2020 年春季学期

计算学部《机器学习》课程

Lab 2 实验报告

姓名	梅智敏
学号	1183710118
班号	1837101
电子邮件	1044388658@qq.com
手机号码	13385658102

Logistic Regression解决二分类问题

一、实验目的

理解逻辑回归模型，掌握逻辑回归模型的参数估计算法。

二、实验要求及实验环境

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。

实验环境：

windows10、python3.7.4、pycharm2020.2

三、数学原理

3.1 实验目的及假设

实验目的：

从$TrainingSet$：$<X^1,Y^1>,<X^2,Y^2>\dots<X^l,Y^l>$中学习到一个分类器

$$f:X\rightarrow Y$$

以便于预测一个新的样本X^{new}所属的类别$label$

实验假设：

- X的每一维属性X_i都是实数，故X可视为形如$<X_1,X_2\dots X_n>$的n维$vector$
- Y是$boolean$值，取值为1或0
- X_i关于Y条件独立
- $P(X_i|Y=y_k)\sim N(\mu_{ik},\sigma_i)$
- $P(Y)\sim B(\pi)$

3.2 转化$P(Y|X)$

3.2.1 利用实验假设

按照前面的实验假设，结合概率论的知识，我们可以得到：

$$\begin{aligned}P(Y=1|X)&=\frac{P(Y=1)P(X|Y=1)}{P(Y=1)P(X|Y=1)+P(Y=0)P(X|Y=0)}\\&=\frac{1}{1+\frac{P(Y=0)P(X|Y=0)}{P(Y=1)P(X|Y=1)}}\\&=\frac{1}{1+exp(ln\frac{P(Y=0)P(X|Y=0)}{P(Y=1)P(X|Y=1)})}\\&=\frac{1}{1+exp(ln\frac{1-\pi}{\pi}+\sum_i ln\frac{P(X_i|Y=0)}{P(X_i|Y=1)})}\end{aligned}$$

由于$P(X_i|Y=y_k)=\frac{1}{\sigma_i\sqrt{2\pi}}exp(\frac{-(X_i-\mu_{ik})^2}{2\sigma_i^2})$

代回原来的式子，可得

$$P(Y=1|X)=\frac{1}{1+exp(w_0+\sum_{i=1}^{n}w_iX_i)}$$
$$w_0=\sum_i\frac{\mu_{i1}^2-\mu_{i0}^2}{2\sigma_i^2}+ln\frac{1-\pi}{\pi};w_i=\frac{\mu_{i0}-\mu_{i1}}{\sigma_i^2}$$

因此

$$P(Y=0|X)=\frac{exp(w_0+\sum_{i=1}^{n}w_iX_i)}{1+exp(w_0+\sum_{i=1}^{n}w_iX_i)}$$
$$ln\frac{P(Y=0|X)}{P(Y=1|X)}=ln(exp(w_0+\sum_{i=1}^{n}w_iX_i))=w_0+\sum_{i=1}^{n}w_iX_i$$

3.2.2 引入odds

这里使用了一个分类的思想：利用$odds$

> 一个事件的几率$odds$是指事件发生的概率与事件不发生的概率的比值，如果事件发生的概率是p，那么该事件的几率$odds=\frac{p}{1-p}$，该事件的对数概率（logit函数）就是

$$logit(p)=ln\frac{p}{1-p}$$

> 依据$logit(p)$>0还是<0，来判定事件发生还是不发生，这便是$odds$概念的作用

将其应用到我们的$LogisticRegression$问题中来便是

$$logit(Y=0|X)=w_0+\sum_{i=1}^{n}w_iX_i$$

若$logit(Y=0|X)>0$则将X分到$Y=0$类，若$logit(Y=0|X)<0$则将X分到$Y=1$类。

故我们的类别分界线就是

$$w_0+\sum_{i=1}^{n}w_iX_i=0$$

将其向量化

$$w^TX=0$$
$$w=[w_0,w_1\dots w_n],X=[1,X_1,X_2\dots X_n]$$

注意这里的w和X都是$n+1$维向量，拓展了一个维度。

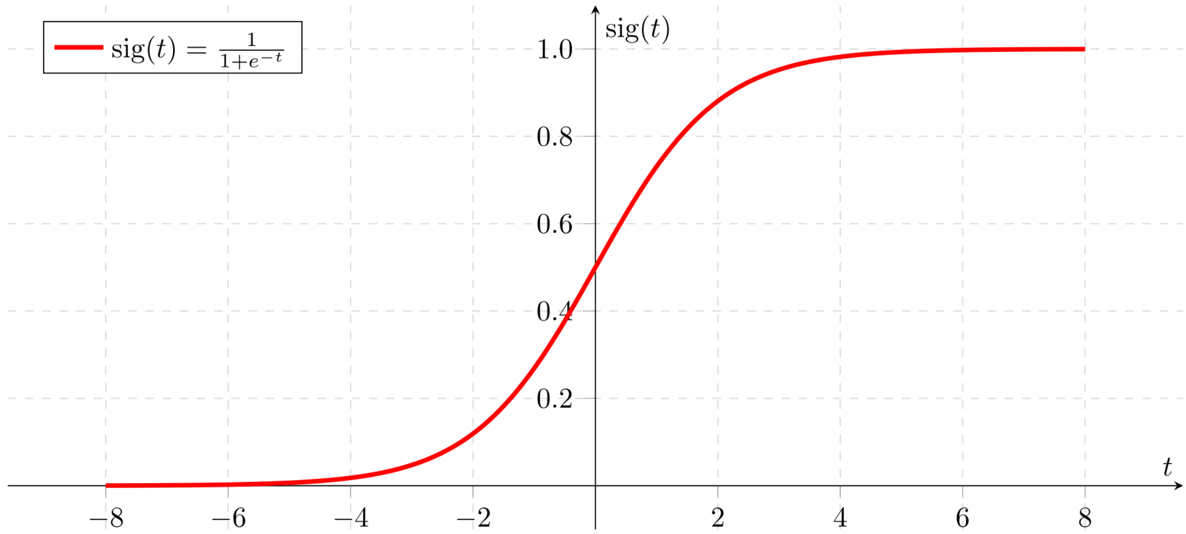
现在，还可以量化求出X属于$Y=1$类和$Y=0$类的概率

$$P(Y=1|X)=\frac{1}{1+exp(w^TX)}=sigmoid(-w^TX)$$

$$P(Y=0|X)=\frac{exp(w^TX)}{1+exp(w^TX)}=\frac{1}{1+exp(-w^TX)}=sigmoid(w^TX)$$

3.2.3 引入sigmoid函数

注意，这里我们引入了一个$sigmoid$函数，图像如下：



> 所谓sigmoid函数，是一个在生物学中常见的S型函数，也称为S型生长曲线；也常常运用于信息科学当中，由于其单增以及反函数单增等性质，Sigmoid函数常被用作**神经网络的激活函数**，将变量映射到0,1之间。
>
> 它是一个从实数域到(0,1)区间的映射，可以用来做二分类。在特征相差比较复杂或是相差不是特别大时效果比较好。Sigmoid作为激活函数有以下优点：
>
> 平滑、易于求导。

我们在这里利用$sigmoid(w^TX)$来表示$P(Y=0|X)$，既满足w^TX是在实数域上，又满足$sigmoid(w^TX)$是在$(0,1)$区间上，且该函数光滑可导，十分契合我们的需求。

