

**模式识别大作业**

题 目 基于logistic模型的分类与应用

学 院 信息科学与工程

专 业 控制科学与工程

组 员 江润强

指导教师 赵海涛

**完成日期： 2018年 10 月25日**

**基于Logistic回归模型的分类与应用**

Logistic Regression(逻辑回归)是模式识别中一个非常常见的模型，在实际的生产环境中也经常被用到，是一种经典的分类模型。本文主要是在老师课上的讲解以及自己课下查阅相关的资料文献的基础上生成的。本文主要介绍了Logistic Regression(逻辑回归)模型的原理以及参数估计、公式的推导过程以及分类的具体应用实例。

**一、Logistic Regression问题引出**

实际生活中，我们可能会遇到下列问题：1、预测一个用户是否点击一个特定的商品 2、判断用户性别 3、判断一条评论是正确的还是负面的…….这些都可以看成是分类的问题，更准确的说是看成二分类的问题。但是要解决这些问题，通常会用到一些分类方法，比如逻辑回归或者支持向量机。他们都属于是有监督的学习。

**二、Logistic Regression原理推导**

首先，线性回归的主要思想是通过历史数据拟合一条直线，用这条直线对新的数据进行预测。线性回归的公式如下：

+….+（1-1）

写成向量形式为：Z=Tx+b,其中T,其中与b确定了后模型也就确定了。

而对于Logistic Regression来说，其思想也是基于线性回归（Logistic Regression 属于广义的线性回归模型）。同时考虑一个二分类的问题，其输出应该记为y{0,1}而线性模型产生的预测值Z=Tx+b，所以问题转化为将实数值z转化为{0,1}的值。于是需要引进一如下的分段函数： 又因为单位阶跃函数并不连续，并不可以作为广义的线性模型。所以我们需要找到一个在某种程度上能够近似替代单位阶跃函数的替代函数，其中**sigmoid**函数就是一个符合条件的函数。

同时实数值公式如下：

（1-2）

其中, 被称作**sigmoid**函数，从中可以看出来logistic Regression算法是将线性函数的结果映射到**sigmoid**函数之中。

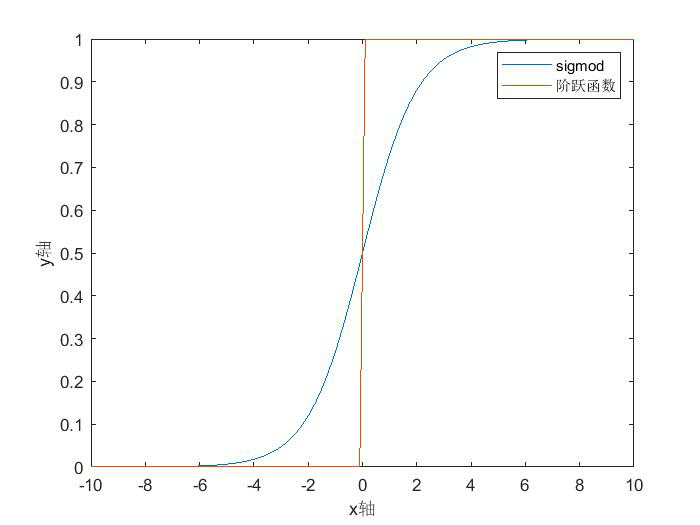
同时**sigmoid**函数的函数图像如下图所示：  
  


图1.1 sigmoid函数图像仿真

从图中我们可以看出，**sigmoid**函数的输出值是介于（0,1）之间的，中间值也恰好是0.5。于是从之前的公式的输出也是介于（0,1）之间的数，也叫说明了数据是属于某一类别的概率，例如：

<0.5，则说明当前数据属于A类；

>0.5，则说明当前数据属于B类。

由此可以看出，我们可以将**sigmoid**函数看成样本数据的概率密度函数。因此函数的值有了特殊的意义，它表示结果取为1的概率，因此对于输入x分类结果为类别1和类别0的概率分别为：

