Machine Learning HW3

Shang-Ying He
Communication Engineering
sunnyhe1122@gmail.com

For this homework we need to implement the neural network's forward pass and back propagation, for the input x forward pass is just like the function blow and w is the weight which are initialized randomly, $\sigma(\cdot)$ is the activation function, I used sigmoid in my code.

$$y_k(\mathbf{x}, \mathbf{w}) = \sigma\left(\sum_{j=1}^{M} w_{kj}^{(2)} h\left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}\right) + w_{k0}^{(2)}\right)$$

$$\sigma(a) = \frac{1}{1+\exp(-a)}.$$

We need to computed the loss after forwarding to the end of network, and I used cross entropy as loss function.

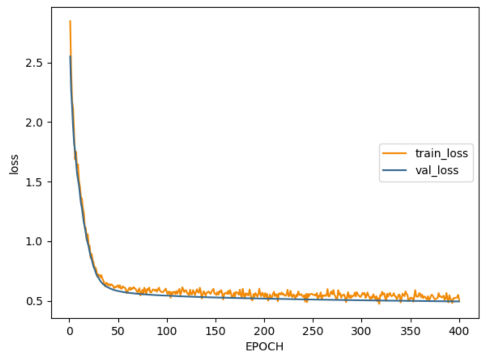
$$\text{loss} = -\sum_i \hat{y}_i \log(y_i) \quad \hat{y}_i: GT \quad y_i: predict$$

For the back propagation, we need to use chain rule and compute the partial differential from loss to all weight of the network, and these values called gradient. Finally, using the function blow to update the weight for every training iteration.

$$\text{weight} = \text{weight} - (\text{gradient} \times \text{learning rate})$$

(a)

In my homework setting for part (a) I set the hidden layer as 256 neurons and using PCA to down scale the feature to 2 dimension and the third dimension for the input is bias which I set as 1. I used the naïve initialization method which is initialized as normal distribution randomly, so the performance is not stable for each training, testing accuracy's range from 0.7 to 0.96, and the best testing accuracy is 0.9659 I have try. Figure blow are training and validation loss, accuracy curves.



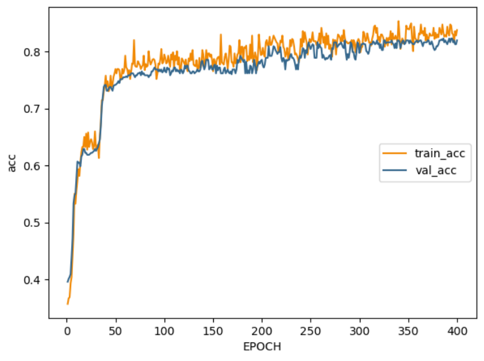
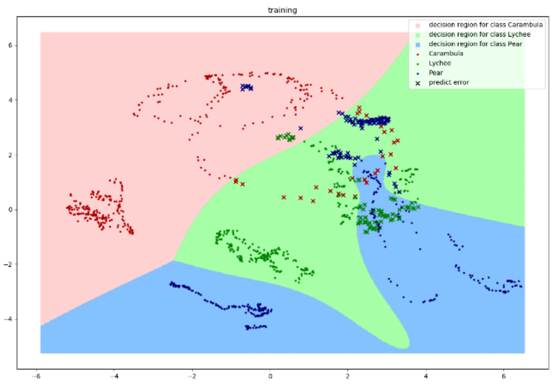
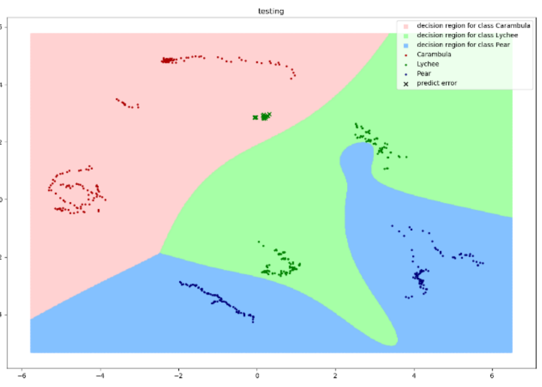


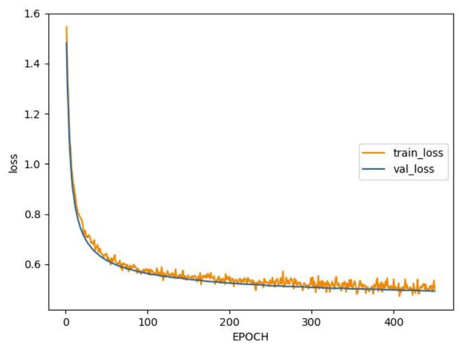
Figure blow is the decision boundary for training and testing data, we can observe that distribution for Pear is very sparse, but model can segment these data simply by using a nonlinear curve, because of sigmoid function can break the linear property in neural network. Data point in the region which have three classes is still difficult for model to learn, because the model without enough parameters to fit the data. But for this problem, we don’t need more parameters to train the model because these data point looks like noise in training data, if we construct more complex model we may yield the model over fit with training data, it would drop the testing accuracy.

