**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： CS2011

学 号： U202014774

姓 名： 王逸

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 2](#_Toc106106990)

[2.1 数据预处理 2](#_Toc106106991)

[2.2 对数几率回归实现 2](#_Toc106106992)

[2.3 决策树桩实现 3](#_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 7](#_Toc106106997)

Adaboost算法实现

# 实验要求

本次实验要求实现分别以对数几率回归和决策树桩作为基分类器的AdaBoost算法，并观察在不同类型的分类器和不同的分类器数目1，10，50，100（即AdaBoost算法中的迭代次数）对预测结果正确率的影响，并将在不同数目基分类器条件下的10折交叉验证的预测结果输出至指定文件下。

在本次实验中，我依次完成了以下工作：

1. 实现决策树桩算法，通过循环依次修改阀值的维度对数据集中指定维度数据小于阀值的标签根据符号修改为1或-1，得到某次循环的的预测数组。再根据每个数据的权重计算权重错误率，若比当前的最小错误率小则修改最小错误率以及本次adaboost迭代获得的最佳决策树桩
2. 实现对数几率回归算法，通过调整学习率的固定值以及正则系数alpha的值，获得了不同参数下的预测结果，并选取了正确率最高进行最终的运行。
3. 实现AdaBoost算法以及十折交叉验证，并将最后的结果输出至指定文件

# 2. 算法设计与实现

## 2.1 数据预处理

在最初读入数据的时候，将label的0和1统一修改为修改为1和-1，有利于后面sign函数的使用以及判断。

## 2.2 对数几率回归实现

逻辑回归也称作logistic回归分析，是一种广义的线性回归分析模型，属于机器学习中的监督学习。其推导过程与计算方式类似于回归的过程。

其核心公式由函数sigmoid(z)给出：

IMG_256

我们可以设函数为：其中w为各特征值的权重值组成的权重向量（与AdaBoost中的D不同），b是截距；本次实现中，我将b和w拼接在一起形成的theta，其中theta[0]=b,theta[1:]=w

IMG_256

对数几率回归需要求解的最优问题的w在本次实验中采用梯度下降法求解，梯度表达式由函数g\_reg()给出：

def g\_reg(self,x\_b,y,theta,l=2):

out = self.sigmoid(x\_b.dot(theta)) # 计算sigmoid函数的输出结果

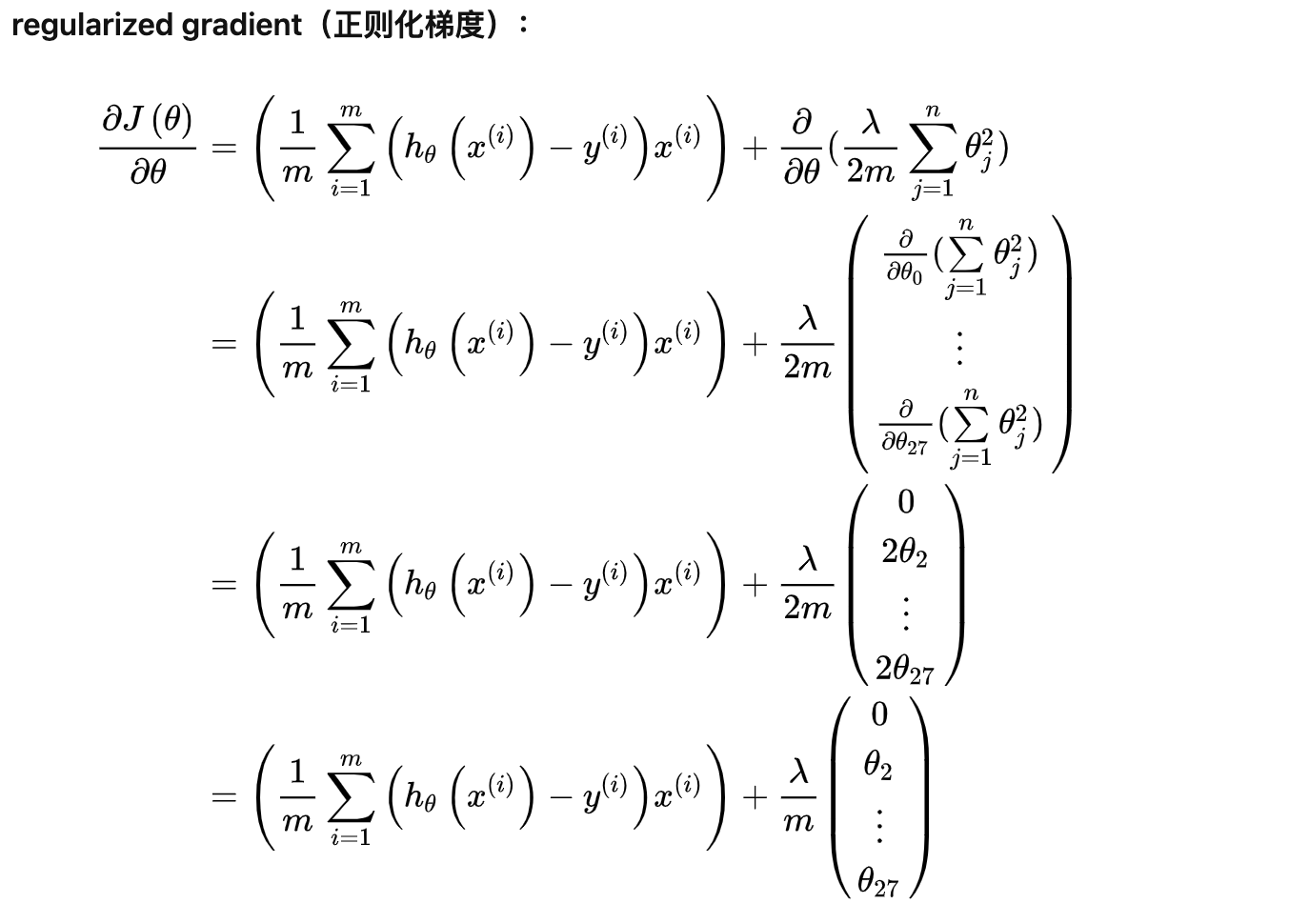
a=(x\_b.T) @ (out - y) / len(x\_b)

