哈尔滨工业大学计算机科学与技术学院

机器学习实验报告

实验题目	姓名	学号	联系方式
logistic 回归	崔路源	1163300402	997898829@qq.com

1. 实验目的实验

- 理解逻辑回归模型
- 掌握逻辑回归模型的参数估计算法

2. 实验要求及实验环境

2.1. 实验要求

1. 实现两种损失函数的参数估计（1.无惩罚项；2.加入对参数的惩罚）
2. 采用梯度下降
3. 采用牛顿法
4. 通过自己产生的数据评价该模型
5. 考察类条件分布不满足朴素贝叶斯假设，会得到什么样的结果。
6. 到UCI网站上，找一实际数据加以测试。

2.2. 实验环境

- matlab-2017b

3. 设计思想

3.1. 算法原理

3.1.1. 预处理

利用矩阵形式计算较为方便

对于 m 条数据，每一条数据有 n 个特征（前面添加的 1 为与w_0搭配的常数项）

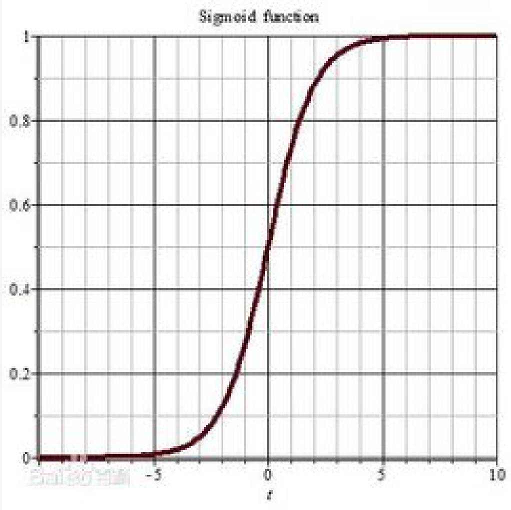
$$X=\begin{bmatrix}1 & x_{11} & x_{12} & \dots & x_{1n}\\ 1 & x_{21} & x_{22} & \dots & x_{2n}\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 1 & x_{m1} & x_{m2} & \dots & x_{mn}\end{bmatrix}_{m\times(n+1)} \qquad Y=\begin{bmatrix}y_1\\ y_2\\ y_3\\ \vdots\\ y_m\end{bmatrix}_{m\times 1} \quad y_i\in\{0,1\} \qquad w=\begin{bmatrix}w_0\\ w_1\\ w_2\\ \vdots\\ w_n\end{bmatrix}_{(n+1)\times 1}$$

$$\text{其中，} x_i=\begin{bmatrix}1 & x_{i1} & x_{i2} & \dots & x_{in}\end{bmatrix}^T$$

sigmoid函数和h_w函数

$$sigmoid(x)=\frac{1}{1+e^{-x}} \qquad h_w(X)=\begin{bmatrix}sigmoid(w^Tx_1)\\ sigmoid(w^Tx_2)\\ \vdots\\ sigmoid(w^Tx_m)\end{bmatrix}_{m\times 1} \quad y_i\in\{0,1\}$$

sigmoid 函数图像如图



3.1.2. 梯度下降法

梯度下降法在拟合正弦曲线的实验中有详细的推导过程，在此不多加赘述，主要讲述梯度下降在 logistic Regression 的应用

定义 logistic Regression 的 cost function

$$J(w) = \frac{1}{m}\sum_{i=1}^{m} cost(h_w(x^{(i)}), y^{(i)})$$

其中

$$cost(h_w(x), y) = \begin{cases} -\log(h_w(x)) & if \quad y = 1 \\ -\log(1 - h_w(x)) & if \quad y = 0 \end{cases}$$

则可得出

$$cost(h_w(x), y) = -y\log(h_w(x) - (1-y)\log(1 - h_w(x))$$
$$J(w) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log(h_w(x^{(i)}) + (1-y^{(i)})\log(1 - h_w(x^{(i)}))]$$

即为了使$J(w)$最小，可以使用梯度下降法，主要循环如下

Repeat {

$w = w - \frac{\alpha}{m}X^T(h_w(X) - Y)$

}

3.1.3. 加惩罚项的梯度下降法

主要步骤和梯度下降相同，代价函数

$$J(w) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log(h_w(x^{(i)}) + (1-y^{(i)})\log(1 - h_w(x^{(i)}))] + \frac{\mu}{2m}||w||^2$$

