感知器算法

## 一、感知器算法

　　　感知机算法是机器学习中的一个二分类监督学习算法，通过一个函数决定由向量代表的一个输入是否属于某一类。它是一种线性分类器：特征通过权重线性组合，然后通过一个线性预测函数来判断。这个算法最早由Frank Rosenblatt在1957年提出。

　　假设输入为x⃗ x→，通过某种确定的非线性变换成一组特征向量Φ(x⃗ )Φ(x→)，特征向量通过一个函数构成广义线性模型：

E:\qq下载\MobileFile\Image\[SHQ2VV3OA9X{E(OF110VJL.png

　　其中，非线性激活函数f(⋅)f(·)是以下形式的阶梯函数：

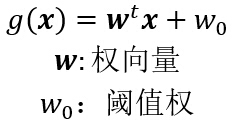
E:\qq下载\MobileFile\Image\H2UZVFG]8}2UGF08{%ML0)B.png

　　一般情况下，Φ(x⃗ )Φ(x→)包含偏分量ϕ0(x⃗ )=1ϕ0(x→)=1。

## 二、算法指导思想

不考虑概率密度函数的确切形式，使用非参数化的方法来求解判别函数。由于线性判别函数具有许多优良的特性，因此这里我们只考虑以下形式的判别函数：它们或者是x的各个分量的线性函数，或者是关于以x为自变量的某些函数的线性函数。在设计感知器之前，需要明确以下几个基本概念：

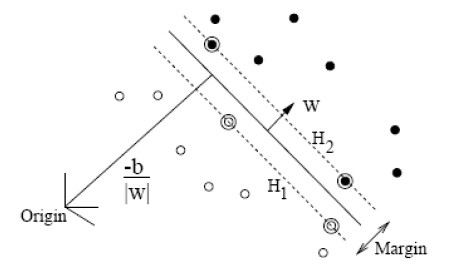
1、判别函数：是指由x的各个分量的线性组合而成的函数：



若样本有c类，则存在c个判别函数，对具有è¿éåå¾çæè¿°形式的判别函数的一个两类线性分类器来说，要求实现以下判定规则：

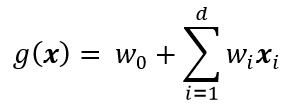


方程g(x)=0定义了一个判定面，它把两个类的点分开来，这个平面被称为超平面，如下图所示。

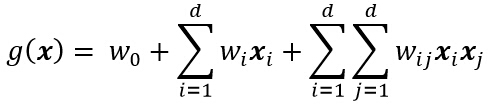


2、广义线性判别函数

线性判别函数g(x)又可写成以下形式：

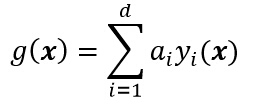


其中系数wi是权向量w的分量。通过加入另外的项（w的各对分量之间的乘积），得到二次判别函数：



因为è¿éåå¾çæè¿°，不失一般性，可以假设è¿éåå¾çæè¿°。这样，二次判别函数拥有更多的系数来产生复杂的分隔面。此时g(x)=0定义的分隔面是一个二阶曲面。

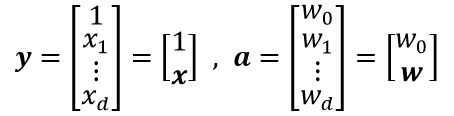
若继续加入更高次的项，就可以得到多项式判别函数，这可看作对某一判别函数g(x)做级数展开，然后取其截尾逼近，此时广义线性判别函数可写成：



或：

è¿éåå¾çæè¿°

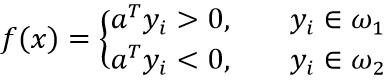
这里y通常被成为“增广特征向量”(augmented feature vector)，类似的，a被称为“增广权向量”，分别可写成：



