哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

2020 年春季学期

计算学部《机器学习》课程

Lab 2 实验报告

姓名	包宇昊
学号	1180300605
班号	1803105
电子邮件	2568235796@qq.com
手机号码	18267116288

目录

建议写出：问题的描述，解决问题的思路，实验的做法，实验结果的分析，结论，自拟标题

1 .实验目标

理解逻辑回归模型，掌握逻辑回归模型的参数估计算法。

2 .实验要求及环境

要求：

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），

可以采用梯度下降、共轭梯度或者牛顿法等。

验证：

1.可以手工生成两个分别类别数据（可以用高斯分布），验证你的算法。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。

2.逻辑回归有广泛的用处，例如广告预测。可以到 UCI 网站上，找一实际数据加以测试。

实验环境：

pycharm 2020.2.2

python 3.7

win10

X86-64

3 .数学思想

二项逻辑回归模型如下（sigmoid 函数）：

$$P(Y=0|x)=\frac{1}{1+e^{\omega\cdot x+b}}$$

$$P(Y=1|x)=\frac{e^{\omega\cdot x+b}}{1+e^{\omega\cdot x+b}}$$

我们需要的 LR 分类器本质上就是权值ω，ωX得到的线即为决策边界。

其中 x 是输入的样本点，$Y\in\{0,1\}$ 是输出，ω 和 b 是参数，ω 称为权值向量，b 称为偏置。为了方便求解我们记:

$$\omega = (\omega^1,\omega^2,\omega^3,\dots,\omega^n,b)^T ,$$

$$x = (x^1,x^2,x^3,\dots,x^n,1)^T$$

再代回之前的模型得到：

$$P(Y=0|x)=\frac{1}{1+e^{\omega\cdot x}},$$

$$P(Y=1|x)=\frac{e^{\omega\cdot x}}{1+e^{\omega\cdot x}}$$

采用似然函数法去计算系数ω

设$P(Y=1\mid x)=\pi(x), P(Y=0\mid x)=1-\pi(x)$

似然函数为$\prod[\pi(x_i)^{y_i}][1-\pi(x_i)]^{1-y_i}$

为计算考虑取对数：

$$L(\omega)=\sum_{i=1}^{N}[y_i log\pi(x_i)+(1-y_i)log(1-\pi(x_i))]$$

$$=\sum_{i=1}^{N}[y_i(\omega\cdot x_i)-log(1+e^{\omega\cdot x_i})]$$

我使用**梯度下降法**进行参数估计

不加正则项的损失函数：

$$L(\omega)=\sum_{i=1}^{N}[-y_i(\omega\cdot x_i)+log(1+e^{\omega\cdot x_i})]$$

不加正则项求ω，对损失函数求导得到梯度：

$$\frac{\partial L}{\partial w_j}=x_{ij}(-y_i+sigmoid(wx_i))$$

加正则项的损失函数：

$$L(\omega)=-\frac{1}{N}\sum_{i=1}^{N}[y_i log\pi(x_i)+(1-y_i)log(1-\pi(x_i))]+\frac{\lambda}{2N}||\omega||_2^2$$

$$=\frac{1}{N}\sum_{i=1}^{N}[-y_i(\omega\cdot x)+log(1+e^{\omega\cdot x_i})]+\frac{\lambda}{2N}||\omega||_2^2$$

加正则项求ω，对损失函数求导得到梯度：

$$\frac{\partial L}{\partial w_j}=\frac{1}{N}[x_{ij}(-y_i+sigmoid(wx_i))+\lambda\cdot\omega]$$

迭代公式如下：$\omega=\omega-\alpha*L'(\omega)$

计算得到ω后再就得到了决策边界，把数据代入进行测试，画出图像，并计算匹配率。

4 .算法实现

Sigmoid 函数防止下溢出和浪费计算资源，对其进行限制：

```
def sig(x): #sigmoid函数
    if x < -10:
        return 0
    else:
        return 1 / (1 + np.exp(-x))
```

计算损失函数：

```
def cal_loss1(X,Y,W):   #损失函数,X.shape[0]即为数据量
    l = 0
    for i in range(X.shape[0]):
        wx = np.dot(X[i],W)
        l = -Y[i]*wx + np.log(1+np.exp(wx))
    loss = l
    return loss
```

梯度下降：按数学公式实现循环即可

计算匹配率：ωx与 0 进行比较，<=0 则是 y=0,>0 是 y=1 的类

匹配率 = 匹配成功的数量/总数量

```
#计算匹配率
def cal_ratio(X,W):
    c = 0  #c为匹配成功的样本数
    a = int(X.shape[0]/2)
    for i in range(a):
        if (np.dot(X[i],W) < 0 or np.dot(X[i],W) == 0):
            c += 1
    for i in range(a,X.shape[0]):
        if (np.dot(X[i],W) > 0):
            c += 1
    return c/X.shape[0]
```

4.1 自己的测试

生成数据：
分别生成满足和不满足贝叶斯分布的数据，改变协方差（cov）即可，体现在协方差矩阵上就是是否半正定。

满足：

```
cov = [[1,0], [0,1]]
mean1 = (1,1)
mean2 = (3,3)
size = int(N/2)
np.random.seed(5)
train0 = np.random.multivariate_normal(mean1, cov, size)  # y为0
train1 = np.random.multivariate_normal(mean2, cov, size)  # y为1
```

