CMPE 442 Assignment #2

Yasemin Direk

1) In this assignment, I implemented a neural network with one hidden layer and then I used this network to learn IRIS dataset.

a) In this part of assignment, I filled the feedforward() method. General logic of feedforward method is that the predictions are made based on the values of the input nodes and the weights. So, I computed the activation of each neuron in the hidden layer. First, I found the input for the hidden layer. To do this, I multiplied (using dot product) the input nodes with their corresponding weights and added them together by using this formula:

$$XW = x1w1 + x2w2 + \dots$$

Then, I passed the results to the sigmoid activation function:

$$g(XW) = \frac{1}{1 + e^{-XW}}$$

The result of sigmoid function is output from hidden layer. Then, to calculate the value for the output layer, I used the values in the hidden layer as inputs and did the same operations.

b) In this part of assignment, I filled the backprop() method. Backpropagation is a method for training the artificial neural network. So, the goal of backpropagation is to optimize the weights. To do this, I computed the error at the output as below:

$$\text{Error} = \text{target} - \text{output}$$

Then, I calculated gradient by using sigmoid_derivative() function.

$$g(z) = z * (1 - z)$$

Then, I calculated the hidden layer error by applying the formula:

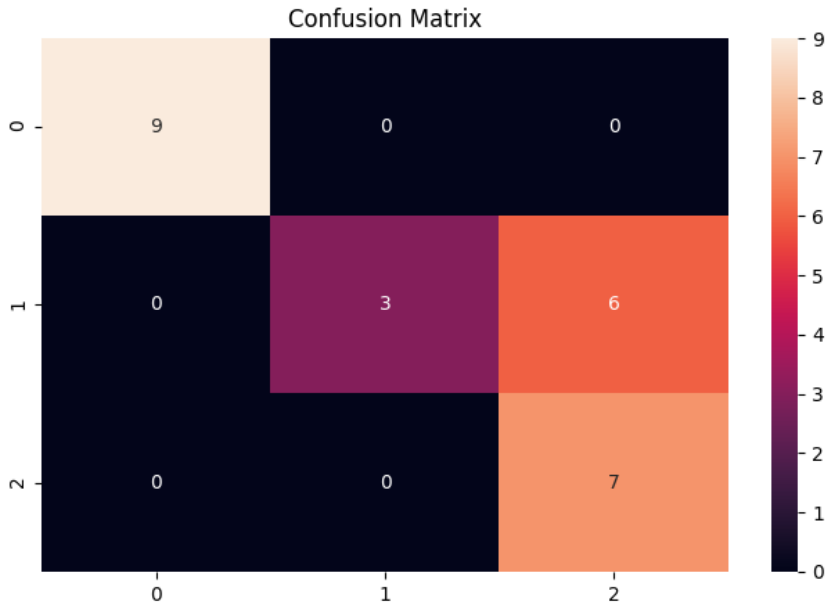
$$\delta^{(2)} = (\theta^{(2)})^T \delta^{(3)} .* g'(z^{(2)})$$

So, I multiplied (using dot product) errors with the transposed weight and multiplied this result with the output derivative. Finally, I updated weights.

2) In this part of assignment, I trained the network for Iris dataset. To do this, I shuffled the samples and I splitted the data into three. 100 samples for training set, 25 samples for validation set and 25 samples for test set.

a) In this part, I fixed iteration number to 1000 and number of units in the hidden layer to 2(sl2 = 2). Then, I trained three networks and for each network that I learned tested it on validation set. I tried different learning rates 0.1, 0.2, 0.3 and tested it on validation set. I calculated accuracy and got the confusion matrix by find_accuracy() method that I wrote. To computing accuracy I first calculate the number of correct predictions and to get the confusion matrix I used sklearn.metrics.confusion_matrix library. Also, to plot the confusion matrix I used matplot.pyplot, pandas and seaborn libraries. Then, I chose the best network according to the accuracies. The network I choose may not be exactly the best, because accuracies may change with each attempt. So, I chose the network which usually gave the best results in these trials as the best network. Finally, I tested the best network on test set also.

