Perceptron

Neural Networks: ECE 5930

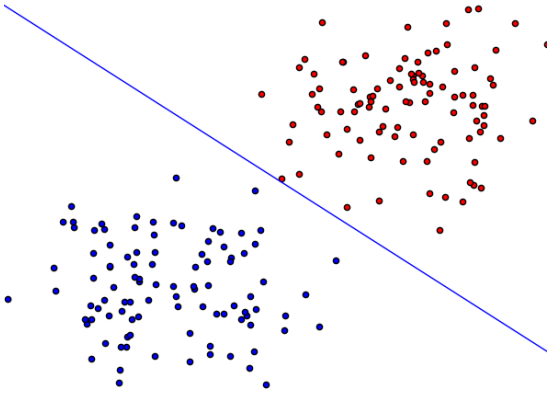

Figure: Linear Data Classifier

Clint Ferrin
Utah State University
Mon Sep 18, 2017

Contents

List of Figures

Program Listings

```
import numpy as np
import matplotlib.pyplot as plt

def graph_vector(a_t, min_max, data_1, data_2, title):
    plt.clf()
    y_min = (-a_t[0]-a_t[1]*min_max[0])/(a_t[2])
    y_max = (-a_t[0]-a_t[1]*min_max[1])/(a_t[2])
    graph_data(data_1,'red')
    graph_data(data_2, 'blue')

    plt.plot([min_max[0],min_max[1]],[y_min,y_max])

    plt.ylabel('Y Data')
    plt.xlabel('X Data')
    plt.title(title)
    plt.xlim(min_max[0:2])
    plt.ylim(min_max[2:])

    plt.show()

def graph_data(data,color):
    plt.scatter(data[:,0], data[:,1], c=color)

def main():
    data_1 = np.loadtxt("data/perceptrondat1",dtype=float, delimiter=' ')
    data_2 = np.loadtxt("data/perceptrondat2",dtype=float, delimiter=' ')

    # plt.ion()
    fig = plt.figure() # make handle to save plot

    # find max and min of data-set.
    tmp = np.vstack((data_1,data_2))
    x_min = min(tmp[:,0])
    x_max = max(tmp[:,0])
    y_min = min(tmp[:,1])
    y_max = max(tmp[:,1])
    min_max = [x_min,x_max,y_min,y_max]

    # create test data frame
    y_i = np.hstack((np.ones((len(data_1),1)),data_1))
    tmp = np.hstack((np.ones((len(data_2),1)),data_2))
    y_i = np.vstack((y_i,(tmp)*-1))

    # set initial values of the "a" vector
    a_t = np.array([-.1,1,1])

    mu = 0.5
    done = 0
    iterations = 0
    times_updated = 0

    while done == 0:
        corrected = 0
        iterations += 1
        for i in range(len(y_i-1)):
            g_x = np.matmul(a_t,y_i[i].T)

            if g_x < 0:
                times_updated += 1
                a_t = a_t + mu*y_i[i]
                graph_vector(a_t,min_max,data_1,data_2,"Iteration " + str(times_updated)
                    )
                plt.pause(0.1)
                corrected = 1

        if corrected == 0:
            done = 1
            print("Data is linearly separable.")
            print("It took %d iterations to find separating line"%(iterations))
            print("It took %d updates to find separating line"%(times_updated))

```

```
71          elif times_updated == 10000:
72              print("Linearly separable line not identified after %d iterations."%(
                    times_updated))
73              done = 1
74
75      graph_vector(a_t,min_max,data_1,data_2,"Final Output")
76      plt.pause(5)
77
78  if __name__ == '__main__':
79    main()
```

