EE5904 Neural Networks: Homework 2

Jin Lexuan

February 18, 2023

Q1.

a

Derive partial of function $f(x, y)$ with respect to variable x and y, and let them equal to zero:

$$\frac{\partial f(x,y)}{\partial x} = 2(x-1) + 400x(x^2 - y) = 0$$

$$\frac{\partial f(x,y)}{\partial y} = 200(y - x^2) = 0$$

So it can get: $y = x^2$ and $x = 1$, so the Rosenbrock's Valley function has a global minimum at $(x, y) = (1, 1)$ and $f(x, y) = 0$.

b

Figure 1' right part shows the trajectory of $f(x, y)$ converging to 0(threshold is set as 10^{-6}).It shows that at 14606th iteration, the value of $f(x, y)$ is smaller than the threshold and can be viewed as 0. Figure 1's left part shows the trajectory of x value and y value during the process of reaching global minimum. Gradient descent is very slow for this situation.

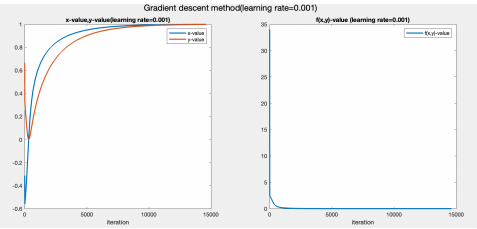

Figure 1: Gradient descent method(lr=0.001)

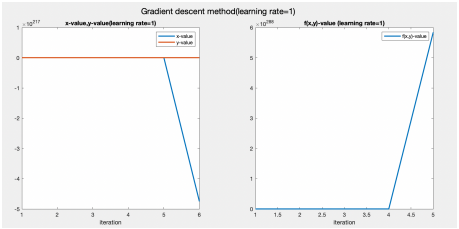

Figure 2: Gradient descent method(lr=1)

When the learning rate is set larger as 1. The result shows in Figure 2 that $f(x, y)$ will soon reach a very large value at the 6th iteration.y value keeps at zero and x value keeps decreasing to a negative number. This comparison of two results show that a large learning rate is not suitable for Steepest(Gradient) descent method.

c

Hessian matrix: $H(n) = \frac{\partial^2 E(w)}{\partial w^2}$,for this case, can get matrix:

$$\begin{bmatrix} 2 + 1200x^2 - 400y & -400x \\ -400x & 200 \end{bmatrix}$$

When Newton's method is applied for this process, Figure 3 shows the process of $f(x, y)$ reaching zero. It presents that the fluctuation of curve is very large. However, Newton Method only needs 6 iteration to reach zero. This result is in line with that second order method is much faster than gradient descent.

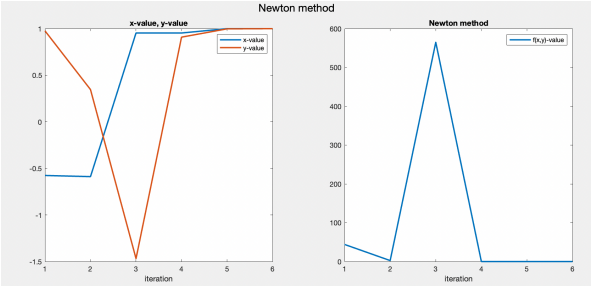

Figure 3: Newton Method

Q2

a

For the sequential mode, the results are shown from Figure 4 to Figure 16. From these figures, it shows when number of hidden neurons $(1, 2, 3, \cdots 10, 20, 50, 100)$ increases, number of piece-wise part of outputs also increases. Epoch time is set as 300. The fitting state of each situation is shown in Table 1. The result is that for sequential mode, when number of hidden neurons is under 10 (not included), it is under-fitting, when it is 10 or 20, it is proper fitting, when it is 50 or 100, it is over fitting. However, outputs can only describe the function in the range of training set $[-1.6, 1.6]$, they cannot make reasonable prediction out of the domain.

