EE655000 Machine Learning HW3

107060011 電機資訊學士班 涂皓鈞

Part 2. Implementation

1) (35 points) Describe your model architecture details and PCA method.

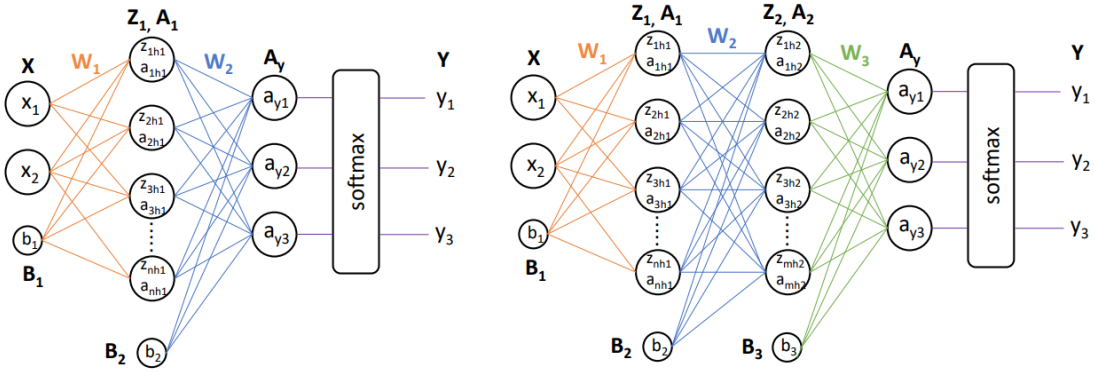


Fig. 1: 2-layer / 3-layer neural network model architecture
(Capital word stands for matrix representation)

a. Model Architecture

2-layer and 3-layer neural network are built as the requirement. Fig. 1 shows the network architecture. All layers are fully connected. The activation function of hidden layer(s) is sigmoid function, while the activation function of the output layer is softmax function. We use stochastic gradient descent (SGD) optimizer to update the weights and bias, which will be introduced in the following sections in detail.

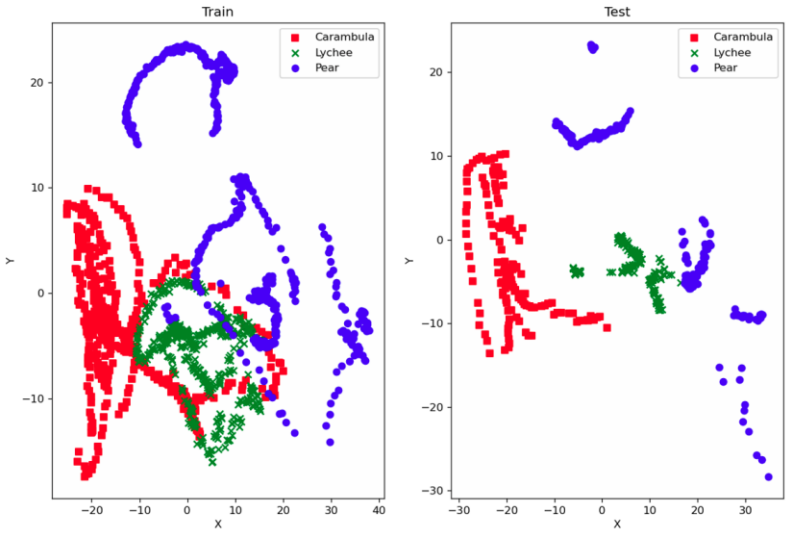


Fig. 2: PCA results of the training/testing set

b. Data Preprocessing

1. Data Shuffling

   Since the label of the dataset is loaded in the order of fruit types, we have to shuffle the data, or it will greatly influence the model performance.

2. Feature Scaling:

   Feature scaling is to prevent the outlier of the dataset affecting the model accuracy too much. Here, we use the function StandardScaler() from sklearn.preprocessing to normalize the dataset. Note that we use the mean and standard deviation of the training set to normalize the training/testing set.