3.2.4 总结

我们得到的X属于2种类别的分界线是

$$w^TX = 0 \tag{1}$$

当$w^TX>0$时认为X属于$Y=0$类；若$w^TX<0$则认为X属于$Y=1$类。

而把X归类为2种类别的概率分别是

$$P(Y=1|X) = sigmoid(-w^TX) \tag{2}$$

$$P(Y=0|X) = sigmoid(w^TX) \tag{3}$$

3.3 找到loss函数

前面我们已经得到了分类的界限$w^TX=0$，那么我们该如何确定这里的参数w呢？

有两种方法：最大似然估计**MLE**和贝叶斯估计**MAP**，两者在$loss$函数里面就分别代表了无正则项的loss函数和有正则项的loss函数。

3.3.1 用MCLE求解w

> MLE的核心思想就是：**将参数w看作唯一真值**，我们的任务就是找到这个w，使得在这组参数下，我们的数据的似然度（概率）最大。

也就是说我们需要求$P(<X,Y>|w)$，但这是很困难的事情，于是我们可以将**MLE**转换为**MCLE**，只需要计算 $P(Y|X，w)$

于是我们的似然函数就是

$$
\begin{aligned}
L(w) &= ln\prod_l P(Y^l|X^l,w) \\
&= \sum_l (Y^l w^T X^l - ln(1+exp(w^T X^l)))
\end{aligned}
$$

我们的损失函数$loss(w)$一般取$-L(w)$，即

$$
loss(w) = \sum_l (-Y^l w^T X^l + ln(1+exp(w^T X^l))) \tag{4}
$$

注意：这里的X^l和Y^l均表示第l个样本

3.3.2 用MAP求解w

> MAP的核心思想是：w是一个随机变量，符合一定的概率分布。
>
> 所以我们的任务就是给w添加一个先验$P(w)$，然后使得$P(w)P(Y|X,w)$最大。

我们假设$w_i \sim N(0,\sigma)$，则似然函数为

$$
L(w) = \sum_l (Y^l w^T X^l - ln(1+exp(w^T X^l))) - \frac{w^T w}{2\sigma^2} + ln(\frac{1}{\sqrt{2\pi}\sigma})
$$

简化为

$$
L(w) = \sum_l (Y^l w^T X^l - ln(1+exp(w^T X^l))) - \frac{\lambda}{2} w^T w
$$

则MAP情况下的$loss$函数为

$$
loss(w) = \sum_l (-Y^l w^T X^l + ln(1+exp(w^T X^l))) + \frac{\lambda}{2} w^T w \tag{5}
$$

相当于在MLE的基础上加了正则项

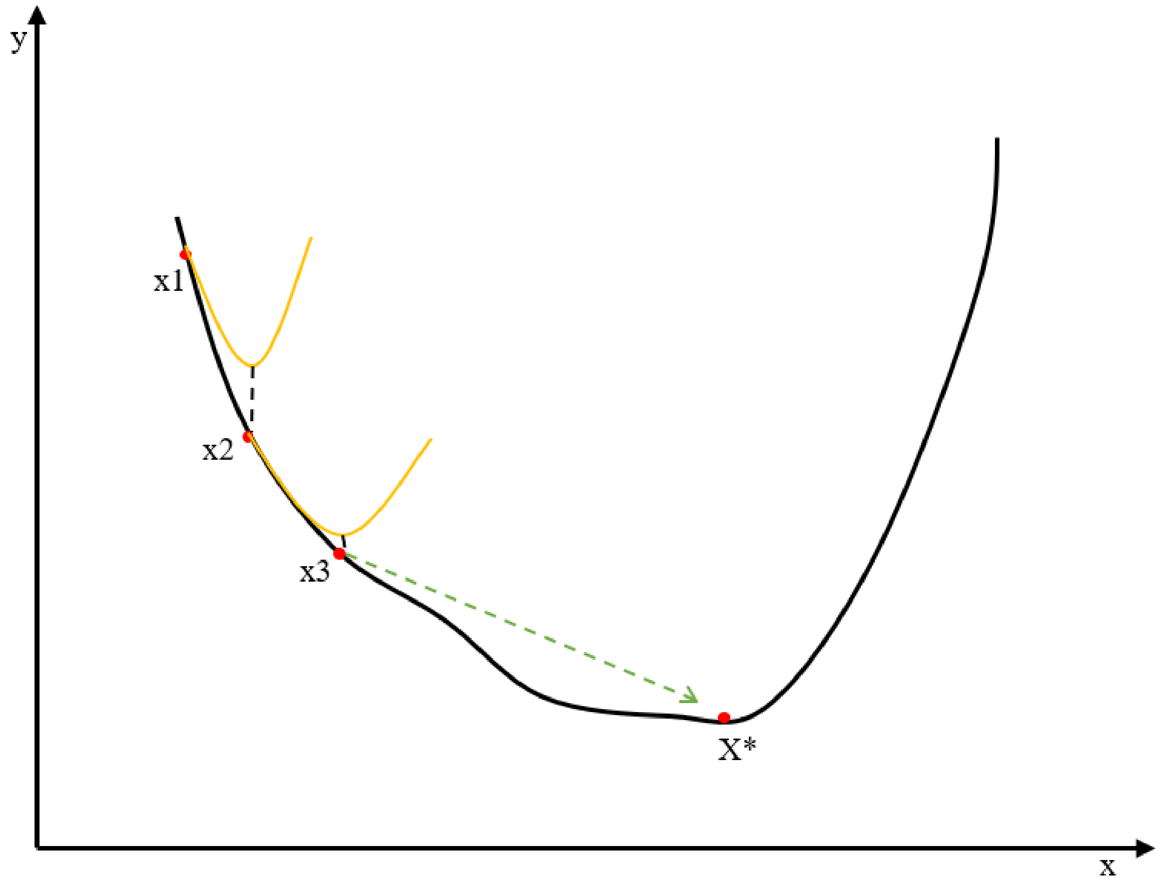
3.4 求出loss函数的优化解——牛顿法

前面我们已经找出了**MLE**和**MAP**情况下的$loss$函数，我们所需的w为

$$
w = \mathrm{argmin}_w \, loss(w) \tag{6}
$$

我们使用**牛顿法**来求解

> 牛顿法的思路是使用二阶泰勒展开去估计曲线，然后用二阶泰勒展开的函数的极值点去估计曲线的极值点，重复迭代直到找到极值点



对于无约束最优化问题

$$min_x f(x)$$

其中x^*为函数极小值点

设$f(x)$有二阶连续偏导数，若第k次迭代值为x^k，则可以将$f(x)$在x^k附近进行**二阶泰勒展开**

$$f(x) = f(x^k) + g_k^T(x - x^k) + \frac{1}{2}(x - x^k)^T H(x^k)(x - x^k) \tag{7}$$

其中，$g_k = \bigtriangledown f(x)$是梯度向量，$H(x)$是海森矩阵

$$H(x) = [\frac{\partial^2 f}{\partial x_i \partial x_j}]_{n\times n}$$

$f(x)$在极值点处$g_k = 0$；特别的，当$H(x)$为正定矩阵时，$f(x)$的极值是极小值。

为了得到$g_k = 0$的点，对7式求导

$$\frac{df(x)}{dx} = g_k + H_k(x - x^k)$$

则极值点处$\frac{df(x)}{dx} = 0$

$$x^{k+1} = x^k - H_k^{-1} g_k$$

这便是我们的迭代公式。

将其应用到我们的$loss$函数就是

$$w^{k+1} = w^k - (\frac{\partial^2 loss(w)}{\partial w \partial w^T})^{-1} \frac{\partial loss(w)}{\partial w} \tag{8}$$

