

**模式识别大作业**

题 目 logistics回归预测

学 院 信息科学与工程

专 业 控制科学与工程

组 员 俞天钧(Y30180693)

指导教师 赵海涛

**完成日期： 2018 年 10 月24日**

**模式识别作业报告——Logistics回归**

组员：俞天钧

本次大作业我的内容是利用logistics回归分析从氙气病症状预测病马的死亡率，作业的内容包括：logistics回归分析的具体思路以及推导过程，logistics回归算法以及算法的优化，logistics回归预测病马死亡率的代码实现，以及对代码的微调优化。

**一、Logistics回归算法所用数据集简介**

该数据集来自2010年1月11日的UCI机器学习数据库，该数据最早由加拿大安大略省圭尔夫大学计算机系Mary McLeish和Matt Cecile收集。其数据来源见http://archive.ics.uci.edu/ml/machine-learning-databases/horse-colic/。

 每一行为一样本，最后一列为标签。 这里的数据包含368个样本和28个特征，该数据集中包含了医院检测马氙气病的一些指标，有的比较客观，同时有的比较主观，随意会影响最后预测的结果。

本作业将使用logistics回归来预测患有氙病的马的存活问题。

1. **Logistics回归分析的具体思路及推导**

**2.0线性回归**

线性回归假设特征和结果满足线性关系。其实线性关系的表达能力非常强大，每个特征对结果的影响强弱可以有前面的参数体现，而且每个特征变量可以首先映射到一个函数，然后再参与线性计算。这样就可以表达特征与结果之间的非线性关系。

我们用 *X1，X2..Xn* 去描述描述对象里面的分量，我们可以做出一个估计函数：

 （其中：）

*θ* 在这儿称为参数，在这的意思是调整 feature 中每个分量的影响力。我们程序也需要一个机制去评估我们 *θ* 是否比较好，所以说需要对我们做出的 *h* 函数进行评估，一般这个函数称为损失函数（loss function）或者错误函数(error function)，描述 *h* 函数不好的程度，我们引入如下函数为 *J* 函数：



注：这里引入1/2在之后会被削掉，是为了之后的推导过程可以更简单直观。

如何调整 *θ* 以使得 *J(θ)*取得最小值有很多方法，其中有最梯度下降法。

**2.1 梯度下降**

在选定线性回归模型后，只需要确定参数 *θ*，就可以将模型用来预测。然而 *θ* 需要在 *J(θ)* 最小的情况下才能确定。因此问题归结为求极小值问题，使用梯度下降法。梯度下降法最大的问题是求得有可能是全局极小值，这与初始点的选取有关。

梯度下降法是按下面的流程进行的：

1. 首先对 *θ* 赋值，这个值可以是随机的，也可以让 *θ* 是一个全零的向量。
2. 改变 *θ* 的值，使得 *J(θ)*按梯度下降的方向进行减少。

已知：，设有：

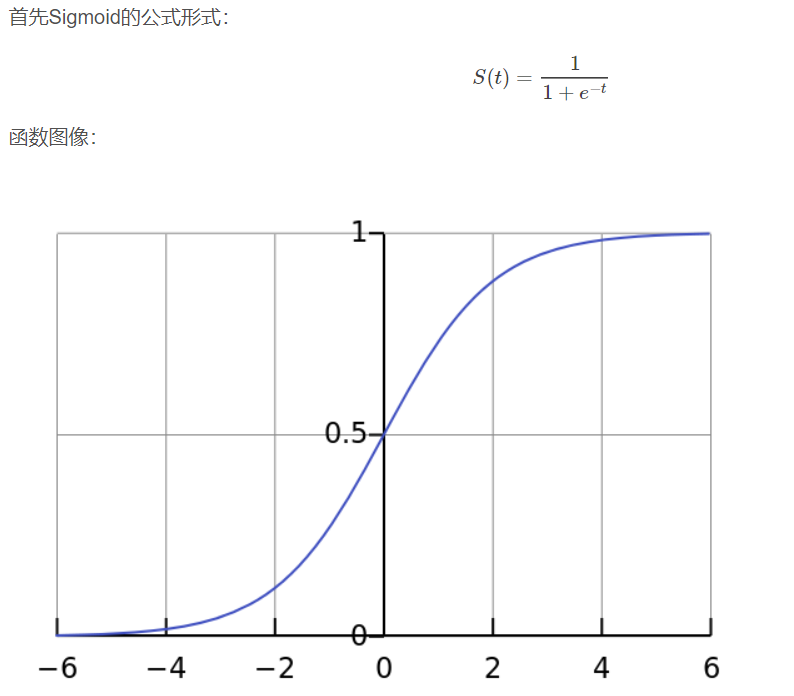


其中，梯度方向由*J(θ)*对*θ*的偏导数确定，由于求的是极小值，因此梯度方向是偏导数的反方向。

结果为 ：

**2.2Logistics回归与Sigmoid函数**

引入Sigmoid函数的概念：



函数的基本性质：

1.定义域：(−∞,+∞)(−∞,+∞)