3. Dimension Reduction (PCA):

   Since **the neurons of the input layer are 2**, and the **dimension of images is (32, 32, 2)**, which can be reshaped as 2048, the dimension reduction might be applied here to feed the input to train the model. The method of the dimension reduction we used is principle component analysis (PCA) from from sklearn.decomposition.PCA(ncomponent=2). Fig. 2 shows the PCA results of the training/testing set. Note that we use the PCA result of the training set to perform the dimension reduction in training/testing set.

c. Feedforward

In the forward pass, the input X is passed on to hidden layer(s) and then to the output layer. Between these passing, the input is multiplied by the weights (W) and added with the bias (B), and go through the activation function (A) to get the output of the next layer. The process can be expressed as:

$$Z_i = W_i \cdot X + B_i,$$

where i denotes the number of the hidden layers, the capital form stand for matrix representation (As Fig. 1 shows). The activation function of hidden layers:

$$A_i = \frac{1}{1 + e^{-Z_i}},$$

The output softmax function:

$$y_j = \frac{e^{-z_j}}{\sum_{i=1}^{3} e^{-z_i}}$$

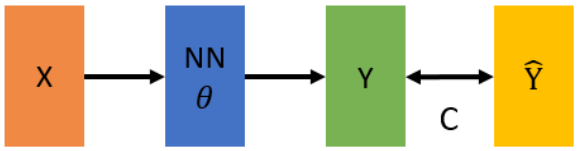
d. Backpropagation

In the backward pass, the resulting value from output nodes is checked for cost value (C) using cross entropy loss function (L), and the weights and bias are

manipulated to minimize the cost value. The process can be expressed as:

$$L(\theta) = \sum_{n=1}^{3} C^n(\theta),$$

$$\frac{\partial L(\theta)}{\partial w} = \sum_{n=1}^{3} \frac{\partial C^n(\theta)}{\partial w} = \frac{\partial C(\theta)}{\partial w},$$



where θ stands for model parameters, $\hat{Y}$ stands for target labels.

For back propagation, it is necessary to find the cost affected by the weights, i.e, to find $\partial C/\partial W$. Take 3-layer neural network for example, the back propagation process can be expressed as follows:

$$\frac{\partial C}{\partial W_1} = \frac{\partial C}{\partial A_y} \cdot \frac{\partial A_y}{\partial Y} \cdot \frac{\partial Y}{\partial A_2} \cdot \frac{\partial A_2}{\partial Z_2} \cdot \frac{\partial Z_2}{\partial A_1} \cdot \frac{\partial A_1}{\partial Z_1} \cdot \frac{\partial Z_1}{\partial W_1},$$

Since the loss function is cross entropy and the output activation function is softmax, $\partial C/\partial A_y$ can be obtained by [1]:

$$\frac{\partial C}{\partial A_y} = Y - \hat{Y},$$

The final result of the back propagation for 3-layer neural network is:

$$\frac{\partial C}{\partial W_1} = (Y - \hat{Y}) \cdot \dot{A_y} \cdot W_3 \cdot \dot{A}_2 \cdot W_2 \cdot \dot{A}_1 \cdot X,$$

For the bias:

$$\frac{\partial C}{\partial B_1} = (Y - \hat{Y}) \cdot \dot{A_y} \cdot W_3 \cdot \dot{A}_2 \cdot W_2 \cdot \dot{A}_1,$$

Similarly, the back propagation for 2-layer neural network is:

$$\frac{\partial C}{\partial W_1} = (Y - \hat{Y}) \cdot \dot{A_y} \cdot W_2 \cdot \dot{A}_1 \cdot X,$$

$$\frac{\partial C}{\partial B_1} = (Y - \hat{Y}) \cdot \dot{A_y} \cdot W_2 \cdot \dot{A}_1$$

where $\dot{A}_i = A_i \cdot (1 - A_i)$.

2) (10 points) Show your test accuracy.

