Homework 2

Luo Zhiyao

Please note that this homework is solved both through Python and MATLAB. Since Pytorch is not built for support naïve networks, some MATLAB setting such as trainlm and trainbr are not supported. My result shows that MATLAB implementation leads to a better result compared with Python implementation.

The below-mentioned results are all from the MATLAB implementation. MATLAB code is attached in the attachment file.

Complete version of source code (both Python and MATLAB) can be found at my Github repo: https://github.com/GilesLuo/nn/. There are also some interesting GIFs for the visualization of training process.

Q1

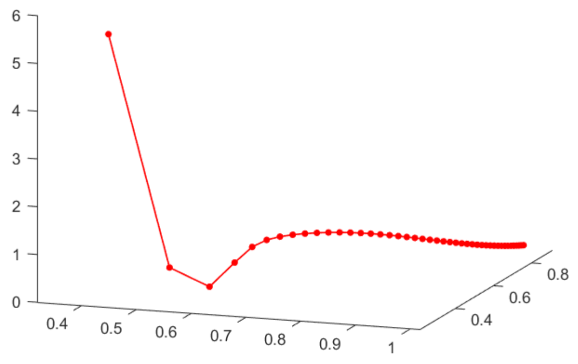
a)

The gradient can be derived as

$$\begin{bmatrix} 400x^3 - 400xy + 2x - 2 \\ -200x^2 + 200y \end{bmatrix}$$

The update rule is

$$x(n+1) = x(n) - lr * \frac{df}{dx};$$
$$y(n+1) = y(n) - lr * \frac{df}{dy};$$



When learning rate is 0.001, 99 steps are spent to get the above result. The MSE error is 0.00095. when learning rate is 0.2, it will never converge. Finally, the error will blow up.

b)

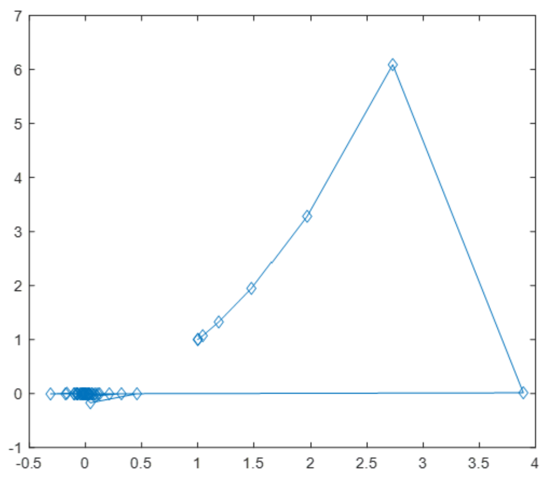
the hessian matrix:

$$\mathrm{H} = \begin{bmatrix} 1200x^2 & -400 \\ -400x & 200 \end{bmatrix}$$

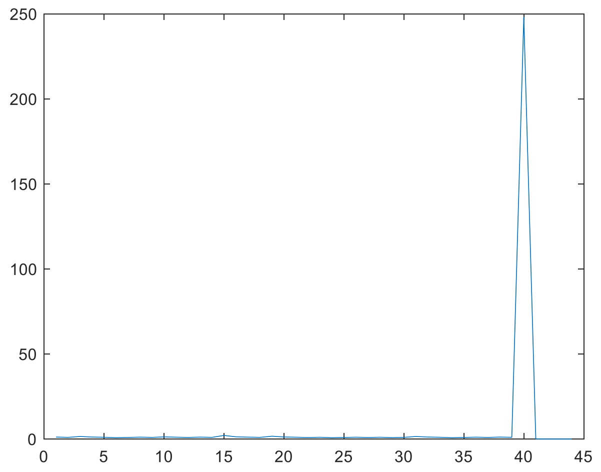
Update rule:

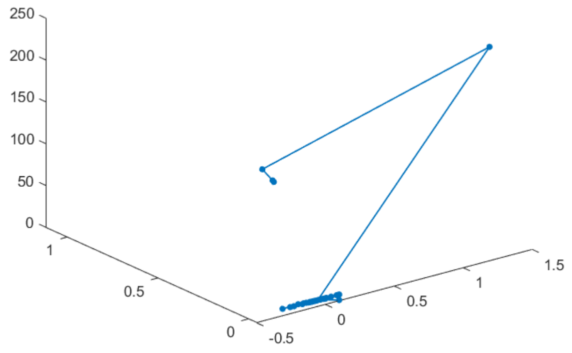
$$\Delta w(n) = -H^{-1}(n) g(n)$$

After 9 iterations the algorithm converges. (x,y)trajectory is shown below.