即为了使$J(w)$最小，使用梯度下降法，主要循环如下

Repeat {

$w = w - \frac{\alpha}{m} X^T (h_w(X) - Y) - \frac{\mu}{m} w$

}

3.1.4. 牛顿法

牛顿法主要解决的是在有些时候某些方程的求根公式可能很复杂（甚至有些方程可能没有求根公式），导致求解困难的问题。利用牛顿法进行迭代求解，得出最后的最优解。

假设有函数$f(w)$，需要找使$f(w)$=0的

步骤：

1. 给出一个w的初始值
2. 对$f(w)$求导，求导数为0 w时的值（就是求$f(w)$切线与x轴交点）
3. 重复步骤2

因为该点的导数值即为切线斜率，而斜率=该点y轴的值/该点x轴的变化值，所以w每次的变化值：

$$w = w - \frac{f(w)}{f'(w)}$$

- 使用这个方法需要函数满足一定条件，适用于Logistic回归

应用于 Logistic Regression

求对数似然的最大值（去掉代价函数的符号，就是求后面式子的最大值），即$J'(w)$求为0时的，根据上述推论，更新规则如下：

$$w = w - \frac{J'(w)}{J''(w)}$$

牛顿方法的收敛速度：二次收敛

每次迭代使解的有效数字的数目加倍：假设当前误差是0.1，一次迭代后，误差为0.001，再一次迭代，误差为0.0000001。该性质当解距离最优质的足够近才会发现

3.2. 算法实现

此处展示算法实现的关键代码

sigmoid 函数

```
function [h_d] = sigmoid(Data,w)
  h_d = 1 ./(1+exp(-(Data(:,1:3) * w)));
 end
```

3.2.1. 梯度下降法

```
function [w,c] = gradient(Data,w,alpha)
  c = 0;
  while true
```

```
    w_0 = w;
    %迭代
    w = w - alpha / 200 * Data(:,1:3)'*(sigmoid(Data,w) - Data(:,4));
    c = c + 1;
    if ( (max(w_0 - w) )< 10^(-6))
        break;
    end
  end
end
```

3.2.2. 带惩罚项的梯度下降

与梯度下降差别不大，只是多了一个惩罚项

```
function [w,c] = gradient(Data,w,alpha,mu)
  c = 0;
  while true
    w_0 = w;
    w = w - alpha / 200 * Data(:,1:3)'*(sigmoid(Data,w)-Data(:,4)) + mu / 200 * w;
    c = c + 1;
    if ( (max(w_0 - w) )< 10^(-6))
        break;
    end
  end
end
```

3.2.3. 牛顿法

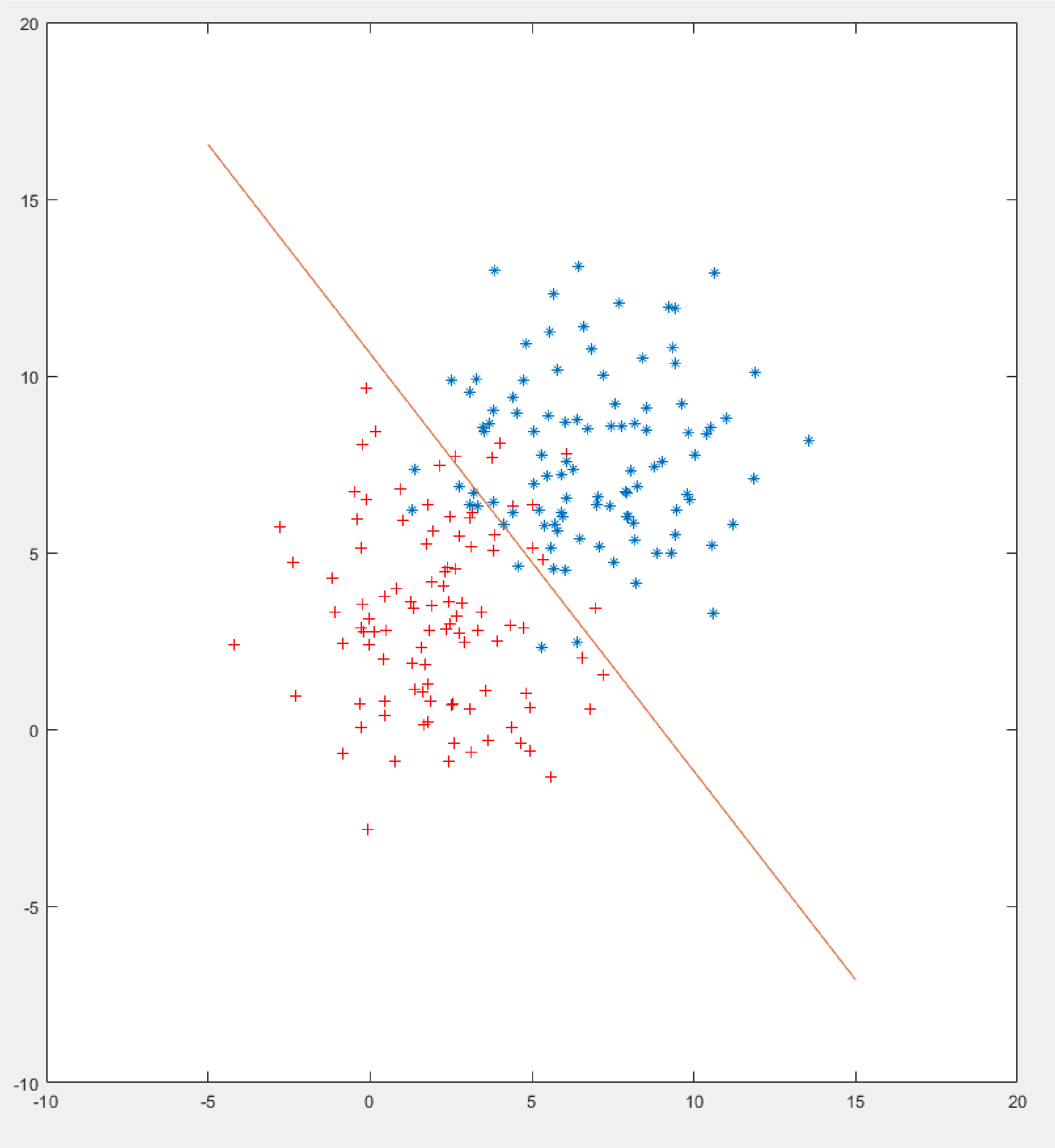
```
function [w,c] = newton(Data)
w = ones(3,1);
c = 0;
while true
    h_w = sigmoid(Data,w);
    %求梯度
    grad = Data(:,1:3)' * (h_w-Data(:,4));
    %求 Hessian 矩阵
    hessian = h_w' * (1 - h_w)*(Data(:,1:3)'* Data(:,1:3));
    temp = pinv(hessian)*grad;
    %迭代更新
    w = w - temp;
    c = c + 1;
    if ((temp' * temp)<10^-7)
        break
    end
end
end
```

