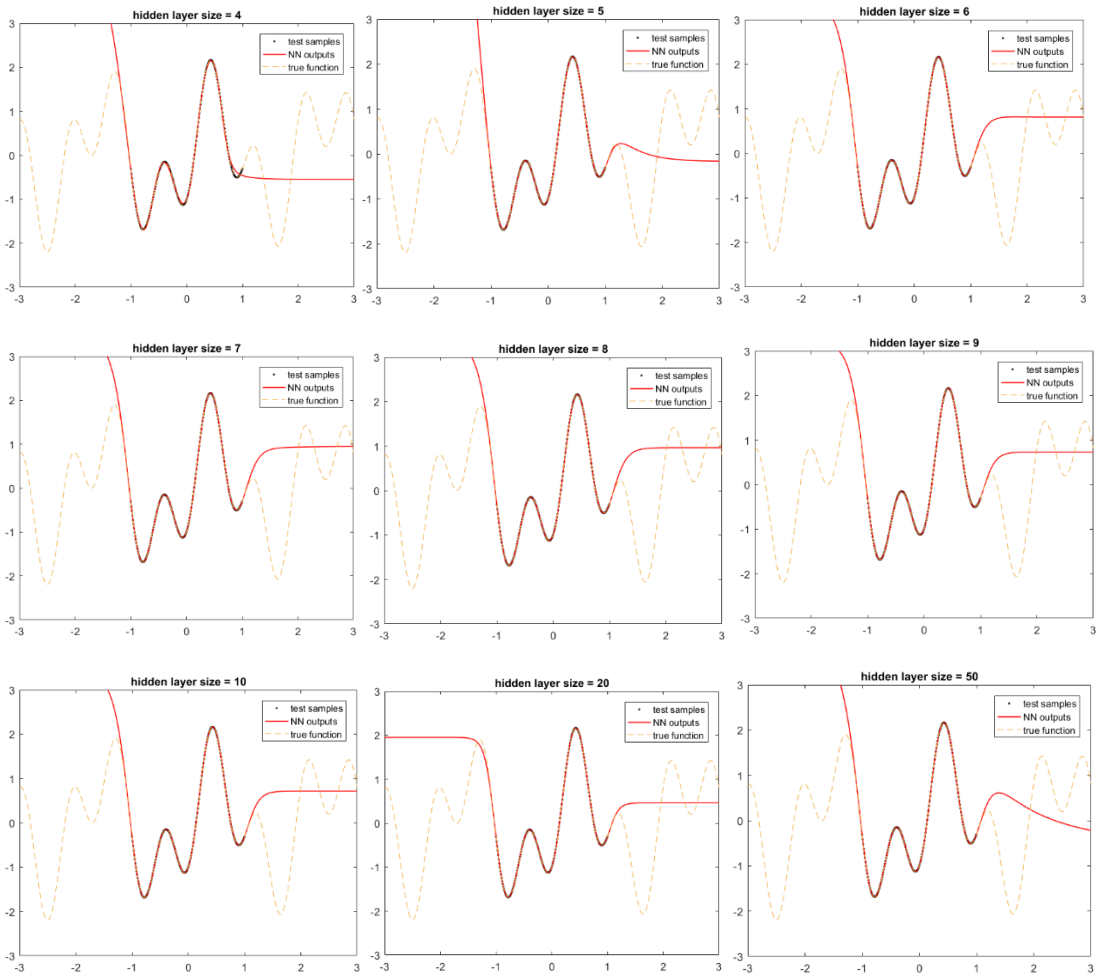


Figure 2.3 Batch mode (trainbr) to approximate the function results

We also calculate the MSE of the testing samples and get the results in table 2.3. It also shows that the hidden layer size between 6 and 10 is proper fitting, and it has a small error, while the others have a large error which indicates that it is overfitting or underfitting.

Table 2.3 Batch mode(trainbr) for function approximation errors

Hidden layer size	1	2	3	4	5	6
error	0.7021	0.1313	0.0367	0.0016	6.5×10^{-6}	1.02×10^{-8}
Hidden layer size	7	8	9	10	20	50
error	1.18×10^{-9}	8.23×10^{-9}	1.2×10^{-10}	2.6×10^{-10}	1.37×10^{-7}	9.02×10^{-7}

Q3:

a) Rosenblatt's perceptron (single layer perceptron) to do pattern recognition of the dataset

The single layer perceptron can solve the linearly separable pattern recognition problem. The dataset

was assigned for ‘group1’ to distinguish the open country and highway pictures. Each picture with size 256*256 can be transformed to a vector with 65536 inputs. We have 503 train images and 167 test images. We build the single layer perceptron and do learning with learning rate 0.01. The training accuracy is around 68% and the testing accuracy is around 65%. After 4000 iterations, the training accuracy will be close to 100% and it shows the overfitting problem. It has a large oscillation as shown in Figure 3.1(1).

b) Apply Rosenblatt’s perceptron to down sampled images

We used the ‘imresize’ function in MATLAB to do down sampling. We change the size of the images to 128*128, 64*64, 32*32. Apply the Rosenblatt’s perceptron to train the down sampled images and get the training accuracy and testing accuracy as shown in Figure 3.2(2)-(4).

