**华中科技大学计算机科学与技术学院**

**机器学习报告**



专 业： 计算机科学与技术

班 级： CS1805班

学 号： U201814615

姓 名： 于祯奇

成 绩：

指导教师： 邹复好

**完成日期： 2020年 6 月 29 日**

目录

[机器学习结课项目——个人收入预测 2](#_Toc44494290)

[一、概述 2](#_Toc44494291)

[1.1 任务描述 2](#_Toc44494292)

[1.2 实验思想介绍 2](#_Toc44494293)

[二、算法介绍及设计 3](#_Toc44494294)

[2.1 实验中使用的第三方库 3](#_Toc44494295)

[2.2 文件读取及数据预处理部分 3](#_Toc44494296)

[2.3 逻辑回归部分 4](#_Toc44494297)

[2.4 交叉熵损失函数部分 4](#_Toc44494298)

[2.5 自适应梯度下降法Adagrad部分 4](#_Toc44494299)

[2.6 计算训练集、测试集的Loss部分 5](#_Toc44494300)

[2.7 计算准确率部分 6](#_Toc44494301)

[三、实验环境及平台 6](#_Toc44494302)

[四、实验结果与分析 6](#_Toc44494303)

[4.1 不同参数对于Loss曲线的影响 7](#_Toc44494304)

[4.2 不同参数对于在测试集上准确率的影响 10](#_Toc44494305)

[五、小结及感想 13](#_Toc44494306)

[5.1 实验小结 13](#_Toc44494307)

[5.2 实验感想 14](#_Toc44494308)

[参考文献 15](#_Toc44494309)

[附录1 项目代码 16](#_Toc44494310)

# 机器学习结课项目——个人收入预测

## 一、概述

### 1.1 任务描述

给定训练集income.csv，要求根据每个人的属性值来判断此人年收入是否大于50K。

（1）CSV文件，大小为4000行×59列;

（2）4000行数据对应着4000个人，ID编号从1到4000;

（3）59列数据中，第一列为ID，最后一列label(1或0)表示年收入是否大于50K，中间的57列为57种属性值。

要求：

（1）将数据中前3000项作为训练集，后1000项作为测试集，使用logistic回归进行二分类，实现语言要求为Python；

（2）在使用梯度下降法时，调整学习率的固定值，有能力的同学可以学习并使用动态调整学习率的方法，探究不同学习率的选择对训练误差收敛速度的影响，绘制misclassification rate曲线进行比较并分析。

### 1.2 实验思想介绍

本次实验中使用逻辑回归模型。

逻辑回归（Logistic regression）是一种广义的线性回归分析模型，是社会学、机器学习，医学领域等统计实证分析的常用方法。在统计学中，逻辑模型（Logistic model）用于对某个类别或事件（例如通过/失败，健康/生病）的概率进行建模。例如确定图像是否包含猫，狗，狮子等。在图像中检测到的每个对象都将被分配一个介于0和1之间的概率，总和为1。

逻辑回归是一种重要的机器学习算法。对于目标是对给定实验数据的随机变量 Y，取值为0或1的概率进行建模。其本质是一个线性回归的模型θ，带入sigmoid函数后得到的函数。

而对于实际问题来说，我们需要考虑的，通常是线性模型中的w与b，也就是权重和偏置值对问题产生的影响。

本课程项目将会从个人收入预测方面出发，根据给定的训练集income.csv，使用python语言进行建模，将数据中前3000项作为训练集，后1000项作为测试集，使用logistic回归进行二分类，采用交叉熵作为代价函数和自适应梯度下降法（Adagrad）对w，b的取值进行迭代，根据每个人的属性值来判断此人年收入是否大于50K。同时，在调整不同学习率和正则化参数的情况下，输出正确率和绘制LOSS曲线，并对其进行分析。

## 二、算法介绍及设计

详细代码实现参见附录1，在本部分就不再赘述。

### 2.1 实验中使用的第三方库

① numpy：数值计算及格式化处理。

② pandas：读取csv函数

③ xlwt：LOSS曲线写入及绘制

### 2.2 文件读取及数据预处理部分

① 使用pd.read\_csv函数，将数据集.csv读入：

df = pd.read\_csv('数据集.csv')

② 若该数据集中有空值，则填入0：

df = df.fillna(0)

③ 使用numpy库中的array函数将其转化为一个array：

array = np.array(df)

④ 二项Logistic回归模型是一种分类模型，由条件概率分布P(Y|X)表示，随机变量Y取0或1。因此，把array的1-4000行，2-58列作为x，同时1-4000行，59列作为y，则有下列赋值：

x = array[:, 1:-1]

y = array[:, -1]