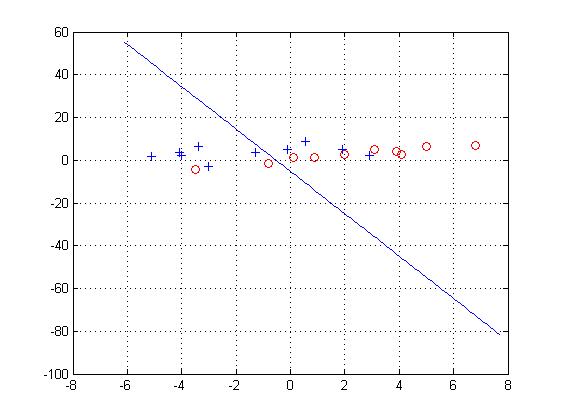
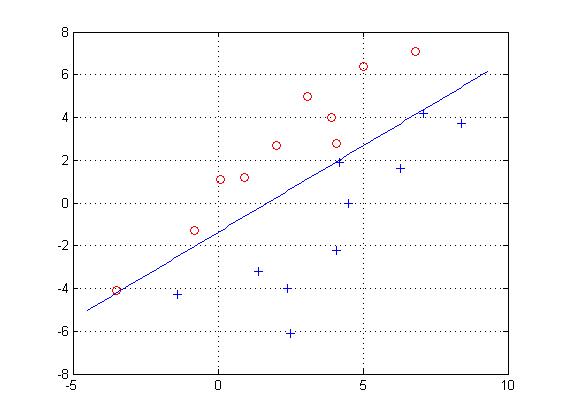
这个从d维x空间到d+1维y空间的映射虽然在数学上几乎没有变化，但十分有用。虽然增加了一个常量，但在x空间上的所有样本间距离在变换后保持不变，得到的y向量都在d维的自空间中，也就是x空间本身。通过这种映射，可以将寻找权向量w和权阈值w0的问题简化为寻找一个简单的权向量a。

3、样本线性可分

即在特征空间中可以用一个或多个线性分界面正确无误地分开若干类样本；对于两类样本点w1和w2，其样本点集合表示为： è¿éåå¾çæè¿°，使用一个判别函数è¿éåå¾çæè¿°来划分w1和w2，需要用这些样本集合来确定判别函数的权向量a，可采用增广样本向量y，即存在合适的增广权向量a，使得：



则称样本是线性可分的。如下图中第一个图就是线性可分，而第二个图则不可分。所有满足条件的权向量称为解向量。



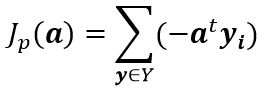
通常对解区限制：引入余量b，要求解向量满足：

è¿éåå¾çæè¿°

余量b的加入在一定程度上可防止优化算法收敛到解区的边界。

## 三、感知器准则函数

这里考虑构造线性不等式è¿éåå¾çæè¿° 的准则函数的问题，令准则函数J(.)为：



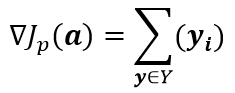
其中Y是被权向量a错分的样本集。当且仅当JP(a\*) = min JP(a) = 0 时，a\*是解向量。这就是感知器（Perceptron）准则函数。