We listed the table 3.1 below to make comparison between different size of images. It is observed that the larger size of the image, the less iteration times for the images classification to do proper fitting. Although the algorithm for training images with lager size has less iterations times, it will spend more time for training. The testing accuracy will be lower with the down sampling, but have not much difference.

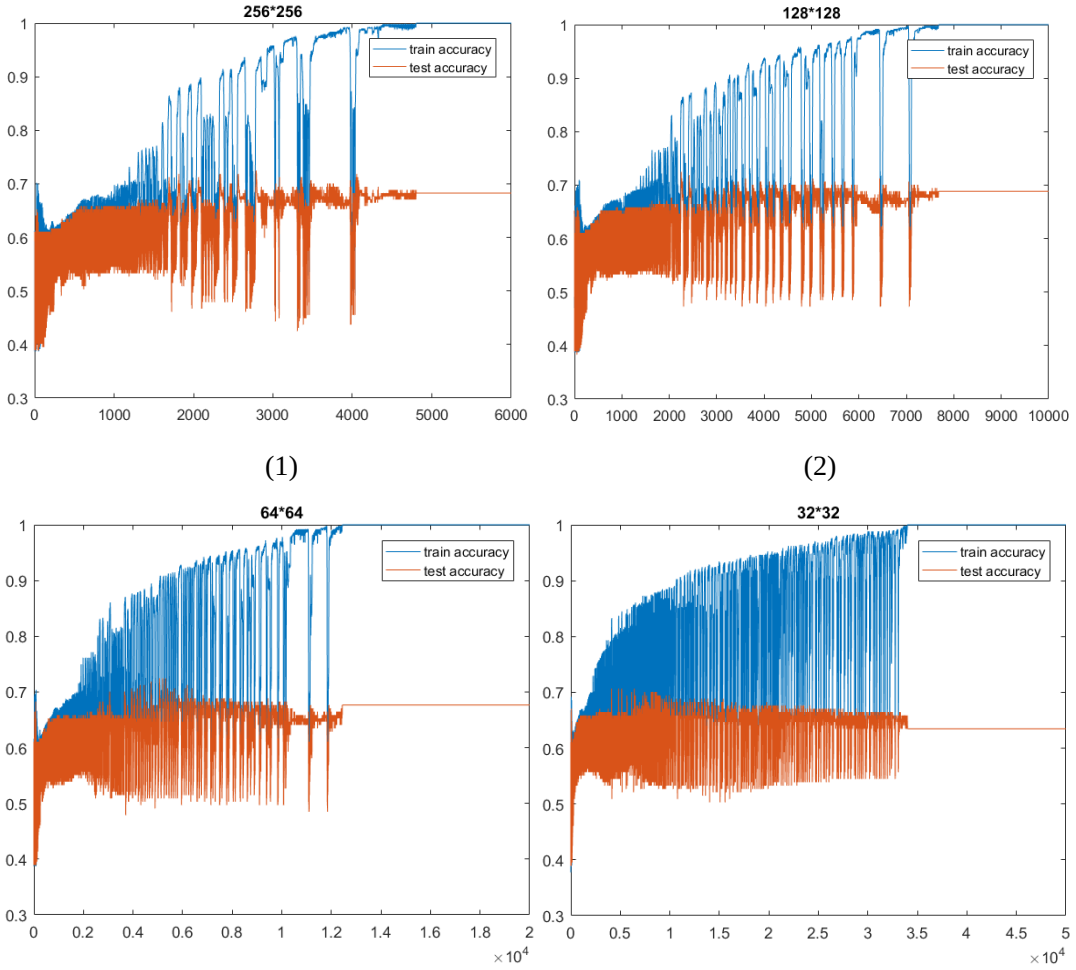


Figure 3.1 The train accuracy and test accuracy of imresize

Then we use the PCA method to images down sampling. It orthogonalized the components of the vectors of the image and orders the resulting orthogonal components so that those with the largest variation come first and eliminates those components which contribute the least to the variation of the vectors in the image. We apply this principal component analysis to our images with the size of 540 instead of 256*256. Within first 100 iterations, the testing accuracy is high and we consider it as a proper fitting area. The training accuracy is near 80% and testing accuracy is about 78%, which is much higher than the imresize method, shown in figure 3.2. The time it spends is much less and the oscillation is smaller than the other method.

