

**** 1

**模式识别与机器学习**

**Perceptron Learning toward Linear Classification**

实验报告

|  |  |
| --- | --- |
| 院（系）名称 | 自动化科学与电气工程学院 |
| 专业名称 | 自动化 |
| 学生姓名 | 伍翌嘉 |
| 学号 | 15031131 |
| 任课老师 | 李阳、秦曾昌 |

2018年5月4日

**Experiment 1: Perceptron Learning toward Linear Classification**

**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, ω1 and ω2, 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 hyper-plane 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 **,** where and  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 , while points on the other side will give a negative value. According to the principle of perceptron learning, the weight vector  can be extended to and the feature vector may be extended to also, thus the plane of classification can be expressed as . The learning rule (algorithm) for the update of weights is designed as



where  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.

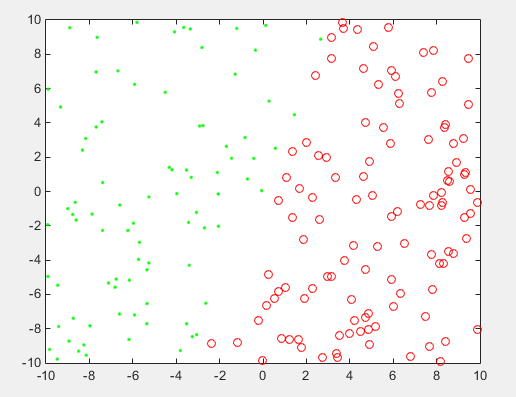
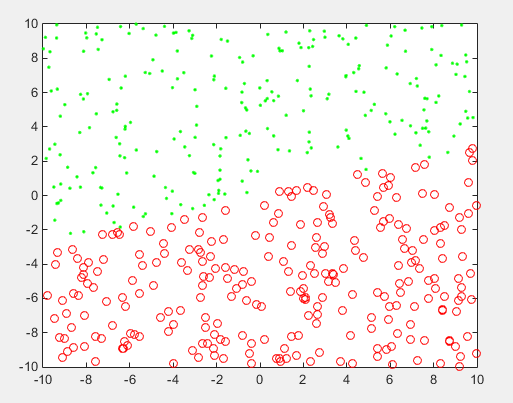
(2) To understand the effect of various parameters on the learning rate and convergence of the algorithm.

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

**4 Contents and Procedure**

**Stage 1:**

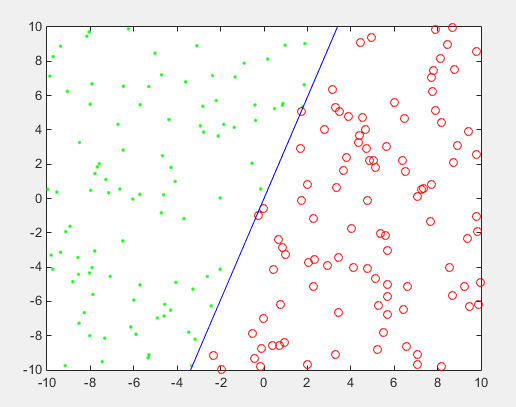
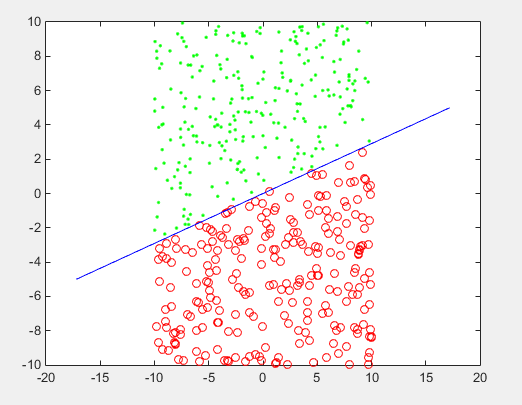
At first, I create a random data generator which can create data in the range of [-10,10] of both x and y direction and separate by the input vector to ensure that designed dataset is linear separable. The number of data can also be changed as shown in Fig.1.



1. **Number=500,W0=(-2,7) (b)Number=200,W0=(-3,1)**

**Fig.1 Data distribution map**

Then, a perceptron trainer is programmed to learn form dataset and finally decide the hyperplane function by predicted weight vector W.