Model parameters setting:

1. 2-layer: neurons of each layer: [2, 10, 3], epoch=100, learning rate=0.001
2. 3-layer: neurons of each layer: [2, 10, 10, 3], epoch=200, learning rate=0.001

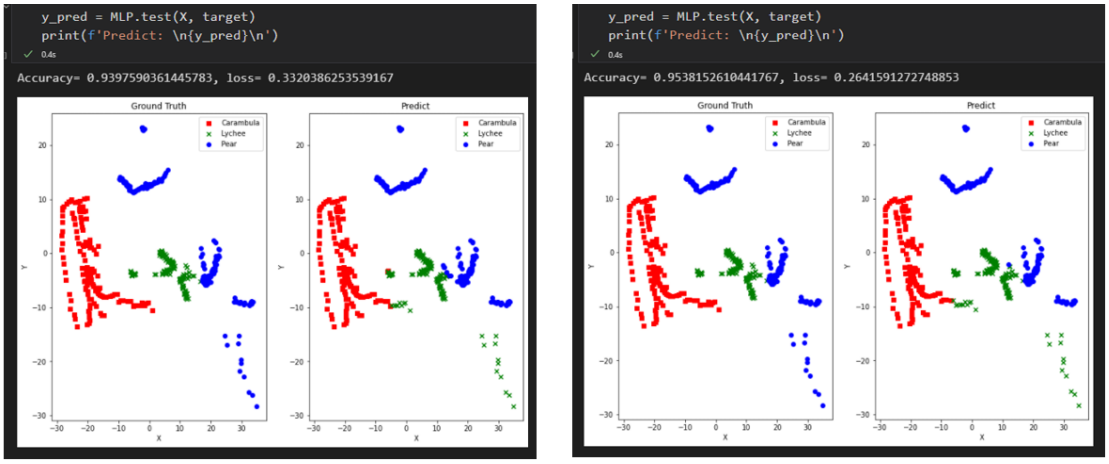


Fig. 3: testing accuracy and data distribution (left: 2-layer; right: 3-layer)

3) (10 points) Plot training loss curves.

The model parameters setting is the same as 2).

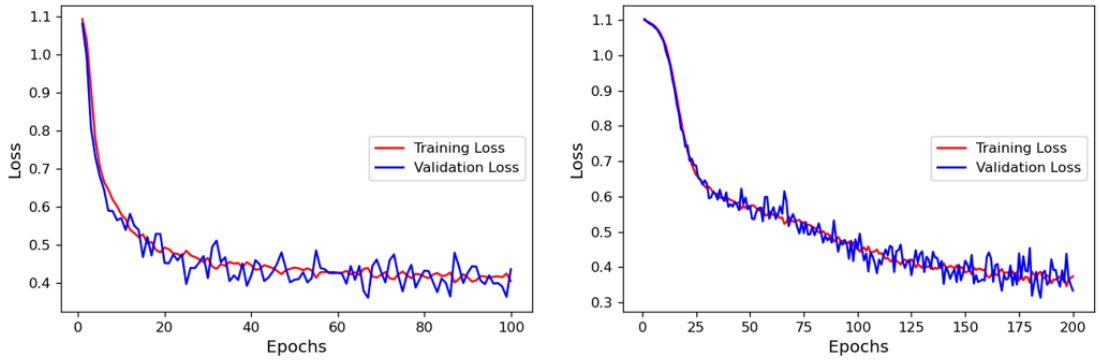


Fig. 4: training/ validation loss vs. epochs (left: 2-layer; right: 3-layer)

4) (45 points) Plot decision regions and discuss the training / testing performance with different settings designed by yourself.

Decision region plots between 2-/3-layer neural network:

1. 2-layer: Neurons = [2, 20, 3], learning rate = 0.01, epoch=200
2. 3-layer: Neurons = [2, 20, 20, 3], learning rate = 0.01, epoch=200