其中

$$\frac{\partial loss(w)}{\partial w} = -\sum_l x^l(Y^l - sigmoid(w^TX)) + \lambda w \tag{9}$$

$$\frac{\partial^2 loss(w)}{\partial w \partial w^T} = \sum_l (XX^T sigmoid(w^TX) sigmoid(-w^TX)) + \lambda I \tag{10}$$

注意：上面的式子对于MLE的$loss$函数而言λ=0，I表示单位阵

四、实验具体流程

4.1 生成数据

设置正例($Y=1$)的比例为40%，训练集、验证集、测试集的比例为6：2：2

利用多维高斯分布函数来生成数据，为便于画图展示，主要使用二维数据。

4.1.1 满足朴素贝叶斯

若满足朴素贝叶斯，则认为X的各维度数据关于Y条件独立，则协方差矩阵为

$$C = \begin{bmatrix} cov_{11} & cov_{12} \\ cov_{21} & cov_{22} \end{bmatrix}$$

其中$cov_{12}=cov_{21}=0$，cov_{11}就是X_1的方差，cov_{22}就是X_2的方差，故

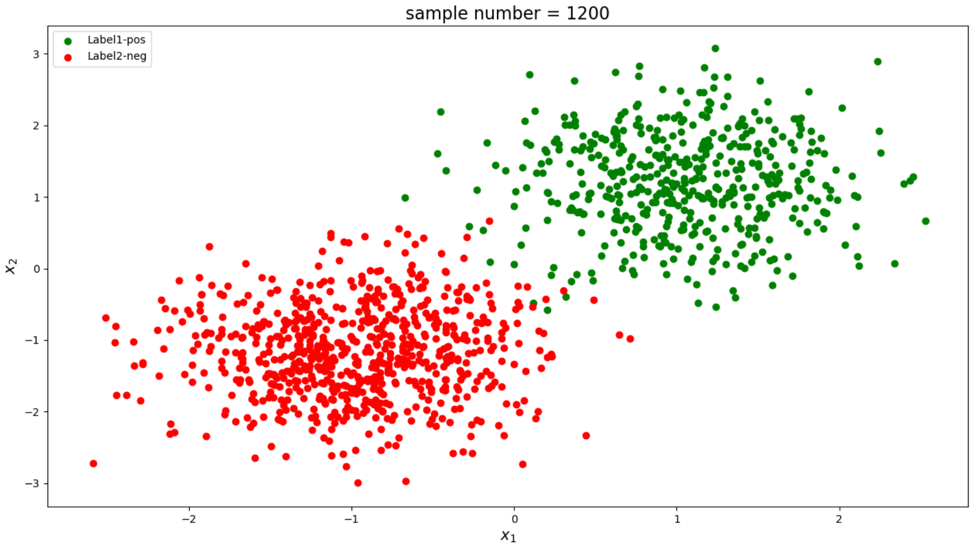
$$C = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}$$

自己设定$\sigma_1^2=0.3, \sigma_2^2=0.4$

展示核心代码：

```
    def Data(N, naive=True, posRate=0.4):
        posNumber = np.ceil(N * posRate).astype(np.int32)
        sigma = [0.3, 0.4]  # cov11与cov22
        cov12 = 0.2
        pos_mean = [1, 1.2]  # 正例的两维度均值
        neg_mean = [-1, -1.2]  # 反例的两维度均值
        x = np.zeros((N, 2))  # x数组
        y = np.zeros(N).astype(np.int32)  # label数组
        if naive:  # 满足朴素贝叶斯假设
            x[:posNumber, :] = np.random.multivariate_normal(pos_mean,
[[sigma[0], 0], [0, sigma[1]]],
                                                              size=posNumber)
            x[posNumber:, :] = np.random.multivariate_normal(neg_mean,
[[sigma[0], 0], [0, sigma[1]]],
                                                              size=N - posNumber)

            y[:posNumber] = 1
            y[posNumber:] = 0
```

实验效果如下



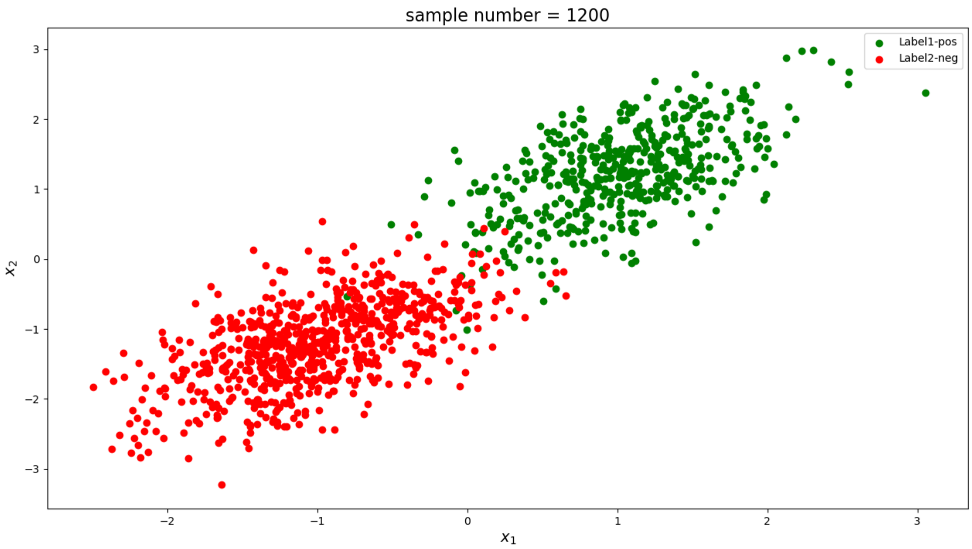
4.1.2 不满足朴素贝叶斯

不满足朴素贝叶斯时，则$cov_{12} \neq 0$，自己设定$cov_{12} = 0.2$

核心代码如下：

```
    else:  # 不满足朴素贝叶斯假设
        x[:posNumber, :] = np.random.multivariate_normal(pos_mean, [[sigma[0], cov12], [cov12, sigma[1]]],
                                                          size=posNumber)
        x[posNumber:, :] = np.random.multivariate_normal(neg_mean, [[sigma[0], cov12], [cov12, sigma[1]]],
                                                          size=N - posNumber)
        y[:posNumber] = 1
        y[posNumber:] = 0
```