## 四、基本的感知器设计

### 1、感知器准则函数的最小化可以使用梯度下降迭代算法求解：

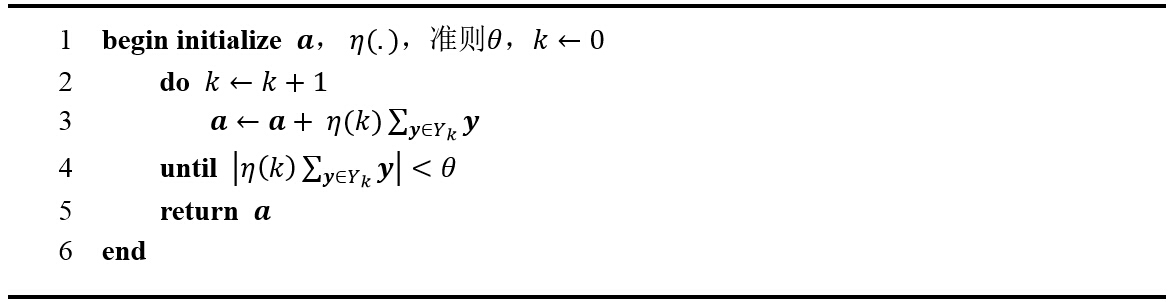
è¿éåå¾çæè¿°

其中，k为迭代次数，η为调整的步长。即下一次迭代的权向量是把当前时刻的权向量向目标函数的负梯度方向调整一个修正量。

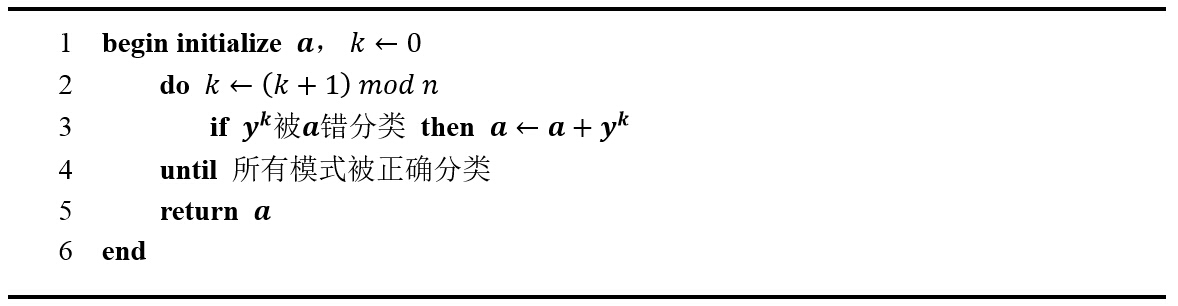


即在每一步迭代时把错分的样本按照某个系数叠加到权向量上。这样就得到了感知算法。

### 2.批处理感知器算法



### 3.固定增量感知器算法



通常情况，一次将所有错误样本进行修正不是效率最高的做法，更常用是每次只修正一个样本或一批样本的固定增量法：

## **五、收敛性分析**

只要训练样本集是线性可分的，对于任意的初值 a(1) ，经过有限次迭代运算，算法必定收敛。而当样本线性不可分时，感知器算法无法收敛。

## 六、优缺点

优点：有效、线性可分下有保证

缺点：仅在线性可分下有效，非线性需要pocket算法

感知器是最简单可以“学习”的机器，是解决线性可分的最基本方法。也是很多复杂算法的基础。感知器的算法的推广有很多种，如带裕量的变增量感知器、批处理裕量松弛算法、单样本裕量松弛算法等等。

## 七、数据集描述

w1 = [ 0.1 1.1;...

6.8 7.1;...

-3.5 -4.1;...

2.0 2.7;...

4.1 2.8;...

3.1 5.0;...

-0.8 -1.3;...

0.9 1.2;...

5.0 6.4;...

3.9 4.0];

w2 = [ 7.1 4.2;...

-1.4 -4.3;...

4.5 0.0;...

6.3 1.6;...

4.2 1.9;...

1.4 -3.2;...

2.4 -4.0;...

2.5 -6.1;...

8.4 3.7;...

4.1 -2.2];

w3 = [-3.0 -2.9;...

0.54 8.7;...

2.9 2.1;...

-0.1 5.2;...

-4.0 2.2;...

-1.3 3.7;...

-3.4 6.2;...

-4.1 3.4;...

-5.1 1.6;...

1.9 5.1];

w4 = [-2.0 -8.4;...

-8.9 0.2;...

-4.2 -7.7;...

-8.5 -3.2;...

-6.7 -4.0;...

-0.5 -9.2;...

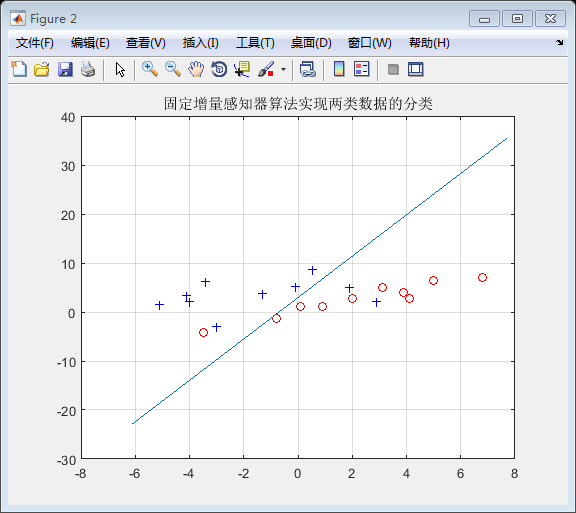
-5.3 -6.7;...