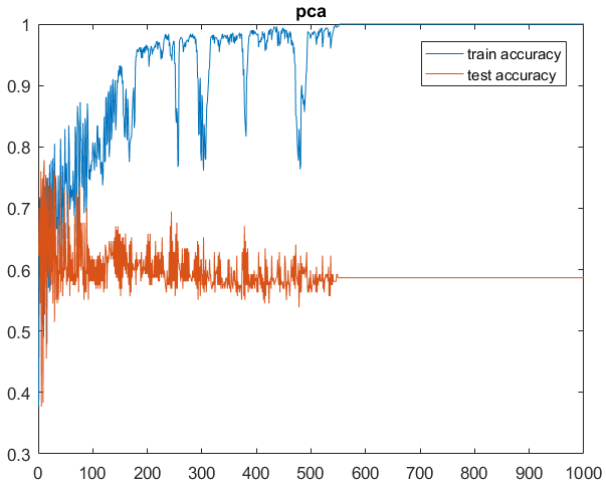


Figure 3.2 The train accuracy and test accuracy with PCA

Table 3.1 Comparison of different size of images with accuracy

Size of the image	Iteration times	Train accuracy	Test accuracy
256*256	1000	0.6725	0.6483
128*128	1000	0.6638	0.6423
64*64	2000	0.6723	0.6558
32*32	5000	0.6643	0.6382
PCA	40	0.7428	0.7632

c) & d) Apply MLP with batch mode, Overfitting analysis and weights regularization

We assign the hidden layers of the MLP as 10 and for 115 epochs. The training accuracy is 83.90% and testing accuracy is 76.05%.

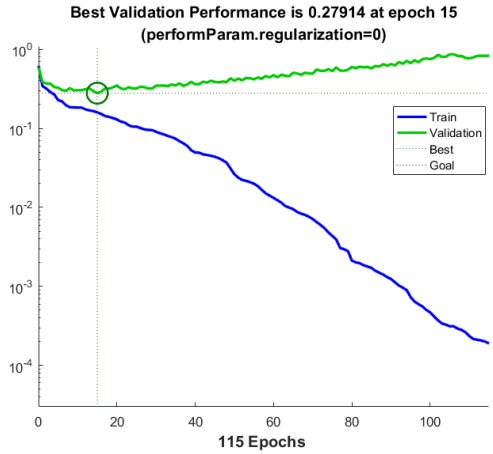
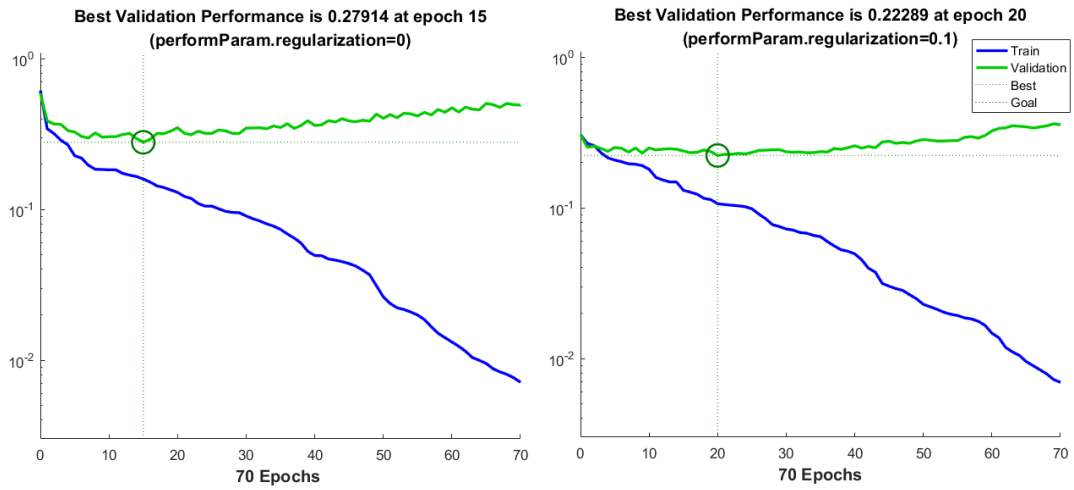


Figure 3.3 The MLP performance of Cross-Entropy

The training MLP in c) is overfitting. We plot the figure of 3.3 and the best validation performance is shown in epoch 15. It is clear that the training loss decent with increasing epochs. Before the epoch 15, the validation loss is decreasing and the network keeps on learning. After the epoch 15, the validation loss keeps on increasing with epochs and the overfitting problem is shown.

