**华中科技大学**

**计算机科学与技术学院**

**《机器学习》结课报告**



专 业： 计算机科学与技术

班 级： CS2006

学 号： U202015469

姓 名： 魏攀悦

成 绩：

指导教师： 张腾

**完成日期： 2022年 7 月 6 日**

目录

[1. 实验要求 2](#_Toc106106989)

[2. 算法设计与实现 3](#_Toc106106990)

[2.1 数据预处理 3](#_Toc106106991)

[2.2 对数几率回归实现 3](#_Toc106106992)

[2.3 决策树桩实现 4](#_Toc106106993)

[2.4 Adaboost算法实现 4](#_Toc106106994)

[3. 实验环境与平台 5](#_Toc106106995)

[4. 结果与分析 5](#_Toc106106996)

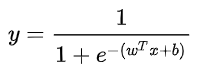
[5. 个人体会 6](#_Toc106106997)

Adaboost算法实现

# 实验要求

实验分别实现了以对数几率回归和决策树桩为基分类器的Adaboost算法。

## 对数几率回归思路及主要工作

对数几率回归和决策树桩是典型的二分类任务学习模型，是线性回归模型和梯度下降法两部分的结合。线性回归模型帮助我们用简单的线性方程实现了对数据的拟合，拟合只完成了回归任务，未完成分类任务。而对数几率回归是在线性模型的基础上做分类，这里只对二分类进行讨论，思路就是在线性模型外面套一个sigmoid函数，也即是。总的来说，对数几率回归模型就是在拟合这条直线，使得这条直线尽可能的将原始数据中的两个类别正确的划分开。

在实现对数几率回归的基分类器时，我首先通过控制训练轮数来调整准确率，发现改进不大。之后修改学习率的值来调整准确率，在调整后训练的速度和准确率在部分数据集上有明显提升，但针对个别数据集的准确率却有所下降，分析可能存在过拟合问题，采取用正确率来控制是否返回。这一点会在后面详细展开。

## 决策树桩思路及主要工作

决策树桩也称为单层决策树，它是一种简单的决策树，通过给定的阈值进行分类。从现实意义上来看，decision stump根据一个属性的一个判断就决定了最终的分类结果，比如根据鸢尾花花萼是大是小判断鸢尾花是否是优质鸢尾花，这方面体现的是单一简单的规则（或叫特征）在起作用。然而decision stump的预测准确率十分有限，仅可作为一个base learning algorithm进行一个初步分类。

在实现决策树桩基分类器时，将每个属性分别作为分类标准进行测试，通过遍历，改变不同的阈值，计算最终的分类误差，找到分类误差最小的分类方式，即为我们要找的最佳单层决策树。

## adaboost思路及主要工作

Adaboost是Boosting中较为著名的一种算法，而Boosting的主要思想是将弱的基学习器（性能不太好的学习器，比如一个准确率略微超过50%的二分类器）提升为强学习器。具体步骤如下：

1. 先将每个样本权重相等的训练集训练为一个初始的基学习器
2. 根据上轮基学习器的预测情况对训练集中样本权重做调整（例如提高被错分类的样本权重使他在下轮训练中得到更多的关注），然后据此训练一个新的基学习器
3. 重复2直到得到m个基学习器，m个基学习器的组合就是最终集成结果

可以看出，这是一个串行的过程。在实现adaboost的过程中，通过提高上一轮被错误分类的样本的权值，降低被正确分类的样本的权值来调整每一轮训练集中的样本权重。对错误率小的基学习器赋予较大的权值，而错误率大的基学习器赋予较小的权值，进行线性加权求和得到最终的集成学习器。