4. 实验结果与分析

4.1. 使用梯度下降法

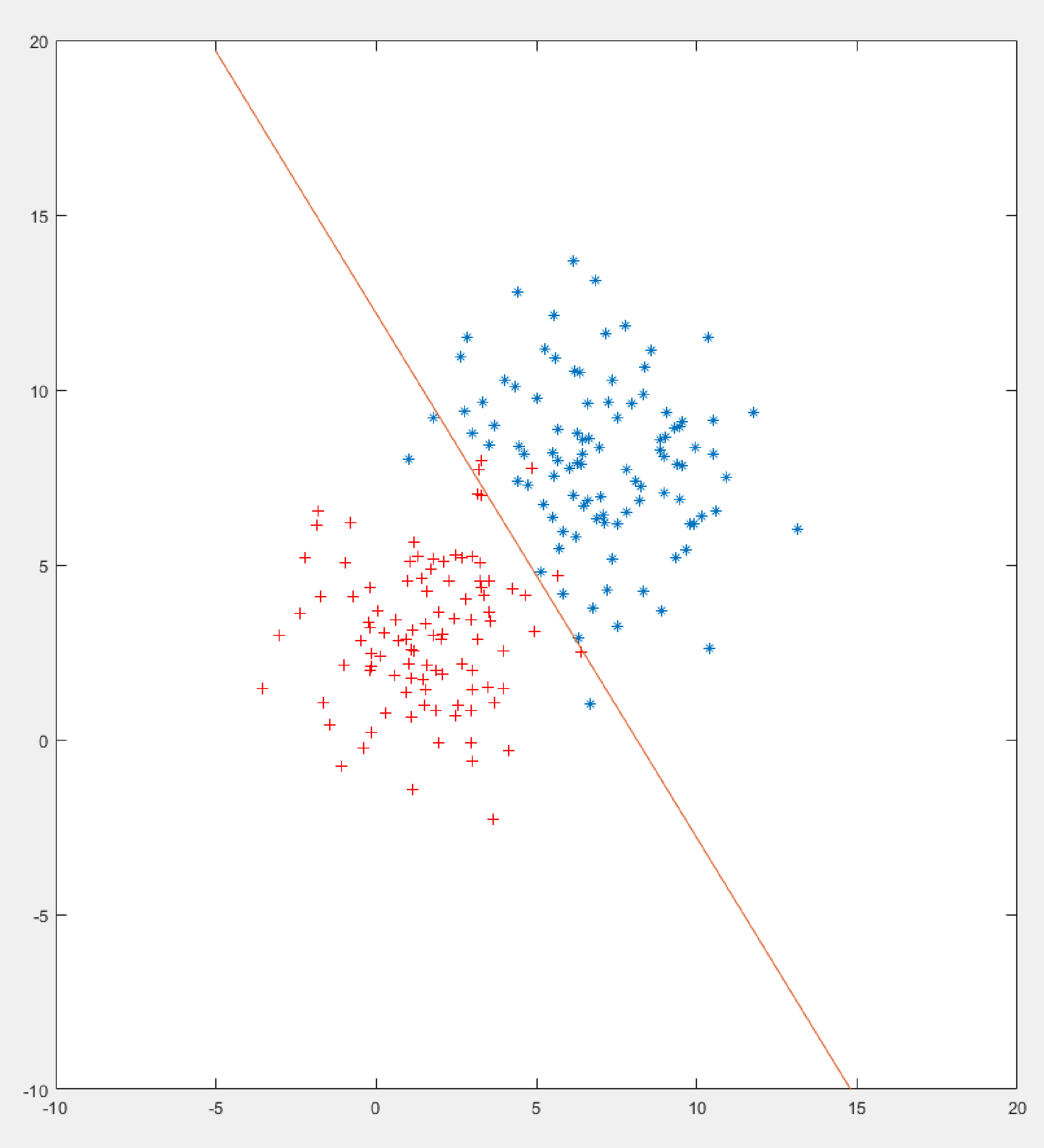
参数的变化小于 10^-6 时，迭代停止

4.1.1. 学习率 0.001