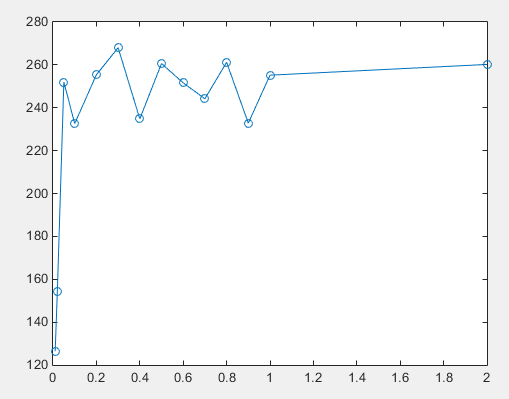
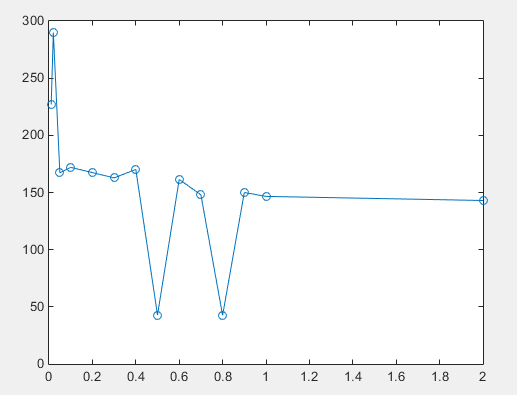
**(a)W=(-426.36,1466.86) (b)W=(-2645.56,868.37)**

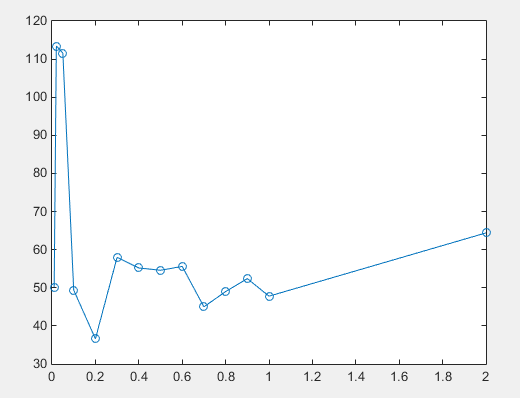
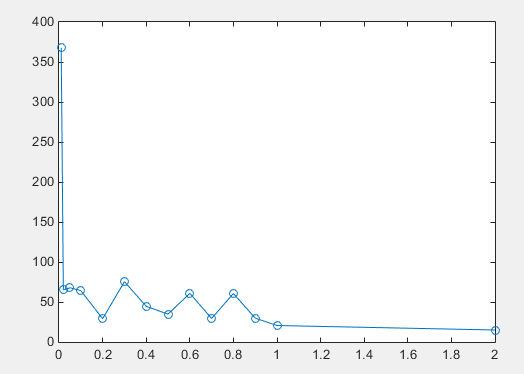
**Fig.2 hyper-plane split result**

**Stage 2: Influence of learning rate , separability to number of iteration**

The second step is to design experiment to explore the relationship among learning rate , separability and number of iteration.

I firstly designed experiment to observe the number of iteration and training time. The initial weight vector is set as (1,1) because when it begin at (0,0), whatever change of learning rate, number of iteration is always the same, just the final vector changing proportionally. Number of dataset is fixed and  is varying from 0.01 to 100 to see the general trend. From those rough test of the large range of learning rate, it can be easily seen that when  is larger than 1, the number of iteration is stable, hardly change. However, in the range of [0.01,1], the number of iteration is fluctuating and changed a lot in each test of 5 random datasets. Fig.3 shows the trend.