⑤ 使用Excel软件查看数据集.csv文件，发现实际上大部分数据值在区间[0,2]内。但是，57列和58列中的数据明显不在这个范围内且与其他数据的平均值相差较大。所以我将这两列分别除以每列的算术平均值，这样数据就会相对于之前更加靠近[0,2]，且能保存其特征信息。

x[:, -1] /= np.mean(x[:, -1])

x[:, -2] /= np.mean(x[:, -2])

⑥ 划分训练集和验证集，取x、y中前3000个样本为训练集，后1000个样本为测试集

x\_train, x\_val = x[:2999, :], x[2999:, :]

y\_train, y\_val = y[0:2999], y[2999:]

### 2.3 逻辑回归部分

① 对于一个逻辑回归中P（y|xi）其建模结果如下所示：

② 将上式拆分计算，首先计算**：**，在本项目中也即是求：。然后计算。

### 2.4 交叉熵损失函数部分

在本项目中，需要迭代计算w，b的最佳取值，采用交叉熵损失函数法，具体如下所示：

① 当分类正确时：函数值取0，反之取1，即为：

则交叉熵损失函数的表达式为：

② 为方便计算，我们取以e为底的对数，则有：

在本项目中的模拟值是p，真实值为y，则在本项目中有：

### 2.5 自适应梯度下降法Adagrad部分

① 观察w对该式的影响，对w求偏导得到以下结果：

同理，我们对b求偏导，得到以下结果：

② 求解该方程，得到以下结果：

③ 为防止过拟合，设正则化参数为r，添加正则化项：

④ 采用自适应梯度下降法Adagrad对参数w，b进行迭代，其中：α为学习率，g为该维度上的梯度，bgrad2sum偏置值的梯度平方和，wgrad2sum为w的梯度平方和。

对于w，我们有如下迭代公式（wgrad2sum初值为0）：

同理，对于b，我们有（bgrad2sum初值为0）：

经过round轮迭代之后，即可返回w、b的值。

### 2.6 计算训练集、测试集的Loss部分

① 在迭代过程中，每迭代一次（步长为1），将该次中的w、b，代入logistic回归公式**，**计算出p，则Loss表达式如下（其中y为真实的结果，取值为0或1）：

② 之后要使用xlwt库函数，即可在excel表中写入LOSS值，进而画出LOSS的变化曲线。

值得注意的是：可能会溢出，因此我们可以添加一个，添加之后变化为：

### 2.7 计算准确率部分

① 利用迭代完成的w、b，和logistic回归模型后，调用测试集测试该计算出p，由已知有：，即p＞0.5则分类结果为1，否则为0。

② 将分类结果与真实结果进行对比，则准确率=相同个数/总训练集数量。

## 三、实验环境及平台

系统环境：Windows 10家庭中文版1909

处理器： Intel® Core™ i7-8750H @ 2.20GHz 六核

内存：8 GB ( 金士顿 DDR4 2666MHz )

IDE：PyCharm 2020.1.2 (Community Edition)

Python版本：Python 3.8

第三方包版本信息如表3.1所示。

表3.1 第三方包信息表

|  |  |
| --- | --- |
| **包** | **版本** |
| numpy | 1.19.0 |
| pandas | 1.0.5 |
| pip | 20.1.1 |
| python-deteutil | 2.8.1 |
| pytz | 2020.1 |
| setuptools | 47.3.1 |
| six | 1.15.1 |
| xlwt | 1.3.0 |

## 四、实验结果与分析

### 4.1 不同参数对于Loss曲线的影响

**（1）学习率α对Loss曲线的影响**

固定迭代次数为200，正则化参数为0.001；变化学习率参数α的取值，测试不同学习率的情况下，Loss值随迭代次数收敛的情况。学习率参数的取值分别为：0.01、0.025、0.05、0.1、0.25、0.5、1、2.5、5、10。按照以上取值编写测试代码，并将测试的结果存放到“实验结果.xls”的“学习率对Loss的影响”sheet中。测试代码部分如图4.1所示，测试结果附件“实验结果.xls”。

将数据保存到Excel表后，利用Excel自身工具绘制在不同学习率的情况下，Loss值随着迭代次数增加的变化情况，即Loss收敛曲线，结果如图4.2和图4.3。由于学习率在大于1和小于1之间的曲线变化较大，所以图4.2表示的是学习率小于1的曲线，图4.3是学习率大于1的曲线。