Value curve is shown below.

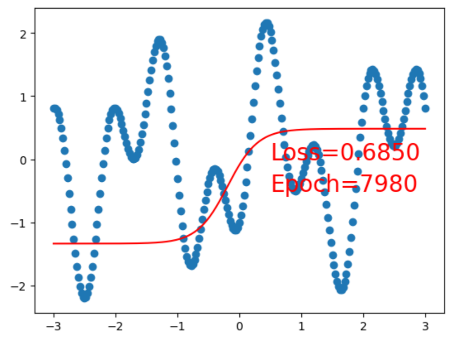




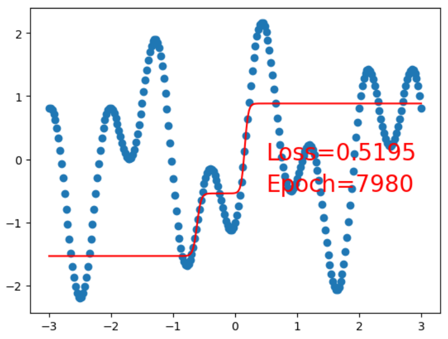
Q2

a)

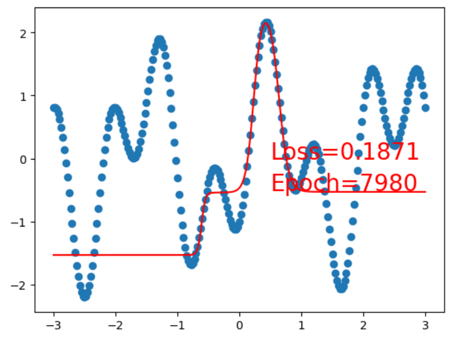
N=1



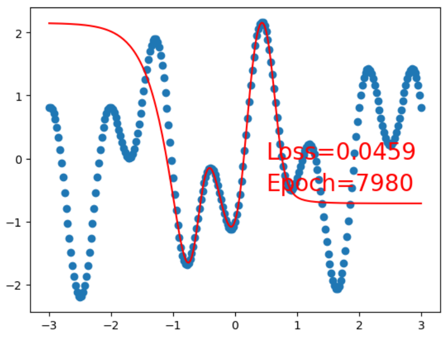
N=2



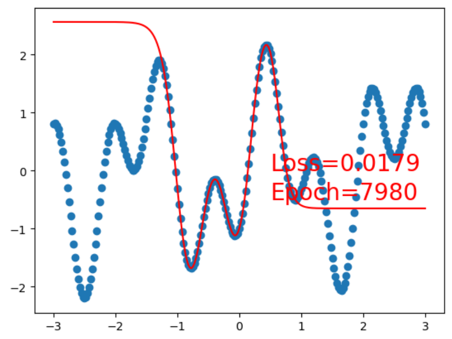
N=3



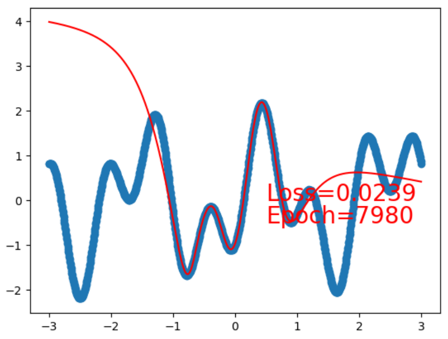
N=4

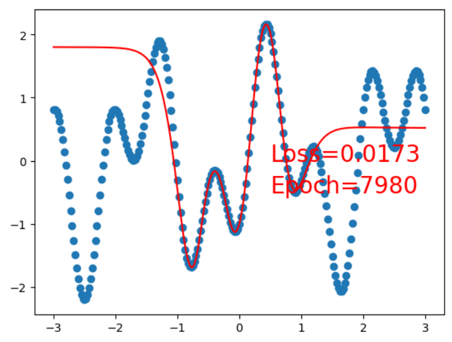
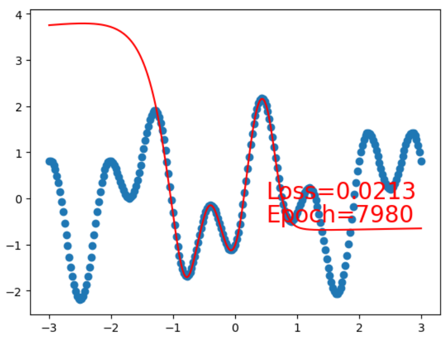