# 2. 算法设计与实现

## 2.1 数据预处理

### 2.1.1 数据缺失值填充

利用fillna和inplace函数对数据中缺失项填充平均值，保证数据的稳定性，降低分析过程中的出错概率。如图1所示



图1：平均值填充缺失项

### 2.1.2 数据归一化处理

在运行决策树桩模型之前，将targets里面的所有label归一化到｛-1，1｝，方便后续数据的划分。如图2所示



图2：数据归一化

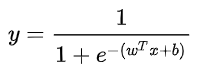
### 2.1.3 十折交叉验证

在最开始的时候利用sklearn中的kfold进行数据集的随机划分，之后按顺序输出到相应的.cvs文件中。但kfold随机划分因为具有随机性，在最后输出到.csv文件时需要根据数据前面的标号进行输出，实现过程复杂（而且老师不让调库），遂舍弃进行手动的顺序划分。

在实现手动十折交叉时，将数据集划分为相等的十份，做十次循环，在第i次寻汉中提取第i组数据作为测试集，剩下的九组数据作为数据集，进行一次模型训练。这样对于一组数据可以得到十个模型、十组准确率，取平均准确率能够较为客观反映出模型的整体准确率。

## 2.2 对数几率回归实现

由前面的思路可知，对数几率函数的形式如下：



这里我们需要用差值来度量他的预测准确度，那么优化的目标函数为：

IMG_256

这表示真实值和预测值的方差，值越小表示拟合效果越好，预测结果越准确。

用梯度下降法来实现上述的优化目标，那么优化的目标的一阶导数为：

IMG_256

根据梯度下降公式对参数进行迭代，就可以得到结果。

对数几率回归实现的伪代码如下：

·将权重w的每一列均置为1 #初始化参数w

·将属性矩阵data后增加一全1列 #参数w中有常数参数

·设置步长n=0.001，正确率rightrate=0.0

·while rightrate<0.75:

· c = sigmoid(np.dot(data, w)) #计算参数w下的预测值

· change = np.dot(np.transpose(data), c-label)

· w = w - change \* n #计算梯度下降，更新参数w

· rightrate = accuracy(test\_data, test\_label) #预测更新正确率

## 2.3 决策树桩实现

预测函数predict(self, x, i, threshold, less\_or\_great)

通过阈值比较对数据进行分类。所有大于阈值的数据会被分到类别1，而在另外一边的数据分到类别1.

决策树 fit(self, x, y, test\_data, test\_label, D)

通过遍历，改变不同的阈值，计算最终的分类误差，找到分类误差最小的分类方式，即为我们要找的最佳单层决策树。

决策树桩伪代码如下：

·将最小错误率err\_min设为inf（正无穷）

·for i in range(feature\_num): #对数据集的每个特征

· for threshold in values: #对数据集的每个阈值取值

· for less\_or\_great in ["lt", "gt"]: #对数据集的每个不等号

· 构建一颗决策树并进行测试

若错误率低于err\_min则当前决策树为最佳决策树

·返回最佳决策树

## 2.4 Adaboost算法实现

根据1.3，我们知道了adaboost的基本思想，则其具体的算法步骤如下：

1. 初始化样本权重
2. 对m=1,2...n重复以下操作的得到n个基学习器：
3. 用样本权重Dm训练数据得到第k个基学习器G（k）
4. 计算G（k）在加权训练数据及上的分类误差率
5. 计算G（k）在最终集成的学习器上的权重
6. 更新训练样本的权重值
7. 构建最终的集成学习器

Adaboost实现的代码：

fit\_sigmoid(*self*, *train\_data*, *train\_label*, *test\_data*, *test\_label*):

        len\_train\_data = len(*train\_data*)

*self*.D = np.ones(len\_train\_data)/len\_train\_data

        pred\_train = [np.zeros(len\_train\_data)]

*for* i *in* range(*self*.T):

*self*.clf = Sigmoid()

*# 以权重D训练ht*

*self*.clf.fit(*train\_data*, *train\_label*,

*test\_data*, *test\_label*, *self*.D)

            pred\_train\_i = *self*.clf.predict(*train\_data*)

            miss = [int(x) *for* x *in* (pred\_train\_i != *train\_label*)]

            e\_t = np.dot(*self*.D, miss)

*self*.alpha[i] = 0.5\*np.log((1-e\_t)/float(e\_t))