2.值域：(−1,1)(−1,1)

3.函数在定义域内为连续和光滑函数

4.处处可导，导数为：*s′(x)=s(x)(1−s(x))*

其中，性质4可以推导：



一般来说，回归不用在分类问题上，因为回归是连续型模型，而且受噪声影响比较大。如果非要应用进入，可以使用Logistics回归。

Logistics回归本质上是线性回归，只是在特征到结果的映射中加入了一层函数映射，即先把特征线性求和，然后使用函数 *g(z)*将最为假设函数来预测。*g(z)*可以将连续值映射到 0 和 1上。

Logistics回归的假设函数如下，线性回归假设函数是。







Logistics回归用来分类 0/1 问题，也就是预测结果属于 0 或者 1 的二值分类问题。这里假设了二值满足伯努利分布，也就是



通过简化可以写得：

假设*m*个训练实例是独立生成的，那么我们可以写出参数的似然性：



对取对数可以得到：



最终我们的目标是最大似然估计函数，即：

**2.3梯度上升以及其改进**

梯度上升算法和我们平时用的梯度下降算法思想类似，梯度上升算法基于的思想是：要找到某个函数的最大值，最好的方法是沿着这个函数的梯度方向探寻！直到达到停止条件为止。梯度上升算法用来求函数的最大值，而梯度下降算法用来求函数的最小值。



推导过程如下：



最终可以得到结论为：

代码部分：



**随机梯度上升**：

梯度上升算法咋i每次更新回归系数时都需要遍历整个数据集，该方法处理较少数据还行，但对于较为庞大的数据集时，该方法的就相对比较落后，需要进行改进：我给出的方案是一次仅用一个样本点来更新回归系数，该方法此方法便称为随机梯度上升算法。

 伪代码部分：

所有回归系数初始化为1

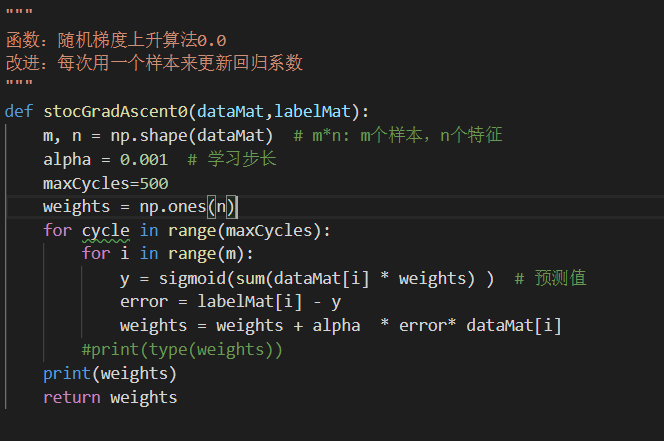
对数据集中每个样本

计算该样本的梯度

使用\*gradient更新回归系数值

返回回归系数值

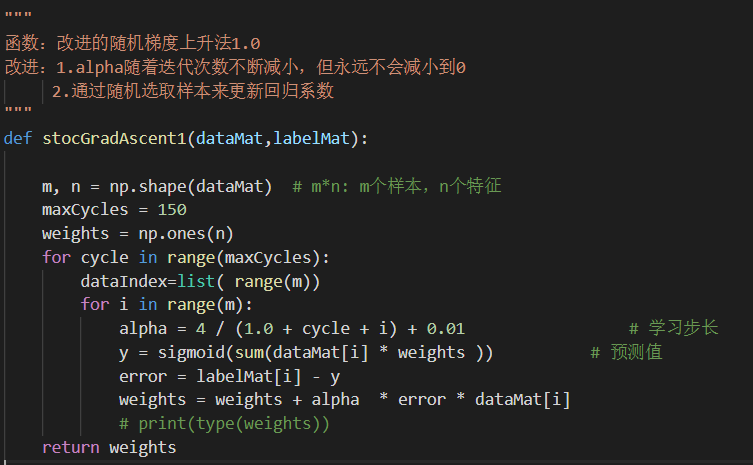
代码部分：



**改进之后的随机梯度上升：**

1. 迭代次数可以进行修改，由于修改方式简单，本文将进行人工修改
2. 学习率在每次迭代的时候都会进行调整，在每次迭代之后会逐渐减小，但永远不为0（即趋近于一个常数），本文中，设有，其中j是最大为150，i最大为样本的个数，用两个for循环不断修改学习率，这样做是为了保证多次迭代之后的新数据任然具有一定的影响。

代码部分：



**三、整体解决方案**

**2.1具体步骤分析：**

1. 收集数据：已知给定了数据文件
2. 准备数据：用python解析文件并且填充缺失值
3. 分析数据：可视化并观察数据
4. 训练算法
5. 测试算法
6. 使用算法

