基于 Logistic Regression 的分类器设计

经过一单元的模式识别课程的学习，我对最小二乘、梯度下降、PCA、SVD 分解、BP 神经网络、Logistic Regression（LR）等有了一定地了解，其中对 LR 算法比较热衷，课下花费了大量时间进行查阅资料，所以此次大作业我根据 LR 算法进行了多个分类器设计，包括二分类和多分类，其中考试录取预测问题的识别率达到了 89%，让我对模式识别这门课有了更深刻的认识。

下面将结合项目对基于 Logistic Regression 的分类器设计进行详细叙述。

一、理论概述

Logistic Regression, 中文有人译作“逻辑斯蒂回归”, 也有译作“对数几率回归”。后者具有更深厚理论理解，稍后会对这个名字的来源加以介绍。这里需要首先指明的是 Logistic 回归模型是一个分类模型而不是一个字面上的回归模型!

首先回顾一下线性回归模型:

$$f(x) = w_0 x_0 + \ w_1 x_1 + \ \cdots + w_n x_n + b \tag{1}$$

写成向量形式:

$$f(x) = \boldsymbol{w^T x + b} \tag{2}$$

写成广义线性回归模型:

$$y = g^{-1}(\boldsymbol{w^T x + b}) \tag{3}$$

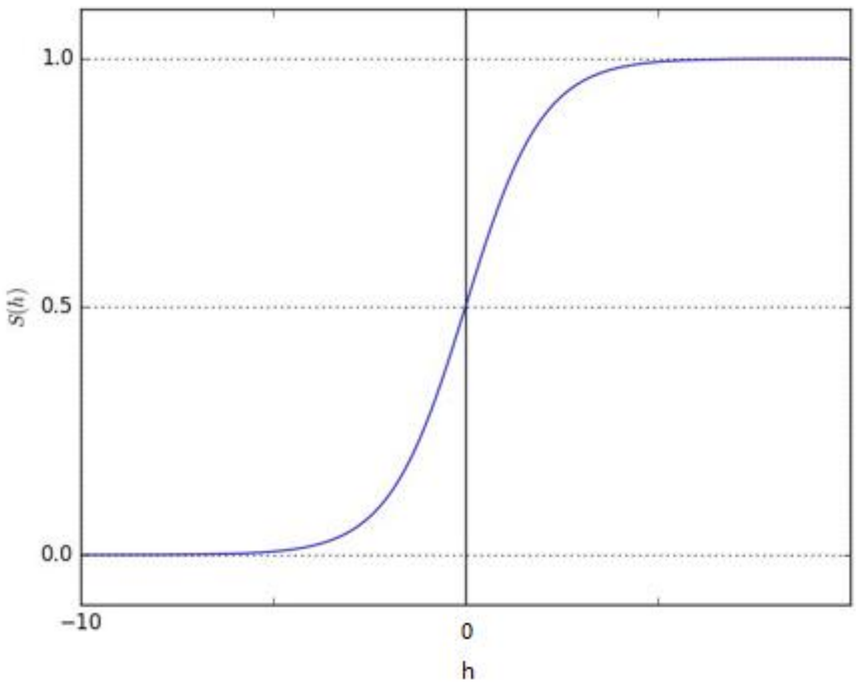
下面介绍从线性回归模型引出 logistic 回归的分类模型。线性回归模型只能够进行回归学习，但是若要是做分类任务则需要在“广义线性回归”模型中：只需找一个单调可微函数将分类任务的真实标记 y 与线性回归模型的预测值联系起来便

可以了！

logistic 回归是处理二分类问题的，所以输出的标记 y={0,1}，并且线性回归模型产生的预测值 $\boldsymbol{z} = \boldsymbol{w}\boldsymbol{x} + \boldsymbol{b}$ 是一个实值，所以需要将实值 z 转化成 0/1 值。这就可以选择映射函数来实现。“sigmoid”函数就是一个很好的备选项：

