北京航空航天大學
BEIHANG UNIVERSITY

线性分类的感知器算法

院（系）名称	自动化科学与电气工程学院
专 业 名 称	模式识别与智能系统
学 生 学 号	15031184
学 生 姓 名	李思奇

2018 年 5 月 2 日

1. 感知器算法的基本原理

1.1 理论原理

设样本特性向量为 d 维随机向量$\mathbf{x}=[x_1,x_2,...,x_d]^T$，状态空间$\Omega$由 2 个可能的状态组成：$\Omega=\{\omega_1,\omega_2\}$。

为讨论方便，将向量$\vec{x}$增加一维，其值取常数 1，即有：

$$\mathbf{x}=[1,x_1,x_2,...,x_d]^T$$

定义权向量为$\boldsymbol{\theta}=\left[\theta_0,\theta_1,..,\theta_d\right]^T$，则对应线性判别函数为$g(\mathbf{x})=\boldsymbol{\theta}^T\mathbf{x}$，决策规则为$x\in\begin{cases}\omega_1,g(\mathbf{x})>0\\\omega_2,g(\mathbf{x})<0\end{cases}$。

对于一组样本$\mathbf{x}^{(1)},\mathbf{x}^{(2)},...,\mathbf{x}^{(n)}$，定义一个新的变量$\mathbf{x}'$，使得对于第一类的样本，有$\mathbf{x}'^{(i)}=\mathbf{x}^{(i)}$，对于第二类的样本则有$\mathbf{x}'^{(i)}=-\mathbf{x}^{(i)}$，即：

$$\mathbf{x}'^{(i)}=\begin{cases}\mathbf{x}^{(i)},\mathbf{x}^{(i)}\in\omega_1\\-\mathbf{x}^{(i)},\mathbf{x}^{(i)}\in\omega_2\end{cases}$$

这样的$\mathbf{x}'$称为规范化增广样本向量。下面的讨论全部基于增广样本向量，并且仍将$\mathbf{x}'$记为$\mathbf{x}$。

对于线性可分样本集，我们的目标是找到一个权向量$\boldsymbol{\theta}=\left[\theta_0,\theta_1,..,\theta_d\right]^T$，使得$\boldsymbol{\theta}^T\mathbf{x}^{(i)}>0$对于任意$i=1,2,...,n$成立。

对于权向量$\boldsymbol{\theta}$，若某个样本$\mathbf{x}^{(i)}$被错分，则$\boldsymbol{\theta}^T\mathbf{x}^{(i)}<0$。定义感知器准则函数为：

$$J_P(\boldsymbol{\theta})=\sum_{\boldsymbol{\theta}^T\mathbf{x}^{(i)}\le 0}(-\boldsymbol{\theta}^T\mathbf{x}^{(i)}) \qquad (1\text{-}1)$$

感知器准则函数式(1-1)可以根据梯度下降法迭代求解：

$$\boldsymbol{\theta}(t+1)=\boldsymbol{\theta}(t)-\alpha\nabla J_P(\boldsymbol{\theta})=\boldsymbol{\theta}(t)+\alpha\sum_{\boldsymbol{\theta}^T\mathbf{x}^{(i)}\le 0}\mathbf{x}^{(i)} \qquad (1\text{-}2)$$

即在每一步迭代的时候，把错分的样本按照某个系数加到权向量上。式中，α为修正步长。通常情况下，为了提高效率，采用一次修正一个样本的算法。

1.2 算法步骤

```
Begin
    Inputs: x=[1,x1,x2,…,xd],theta=[0,0,…,0]; //给定增广样本向量及权向量的初值
    alpha=1; //设定修正步长
    i=0;j=0;count=0; //计数用变量
    for i=1:10000 //上限 10000 次迭代
        if( theta' * x(:,j) <= 0 ) //用第 j 个样本训练，若分类错误则修正权向量
            theta = theta + alpha * x(:,j);
            count = 0;
        else    count = count + 1; //计训练成功样本数加 1
        end
        j=j+1; //选择下一个样本
        if(j == (n+1))     j = 1; //训练完最后一个样本后，重新选择第一个样本
        if(count ==n)     break; //全部样本训练成功则退出迭代
        end
    end
End
Output: theta;
```

2. 基础实验

2.1 基础感知器分类算法的建立

从最简单的情况开始考虑，构建一个二维特征、样本量为 n=400 的数据集，其中正例与反例各 200 例。利用 MATLAB 做出数据集中样本分布如图 2.1.1 所示。

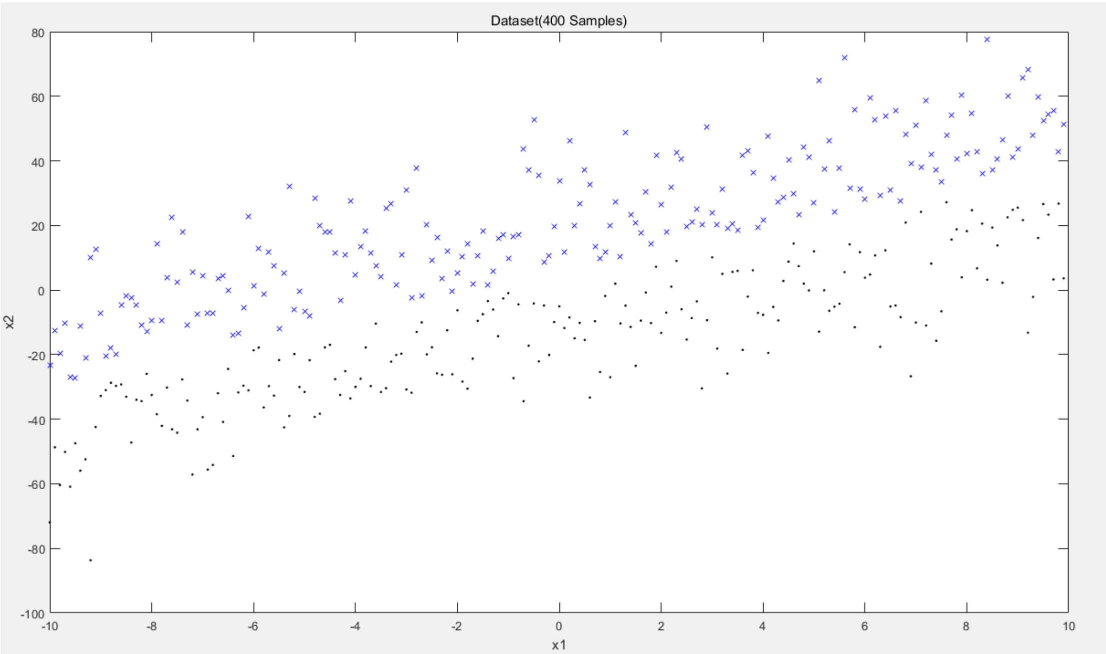

图 2.1.1 数据集样本的分布

取权向量初值为$\boldsymbol{\theta}=[\theta_0,\theta_1,\theta_2]^T=[0,0,0]^T$，修正步长$\alpha=1$进行迭代。