Hidden Neuron	1	2	3	4	5	6	7	8	9	10	20	50	100
Fitting State	under-fitting									proper fitting		over-fitting	

Table 1: Fitting state for different hidden neurons(sequential)

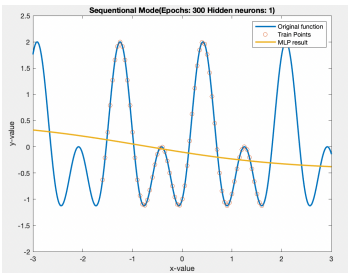

Figure 4: Sequential(1)

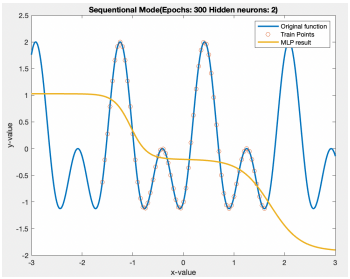

Figure 5: Sequential(2)

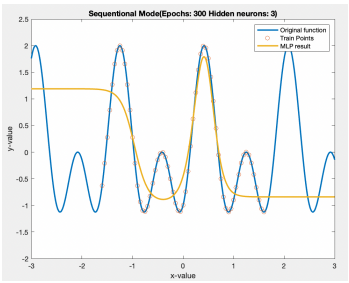

Figure 6: Sequential(3))

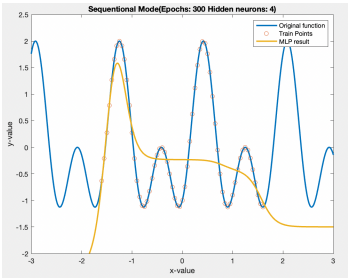

Figure 7: Sequential(4)

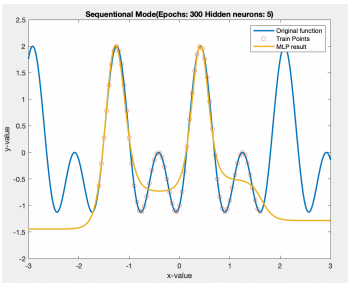

Figure 8: Sequential(5)

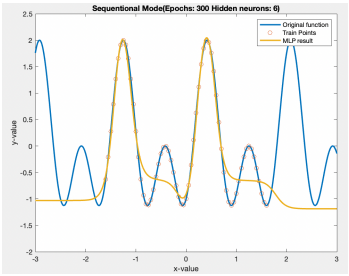

Figure 9: Sequential(6))

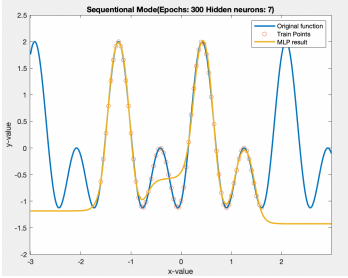

Figure 10: Sequential(7)

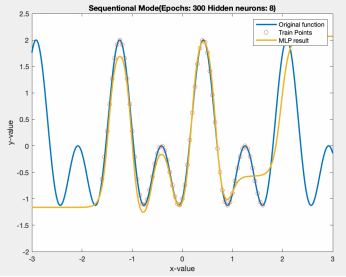

Figure 11: Sequential(8)

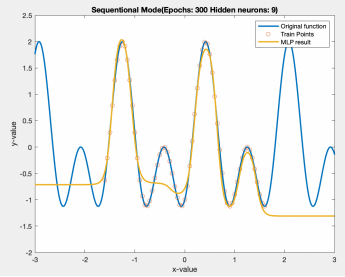

Figure 12: Sequential(9))

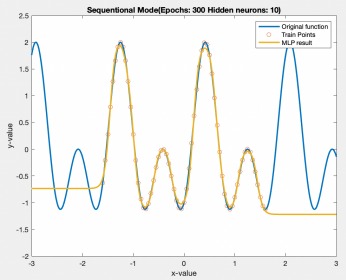

Figure 13: Sequential(10)

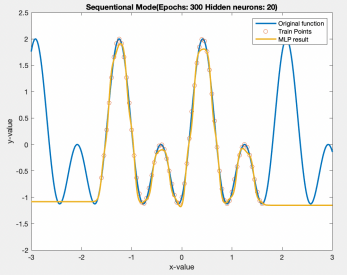

Figure 14: Sequential(20)

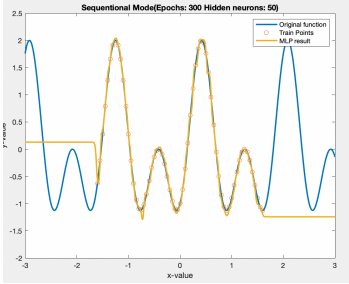

Figure 15: Sequential(50)

