COMP5400M: BIC

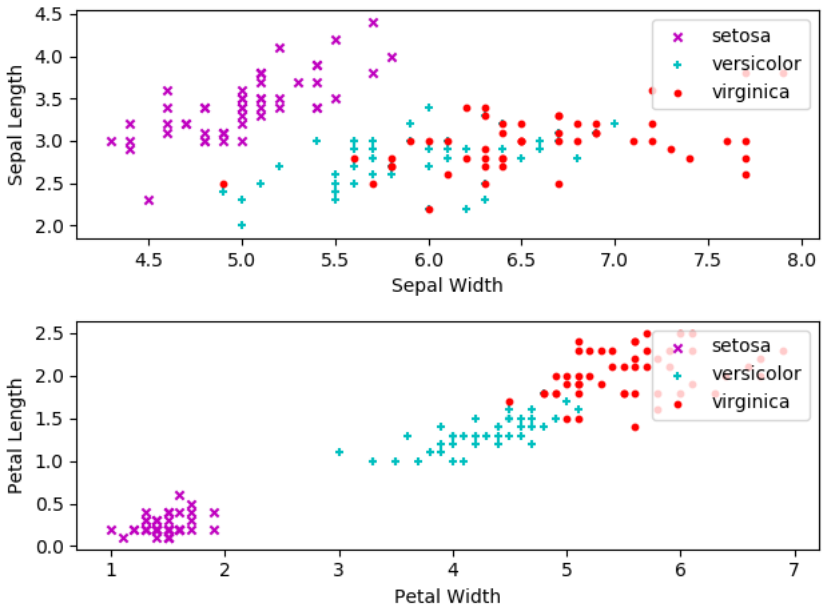
Coursework 1

Perceptron Algorithm

ID: 201268437

Data: 15th/3/2019

1.



2.

Based on the plots, a perceptron can classify setosa and non-setosa. From the first figure, setosa data value distributed in the upper left, there is no intersection between setosa and non-setosa. So, there must be one perceptron can split setosa. Obviously happened on second figure, setosa cluster is far away from non-setosa.

If just based on the plots without using machine, I can calculate it because weights and bias have to update multiple times, there are too much works to do to update weight and bias.

But if using machine, I built a perceptron to calculate four weight and bias. Firstly, set the initialize value for weights and bias. Secondly load data to test the weights and bias. Thirdly, predict the dot of weights and add bias. If the predict value more than 0 then pass to next value, and if not, to use formula like w+learning_rate*x*y to update weights and bias, utile there is no error classify point.

[2.9,7.7,-11.4,-4.8] for four weights and 1 for bias is one of my results. I have done some tests to prove it.

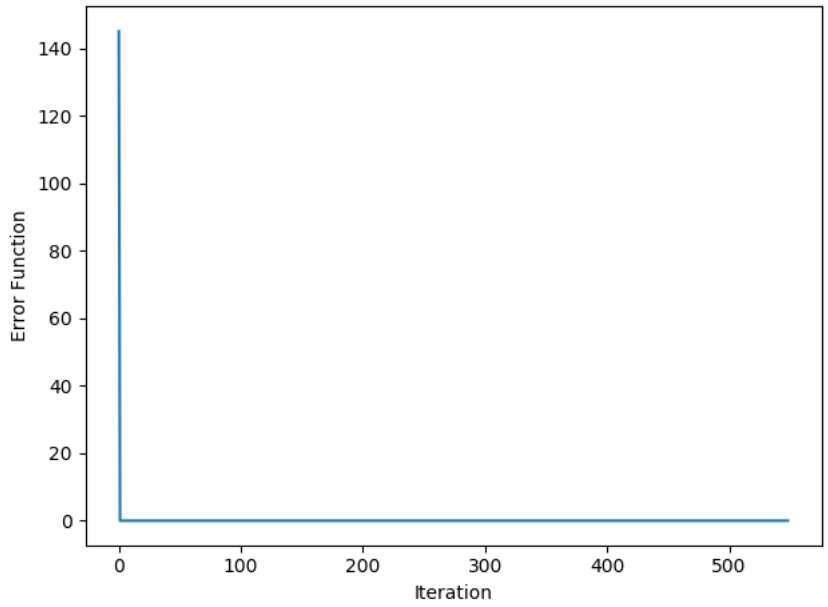
Test: Choose the relative closest point.