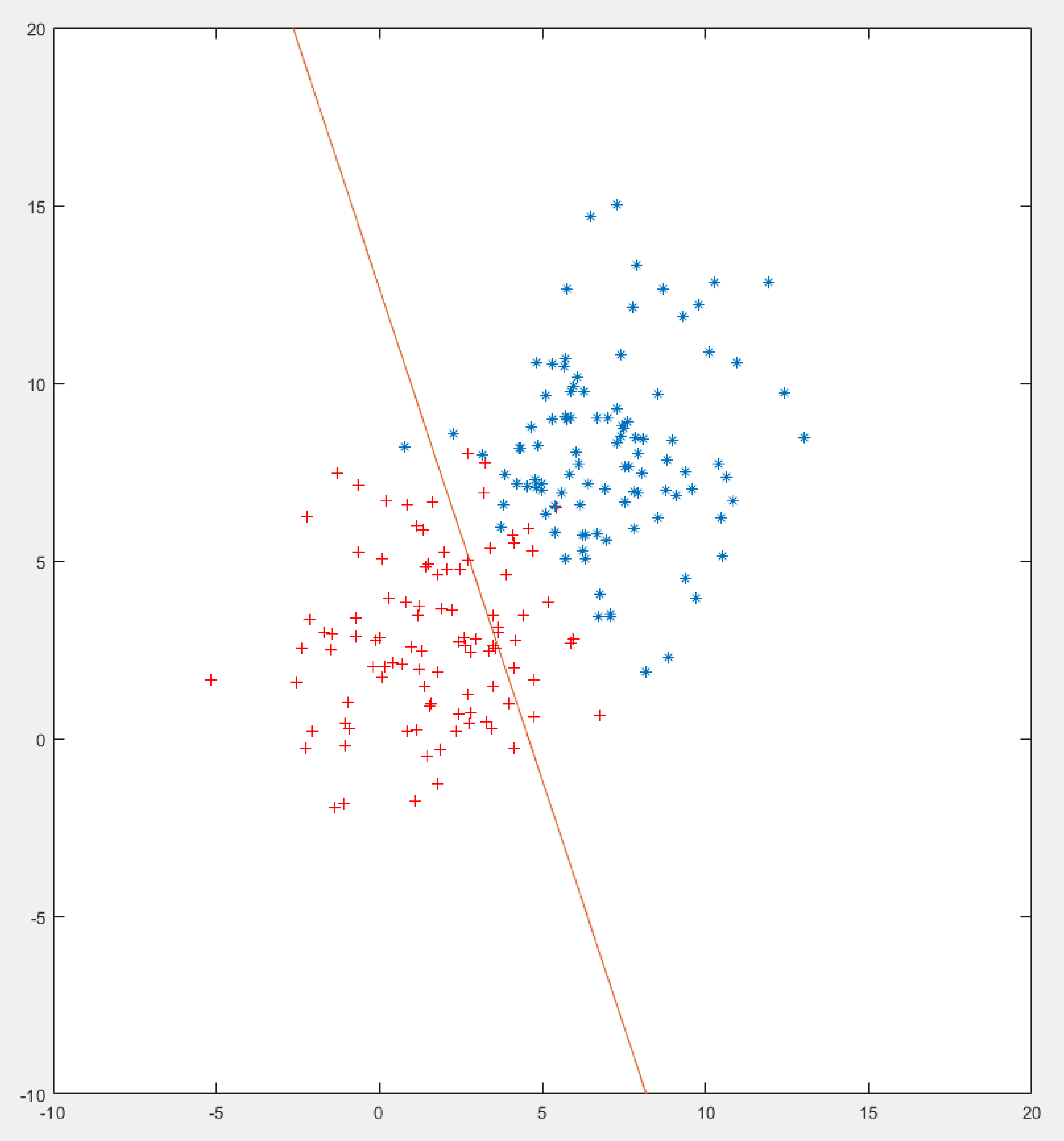
迭代次数为 914846, w = [-8.67892092923665；0.964453570602148；0.816248977268106]

