Connectionist Computing

Report

Bingbing Li

18210705

(Bingbing.li@ucdconnect.ie)

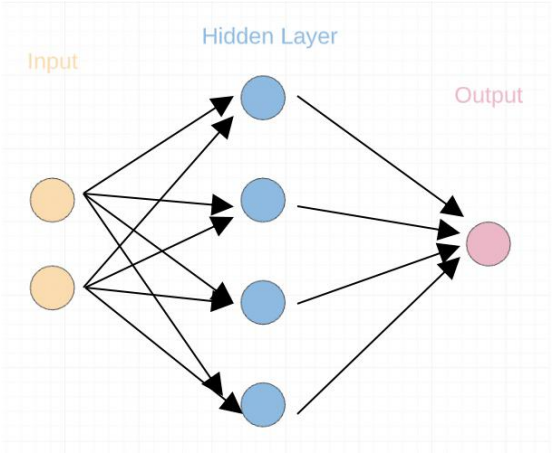
Code Choice

I chose Python as the language for building Multi-Layer Perceptron. Because in the undergraduate modules about machine learning, we all require writing in Python. Although the previous predictions will directly use built-in frameworks and libraries in Python, and the use of libraries such as TensorFlow on machine learning is not allowed in this assignment. However, Python language is still very familiar to me for data analysis and machine learning. In addition, Python is simple and consistent, which is easy to build models and understand. Python has its own library called Numpy and Matplotlib. Numpy is high-performance scientific computing, and Matplotlib is a great way to virtualize the data, which is good for analyzing models during the test. So I used python to build Multi-Layer Perceptron.

Test Quesitons

Q1. XOR Function Test

There are three layers in the network: 2 input neurons, 4 hidden neurons and one output neuron.



The first step of neural network is training in which the possible values of [(0,0),(0,1),(1,0),(1,1)] goes as input with their respective answers [0,1,1,0]. The neural network modifies its weights with different learning rates and iterates for 100,000 times according to the answers. After completion of backpropagation, the neural network is ready to predict the answers. I used the sigmoid algorithm to predict the outputs.

Before the backpropagation training, the output was all around 7.5 no matter the target value is 0 or 1. I used 5 iterations with different epochs and learning rates to train XOR Test, the results and error can be found in the “xortest.txt” file. The error of the output was smaller at every epoch. It achieved the lowest error, which is around 0.001451 when the learning rate is 1 and the epoch is 1,000,000.

The table (Figure 1) below shows the training average accuracy with learning rate equals to 1.0, 0.75, 0.5, 0.25, 0.1, 0.05 separately, and epoch equals to 100, 1000, 10000, 100000, 1000000 respectively.

Learning Rate	Epoch = 100	Epoch = 1000	Epoch = 10000	Epoch = 100000	Epoch = 1000000
1.0	0.500030	0.843220	0.983436	0.995519	
0.75	0.500026	0.617660	0.979526	0.994835	0.998329
0.5	0.500021	0.501616	0.972554	0.993666	0.997963
0.25	0.500018	0.500084	0.953999	0.990809	0.997144
0.1	0.500028	0.500031	0.847448	0.983319	0.995530
0.05	0.500083	0.500022	0.501766	0.972507	0.993673

Figure1. Accuracy with different Learning Rate and Epochs

Looking at above result, the highest average accuracy is when the learning rate is 1, and Epoch is 1,000,000, where I highlighted in yellow. It indicates that the more of the iteration and the learning rate close to 1, the higher average accuracy I can get in my model. I plot the change of the accuracy in different learning rates with the highest epoch (Figure 2). We can see that the higher of the learning rate, the increase rate of the accuracy is slower.

Figure 3 is the average accuracy in different epoch with the learning rate equals 1. When the epoch is 100, the average accuracy was about 0.5. But when the epoch is 10 times higher, the average accuracy has a significant improvement, it becomes 0.8. When the epoch was equal to 10000 or greater, the average accuracy did not change much.

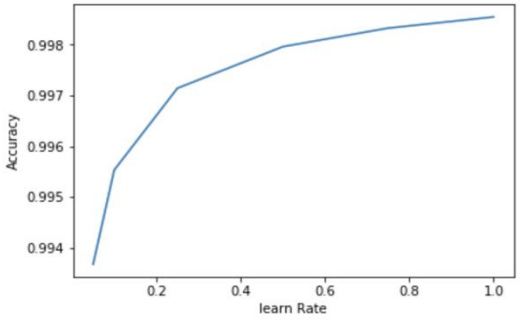


Figure2. Accuracy when Epoch equals to 1000000

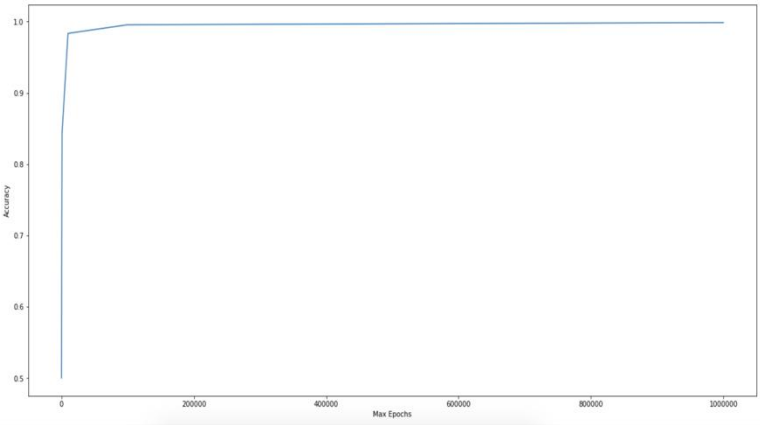


Figure3. Accuracy when Learning Rate equals to 1

Overall, the error is very low and accuracy is very high. Due to the limited amount of data, the same data are used in training data and testing data, so the trained model is prone to overfitting and has poor generalization ability.