运行程序可得到：经过 i=19532 次迭代后，得到最终权向量$\boldsymbol{\theta}$对于所有样本全部分类正确。做出分类面$g(\mathbf{x})=\boldsymbol{\theta}^T\mathbf{x}=0$如图 2.1.2 所示。

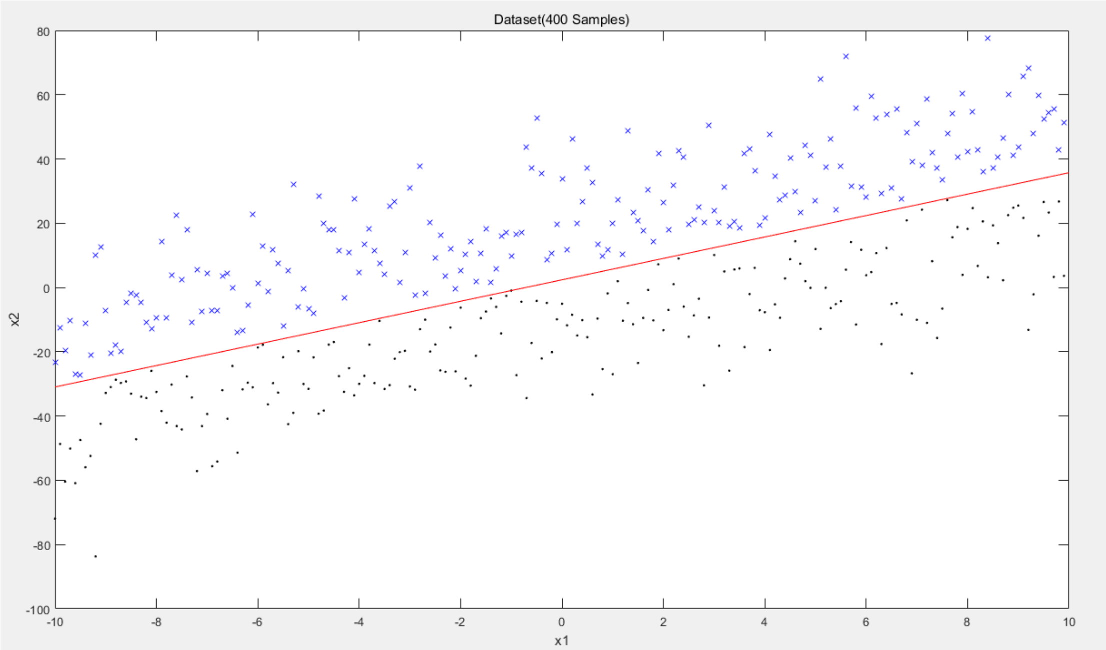

图 2.1.2 n=400 时的分类结果

由图可以看出，该分类判别函数将两类数据完全分开。

2.2 改变样本数目再次实验

取样本数为 n=200 和 n=800、权向量初值为$\boldsymbol{\theta}=[\theta_0,\theta_1,\theta_2]^T=[0,0,0]^T$，修正步长$\alpha=1$为例。