不满足：`cov = [[2,1], [1,2]]`

不加正则项迭代：

```
for num in range(n):
    loss0 = loss
    for j in range(W.shape[0]):
        for i in range(X.shape[0]):
            dw[j] += X[i][j] * (-Y[i] + sig(np.dot(X[i], W)))  #???
        dw[j] = dw[j]/X.shape[0]
    W = W - a * dw
    # print(loss)
    # print(W)
    loss = cal_loss1(X, Y, W)
    L[num] = loss
    if (abs(loss - loss0) < 1e-10):
        break
```

带正则项迭代：

```
dw[j] = (dw[j]+lemda*W[j])/X.shape[0] + lemda*dw[j]
```

4.2 UCI 测试

UCI 测试需要读取数据，忽略其中的“,”，再对矩阵进行切分和扩充，以满足计算需要：

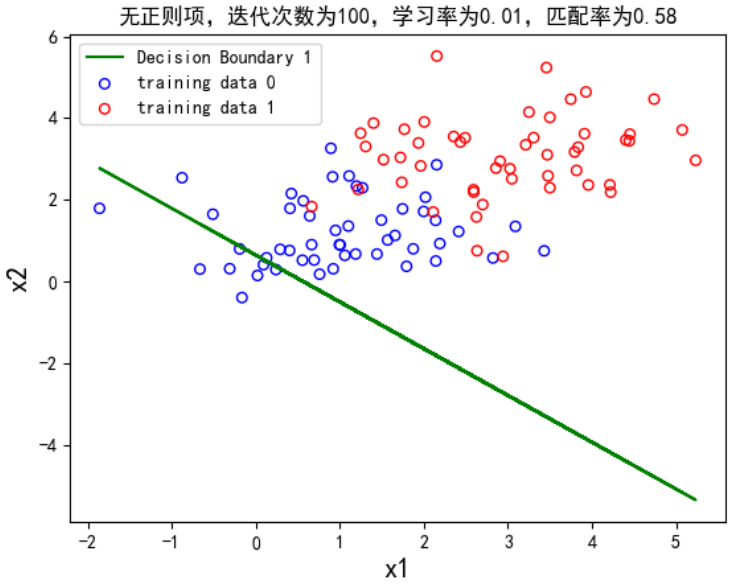
```
# data = np.loadtxt("cmc.data",delimiter=',')    #对该数据进行测试时，dw手
data = np.loadtxt("data_banknote_authentication.txt", delimiter=',')
x = data[:,:data.shape[1]-1]
X = np.insert(x,x.shape[1],values=1,axis=1)
Y = data[:,data.shape[1]-1]
```