Point	Sample_Mark	Process	Result
4.5,2.3,1.3,0.3,Iris-setosa		sign(4.5*0.5+2.3*0.88+1.3*(-1.74)+0.3*(-0.82)+0.2)	1
5.0,3.0,1.6,0.2,Iris-setosa	1	sign(5.0*0.5+3.0*0.88+1.6*(-1.74)+0.2*(-0.82)+0.2)	1
4.9,2.5,4.5,1.7,Iris-virginica	-1	sign(4.9*0.5+2.5*0.88+4.5*(-1.74)+1.7*(-0.82)+0.2)	-1
4.9,2.4,3.3,1.0,Iris-versicolor	-1	sign(4.9*0.5+2.4*0.88+3.3*(-1.74)+1.0*(-0.82)+0.2)	-1

3.
This task that I used error function to collect information to generate plot.
The error function which I used is -$\sum y((weight * x) + bias)$
Setosa:

I expect the algorithm will be converging. Because the dataset is linear separability to compare setosa vs. non-setosa. All datasets of linear separability can be converging, even if without learning rate. If a dataset is not linear separability, and learning rate set a high rate, the dataset will not be converging.

After testing this algorithm, it does converge with loss rate, but it was so fast. I set 1000 iteration to loop the train, but it converges to 0 loss after two iterations. This algorithm without learning rate that is the learning rate is 1, so it can be converge in theory. The output after test is correct that can classify setosa and non-setosa.
Here is this algorithm error function plot.




```
================================setosa Information================================
Accuracy :1.0
Weight : [ 1.8  5.  -7.3 -3.2]
Bias: 1.0
Iterations :549
Average Error Function: 0.25622950819672136
Learning Rate: 1
[[100   0]
 [  0  50]]
```

Versicolor:

This flower looks like that is different to convergence. Because based on the plot, there are many points cover non-versicolor that means weights and bias of this flower will keep update, it cannot reduce the error function. After testing this algorithm, this

error function of it is very unstable. Looks like output is 75% accuracy that is the average of accuracy of this algorithm.

I have tried to set learning rate as 0.3 and stop this train after 500 iterations and find the accuracy more than 73%. Because I test many times, and maximum accuracy is around 74 or 75 percentage.

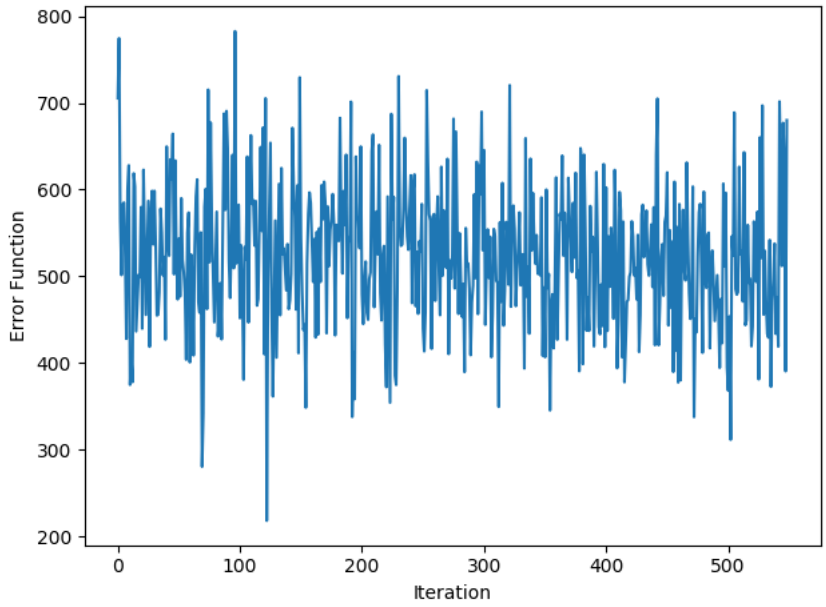
This is learning rate which is 1

```
================================versicolor Information================================
Accuracy :0.74
Weight : [  8.9 -42.1  22.8 -75.2]
Bias: 1.0
Iterations :549
Average Error Function: 1771.6136976320583
Learning Rate: 1
[[76 24]
 [24 26]]
```

This is learning rate which is 0.3

```
================================versicolor Information================================
Accuracy :0.74
Weight : [  5.55 -13.26   6.18 -23.94]
Bias: 1.0
Iterations :549
Average Error Function: 530.7918287795993
Learning Rate: 0.3
[[74 26]
 [28 22]]
```

