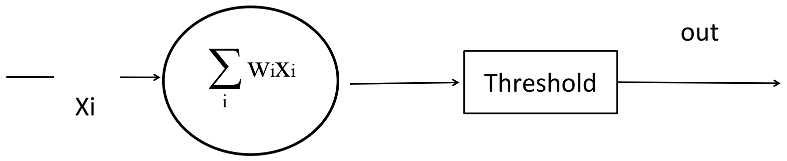
Report

Contents

1 Basic theory

Perceptron is a linear classifier. Also, it is used in supervised learning. It helps to classify the given input data.



The basic principle of PLA is to correct point by point. Firstly, randomly select a classification surface on the hyperplane, and then statistics the points which have not been classified correctly; then randomly correct a certain error point, and change the position of the straight line, so that the error point can be corrected. Repeat this process, and the classification surface will change continuously, until all the points are completely classified correctly, which means we get the best classification surface.

2 Test

Test1

(1)Requirment:

DataEmit <5,2,3> 10 10

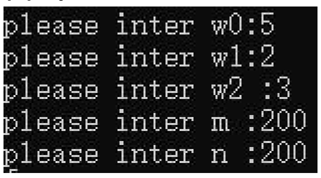
DataEmit <5,2,3> 50 50

DataEmit <5,2,3> 100 100

DataEmit <5,2,3> 150 150

DataEmit <5,2,3> 200 200

(2)Input format:

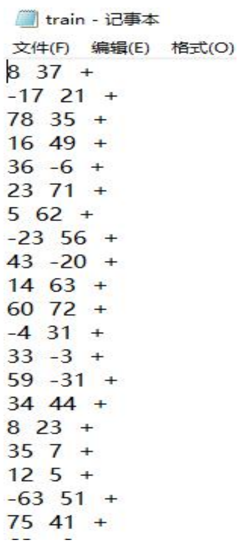


```
please inter w0:5
please inter w1:2
please inter w2 :3
please inter m :200
please inter n :200
```

(3)Output w:

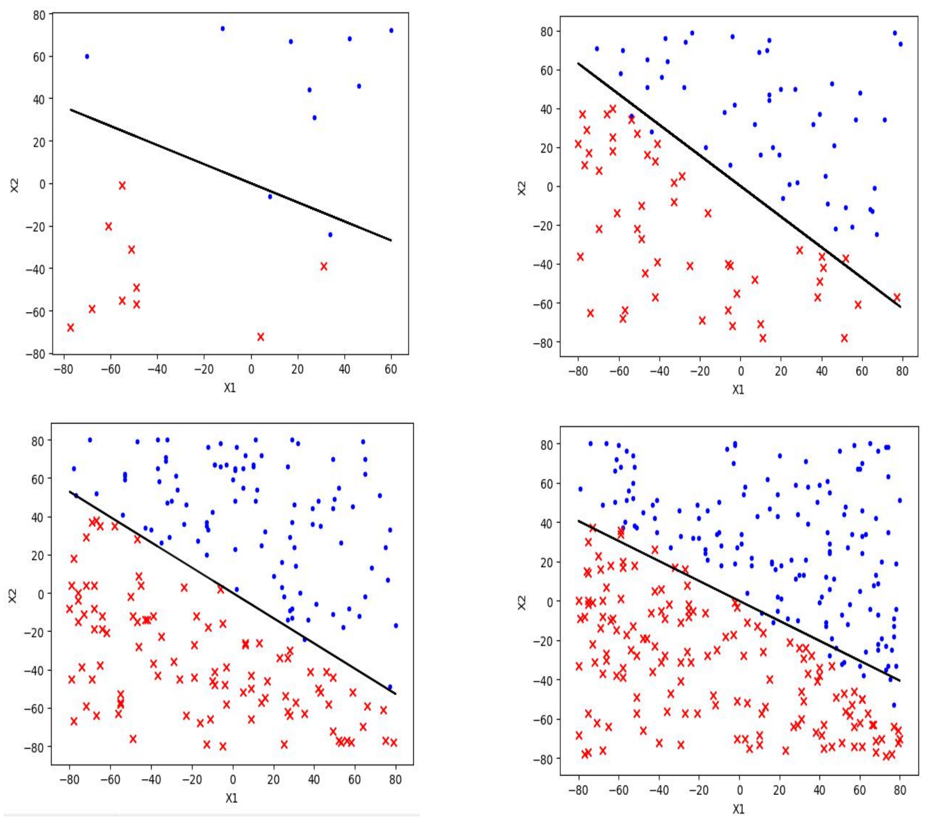
1. [2 40 89]
2. [1 104 132]
3. [7 103 156]
4.[8 104 205]
5.[14 131 157]