I tried to do weight regularization with setting the 'performParam.regularization' equal to 0.1, 0.5, 0.8 and get the learning results as shown in figure 3.4. The detail of the figures are shown in table 3.2. With tuning the regularization parameter, we can observe that he validation cross-entropy increases at a lower speed with the decreasing of the training cross-entropy. The gap between the best validation performance and the performance at epoch 70 will be smaller with bigger regularization parameters



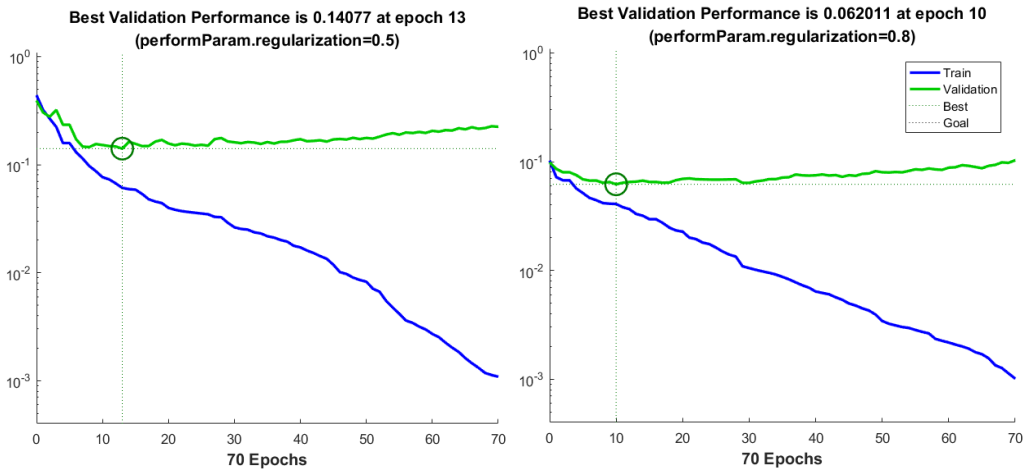


Figure 3.3 The MLP performance by tuning the weights of Cross-Entropy

Table 3.2 Performance of the MLP with different regularization parameters

Parameters tuning performParam.regularization	Best validation performance		Cross-entropy at epoch 70	Cross-entropy difference
	Cross-entropy	Epoch		
0	0.2791	15	0.4931	0.2140
0.1	0.2229	20	0.3569	0.1340
0.5	0.1408	13	0.2243	0.0835
0.8	0.0620	10	0.1031	0.0441

e) After applying the sequential mode for pattern recognition of the images, we get the result of training accuracy and testing accuracy of 87.48% and 67.78%. The classification accuracy is worse than training by batch mode. Sequential learning, on the other hand, involves training the model incrementally, one observation at a time. After each observation, the model is updated and the weights are adjusted to minimize the error. Incremental learning is faster and more memory-efficient than batch learning, as it does not require storing the entire dataset in memory. However, it can be more prone to overfitting and may require more fine-tuning to achieve optimal performance.

Batch learning involves training the model using the entire dataset at once. The model is updated after the entire dataset has been processed, and the weights are adjusted based on the overall error. This approach requires significant computational resources and is typically slower than incremental learning. However, it can result in a more accurate model if the dataset is not too large and if the model architecture is appropriate.

f) Through the several steps and try and errors above, I have learned that many data pre-processing methods, training parameters, model selection will affect the performance of MLP.

(1) Training data pre-processing

Increase the size of the training data can help improve the generalization performance of the neuron network.

We can also do sampling dropout and down sampling to reduce the inputs to the network. PCA is a good metho to get the principle feature of the data to reduce the size of the data.

(2) Mode selection

Different mode will have their advantages and disadvantages. The sequential mode and batch mode are two main methods for training. We can choose the proper mode according to the time limit and accuracy command.

(3) Analysis and regularization

After the training of the neuron networks, we get the results such as training accuracy, testing accuracy, errors and so on. We should analysis the results and evaluate the model. Overfitting is a problem which will affect the performance of the network. Regularization can help prevent overfitting by adding penalty term to the error function. The number of hidden layers will affect the performance of the neuron network. If we don't get the expected results, the size of the hidden layer may be incorrect.