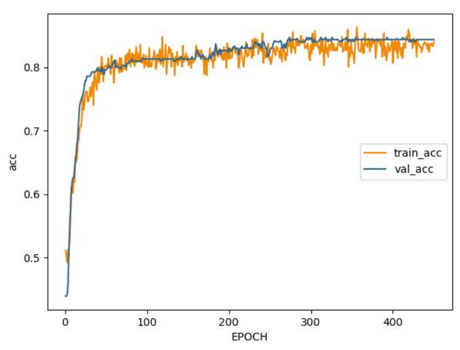


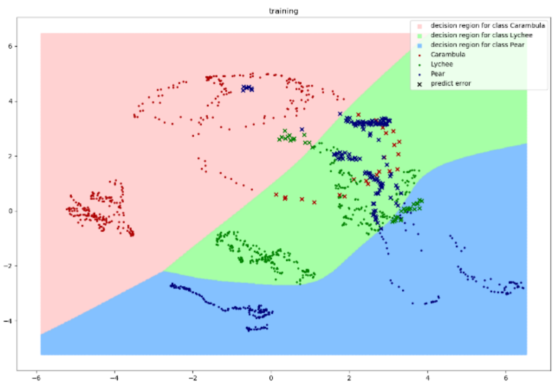


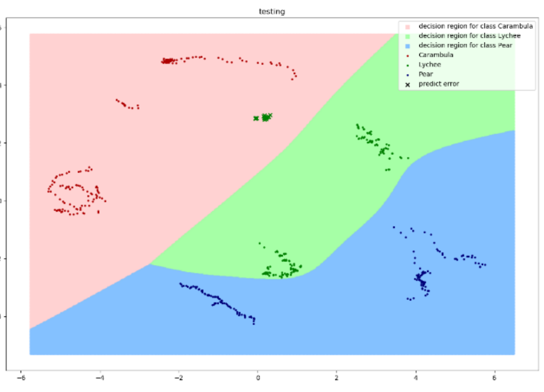
(b)

For part (b) figures blow are performance and decision boundary, I set the second hidden layer as 64 neurons, and first hidden layer still set as 256 neurons which is the same as part (a), and finally got the best testing accuracy is 0.9659 which is the same as part (a), but I am confused to the result. Usually, it should over fit to the data but we can observe that decision boundary is not much complex.

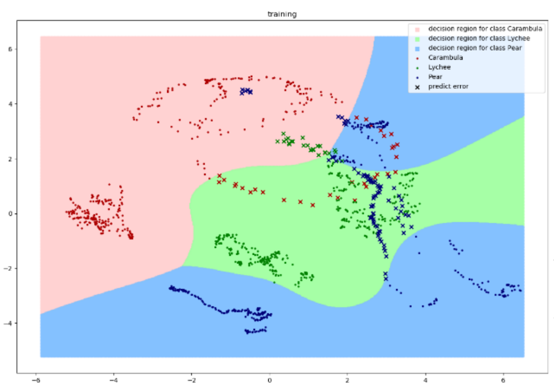


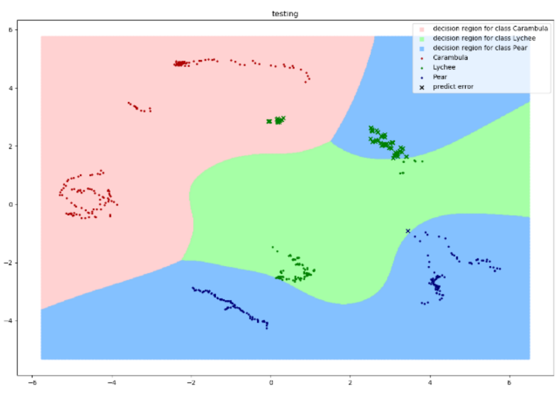




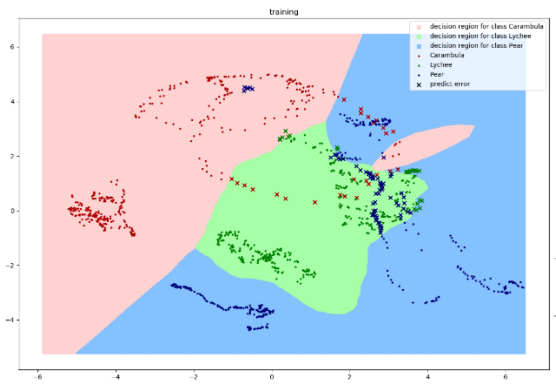


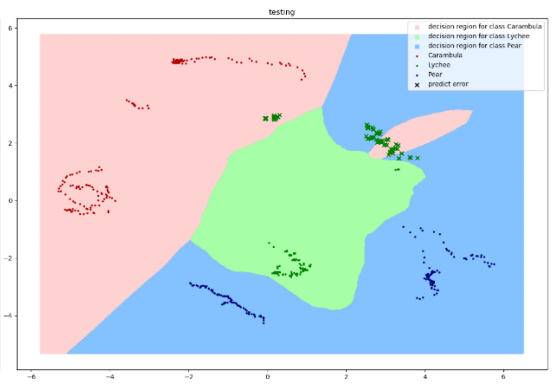
So I did more experiments to observe the result, I set the two hidden layers' neurons as 256 and 1024, finally got the result as blow, it looks more complex but far away from my imagination.





I also try to set activation function as relu, hidden layer neurons set 256 for one hidden layer and figures blow is decision boundary, we can observe that it is overfitting, and I found that if use relu as activation function loss would reduce more quickly and easily than sigmoid, in other words it is easier to overfitting


training
decision region for class Carambula
decision region for class Lychee
decision region for class Pear
Carambula
Lychee
Pear
predict error


testing
decision region for class Carambula
decision region for class Lychee
decision region for class Pear
Carambula
Lychee
Pear
predict error