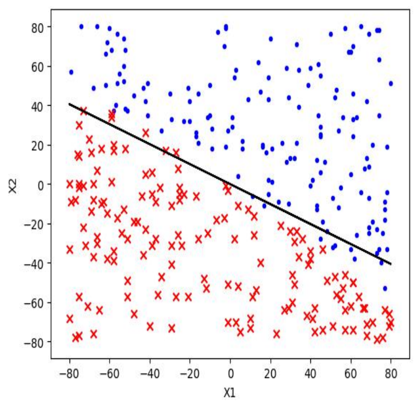
(4)Train.txt :



```
8 37 +
-17 21 +
78 35 +
16 49 +
36 -6 +
23 71 +
5 62 +
-23 56 +
43 -20 +
14 63 +
60 72 +
-4 31 +
33 -3 +
59 -31 +
34 44 +
8 23 +
35 7 +
12 5 +
-63 51 +
75 41 +
```

(5) Output Graph:





Test2

(1)Requirment:

DataEmit <-10,2,6> 1 10

DataEmit <-10,2,6> 5 50

DataEmit <-10,2,6> 5 100

DataEmit <-10,2,6> 5 150

DataEmit <-10,2,6> 5 200

DataEmit <-10,2,6> 5 200

(2)Output w:

1[1 5 43]

2[-2 64 74]

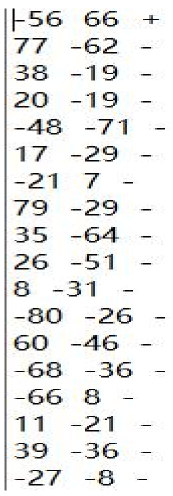
3[-4 3 138]

4[-7 26 119]

5[-10 52 145]

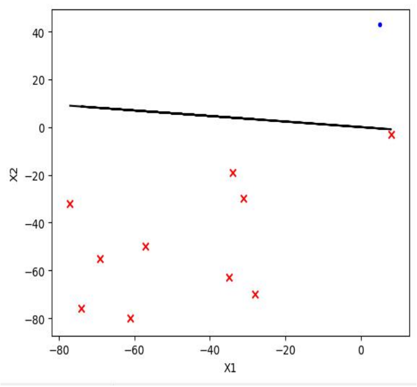
6 [-10 55 152]

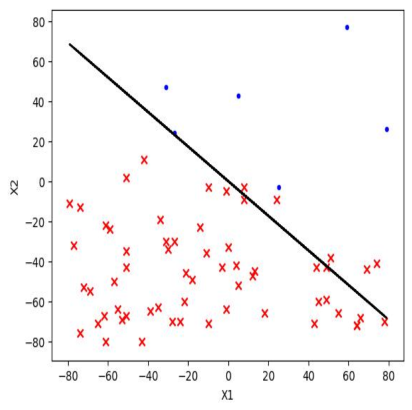
(3)Train.txt :

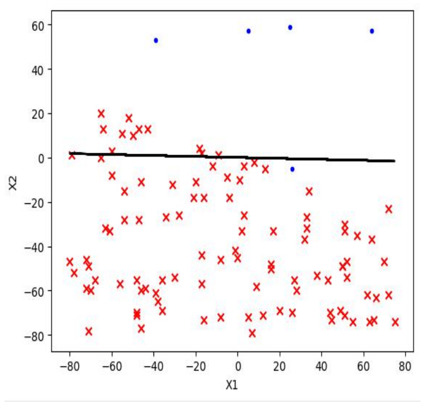


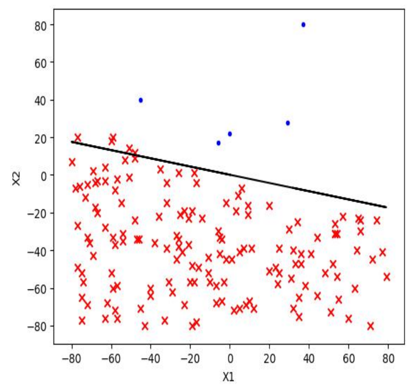
```
-56 66 +
77 -62 -
38 -19 -
20 -19 -
-48 -71 -
17 -29 -
-21 7 -
79 -29 -
35 -64 -
26 -51 -
8 -31 -
-80 -26 -
60 -46 -
-68 -36 -
-66 8 -
11 -21 -
39 -36 -
-27 -8 -
```