样本数 n=200 时，经过 i=3971 次迭代后将所有样本分类正确。分类结果如图 2.2.1 所示。

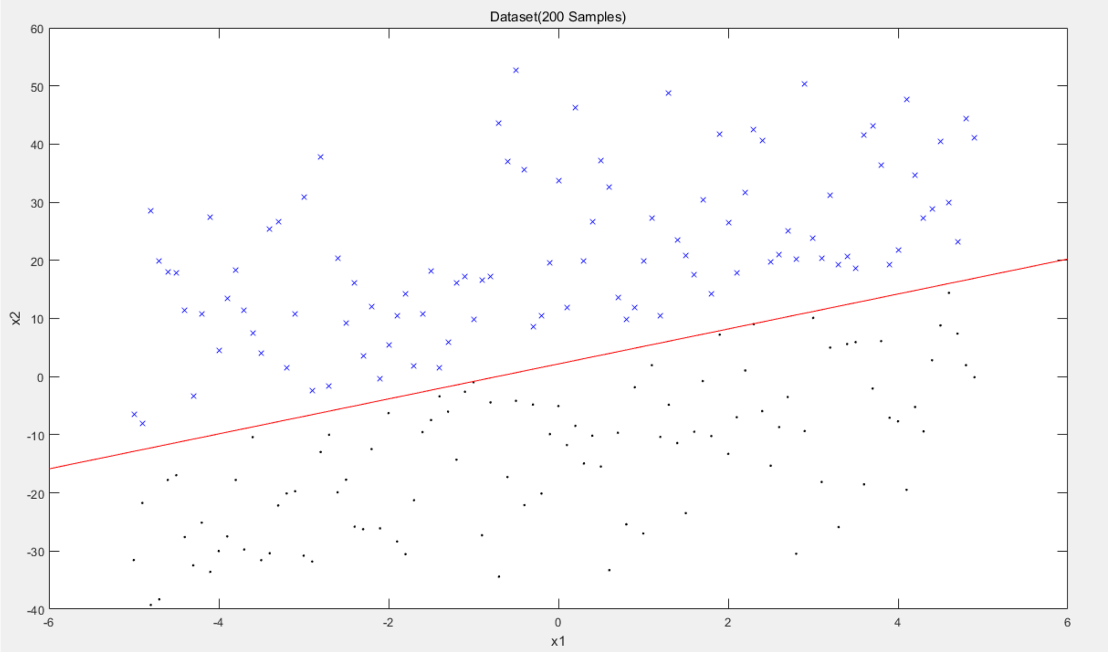

图 2.2.1 n=200 时的分类结果

样本数 n=800 时，经过 i=110641 次迭代后将所有样本分类正确。分类结果如图 2.2.2 所示。

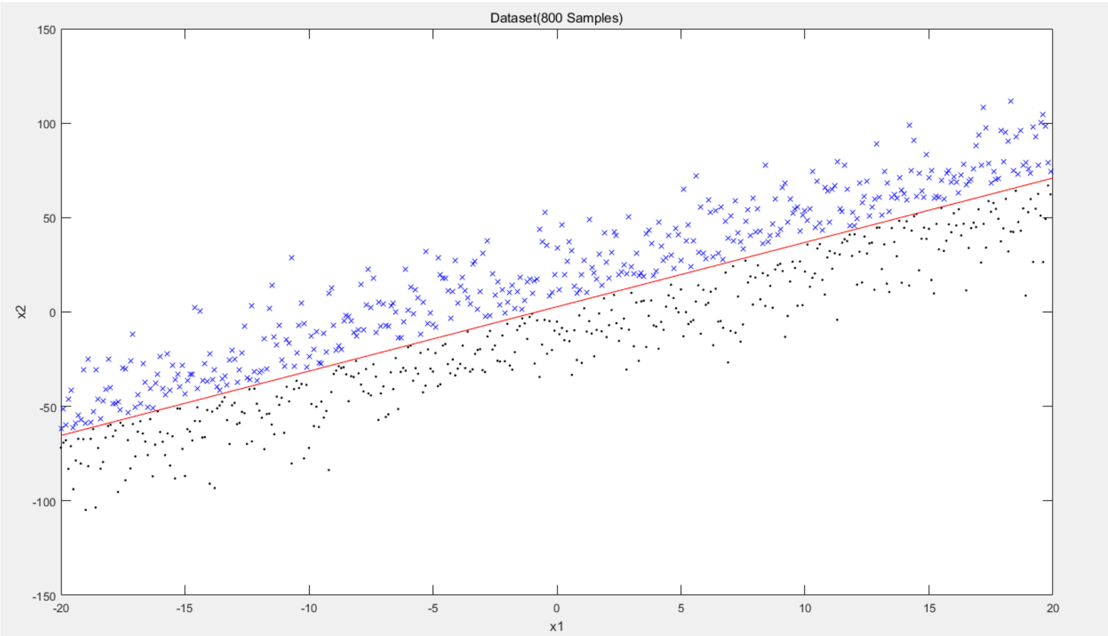

图 2.2.2 n=800 时的分类结果

同理样本数 n 取其他值的时候迭代次数如表 2.2.1 所示。

表 2.2.1 样本数 n 不同取值时的迭代次数

n	100	200	300	400	500	600	700	800
i	1867	3971	4850	19532	26400	106391	113998	110641

由此可见，大体上随着样本数 n 的增大，迭代次数 i 增大。在 n 从 500 变到 600 的过程中，迭代次数 i 有大幅增长。

2.3 增大类间距进行实验

在 2.1 中样本基础上，增大类间距，样本集如图 2.3.1 所示。

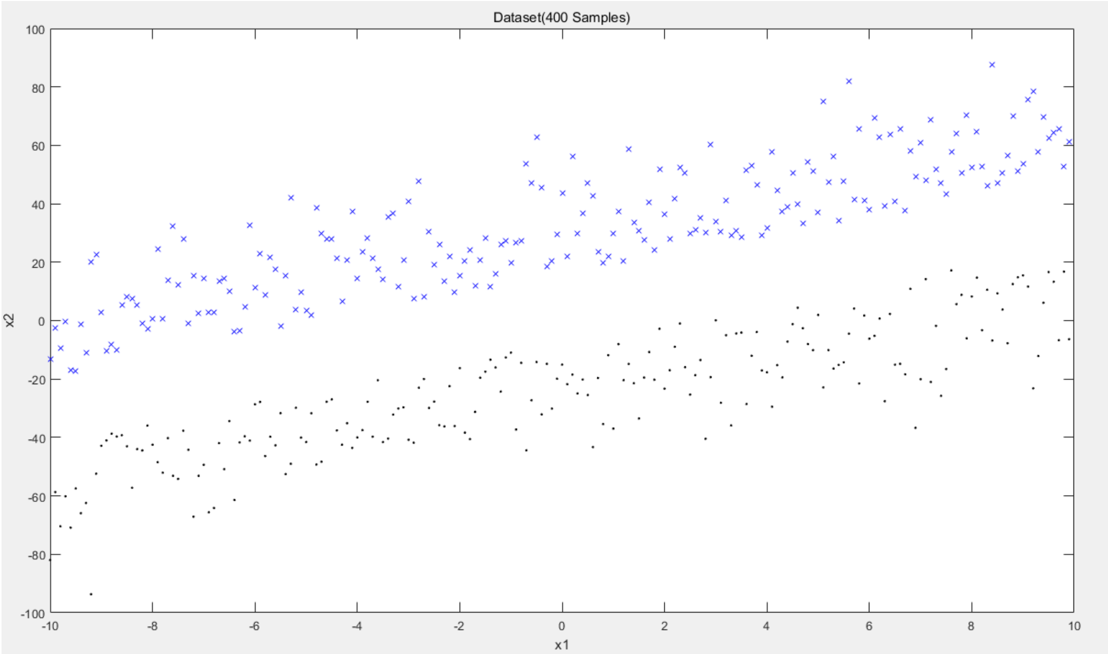

图 2.3.1 增大类间距后的样本集(n=400)

再次利用感知器算法进行分类。取权向量初值为$\boldsymbol{\theta}=[\theta_0,\theta_1,\theta_2]^T=[0,0,0]^T$，修正步长$\alpha=1$进行迭代。

运行程序可得到：经过 i=410 次迭代后，得到最终权向量$\boldsymbol{\theta}$对于所有样本全部分类正确。做出分类面$g(\mathbf{x})=\boldsymbol{\theta}^T\mathbf{x}=0$如图 2.3.2 所示。