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

该函数的图像如下所示：



由图像可知，当预测值大于 0 便可判断为正例，小于 0 则判断为反例。这样我们在原来的线性回归模型外套上 sigmoid 函数便形成了 logistic 回归模型的预测函数，可以用于二分类问题：

$$y = \frac{1}{1 + e^{-(w^T x + b)}} \tag{5}$$

对上式的预测函数做一个变换，即得：

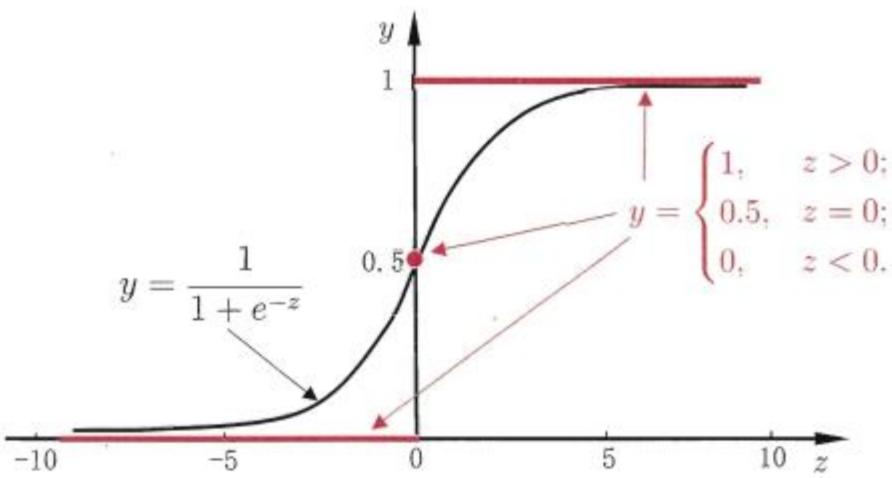
$$ln\frac{y}{1-y} = w^T x + b \tag{6}$$

若将 y 视作样本 x 作为正例的可能性，则 1-y 是其反例可能性，二者比值就 i 叫做“几率”（odds），反映了 x 作为正例的相对可能性。对几率取对数即得式（6）左侧部分，叫做“对数几率”（log odds，亦叫做 logit）。这就是“对数几率回归”名称的来源。

其实，最理想的是“单位阶跃函数”（***unit-step function***）：

$$y = \begin{cases} 0, & z < 0 \\ 0.5, & z = 0 \\ 1, & z > 0 \end{cases} \tag{7}$$

即若预测值 z 大于 0 就判定为正例，小于零则判为反例，临界值 0 则可判定为任意，如图示意：



二、参数求解

下面开始介绍如何求解模型：式（5）中的未知参数。

2.1 构造预测函数

若将式（5）中的 y 看作类后验概率估计 $p(y=1 \mid x)$，则式（6）可重写为：

$$ln\frac{p(y=1 \mid x)}{p(y=0 \mid x)} = \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + \boldsymbol{b} \tag{8}$$

显然有：

$$p(y=1|x) = \frac{e^{w^Tx+b}}{1+e^{w^Tx+b}} \tag{9}$$

$$p(y=0|x) = \frac{1}{1+e^{w^Tx+b}} \tag{10}$$

为了便于讨论，令 $\boldsymbol{\theta} = (\boldsymbol{w}, \boldsymbol{b})$，则$\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + \boldsymbol{b} = \boldsymbol{\theta}^{\boldsymbol{T}}\boldsymbol{x}$，其中 *x*=(*x*; 1)

构造预测函数为：

$$\hbar_\theta(x) = g(\theta^T x) = \frac{1}{1+e^{\theta^T x}} \tag{11}$$

则有：

$$P(y=1|x;\theta) = h_\theta(x) \tag{12}$$

$$P(y=0|x;\theta) = 1 - h_\theta(x) \tag{13}$$

2.2 构造目标函数

通过最大似然估计，给定数据集{x, y}，式（12）（13）可合写为：

$$p(y|x;\theta) = [h_\theta(x)]^y \times [1-h_\theta(x)]^{1-y} \tag{14}$$

取似然函数：

$$L(\theta) = \prod p(y|x;\theta) = \prod [h_\theta(x)]^y \times [1-h_\theta(x)]^{1-y} \tag{15}$$