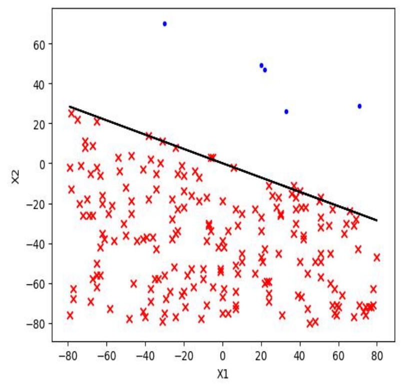
(4)Output Graph:

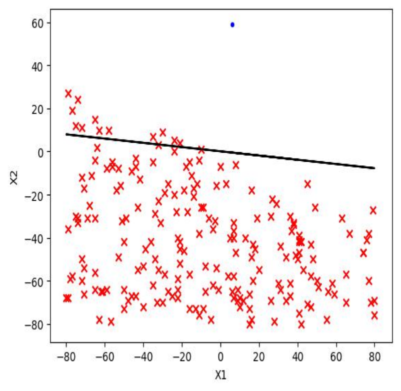












Test3

(1)Requirment:

DataEmit <-10,10,-28> 1 1

DataEmit <-10,10,-28> 5 5

DataEmit <-10,10,-28> 500 500

DataEmit <-10,10,-28> 700 700

DataEmit <-10,10,-28> 2000 2000

DataEmit <-10,10,-28> 10000 10000

(2)Output w:

1[1 57 -64]

2[1 67 -70]

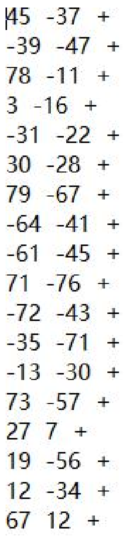
3 [1 75 -212]

4 [15 113 -252]

5 [-31 99 -303]

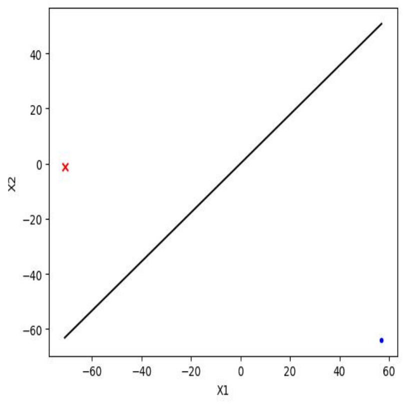
6[-91 238 -504]

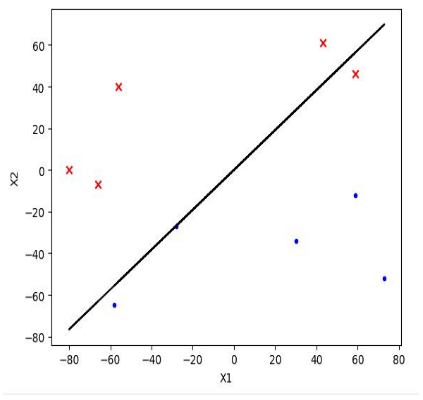
(3)Train.txt :

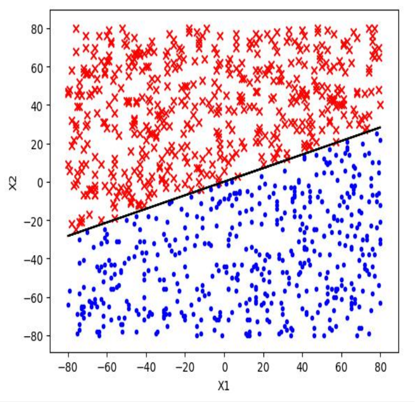


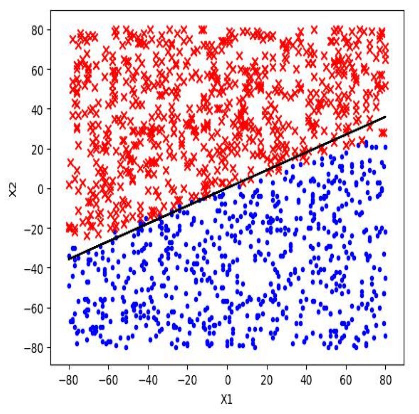
```
45  -37  +
-39  -47  +
78  -11  +
3  -16  +
-31  -22  +
30  -28  +
79  -67  +
-64  -41  +
-61  -45  +
71  -76  +
-72  -43  +
-35  -71  +
-13  -30  +
73  -57  +
27  7  +
19  -56  +
12  -34  +
67  12  +
```