（1-3）

（1-4）

所以事件发生与不发生的概率比为：

（1-5）

这个比值称为事件的发生比,简记为odds。对他取对数有：

Tx+b （1-6）

由公式可以看出模型的本质实际是用线性回归的预测结果去逼近真实的对数几率。

**三、回归参数的计算方法**

从上面分析可以知道Logistic回归是围绕一个Logistic函数展开。接下来通过极大似然估计来求对应的分类器参数。首先假设有m个观测样本，他们的观测值分别是y1,y2,y3,…ym。同时设pi=P(yi=1|xi)为给定条件下得到的yi=1的概率,同样的yi=0的概率为P（yi=0|xi）=1-pi。于是便可以得到一个服从二项分布的观测值概率模型为：

P(yi)=piyi(1-pi)1-yi （1-7）

又由于各个观测样本之间相互独立，那么他们的联合分布由概率论的知识有：联合分布等于各边缘分布的乘积。于是建立如下的极大似然模型：

对于给定m个的数据集（xi,yi），Logistic回归模型的极大似然化为：

= （1-8）

取对数化简为：

（1-9）

从极大似然估计的原理知，令每个样本属于其真实标记的概率越大越好。设行向量βT=（b,）,X=(1,xi)。那么Tx+b可以写成βT X。再令

（1-10）

（1-11）

（1-12）

则对数化后似然函数（1-9）的通项可以改写为下面的式子：

(1-13)

将（1-13）代入（1-9），同时由（1-3）与（1-4）可以得到（1-13）的等价形式为下面的式子：

(1-14)

同时因为式（1-14）是一个高阶可导的凸函数，于是由最优化的知识可以知道，这个函数的最值可以通过经典的数值优化的思想进行求解：比如梯度下降法、二分法、牛顿法、斐波那契数列等方法均可以求取。此时可以得到的最值记作\*。这里选择梯度下降法进行求取，梯度下降法的原理公式如下式所示：

j+1=j- (1-15) 为步长因子

当用梯度下降法求解出\*时，对应的参数与b就均可以得到，进而代入式（1-2）中将样本数据分为两类。这个时候也就实现了对应的logistic回归的二分类的具体应用。

**四、实验结果分析**

本次实验的主程序在附件的main.py 中，Logistic regression 函数以及梯度下降法的优化算法求取对应参数在附件grad.py中。

运行结果分析：

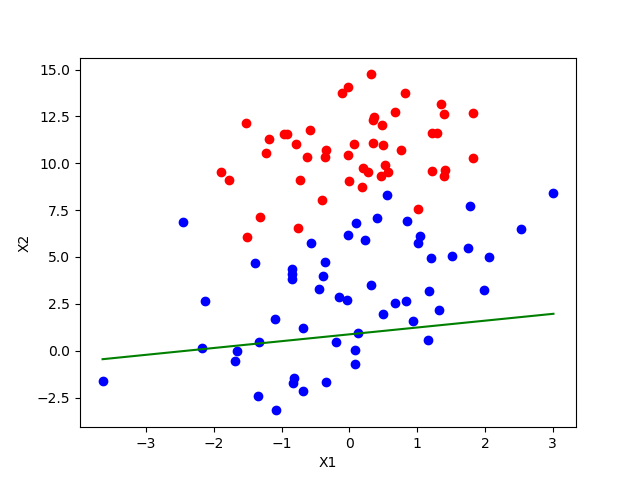
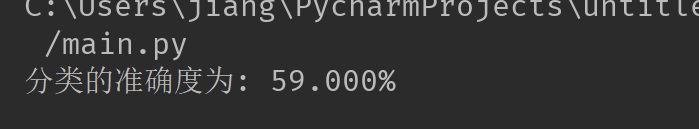