4.1.2. 学习率 0.1



迭代次数：89289，w = [-13.0086995565818；1.60029683508364；1.06765768949902]

4.1.3. 学习率 0.00001

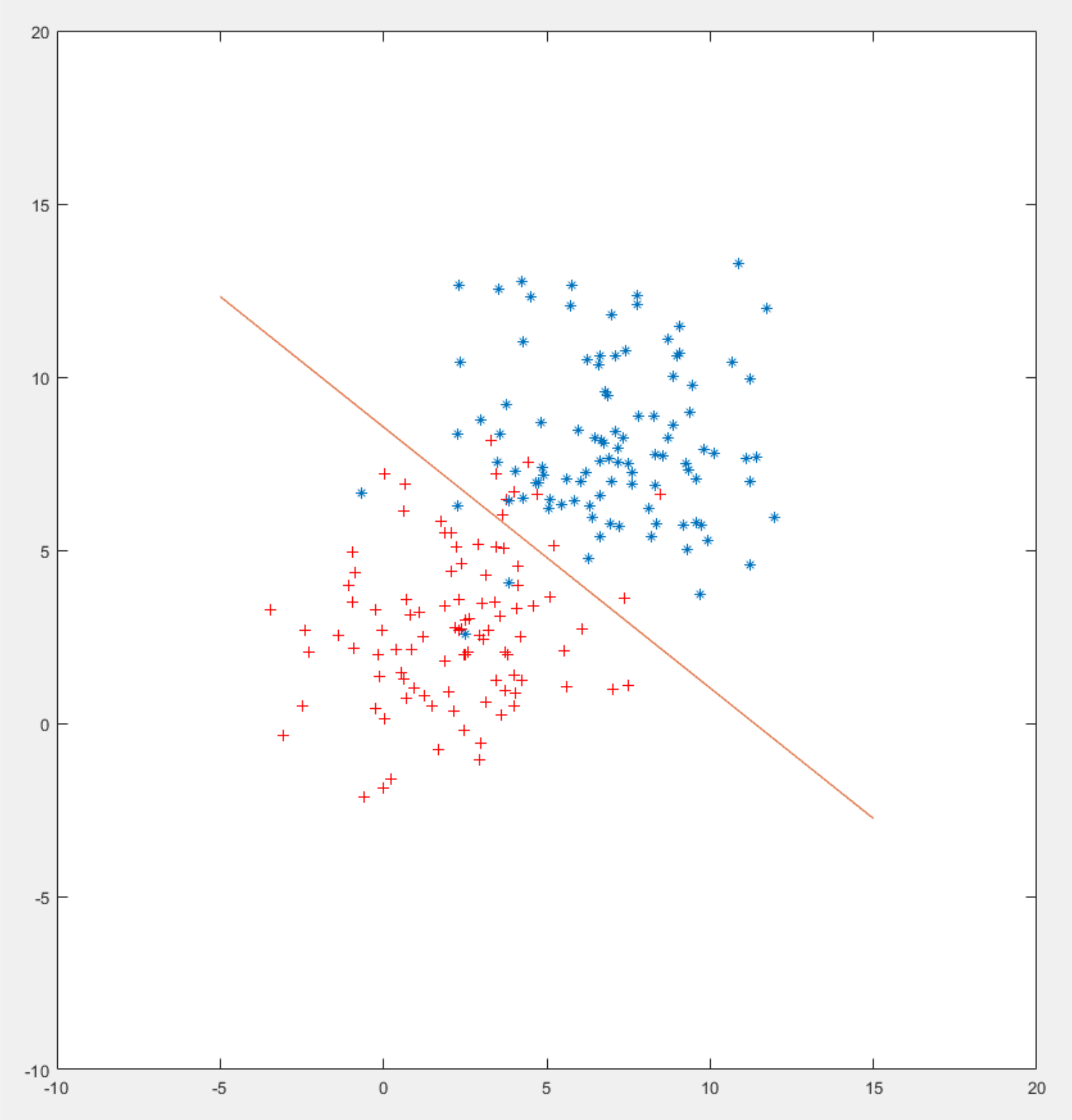


迭代次数：1552234，w = [-1.56143419113930；0.341458461221696；0.123288967088556]

