

# 感知机

## 一、实验名称：感知机

## 二、实验目的

实现二类线性分类的感知机模型，基于损失函数，利用梯度下降法对其进行极小化。

## 三、实验原理

➤ 令  $x \in R^d$  表示样本的特征向量,  $y = \{+1, -1\}$  表示样本类别, 如下函数:

$$f(x) = \text{sign}(w^T x + b)$$

称为感知机, 其中  $w \in R^d$  称为权重,  $b$  称为偏置。

➤ 给定  $n$  个训练数据  $(x_i, y_i)$ , 可以通过最小化如下模型训练感知机模型:

$$\min_{w, b} - \sum_{i \in M} (w^T x_i + b) \cdot y_i$$

其中  $M$  是误分类点集合。

在实际模型中, 利用梯度下降法最小化损失函数, 即在数据点误分类时, 利用损失函数对参数的偏导和步长对参数进行变更。

## 四、实验步骤

- 1、生成数据集: 利用 **中心点** 加高斯噪音, 生成两类数据集, 样本量 **n=100**, 维度为 **2 维**, 并为两类加不同标签。
- 2、感知机模型: 初始化系数 **w** 和 **b**, 并设置 **步长 c=0.1**, 构建损失函数 **y\*(wx+b)**。
- 3、梯度下降法迭代: 在迭代 **循环次数 x=1000** 中, 每次循环随机抽取一个数据点, 代入损失函数, 若大于零则进入下一次迭代, 若小于零则按照步长 **c** 对 **w**、**b** 进行如下改变:

$$w = w + cxy$$

$$b = b + cy$$

- 4、结果展示: 在图像中标注数据集、测试集和分类超平面, 并显示 **w**、**b** 以及数据集、测试集的错误率和代码运行时间。
- 5、训练: 改动参数 (如上标红), 进行调试训练, 综合考虑, 获得合适的参数。

## 五、代码

主要的训练部分已标红

**% 数据生成**

**n = 100;**

**% 样本量**

**center1 = [1, 1];**

**% 数据中心**

**center2 = [3, 4];**

**X = zeros(2\*n, 2);**

**% 数据点 (2 维) : 中心点+高斯噪声**

**X(1:n, :) = ones(n, 1)\*center1 + randn(n, 2);**

```

X(n+1:2*n,:) = ones(n,1)*center2 + randn(n,2);
Y = zeros(2*n,1);
% 类别标签（第一类为1，第二类为-1）
Y(1:n) = 1;
Y(n+1:2*n) = -1;

% 图一：两类数据点
figure(1)
set(gcf, 'Position', [1, 1, 700, 600], 'color', 'w')
set(gca, 'FontSize', 18)
plot(X(1:n,1), X(1:n,2), 'ro', 'LineWidth', 1, 'MarkerSize', 10);
% 第一类数据点
hold on;
plot(X(n+1:2*n,1), X(n+1:2*n,2), 'b*', 'LineWidth', 1, 'MarkerSize', 10);
% 第二类数据点
hold on;
xlabel('x axis');
ylabel('y axis');
legend('class 1', 'class 2');

% 感知机模型：y=x*w+b
tic()
w = zeros(2,1);
b = zeros(1);
x = 1000; % 迭代次数
c = 0.1; % 步长

for i = 1:x
    a = randi([1, 2*n]);
    random = X(a,:);
    judge = (random*w+b)*Y(a);
    if judge <= 0
        w = w+(X(a,:))'*c.*Y(a);
        b = b+c*Y(a);
    end
end
toc()

% 图二：分类器可视图（x1 为横轴，y1 为纵轴）
x1 = -2:0.00001:7;
y1 = (-b*ones(1, length(x1))-w(1)*x1)/w(2);

figure(2)