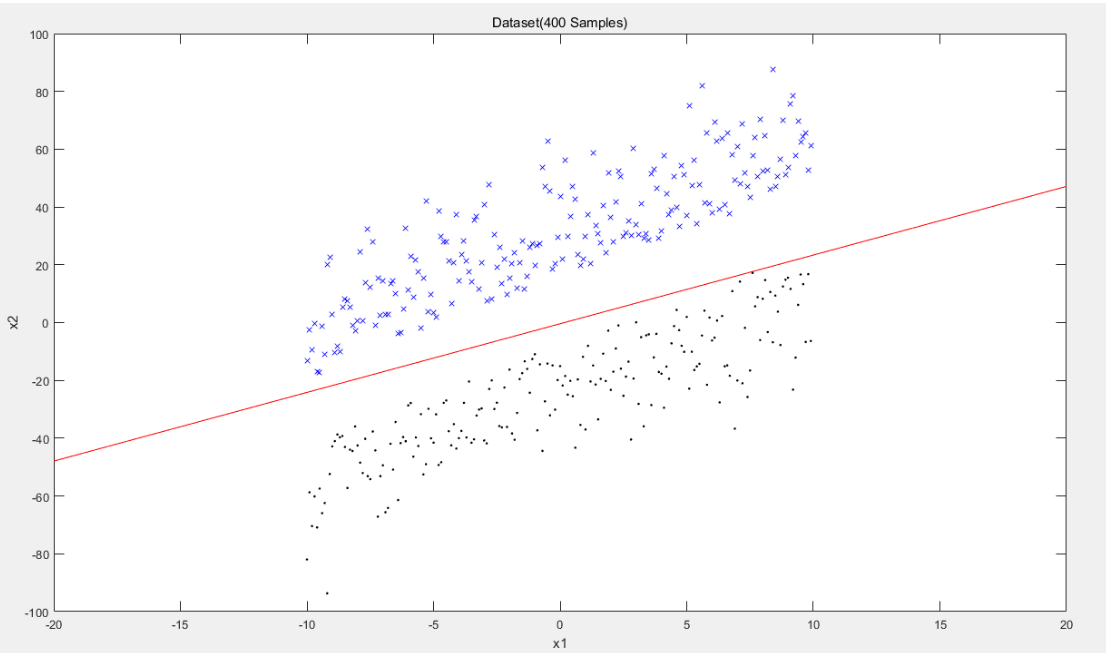

图 2.3.1 增大类间距后的分类结果

可以看出，增大类间距之后，迭代次数明显减小。这符合常识，即类间距越大，分类越容易。

2.4 讨论修正步长α的影响

2.4.1 权向量初值为$\boldsymbol{\theta}=\left[\theta_0,\theta_1,\theta_2\right]^T=\left[0,0,0\right]^T$的情况

以样本数 n=400 为例。假设权向量初值为$\boldsymbol{\theta}=\left[\theta_0,\theta_1,\theta_2\right]^T=\left[0,0,0\right]^T$。从 0.001 到 1 改变修正步长$\alpha$，记录算法迭代次数，如表 2.4.1 所示。

表 2.4.1 权向量初值为 0 时修正步长α与迭代次数 i 的关系

α	0.001	0.01	0.05	0.1	0.3	0.5	0.7	1
i	19532	19532	19532	19532	19532	19532	19532	19532

由此可见，当权向量初值为$\boldsymbol{\theta}=\left[\theta_0,\theta_1,\theta_2\right]^T=\left[0,0,0\right]^T$时，算法迭代次数 i 与修正步长$\alpha$无关。

记录α取不同值的时候，经过迭代后权向量最终取值如表 2.4.2 所示。

表 2.4.2 修正步长α与权向量$\boldsymbol{\theta}$的最终取值

α	0.001	0.01	0.05	0.1	0.5	0.7	1
$\boldsymbol{\theta}$	$\begin{bmatrix}-0.0950\\-0.1326\\0.03977\end{bmatrix}$	$\begin{bmatrix}-0.950\\-1.326\\0.3977\end{bmatrix}$	$\begin{bmatrix}-4.750\\-6.630\\1.9885\end{bmatrix}$	$\begin{bmatrix}-9.500\\-13.26\\3.9770\end{bmatrix}$	$\begin{bmatrix}-47.50\\-66.30\\19.885\end{bmatrix}$	$\begin{bmatrix}-66.50\\-92.82\\27.84\end{bmatrix}$	$\begin{bmatrix}-95.00\\-132.6\\39.770\end{bmatrix}$

可见$\theta=\alpha\left[-95.00,-132.6,39.770\right]^T$。即权向量初值为$\boldsymbol{\theta}=\left[\theta_0,\theta_1,\theta_2\right]^T=\left[0,0,0\right]^T$时，对于不同的修正步长$\alpha$，权向量在每次迭代后的结果均为相互平行的向量，最终结果为一组平行向量，比值为相对应的步长α的比值。

2.4.2 权向量初值为$\boldsymbol{\theta}=\left[\theta_0,\theta_1,\theta_2\right]^T\neq\left[0,0,0\right]^T$的情况

以样本数 n=400 为例。为不失一般性在 MATLAB 用 rand 指令生成随机向量作为初值。生成权向量初值为$\boldsymbol{\theta}=\left[\theta_0,\theta_1,\theta_2\right]^T=\left[0.5293,0.7924,0.7715\right]^T$。从 0.001 到 1 改变修正步长$\alpha$，记录算法迭代次数，如表 2.4.3 所示。

表 2.4.3 权向量初值为 0 时修正步长α与迭代次数 i 的关系

α	0.001	0.003	0.006	0.008	0.01	0.03	0.05	0.08
i	40541	39492	21932	19132	27132	19932	7125	17932
α	0.1	0.2	0.3	0.4	0.5	0.6	0.8	1
i	18732	7525	16732	20725	18332	21525	12732	7932