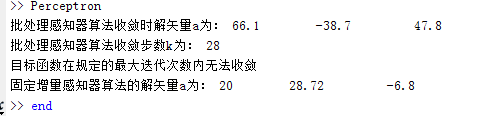
-8.7 -6.4;...

-7.1 -9.7;...

-8.0 -6.3];

## 八、实验结果





# Perceptron代码

**BatchPerceptron.m**

% 批处理感器算法

function BatchPerceptron(w1, w2)

figure;

plot(w1(:,1),w1(:,2),'ro');

hold on;

grid on;

plot(w2(:,1),w2(:,2),'b+');

% 对所有训练样本求增广特征向量y

one = ones(10,1);

y1 = [one w1];

y2 = [one w2];

w12 = [y1; -y2]; % 增广样本规范化

y = zeros(size(w12,1),1); % 错分样本集y初始为零矩阵

% 初始化参数

a = [0 0 0]; % [0 0 0];

Eta = 1;

time = 0; % 收敛步数

while any(y<=0)

for i=1:size(y,1)

y(i) = a \* w12(i,:)';

end;

a = a + sum(w12(find(y<=0),:));%修正向量a

time = time + 1;%收敛步数

if (time >= 300)

break;

end

end;

if (time >= 300)

disp('目标函数在规定的最大迭代次数内无法收敛');

disp(['批处理感知器算法的解矢量a为： ',num2str(a)]);

else

disp(['批处理感知器算法收敛时解矢量a为： ',num2str(a)]);

disp(['批处理感知器算法收敛步数k为： ',num2str(time)]);

end

%找到样本在坐标中的集中区域，以便于打印样本坐标图

xmin = min(min(w1(:,1)),min(w2(:,1)));

xmax = max(max(w1(:,1)),max(w2(:,1)));

xindex = xmin-1:(xmax-xmin)/100:xmax+1;

yindex = -a(2)\*xindex/a(3)-a(1)/a(3);

plot(xindex,yindex);

title('批处理感知器算法实现两类数据的分类');

end

**FixedIncrementPerceptron.m**

% 固定增量感知器算法

function FixedIncrementPerceptron(w1, w2)

[n, d] = size(w1);

figure;

plot(w1(:,1),w1(:,2),'ro');

hold on;

grid on;

plot(w2(:,1),w2(:,2),'b+');

% 对所有训练样本求增广特征向量y

one = ones(10,1);

y1 = [one w1];

y2 = [one w2];

w12 = [y1; -y2]; % 增广样本规范化

y = zeros(size(w12,1),1); % 错分样本集y初始为零矩阵

% 初始化参数

a = [0 0 0];

Eta = 1;

% k = 0;

time = 0; % 收敛的步数

yk = zeros(10,3);

y = a \* w12';

while sum(y<=0)>0

% for i=1:size(y,1)

% y(i) = a \* w12(i,:)';

% end;

y = a \* w12';

rej=[];

for i=1:2\*n %这个循环计算a(K+1) = a(k) + sum {yj被错误分类} y(j)

if y(i)<=0

a = a + w12(i,:);

rej = [rej i];

end

end

% fprintf('after iter %d, a = %g, %g\n', time, a);

% rej

time = time + 1;

if ((size(rej) == 0) | (time >= 300))

break;

end

end;

if (time >= 300)

disp('目标函数在规定的最大迭代次数内无法收敛');

disp(['固定增量感知器算法的解矢量a为： ',num2str(a)]);

else

disp(['固定增量感知器算法收敛时解矢量a为： ',num2str(a)]);

disp(['固定增量感知器算法收敛步数kt为： ',num2str(time)]);

end

%找到样本在坐标中的集中区域，以便于打印样本坐标图

xmin = min(min(w1(:,1)),min(w2(:,1)));

xmax = max(max(w1(:,1)),max(w2(:,1)));

xindex = xmin-1:(xmax-xmin)/100:xmax+1;

% yindex = -a(2)\*xindex/a(3)-a(1)/a(3);

yindex = -a(2)\*xindex/a(3) - a(1)/a(3);

plot(xindex,yindex);

title('固定增量感知器算法实现两类数据的分类');

end

**Perceptron.m**

close all;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% 感知器实验

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

w1 = [ 0.1 1.1;...