```

```

set(gcf, 'Position', [1, 1, 700, 600], 'color', 'w')
set(gca, 'FontSize', 18)
plot(X(1:n, 1), X(1:n, 2), 'ro', 'LineWidth', 1, 'MarkerSize', 10);
% 第一类数据点
hold on;
plot(X(n+1:2*n, 1), X(n+1:2*n, 2), 'b*', 'LineWidth', 1, 'MarkerSize', 10);
% 第二类数据点
hold on;
plot(x1, y1, 'k', 'LineWidth', 1, 'MarkerSize', 10);
% 分类界面
xlabel('x axis');
ylabel('y axis');
legend('class 1', 'class 2', 'classification surface');

% 测试
m = 10;
Xt = zeros(2*m, 2);
Xt(1:m, :) = ones(m, 1)*center1 + randn(m, 2);
Xt(m+1:2*m, :) = ones(m, 1)*center2 + randn(m, 2);
Yt = zeros(2*m, 1);
Yt(1:m) = 1;
Yt(m+1:2*m) = -1;

% 图三：测试结果
figure(3)
set(gcf, 'Position', [1, 1, 700, 600], 'color', 'w')
set(gca, 'FontSize', 18)
plot(X(1:n, 1), X(1:n, 2), 'ro', 'LineWidth', 1, 'MarkerSize', 10);
% 画第一类数据点
hold on;
plot(X(n+1:2*n, 1), X(n+1:2*n, 2), 'b*', 'LineWidth', 1, 'MarkerSize', 10);
% 画第二类数据点
hold on;
plot(Xt(1:m, 1), Xt(1:m, 2), 'go', 'LineWidth', 1, 'MarkerSize', 10);
% 画第一类测试点
hold on;
plot(Xt(m+1:2*m, 1), Xt(m+1:2*m, 2), 'g*', 'LineWidth', 1, 'MarkerSize', 10);
% 画第二类测试点
hold on;
plot(x1, y1, 'k', 'LineWidth', 1, 'MarkerSize', 10);
% 画分类界面
xlabel('x axis');
xlabel('x axis');
ylabel('y axis');

```

```
legend('class 1: train','class 2: train','class 1: test','class 2:
test','classification surface');
```

% 结果与错误率

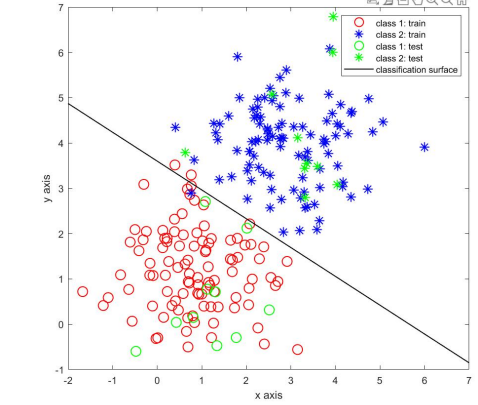
```
disp(w)
disp(b)
```

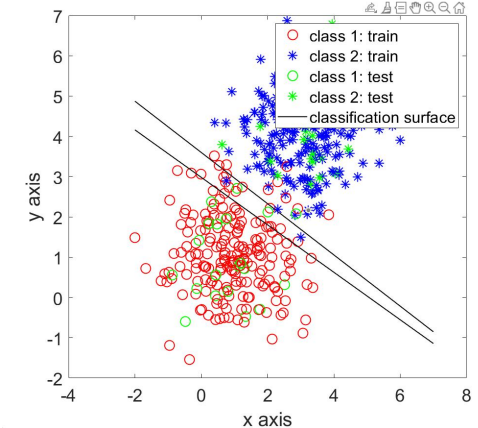
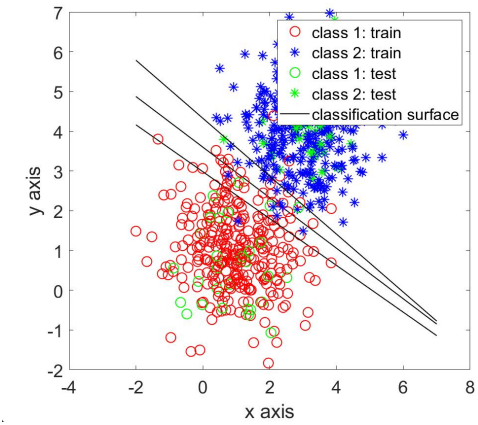
```
wrong = 0;
for i = 1:2*n
    if (X(i,:)*w+b)*Y(i) < 0
        wrong = wrong + 1;
    end
end
wrong = wrong / (2*n);
test_wrong = 0;
for i = 1:2*m
    if (Xt(i,:)*w+b)*Yt(i) < 0
        test_wrong = test_wrong + 1;
    end
end
test_wrong = test_wrong / (2*m);
disp(wrong)
disp(test_wrong)
```