可见迭代次数 i 与修正步长α紧密相关，在使用感知器算法进行分类时，选择恰当的修正步长可以极大地减少迭代次数。

3. 扩大特征向量维度进行实验

感知器算法可用于特征向量为任意维度的线性可分的数据集。但由于特征向量维度大于三时，无法将结果可视化，不便于观察分类的正确性。因此选择将维度扩大到三维进行实验。

3.1 基础实验

构建一个线性可分的数据集，增广特征向量维度为 d=4，样本数量为 n=400，其中正例 205 个，反例 195 个。

用 MATLAB 做出样本分布如图 3.1.1 所示。

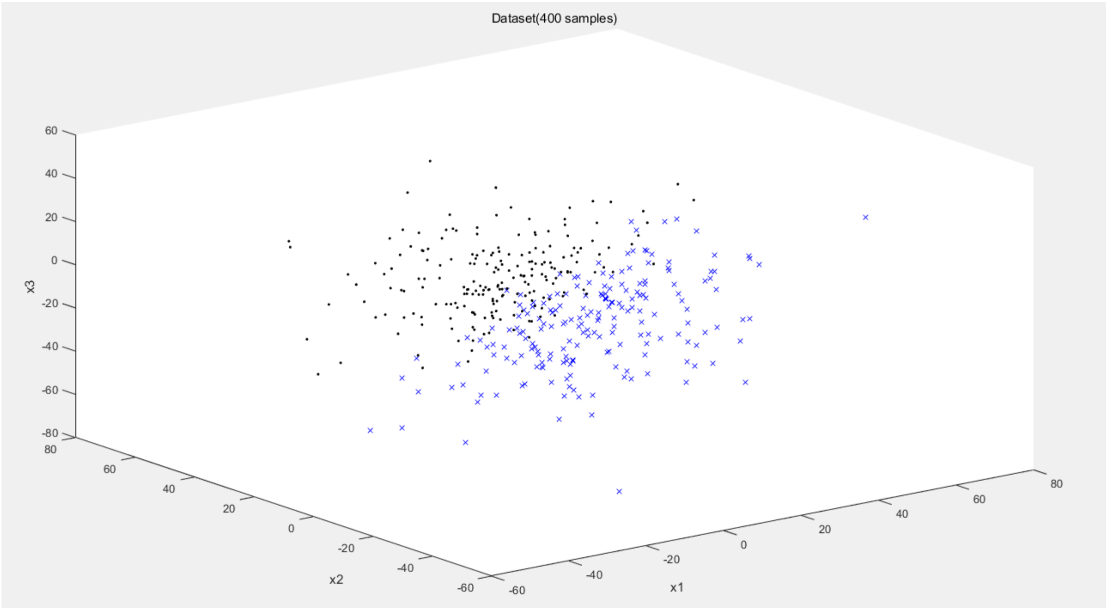

图 3.1.1 样本分布图

取权向量初值为 $\boldsymbol{\theta}=[\theta_0,\theta_1,\theta_2,\theta_3]^T=[0,0,0,0]^T$，修正步长 $\alpha=1$ 进行迭代。

运行程序可得到：经过 i=44394 次迭代后，得到最终权向量 $\boldsymbol{\theta}$ 对于所有样本全部分类正确。做出分类面 $g(\mathbf{x})=\boldsymbol{\theta}^T\mathbf{x}=0$ 如图 3.1.2、图 3.1.3 所示。

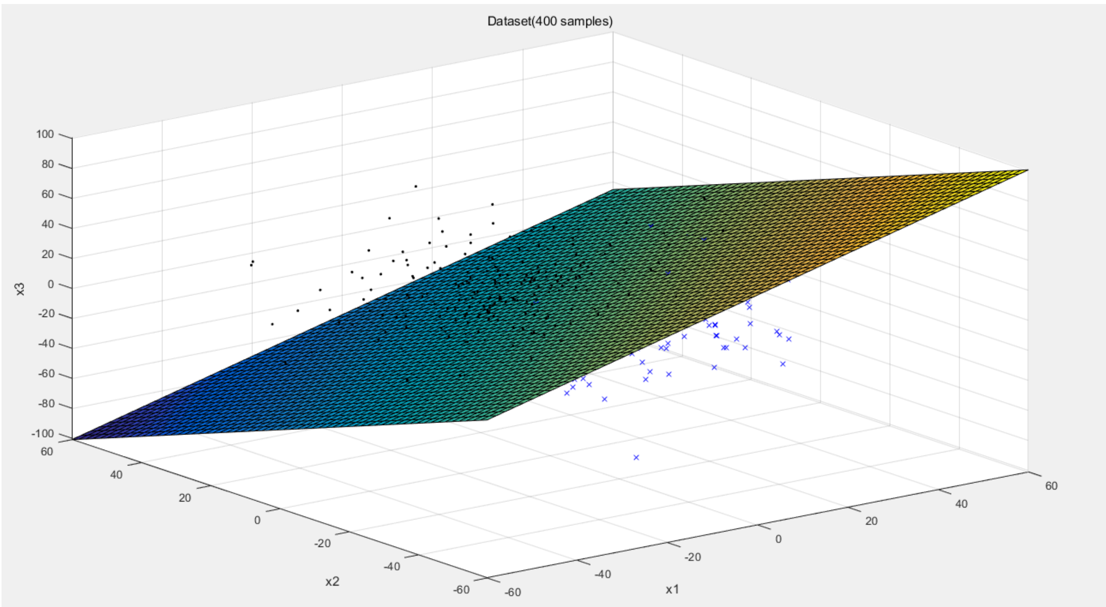

图 3.1.2 分类结果

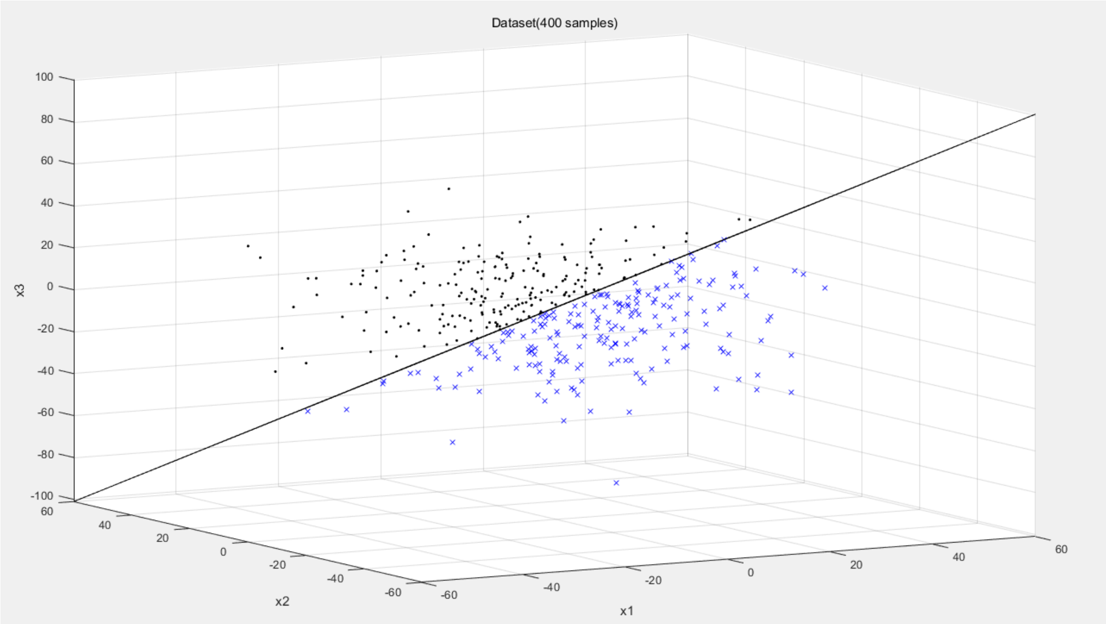

图 3.1.3 分类结果

由图可以看出，该分类判别函数将两类数据完全分开。

3.2 改变样本数目再次实验

取样本数为 n=160 和 n=280、权向量初值为$\boldsymbol{\theta}=[\theta_0,\theta_1,\theta_2,\theta_3]^T=[0,0,0,0]^T$，修正步长$\alpha=1$为例。