图1.2 迭代次数为10次

10次迭代对应的准确率：



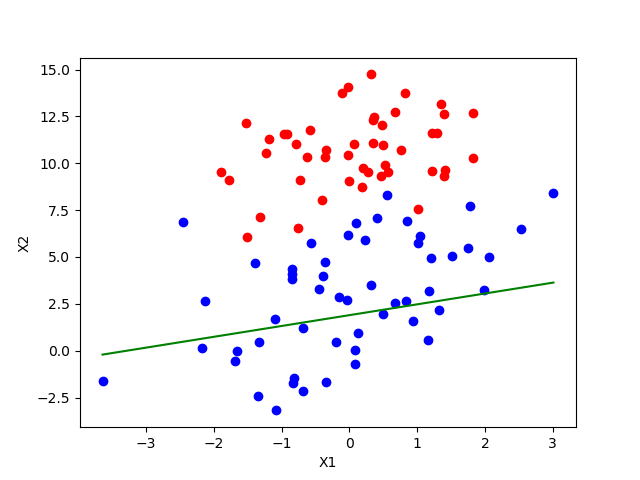
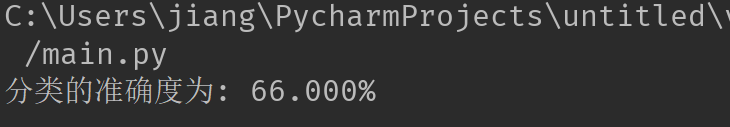


图1.3迭代次数为30次

30次迭代对应的准确率：



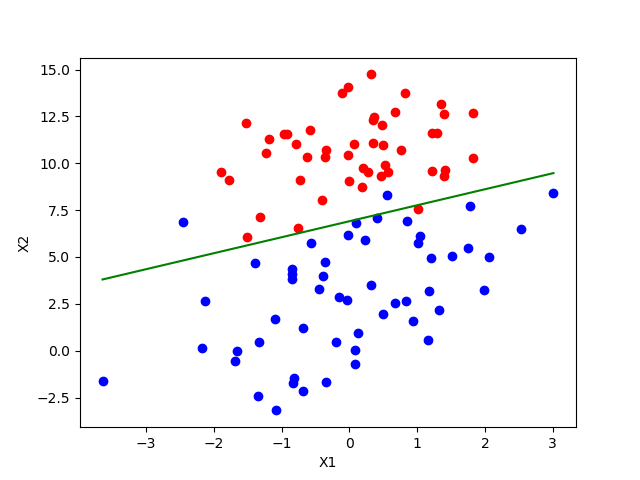
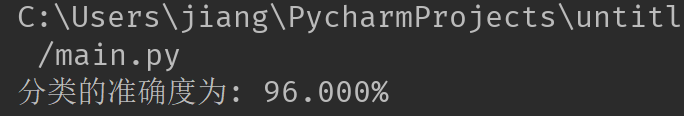


图1.4迭代次数为100次

100次迭代对应的准确率：



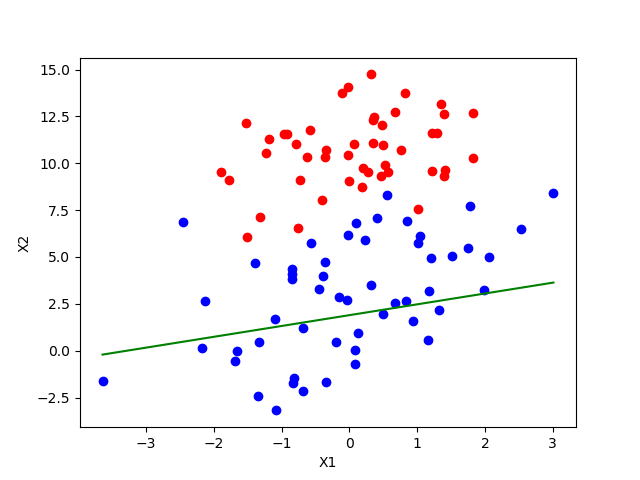
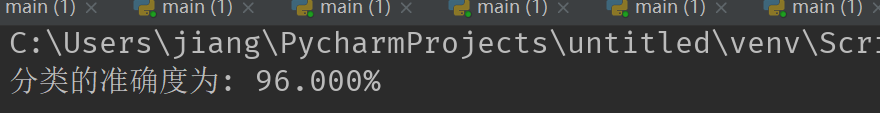


图1.5迭代次数为1000次

1000次迭代对应的准确率:



通过上面的实验结果的图形以及准确精度的计算可以知道：一般在一定的范围内迭代的次数越多，分类的精度也就越高。

**五、作业总结**

这是模式识别课的第一次大作业，也是第一次接触数据的处理与算法的推导和编程的实现。可想而知，这里面一定碰到了许多的困难，同时也一定查阅了相关的资料。也是第一次接触python的程序与算法的结合。在这个过程中，暴露出自己的编程基础的薄弱，这一定是以后的需要努力迎头赶上的地方。同时也发现本科教学与研究生教学的不同与差异。模式识别这门课，对于数学的基础是有一定要求的，自己还需要在某些知识点上去补充与提高。