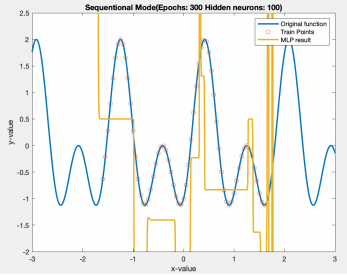

Figure 16: Sequential(100)

b

For the trainlm mode(Levenberg-Marquardt back propagation), the results are shown from Figure 17 to Figure 29. From these figures, it shows when number of hidden neurons $(1, 2, 3, \cdots 10, 20, 50, 100)$ increases, number of piece-wise part of outputs also increases. Epoch time is set as 300. The fitting state of each situation is shown in Table 1. The result is that for sequential mode, when number of hidden neurons is under 8 (not included), it is under-fitting, when it is 8, 9, 10, 20 or 50, it is proper fitting, when it is 100, it is over fitting. Compared with sequential mode, trainlm mode has a larger range of fitting.
However, outputs can only describe the function in the range of training set $[-1.6, 1.6]$, they cannot make reasonable prediction out of the domain.

Hidden Neuron	1	2	3	4	5	6	7	8	9	10	20	50	100
Fitting State	under-fitting							proper fitting					over-fitting

Table 2: Fitting state for different hidden neurons(trainlm mode)

c

For the trainbr mode(Bayesian Regulation backpropagation), the results are shown from Figure 30 to Figure 42. From these figures, it shows when number of hidden neurons $(1, 2, 3, \cdots 10, 20, 50, 100)$

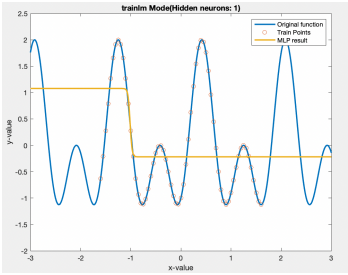

Figure 17: trainlm mode(1)

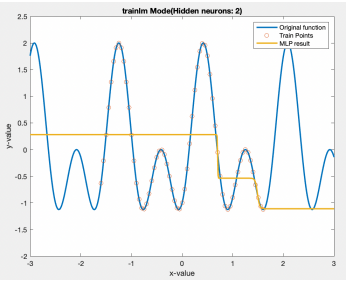

Figure 18: trainlm mode(2)

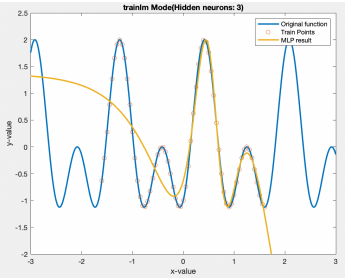

Figure 19: trainlm mode(3))

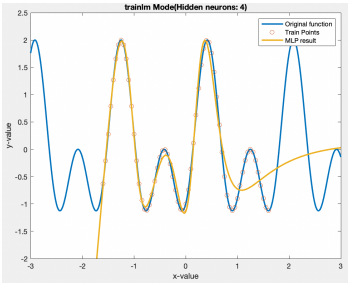

Figure 20: trainlm mode(4)

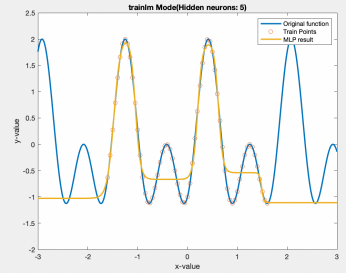

Figure 21: trainlm mode(5)

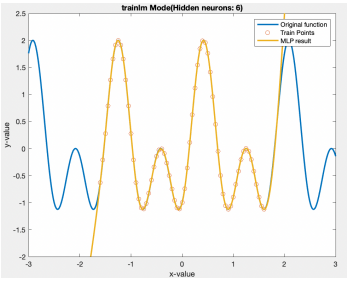

Figure 22: trainlm mode(6))

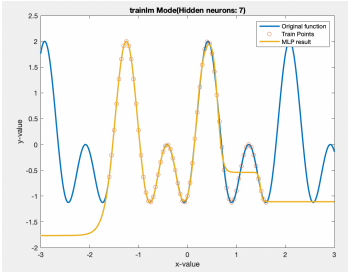

Figure 23: trainlm mode(7)