Plots of Data

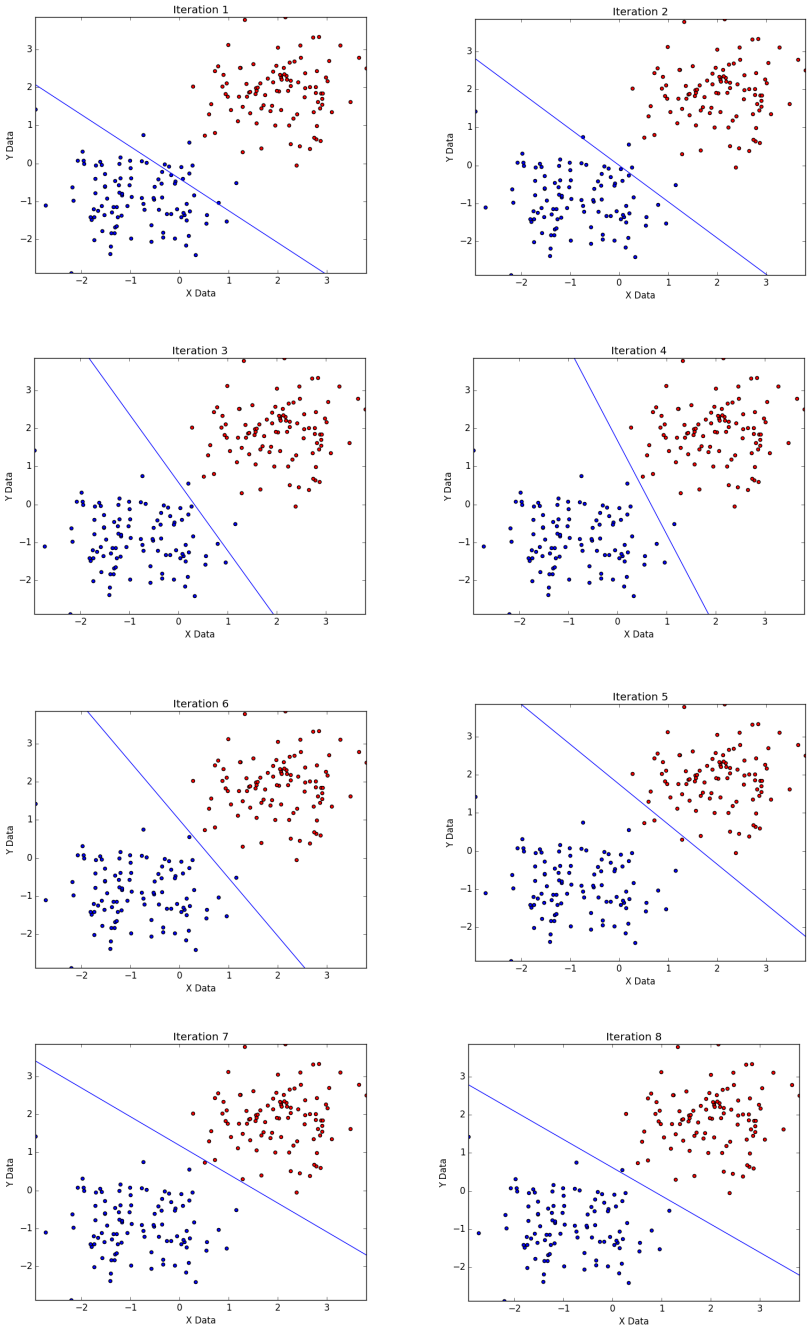

Figure 1: Iterations 1 through 8

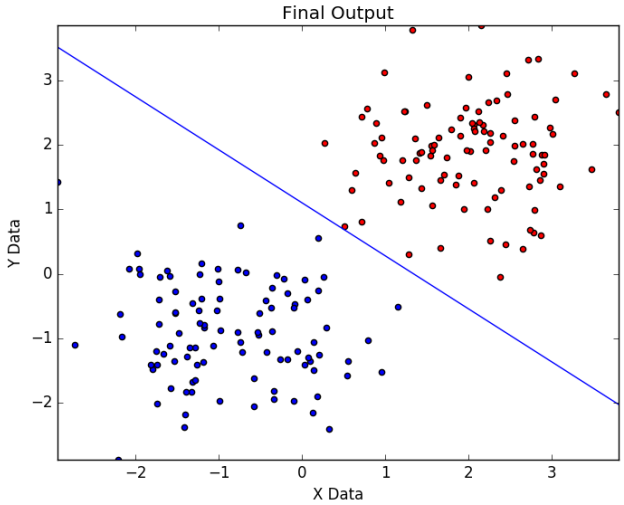

Figure 2: Final Line Output

The Perceptron algorithm was able to settle on a new line after 9 updates of the vector a_t and four times through the for loop as seen in the previous image.

Discussion Questions

Analyzing the Data

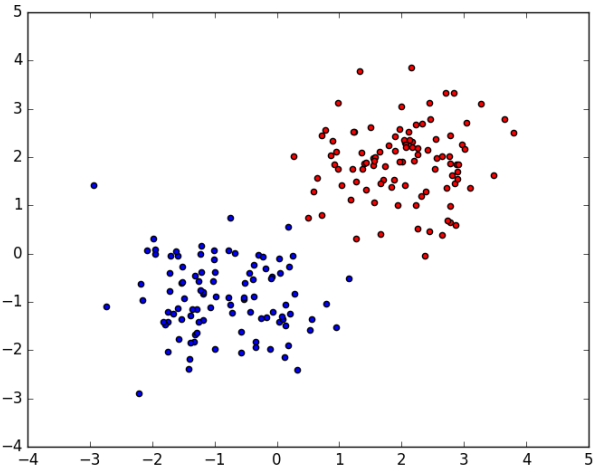

Figure 3: Linearly Seperable

As seen in Figure 3, the data appears to be linearly separable.

Different A Values

When different values of a are chosen, the program takes more or less time to converge to a specific value.

For example, when the a vector was set to $a_t = [-.1, 1, 1]$ the program only took 4 updates to the vector a_t and 3 iterations through the for loop to find the correct line. Additionally, the line was more centered between the two data sets than when I used an initial value of $a_t = [1, 1, 1]$.

The initial guess for the data is important when determining the fastest approach to the correct value.