本次大作业的完成也发现自己的团队意识比较差，能力的提高不是一时的。三人行，必有我师焉。在今后的学习中应该多多向周围的同学请教。感谢赵老师对于作业的监督与要求，尽管自己的作业完成的并不完美，但这至少是自己的一次尝试。有了这一次的体验与经历，相信以后的作业会有所体现的。

**附件：**

程序代码：

main.py

from grad import \*  
from numpy import \*  
  
def loadData():  
 train\_x = []  
 train\_y = []  
 fileIn = open('test.txt')  
 for line in fileIn.readlines():  
 lineArr = line.strip().split()  
 train\_x.append([1.0, float(lineArr[0]), float(lineArr[1])])  
 train\_y.append(float(lineArr[2]))  
 return mat(train\_x), mat(train\_y).transpose()  
  
#读取测试集  
train\_x, train\_y = loadData()  
test\_x = train\_x;  
test\_y = train\_y  
  
#计算回归系数  
opts = {'alpha': 0.01, 'maxIter': 1000, 'optimizeType': 'smoothStocGradDescent'}  
optimalWeights = trainLogRegres(train\_x, train\_y, opts)  
  
#在测试集上进行测试  
accuracy = testLogRegres(optimalWeights, test\_x, test\_y)  
  
#显示结果  
print('分类的准确度为: %.3f%%' % (accuracy \* 100))  
showLogRegres(optimalWeights, train\_x, train\_y)

grad.py

from numpy import \*  
import matplotlib.pyplot as plt  
import time  
#计算sigmod函数  
def sigmoid(inX):  
 return 1.0 / (1 + exp(-inX))  
  
#回归系数计算函数  
def trainLogRegres(train\_x, train\_y, opts):  
 # 得到计算起始时间  
 startTime = time.time()  
  
 numSamples, numFeatures = shape(train\_x)  
 alpha = opts['alpha'];#得到控制台输入的初始回归系数  
 maxIter = opts['maxIter']#得到控制台输入的最大迭代次数  
 weights = ones((numFeatures, 1))  
  
 # 通过梯度下降法迭代回归系数  
 for k in range(maxIter):  
 output = sigmoid(train\_x \* weights)  
 error = train\_y - output  
 weights = weights + alpha \* train\_x.transpose() \* error  
 return weights  
  
  
# 测试，利用得到的回归系数进行分类，并与正确值进行比较，得到精确度  
def testLogRegres(weights, test\_x, test\_y):  
 numSamples, numFeatures = shape(test\_x)  
 matchCount = 0  
 for i in range(numSamples):  
 predict = sigmoid(test\_x[i, :] \* weights)[0, 0] > 0.5  
 if predict == bool(test\_y[i, 0]):  
 matchCount += 1  
 accuracy = float(matchCount) / numSamples  
 return accuracy  
  
  
# 通过matplotlib库进行绘图  
def showLogRegres(weights, train\_x, train\_y):  
 numSamples, numFeatures = shape(train\_x)  
 if numFeatures != 3:  
 return 1  
 #通过循环绘出每个点，并对不同的类用不同的颜色画出  
 for i in range(numSamples):  
 if int(train\_y[i, 0]) == 0:  
 plt.plot(train\_x[i, 1], train\_x[i, 2], 'or')  
 elif int(train\_y[i, 0]) == 1:  
 plt.plot(train\_x[i, 1], train\_x[i, 2], 'ob')  
  
 #通过回归系数画出分类线  
 min\_x = min(train\_x[:, 1])[0, 0]  
 max\_x = max(train\_x[:, 1])[0, 0]  
 weights = weights.getA()  
 y\_min\_x = float(-weights[0] - weights[1] \* min\_x) / weights[2]  
 y\_max\_x = float(-weights[0] - weights[1] \* max\_x) / weights[2]  
 plt.plot([min\_x, max\_x], [y\_min\_x, y\_max\_x], '-g')  
 plt.xlabel('X1');  
 plt.ylabel('X2')  
 plt.show()