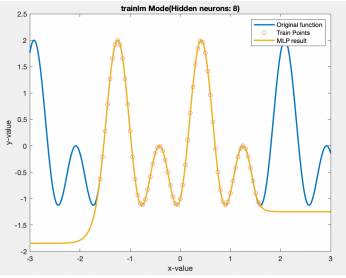

Figure 24: trainlm mode(8)

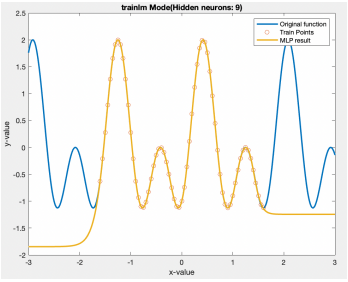

Figure 25: trainlm mode(9))

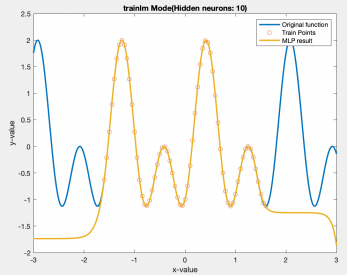

Figure 26: trainlm mode(10)

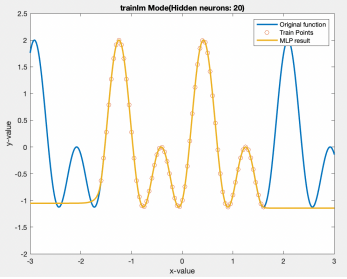

Figure 27: trainlm mode(20)

increases, number of piece-wise part of outputs also increases. Epoch time is set as 300. The fitting state of each situation is shown in Table 1. The result is that for sequential mode, when number of hidden neurons is under 7 (not included), it is under-fitting, when it is over 7, it is proper fitting. Trainbr mode does not have over-fitting situation when number of hidden neurons is not over 100. Compared with sequential mode and trainlm mode, trainbr mode has a larger range of fitting. The reason is that "trainbr" algorithm in MATLAB is a regularization algorithm which can prevent over-fitting. regularization algorithm essentially filters out the high frequency part of the signal. However, similarly, outputs can only describe the function in the range of training set $[-1.6, 1.6]$, they cannot make reasonable prediction out of the domain.

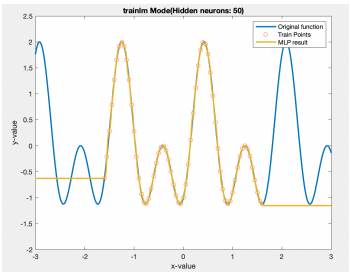

Figure 28: trainlm mode(50)

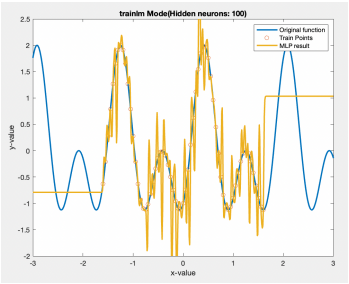

Figure 29: trainlm mode(100)

Hidden Neuron	1	2	3	4	5	6	7	8	9	10	20	50	100
Fitting State	under-fitting						proper fitting						

Table 3: Fitting state for different hidden neurons(trainbr mode)

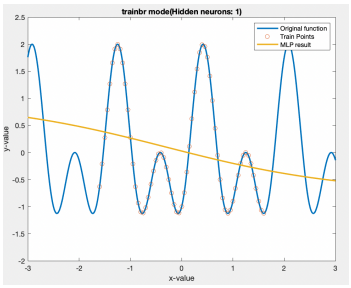

Figure 30: trainbr mode(1)

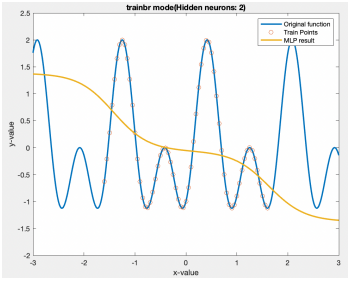

Figure 31: trainbr mode(2)

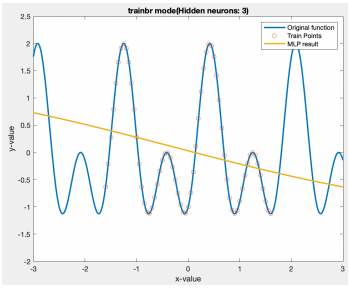

Figure 32: trainbr mode(3))

