Project 1: Single-layer Linear Neural Networks

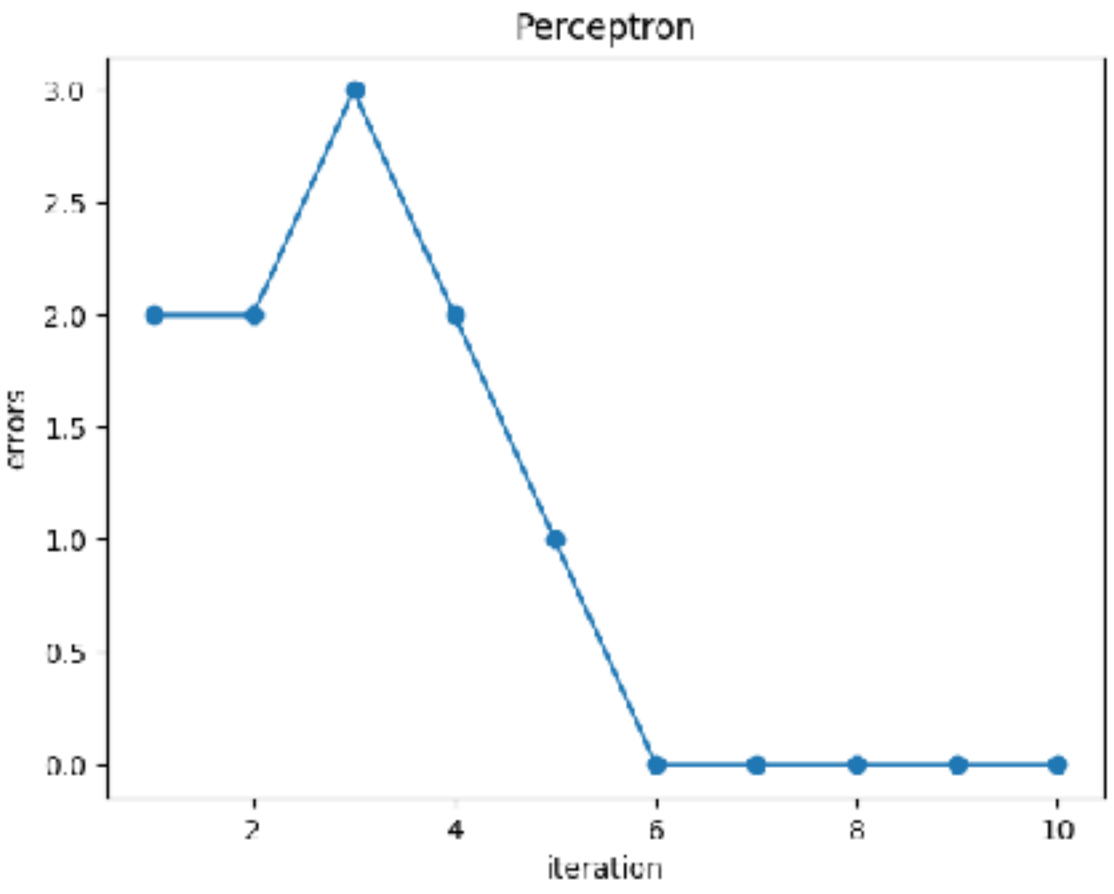
Analyzing the predictive power and the running time of Perceptron, Adaline and SGD.

A. Perceptron

For this classifier, I created a module name perceptron.py and a class named Perceptron and integrated functions like init function to initialize the class, learn function for training, testdatairis for testing the dataset of iris and finally predict function to predict the output. I divided the iris data into training data and test data in ration 2:1. According to the outcome of the program, the accuracy was 0.78. It took about 0.1607 seconds to run the Perceptron algorithm and test the dataset.

I used eta=0.01 and n_iter(epoch)=10 for this case and the error was [2, 2, 3, 2, 1, 0, 0, 0, 0, 0]. It converged in the 6th iteration.