六、调试训练

以下通过修改不同参数，调试模型。改变的参数分别为数据集样本量 n、迭代次数 x、步长 c、数据中心、数据维度。经过调试，源代码中的参数是合理的。

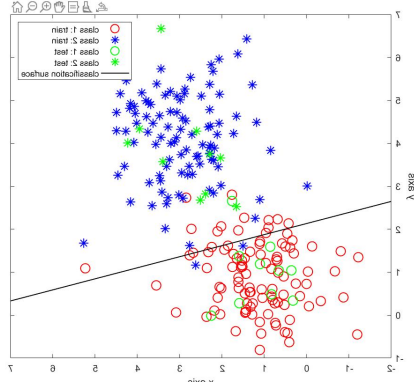
1、n=100, x=1000, c=0.1，连续重复 3 次，展示在一张画布上

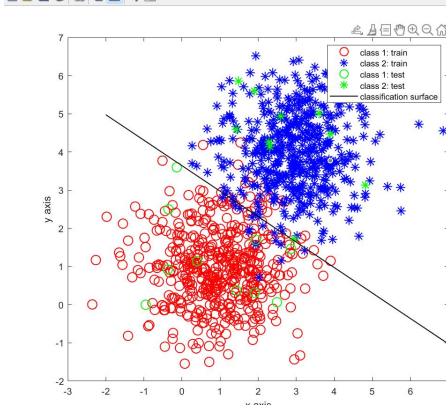
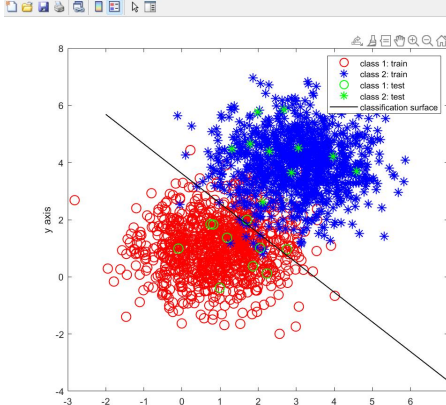
次数	图三（测试）	w	b	数据集 错误率	测试集 错误率	耗时 (ms)
1		-0.3356 -0.5276	1.9 000	0.0150	0	1.02 8

2		-0.4346 -0.7375	2.2 000	0.0500	0.0500	1.11 5
3		-0.3875 -0.5315	2.3 000	0.0450	0	0.71 2