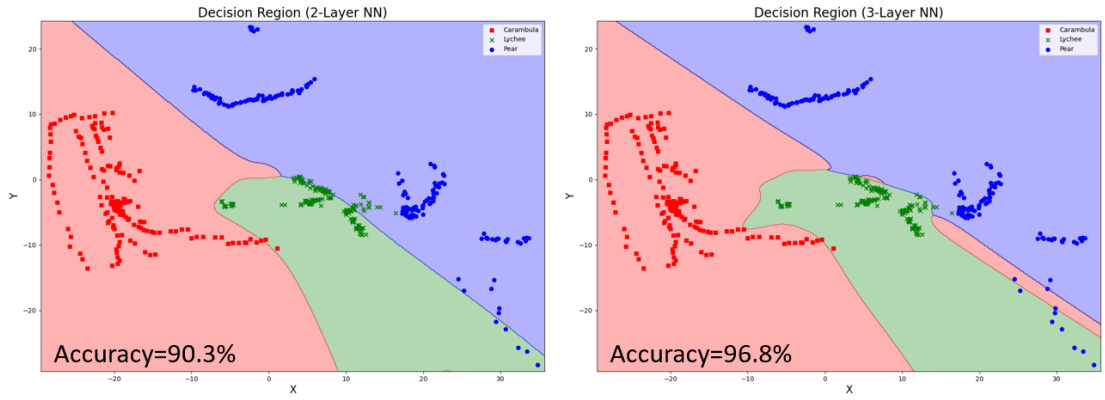


Fig. 3: Decision region of 2-/3-layer neural network.

Fig. 3 shows the decision region plots. Note that the decision region is decided by the model predicting results, and the labelling points scattering on the decision region is the ground truth of the testing set. If the color of the decision region and data points are the same, it means the classifier has the right prediction.

From Fig. 3, we can observe that the curve of the decision region in 3-layer model is more complicated than that in 2-layer model. The reason for this situation is that 3-layer model has more parameters than 2-layer model. Thus, 3-layer model has better performance than 2-layer model if it is well-trained (no underfitting or overfitting occurs).

Model analysis for different parameter settings:

In this part, I only analyze 3-layer neural network, since the biggest difference between 2-/3-layer is the number of parameters. 3-layer neural network can also regulate the number of parameters via adjusting nodes in hidden layers. Further, deep-layer models usually have better performance, and it is more analytically flexible.

1. Different training epochs

Neurons = [2, 10, 10, 3], learning rate = 0.001

Epoch	Training accuracy/loss	Testing accuracy/loss
50	0.6005/0.765	0.6405/0.755
100	0.8442/0.524	0.8514/0.511
200	0.9271/0.327	0.9257/0.320
400	0.9422/0.178	0.9377/0.181
600	0.8852/0.279	0.9598/0.164
800	0.9196/0.266	0.9759/0.091
1000	0.9849/0.063	0.9859/0.064

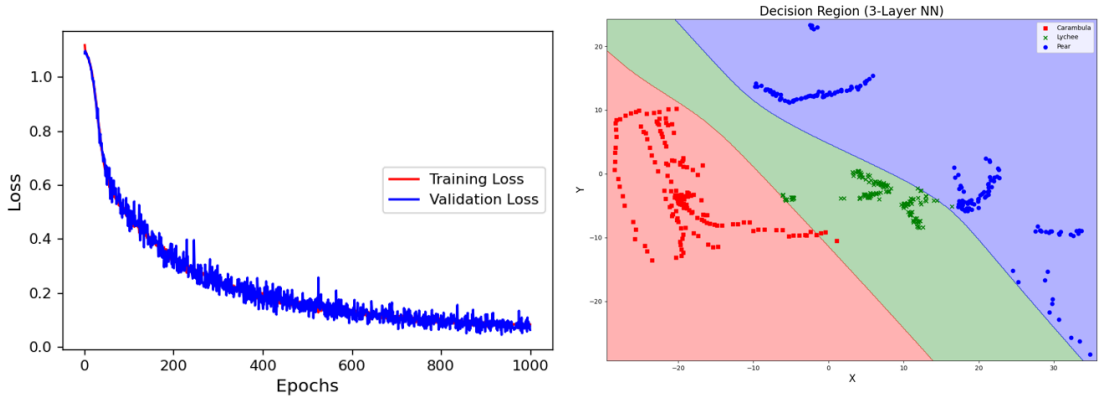


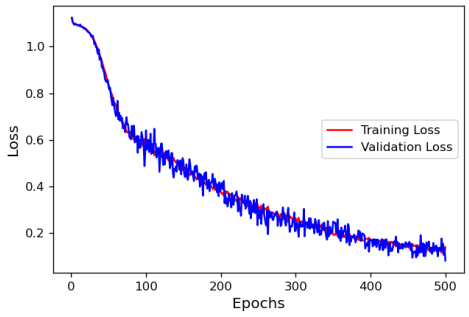
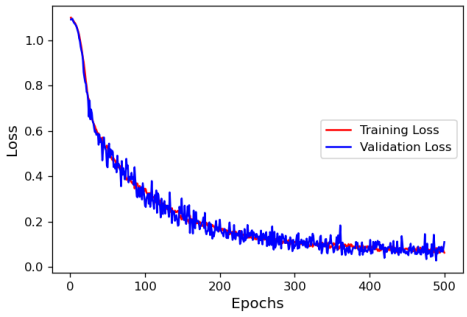
Fig. 4: Training/validation loss and decision region of the model (epoch = 1000).