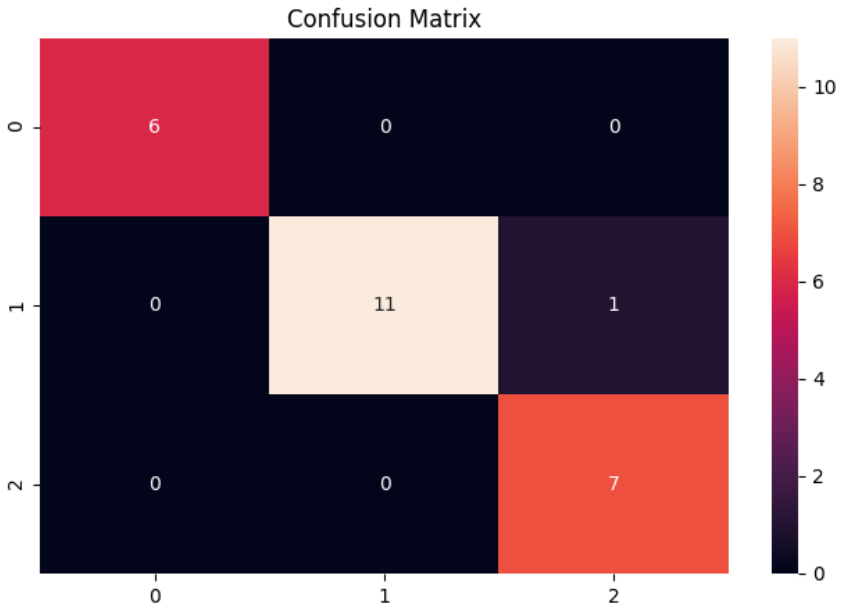
Confusion Matrix and Accuracy for sl2 = 2, lrt = 0.1, iterNo = 1000 (on validation set)



Number of correct predictions = 19

Accuracy = 0.76

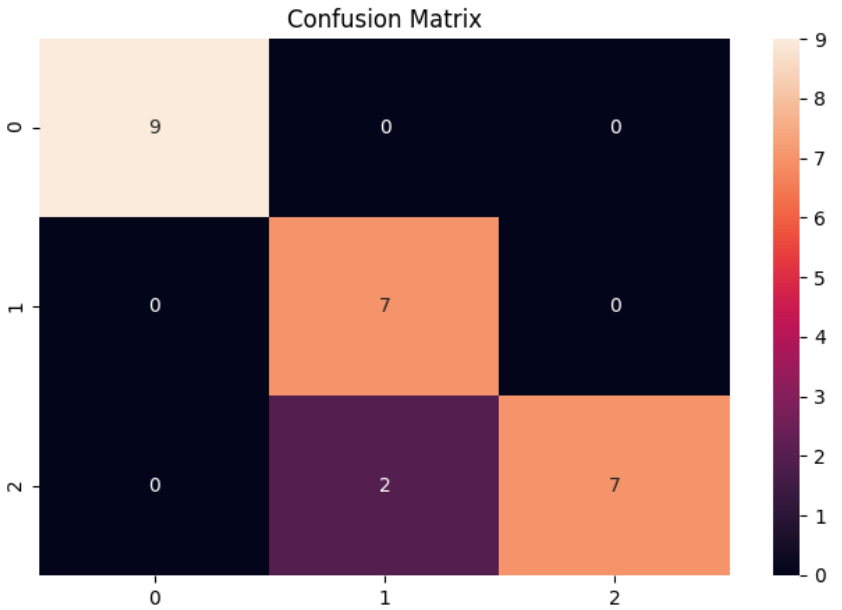
Confusion Matrix and Accuracy for sl2 = 2, lrt = 0.2, iterNo = 1000 (on validation set)



Number of correct predictions = 24

Accuracy = 0.96

Confusion Matrix and Accuracy for sl2 = 2, lrt = 0.3, iterNo = 1000 (on validation set)