6.8 7.1;...

-3.5 -4.1;...

2.0 2.7;...

4.1 2.8;...

3.1 5.0;...

-0.8 -1.3;...

0.9 1.2;...

5.0 6.4;...

3.9 4.0];

w2 = [ 7.1 4.2;...

-1.4 -4.3;...

4.5 0.0;...

6.3 1.6;...

4.2 1.9;...

1.4 -3.2;...

2.4 -4.0;...

2.5 -6.1;...

8.4 3.7;...

4.1 -2.2];

w3 = [-3.0 -2.9;...

0.54 8.7;...

2.9 2.1;...

-0.1 5.2;...

-4.0 2.2;...

-1.3 3.7;...

-3.4 6.2;...

-4.1 3.4;...

-5.1 1.6;...

1.9 5.1];

w4 = [-2.0 -8.4;...

-8.9 0.2;...

-4.2 -7.7;...

-8.5 -3.2;...

-6.7 -4.0;...

-0.5 -9.2;...

-5.3 -6.7;...

-8.7 -6.4;...

-7.1 -9.7;...

-8.0 -6.3];

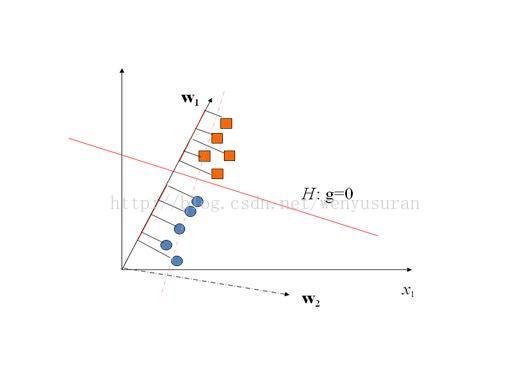
BatchPerceptron(w1, w2);

FixedIncrementPerceptron(w1, w3);