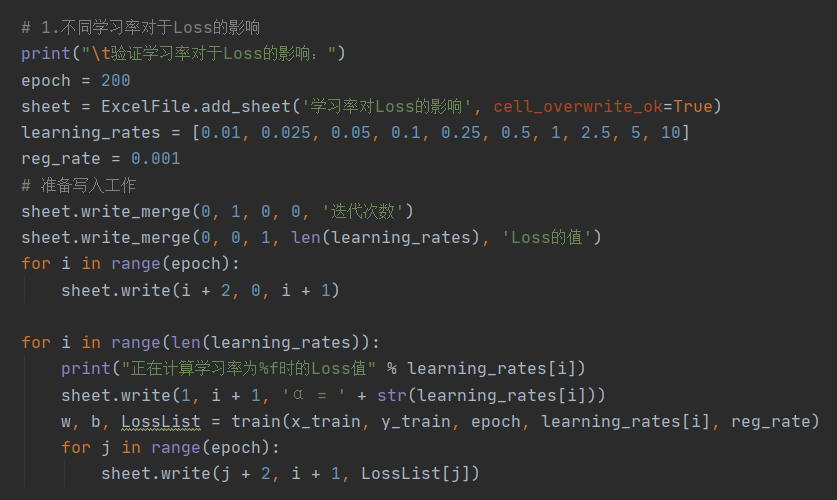
****

图4.1 验证不同学习率对Loss的影响部分测试代码

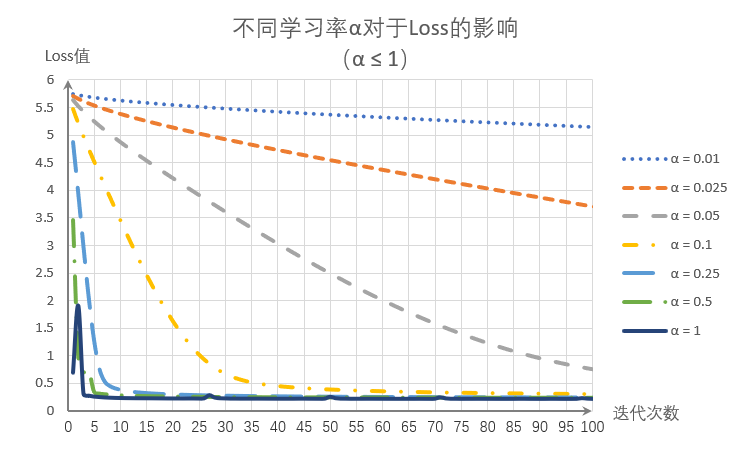


图4.2 不同学习率的Loss收敛曲线图（α≤ 1）

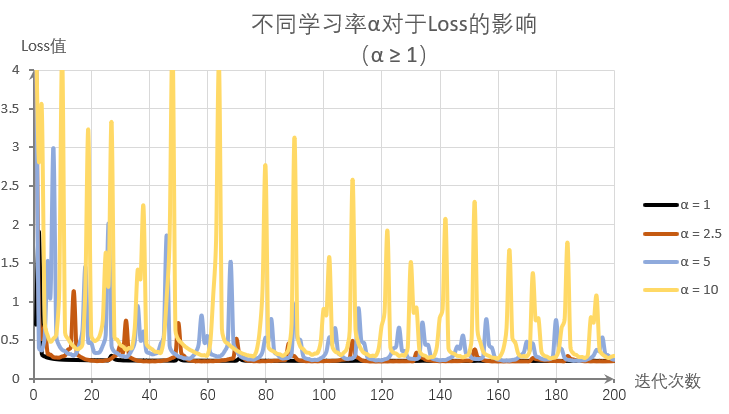


图4.3 不同学习率的Loss收敛曲线图（α≥ 1）

从图4.2中可以得出，当0.001≤α≤1时，学习率参数α越大，Loss曲线的收敛速度越快，用较少的迭代次数就能进入稳定状态；而α越小，Loss曲线收敛速度越慢，进入稳定状态的迭代次数增加，表现最好的曲线就是用黑色表示的实线，代表着α=1时的Loss收敛曲线。

从图4.3中可以看出，当1≤α≤10时，学习率参数α越大，Loss曲线的波动越大，收敛曲线处于不稳定状态且可能一直波动下去，而α越小，Loss曲线则相对稳定。

综上所述，α在1的附近进行取值时，LOSS函数收敛速度最大且最稳定。

**（2）正则化参数r对LOSS曲线的影响**

固定迭代次数为50，学习率为1；变化正则化参数r的取值，测试不同正则化参数的情况下，Loss值随迭代次数收敛的情况。正则化参数取值为：0.0001、0.001、0.01、0.1、1、10。按照以上取值编写测试代码，并将测试的结果存放到“实验结果.xls”的“正则化参数对Loss的影响”sheet中。测试代码部分如图4.4所示，测试结果详见附件“实验结果.xls”。。