4.2. 带惩罚项的梯度下降法

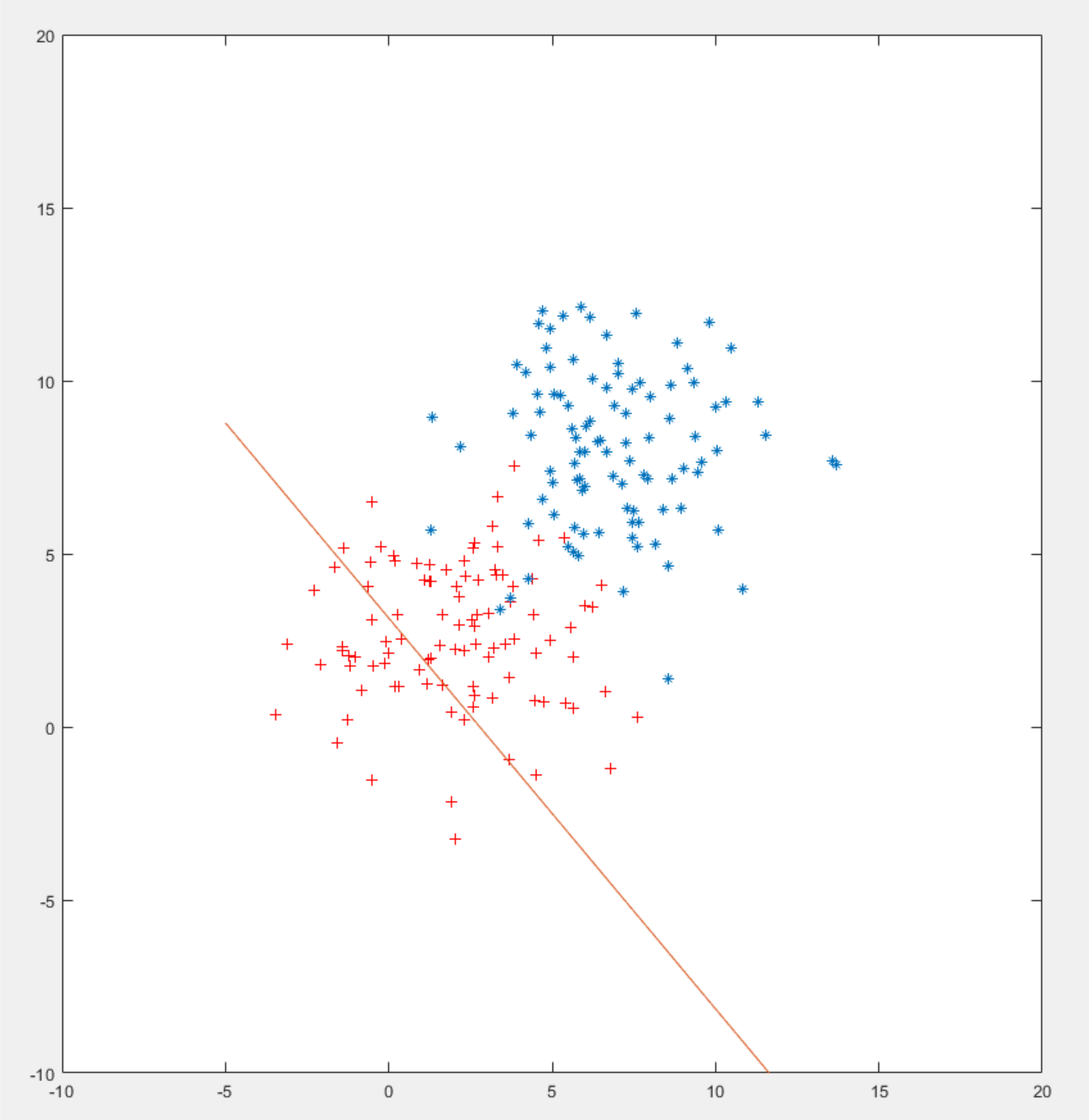
为研究惩罚系数的影响，因此选择学习率为0.001，当参数的变化小于 10^-6 时，迭代停止