Another example is when $a_t = [-1, 90, 90]$. In this case, the step size plays an important role. If the step size is too small, it will take a very long time for the graph to converge. In this case it took 71 iterations and 111 updates.

Data Not Linearly Separable

When the data is no longer linearly separable, the program will run continuously without a break statement.

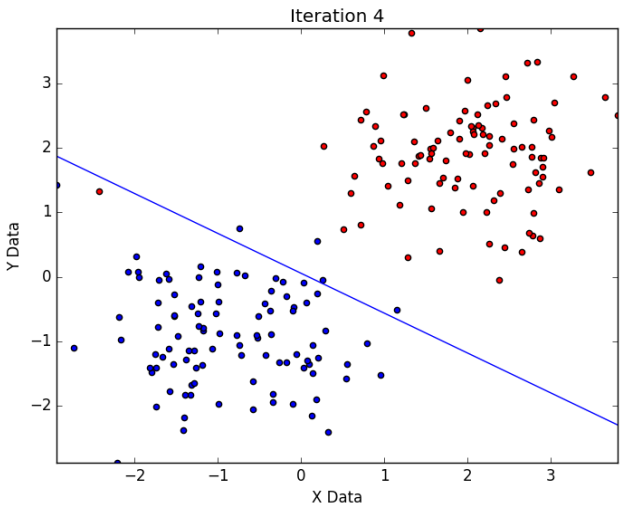

Figure 4: Not Linearly Separable

When I ran the code seen in Figure 4, the program began to jump back and forth around the misplaced point. The program will continue to run forever, but I included a statement that will stop the program from running after 10000 unsuccessful iterations.

Another possible approach to avoid the problem might be to create a count that increments every time the program does not send the line in the same direction as previously. If the program reaches a certain amount of switchbacks (i.e., 15), then you can know the data is not linearly separable.