样本数 n=160 时，经过 i=9372 次迭代后将所有样本分类正确。分类结果如图 3.2.1 所示。

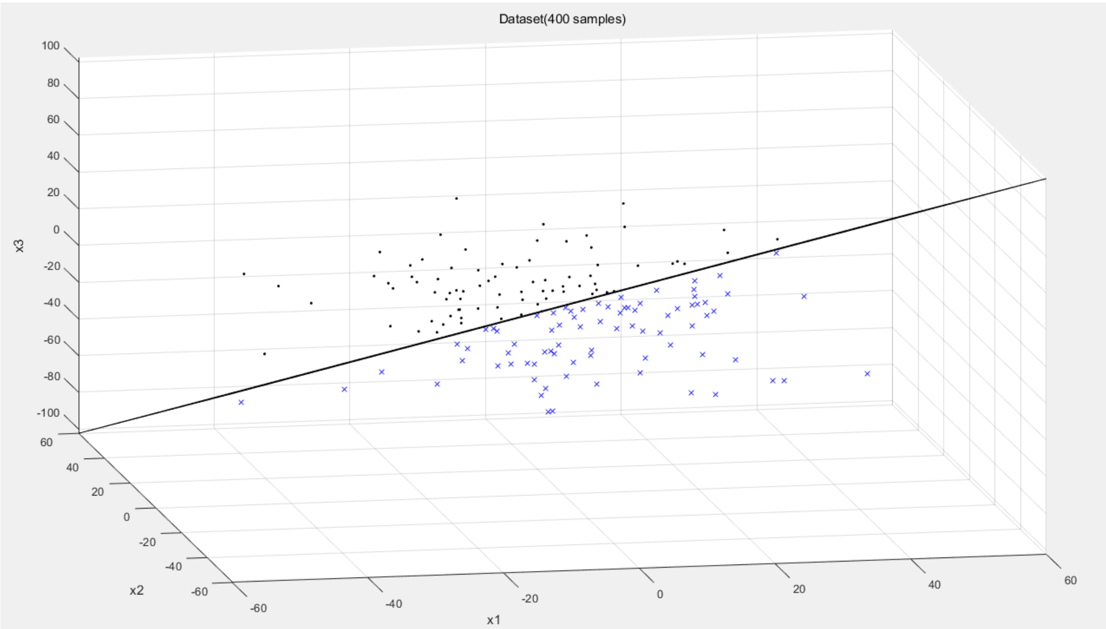

图 3.2.1 n=160 时的分类结果

样本数 n=280 时，经过 i=31905 次迭代后将所有样本分类正确。分类结果如图 3.2.2 所示。

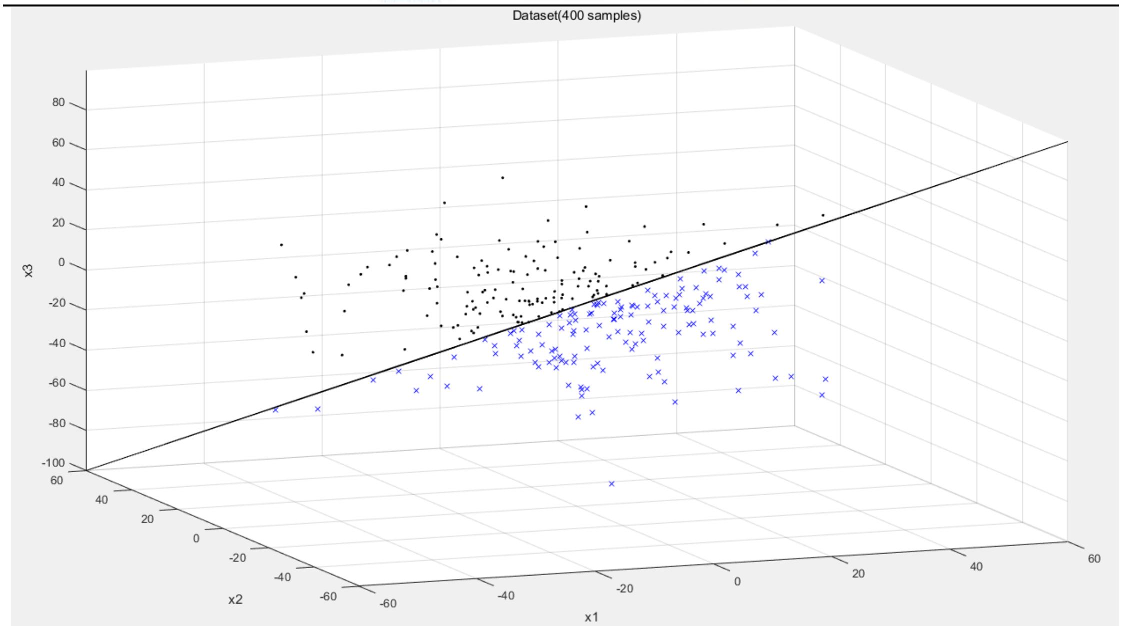

图 3.2.2 n=280 时的分类结果

同理样本数 n 取其他值的时候迭代次数如表 2.2.1 所示。

表 2.2.1 样本数 n 不同取值时的迭代次数

n	100	160	200	240	280	320	360	400
i	623	9372	8122	7328	31905	74748	148682	44394

由此可见，大体上随着样本数 n 的增大，迭代次数 i 增大。与 2 中相比，特征向量维度增加，迭代次数也增大。

3.3 讨论修正步长 α 的影响。

3.3.1 权向量初值为 $\mathbf{\theta}=[\theta_0,\theta_1,\theta_2,\theta_3]^T=[0,0,0,0]^T$ 的情况

与 2.4.1 相同，当权向量初值为 $\mathbf{\theta}=[\theta_0,\theta_1,\theta_2,\theta_3]^T=[0,0,0,0]^T$ 时，对于同一组训练数据，迭代次数 i 不随着修正步长 α 的变化而变化。

3.3.2 权向量初值为 $\mathbf{\theta}=[\theta_0,\theta_1,\theta_2,\theta_3]^T\neq[0,0,0,0]^T$ 的情况

以样本数 n=200 为例。为不失一般性在 MATLAB 用 rand 指令生成随机向量作为初值。生成权向量初值为 $\mathbf{\theta}=[\theta_0,\theta_1,\theta_2,\theta_3]^T=[0.3730,0.4360,0.9867,0.8962]^T$。从 0.001 到 1 改变修正步长 α，记录算法迭代次数，如表 3.3.1 所示。

表 3.3.1 权向量初值非 0 时修正步长 α 与迭代次数 i 的关系

α	0.001	0.003	0.006	0.008	0.01	0.03	0.05	0.08
i	137922	111722	105322	85522	105722	115522	120722	79322
α	0.1	0.2	0.3	0.4	0.5	0.6	0.8	1
i	109922	106722	83522	13922	92722	14242	13442	103922