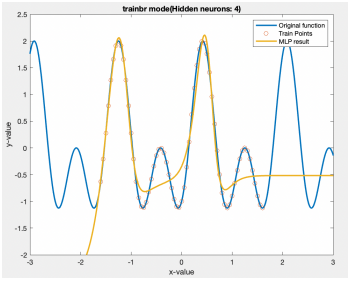

Figure 33: trainbr mode(4)

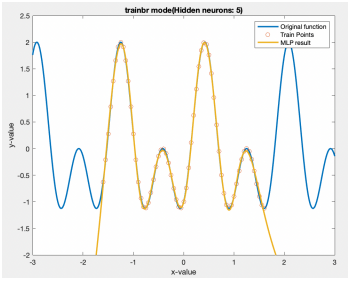

Figure 34: trainbr mode(5)

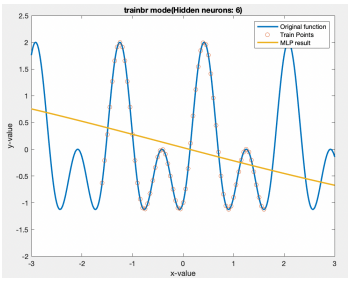

Figure 35: trainbr mode(6))

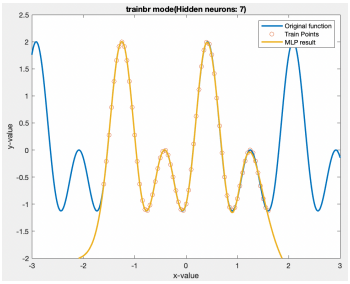

Figure 36: trainbr mode(7)

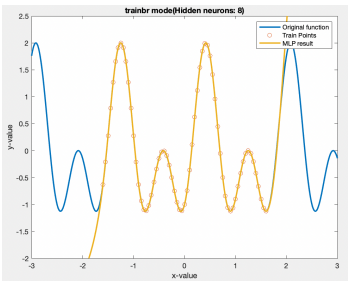

Figure 37: trainbr mode(8)

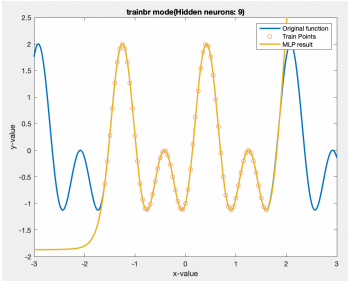

Figure 38: trainbr mode(9))

Q3

My matric number: A0232696W, $mod 96, 3 = 0$, so I chose (cat VS. airplane). For the further questions, 1000 images of cat and airplane are processed from RGB to gray-scale and saved as ".mat" files which

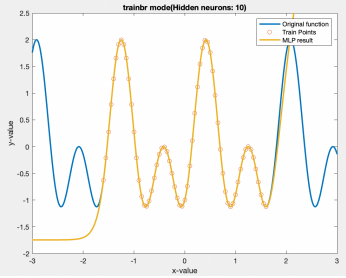

Figure 39: trainbr mode(10)

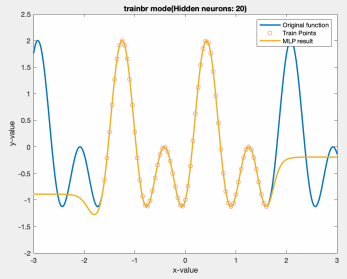

Figure 40: trainbr mode(20)

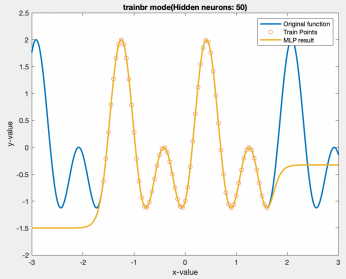

Figure 41: trainbr mode(50)

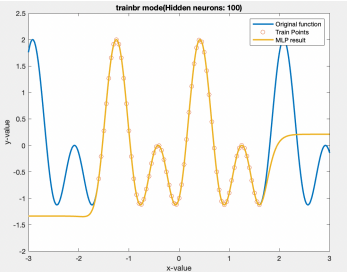

Figure 42: trainbr mode(100)

can be easily loaded in MATLAB.

a

In this sub-question, Rosenblatt's perceptron (single layer perceptron) is applied to the dataset. When learning rate is set as 0.001 The training process takes 4124 epochs to reach 100%. The trajectory of training accuracy and validation accuracy is shown in Figure 43 and 44. It can be observed that. Training accuracy increases in fluctuation, and finally reach 100%, meanwhile validation accuracy keeps fluctuating and final accuracy is 60%.