Fisher分类器

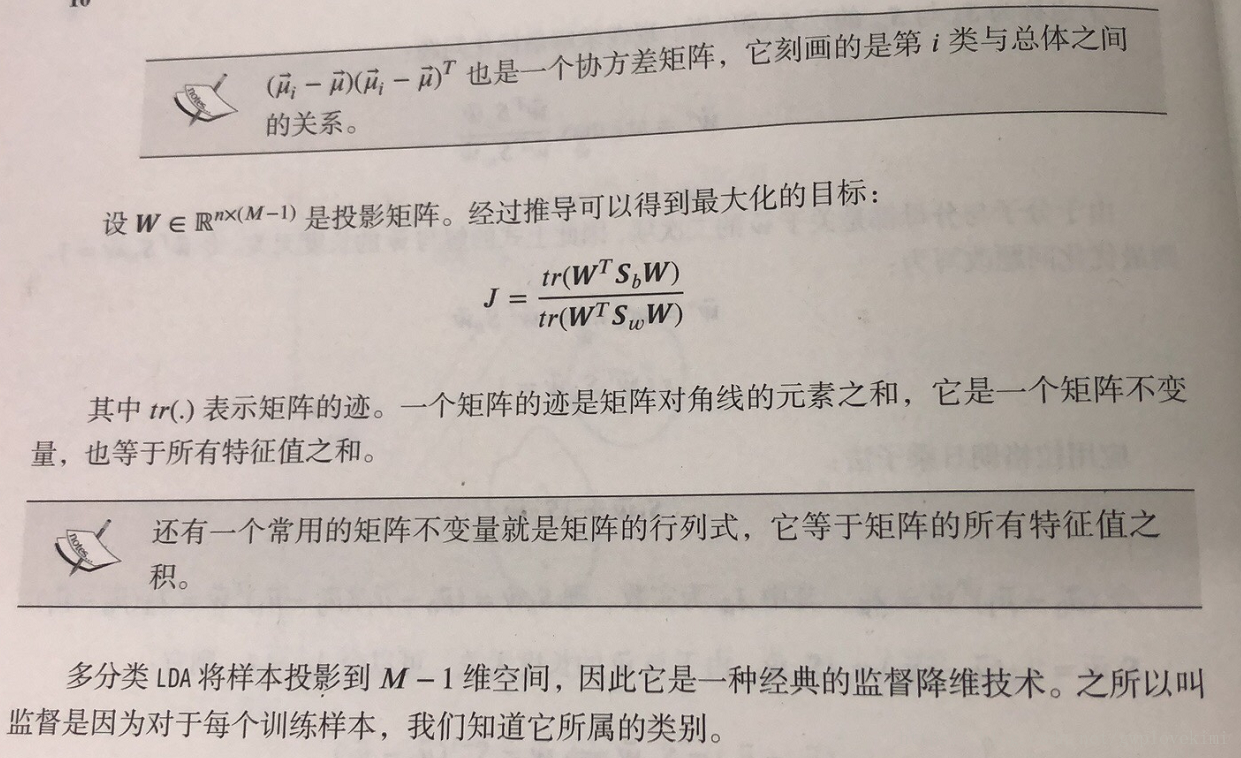
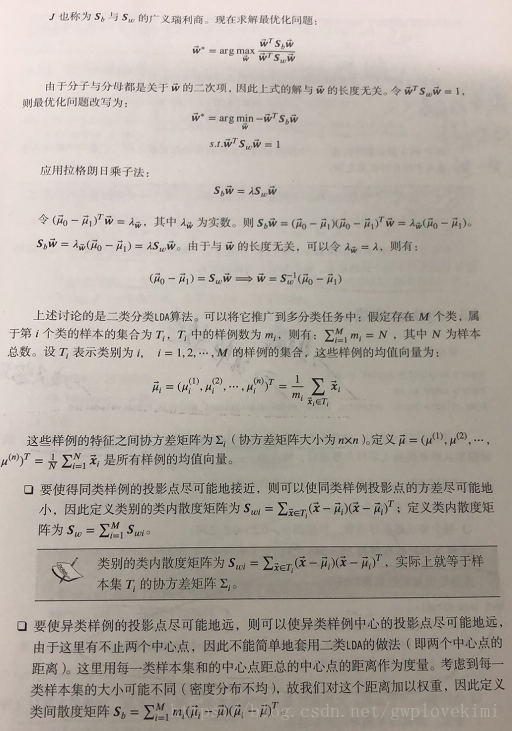
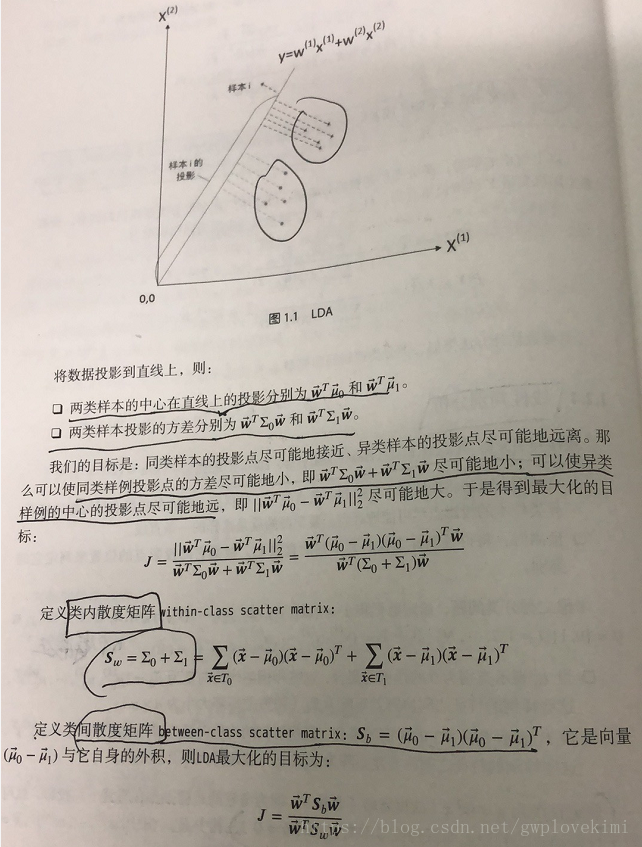
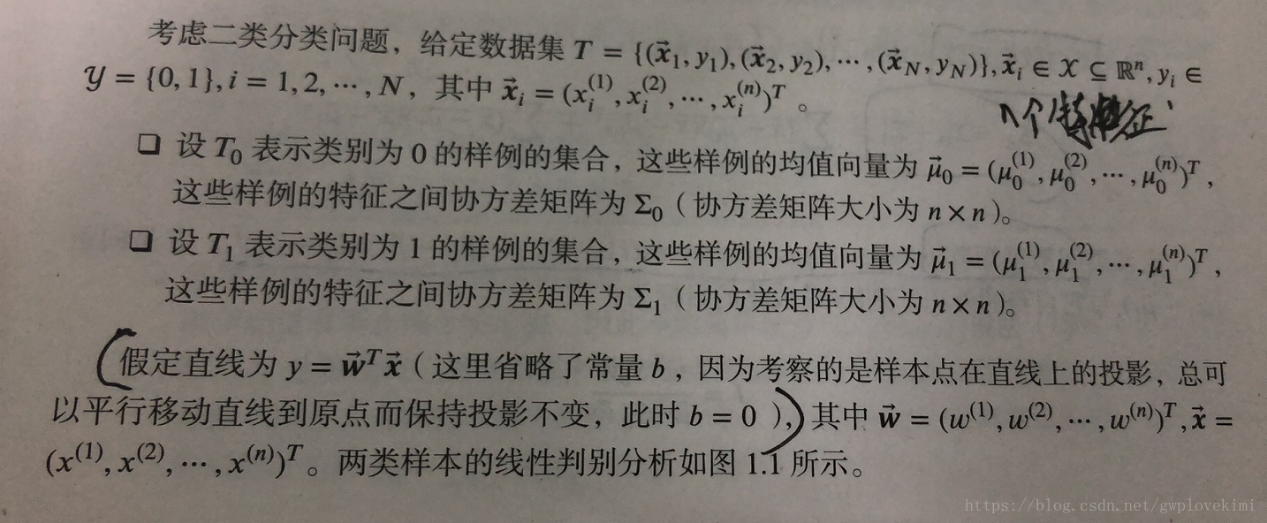
# 一、Fisher的基本思想是：