可见迭代次数 i 与修正步长 α 紧密相关，在使用感知器算法进行分类时，应选择恰当的修正步长以减少迭代次数。

4. 改进方法

虽然利用感知器方法可以求出分类判别函数，但是由 2.2 及 3.2 可以看出，感知器算法需要迭代的次数很多。当样本数较大时，迭代次数很大，对于计算资源消耗很大，效率较低。因此选择另一种算法——对数几率回归。

4.1 基本原理

设样本特性向量为 d 维随机向量 $\mathbf{x}=[x_1,x_2,...,x_d]^T$，为讨论方便，将向量 $\vec{x}$ 增加一维，其值取常数 1，即有：

$$\mathbf{x}=[1,x_1,x_2,...,x_d]^T$$

定义线性化系数 $\boldsymbol{\theta}=\left[\theta_0,\theta_1,..,\theta_d\right]^T$，则对应线性判别函数为 $g(\mathbf{x})=\boldsymbol{\theta}^T\mathbf{x}=z$。

考虑二分类问题，定义其输出标记 $y\in\{0,1\}$。而线性回归模型产生的预测值为实数，于是可以利用 Sigmoid 函数将预测值转换为一个接近 0 或者 1 的值，Sigmoid 函数表达式如下：

$$y=\frac{1}{1+e^{-z}} \tag{4-1}$$

其中，z 为线性预测值，带入 z 的表达式有：

$$y=\frac{1}{1+e^{-\boldsymbol{\theta}^T\mathbf{x}}} \tag{4-2}$$

定义代价函数为：

$$J(\theta)=\frac{1}{2m}\sum_{i=1}^{m}\cos t(h_\theta(x^{(i)}),y^{(i)}) \tag{4-3}$$

其中：

$$\cos t(h_\theta(x^{(i)}),y^{(i)})=\begin{cases}-\log\left[h_\theta(x^{(i)})\right],y^{(i)}=1\\-\log\left[1-h_\theta(x^{(i)})\right],y^{(i)}=0\end{cases}\quad,\quad h_\theta(x)=\frac{1}{1+e^{-\theta^T x}}$$

$$=-y^{(i)}\log\left[h_\theta(x^{(i)})\right]-(1-y^{(i)})\log\left[1-h_\theta(x^{(i)})\right]$$