效果如下



可以明显发现：不满足朴素贝叶斯假设时，数据点呈现"长条"状，这表明X的2个维度之间有线性相关关系，与我们的预期想契合。

4.2 有无正则项的对比

在无正则项的时候，依据lab1的结论，当训练集数据点数据很少时，会有过拟合现象。然后克服过拟合的方法有2种：

- 增加训练集的样本数量
- 增加正则项

下面我们按照这个思路来进行有无正则项的对比实验

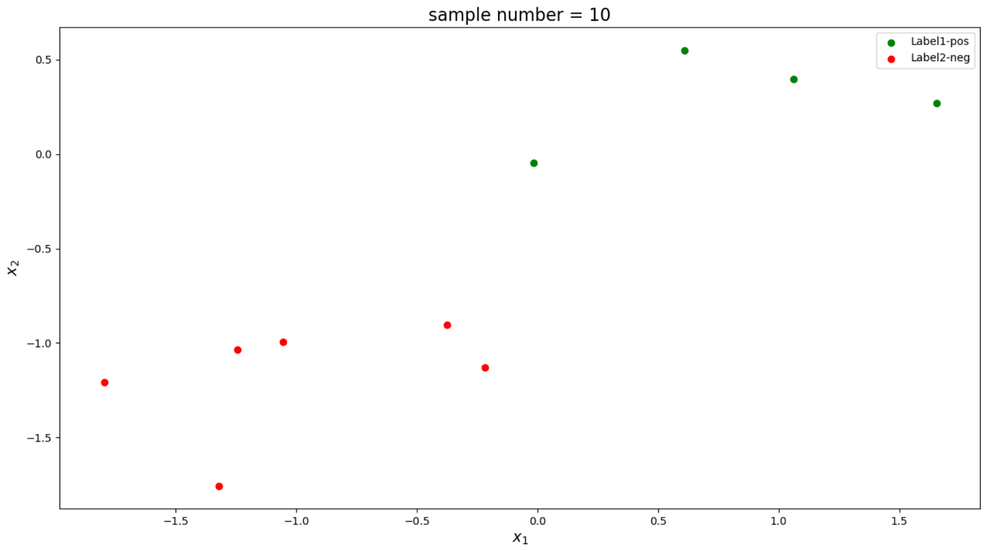
先展示一下Newton法求优化解的核心代码，至于算法的数学原理前面已经给出，这里不再赘述

```
    def __derivative(self, w):
        """
        求出导函数
        :param w: 当前的w
        :return: 返回当前的导数
        """
        result = np.zeros(self.__n)
        # 依次取出X和Y的所有行
        for i in range(self.__m):
            result += (self.x[i] * (self.y[i] -
                                    (1.0 - self.__sigmoid(w @ self.x[i]))))
        return -1 * result + self.hyper * w

    def __second_derivative(self, w):
        """
        求出hessian matrix的逆矩阵
        :param w:
        :return:
        """
        ans = np.eye(self.__n) * self.hyper
        # 依次取出X和Y的所有行
        for i in range(self.__m):
            temp = self.__sigmoid(w @ self.x[i])
            ans += self.x[i] * np.transpose([self.x[i]]) * temp * (1 - temp)
        # 最后求逆矩阵
        return np.linalg.pinv(ans)

    def solve(self):
        w = self.w_0
        while True:
            gradient = self.__derivative(w)
            # 满足精度要求即可退出迭代
            if np.linalg.norm(gradient) < self.delta:
                break
            # 使用迭代公式进行下一次迭代
            w = w - self.__second_derivative(w) @ gradient
        return w
```

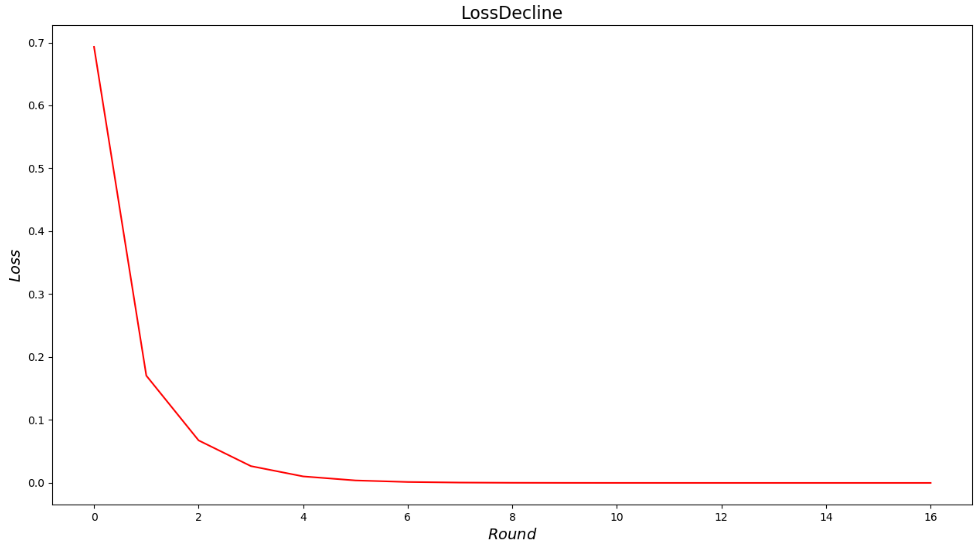
首先使用样本数量很少的训练集来训练，然后将训练得到的结果w应用到一个较大的测试集上测试它的泛化性能

4.2.1 无正则

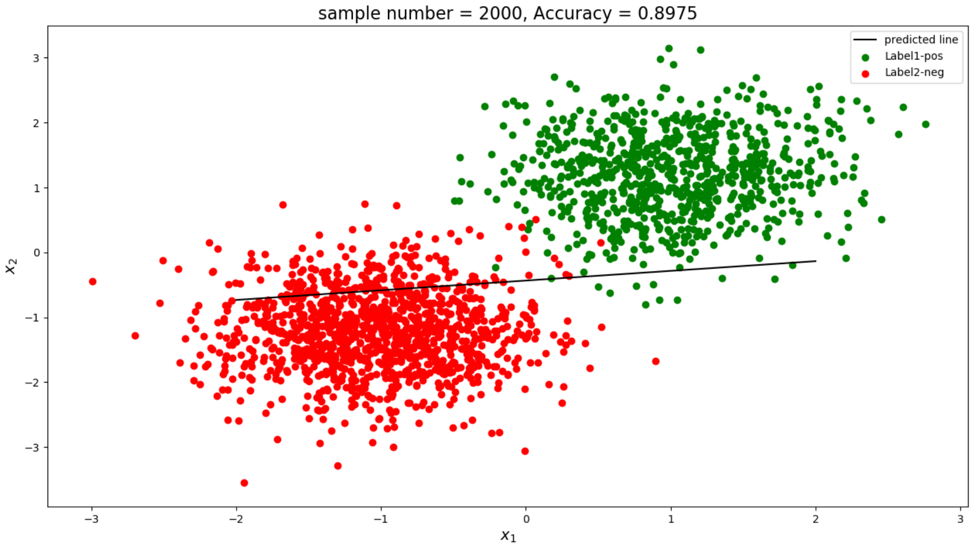
我们仅仅使用$N=10$的**小训练集**:



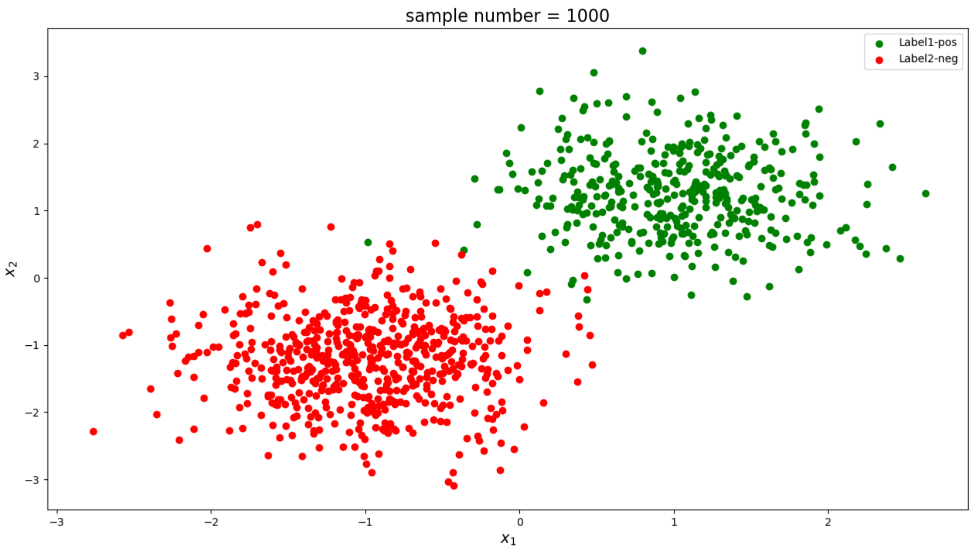
$Newton$法迭代情况如下，可见$Newton$法收敛还是比较快的:



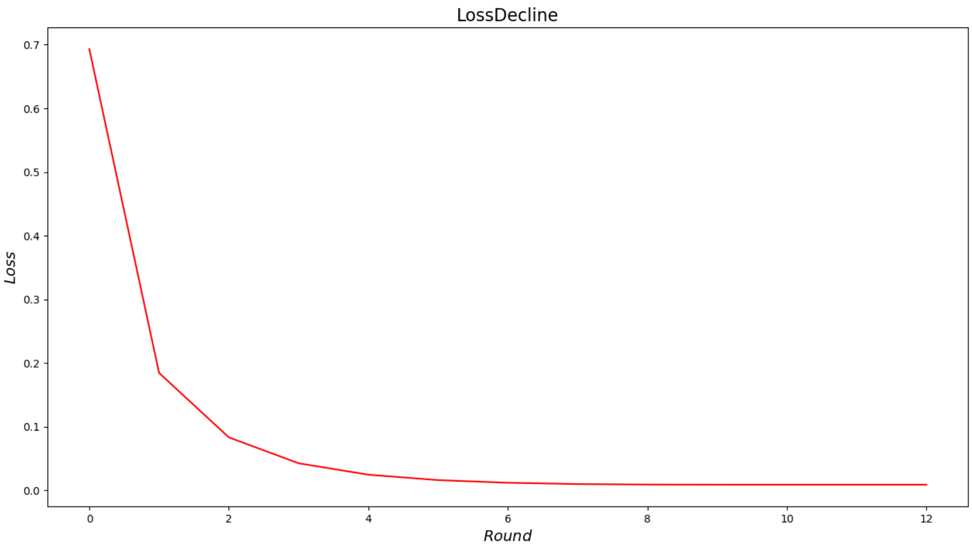
使用$N=2000$的测试集来测试它的泛化性能，正确率为89.75%:



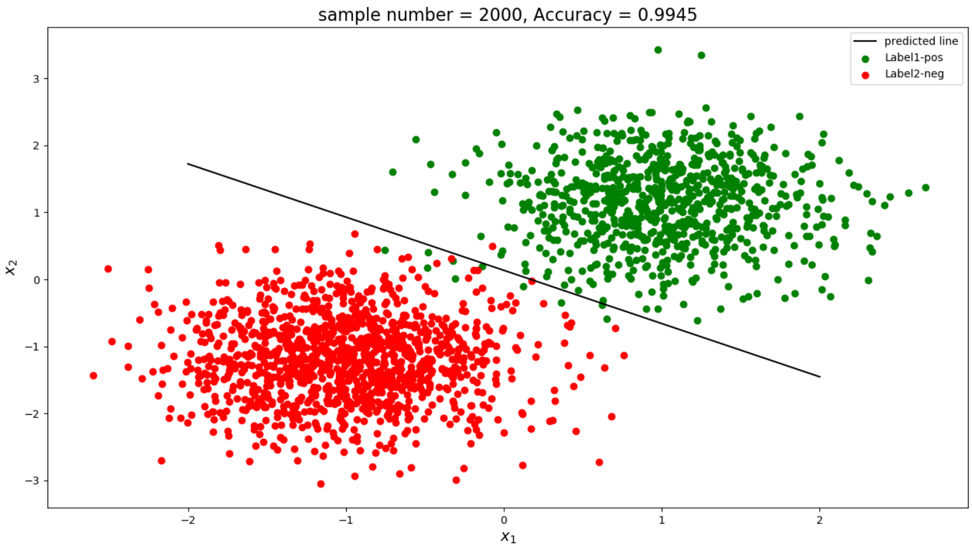
接下来**使用更大的训练集**$N = 1000$：



对应的收敛情况如下，可见在样本数较大时，迭代轮数并未受到太大影响。

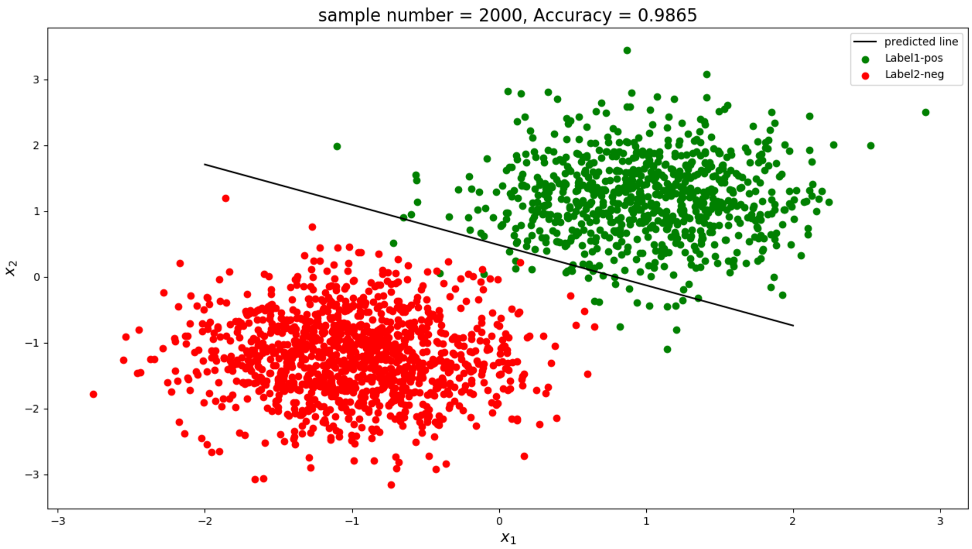


但是大训练集可以有效克服过拟合，此处准确率为99.45%



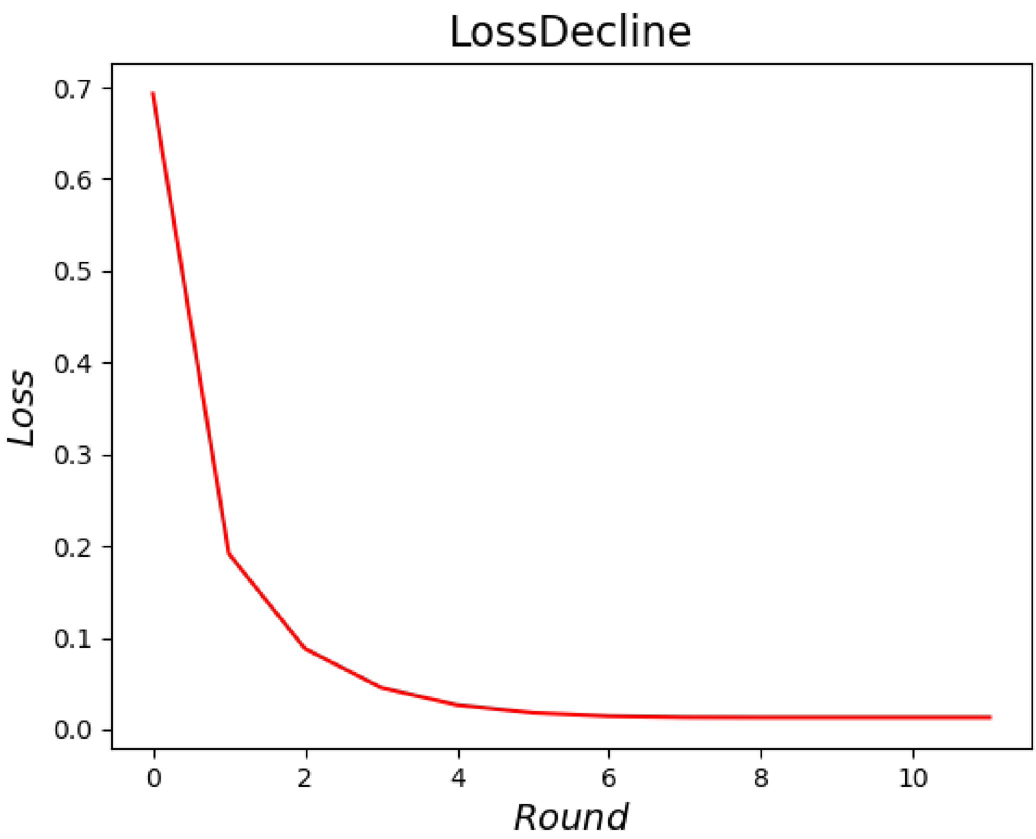
4.2.2 有正则

使用一样的$N=10$的训练集，但是这次加上正则项，并先设定$\lambda=0.1$，正确率为98.65%：

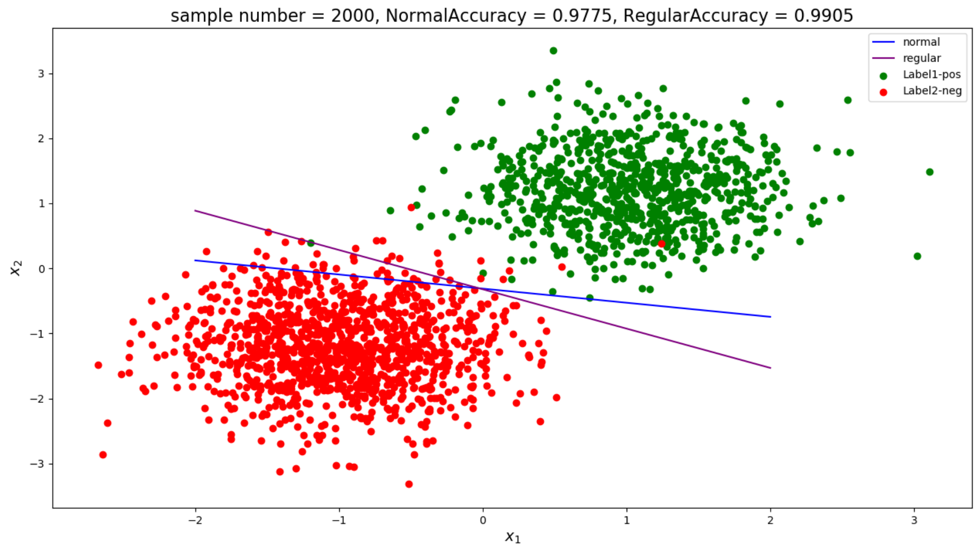


正确率大幅提高，这也与再一次证明了加入正则项可以克服过拟合的结论

收敛情况如下，可见增加正则项也不会明显改变迭代轮数：



至于正则项的直观作用，可以参看下图

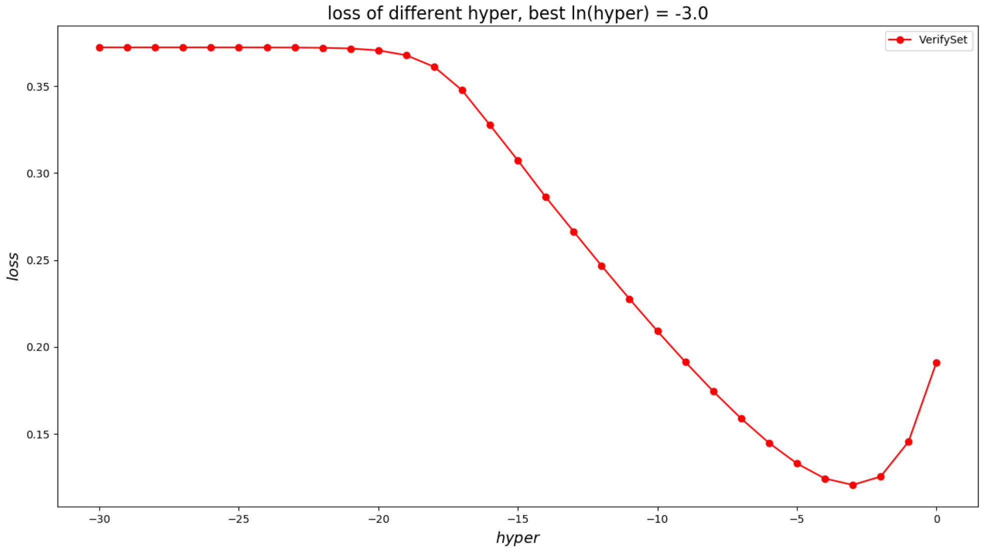


即：**正则项会改变求得的w，从而导致分界线的斜率和截距变化，具有更好的泛化性能。**

至于正则项的超参数λ，我通过给它设定一个范围，**然后在验证集上进行测试**，选择泛化性能最好的超参数

$$\lambda = exp(-3)$$

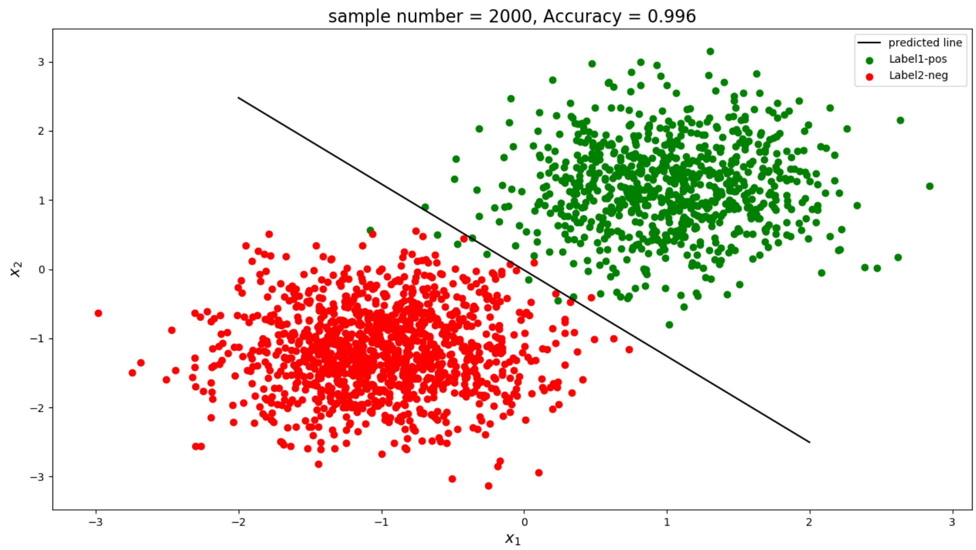
并在后续实验中均使用此值



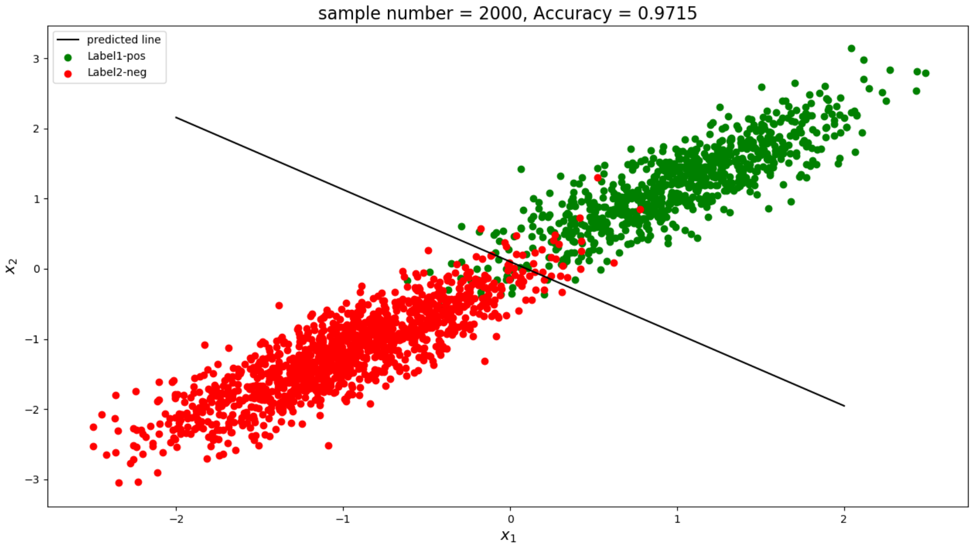
4.3 是否满足朴素贝叶斯的对比

在数学原理部分已经叙述过，不满足朴素贝叶斯假设时协方差矩阵就不是对角阵，下面皆使用 $N = 1000, \lambda = exp(-3)$进行实验

满足朴素贝叶斯时:



不满足朴素贝叶斯时，我们使用$cov_{12} = 0.2$进行测试：



分别进行10次实验，记录各自准确率：

	1	2	3	4	5	平均
满足	0.996	0.994	0.995	0.994	0.992	0.994
不满足	0.972	0.976	0.974	0.973	0.968	0.973

可见，在其他条件相同时，**"不满足朴素贝叶斯"的准确率略低于"满足朴素贝叶斯"**

原因就在于：我们实验使用的是Logistic Regression，得到的分类器$w^T X = 0$是个线性分类器，它只是给X的每个维度加个权重w_i，并没有考虑到各个维度之间的相关性，即默认满足了朴素贝叶斯假设。

4.4 使用UCI数据集进行测试