通过连续的测试，在最后一张图上，呈现了三个不一样的分割线，它们都较好地完成了分类任务。引起结果不同的原因可能是多方面的，以下通过对不同参数的实验进行分析。

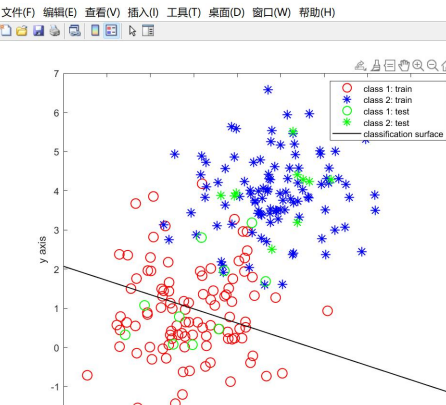
2、 $x=1000$ ， $c=0.1$ ，改动  $n$

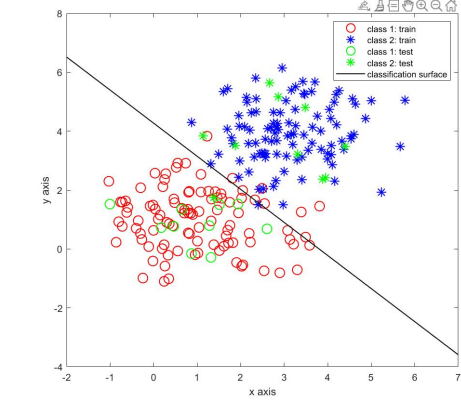
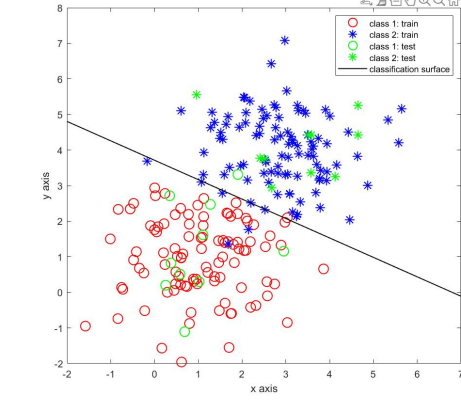
n	图三（测试）	w	b	数据集 错误率	测试集 错误率	耗时 (ms)
100		-0.2415 -0.9379	2.0 000	0.0850	0.0500	0.87 9

500		-0.4033 -0.6044	2.2 000	0.0410	0	1.99 4
1000		-0.6611 -0.6371	2.3 000	0.0785	0.1000	2.37 7

在不断地增大数据集样本量的过程中，运行时间也在增长。可以看出，不同数据集给出的参数是大体相当的，且对测试集的错误率也大体相等，这也表示，针对本题设下的情景，增大数据样本量对提高正确率并无裨益。并且，由于数据量增加导致偏离中心点的数据量变多，可能会积累不确定的误差。当然，虽然这里没有进行测试，但是当数据集样本量过小时，其无法良好地反应数据集的真实情况，也会使得效果不佳。

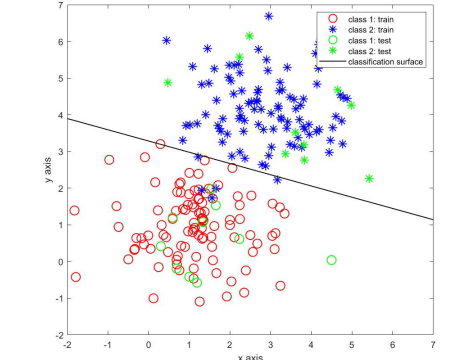
3、 $n=100$ ， $c=0.1$ ，改动  $x$

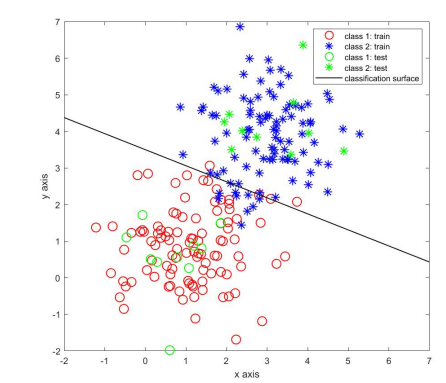
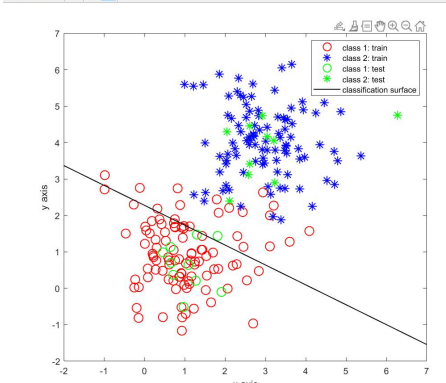
x	图三（测试）	w	b	数据集 错误率	测试集 错误率	耗时 (ms)
100		-0.215 0 -0.593 3	0.8 000	0.2200	0.2000	0.21 5

1000		-0.3554 -0.7296	1.9000	0.0850	0.0500	0.820
10000		-0.5442 -0.9963	3.7000	0.0400	0.0500	5.319