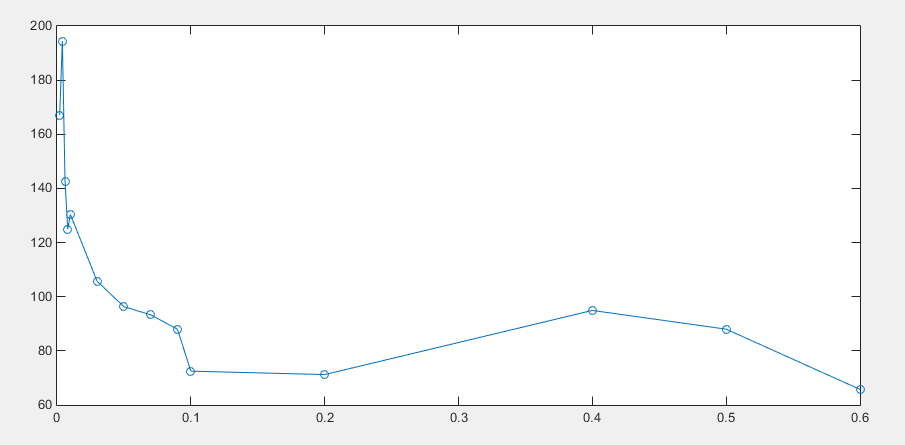


**Fig.9 number of iteration - learning rate**

Therefore, I train the model with 200 random datasets and analyse the average performance of them. It is obvious that when learning rate is smaller than 0.1, the less the rate, the more the number of iteration, as shown in Fig.10 and Table 1. However, the training time when  is 0.01 is much smaller than the other, which shouldn’t be the an error because it is an average of 200 tests. It may needs some more analysis.

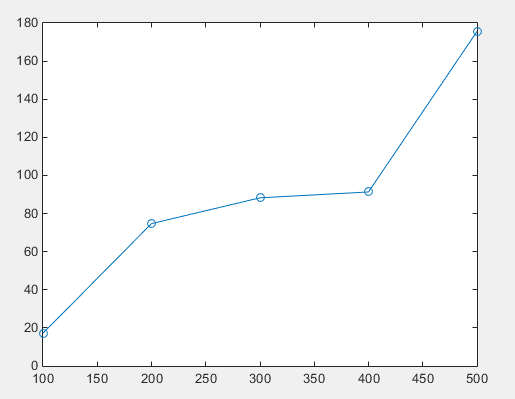
**Table 1 training parameter**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0.001 | 0.002 | 0.004 | 0.006 | 0.008 | 0.01 | 0.03 | 0.05 | 0.07 | 0.09 |
| iteration | 242.48 | 167.05 | 194.45 | 142.73 | 124.95 | 130.36 | 105.67 | 96.32 | 93.315 | 87.98 |
| time | 0.0500 | 0.0864 | 0.2723 | 0.1348 | 0.105 | 0.1199 | 0.0681 | 0.0545 | 0.0534 | 0.0506 |
|  | 0.1 | 0.2 | 0.4 | 0.5 | 0.6 | 0.8 | 1 | 2 | 5 | 10 |
| iteration | 72.505 | 71.24 | 94.93 | 88.01 | 65.76 | 87.6 | 73.28 | 71.945 | 69.14 | 73.415 |
| time | 0.0487 | 0.0479 | 0.0587 | 0.0515 | 0.0390 | 0.0607 | 0.0512 | 0.0463 | 0.0493 | 0.0556 |



**Fig.10 number of iterating of 200 random dataset**

As for the influence of separability, I define it as the difficulty of finding the final hyper-plane, which can also be seen as the degree of dispersion in this experiment, so I take the number of data in a fixed religion [-10,10] as its measurement. The result is shown in Fig.11. It is similar to intuitive thought that the more the data, the more the number of iteration.



**Fig.11 separability influence**

**Stage 3:Synthesis analysis of separability and learning rate**

Combining thinking the influence of separability and learning rate, when separability is bigger, which means data are more distributed, the nice learning rate is bigger.

To be honest, I haven’t finish the analysis of this part, just make a speculation. I will finish it and also try to figure out the reason of problem above: Why the training time is less when learning rate is 0.01 when number of iteration is high? When I finish, I will upload it as additional document in Github.

**5 Experiences**

From this experiment, I understand more about perceptron learning. Initially, I thought I have understand the principle of it, but when I have to program, many problems came out, especially don’t know how to change the thought into coding. Through the process, I found the influence of learning rate and separability and their combined effect to perceptron learning. Though the result is not the best fitting of the designed separated line, it can always find out a viable solution. Perceptron is a nice classifier for beginners, but not practical in real data analysis because seldom of them are linear separable.

**Appendix**

The link of electronic version and codes of this project is:

*https://github.com/sixer51/Pattern-Recognition-Machine-Learning*