Perceptron Learning toward Linear Classification

Experiment Report

School	School of Automatic Science
Major	Pattern Recognition and Intelligent System
Student	Tianxiang Lan
Student ID	15231087
Teacher	Yang Li

April 4,2018

1 Introduction

Linear perceptron is one of the simplest learning algorithms for a two-class classifier. Given a set of data points in d-dimensions, belonging to two classes, ω and ω , the algorithm tries to find a linear separating hyper-plane between the samples of the two classes. If the samples are in one, two or three dimensions, the separating hyperplane would be a point, line or a plane respectively. The specific algorithm that we look into is a special case of a class of algorithms that uses gradient descent on a carefully defined objective function to arrive at a solution.

2 Principle and Theory

Assume that the samples of the two classes are linearly separable in the feature space. .i.e., there exists a plane $G(X) = W^T X + w_{n+1} = 0$,where $W \in R^{\mathrm{n}}$ and $X \in R^{\mathrm{n}}$,such that all samples belonging to the first class are on one side of the plane, and all samples of the second class are on the opposite side. If such planes exist, the goal of the perceptron algorithm is to learn any one such plane, given the data points. Once the learning is completed and the plane is determined, it will be easy to classify new points in the future, as the points on one side of the plane will result in a positive value for $G(X) = W^T X + w_{n+1}$, while points on the other side will give a negative value.

According to the principle of perceptron learning, the weight vector $W \in R^{\mathrm{n}}$ can be extended to $\hat{W} = (w_1, w_2, \dots, w_n, w_{n+1})^T \in R^{n+1}$ and the feature vector may be extended to $\hat{X} = (x_1, x_2, \dots, x_n, 1)^T \in R^{n+1}$ also, thus the plane of classification can be expressed as $G(X) = \hat{W}^T \hat{X} = 0$ .The learning rule (algorithm) for the update of weights is designed as

$$\hat{W}(t+1) = \begin{cases} \hat{W}(t) & \hat{W}^T(t)\hat{X}(t) > 0 \\ \hat{W}(t) + \eta \hat{X}(t) & otherwise \end{cases}$$

where η is the learning rate which may be adjusted properly to improve the convergence efficiency of the learning course.

3 Objective

The goals of the experiment are as follows:

(1) To understand the working of linear perceptron learning algorithm.

(2) To understand the effect of various parameters on the learning rate and convergence of the algorithm.

(3) To understand the effect of data distribution on learnability of the algorithm.

4 Contents and Procedure

Stage 1:

(1)According to the above principle and theory in section 1.2, design and compile the programme codes of perceptron learning toward the linear classification of two classes.

(2)Create a linearly separable pattern dataset with more than 50 samples for each one of two classes.

(3)Initialize the weight vector W(0) and select a proper value for the learning rate $\eta \in (0,1]$

(4)Run your programme with your dataset and record the final result, pay attention to the number of iterations for convergence.

Stage 2:

Repeat the above procedure (2) ~ (4) in stage 1 for the different datasets with varying amount of separation between two classes of patterns. Note down your observations.

Stage 3:

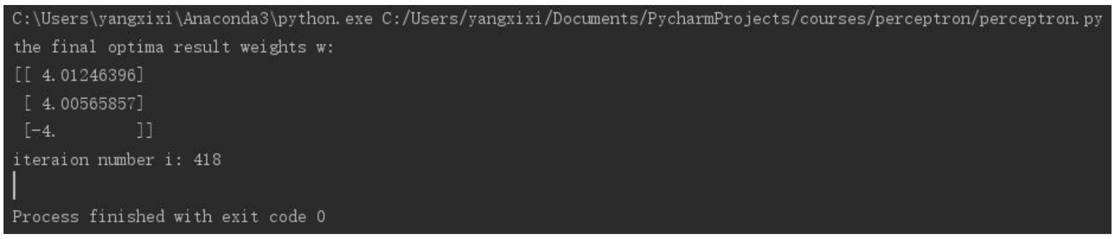
Study the effect of varying the learning rate η with different amounts of separability.

Stage 4：

Explore the questions in the previous section and design experiments to answer these questions. Complete and submit an experiment report about all experiment results with comparative analysis and a summary of experiences in this experiment study.