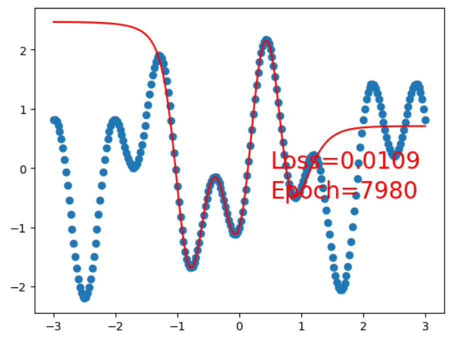
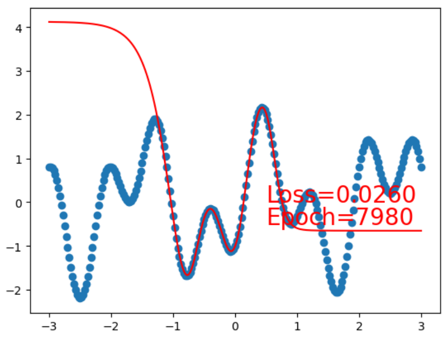
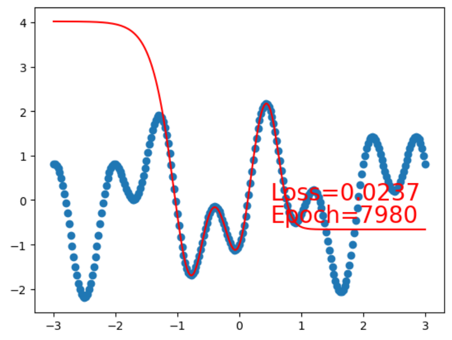
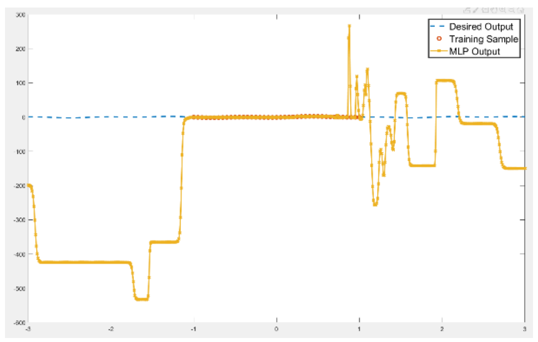


N=5



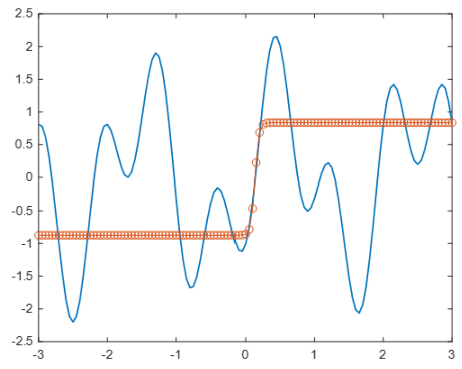
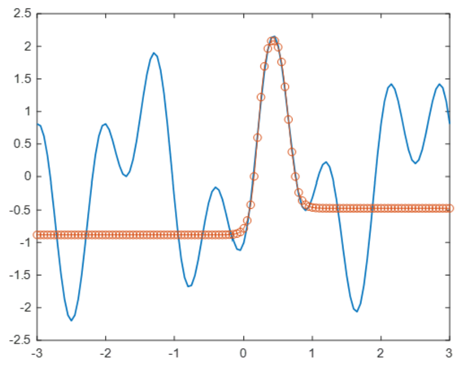
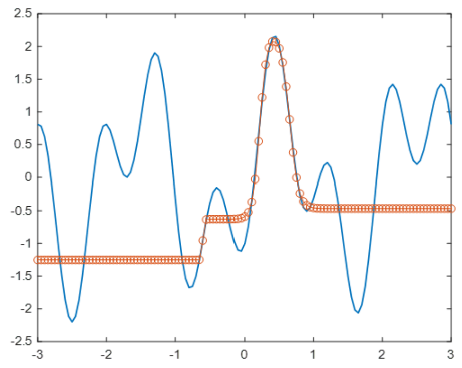
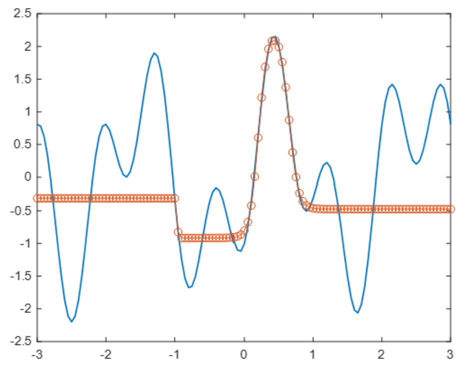
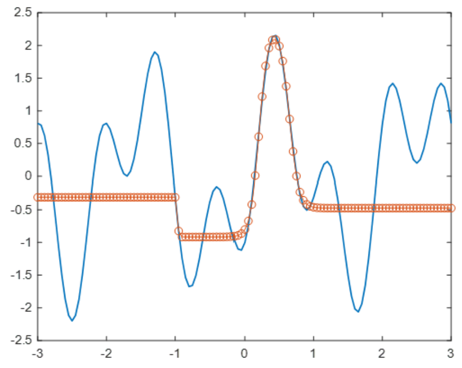
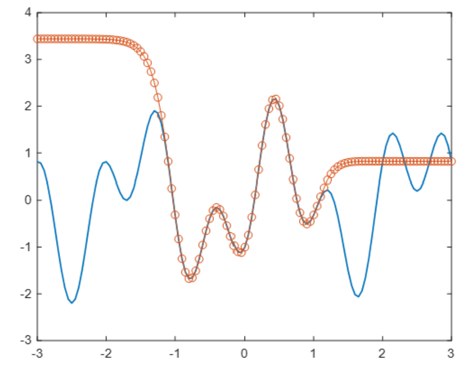
N=6