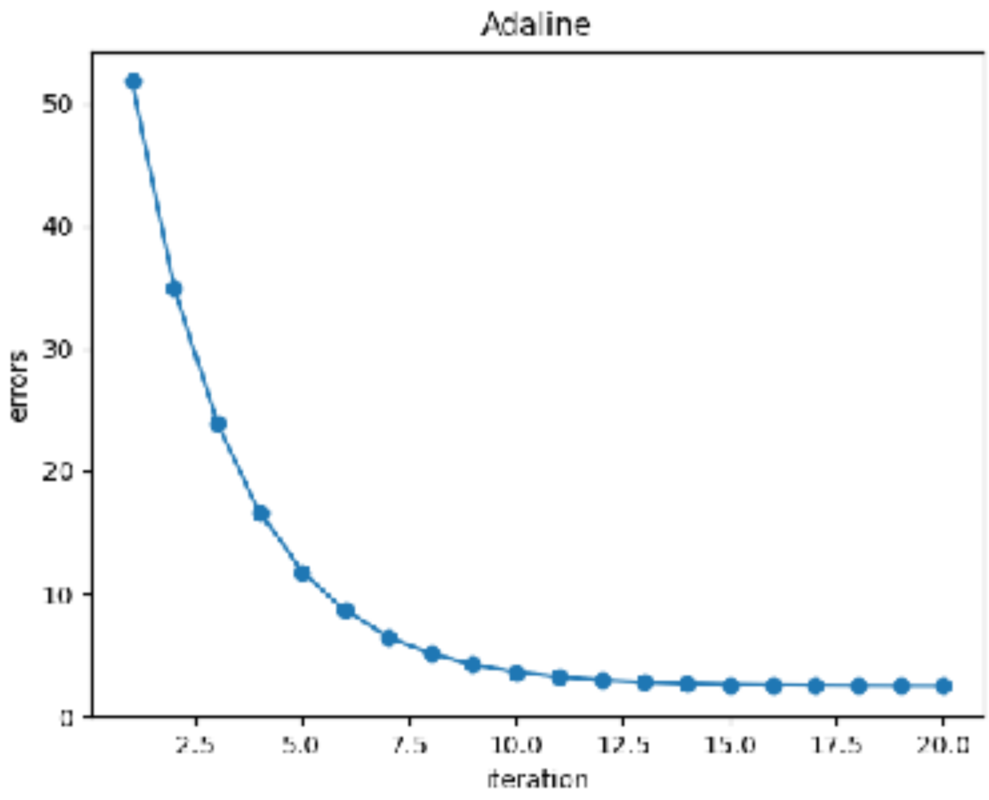
The diagram for errors in this classification algorithm is as follows:



B. Adaline

For this classifier, I created a module named adaline.py and a class named Adaline and integrated functions like init function to initialize the class, learn function for training, testdatairis for testing the dataset of iris and finally predict function to predict the output. These functions are all similar to the perceptron except the implementaion of changing weight. In the perceptron classifier we just add 0 or 1 to the weight but in the Adaline, we calculate values like sum of errors till the iteration. Instead of only 0 or 1 you get real numbers as update to the weights in Adaline. I also divided the iris data into training data and test data in ration 2:1. I also standarize the dataset so that it would converge much faster. According to the outcome of the program, the accuracy was 0.78. It took about 0.162 seconds to run the Adaline algorithm and test the dataset.

I used eta = 0.01 and n_iter(epoch) = 20 for this case and the error was [51.820849896477313, 35.028623878248908, 23.946153751185413, 16.631986827519402, 11.804810469370524, 8.6189887844326467, 6.5164221791982389, 5.128778184448767, 4.2129661234321718, 3.6085519262195551, 3.2096529191605314, 2.9463890538071347, 2.7726411587902837, 2.6579716769673247, 2.5822925439322328, 2.532346114543242, 2.499382658057121, 2.4776275601090814, 2.4632697124444074, 2.4537938741809193]. The sum of errors value seem to be in range of 2 after 12th iteration, so it may be the converging point. The diagram for errors in this classification algorithm is as follows:

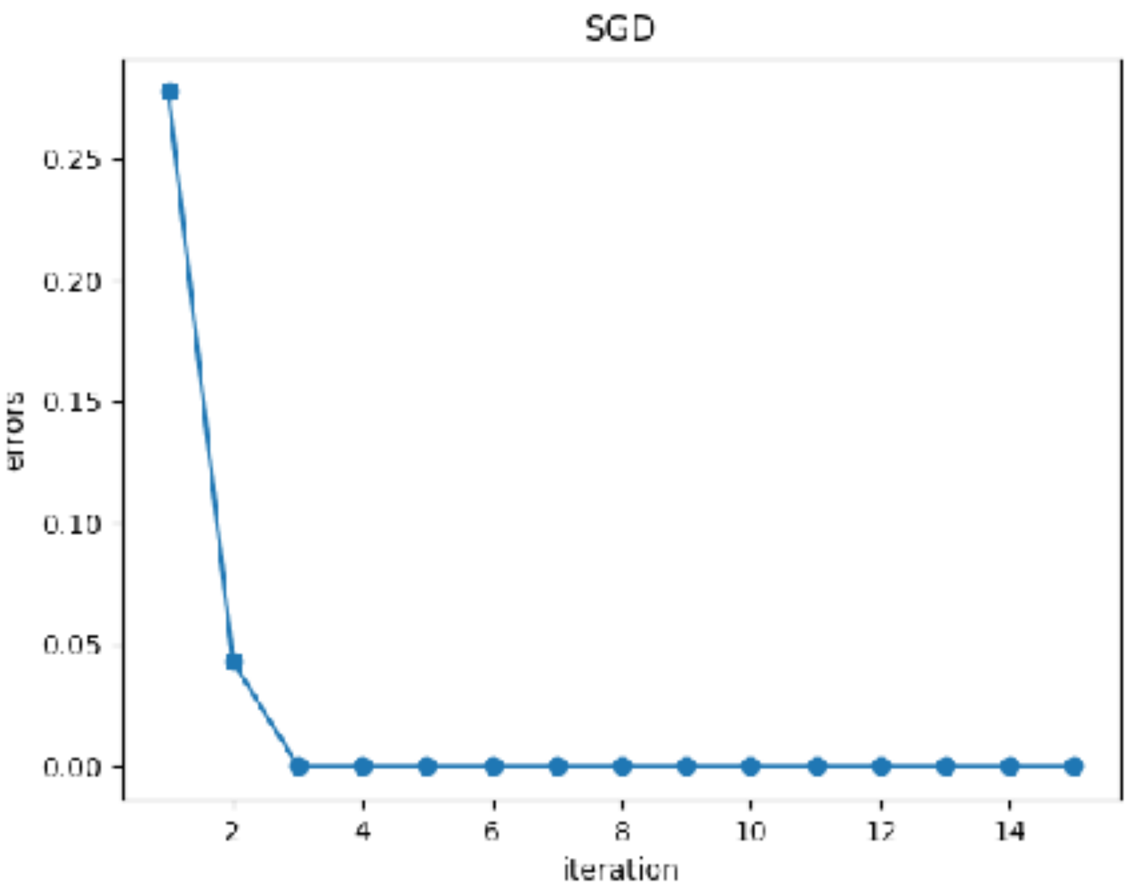


C. SGD

For this classifier, I created a module name sgd.py and a class named SGD and two different functions other than that similar to adaline which are shuffle and update weights. We need to update the weights before each iteration to avoid repetitive cycles in this classfication algorithm and use permutation in the shuffle function. I also divided the iris data into training data and test data in ration 2:1 same as before. According to the outcome of the program, the accuracy was 0.78. It took about 0.1610 seconds to run the SGD algorithm and test the dataset.

I used eta = 0.01 and n_iter(epoch) = 15 for this case and the cost values for this algorithm was [0.28236679211207444, 0.034841918469316745, 0.004946029146162525, 4.313320400885156e-07, 9.8607613152626476e-32, 9.8607613152626476e-32, 9.8607613152626476e-32, 9.8607613152626476e-32, 9.8607613152626476e-32, 9.8607613152626476e-32, 9.8607613152626476e-32, 9.8607613152626476e-32, 9.8607613152626476e-32, 9.8607613152626476e-32, 9.8607613152626476e-32]. The sum of errors value seem to be same after 6th iteration, so it may be the converging point.

The diagram for errors in this classification algorithm is as follows:



D. Multiclass classifier using one-vs-rest strategy

For the Multiclass classifier, I have tried implementing it. I have created a module name onevsrest.py and a class named Onevsrest. I know that we need to train the classifier using SGD binary classsifier using the same number of times as the classsification groups present. And then when a test instance is passed it would be passed to the model which has highest probability of predicting it correct as the final prediction. I have just transferred this logic to program code in the onevsrest.py file.

Analysis:

Looking at the runtime, accuracy and the way the weights of the input are updated, SGD classifier is the best among all.

Discussion

I tried running the algorithms again to check the runtime and they changed everytime. Firstly I saw that the perceptron is fastest but then it changed and SGD was the fastest one. I only utilized the iris dataset because I couldnot find linearly separable dataset. There were many datasets like diabetes one, wine quality one and others, but none of them were linearly separable. I tried plotting the graph using any two columns of the graph and visualizing if they are linearly separable or not.

Conclusion

This project about implementing the classifiers helped me get familiar with three main classification algorithms, how they are utilized to train and test datasets and what are the runtime for these algorithms, their accuracy and finally some idea about the convergence point. I am quite confused in the convergence of Adaline and SGD.