和上一部分相同，绘制在不同正则化参数的情况下，Loss值随着迭代次数增加的变化情况，即Loss收敛曲线，结果如图4.5所示。

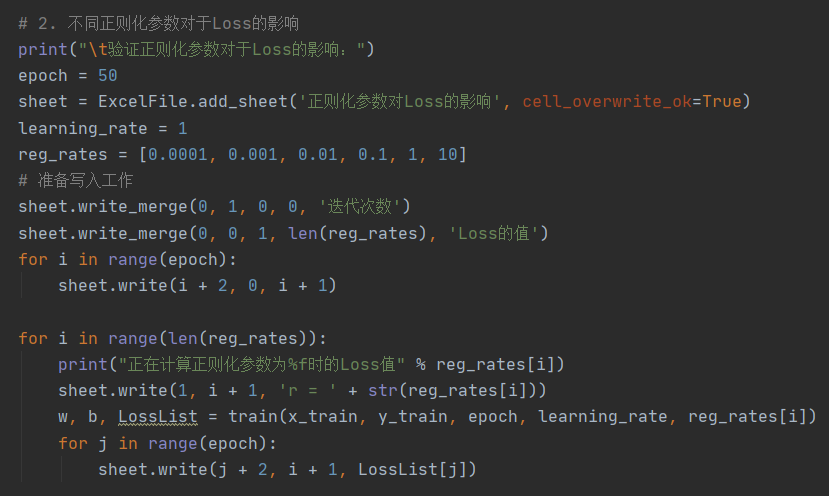


图4.4 验证不同正则化参数对Loss的影响部分测试代码

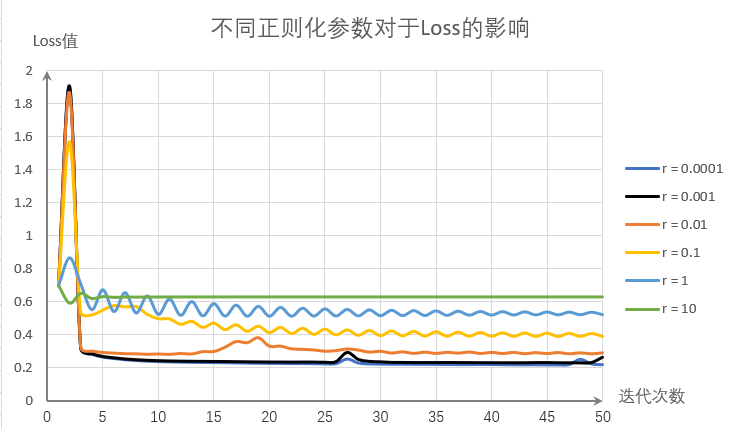


图4.5 不同正则化参数的Loss收敛曲线图

从图4.5中可以看出，当r=0.001和r=0.0001时，Loss曲线近乎重叠，很快就达到了比较稳定的状态，且相比于起始Loss值有明显下降，最后趋近于0.2。在0.001＜r＜10时，随着r的增加，Loss曲线开始出现波动，而且随着r的增加，波动愈加剧烈，所处的波动范围也越来越大。当r=10时，虽然基本上没有波动委屈也很快的达到收敛状态，但是Loss相对于起始状态几乎没有变化，对于Loss的减小没有效果。

综上所述，为了使Loss的值较低且不在发生波动，应将正则化参数选择在0.001以下。

### 4.2 不同参数对于在测试集上准确率的影响

**（1）学习率α对准确率的影响**

固定迭代次数为100，正则化参数为0.001；变化学习率参数α取值，测试不同学习率对于模型在测试集上准确率的影响。学习率参数的取值为：利用numpy库中的linspace函数，在0.01到20中等间距取200个值。按照以上取值编写测试代码，并将测试的结果存放到“实验结果.xls”的“学习率对准确率的影响”sheet中。测试代码部分如图4.6所示，测试结果详见附件“实验结果.xls”。