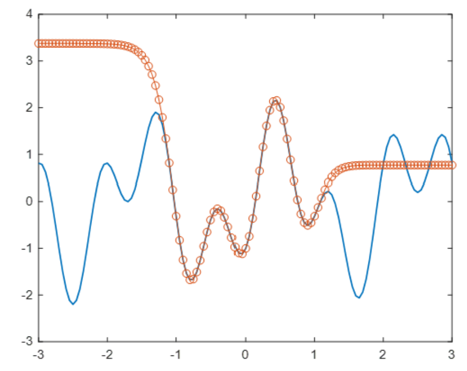
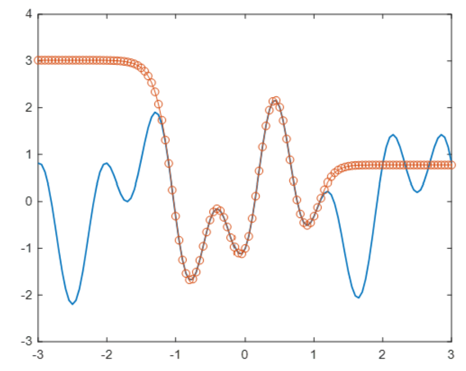
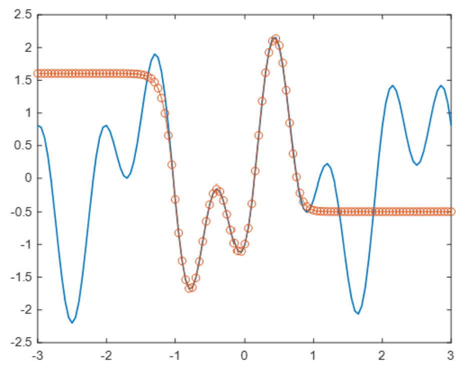
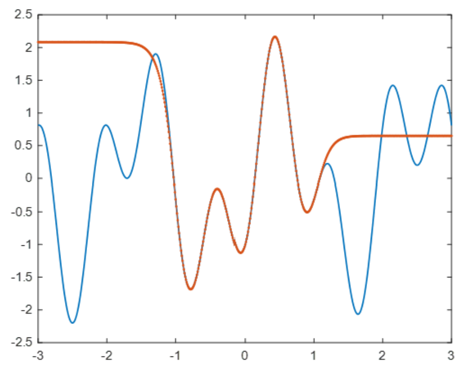
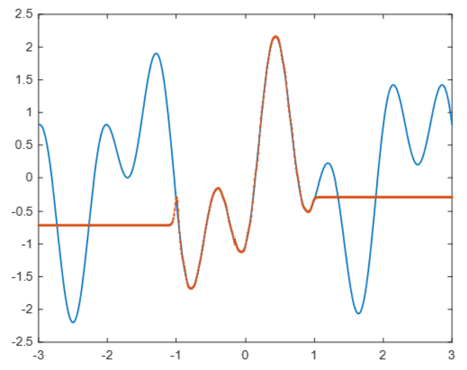
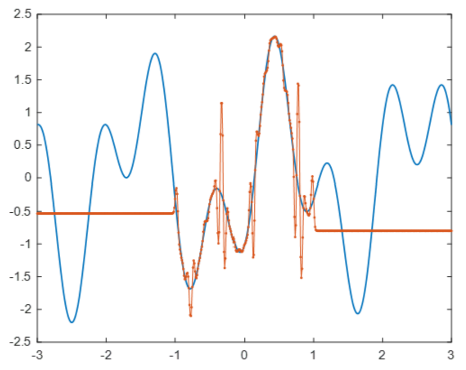


N=7 	N=8 
N=9 	N=10 
N=20 	N=50 

As shown in Table that the minimal number of neurons to fit the given function is 6, which is the same with lecture slides. MLP will become over-fitting when neuron number is 50 since it exceeds too much from 6. None of these MLPs can make proper predictions outside of the input domain a lot, e.g. 3, which verify the statement that network is only good at doing what it is trained to do.