regularized\_term = (l/x\_b.shape[0]) \* theta

regularized\_term [0,0]=0.

return a+regularized\_term

返回值中的a是不考虑正则化情况下即lambdas=0时的梯度，而regularized\_term则是L2正则化项表达式，推导过程如下：



梯度下降的训练过程由函数g\_descent()给出，通过测量参数向量theta相关的损失函数的局部梯度，并不断沿着降低梯度的方向调整，直到梯度降为0，达到最小值。参数列表中的l是正则系数lambda，通过对其进行修改调试最终选择l=8.

训练一个基于对数几率回归实现的基分类器，首先调用fit()函数得到一个已经训练好的模型lr，再调用lrTrain()函数得到AdaBoost所需的最佳分类器参数数组，权重错误率以及预估数组。

## 2.3 决策树桩实现

决策树桩DecitionTree部分的实现定义了两个函数：

1. stumpClassify(dataMatrix,dimen,threshVal,threshIneq):

通过传入进来的维度以及阈值比较对预测数据集进行预估分类，返回预测结果。用于在训练找到某次循环中数据集上的单层最佳决策树，也可以用于Adaboost最后加权分类器预测结果的获得。仔细观察该函数的参数列表也可以得到最佳单层决策树需要保存的参数信息有哪些。

:param dataMatrix: 数据集

:param dimen: 维度

:param threshVal: 阈值

:param threshIneq: 不等号

:return:类别估计数组

该部分代码如下：

#按列数创造一个类别估计数组，初始值为1

retArr=ones((shape(dataMatrix)[0],1))

#第dimen列的值小于/大于threshVal置-1，分类准则

if threshIneq == 'lt':

retArr[dataMatrix[:,dimen]<=threshVal] = -1.0

else:

retArr[dataMatrix[:,dimen]>threshVal] = 1.0

return retArr

1. buildStump(dataArr, classLabels, D):

buildStump()首先遍历样本特征；然后对于每个特征，设置不同的门限和不等式类型，比较分类结果；计算加权分类误差，，得到最佳单层决策树。

找到数据集上最佳的单层决策树，确定数据集中的步数numSteps=10。由于本次数据集的每个数据的特征是按列给出，所以该函数内部的第一层循环应按列循环遍历维度；根据步数确定遍历步长，并进行第二次循环寻找阀值；内部的第三次循环则是根据比较符号对数据集进行标签划分。

三次循环确定一个决策树桩的基本信息，由传进来的Adaboost权重向量D计算权重错误率，若比当前的最小错误率小，则更新当前的最佳树桩参数信息。

:param dataArr:数据集

:param classLabels:标签

:param D:权重向量

:return:最佳单层决策树相关信息字典、错误率、类别估计值

## 2.4 Adaboost算法实现。

Boosting算法是一种通过多次学习来提高算法精度的方法，因为采用综合的原则使得算法的效率能够得到明显的改善，将弱分类器提升为强分类器。AdaBoost 提高前一轮弱分类器错误分类样本的权值，降低那些被正确分类的权值。对于弱分类器的组合，AdaBoost采取加权多数表决的办法，即加大分类误差率晓得弱分类器的权值，使其在表决中起到较大的作用，减少分类误差率大的弱分类器的权值，使其在表决中起到较小的作用。