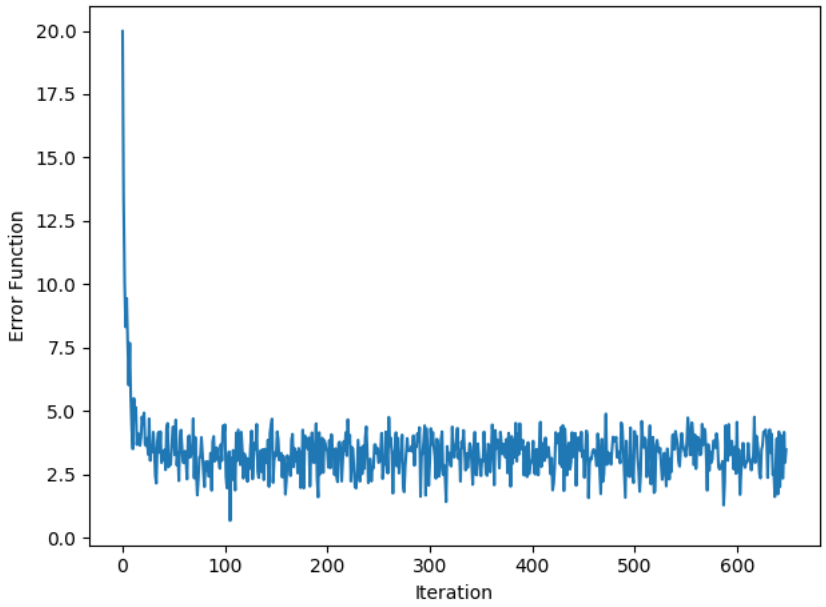
0.3 is my best learning rate which I find. Compare the results of two figures, we can see the average error function have a high difference which is around 1000. After set learning rate, it just can make the weight update slowly. But due to the versicolor dataset is a non-linear separability, there are many points is covering non-versicolor dataset. So, the error function will be a high number and cannot converging. The learning rate just can make error function small, so this algorithm is achieved my expected.



This plot is the error function of versicolor dataset which learning rate is 0.3. Compare with another two plots (setosa vs. non-setosa, virginica vs. non-virginica), This perceptron major difference is this algorithm is very unstable, it cannot converge from start point. It just keeps updating weights and bias to find a best classify to have a good accuracy. Compare with another two plots, the other plots are obviously show the error function to decrease a low number in an arrange and update weight stably.

Virginica:

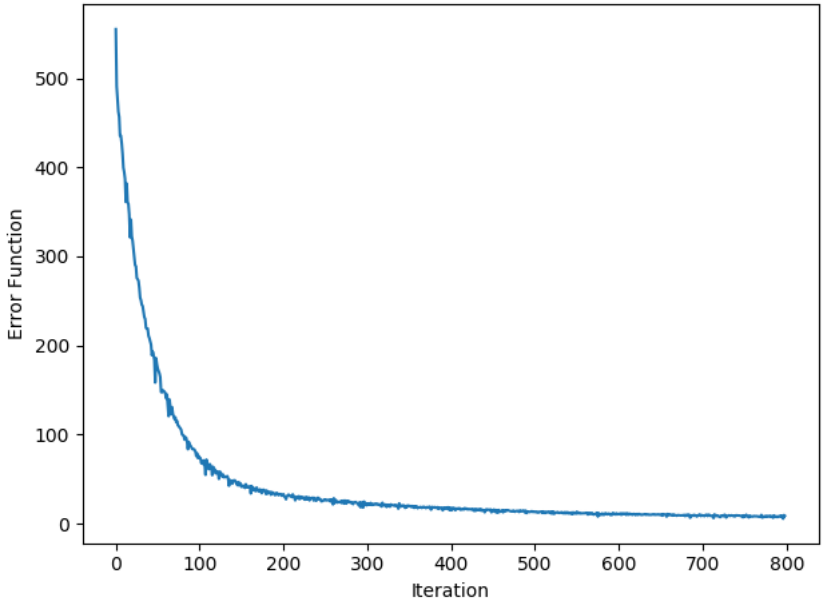
This algorithm which I expected it cannot converges. Because the reason is same as the second algorithm, there are some point is covering non-virginica dataset. It is linear separability dataset, so the error function will have a little bit large change. And the fact of testing is like my expected. The output is around 0.98 accuracy. It should be corrected.




```
==========================================virginica Information==========================================
Accuracy :0.98
Weight : [-113.  -123.8  148.   202.5]
Bias: 1.0
Iterations :549
Average Error Function: 269.39533697632055
Learning Rate: 1
[[96  4]
 [ 4 46]]
```

