成绩

Experiments for “Pattern Recognition and Machine Learning”

Experiment 1
Perceptron Learning toward Linear Classification

院（系）名称	自动化科学与电气工程学院
专 业 名 称	自动化
学 生 学 号	15071135
学 生 姓 名	刘雨鑫

2018 年 5 月 4 日

1.1 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, w1 and w2, 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.

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

$$\widehat{W}(t+1) = \widehat{W}(t) + \frac{1}{2}\eta\{\widehat{X}(t) - \widehat{X}(t) sgn[\widehat{W}(t)^T \widehat{X}(t)]\}$$

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

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

1.3 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.

1.4 Contents and Procedure

I chose MATLAB to program the Linear perceptron,and I wrote a function named linLinear_Classification_picture(N,e) which could give the classification results by the number of samples and the learning rate.

First, I generated 200 random samples that could be linearly classified, and it is like the fig.1.

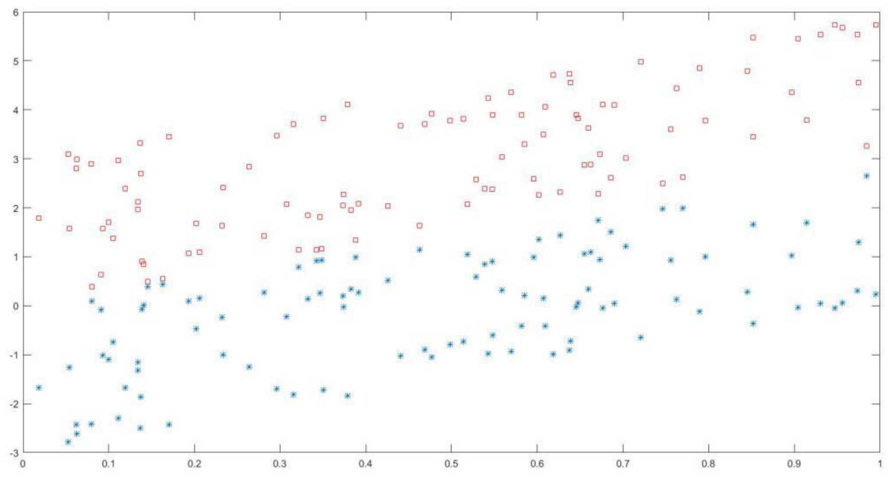

Fig.1 the random samples

I set the learning rate to be 0.8, the initializtion of the weight vector is generated by random numbers. Then I run the program I wrote and the result is shown in fig.2. As you can see, the program succeeded in separating the two samples. At this time, the number of iterations is 8600,the final weight vector is [-17.2693 5.6916 0.0005].

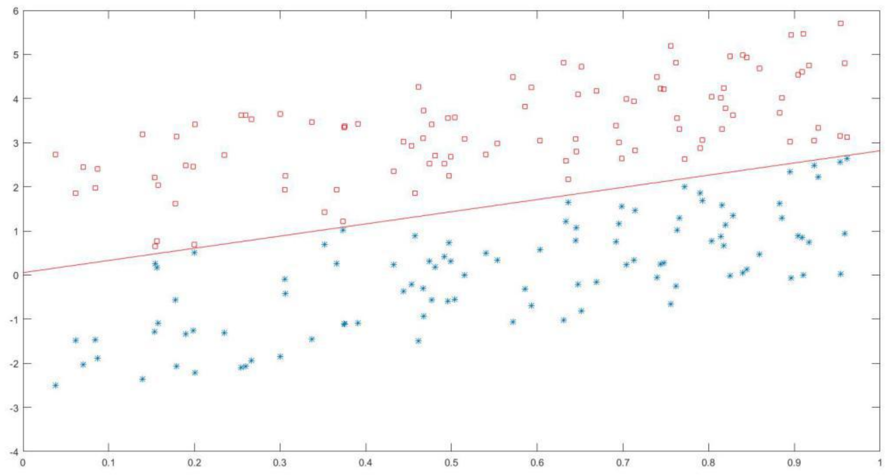

Fig.2 the result of 100 samples severally

I changed the number of samples but keep the learning rate at 0.8, then repeated the experiment， the result is shown in table 1, and fig.3. The iterations shown in table 1 is 1/10000 of the real value.

Table 1 results of different amount of samples

amount	100	300	500	700	900	1100
iterations	0.86	1.8	21.6	624.4	52.74	5.72

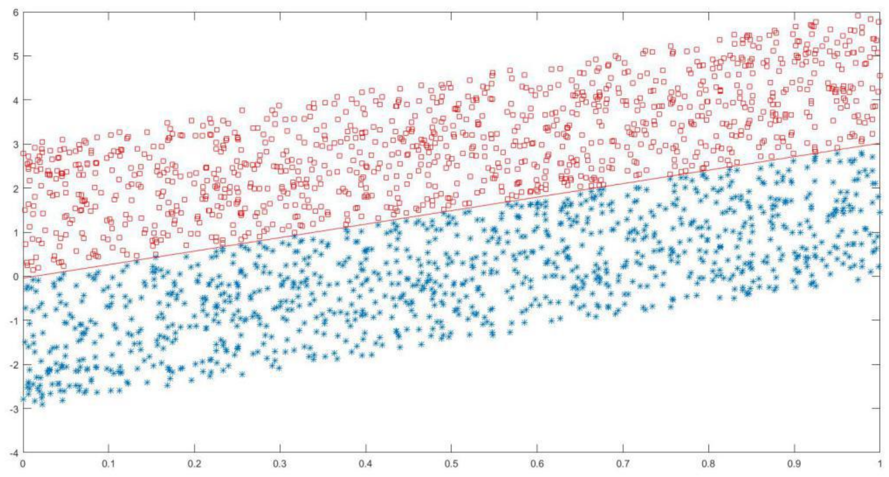

Fig.3 the result of 1100 samples severally

From the table above, we can see the iterations increases first and then decreases as the number of samples increases.

I set the number of samples at 200, and assign 0.1, 0.5, 1, 2 to the learning rate in turn and the results are shown in table 2. The iterations shown in table 1 is 1/10000 of the real

Table 2 results of same amount of samples

Learning rate	0.1	0.5	1	2
iterations	47.56	1.44	0.68	39.44

From the table above, we can see the iterations decreases first and then increases as the learning increases. The smaller the learning rate, the greater the number of iterations. Because the samples generated by the program are random and different each time, I can only roughly analyze this result by the accuracy of the classification. If sample separability is high, programs are easier to classify, if sample separability is small, the number of iterations increases.

1.5 Experience

Through this experiment, I fully understood the principle of linear perceptron and how to write the corresponding program. I've learned that learning efficiency and sample separation can affect the accuracy of the algorithm and the number of iterations. At first I had no idea, but with the help of my classmates and my self-study, I managed to figure it out. Finally, thanks to the teachers and classmates for their help.

1.6 Code

The code can be download at https://github.com/Jackboomer/Experiments-for-Pattern-Recognition-and-Machine-Learning.git .