4.2.1. 惩罚系数 0.001



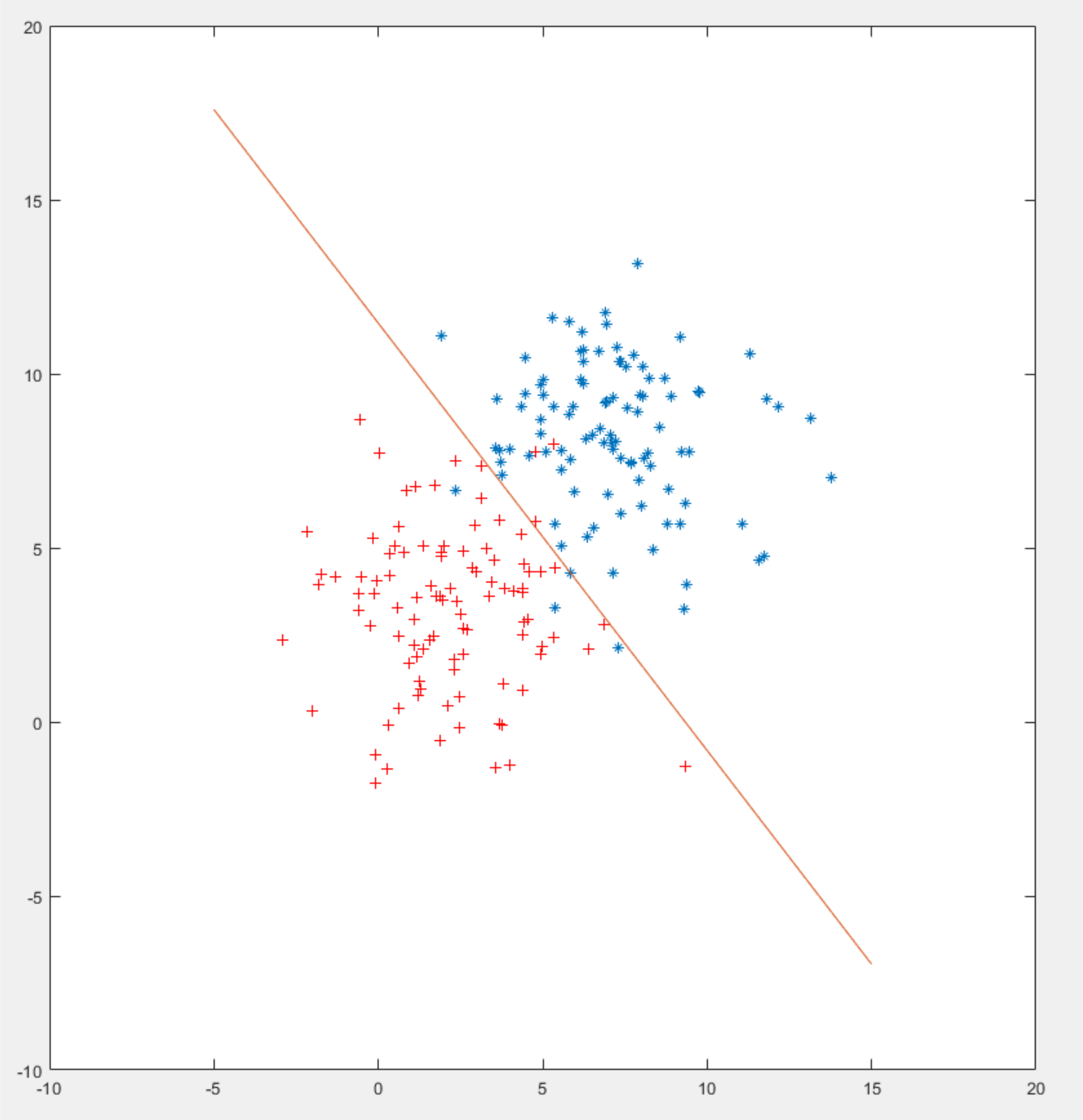
迭代次数：223821，w = [-4.54068551080060；0.400077172601330；0.530802723483825]

4.2.2. 惩罚系数 0.1



迭代次数：11554，w = [-0.327214259020837；0.117604845471224；0.104187857521533]