在Excel中绘制在不同正则化参数的情况下，准确率随着学习率增加的变化情况，结果如图4.7所示。

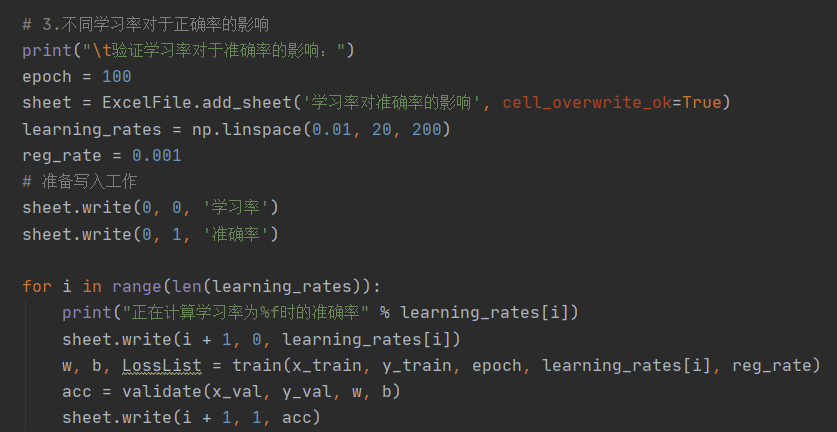


图4.6 验证不同学习率对准确率的影响部分测试代码

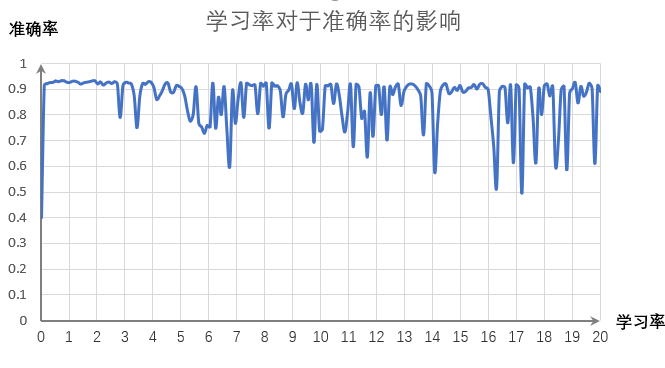


图4.7 不同学习率下的准确率示意图

从图4.7中可以发现，在学习率小于3附近的时候，准确率相对而言比较稳定，在学习率大于3之后，开始比较剧烈的波动，比较难以预测。除此之外其中的最大值和前面小于3部分的平均值相差无几，没有明显优点。

综上所述，结合4.1的（1）部分的内容，最优的学习率选取应该为1附近。

**（2）正则化参数r对准确率的影响**

固定迭代次数为100，学习率为1；变化正则化参数r取值，测试不同正则化参数对于模型在测试集上准确率的影响。正则化参数的取值为：利用numpy库中的linspace函数，在0到0.01中等间距取100个值。按照以上取值编写测试代码，并将测试的结果存放到“实验结果.xls”的“正则化参数对准确率的影响”sheet中。测试代码部分如图4.8所示，测试结果详见附件“实验结果.xls”。

在Excel中绘制在不同正则化参数的情况下，准确率随着正则化参数增加的变化情况，结果如图4.9所示。

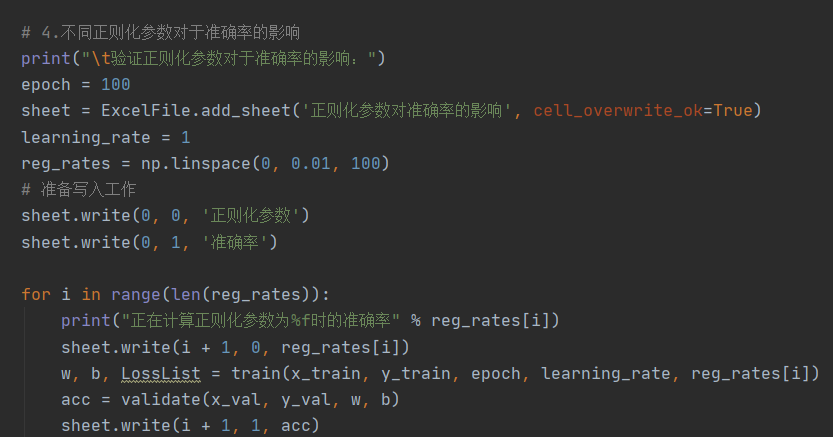


图4.8 验证不同正则化参数对准确率的影响部分测试代码

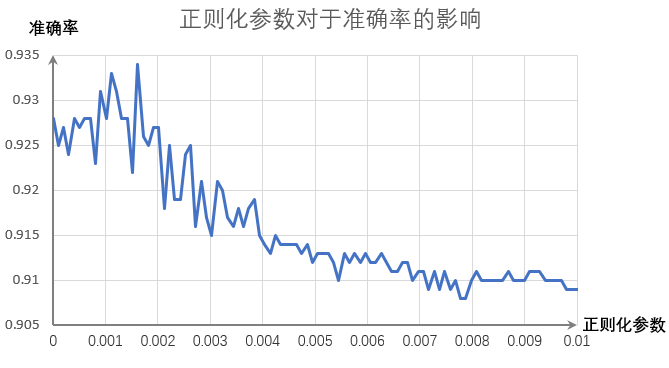


图4.9 不同正则化参数下的准确率示意图

从图4.7中可以看出，准确率的最大值出现在r为0.001到0.002之间。虽然曲线有些波动，但是总体来说，在r为0到0.001之间，准确率随着r的增大而增加；在0.001到0.01之间，准确率随着r的增大而减小

综上所述，结合4.1（2）部分内容，选取r=0.001最佳。

**（3）迭代次数对准确率的影响**

综合上述测试，固定学习率为1，正则化参数为0.001；变化迭代次数，测试不同迭代次数对于模型在测试集上准确率的影响。迭代次数的取值为：利用numpy库中的linspace函数，在10到2000中等间距取40个值。按照以上取值编写测试代码，并将测试的结果存放到“实验结果.xls”的“正则化参数对准确率的影响”sheet中。测试代码部分如图4.10所示，测试结果详见附件“实验结果.xls”。