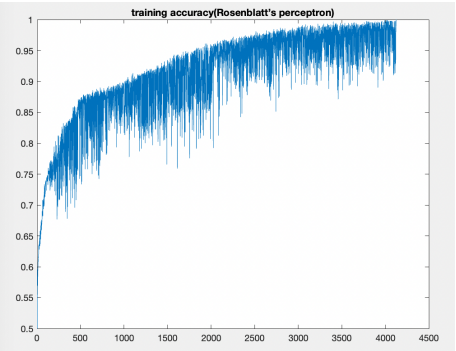

Figure 43: training accuracy(Rosenblatt's)

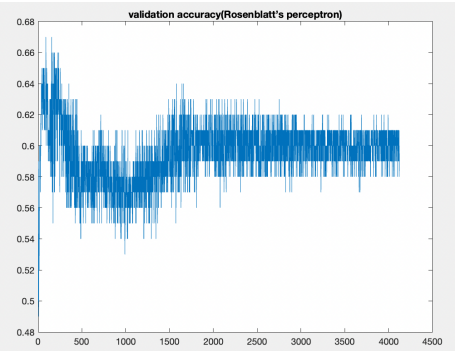

Figure 44: validation accuracy(Rosenblatt's)

b

In this sub-question calculate the global mean and variance of the whole dataset, then subtract the mean value from each image and divide each image by the variance.Result shows in Figure 45 and 46. It takes 462 epochs for training accuracy reaching 100%, which is faster than raw data set in Qa. The final accuracy of validation is 63%, which is little higher in Qa. The reason is that normalization can make data to a same distribution. This can enable gradient to converge in a faster way(in comparison

of Qa and Qb, Qa takes ten times of Qb epochs to reach the goal) and avoid gradient explosion phenomenon.

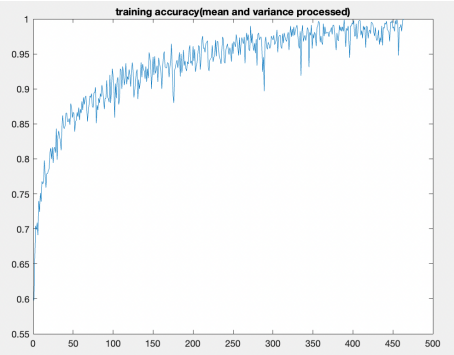

Figure 45: training accuracy(processed)

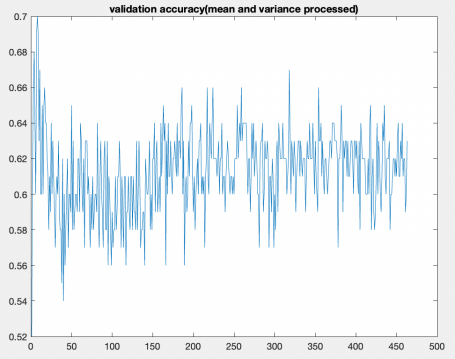

Figure 46: validation accuracy(processed)

c

When apply MLP to the dataset of your assigned group using batch mode training. After the training procedure, calculate the classification accuracy for both the training set and validation set. I tried 'crossentropy', 'mse' and 'sse' 3 performance function. 'sse' takes too long to get the result, I only record the result of 'crossentropy' and 'mse' in Table 4. It shows that 'crossentropy' takes fewer iterations and higher validation than 'mse'. The performance of 'crossentropy' and 'mse' is shown in Figure 47 and Figure 48.

performance function	iterations	train accuracy	validation accuracy
'crossentropy'	570	100%	71.7%
'mse'	747	100%	69.0%

Table 4: Comparison of performance function

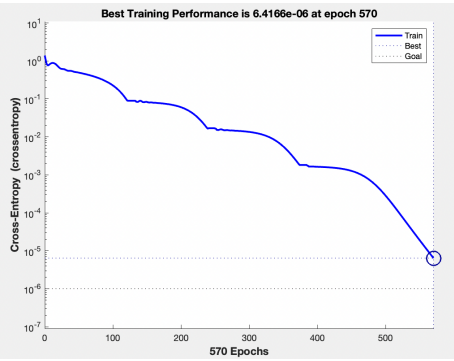

Figure 47: MLP training(cross-entropy)