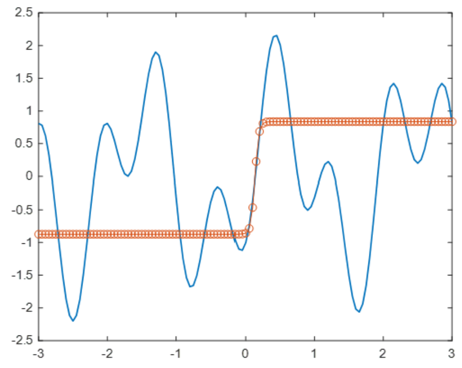
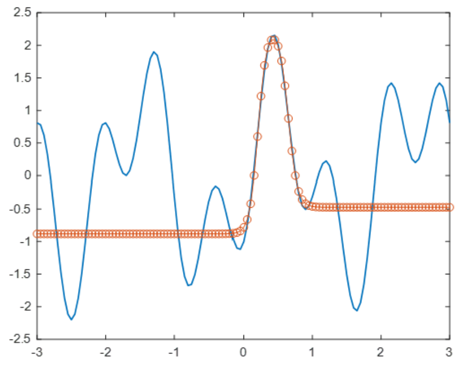
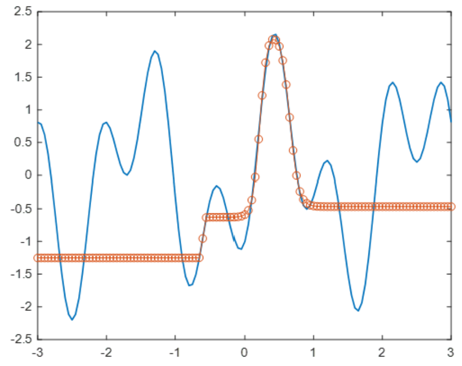
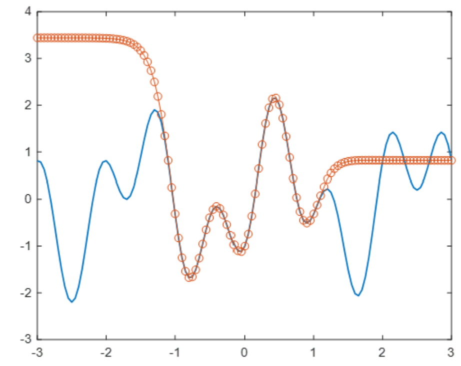
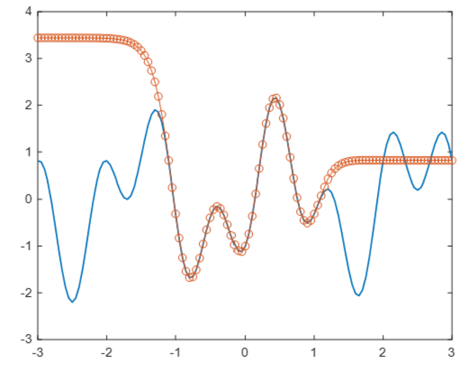
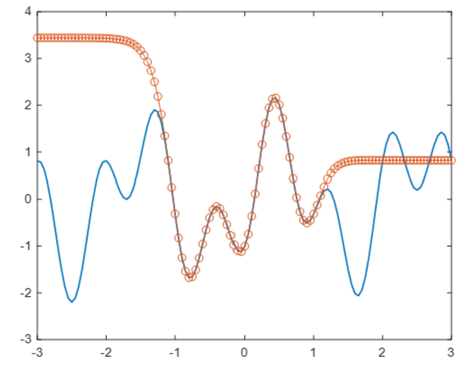
b)