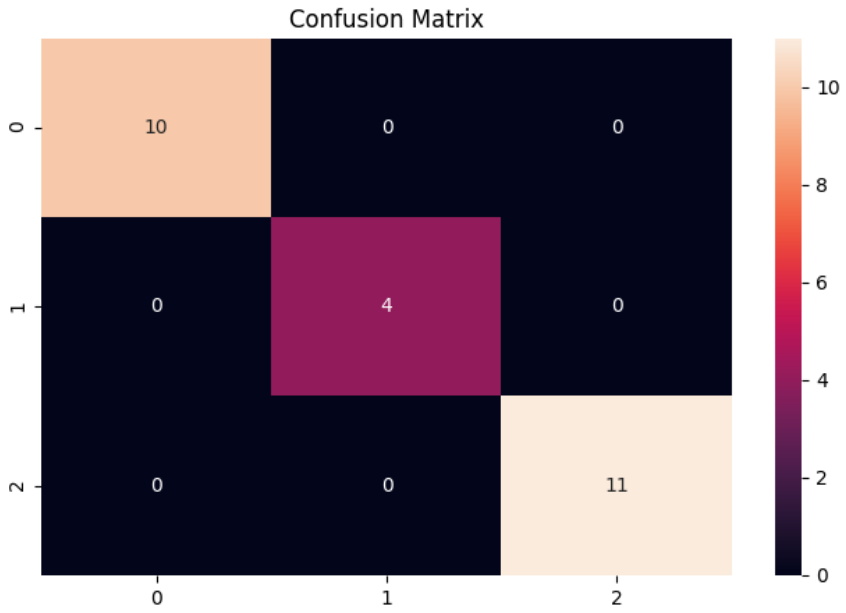


Number of correct predictions = 23

Accuracy = 0.92

According to these results, I chose second network (sl2 = 2, lrt = 0.2) as the best network, because this network has the highest accuracy mostly. The learning rate is an important hyperparameter that controls the rate or speed at which the model learns and choosing the right learning rate is important too. If the learning rate is too smaller, it will take a much longer time to reach an ideal state. On the other hand, if the learning rate is too large, then it might not converge. According to my observations, the best learning rate is 0.2 in this model. So, I tested the best network that I chose on test set also.

Confusion Matrix and Accuracy for sl2 = 2, lrt = 0.2, iterNo = 1000 (on test set)

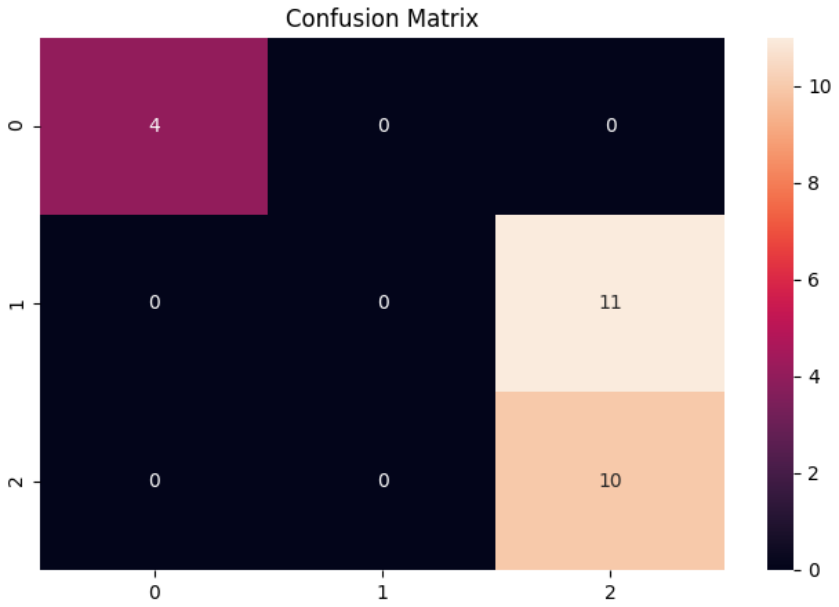


Number of correct predictions = 25

Accuracy = 1.0

b) In this part, I fixed iteration number to 500 and learning rate to 0.2. Then, I trained three networks and for each network that I learned tested it on validation set. I tried different number of units in the hidden layer sl2 = 2, 3, 4 and tested it on validation set. I calculated accuracy and got the confusion matrix by find_accuracy() method that I wrote.