4.4.1 选用Skin数据集

> Skin数据集：通过从各种年龄组（年轻人，中年人和老年人），种族组（白人，黑人和亚洲人）的面部图像中随机抽取B，G，R值以及从FERET数据库和PAL数据库获得的性别来收集皮肤数据集。
>
> 学习样本总量为245057; 其中50859是皮肤样本，194198是非皮肤样本。

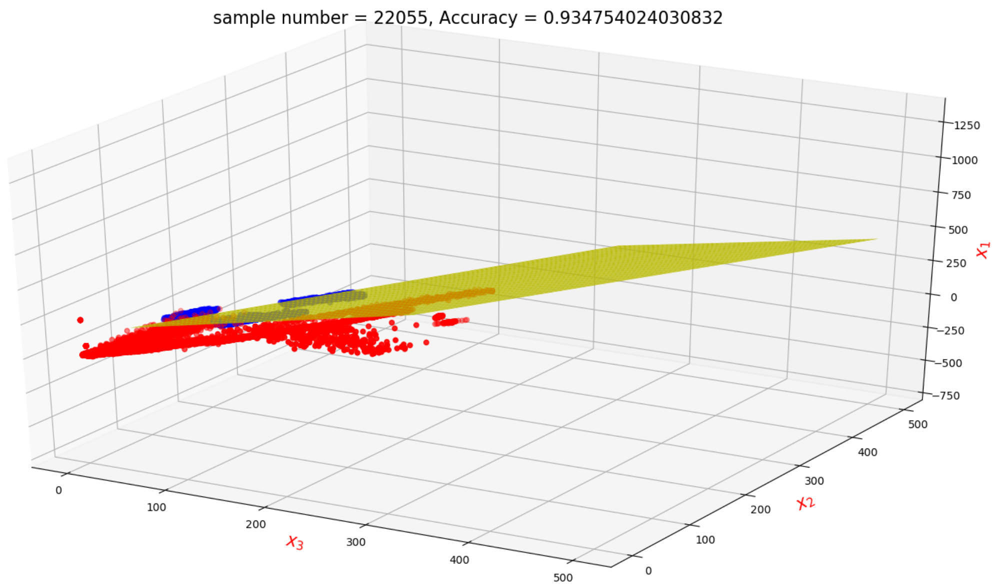
此数据集中X有3个维度，Y有2种取值。正好适合我们进行二分类任务，同时由于X有3个维度，因此可以将其在3D图种显示出来。

从UCI上获取的数据集首先要读取出X和Y

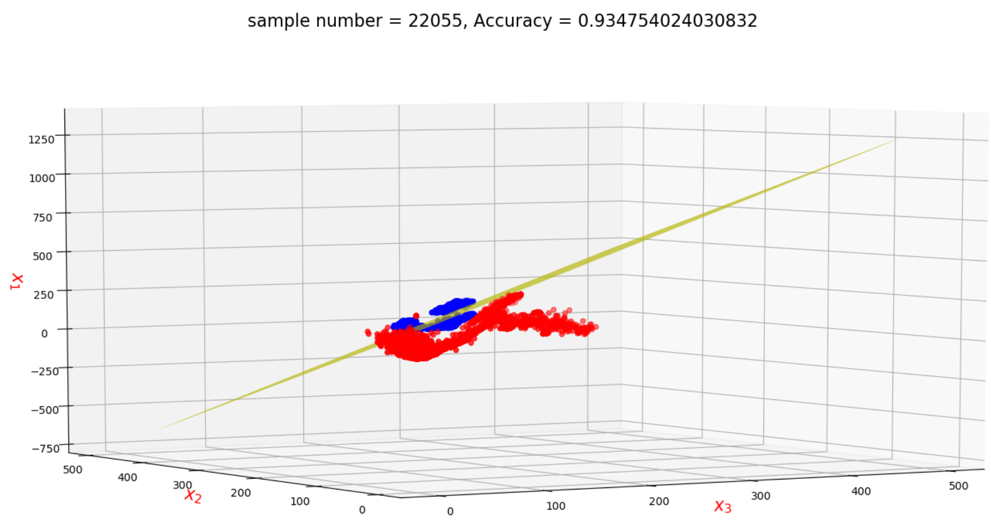
```
def ReadUCI(path):
    data_set = pd.read_csv(path)  # linux 相对路径
    x = data_set.drop('label', axis=1)
    y = data_set['label']
    dataX, dataY = np.array(x, dtype=float), np.array(y)
    N = len(dataY)
    posNumber = 0
    for i in range(N):
        if dataY[i] == 1:
```

```
            posNumber += 1

    # 调节读取的数据量
    useRate = 0.5
    posUseNumber = int(math.ceil(posNumber * useRate))
    negUseNumber = int(math.ceil((N-posNumber) * useRate))

    x_Bool = np.zeros(N).astype(np.int32)  # x对应的bool数组，用于切片
    x_Bool[:posUseNumber] = 1
    x_Bool[posNumber:(posNumber + negUseNumber)] = 1
    dataX, dataY = dataX[x_Bool == 1], dataY[x_Bool == 1]

    # 划分训练集和测试集
    Train_x, Train_y, Test_x, Test_y = SplitData(dataX, dataY)
    return Train_x, Train_y, Test_x, Test_y
```