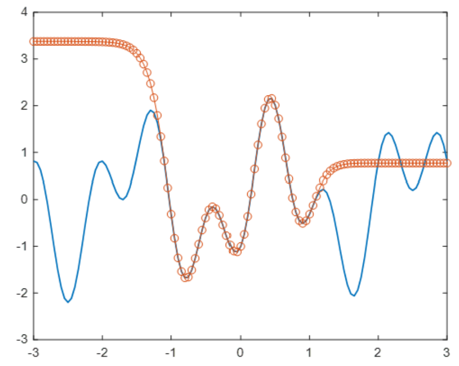
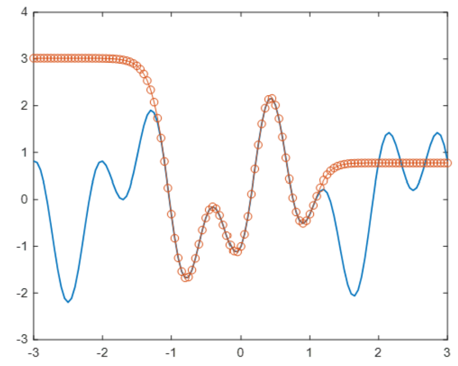
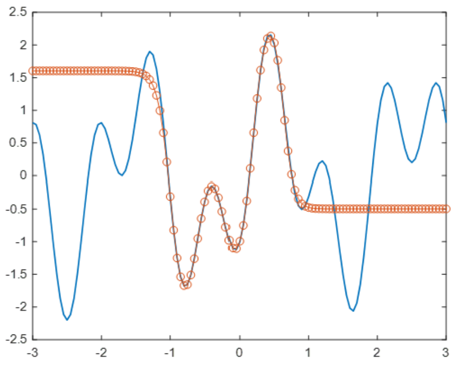
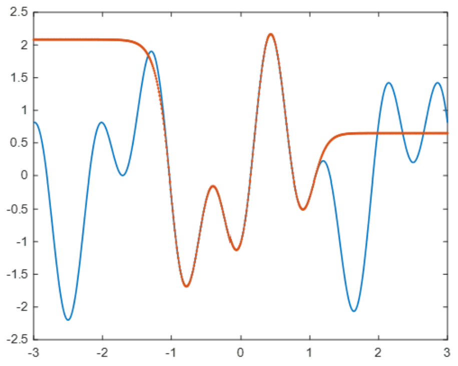
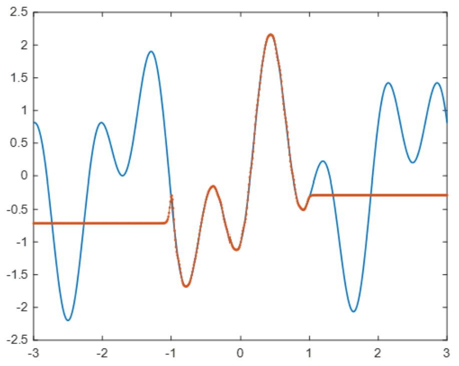
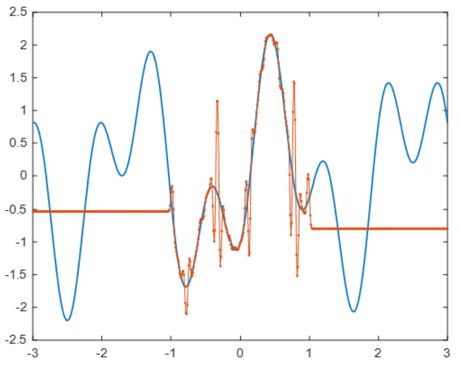
N=1 	N=2 
N=3 	N=4 
N=5 	N=6 
N=7	N=8

	
N=9 	N=10 
N=20 	N=50 

As shown in Table that the minimal number of neurons to fit the given function is 6, which is the same with lecture slides. MLP will become over-fitting when neuron number is 50 since it exceeds too much from 6, but the fitting result is better than sequential learning. None of these MLPs can make proper predictions outside of the input domain a lot, e.g. 3, which verify the statement that network is only good at doing what it is trained to do.

c)

N=1 	N=2 
N=3 	N=4 
N=5 	N=6 
N=7	N=8

	
N=9 	N=10 
N=20 	N=50 

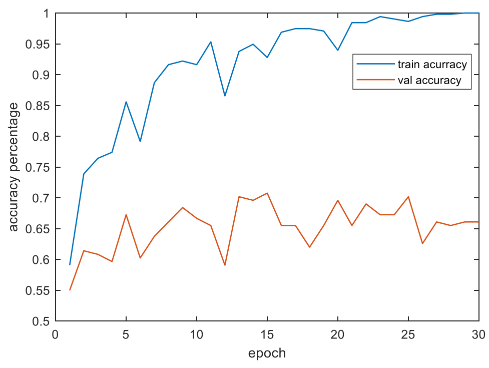
Using trainbr, when number of neuron is 50, MLP is well-fitting instead of overfitting, because trainlm contains regularization to prevent model from overfitting.