5 .测试效果

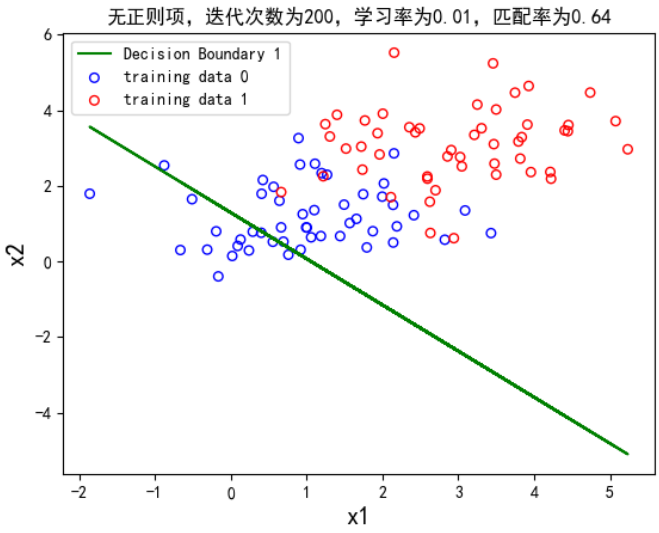
5.1 迭代次数的影响

数据量 N=100，学习率 a=0.01，不加正则项

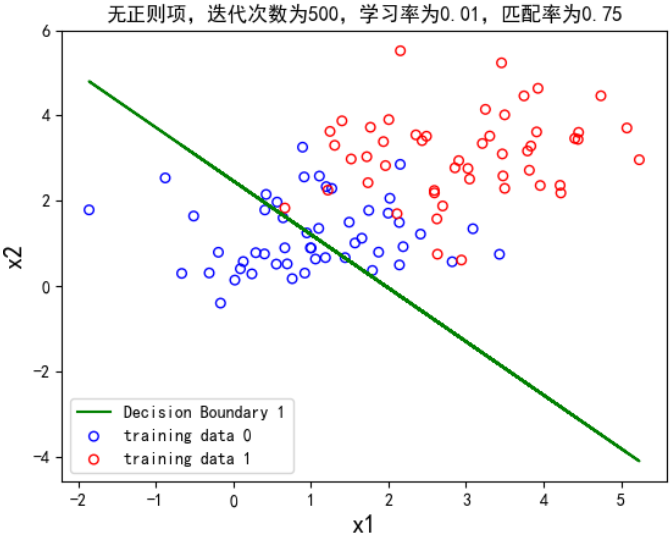
迭代次数为 100：



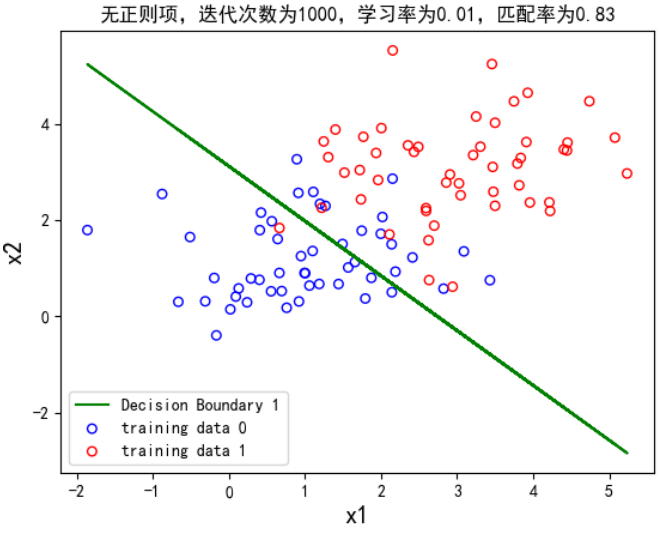
迭代次数为 200：



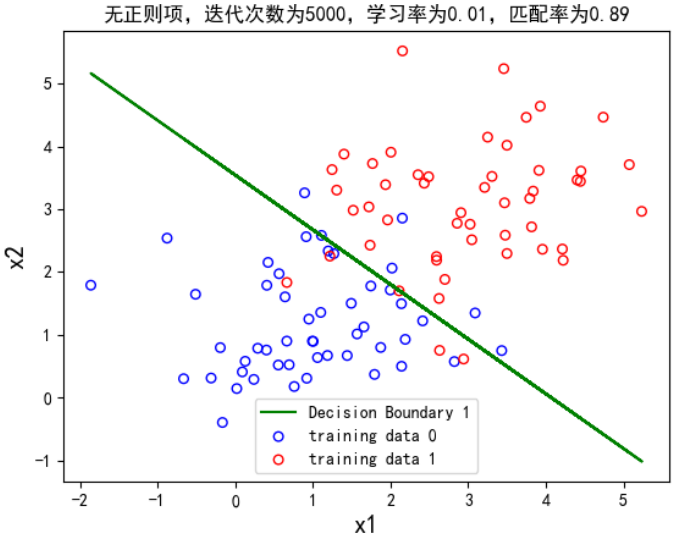
迭代次数为 500：



迭代次数为 1000：



迭代次数为 5000：

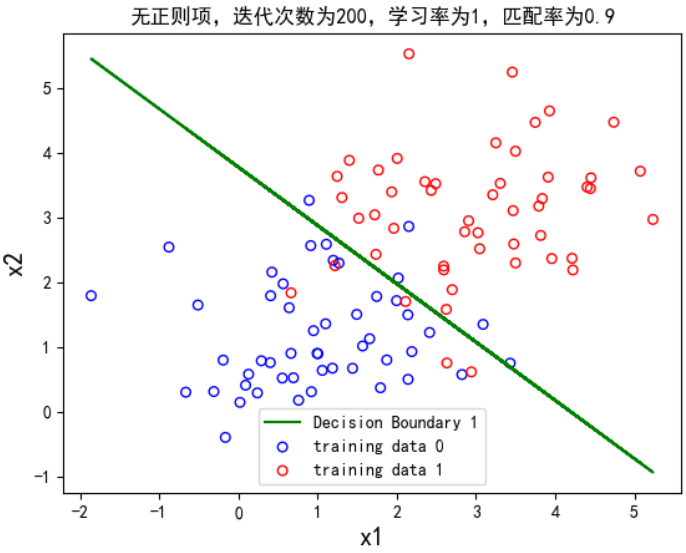


如图，显然：迭代次数越高，匹配率越高（但次数太高可能出现震荡，我的 break 条件也不能完全约束）