**2.2准备数据**

**具体任务：处理了数据中的缺失值**

我们可以采用以下的方法来解决这个问题：

1.使用可用特征的均值来填补缺失值；

2.使用特殊值来填补缺失值，如0；

3.忽略有缺失值的样本；

4.使用相似样本的均值添补缺失值；

5.使用另外的机器学习算法预测缺失值。

这里我们使用的是第二种，选择理由是数值0在迭代过程中对其他数据不会有影响，即当遇到原本的缺失数值时，特征的系数不做更新。

这里用到pandas库中的DataFrame.fillna()方法，具体代码如下：

import pandas as pd

import numpy as np

df=pd.DataFrame(TrainData)

df.ix[1:,-1]=np.nan

print('\nzero')

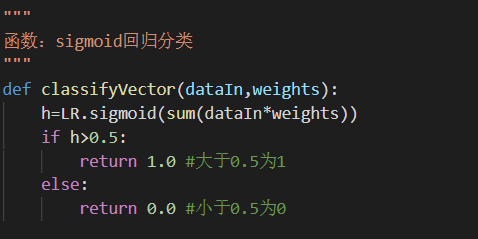
print(df.fillna(0))

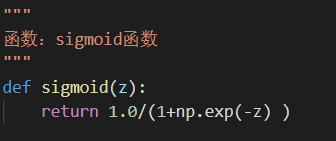
我们在代码中直接使用预处理完成的文本文件

**2.3训练算法及程序实现**

使用logistics回归方法进行分类需要做的只是把测试集上每个特征值向量乘以最优化方法得到的回归系数，再将该乘积求和，最后输入到Sigmoid函数中即可。如果对应的sigmoid值大于0.5，则预测类的标签为1，反之，为0。

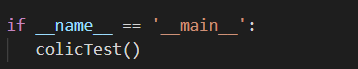
本文使用随机梯度上升算法，算法代码见上方。





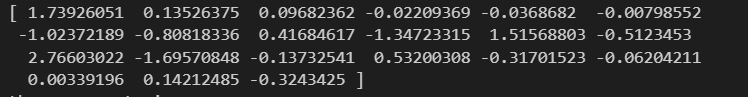






**2.4评定测试**

通过代码的实现，最后得到回购系数为：



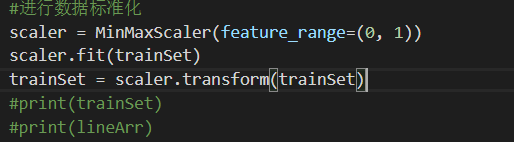
得到错误率为：

因为众多原因（比如不知道各个数据的加权因子，拥有30%左右的nan值，并且用0取代nan值是否可行等等），最后得到预测正确率大约在65%左右，勉强可以使用。

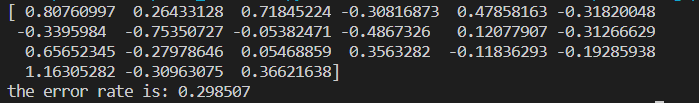
**2.5调试及预测结果**

对代码进行进一步优化，方法有如下几种：

1. 对数据进行归一化

、

结果为：

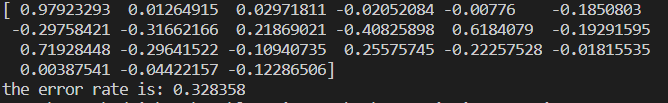


可以看出效果显著，成功率达到70%

1. 改变学习率以及迭代次数

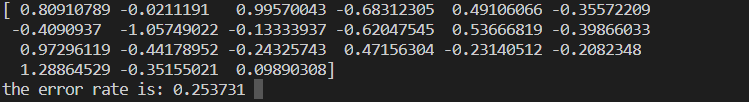
通过改变学习率为0.0006，迭代次数还是500

得到的结论为：



比之前的要好。

**综合两者，得到最终的正确率达到75%**



**四、作业总结**

本次实验我参考了‘机器学习实战’这本书，通过这本书上logistics回归的学习，我第一次写了这个作业，本次作业其实还有很多不满意的地方：比如最后的正确率还是太低，比如并没有进一步对数据进行更优化的处理。同时，我还发现了我编程上的大量不足：python函数的运用的不熟练等等，但通过这次作业，我更加深刻地了解了logistics回归的理论，同时还学习了继续进行其他机器学习的方法，我认为过程是不完美的，并且是痛苦的，但第一次已经过去了，我可以在第二次，第三次更好的学习，希望我以后可以拿出比这次作业更好的笔记来。

**附：文件说明**

本次附件一共包含有：

1 大作业报告；

2 最终的Python实现程序源码：logistics.py

3 训练集与测试集文件