无正则项时：

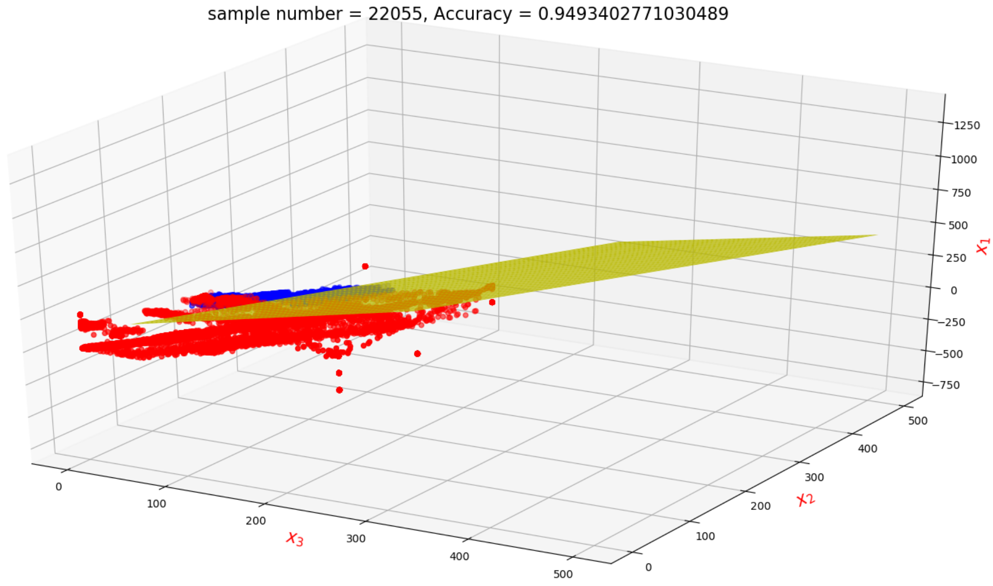


因为从正面看分界不是很清晰，所以我们找到分界的视角再看一下

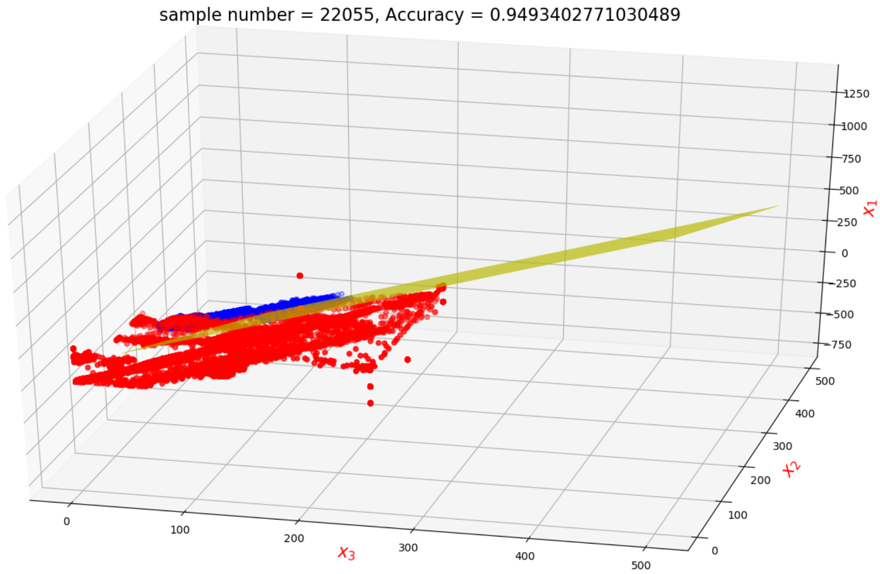


可见在不加正则项的时候，准确率也还不错，达到93.48%

加正则项时：



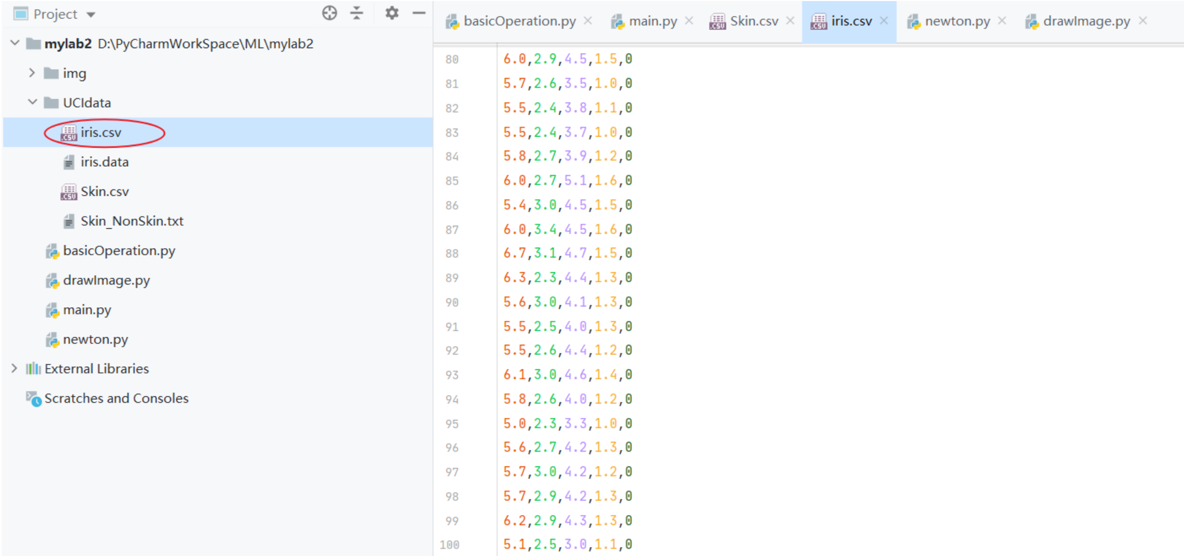
同样展示分界面：



可见：加入正则项会使得准确率略微上升，达到了95%

选用iris数据集

> iris数据集中每个样本中X有4个维度，Y有3种取值。
>
> 于是我先将数据集进行处理，只留下Y有2种取值的那部分，以便于我们直接进行二分类。处理之后，数据集中共有100个样本点。



在此数据集上进行测试：

```
main × basicOperation ×
E:\Anaconda3\python.exe D:/PyCharmWorkSpace/ML/mylab2/main.py
训练样本数为：70
测试样本数为：30
w为[-11.1499584   -0.20823898  14.65407722  -7.60462384 -14.77399154]
准确率为：1.0

Process finished with exit code 0
```

可见：准确率为100%

五、结论

- 对于在训练集样本数很少时，加入正则项可以有效解决过拟合问题。
- 类条件分布在满足朴素贝叶斯假设时的Logistic Regression分类表现，要比不满足假设时略好
- Logistics Regression可以很好地解决简单的线性分类问题
- 使用牛顿法求优化解时，收敛速度较快，且样本点数目对它的收敛速度影响不大。

六、参考文献

- 周志华 著. 机器学习, 北京: 清华大学出版社, 2016.1
- 李航 著. 统计学习方法, 北京: 清华大学出版社, 2019.5