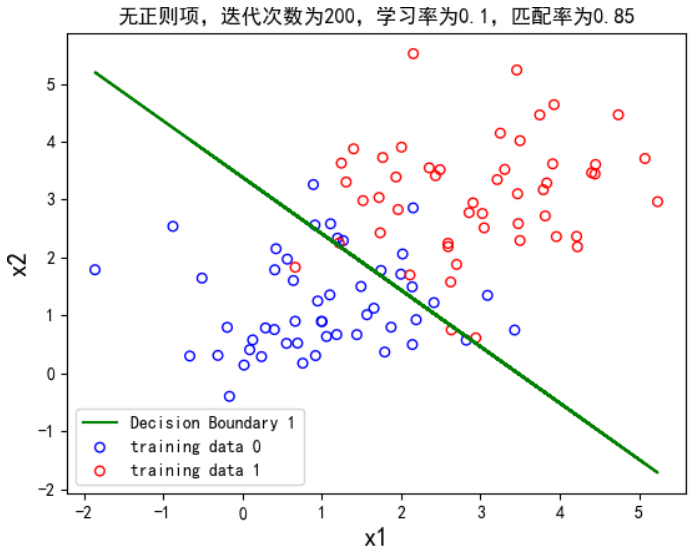
5.2 学习率的影响

数据量 N=100，步长为 1000，不加正则项

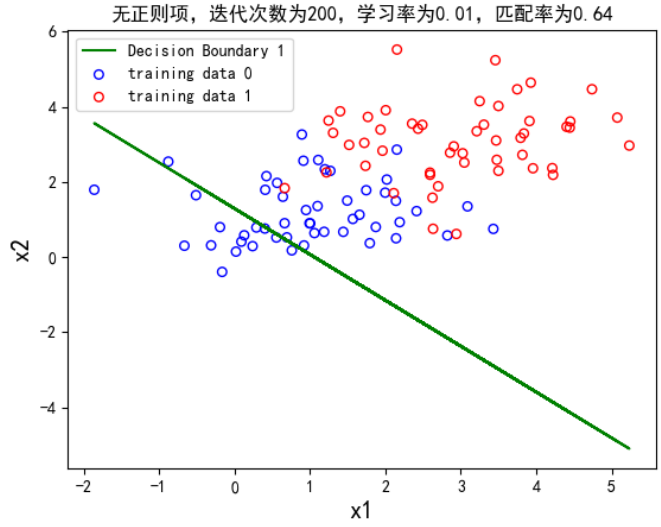
学习率为 1：



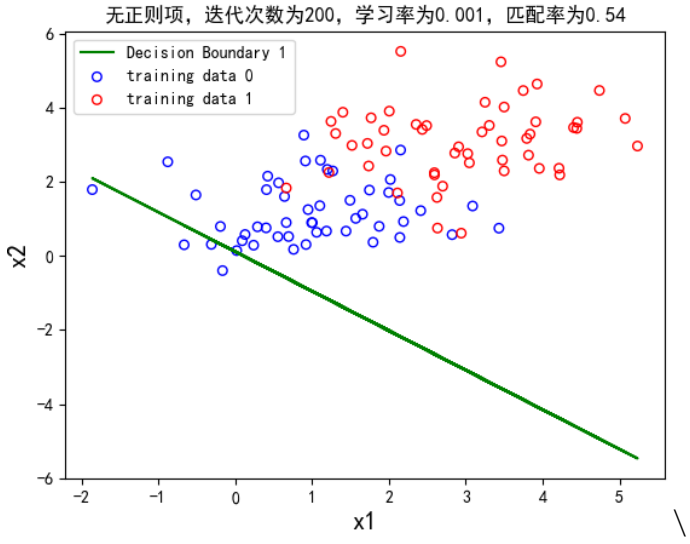
学习率为 0.1：



学习率为 0.01：



学习率为 0.001：



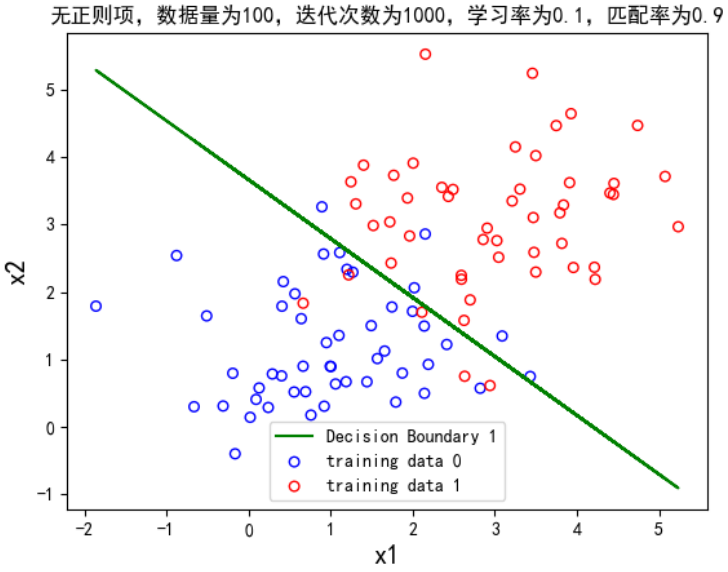
\

可以看出，当数据量 N=100,步长 1000 时，学习率 0.1-1 之间较合适。
而和 5.1.1 中的图像进行比较发现，学习率和样本数量也有关系，需要根据实际情况进行调整。