Fisher准则基本原理：找到一个最合适的投影轴，使两类样本在该轴上投影之间的距离尽可能远，而每一类样本的投影尽可能紧凑，从而使分类效果为最佳。



例如上图中：通过将方块点和圆点向w1投影，然后再在设置合适的阈值即可将方块和圆点分离。

# 二、推导过程



# 三、优缺点

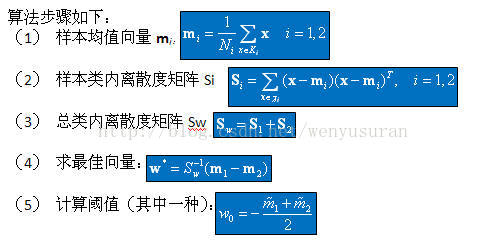
## 优点：

1. 一般对于线性可分的样本，总能找到一个投影方向；降维仍然线性可分。
2. 可以直接求解法向量
3. 适用随机模型，多类情况

## 缺点：

1. 如果M1=M2，W=0。则这样的样本线性不可分；M1！=M2，未必线性可分；SW不可逆，未必不可分。
2. 对线性不可分的情况，无法确定其分类。

# 四、算法步骤



# 五、数据集描述

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%分别产生x轴和y轴都为正态分布的随机序列

%假设x轴和y轴序列相互独立，可产生二维正态分布随机序列

%w1、w2分别用来保存两个训练集的横坐标和纵坐标

%用normrnd函数产生正态分布函数

%normrnd（mean,omega,[row,column]）

%mean:均值;omega:标准差

%row:产生随机序列的行数;column:产生随机序列的列数

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

X1 = normrnd(40,10,[200,1]);