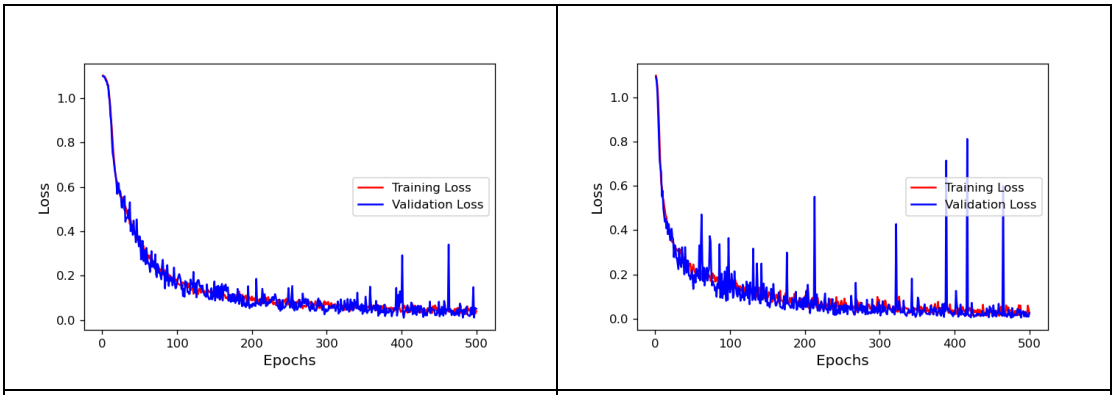
From the table above, it can be seen that the more epoch the model has trained, the higher accuracy / the smaller loss we can obtain. However, if the number of epoch is too much, the overfitting problem would occur. Fig. 4 shows the training/validation loss vs. epoch, which indicates that the loss of the model would saturate when epoch is equal to about 1000. The validation loss can help us to choose appropriate number of epoch to prevent overfitting.

2. Different learning rates

Neurons = [2, 10, 10, 3], epoch = 500

Learning rate	Training accuracy/loss	Testing accuracy/loss
0.001	0.9396/0.139	0.9478/0.121
0.002	0.9874/0.063	0.9919/0.052
0.004	0.9849/0.040	0.9859/0.038
0.008	0.9924/0.026	0.9959/0.028

	
Learning rate = 0.001	Learning rate = 0.002



Learning rate = 0.004	Learning rate = 0.008

Fig. 5: Training/validation loss of the model with different learning rate.

From the table record above, it can be observed that the more the learning rate we increased, the higher accuracy/ the smaller loss we can obtain. Since the large learning rate can enable the model to learn faster. However, from the curve of the training/validation loss in Fig. 5, we can find that the loss is floating and unstable when the learning rate is too large (learn too fast). It can be expected that result of the model is also unstable. But if the learning rate is too small, the requirement to the number of epoch might increase to converge the model. Hence, a proper value of learning rate is important, which would greatly affect the performance and required training time of the model.

3. Different nodes in hidden layers

Learning rate = 0.1, epoch = 50

Nodes	Training accuracy/loss	Testing accuracy/loss
[2, 3, 3, 3]	0.8693/0.417	0.9196/0.369
[2, 5, 5, 3]	0.9271/0.244	0.9257/0.237
[2, 10, 10, 3]	0.9371/0.171	0.9437/0.146
[2, 20, 20, 3]	0.9673/0.104	0.9819/0.094

From the result of the table, we can observe that if neurons of hidden layers are increased, the performance of the model would be better. Since the number of parameters are added, the model architecture is more complex, which has more opportunity to get better performance. However, if the model is too complicated, the required number of epoch might increase, and the chance of overfitting condition might also rise.

[1] https://peterroelants.github.io/posts/cross-entropy-softmax/