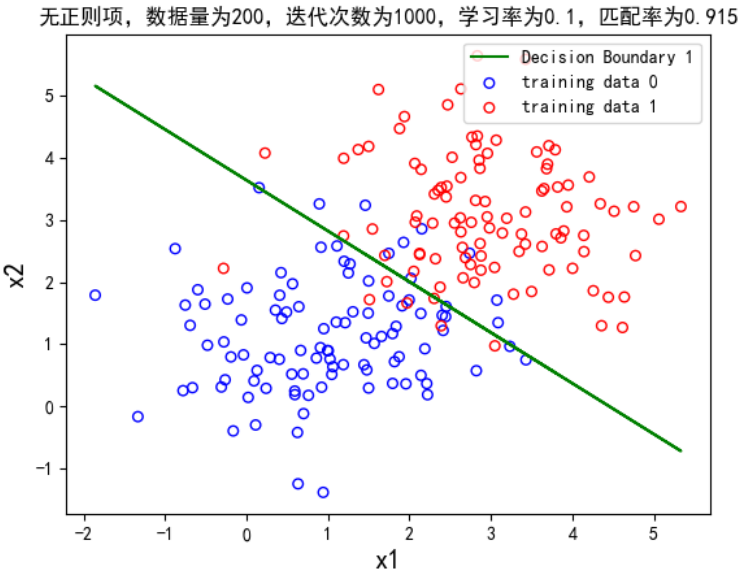
5.3 样本数量的影响

学习率 a=0.01，步长为 1000,不加正则项

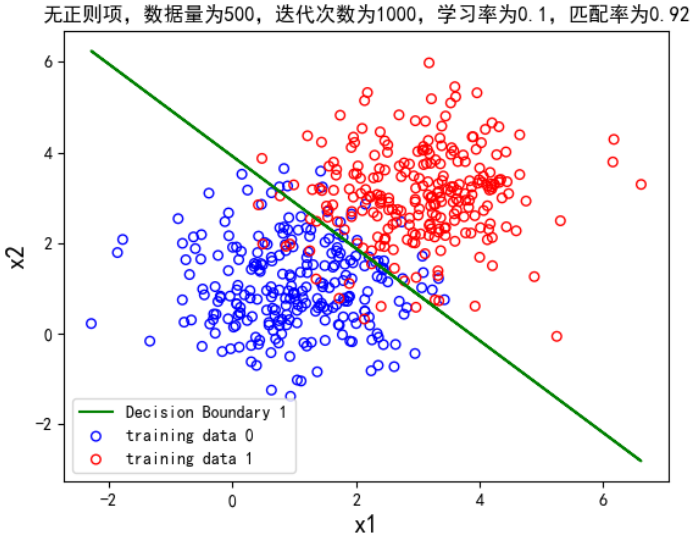
N=100：



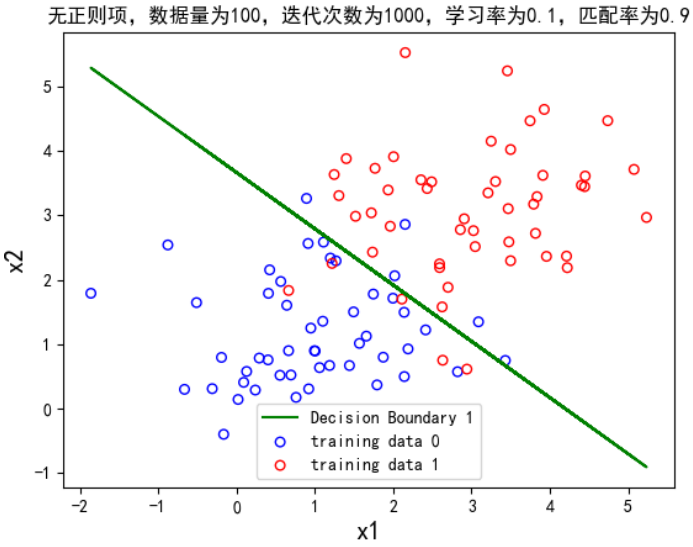
N=200：



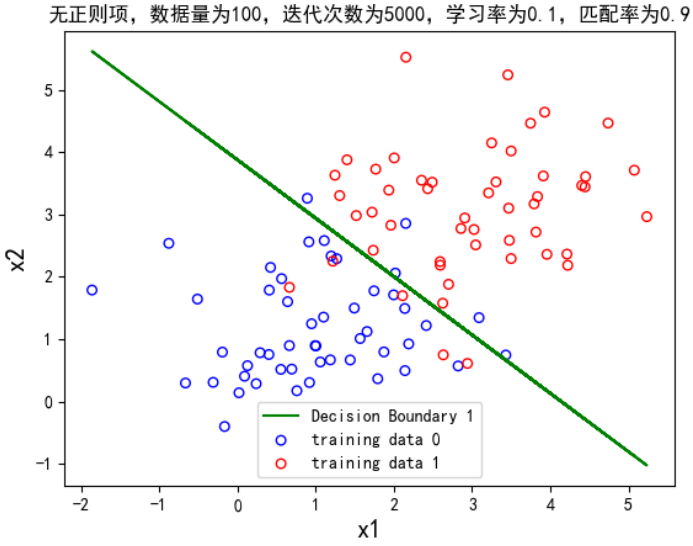
N=500：



N=1000：



N=5000：

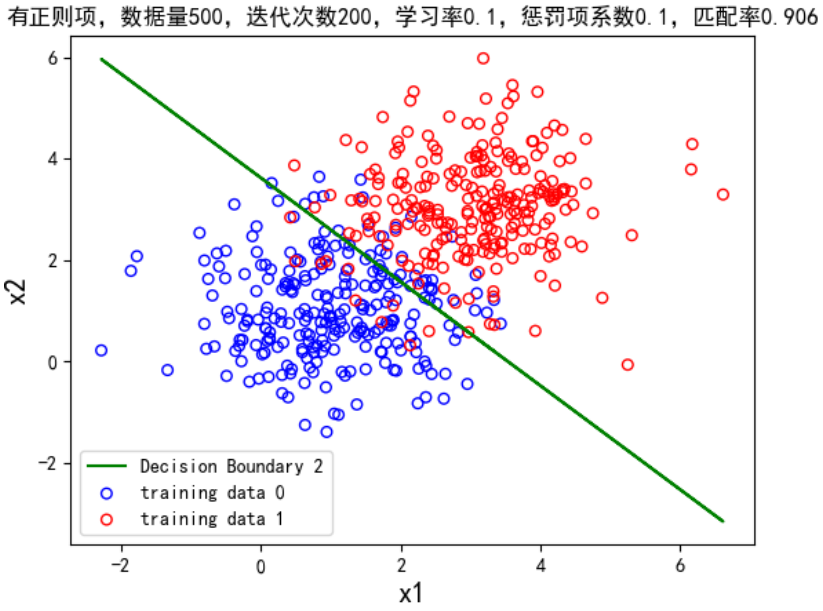


如图，其他条件不变的情况下，在一定范围内样本量越大匹配率越高（本处为200-1000），但是由于样本量和学习率等之间的相关关系，需要对别的参数进行调整才能达到最好的效果。