在不断地增大迭代次数的过程中，运行时间也在增长。在迭代中，当分类错误修正参数时，会比分类正确时间长，但这是模型训练过程中的有效时间。由图一可知，迭代一百次时并未能找到适当的参数，无法正确分类，可以认为，有效的训练次数不足。经过多次试验，发现 1000 次迭代已能将本题设下的情景的要求满足，更多的迭代次数基本是浪费时间。相比于数据样本数增加可能带来的误差，增加迭代次数不会造成这方面的损失。

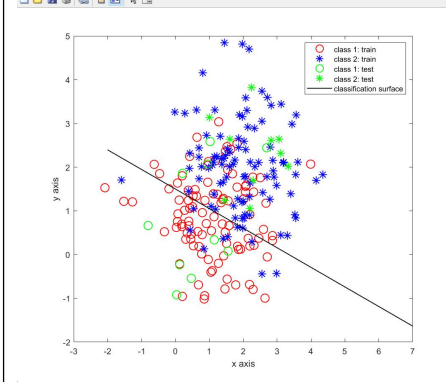
4、 $n=100$ ， $x=1000$ ，改动  $c$

c	图三（测试）	w	b	数据集 错误率	测试集 错误率	耗时 (ms)
1		-1.7729 -5.7853	19	0.0300	0	0.001341

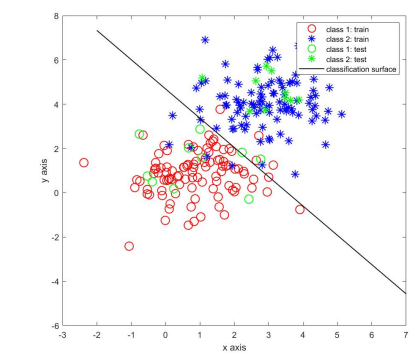
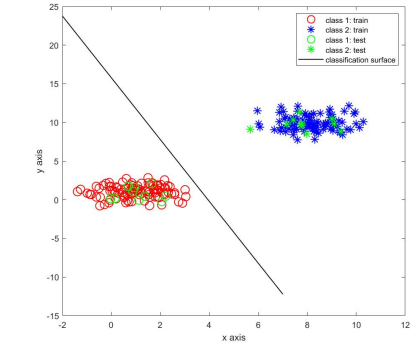
0.1		-0.2882 -0.6572	2.3 000	0.0650	0	0.89 0
0.0 1		-0.0479 -0.0878	0.2 000	0.0950	0.0500	1.19 3

步长与迭代次数是相配对的一对参数，在迭代次数不充足的情况下，步长过小可能导致无法达到正确的结果，如以上图三的结果就相较不理想。而步长过大，会跳过正确的结果，导致在正确结果区间以外反复跳跃，也会造成效果不佳。

##### 5、数据中心 (n=100, x=1000, c=0.1)

数据 中心	图三（测试）	w	b	数据集 错误率	测试集 错误率	耗时 (ms)
1, 1 2, 2		-0.2388 -0.5334	0.8 000	0.2600	0.2000	1.01 3



原题 1, 1 3, 4		-0.4793 -0.3625	1.7 000	0.0500	0.1000	1.05 4
1, 1 8, 10		-0.1014 -0.0253	0.4 000	0	0	1.09 6

当中心点过近时，对步长精细度的要求提高，原设的步长已无法达到较高的精度，错误率大幅上升；而且本题设使用生成数据点的高斯噪声，在距离过近时，会有较大的重合区，这极大降低了准确度。而中心点过远时，正确分类区间过大，有效迭代次数变少，虽然错误率极低，但更难得到最优解。

#### 6、不同维度 (n=100, x=1000, c=0.1)

改变为 100 维，对代码进行修改，因为无法展现图像，所以去掉了可视化部分，改动部分已标红：

```

n = 100;
center1 = rand(1, 100);
center2 = rand(1, 100);
center2 = center2 + 1;
X = zeros(2*n, 100);
X(1:n, :) = ones(n, 1)*center1 + randn(n, 100);
X(n+1:2*n, :) = ones(n, 1)*center2 + randn(n, 100);
Y = zeros(2*n, 1);
Y(1:n) = 1;
Y(n+1:2*n) = -1;

tic()
w = zeros(100, 1);
b = zeros(1);
x = 1000;
c = 0.1;