对数似然函数为：

$$l(\theta) = logL(\theta) = \sum \left(ylogh_\theta(x) + (1-y)\log\left(1-h_\theta(x)\right)\right) \tag{16}$$

最大似然估计就是要求得使 $l(\theta)$ 取最大值时的θ，这里可以使用梯度上升法求解，求得的θ就是要求的最佳参数。但是当我们做一下变换的话，则有，

目标函数：

$$J(\theta) = -\frac{1}{m} l(\theta) \tag{17}$$

此时，$l(\theta)$ 乘了一个负系数-1/m，所以 $J(\theta)$ 取最小值时的 θ 为要求的最佳参数。这样就转换为了我们熟悉的优化问题，可以用梯度下降法或者牛顿法求解。

2.3 构造代价函数

根据目标函数可以构造出代价函数。

将（16）式代入（17）式得：

$$J(\theta) = -\frac{1}{m} \sum \left(y log h_\theta(x) + (1-y)\log\left(1 - h_\theta(x)\right) \right) \tag{17}$$

构造代价函数为：

$$Cost(h_\theta(x), y) = \begin{cases} -\log\left(h_\theta(x)\right), & y = 1 \\ -\log\left(1 - h_\theta(x)\right), & y = 0 \end{cases} \tag{18}$$

2.4 梯度下降法求 J(θ)的最小值

求 J(θ)的最小值可以使用梯度下降法，根据梯度下降法可得 θ 的更新过程：

$$\theta_{\mathrm{j}} \coloneqq \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta), \qquad (j = 0, \cdots, n) \tag{19}$$

式中为α学习步长，下面来求偏导：

$$\begin{aligned}
\frac{\partial}{\partial\theta_j}J(\theta) &= -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\frac{1}{h_\theta(\mathrm{x}^{(i)})}\frac{\partial}{\partial\theta_j}h_\theta(\mathrm{x}^{(i)})-(1-y^{(i)})\frac{1}{1-h_\theta(\mathrm{x}^{(i)})}\frac{\partial}{\partial\theta_j}h_\theta(\mathrm{x}^{(i)})\right)\\
&= -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\frac{1}{g(\theta^T x^{(i)})}-(1-y^{(i)})\frac{1}{1-g(\theta^T x^{(i)})}\right)\frac{\partial}{\partial\theta_j}g(\theta^T x^{(i)})\\
&= -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\frac{1}{g(\theta^T x^{(i)})}-(1-y^{(i)})\frac{1}{1-g(\theta^T x^{(i)})}\right)g(\theta^T x^{(i)})\left(1-g(\theta^T x^{(i)})\right)\frac{\partial}{\partial\theta_j}\theta^T x^{(i)}\\
&= -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\left(1-g(\theta^T x^{(i)})\right)-(1-y^{(i)})g(\theta^T x^{(i)})\right)x_j^{(i)}\\
&= -\frac{1}{m}\sum_{i=1}^{m}\left(y^{i}-g(\theta^T x^{(i)})\right)x_j^{(i)}\\
&= -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}-h_\theta(\mathrm{x}^{(i)})\right)x_j^{(i)}\\
&= \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(\mathrm{x}^{(i)})-y^{(i)}\right)x_j^{(i)}
\end{aligned} \qquad (20)$$

上式求解过程中用到如下的公式：

$$\begin{aligned}
f(x) &= \frac{1}{1+e^{g(\mathrm{x})}}\\
\frac{\partial}{\partial x}f(x) &= \frac{1}{\left(1+e^{g(\mathrm{x})}\right)^2}e^{g(\mathrm{x})}\frac{\partial}{\partial x}g(x)\\
&= \frac{1}{1+e^{g(\mathrm{x})}}\frac{e^{g(\mathrm{x})}}{1+e^{g(\mathrm{x})}}\frac{\partial}{\partial x}g(x)\\
&= f(x)\left(1-f(x)\right)\frac{\partial}{\partial x}g(x)
\end{aligned} \qquad (21)$$