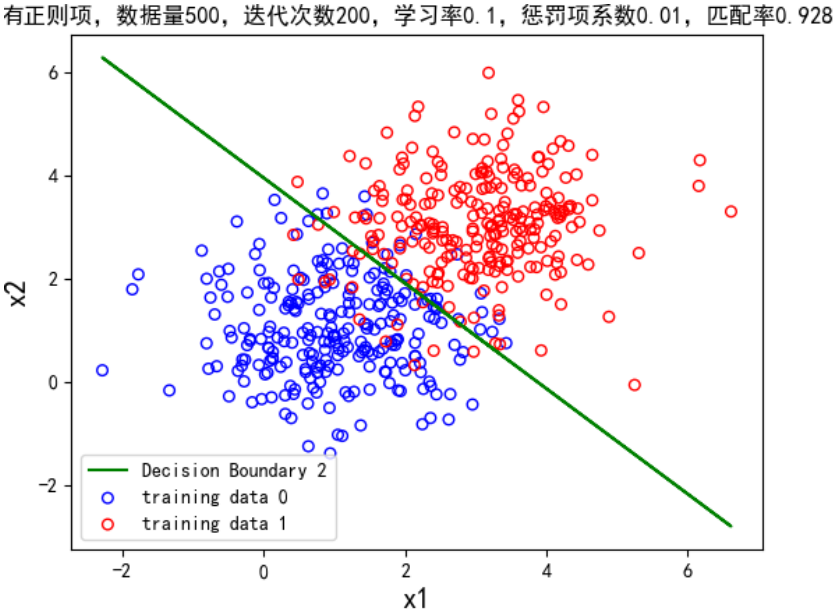
5.4 正则项系数的影响

数据量 N=500，学习率 a=0.1，步长为 200,加正则项

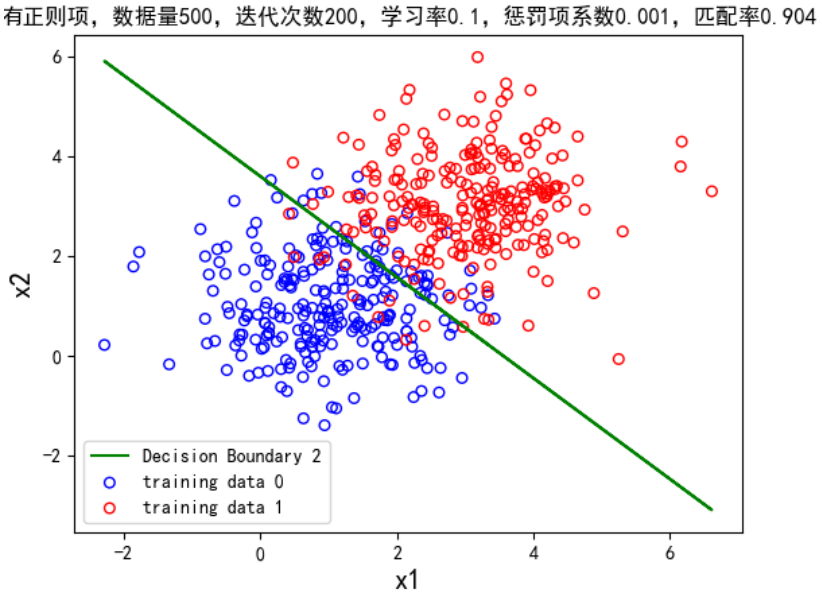
正则项系数$\lambda = 0.1$：



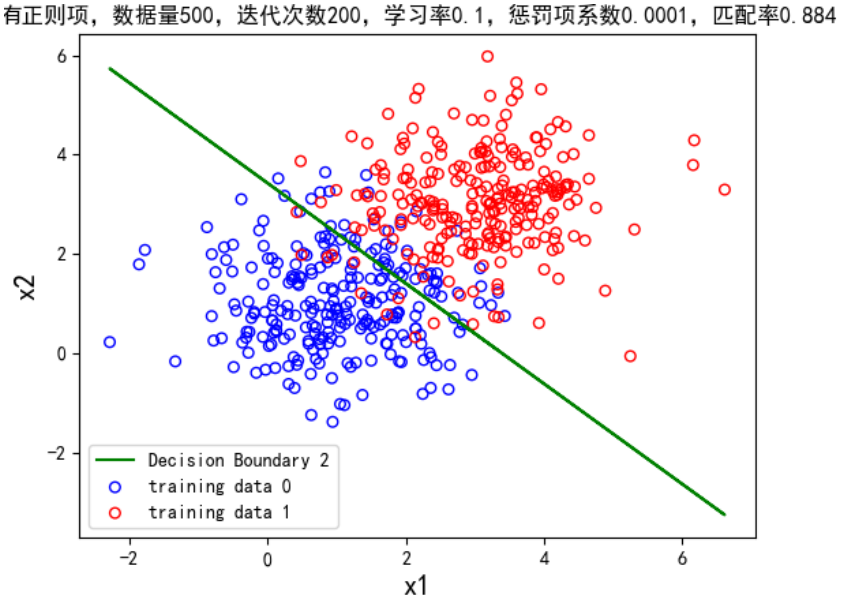
正则项系数$\lambda = 0.01$：



正则项系数$\lambda = 0.001$：



正则项系数$\lambda = 0.0001$：

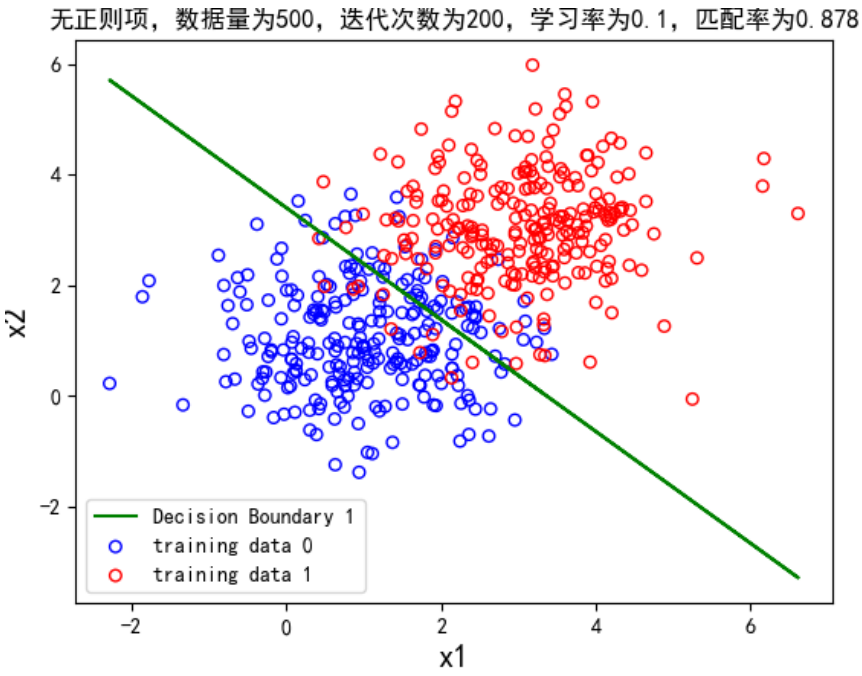


如图可得：当此情况其他条件不变时，λ在 0.01 左右时匹配率最好，过高过低都会导致匹配率下降。

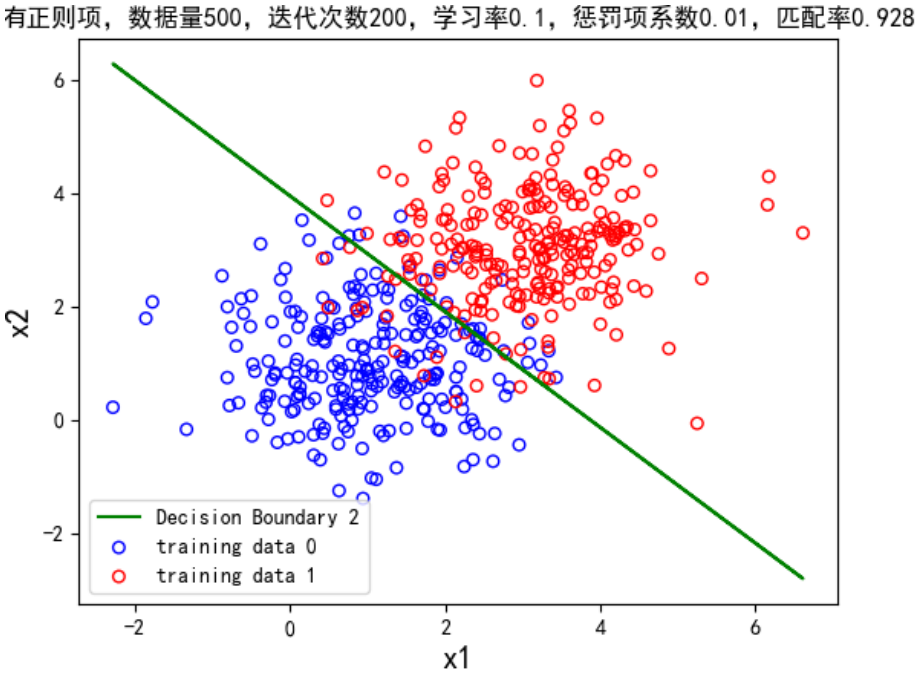
5.5 比较是否加正则项

数据量 N=500，学习率 a=0.1，步长 200，$\lambda = 0.01$

不加正则项：



加正则项：

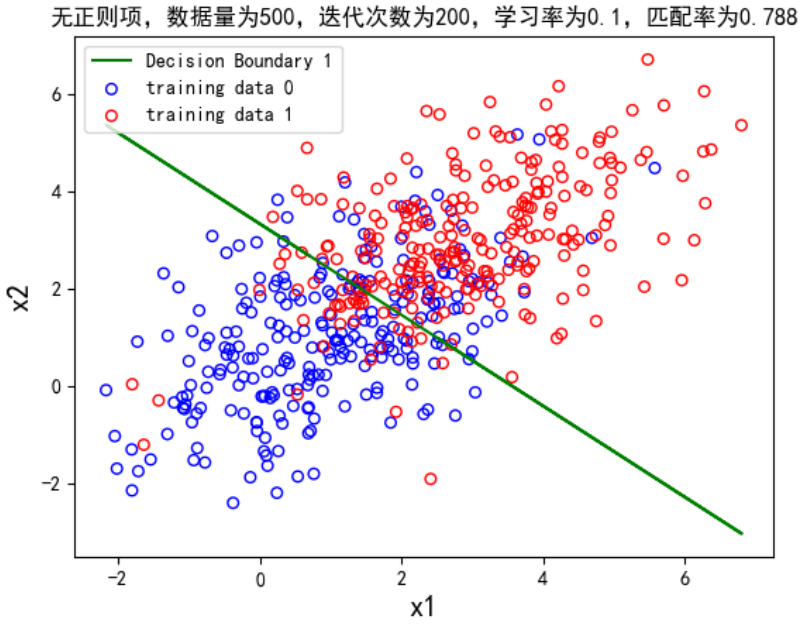


可见，加正则项有一定影响，此时使匹配率提高。但和之前其他情况比较发现，数据量和迭代次数较大的情况下，添加正则项的效果就很不明显了。