Y1 = normrnd(40,10,[200,1]);

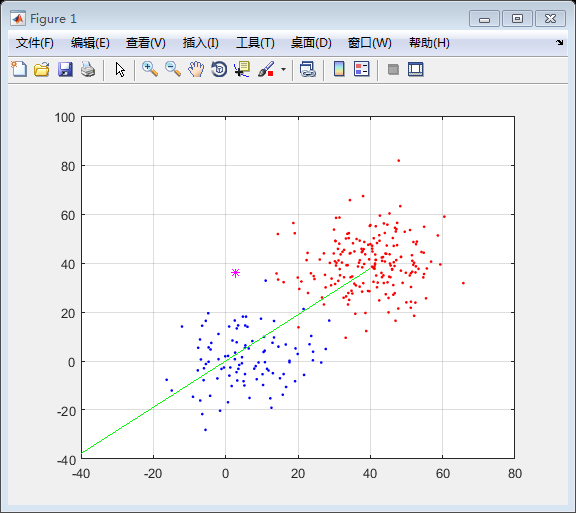
w1=[X1, Y1];

X2 = normrnd(5 ,10,[100,1]);

Y2 = normrnd(0 ,10,[100,1]);

w2=[X2, Y2];

# 六、实验结果



代码

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%本m文件实现fisher算法，并对两个二维正态分布随机序列

%进行训练，进而可在屏幕上任意取点，程序可输出属于第一类

%还是第二类

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%分别产生x轴和y轴都为正态分布的随机序列

%假设x轴和y轴序列相互独立，可产生二维正态分布随机序列

%w1、w2分别用来保存两个训练集的横坐标和纵坐标

%用normrnd函数产生正态分布函数

%normrnd（mean,omega,[row,column]）

%mean:均值;omega:标准差

%row:产生随机序列的行数;column:产生随机序列的列数

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

X1 = normrnd(40,10,[200,1]);

Y1 = normrnd(40,10,[200,1]);

w1=[X1, Y1];

X2 = normrnd(5 ,10,[100,1]);

Y2 = normrnd(0 ,10,[100,1]);

w2=[X2, Y2];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%以下部分为fisher算法的实现

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%计算样本均值

m1=mean(w1)';

m2=mean(w2)';

%s1、s2分别代表表示第一类、第二类样本的类内离散度矩阵

s1=zeros(2);

[row1,colum1]=size(w1);

for i=1:row1

s1 = s1 + (w1(i,:)'-m1)\*(w1(i,:)'-m1)';

end;

s2=zeros(2);

[row2,colum2]=size(w2);

for i=1:row2

s2 = s2 + (w2(i,:)' - m2)\*(w2(i,:)' - m2)';

end;

%计算总类内离散度矩阵Sw

Sw=s1+s2;

%计算fisher准则函数取极大值时的解w

w=inv(Sw)\*(m1-m2);

%计算阈值w0

ave\_m1 = w'\*m1;

ave\_m2 = w'\*m2;

w0 = (ave\_m1+ave\_m2)/2;

%画出两类训练样本点

figure(1)

plot(X1,Y1,'.r',X2,Y2,'.b');%画出两类样本点

hold on;grid;

%画出取极大值时的解w

x = [-40:0.1:40];

y = x\*w(2)/w(1);

plot(x,y,'g')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%以下为测试部分

%利用ginput随机选取屏幕上的点（可连续取10个点）

%程序可根据点的位置自动地显示出属于那个类

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for i=1:10

[x,y]=ginput(1);

plot(x,y,'m\*');

sample=[x,y];

holdall

if(sample\*w- w0>0)

disp('it belong to the first class');

else

disp('it belong to the second class');

end;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

对比

1. Fisher分类器和感知器算法都是解决线性可分问题
2. Fisher分类器比感知器算法效率高，覆盖范围广
3. 优缺点如上所述，非常类似