因此，并忽略 1/m，则（11）式的更新过程可以写成：

$$\theta_{\mathrm{j}} \coloneqq \theta_j-\alpha\sum(h_\theta(x)-y)x_j\,, \qquad (j=0,\cdots,n) \qquad (22)$$

向量化上述过程，可得θ更新的步骤如下：

（1）求 A=x.θ；

（2）求 E=g(A)-y；

（3）求 θ:=θ-α.x'.E,x'表示矩阵 x 的转置。

三、代码实现

这里给出梯度下降法的代码实现：

```
# 梯度下降法
def grad_desc(data_mat, label_mat, rate, times):
    """
    :param data_mat: 数据特征
    :param label_mat: 数据标签
    :param rate: 速率
    :param times: 循环次数
    :return: 参数向量
    """
    data_mat = np.mat(data_mat)
    label_mat = np.mat(label_mat)
    m, n = np.shape(data_mat)
    weight = np.ones((n, 1))
    for i in range(times):
        h = sigmoid(data_mat * weight)
        error = h - label_mat
        weight = weight - rate * data_mat.transpose() * error
    return weight
```

从上面代码分析可以看出，虽然只有十多行的代码，但是里面却隐含了太多的细节，如果没有相关基础确实是非常难以理解的。

这里将随机梯度下降法的代码实现也一并给出：

```
# 随机梯度下降法
def random_grad_desc(data_mat, label_mat, rate, times):
    data_mat = np.mat(data_mat)
    m, n = np.shape(data_mat)
    weight = np.ones((n, 1))
    for i in range(times):
        for j in range(m):
            h = sigmoid(data_mat[j] * weight)
            y = 1 if label_mat[j] == 1 else 0
            error = h - y
            weight = weight - rate * data_mat[j].transpose() * error
    return weight
```

四、多类分类问题的 Logistic Regression

上面介绍的 Logistic 回归模型是二项分类模型，用于二类分类。可以将其推广到多项 Logistic 回归模型（multi-nominal logistic regression model），用于多类分类。假设离散型随机变量 Y 的取值集合是{1，2，…，K}，那么多项回归模型是：

$$P(Y=k\mid x)=\frac{\exp(w_k\cdot x)}{1+\sum_{k=1}^{K-1}\exp(w_k\cdot x)},\quad k=1,2,\cdots,K-1 \qquad (23)$$

$$P(Y=K\mid x)=\frac{1}{1+\sum_{k=1}^{K-1}\exp(w_k\cdot x)} \qquad (24)$$

五、项目相关介绍

5.1 问题介绍

我们将建立一个逻辑回归模型来预测一个学生是否被大学录取。假设你是一个大学系的管理员，你想根据两次考试的结果来决定每个申请人的录取机会。你有以前的申请人的历史数据，你可以用它作为逻辑回归的训练集。对于每一个培训例子，你有两个考试的申请人的分数和录取决定。为了做到这一点，我们将建立一个分类模型，根据考试成绩估计入学概率。