在Excel中绘制在不同正则化参数的情况下，准确率随着正则化参数增加的变化情况，结果如图4.11所示。



图4.10 验证不同迭代次数对准确率的影响部分测试代码

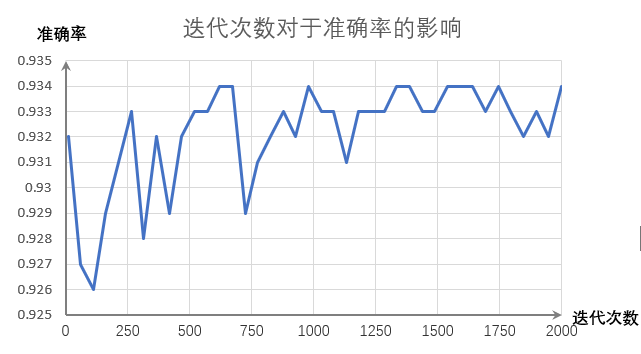


图4.11 不同迭代次数下的准确率示意图

观察图4.11可以发现，在迭代次数小于1000时准确率波动比较明显，这可能是由于在这个阶段正则化参数和学习率导致的Loss值不稳定，没有接近于收敛，但是这其中也出现了最大值，即正确率为93.4%。在迭代次数大于1000后，相对而言准确率比较稳定，徘徊在93.3%上下。

但是需要注意的时，由于本次实验的测试集中只有1000个元素，所以只要多判断对一个元素，那么整体的准确率就会提升0.1%，所以在图中看上去的巨大波动在测试集数量变多之后可能就不再如此明显。

## 五、小结及感想

### 5.1 实验小结

在本项目中，针对老师所给的数据集，我基于logistic回归算法，利用Python语言进行建模，较为成功的根据已经给出的数据预测其标签。同时，也调整学习率和正则化参数以及迭代次数，绘制出了Loss收敛曲线和准确率曲线，并经过分析得到了其相对的最佳取值，具体的分析过程在第四节中已经给出。

经过测试，结合Loss曲线的收敛速度以及稳定性，还有最终测试集上的准确率可以得到α的取值为1，r的取值为0.001时，效果比较好。。

在此α和r都相对确定的基础上上，我还测试了不同迭代次数对分类准确率的影响。发现在迭代次数较小时，准确率的波动比较明显；迭代次数较大时，分类准确率会稳定在一个值附近。经测试，发现这个值为93.3%，而最大的正确率为93.4%。

### 5.2 实验感想

在本项目的实现过程中，我也收获到了很多。首先增强了我对于单独枯燥的数学知识的理解，这是我第一次将比较纯粹的数学应用到实际的问题之中，在这个过程中我进一步理解了一些数学问题，比如为什么在代价函数中要加入正则项等等。也清晰的认识了所谓的模型也就是一些数学上的参数，在数学表达式中的取值不同也会导致最终模型的结果不同。

其次是通过使用python中使用的一些函数和包，通过代码的编写和调试，我理解了如何构建一个模型的方法，如何选择合适的数据结构对数据进行存储，以及将数学表达式转化为实际代码等问题。

经过本次项目，还提升了我的数据处理和分析的能力，我通过上网查阅相关资料，了解了xlwt库函数，将数据直接写入到Excel表格之中，大大减少了我的工作量。也学会了如何能够可视化地将变量对系统的影响直观清晰地表示出来，领悟了评价一个系统、一个算法好坏的标准不光是它分类的成功率，还有迭代速度，空间的占用等等。

除此之外我在实现的过程中也遇到了很多问题，首先有一个瓶颈就是我的数学推导能力，书上和网上的知识往往是广义且宽泛的，所以在针对一个特殊的问题时，往往需要将书上的公式适当变换，必要时需要将多个知识点进行融会贯通。不仅如此，Python语言实现这些数学规则的时候，与人在计算时的规则不同，需要借用入numpy库中的函数合理进行变换，而在程序出现问题时，实际上有的时候是因为计算机的浮点计算精度、公式代码标准中的特殊要求所导致的。就比如在最开始输出Loss值得时候，由于取ln值得内容没有经过特殊处理，导致经常输出错误代码，没有办法直接显示出Loss的值，在经过网上查阅资料之后这个问题才得以解决。

还有，测试数据的取值也是困扰着我的一个问题。想要很好的展现模型的特点就需要比较好的测试数据取值，我在一开始是自己选取数据取值，就是查阅网上的相关资料，根据一个区间来自己拟定取值范围，但是后来发现统计出来的数据不具有典型性，这是因为参数在变化很小的基础上准确率就有比较大的变化，所以最终统计出来的结果有些并不能真实的反映模型准确率的变化趋势。于是我查阅了相关资料，找到了numpy库中的linspace函数，可以做到在一个范围内等间距的取值，解决了我的相关问题。