5 Results and Analyse

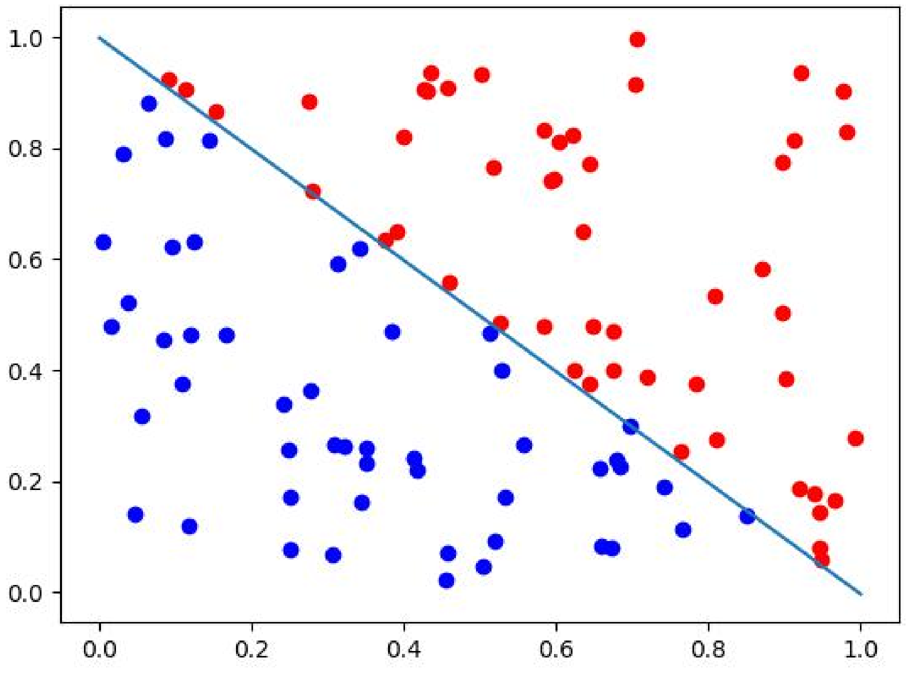
（1）Firstly, I created a dataset rand from (0,1).Then I used a line to separate them with weights [1, 1, -1].And then I set a learning rate as $\eta = 1$ .By learning, the modal got weights and iteration number like the following picture,the learned weight is the same direction as the given weights [1, 1, -1].



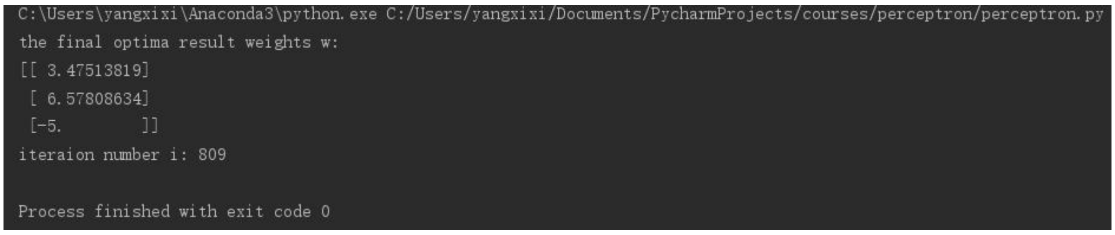
```
C:\Users\yangxixi\Anaconda3\python.exe C:/Users/yangxixi/Documents/PycharmProjects/courses/perceptron/perceptron.py
the final optima result weights w:
[[ 4.01246396]
 [ 4.00565857]
 [-4.        ]]
iteraion number i: 418

Process finished with exit code 0
```

The dataset and the decision boundary looks like the following picture.