5.2 部分代码示例

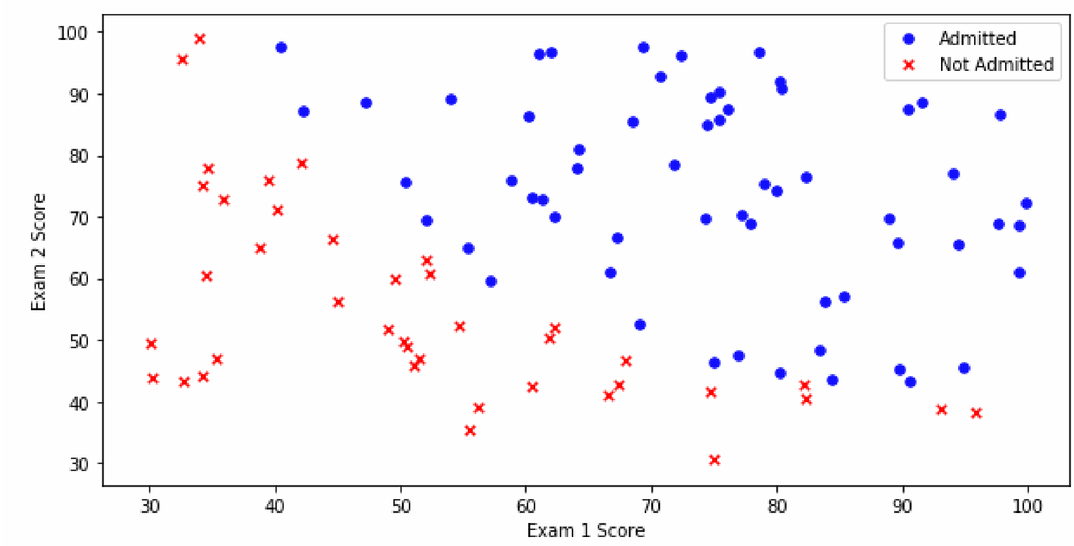
1. 导入数据集

```
path = 'LogiReg_data.txt'

pdData = pd.read_csv(path, header=None, names=['Exam 1', 'Exam 2', 'Admitted'])
```

2. 数据示图

```
positive = pdData[pdData['Admitted'] == 1]

negative = pdData[pdData['Admitted'] == 0]



fig, ax = plt.subplots(figsize=(10,5))

ax.scatter(positive['Exam 1'], positive['Exam 2'], s=30, c='b', marker='o', label='Admitted')

ax.scatter(negative['Exam 1'], negative['Exam 2'], s=30, c='r', marker='x', label='Not Admitted')

ax.legend()

ax.set_xlabel('Exam 1 Score')
```

```
ax.set_ylabel('Exam 2 Score')
```



3. 数据处理

```
pdData.insert(0, 'Ones', 1)  # 插入一列，保障系数 beta0
```

4. 预测

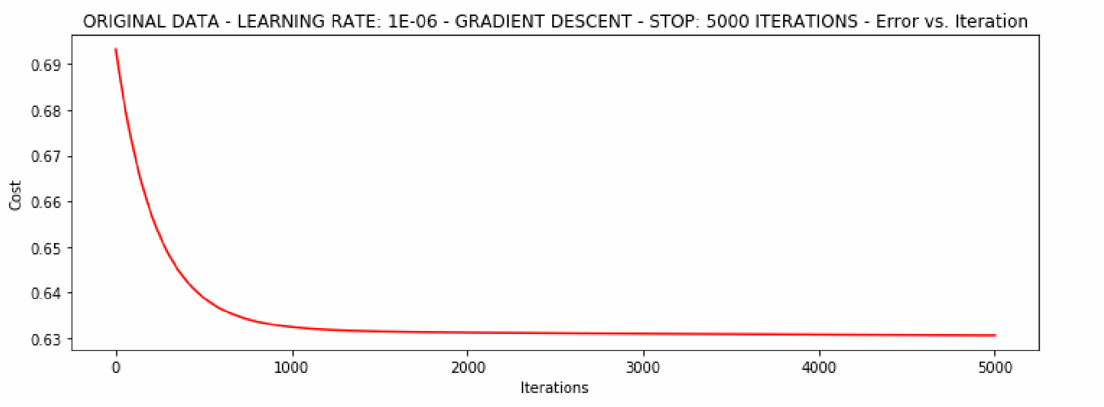
```
#设定阈值
def predict(X, theta):
    return [1 if x >= 0.5 else 0 for x in model(X, theta)]
```