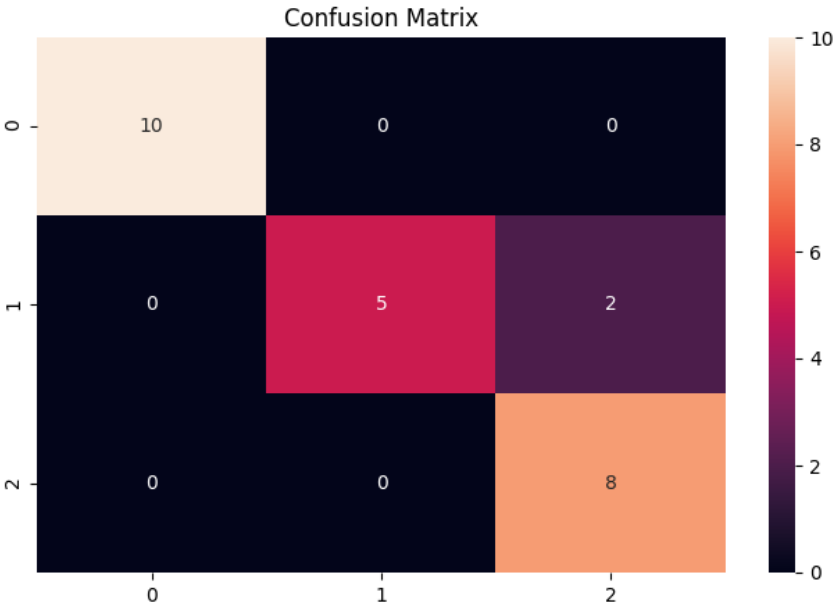
Confusion Matrix and Accuracy for sl2 = 2, lrt = 0.2, iterNo = 500 (on validation set)



Number of correct predictions = 14

Accuracy = 0.56

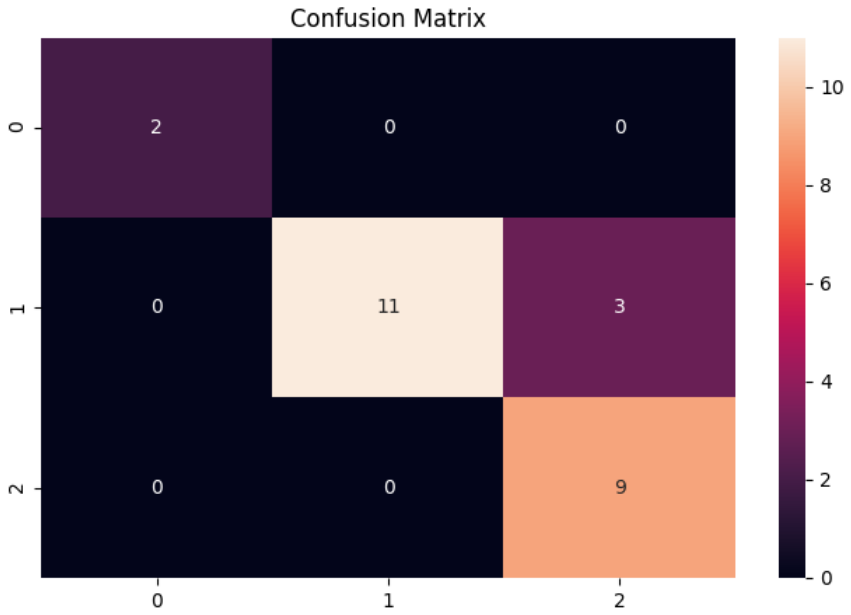
Confusion Matrix and Accuracy for sl2 = 3, lrt = 0.2, iterNo = 500 (on validation set)



Number of correct predictions = 23

Accuracy = 0.92

Confusion Matrix and Accuracy for sl2 = 4, lrt = 0.2, iterNo = 500 (on validation set)