```

```

for i = 1:x
    a = randi([1,2*n]);
    random = X(a,:);
    judge = (random*w+b)*Y(a);
    if judge <= 0
        w = w+(X(a,:)')*.c.*Y(a);
        b = b+c*Y(a);
    end
end
toc()

m = 10;
Xt = zeros(2*m,100);
Xt(1:m,:) = ones(m,1)*center1 + randn(m,100);
Xt(m+1:2*m,:) = ones(m,1)*center2 + randn(m,100);
Yt = zeros(2*m,1);
Yt(1:m) = 1;
Yt(m+1:2*m) = -1;

disp(w)
disp(b)

wrong = 0;
for i = 1:2*n
    if (X(i,:)*w+b)*Y(i) < 0
        wrong = wrong + 1;
    end
end
wrong = wrong / (2*n);
test_wrong = 0;
for i = 1:2*m
    if (Xt(i,:)*w+b)*Yt(i) < 0
        test_wrong = test_wrong + 1;
    end
end
test_wrong = test_wrong / (2*m);
disp(wrong)
disp(test_wrong)

```

w 有 100 维，这里不做展示。

b=1.8000

数据集错误率=0.0050

测试集错误率=0

可以发现，感知机也很好地完成了高维度地线性分类，得到了一个超平面。

## 七、分析

感知机模型作为利用线性超平面进行分割的模型，训练部分简单，逻辑也很简单直接，对线性可分的数据的分类能力比较好。但同时，感知机也有很多问题，比如

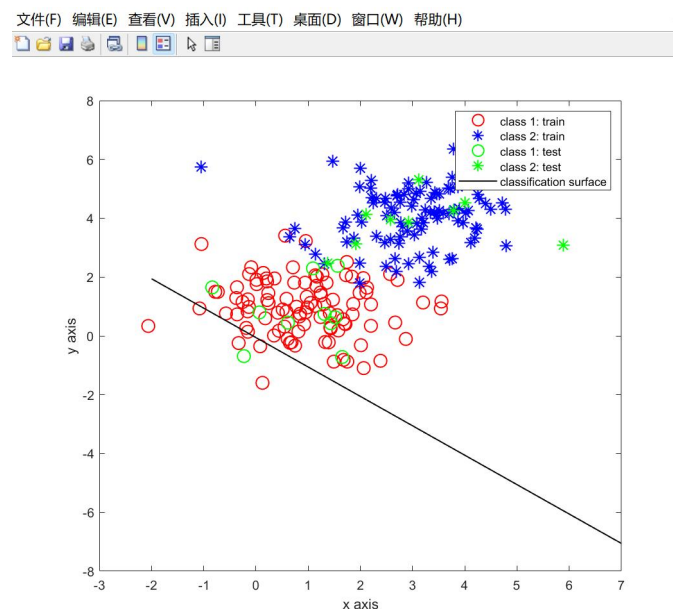
- 1、面对线性不可分的数据时，正确率下降。
- 2、即使处理同一数据集，也会因为随机抽取样本的数据、参数选择、初始点设置等因素获得不同的结果，即只能找到正确解，却无法找到最优解。
- 3、需要人为地根据实际问题修正参数，这需要反复训练。

## 八、附加题

1、见六、3、4、。

3、去掉“if  $y_i(w^T x_i + b) \leq 0$ ”语句，相当于无论数据点分类是否正确，都将参数更新。

以下给出了  $n=100$ ， $x=1000$ ， $c=0.1$  时，去掉该语句的结果：



可以看出，由于在迭代过程中失去了对正确与错误的区分，所得到的结果并不能对数据集进行合理的线性分割。这里可以看作一个监督学习的过程，正确时不进行操作作为一种肯定，而错误时更新参数是纠偏。

4、见六、5、。

5、见六、6、。