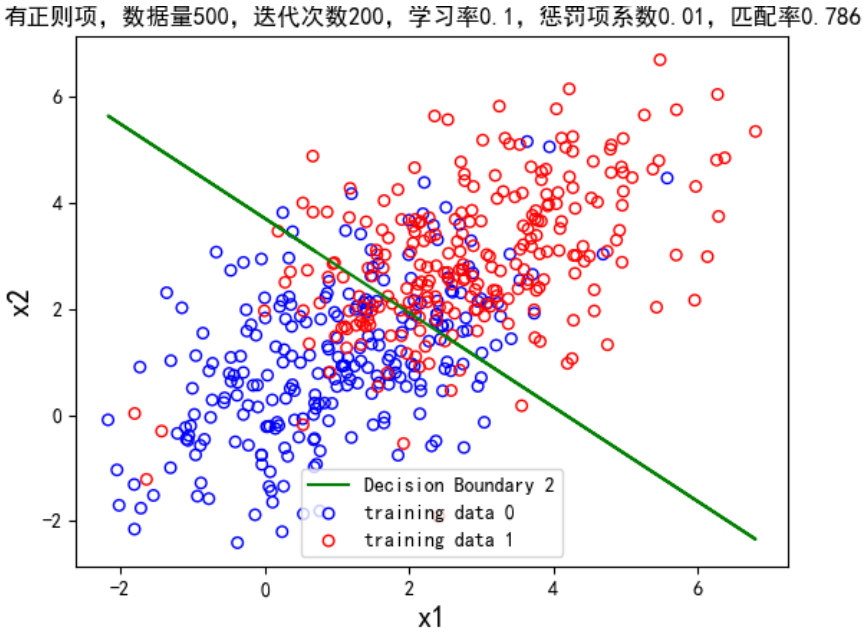
5.6 是否满足贝叶斯假设

不满足贝叶斯假设，数据量 N=500，学习率 a=0.1，步长 200，$\lambda = 0.01$

不加正则项：



加正则项：



可见，正则项对不满足匹配贝叶斯假设的情况效果也不明显。
而通过和 5.5 的两幅图像对比可知：满足贝叶斯假设的情况得到的匹配率更高，分类效果更好。

5.7 进行 UCI 测试

我找到的 UCI 数据集信息如下：

钞票认证数据集

下载: 数据文件夹, 数据集描述

摘要： 数据是从用于评估钞票认证程序的图像中提取的。

数据集特征:	多变量	**实例数:**	1372	**区:**	电脑
属性特征:	真实	**属性数量:**	5	**捐赠日期**	2013-04-16
相关任务:	分类	**缺少价值?**	不适用	**网络点击数:**	268229

资源：

数据库所有者：Volker Lohweg（应用科学大学，Ostwestfalen -Lippe，volker.lohweg '@' hs-owl.de）