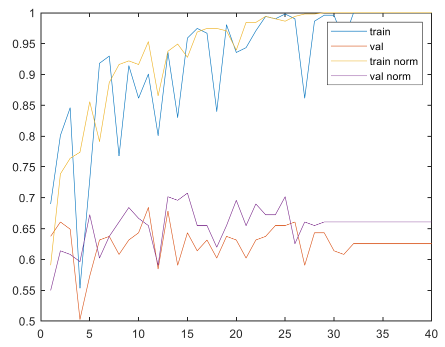
Another difference is that the minimum neuron number becomes 4, which is not consistent with lecture slides. At last, still none of these trained MLPs could make reasonable predictions outside of the input domain. Although there are some points near the input domain that can be proper predicted by MLP, those points are ‘unconsciously’ predicted. There is no guarantee that points near the input domain can also be proper predicted, an example can be found when n=20.

Q3

a)

We use perceptron to classify the images into two categories: tall building [0] and forest [1]. The training result is shown in left figure. The training accuracy converges to 100%, which means that the training set is linear separable. The maximum validation accuracy is around 70%.   Right figure shows the comparison between training with original images and training with normalized images. It can be seen that accuracy is improved by 2% with normalization.





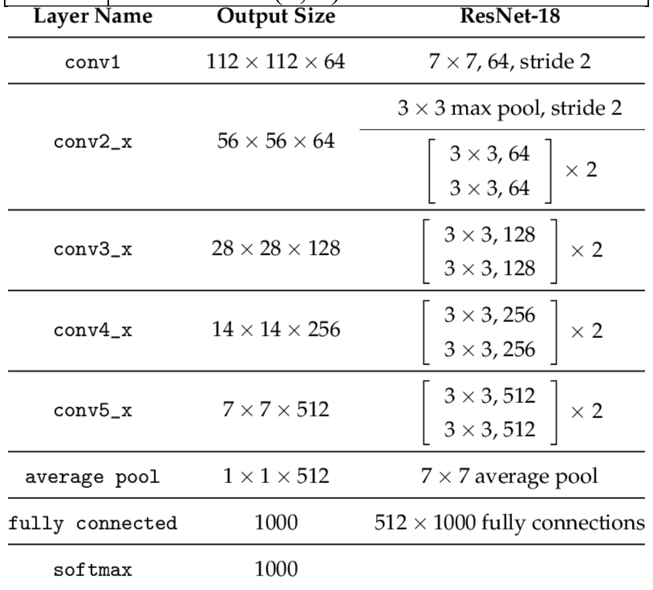
b)

The training result is shown in table. For each set of training, the experiment result is computed using the mean of repetition training to reduce the influence of weight initialization. We can find out that reducing the image size can slightly improve validation accuracy when image size is 128*128, since it compresses redundant information such that perceptron can work better. However, if image size is smaller than 128, input information will lose too much, which lead to worse performance.

Net	Image Size	Training Accuracy	Validation Accuracy	Convergence Epoch
Perceptron	256	100%	68.42%	46
	128	100%	69.59%	87
	64	100%	67.25%	231
	32	100%	66.67%	233

Other way to extract features is to use convolutional neural network. For convenience I only tried two types of CNN: simple CNN and Resnet, their network structures are shown in tables below.

Layer	Simple CNN
1	Conv2d(1,6,3,stride=1, padding=1),
2	ReLU
3	MaxPool2d(2, 2),
4	Conv2d(6,16,5,stride=1, padding=0),
5	ReLU
6	MaxPool2d(2, 2)



Layer Name	Output Size	ResNet-18
conv1	$112 \times 112 \times 64$	7×7, 64, stride 2
conv2_x	$56 \times 56 \times 64$	3×3 max pool, stride 2
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$
conv3_x	$28 \times 28 \times 128$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$
conv4_x	$14 \times 14 \times 256$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$
conv5_x	$7 \times 7 \times 512$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$
average pool	$1 \times 1 \times 512$	7×7 average pool
fully connected	1000	512×1000 fully connections
softmax	1000	

Obviously, CNN produces much superior result than fully connected network, because convolutional layer can better extract the features of images. The maximum validation accuracy of CNN network is 98.83%, where only 1 image is mis-classified.