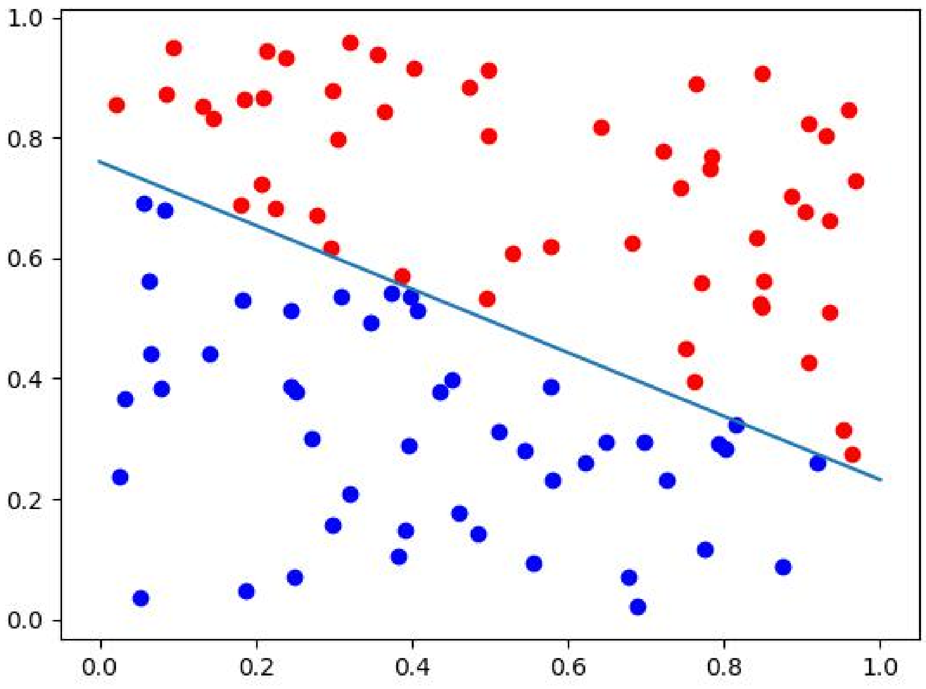
（2）I changed the distribution of dataset such as using a line to separate them with weights [0.7, 1.3, -1].The result are as follows.



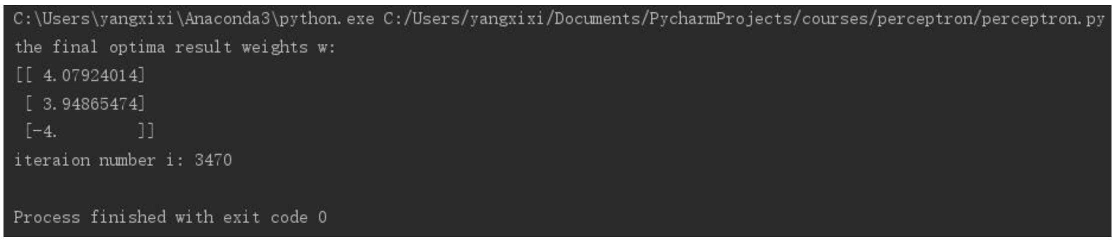
```
C:\Users\yangxixi\Anaconda3\python.exe C:/Users/yangxixi/Documents/PycharmProjects/courses/perceptron/perceptron.py
the final optima result weights w:
[[ 3.47513819]
 [ 6.57808634]
 [-5.        ]]
iteraion number i: 809

Process finished with exit code 0
```

Also the learned weight is the same direction as given.The distribution of dataset and super plane are as follows.



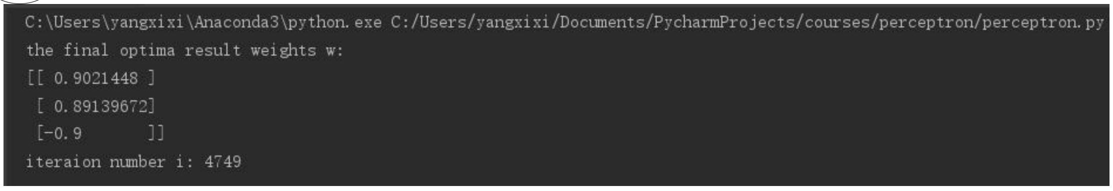
（3）I changed the learning rate to 0.5,the results are as follows.



```
C:\Users\yangxixi\Anaconda3\python.exe C:/Users/yangxixi/Documents/PycharmProjects/courses/perceptron/perceptron.py
the final optima result weights w:
[[ 4.07924014]
 [ 3.94865474]
 [-4.        ]]
iteraion number i: 3470

Process finished with exit code 0
```

And I changed the learning rate to 0.1,the results are as follows.



```
C:\Users\yangxixi\Anaconda3\python.exe C:/Users/yangxixi/Documents/PycharmProjects/courses/perceptron/perceptron.py
the final optima result weights w:
[[ 0.9021448 ]
 [ 0.89139672]
 [-0.9       ]]
iteraion number i: 4749
```

As you see,the iteration number i would go large when learning rate comes down from $\eta = 1$ to $\eta = 0.1$ .Therefore,the learning rate effects the speed of convergence.

All the code and instruction files are on my github.You can run them if you want: https://github.com/Deep-Lan/Perceptron.git