*# 更新权重*

            new\_miss = [x *if* x == 1 *else* -1 *for* x *in* miss]

*self*.D = np.multiply(

*self*.D, np.exp([float(x)\**self*.alpha[i] *for* x *in* new\_miss]))

*self*.D = *self*.D/sum(*self*.D)

在实现adaboost算法时，刚开始忽略了在权重d上训练基学习器，导致每个样本的权重相等，最后集成时的输出四个正确率相等。在思考和调试之后，发现没有加入权重d才导致这种结果。

另外实现adaboost时采取了两种实现方式，第一种是在基分类器外面计算权重d，加权错误率e\_t；第二种是在基分类器里面计算权重d和加权错误率e\_t。观察发现两种方式对运行时间产生的影响不大。

# 3. 实验环境与平台

Pyhhon版本：3.8.0

设备名称 LAPTOP-OCJ7S5SF

处理器 Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz 2.59 GHz

机带 RAM 16.0 GB (15.9 GB 可用)

系统类型 64 位操作系统, 基于 x64 的处理器

# 4. 结果与分析

Base=0 基分类器为对率回归时预测准确率：

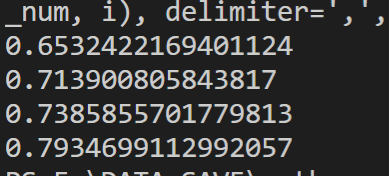


图3：base=0时准确率

Base=1 基分类器为决策树桩时预测准确率：

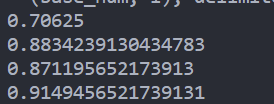


图4：base=1时准确率

实验结果与分析：

1. 对于对数几率回归：

这里采用测试正确率来控制迭代次数。在最开始的时候选择使用训练轮数来控制迭代次数，出现了两个问题：

第一，在十折交叉验证的过程中出现了正确率为0或为1的结果。这种极端的情况让我非常不解。在思考之后我觉得原因可能为：在划分数据集上，我采用顺序划分，而给出的label恰好前面一部分全1，后面一部分全0，使得分类器不能较好的学习特征、还有就是迭代次数的大小问题，过小可能对特征学习的不够，而过大则容易出现过拟合的情况。

第二，在取不同训练轮数时的结果差别很大，有的甚至出现了在相同训练轮数下100个基分类器的结果远远小于1个或5个基分类器的情况，如下表1。对此我也十分不解，但没有想到问题所在。

表1 训练轮数对准确率的影响

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 训练轮数 | 10 | 20 | 30 | 50 | 100 | 200 |
| 1个基分类器 | 0.5983 | 0.6611 | 0.5902 | 0.6559 | 0.6260 | 0.4307 |
| 5个基分类器 | 0.5983 | 0.7016 | 0.3978 | 0.4497 | 0.3032 | 0.2758 |
| 10个基分类器 | 0.5983 | 0.7016 | 0.3100 | 0.3790 | 0.5657 | 0.3964 |
| 100个基分类器 | 0.6983 | 0.7016 | 0.2130 | 0.2019 | 0.5961 | 0.3029 |

为解决上述两个问题，我决定采用测试正确率来控制迭代次数。但这又带来了一些弊端。

第一，训练时间过长。只有在正确率达到某一个设置阈值的情况下才会跳出，因此这需要多次学习，导致训练时间过长。

第二，每个模型的准确率固定在某个值附近。例如阈值设置为0.75，则每个模型的准确率都在0.75左右，这使得adaboost对样本权重的调整是有限的，最终的集成分类器的优化效果也是有限的。

如下表为学习率对准确率的影响

表2 学习率对结果的影响

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 学习率 | 0.0001 | 0.0005 | 0.001 | 0.005 | 0.01 | 0.05 |
| 1个基分类器 | 0.6532 | 0.6458 | 0.6428 | 0.6208 | 0.6194 | 0.6186 |
| 5个基分类器 | 0.7139 | 0.7065 | 0.6987 | 0.6897 | 0.6941 | 0.6750 |
| 10个基分类器 | 0.7385 | 0.7428 | 0.7446 | 0.7310 | 0.7249 | 0.7004 |
| 100个基分类器 | 0.7934 | 0.7862 | 0.7895 | 0.7625 | 0.7349 | 0.7219 |