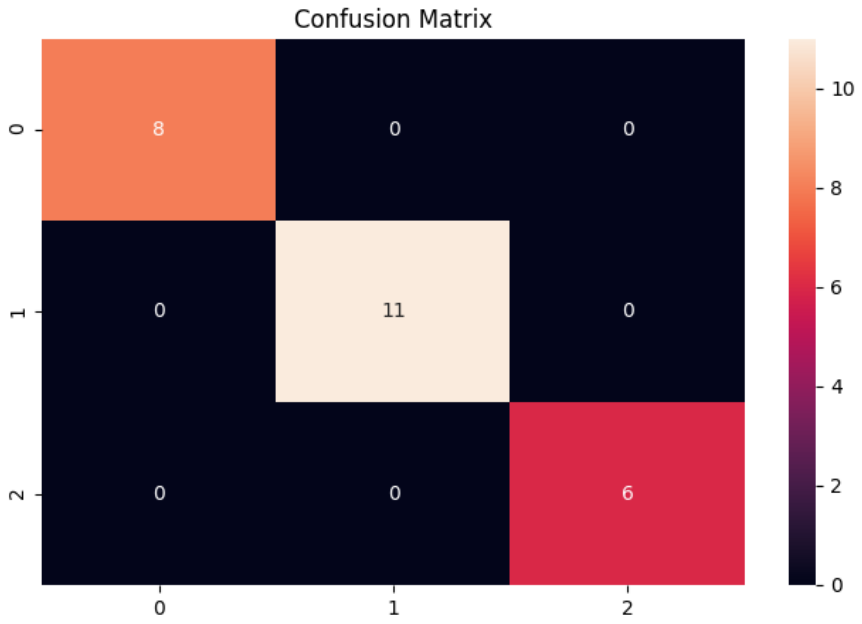
Number of correct predictions = 22

Accuracy = 0.88

According to these results, increasing the number of units might improve the accuracy but if the model is not complex, increasing the number of units much more than ideal number will cause a decrease in accuracy. Also, using too small number of units in the hidden layers will cause a decrease in accuracy. Briefly, using too few neurons in the hidden layers will result in underfitting, and using too many neurons in the hidden layers may result in overfitting. Deciding the number of units in the hidden layer is very important but there is no theory or rule to make this decision. However, there are some methods to choose the right number of units. One of them is that the number of hidden inputs should be between the size of the input layer and the size of the output layer. Our input layer size (input features) is 4 and output layer size is 2. According to this, the number between these values is 3, that is the number of units that give the highest accuracy according to the results. When I choose sl2 = 2 with iteration number = 500 and learning rate = 0.2, I get lower accuracy than sl2 = 3. Also, when I choose sl2 = 4, I get lower accuracy than sl2 = 3, but this result is still better than the result of sl2 = 2. So, both sl2 = 3 and sl2 = 4 give high accuracy, but if I need to make a choice between them, I choose 3 for the number of units.

c) In this part, I learned a model with the hyperparameters that appropriate according to my observations by combining train set and validation set and tested the model on the test set. According to the results of the previous parts, best learning rate is 0.2 and best choice for number of units in hidden layer is 3. So, I combined these results and trained the model on train set, then I tested the network on both validation and test set. I tried both iterNo = 500 and iterNo = 1000, but I could not see a big difference between them. However, I chose iterNo = 1000 because it usually provides a little bit higher accuracy than iterNo = 500.

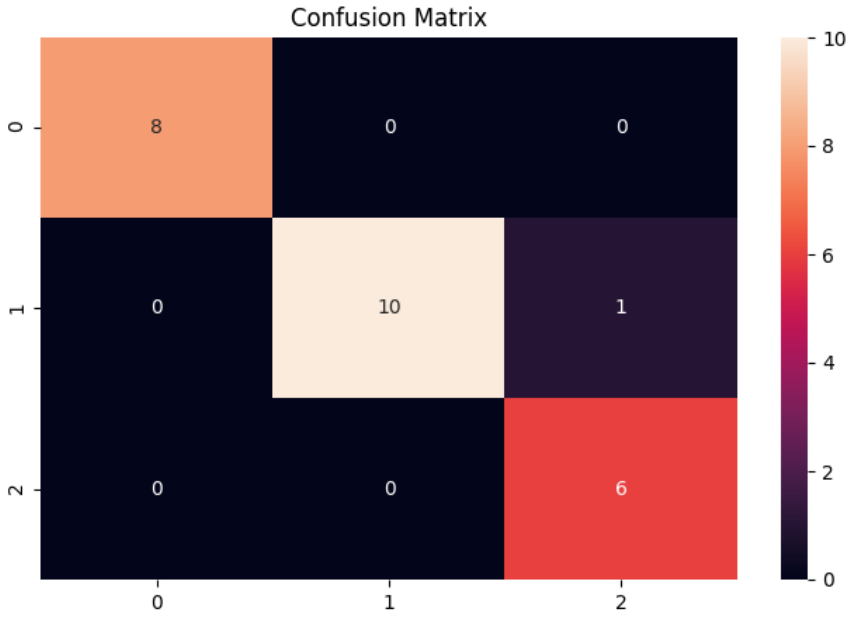
Confusion Matrix and Accuracy for sl2 = 3, lrt = 0.2, iterNo = 1000 (on validation set)



Number of correct predictions = 25

Accuracy = 1.0

Confusion Matrix and Accuracy for sl2 = 3, lrt = 0.2, iterNo = 1000 (on test set)



Number of correct predictions = 24

Accuracy = 0.96

Accuracy on validation set is higher than the accuracy on test set, but still it is the best accuracy on both validation and test set compared the previous parts. Because of that these hyperparameters are best choice according to my observations.