总而言之，本项目进行的比较顺利，在完成的过程中也让我对老师上课所讲的理论知识有了更深刻的理解，真正做到了融会贯通。虽然机器学习的课程结束了，但是我还是会继续学习相关领域的知识，动手完成一些开源的项目来锻炼自己的能力，让自己变得更加优秀！

# 参考文献

[1] MonkyK，《机器学习总结（三）——损失函数》

https://blog.csdn.net/chkay399/article/details/81878157

[2] alexanderkun，《正则化为什么能防止过拟合》

https://www.cnblogs.com/alexanderkun/p/6922428.html

[3] 何坤，《计算机学习内部讲义》

[4] Rafal Skolasinski，《python divide by zero encountered in log - logistic regression》

https://stackoverflow.com/questions/38125319/python-divide-by-zero-encountered-in-log-logistic-regression

[5]Wikipedia，《Logistic regression》

https://en.wikipedia.org/wiki/Logistic\_regression

# 附录1 项目代码

1. **import** pandas as pd
2. **import** numpy as np
3. **import** xlwt

6. # 更新参数，训练模型
7. **def** train(x\_train, y\_train, epoch, learning\_rate, reg\_rate):
8. num = x\_train.shape[0]
9. dim = x\_train.shape[1]
10. bias = 0  # 偏置值初始化
11. weights = np.ones(dim)  # 权重初始化
12. bg2\_sum = 0  # 用于存放偏置值的梯度平方和
13. wg2\_sum = np.zeros(dim)  # 用于存放权重的梯度平方和
14. Epsilon = 1e-5
15. LossList = []
17. **for** i **in** range(epoch):
18. b\_g = 0
19. w\_g = np.zeros(dim)
20. # 在所有数据上计算梯度，梯度计算时针对损失函数求导
21. **for** j **in** range(num):
22. y\_pre = weights.dot(x\_train[j, :]) + bias
23. sig = 1 / (1 + np.exp(-y\_pre))
24. b\_g += (-1) \* (y\_train[j] - sig)
25. **for** k **in** range(dim):
26. w\_g[k] += (-1) \* (y\_train[j] - sig) \* x\_train[j, k] + 2 \* reg\_rate \* weights[k]
27. b\_g /= num
28. w\_g /= num
30. # adagrad
31. bg2\_sum += b\_g \*\* 2
32. wg2\_sum += w\_g \*\* 2
33. # 更新权重和偏置
34. bias -= learning\_rate / bg2\_sum \*\* 0.5 \* b\_g
35. weights -= learning\_rate / wg2\_sum \*\* 0.5 \* w\_g
37. # 在计算loss时，由于涉及到log()运算，因此可能出现无穷大，计算并打印出来的loss为nan
39. loss = 0
40. acc = 0
41. result = np.zeros(num)
42. **for** j **in** range(num):
43. y\_pre = weights.dot(x\_train[j, :]) + bias
44. sig = 1 / (1 + np.exp(-y\_pre))
45. **if** sig >= 0.5:
46. result[j] = 1
47. **else**:
48. result[j] = 0
50. **if** result[j] == y\_train[j]:
51. acc += 1.0
52. loss += (-1) \* (y\_train[j] \* np.log(sig + Epsilon) + (1 - y\_train[j]) \* np.log(1 - sig + Epsilon))
53. LossList.append(loss / num)
55. **return** weights, bias, LossList

58. # 验证模型效果
59. **def** validate(x\_val, y\_val, weights, bias):
60. loss = 0
61. acc = 0
62. num = x\_val.shape[0]
63. result = np.zeros(num)
65. **for** j **in** range(num):
66. y\_pre = weights.dot(x\_val[j, :]) + bias
67. sig = 1 / (1 + np.exp(-y\_pre))
68. **if** sig >= 0.5:
69. result[j] = 1
70. **else**:
71. result[j] = 0