This is virginica algorithm which learning rate is 0.01.

```
==================================virginica Information=================================
Accuracy :0.98
Weight : [-1.443 -1.627  1.893  2.294]
Bias: 1.0
Iterations :649
Average Error Function: 3.3884781201848915
Learning Rate: 0.01
[[96  4]
 [ 4 46]]
```

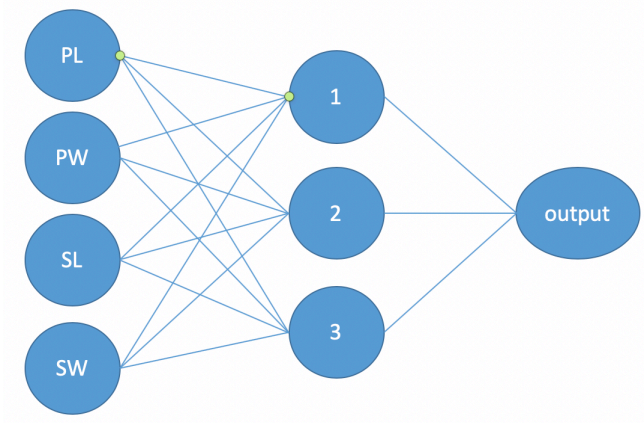


This is a plot for virginica which learning rate is 0.01. It has surprised me, because it looks like converge.

Because the final accuracy result of this algorithm will stop around 0.98, so I add some criterion to stop it, get 50 iterations data to get a maximum accuracy to get it if accuracy is more than 0.97 and the average of the 50 iterations data is more than 0.95.

Two screen shots show the different information between learning rate for 1 and learning rate for 0.01. From the average error function, we can know different learning rates can slow down the error function. Learning rate is the rate of speed of update weight and bias, so when learning rate is small, the weight and bias after updating will be small, then the wrong classify will be less.

4.



In this task, I used 2-layers perceptron to classify the dataset. It will receive from four inputs which are sepal length, sepal width, petal length and petal width. To train it in first layer perceptron, then get the best weights and bias after training. To use the output data which have the best weights and bias to send to second perceptron. Before send to second hidden perceptron, the data will be convert to three parameter format like (1.,0.,0.). Then use the second perceptron, to classify data. The result will display all weights, bias, accuracy and some plots.

There are some different from the first perceptron, second perceptron change the sigmod function from get 1 and -1 to get 1 and 0. Because that will easy to update weight. And the requirment of updating weight is different, the first perceptron just judge this data is the flower or not, the second perceptron need to consider when this data is not the flower which I want, it have to be 0. Only the flower which I want that should be 1.

The three screen shot is for the first perceptron result:

```
==================================setosa Information==================================
Accuracy :1.0
Weight : [ 0.39  1.05 -1.92 -0.9 ]
Bias: 1.0
```

```
==================================versicolor Information==================================
Accuracy :0.7466666666666667
Weight : [  5.49 -12.84   5.43 -21.57]
Bias: 1.0
```

```
==================================virginica Information==================================
Accuracy :0.98
Weight : [-6.512 -5.719  9.877  4.593]
Bias: 1.0
```

This screen shot is for the final result.

```
Accuracy :0.9066666666666666
Setosa Weight :[ 1.2  0.  -0.3]
Setosa Bias :-0.8
Versicolor Weight :[-2.7  0.6 -1.2]
Versicolor Bias :0.09999999999999998
Virginica Weight :[-0.03 -0.01  0.06]
Virginica Bias :-0.02000000000000001
```

This network's accuracy is around 90 percent and it can arrive 94 percent at best time. It may affect from the sample which is output from first perceptron. It might be the first perceptron did not get the best weight and best bias, then output the result. So, when train the output in second hidden perceptron, it will increase error count that generate something like (1.,1.,0.), it will affect the accuracy. And due to the second perceptron's bias is small, so I did not take it out when finish to train a flower, that is setosa's bias is the initialize bias of versicolor.