Net	Image Size	Training Accuracy	Validation Accuracy	Convergence Epoch
Simple CNN	256	96.09%	91.27%	31
	128	96.09%	89.22%	27
	64	96.09%	85.71%	30
	32	100%	86.28%	35
ResNet18	256	100%	93.83%	57
	128	100%	92.35%	46
	64	100%	92.78%	47

	32	100%	88.91%	30

c)

Net	Neuron Number	Training Accuracy	Validation Accuracy	Convergence Epoch
MLP 1*n*1 batch	1	100%	68.42%	46
	2	99.54%	66.44%	162
	5	100%	67.14%	177
	10	100%	67.11%	192
	20	100%	66.92%	172
	50	100%	66.5%	201

In this section, MLP is applied. I use patternnet() with crossentropy loss function. the training function is default 'trainscg'. Performance of MLP is shown in the table above.

It is worth noting that when neuron number is 2 the training accuracy is not 100%, that is due to the epoch limitation. If the network is trained with more epochs, training accuracy will reach 100%.

Validation accuracy is much lower that the training accuracy may due to three reasons:

- Training set cannot well represent the val set.
- The network is overfitting.
- Backpropagation method is not suitable.
- The generalization ability of MLP network structure is low.

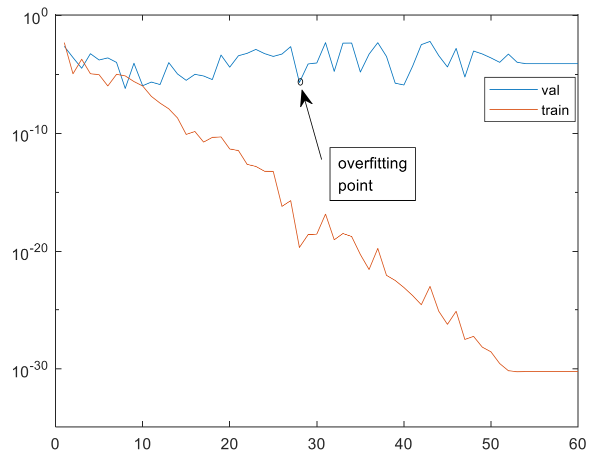
By comparing the result from b), we can know that training set is a good representation of val set, because CNN's validation accuracy can achieve 90% or even higher. So it shouldn't be the problem of dataset.

The validation accuracy is derived by picking the highest val accuracy among all training history for epoch 1 to the last epoch. Since the highest validation accuracy is still too much away from training accuracy, overfitting problem is not the main issue.

To test the influence of backpropagation method, I correspondingly tried trainscg, traingd and traindx, finding out that traingd can result in best performance, which can improve validation accuracy to nearly 72%, but the big gap between training and validation still exists.

Therefore, we can conclude that MLP is not suitable for image recognition. In fully connected network training, we consider the input as a one-hot-vector instead of matrixes. This dimension-reduced operation compresses 2D spatial data into 1D, which lose most of the position information of the data. To avoid this, implementing feature extracting algorithms such as CNN and PCA can dramatically improve the performance.

d)



Regulation rate	0	0.1	0.2	0.5	0.8	0.9
Training accuracy	100%	98.15%	96.80%	95.33%	79.29%	63.25%
Validation accuracy	70.0%	70.3%	69.2%	65.6%	64.1%	62.2%

The validation accuracy is not improved by implementing regulation. The validation accuracy even decreases a lot if regulation rate is too high. An effective way to prevent overfitting is to apply data argumentation. Training image can be resized, shifted, rotated to derive a training set with several times larger number of data. And data argumentation can be realized in Python through 2 lines. By using data argumentation, the validation accuracy can be improved to 78%. Source code is provided in my Github.

e)

Net	Neuron Number	Training Accuracy	Validation Accuracy	Convergence Epoch

MLP 1*n*1 Sequential	1	100%	63.72%	45
	2	100%	67.12%	145
	5	100%	66.14%	169
	10	100%	65.11%	188
	20	100%	66.50%	172
	50	100%	66.29%	197

The above table shows the training performance using sequential mode, where parameter setting is consistent with those in batch mode training. By inspection we can find that batch mode and sequential achieve similar classification performance, whereas using sequential mode is nearly 50 times slower than using batch mode.

Using sequential mode or batch mode is problem dependent. Generally using sequential model is more accurate, while batch mode is faster and more robust. Normally in neural network research neither these two modes are used. Instead, we prefer mini batch training, which integrates the advantages of both modes. To stabilize the training process, batch size is usually set to 256 or bigger, and decrease experiment by experiment to achieve higher benchmark. In this task, batch size=8 can lead to the best result (realized by Python).

e)

A simple improvement would be using Resnet, setting batch size to 4, and applying image argumentation. Best validation accuracy can reach 98%, which means that only 1 image is mis-classified. Reasons have been discussed in former sections.