1. 对于决策树桩：

在实现最佳决策树时，首先用的是以步长作为划分属性的依据之一，但正确率不高。在网上查阅资料并和同学讨论后，修改为以阈值取值作为划分依据，得到了更好的效果。

# 5. 个人体会

这是我首次完成机器学习的一整套项目，整套流程说起来简单，真正实现起来还是有一定的难度的，在不同的阶段都遇到了各型各色的问题。

比如在数据处理阶段，处理十折交叉验证时，对于顺序划分数据集，在取最后一份数据时，也按照前面的循环写就会报错。编译器显示取到的数组大小为（367，），但我的操作data[3312:3680,]取的数组范围是[3312,3379],确实是368。在查阅相关资料后发现，当输入的值超过边界大小时，编译器会自动取边界值3379，也就是等价于我输入的是data[3312:3379,]，而此时取不到最后一个元素，故报错。为此，当取最后一份数据时，我采用倒序取值，达到了想要的效果。

这种问题如果不是做实验得到，在课本上是无法注意到的。相对与课本的广泛与细致，实验能给我带来更多对于程序语言的运用层面的知识，这让我受益匪浅。

在完成对率回归模型时，我将常量b添加到特征矩阵的最后一列，进行合并运算。这样可以减少对b的单独运算，避免遗忘，也使代码更加简洁。但在这一过程的矩阵转置和矩阵乘法包括向量乘法，常常因为纬度不能对齐而报错。这方面的工作十分考验我的耐心和细心，而这些技能也是程序员所需要具备的。

在完成决策树桩模型时，刚开始的输出的准确率全部大于1，我非常惊奇。在进行检查后发现，我在分类前没有将target全部归一化到｛-1，1｝，而在做预测时以为自己已经做了归一化，才出现这样令人啼笑皆非的情况。

另外，对于这一整套项目，我在处理的时候尝试用工程项目的思路解决。由大的框架再到小的函数，一一解决。首先搭建好决策树桩模型和对率回归模型，之后填充adaboost模型，最后进行模型融合。

我还根据模型融合时得到的经验，进一步优化模型，通过调整正负样本的比例，成功再一次提高了正确率。  
 总的来说，本次实验比较考验我们的快速学习能力以及快速理解能力，提交截止日期延长之后减轻了一些压力，但还是很难能力有限的条件下进一步深挖相关知识。  
 但这也是一次还不错的机器学习入门体验，不断分析问题，优化参数，并看到自己的正确率不断提升还是很有成就感的。也让我知道了要做出一套准确的预测，需要有的不仅是精良的模型，还需要精炼到位的数据处理，四两拨千斤的参数调整，以及各个模块之间的协同作业，相互优化，这样才能让整个模型的预测效果不断提升。

大作业问题：

1.对ada算法的新认识：

在完成一整个ada算法的实现流程后，我想简单谈一谈与我而言ada算法的优缺点

优点：

（1）Adaboost提供一种框架，在框架内可以使用各种方法构建子分类器。可以使用简单的弱分类器，不用对特征进行筛选

（2）Adaboost算法不需要弱分类器的先验知识，最后得到的强分类器的分类精度依赖于所有弱分类器。无论是应用于人造数据还是真实数据，Adaboost都能显著的提高学习精度。

（3）Adaboost对同一个训练样本集训练不同的弱分类器，按照一定的方法把这些弱分类器集合起来，构造一个分类能力很强的强分类器

缺点：

在Adaboost训练过程中，Adaboost会使得难于分类样本的权值呈指数增长，训练将会过于偏向这类困难的样本，导致Adaboost算法易受噪声干扰。此外，Adaboost依赖于弱分类器，而弱分类器的训练时间往往很长。

2.基分类器类型、超参数设置对最终模型性能的影响

参照前文，已经提及。