其优点在于泛化错误低，，适用度高，不用进行参数的调整；但它对数据的异常十分敏感，适用于数值和标称值。

Adaboost的步骤如下：

1. 统计训练数据集中的样本数量m，并赋予每个数据一个初始权重。在Adaboost开始阶段，每个数据集的权重相等，那么每个数据的权重Di=1/m，这些权重构成了权重向量D=[D1,D2,D3 ,...Dm]
2. 在训练集上训练出一个弱分类器并计算该分类器的错误率e，并根据e来计算该弱分类器的权重α，并根据该分类器的预测结果和α对权重向量D进行更新。α和D的计算公式如下：

IMG_256

IMG_256

IMG_256

1. 然后在同一数据集上再次训练弱分类器。第二次训练当中,将会调整每个样本的权重, 其中第一次分对的权重会降低,分错的权重会提高。
2. 依次重复2-3步，知道迭代次数满足指定值或者错误率为零后得到一个分类器数组，最后根据每个分类器的权重αi对分类器的预测结果进行加权求和，得到最终的分类结果

算法流程如下：

FOR EACH Iteration:

调用Train（）计算最佳单层分类器

将最佳单层分类器加入单层分类器列表

计算$\alpha$

计算新的权值向量$\mathbf{D}$

更新累积类别估计

IF 错误率 == 0

BREAK

END IF

END FOR

在Adaboost的fit部分定义了两个函数，分别是：

1. adaTrain(dataArr,classLabels,iter):

完成基于单个分类器的AdaBoost训练过程，在此过程中完成单层分类器的选择，分类器权重α的计算，数据集权重向量D的更新，当迭代次数等于iter或者错误率为0时中止循环完成训练。参数列表意义如下：

:dataArr: 数据集

:classLabels: 标签

:iter: 基分类器数目

:return:多个弱分类器，包含其对应的alpha值

1. adaClassify(dataToClass,ClassifierArr):

利用训练出的多个弱分类器，在测试集上进行分类预测，对每个分类器预测的结果乘以相应的加权值α后进行求和并返回sign（pre）

: datToClass:一个或多个待分类样例

: classifierArr:多个弱分类器组成的数组

:return:预测的类别值

# 3. 实验环境与平台

Python版本：Mac系统下Python3.10.1 64-bit

GPU：2 GHz 四核Intel Core i5

CPU：16 GB 3733 MHz LPDDR4X

平台：VsCode 1.63.0（Universal）

# 4. 结果与分析

在这里给出十折交叉验证的预测结果及正确率，并进行一定的分析。在这里，由于在对数几率回归上花的时间较多，所以着重分析了其相关参数对实验结果的影响。

1. 以决策树桩作为基分类器，不同分类器数目对预测结果正确率的影响：

从0-3依次为iter=1，5，10，100的十折交叉验证结果；可以看出在以决策树桩作为基分类器的情况下，分类器数目对结果有显著影响。

1. 以对数几率回归作为基分类器，不同分类器数目对预测结果正确率的影响：

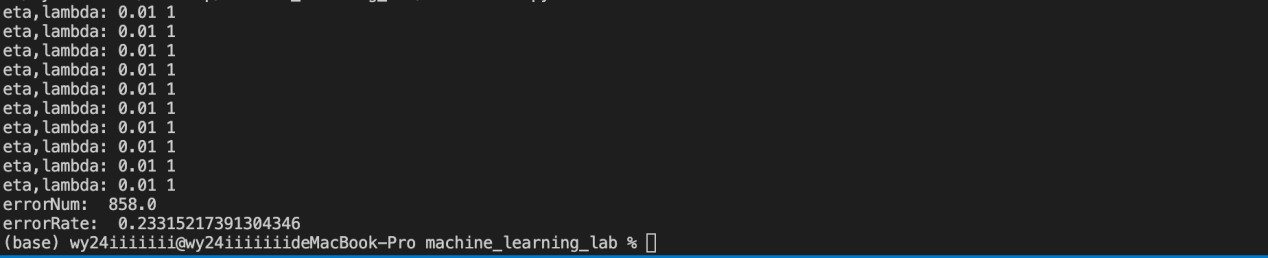
依次为iter=1，5，10，100的十折交叉验证结果；从结果来看在以对数几率回归作为基分类器的情况下，可能由于实现的代码原因，分类器数目对结果影响效果并没有想象中的显著。