式中，m 为样本数。

对数几率回归法目标是最小化代价函数 $J(\theta)$，可以采用提梯度下降法或 MATLAB 中 fminunc 函数求解。

4.2 对数几率回归法实验结果

为方便起见，仅对于二维特征样本进行分类。数据集与 2 中数据集相同。分别利用自己编写的梯度下降法与 MATLAB 包含的 fminunc 函数进行求解。以样

本数 n=400 为例。

首先利用梯度下降法求解。经过 i=300 次迭代，得到分类结果如图 4.2.1 所示。

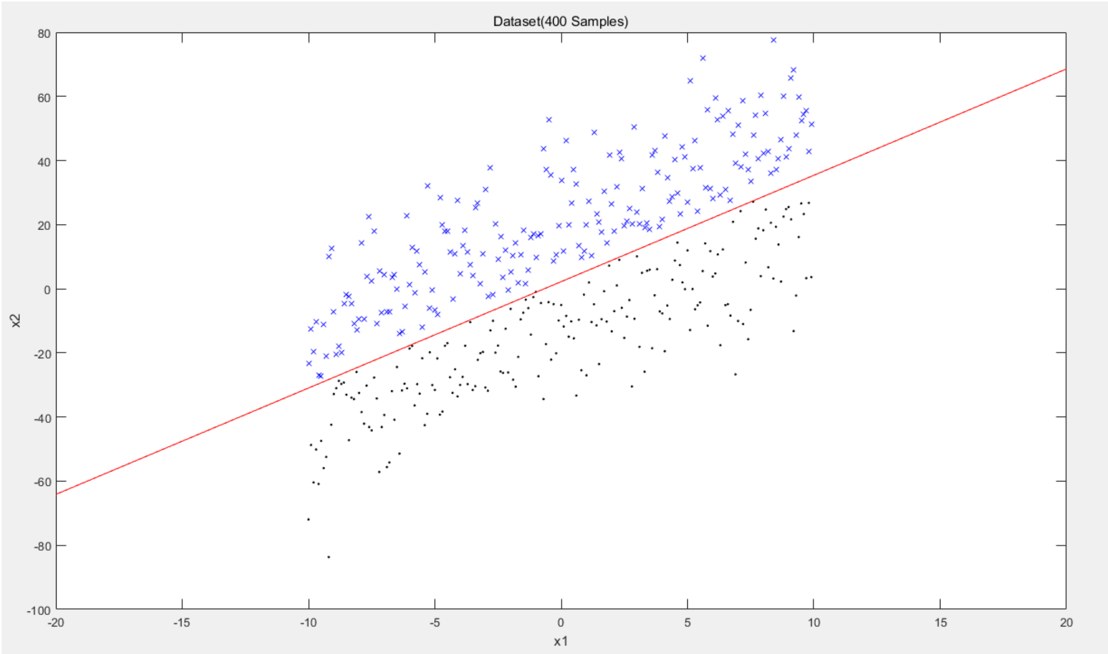

图 4.2.1 n=400 时梯度下降法分类结果

由图中可以看出，经过 300 次梯度下降算法迭代，所得到的分类器已经可以全完将两类样本分开。

再利用 MATLAB 中包含的 fminunc 函数进行拟合，经过 i=15 次迭代，分类结果如图 4.2.2 所示。

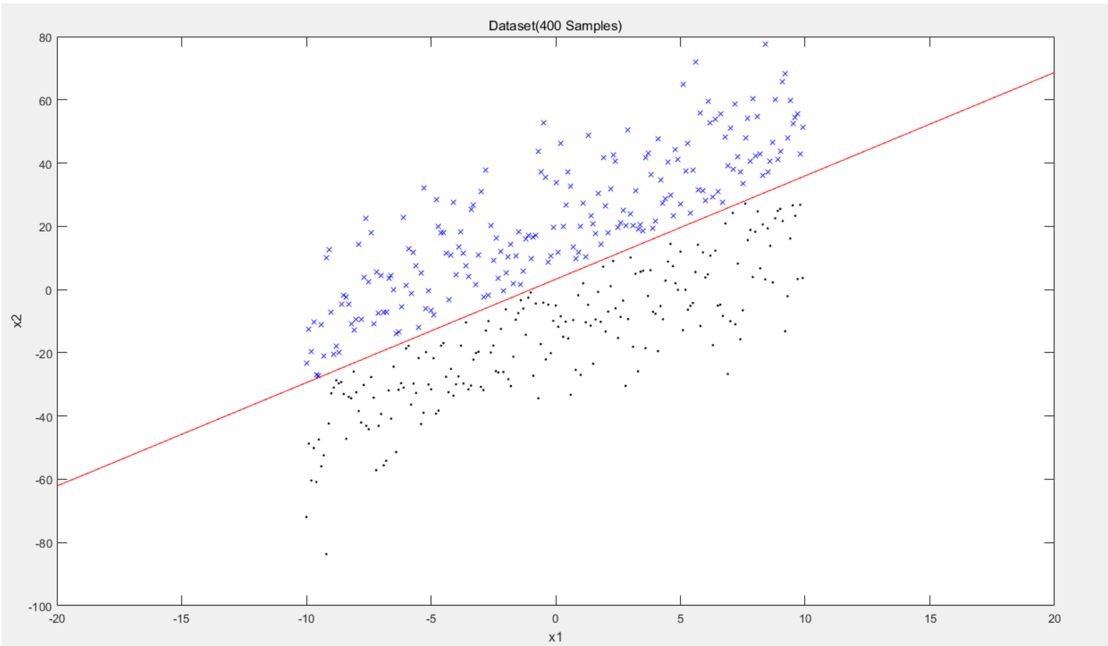

图 4.2.2 n=400 时调用 fminunc 函数分类结果

可见，利用 MATLAB 中包含的 fminunc 函数，仅通过 i=13 次迭代即可以得到线性分类器使两类样本完全分开。

同样的，改变样本集数目进行实验并记录所需迭代次数，如表 4.2.1 和表 4.2.2

所示。

表 4.2.1 n 不同时梯度下降算法的迭代次数

n	100	200	300	400	500	600	700	800
i	80	130	170	300	570	740	1160	1520

表 4.2.2 n 不同时 fminunc 函数的迭代次数

n	100	200	300	400	500	600	700	800
i	2	3	8	13	15	15	17	20

对比表 2.2.1 可以看出，使用对数几率回归法，可以极大地减少迭代次数，提升效率。

5. 实验结论

由以上实验可以看出，感知器算法可以对样本进行线性分类。然而感知器算法的局限性很强，只能用于完全线性可分的样本集。对于线性不可分的样本集，采用感知器算法会无法收敛，无法得出分类结果。使用 4 中所述的逻辑线性回归的算法，可以对于线性不可分的样本进行分类，并且使误差最小化，可以认为是感知器算法的一种改进形式。

附录：MATLAB 代码如下

1. 感知器算法

```
clear;clc;clf;
X1 = load('data1.txt');
X2 = load('data2.txt');
x1_1 = [X1(:,1),X1(:,2)]; %y1=[X1(:,3)];
x2_1 = [X2(:,1),X2(:,2)]; %y2=[X2(:,3)];
n=200;
alpha = 1;
for i=1:n
    x1(i,:)=x1_1(200-(n/2)+i,:);
    x2(i,:)=x2_1(200-(n/2)+i,:);
end
for i=1:n
    x1(i,2)=x1(i,2)+10;
    x2(i,2)=x2(i,2)-10;
end
plot(x1(:,1),x1(:,2),'bx',x2(:,1),x2(:,2),'k.');
title('Dataset(400 Samples)');
 xlabel('x1');
 ylabel('x2');
hold on;
n1=size(x1,1);
n2=size(x2,1);
x11=[ones(n1,1) x1];
x21=[ones(n2,1) x2];
x21 = -x21;
x = [x11;x21]';
theta = [1;1;1];
count = 0;
j = 1;
for i=1:10000000
    if(theta' * x(:,j) <= 0 )
        theta = theta + alpha * x(:,j);
        count = 0;
    else
        theta = theta;
        count = count + 1;
    end
    j=j+1;
    if(j == (n1+n2+1))
        j = 1;
    end
    if(count == (n1+n2+1))
        break;
    end
end
l1=-20:0.01:20;
l2 = ( -theta(2)*l1 - theta(1) )/theta(3);
plot(l1,l2,'r');
```

2. 逻辑线性回归

```
% sigmoid 函数
function g = sigmoid(z)
g = zeros(size(z));
[a,b]=size(z);
for i=1:a
    for j=1:b
        g(i,j) = 1/(1+exp(-z(i,j)));
    end
end
end

% 代价函数及其偏导数
function [J, grad] = costFunction(theta, X, y)
m = length(y);
J = 0;
grad = zeros(size(theta));
[n,p] = size(theta);

sum = 0;
A = X*theta;
for i=1:m
    sum = sum +   ( -y(i)*log( sigmoid( A(i) ) ) -
(1 - y(i))*log( 1-sigmoid( A(i) ) ) );
end
J=sum/m;

for j=1:n
    sum1=0;
    for i=1:m
        sum1 =
sum1+(sigmoid( A(i) )-y(i))*X(i,j);
    end
```

```
    grad(j) = sum1/m;
end
end

% 梯度下降函数
function [theta] = gradientDescent(X, y, theta, alpha, num_iters)
m = length(y);
for iter = 1:num_iters
    [J1, grad1] = costFunction(theta, X, y);
    theta = theta - alpha * grad1;
end
end

% 逻辑线性回归主函数
clear;clc;
X1_1 = load('data1.txt');
X2_1 = load('data2.txt');
n=400;
alpha = 1;
for i=1:n
    X1(i,:)=X1_1(200-(n/2)+i,:);
    X2(i,:)=X2_1(200-(n/2)+i,:);
end
x1 = [X1(:,1),X1(:,2)]; y1=zeros(n,1);
x2 = [X2(:,1),X2(:,2)]; y2=[X2(:,3)];
plot(x1(:,1),x1(:,2),'bx',x2(:,1),x2(:,2),'k.');
title('Dataset(400 Samples)');
xlabel('x1');
ylabel('x2');
hold on;
X = [x1;x2];
X = [ones(2*n,1),X];
y = [y1;y2];
initial_theta = [0;0;0];
%    options = optimset('GradObj', 'on', 'MaxIter',20);
%    [theta, cost] = fminunc(@(t)(costFunction(t, X, y)), initial_theta, options);
theta = gradientDescent(X, y, initial_theta, alpha, 1470)

l1=-20:0.01:20;
l2 = ( -theta(2)*l1 - theta(1) )/theta(3);
plot(l1,l2,'r');
```