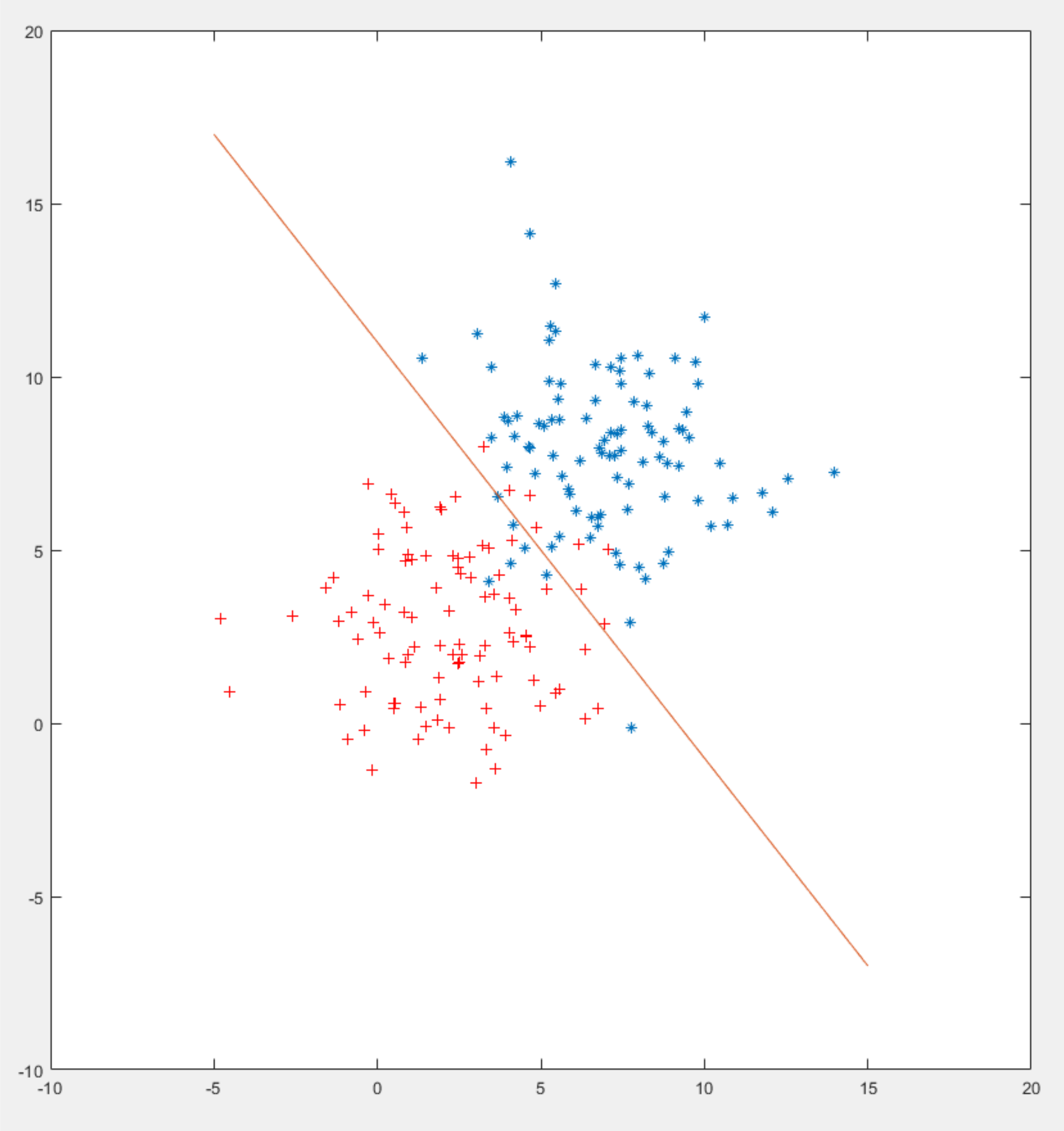
4.2.3. 惩罚系数 0.00001



迭代次数：1611772， w = [-11.7738150453466；1.26180981837968；1.02792947059673]

4.3. 牛顿法

当 $(hessian^{-1} \times grad)^T \times (hessian^{-1} * grad)$ 小于 10 ^ -8 时



迭代次数6912，w = [-11.0532732191919；1.20616207385866；1.00517595407848]

5. 结论

在本实验中，主要学习了在 Logistic Regression 中使用梯度下降法和牛顿法的具体编程实现，并且比较了惩罚系数对决策边界的影响

- 梯度下降法需要选择合适的步长，确保迭代次数在一个合理的可控范围内，加入惩罚项之后，需要对惩罚系数做出调整才能得到较好的决策边界
- 牛顿法收敛速度为二次收敛，迭代速度很快，但是牛顿法由于需要产生 Hassian 矩阵，并且在对 Hassian 矩阵求逆的时候需要的计算量也很大，是影响牛顿法效率的一个重要因素。

6. 参考文献

- 机器学习-斯坦福 学习笔记4 - 牛顿方法: https://blog.csdn.net/maverick1990/article/details/12564973
- 牛顿法解机器学习中的Logistic回归：https://blog.csdn.net/baimafujinji/article/details/51179381
- 维基百科

7. 附录

- 画数据
    - drawData.m
- 梯度下降
    - 无惩罚项：LR_GradientDescent_NoPenalty.m
    - 有惩罚项：LR_GradientDescent_Penalty.m
- 牛顿法
    - LR_NewtonMethod.m