数据库捐赠者：HeleneDörksen（应用科学大学，Ostwestfalen -Lippe，helene.doerksen ' @' hs-owl.de）
接收日期：2012年8月

属性信息：

1.小波变换图像的方差（连续）
2.小波变换图像的偏度（连续）
3.小波变换图像的偏斜（连续）
4.图像熵（连续）
5.类（整数）

测试结果：

UCI 1 为无正则项的梯度下降所得匹配率

UCI 2 为有正则项的梯度下降所得匹配率

```
Matching ratio UCI 1 =  0.8943148688046647
Matching ratio UCI 2 =  0.9314868804664723
```

还测试了几组别的数据，为防止溢出在较大维数和数据值较大的样本测试时，需对 loss 进行修正，放缩以避免溢出。

6 .实验结论

经验：

1. sigmoid 函数可能产生溢出或运行过长，进行了分类对过小的 sigmoid 取值直接取 0
2. 数据量过大的情况下也可能产生溢出，需对 loss 进行修正。

结论：

1. 类条件分布不满足朴素贝叶斯分布时分类效果不如满足朴素贝叶斯分布的情况
2. 逻辑回归模型中，采用梯度下降法时正则项的效果并不是很大，不如实验 1 明显
3. 迭代次数、学习率需要动态调整。
4. 这个实验出现了溢出情况，之前别的实验比较少见。