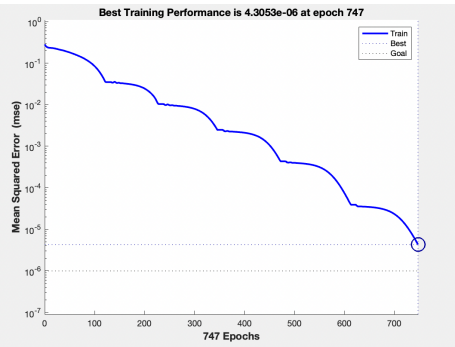

Figure 48: MLP training(mse)

d

From Figure 49, it shows that my MLP has over-fitting and it happens at the 73th iteration. I tried different regulation parameter from 0.0 to 0.95, interval is 0.05 and the result is shown in Figure 50. The regulation result shows that regulation method can increase the accuracy for a little, but for some certain value of regularization, it even decreases the accuracy. There is not obvious relationship between accuracy and regulation parameter. Weights regularization helps this situation but very limited.

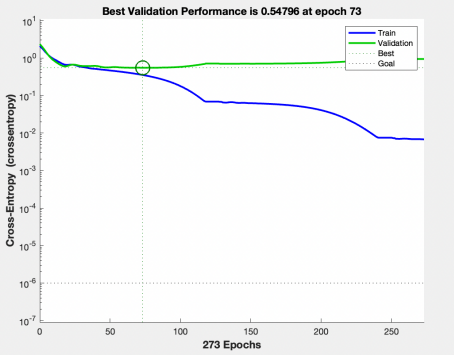

Figure 49: best validation performance

Figure 50: accuracy(with regularization)

e

In this part sequential training mode is applied. Figure 51 and 52 shows accuracy during sequential mode training. Compared with the result in question C. It shows that final accuracy of sequential mode training is 100% and final validation accuracy is 72% which is very close to batch mode result. Under this circumstance, I recommend batch mode, which is faster for training accuracy to reach zero and get the result. With similar result, we should choose the one consuming less time. Although sequential mode can reduce over-fitting situation to some extend, batch mode can also avoid over-fitting by introducing regularization. So batch mode is my recommendation.

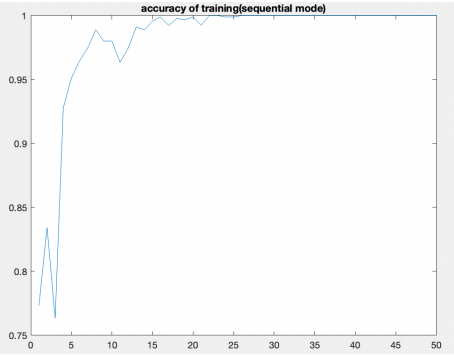

Figure 51: training(sequential mode)

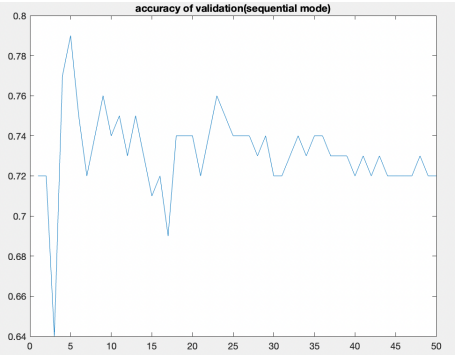

Figure 52: validation(sequential mode)

f

1) Suitable number of hidden neurons should be chosen. It should not be to small to solve the question, or too large to avoid over-fitting.
2) Some methods should be applied to process original data. Inspired by question b), performance of model may be affected by global mean and variance. I think there are also other factors can influence performance. Original data should be processed before putting into MLP training directly.
3) The best performance of validation accuracy may not happen when trainparam.goal is reached, but happens during the reaching process. Records of performance during iteration should be observed to find when best performance happens and when over-fitting starts.
4) MATLAB provides many choices of parameters and methods for us to use. There is no universal structures for every questions, but there is suitable one for a certain question. Performance of MLP is improved by trial and learn.