74. **if** result[j] == y\_val[j]:
75. acc += 1.0
76. **return** acc / num

79. **def** main():
80. # 从csv中读取有用的信息
81. df = pd.read\_csv('数据集.csv')
82. # 空值填0
83. df = df.fillna(0)
84. # (4000, 59)
85. array = np.array(df)
86. # (4000, 57)
87. x = array[:, 1:-1]
88. # scale
89. x[:, -1] /= np.mean(x[:, -1])
90. x[:, -2] /= np.mean(x[:, -2])
91. # (4000, )
92. y = array[:, -1]
94. # 划分训练集与验证集
95. x\_train, x\_val = x[:2999, :], x[2999:, :]
96. y\_train, y\_val = y[0:2999], y[2999:]
98. ExcelFile = xlwt.Workbook()
100. # 1.不同学习率对于Loss的影响
101. **print**("\t验证学习率对于Loss的影响：")
102. epoch = 200
103. sheet = ExcelFile.add\_sheet('学习率对Loss的影响', cell\_overwrite\_ok=True)
104. learning\_rates = [0.01, 0.025, 0.05, 0.1, 0.25, 0.5, 1, 2.5, 5, 10]
105. reg\_rate = 0.001
106. # 准备写入工作
107. sheet.write\_merge(0, 1, 0, 0, '迭代次数')
108. sheet.write\_merge(0, 0, 1, len(learning\_rates), 'Loss的值')
109. **for** i **in** range(epoch):
110. sheet.write(i + 2, 0, i + 1)
112. **for** i **in** range(len(learning\_rates)):
113. **print**("正在计算学习率为%f时的Loss值" % learning\_rates[i])
114. sheet.write(1, i + 1, 'α = ' + str(learning\_rates[i]))
115. w, b, LossList = train(x\_train, y\_train, epoch, learning\_rates[i], reg\_rate)
116. **for** j **in** range(epoch):
117. sheet.write(j + 2, i + 1, LossList[j])
119. # 2. 不同正则化参数对于Loss的影响
120. **print**("\t验证正则化参数对于Loss的影响：")
121. epoch = 50
122. sheet = ExcelFile.add\_sheet('正则化参数对Loss的影响', cell\_overwrite\_ok=True)
123. learning\_rate = 1
124. reg\_rates = [0.0001, 0.001, 0.01, 0.1, 1, 10]
125. # 准备写入工作
126. sheet.write\_merge(0, 1, 0, 0, '迭代次数')
127. sheet.write\_merge(0, 0, 1, len(reg\_rates), 'Loss的值')
128. **for** i **in** range(epoch):
129. sheet.write(i + 2, 0, i + 1)
131. **for** i **in** range(len(reg\_rates)):
132. **print**("正在计算正则化参数为%f时的Loss值" % reg\_rates[i])
133. sheet.write(1, i + 1, 'r = ' + str(reg\_rates[i]))
134. w, b, LossList = train(x\_train, y\_train, epoch, learning\_rate, reg\_rates[i])
135. **for** j **in** range(epoch):
136. sheet.write(j + 2, i + 1, LossList[j])
138. # 3.不同学习率对于正确率的影响
139. **print**("\t验证学习率对于准确率的影响：")
140. epoch = 100
141. sheet = ExcelFile.add\_sheet('学习率对准确率的影响', cell\_overwrite\_ok=True)
142. learning\_rates = np.linspace(0.01, 20, 200)
143. reg\_rate = 0.001
144. # 准备写入工作
145. sheet.write(0, 0, '学习率')
146. sheet.write(0, 1, '准确率')
148. **for** i **in** range(len(learning\_rates)):
149. **print**("正在计算学习率为%f时的准确率" % learning\_rates[i])
150. sheet.write(i + 1, 0, learning\_rates[i])
151. w, b, LossList = train(x\_train, y\_train, epoch, learning\_rates[i], reg\_rate)
152. acc = validate(x\_val, y\_val, w, b)
153. sheet.write(i + 1, 1, acc)
155. # 4.不同正则化参数对于准确率的影响
156. **print**("\t验证正则化参数对于准确率的影响：")
157. epoch = 100
158. sheet = ExcelFile.add\_sheet('正则化参数对准确率的影响', cell\_overwrite\_ok=True)
159. learning\_rate = 1
160. reg\_rates = np.linspace(0, 0.01, 100)
161. # 准备写入工作
162. sheet.write(0, 0, '正则化参数')
163. sheet.write(0, 1, '准确率')
165. **for** i **in** range(len(reg\_rates)):
166. **print**("正在计算正则化参数为%f时的准确率" % reg\_rates[i])
167. sheet.write(i + 1, 0, reg\_rates[i])
168. w, b, LossList = train(x\_train, y\_train, epoch, learning\_rate, reg\_rates[i])
169. acc = validate(x\_val, y\_val, w, b)
170. sheet.write(i + 1, 1, acc)
172. # 5.不同迭代次数对于准确率的影响
173. **print**("\t验证迭代次数对于准确率的影响")
174. epochs = np.linspace(10, 2000, 40, dtype=int)
175. learning\_rate = 1
176. reg\_rate = 0.001
177. sheet = ExcelFile.add\_sheet('迭代次数对于准确率的影响', cell\_overwrite\_ok=True)
178. # 准备写入工作
179. sheet.write(0, 0, '迭代次数')
180. sheet.write(0, 1, '准确率')
182. **for** i **in** range(len(epochs)):
183. **print**("当前迭代次数为%d" % epochs[i])
184. sheet.write(i + 1, 0, float(epochs[i]))
185. w, b, LossList = train(x\_train, y\_train, epochs[i], learning\_rate, reg\_rate)
186. acc = validate(x\_val, y\_val, w, b)
187. sheet.write(i + 1, 1, acc)
189. # 保存实验结果记录表
190. ExcelFile.save('./实验结果.xls')

193. **if** \_\_name\_\_ == '\_\_main\_\_':
194. main()