5. 精确度计算

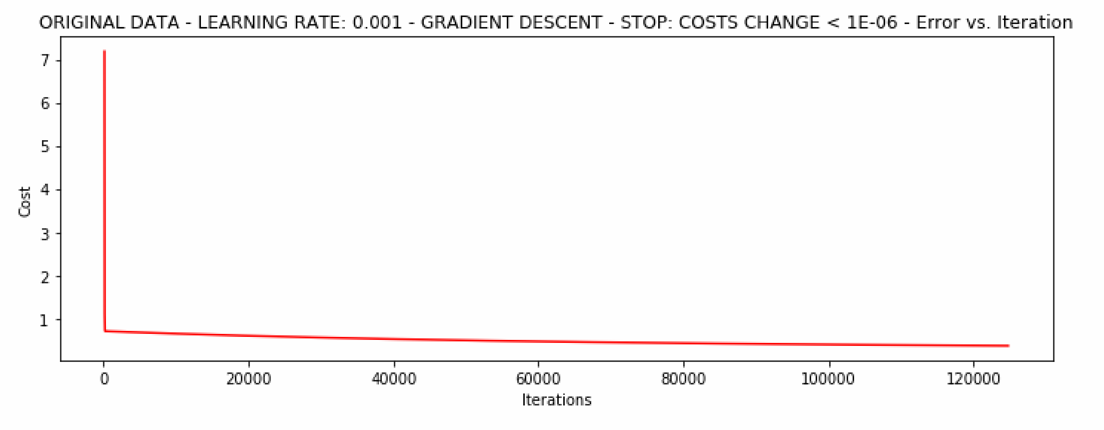
```
scaled_X = scaled_data[:, :3]
y = scaled_data[:, 3]
predictions = predict(scaled_X, theta)
correct = [1 if ((a == 1 and b == 1) or (a == 0 and b == 0)) else 0 for (a, b) in
zip(predictions, y)]
accuracy = (sum(map(int, correct)) % len(correct))
print('accuracy = {0}%'.format(accuracy))
```