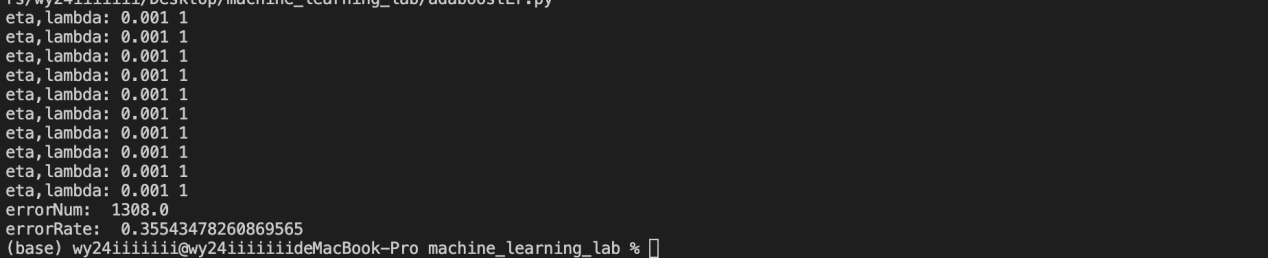


1. 以对数几率回归作为基分类器，不同的学习率eta和正则系数lambdas对预测结果正确率的影响：

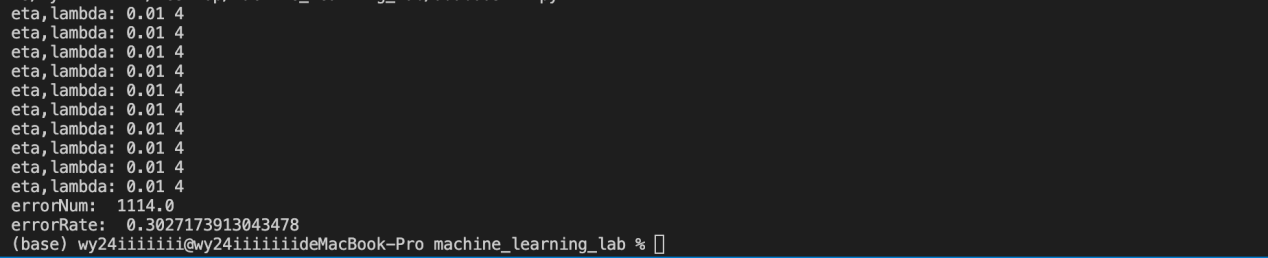
Eta=0.01,lambda=1



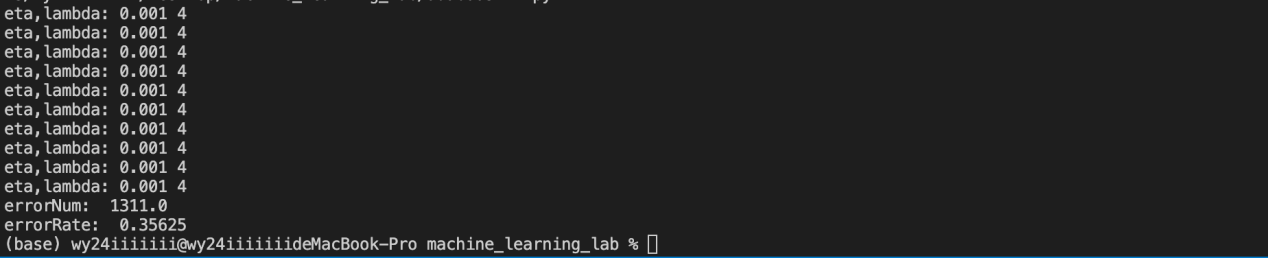
Eta=0.001,lambda=1,错误率提高



Eta=0.01,lambda=4,错误率提高，但较eta改变带来的错误较少



Eta=0.001,lambda=4,错误率提高，但与情况2相比变化较少



# 5. 个人体会

本次实验中遇到的最大的困难是将对数几率回归的梯度与AdaBoost算法中的样本权重D进行结合，从而实现AdaBoost的加权多数表决；还有比较麻烦的一点是对于正则化系数的选择。

对于第一个问题，通过在网上查询了相关资料后，阅读了python中对数几率回归模型有关样本权重的底层代码，结合了ppt中标签值为{1,-1}的梯度函数的表达式，将样本权重D乘到梯度函数中，实现加权筛选。在LR中选择该层最佳的分类器参数时，通过将D与损失函数相乘得到加权损失wl，通过将wl与目前的最小错误率进行比较，若小于则更新自身的best列表，从而实现筛选。

对于第二个问题则是通过尝试老师给出的正则系数范围，挑选出了正确率最高的lambda=8，作为最终的参数。

但是这次实验由于个人能力的不足在以对数几率回归作为基分类器时运行所需的时间较长，而且正确率也不尽如人意，希望在之后的学习中能做到对其的进一步改善，提高学习的性能和最终的正确率。