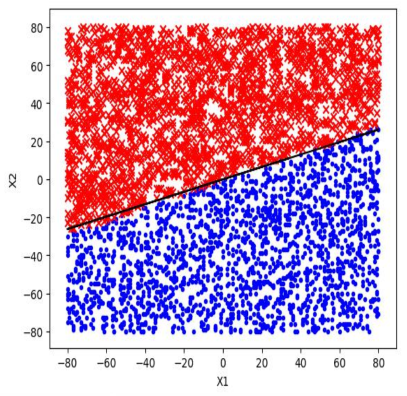
(4)Output Graph:

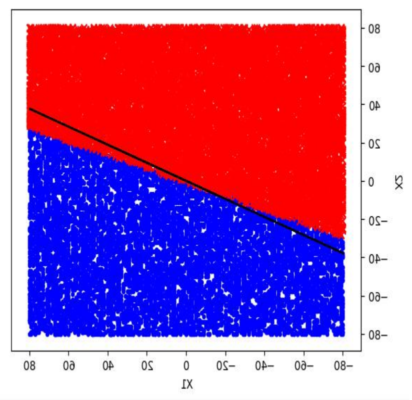












3. Analysis

3.1The size of training set:

(1) too small:

A training set which is too small may result in inaccurate results.For example, in text3, when there are only two pairs of inputs, there are countless straight lines that can match these two points. Therefore, the result is meanless.

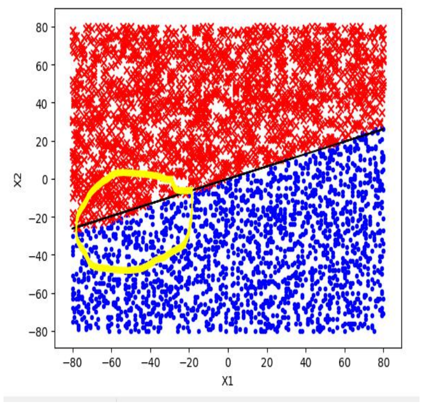
That is to say, when the training set is too small, a correct(or useful in practice) classifier cannot be obtained.

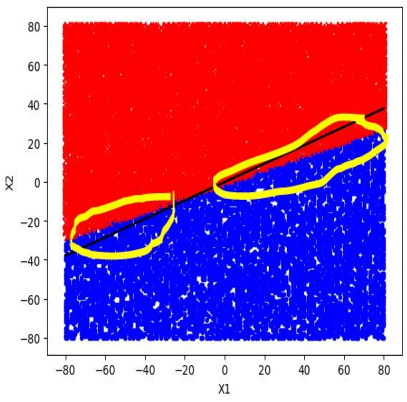
(2) Too large:

From the experimental results we can see, a too large training set can leads to two mainly problems.

First, efficiency. In Experiment 3(6), although it takes only a few tens of seconds for a computer to process 20000 datas, it is far longer than the time spent to process 4000 datas. More importantly, in real-life applications, not any data has meaning.

Second, through experiments, I found that when the data set is too large, chances are that the result is not ideal.Therefore, in practical applications, it is important to select a data set with appropriate size.





2 Unbalanced training set

From text 2, we can see that when the training set is unbalanced, the results are greatly affected, and the curve does not classify the training set very well.However, in practice, due to sample shortages, noise and other reasons, we are likely to have unbalanced training set.

so through reviewing some papers, I found that facing of unbalanced training set, there are several ways to deal with it:

(1)re-sampling (2)Training set partitioning method

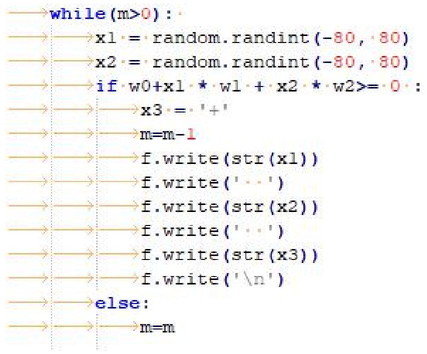
(3)Classifier integration (4)Cost sensitive

and so on.

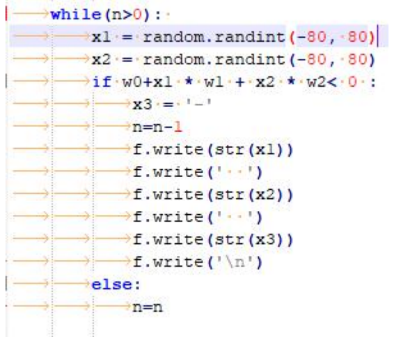
4. code

4.1 DataEmit

generate x1， x2,anf use x3 to store labels.And the function will also generate a train.txt.