5.3 关于梯度下降法

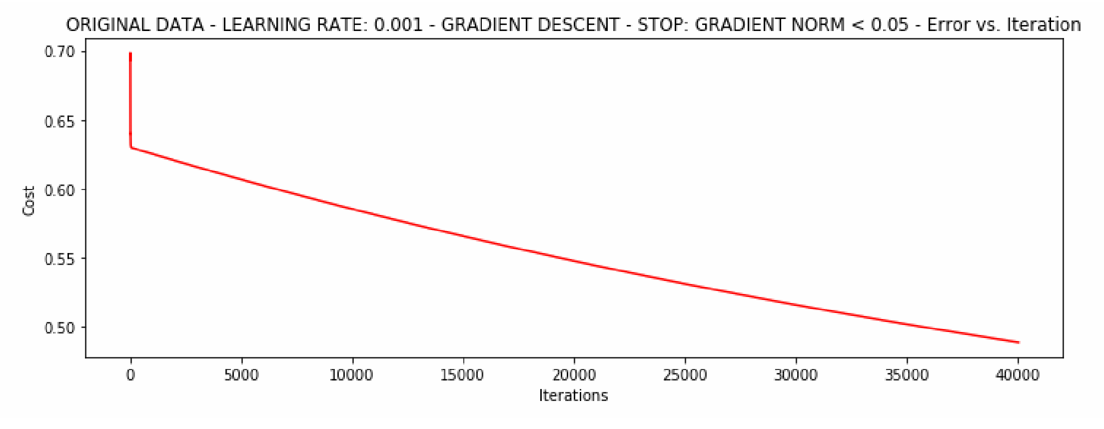
我们可以选择多种条件控制迭代，比如设定迭代次数，超出则停止：



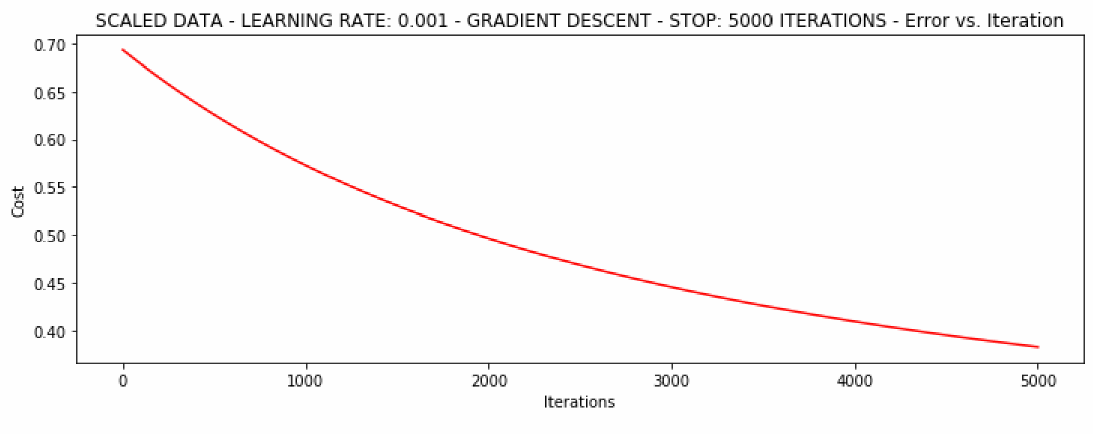
也可以根据损失值停止：



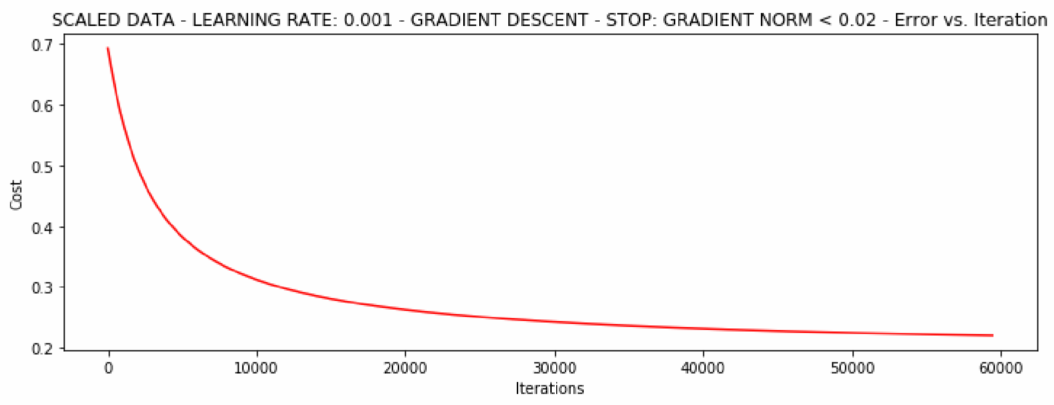
还可以根据梯度变化停止：



同时也可以选用不同的梯度下降法，比如 Mini-batch descent



另外调整学习率也是很关键的:



最后，综合得出识别精确率 89%。

六、总结

后顾看看前些日子，真的很不容易。熬了好几个晚上，起初准备做支持向量机，但真的好难，啃了几天没啃下来，在用了 OpenCV 的库函数进行了手写数字识别之后，就放弃了。转做 Logistic Regression，做这个也很不容易。刚开始是选的 Lentcode 上商品分类的那道题，但是那个题是 9 类分类，而我对 LG 算法的了解还处于二值分类问题上。查了些资料，感觉有点眉目了就开始做，但类别太多

实现起来还是力不从心。就查阅各类测试网站，看到了经典的葡萄酒分类问题，只有三类，查阅了相关资料后，我进行了实现，但效果不太好。这个时候我对 LR 已经非常熟悉了，就最终选做了预测考试录取的问题。实现起来还好，有 jupyter notebook 的帮助真的是事半功倍，感谢赵老师上课期间的科普！

这次的大作业完成很艰难，但却让我印象深刻，收获很多。在我查阅资料的同时，发现了很多各式算法，有 K-means 的、决策树的、CNN 的，都是我不曾接触的，看到这类算法的识别率，再看看我自己算法的效果极大地增强了我对模式识别、机器学习的兴趣，我一定会坚持不懈，继续走下去的。最后，再次感谢赵老师课上课下的指导和分享！