Q2.

At the end of the training, the Multi-layer Perceptron predicts correctly in all the examples when the Epoch equals to 10,000 or higher, and the average accuracy is more than 90%.

Q3. Sin Function Test

This function is to compute sin(x1-x2+x3-x4), I used the Tanh function to get this prediction. In this function, there are 4 inputs x1, x2, x3, x4, every 4 inputs combine is a vector. I generated 10 hidden neurons and 1 output. There are more datasets than in the first test. I tried to use different learning rate, and found that the higher the learning rate, the lower the average accuracy, so I used different lower learning rates to make this model more accurate. the results and error can be found in the "sintest.txt" file

I didn't train it with many iterations, instead, I only did epoch equals 100,000, then I could get a very high predict average accuracy. From the figures below, the one highlighted in yellow has the highest accuracy of around 0.989% when the learning rate equals to 0.001. After that, the average accuracy starts decreasing, it indicates that the lower learning rate will not absolutely cause lower error. The change of accuracy with different low learning rate will be a normal distribution.

Learning Rate	Epoch = 100000
0.02	0.979934
0.001	
0.0006	0.986880
0.0001	0.974519

Figure 4. Accuracy with different Learning Rate

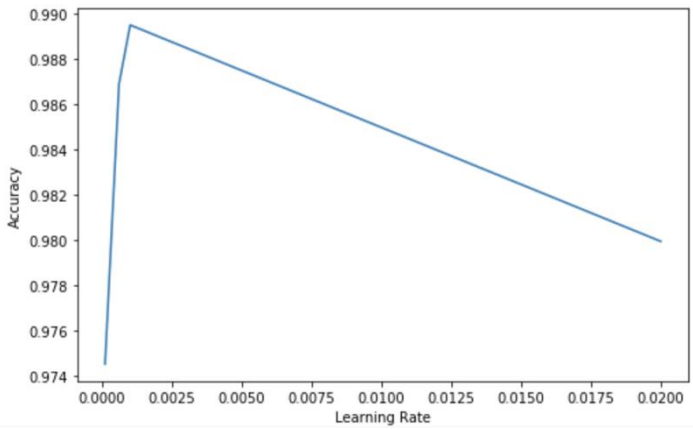


Figure 5. Accuracy with different learning Rate

Q4.

The average learning error on the training set is around 0.015643, the average test error is around. From the line chart below, we can find that the error on the training set is lower than it on the test set when the learning rate is lower. The reason for it is because the training error is calculated by the classification error of a model on the same data the model was trained on, but the test error is using a new dataset to calculate the error. However, when the learning rate was high, the error was higher relatively and the training set's error is higher than test set error, I think the module that I created is more adapted to lower learning rate. Overall, I think the module that I created has very high accuracy. The model is a little overfitting. In this way, I will try to change my model's capacity or limit the capacity of the neural net, such as hidden units, weights. So I am very satisfied with what I learned from this test with classification.

Figure6. comparison of train error and test error in different learning rate

Letter Recognition Test

This test is to apply the MLP machine learning algorithm on the UCI Letter Recognition Data Set. Given the 16 qualities, it should have the option to precisely anticipate the letter classification as one of the 26 capital letters in the English letters.

I split the dataset in a training part containing approximately 4/5 of the records, and a testing part containing 1/5. The model has 16 inputs, as there are 16 attributes to describe the letters, and another one is the target letter that we should predict. The target value

shouldn't be in the input. I selected 10 hidden units and 26 outputs (one for each letter of the alphabet). I train the model for 100000 epochs.

I normalized the 16 attributes, because they are all numeric features and have different ranges, which can skew certain distance function and have a bad influence on the accuracy of the model prediction. The output is a matrix that has 26 elements, each element represents the possibility from letter A to letter Z. Finally, I select the element with the largest predicted result as the output of my model.

The error is lower in each epoch, which can be found in the “letter_recognition1.txt” file. Accuracy in the test set is quite low, which is about 0.318. I also print each letter’s prediction accuracy. I tried different learning rates and found that the lower learning rate, the higher accuracy. I also try to use different numbers of hidden neurons to get the effect of the hidden neurons, but the more neurons, the higher errors. In this test, I trained the model took more than one hour to get each result. Due to the limit time, I didn’t improve my model, but I will try to do more tests with different parameters and adjust the model in the future.

Conclusion

MLP plays a very important role in research for their ability to solve problems stochastically, often allowing for approximate solutions to an extremely complex problems such as fitness approximation.

The predictive model that was executed was more accurate than I hoped in the first two tests. However, not all that precise in the letter recognition test. It despite everything gives knowledge on the ways to deal with an issue this way.

Due to the limited amount of data in the XOR test, the error was very low and easy to overfitting. While in the SIN test, there are more data and I split them into train set and test set. The accuracy result is also very high. It can be changed due to different learning rates or the number of hidden neurons. It’ s very easy to change these parameters and train again to get higher accuracy. The flexibility that the MLP model would provide would be a good fit for the tests. For the letter recognition test, I didn’t get very good accuracy, but I get the

effect of how to change the parameters on the result. Then I can change the parameters or modify the model in the future to study this experiment.

Through experiments, I have a deeper understanding of the Multi-Layer Perceptron. At the same time, if possible, I will continue to study this data set in the later stage, analyze each feature and dig out more useful information and rules. I will try to improve the accuracy of data prediction and improve my ability through experiments.