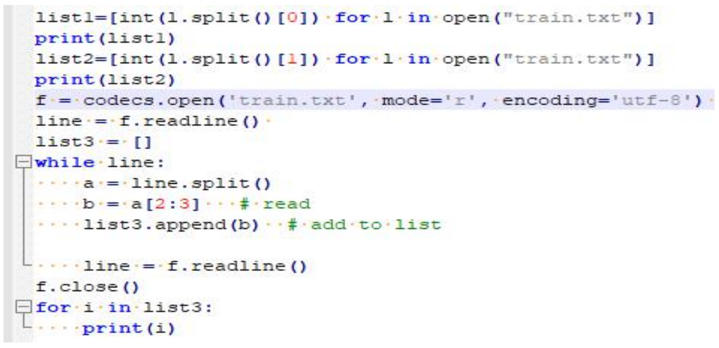


```
    while(m>0):
        x1 = random.randint(-80, 80)
        x2 = random.randint(-80, 80)
        if w0+x1 * w1 + x2 * w2>= 0 :
            x3 = '+'
            m=m-1
            f.write(str(x1))
            f.write('  ')
            f.write(str(x2))
            f.write('  ')
            f.write(str(x3))
            f.write('\n')
        else:
            m=m
```



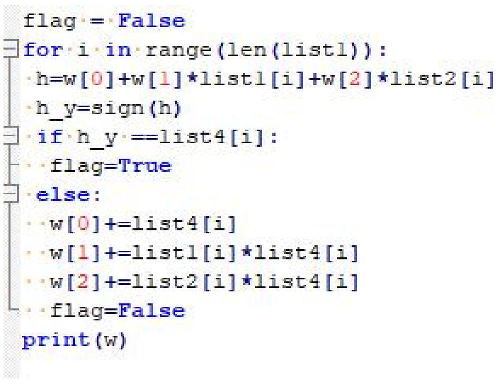
```
    while(n>0):
        x1 = random.randint(-80, 80)
        x2 = random.randint(-80, 80)
        if w0+x1 * w1 + x2 * w2< 0 :
            x3 = '-'
            n=n-1
            f.write(str(x1))
            f.write('  ')
            f.write(str(x2))
            f.write('  ')
            f.write(str(x3))
            f.write('\n')
        else:
            n=n
```

4.2 get data



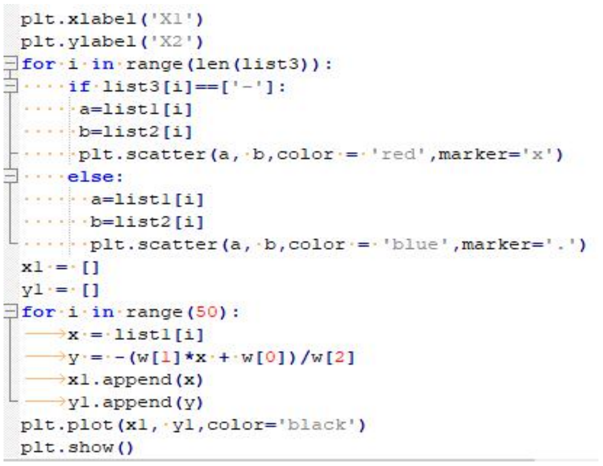
```python
list1=[int(l.split()[0]) for l in open("train.txt")]
print(list1)
list2=[int(l.split()[1]) for l in open("train.txt")]
print(list2)
f = codecs.open('train.txt', mode='r', encoding='utf-8') 
line = f.readline() 
list3 = []
while line:
    a = line.split()
    b = a[2:3]   # read
    list3.append(b)  # add to list

    line = f.readline()
f.close()
for i in list3:
    print(i)
```

3 train



```python
flag = False
for i in range(len(list1)):
 h=w[0]+w[1]*list1[i]+w[2]*list2[i]
 h_y=sign(h)
 if h_y ==list4[i]:
  flag=True
 else:
  w[0]+=list4[i]
  w[1]+=list1[i]*list4[i]
  w[2]+=list2[i]*list4[i]
  flag=False
print(w)
```

4 Visualization



```python
plt.xlabel('X1')
plt.ylabel('X2')
for i in range(len(list3)):
    if list3[i]==['-']:
     a=list1[i]
     b=list2[i]
     plt.scatter(a, b,color = 'red',marker='x')
    else:
      a=list1[i]
      b=list2[i]
      plt.scatter(a, b,color = 'blue',marker='.')
x1 = []
y1 = []
for i in range(50):
	x = list1[i]
	y = -(w[1]*x + w[0])/w[2]
	x1.append(x)
	y1.append(y)
plt.plot(x1, y1,color='black')
plt.show()
```