**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：

学 号：

姓 名：

成 绩：

指导教师： 何琨

**完成日期： 2022年 7 月 5 日**

目录

[1. 实验要求 2](#_Toc107763644)

[2. 算法设计与实现 2](#_Toc107763645)

[2.1 数据预处理 2](#_Toc107763646)

[2.2 决策树桩实现 2](#_Toc107763647)

[2.3 对数几率回归实现 3](#_Toc107763648)

[2.3.1梯度下降法求参数w的值 5](#_Toc107763649)

[2.3.2随机梯度下降法求w值 5](#_Toc107763650)

[2.3.3改进的随机梯度下降算法 6](#_Toc107763651)

[2.3.4正则化处理实现对数几率回归 6](#_Toc107763652)

[2.4 Adaboost算法实现 7](#_Toc107763653)

[3. 实验环境与平台 8](#_Toc107763654)

[4. 结果与分析 8](#_Toc107763655)

[5. 个人体会 10](#_Toc107763656)

Adaboost算法实现

# 实验要求

实验要求实现分别实现以决策树桩和对数几率回归为基学习器的AdaBoost算法基分类器数目取 1，5，10，100 这四种数值。

在决策树桩部分，自己实现的时最简单的决策树桩，选择所给属性中的一个属性挑选阈值最为划分依据。

在对数几率部分，自己先后尝试了梯度下降法、随机梯度下降法、正则化实现对数几率。

自己在最后也尝试分析基学习器数目对AdaBoost算法影响以及对数几率回归中学习率对最后准确率影响。

# 2. 算法设计与实现

## 2.1 数据预处理

关于数据处理，自己所想的有两类处理方式，第一类时归一化处理，将数据大小限制在[0,1]内，自己没有调用库实现，具体实现的思路是对57个属性每一个属性进行处理，选则每一个属性中最大值于最小值分别为max、min，当前数据值为data，则将数据处理为data = 。

第二类是选择部分属性训练模型以及预测数据。实现思路为计算每一类属性与标签之间的相关性系数（我所选择的是计算皮尔斯相关系数），对每种属性的的相关系数进行升序排列，选取其中相关性最高的30个属性进行模型训练。

在实现决策树部分时，自己首先进行了数据处理，随后发现数据不需要进行处理，不处理数据的十折交叉验证准确率明显更高于处理数据后。我认为是因为决策树不像对数回归几率算法，决策树的实现是依据某一属性值进行划分跟归一化关系并不大。但是实现决策树桩还是需要进行数据处理的，决策树桩的分类为{-1,1}，而所给的标签值为{0,1}，因此需要在训练时将0替换为-1，而且在adaboost里计算每个基α时也需要值为{-1,1}，所以这一步处理是必要的，在adaboost最后预测数据后应该再将-1改为0。

在实现对数几率回归算法时，使用了上述的两种数据处理方式，发现对于准确率有更大提升的是计算皮尔斯系数后进行排序选择部分属性训练模型，而归一化处理对于准确率的提升不太明显，但是可以避免sigmoid函数运行时发出数据可能溢出的警告。另外，对数几率作为基学习器实现adaboost时应该同决策树桩一样，将标签值中的0换位-1，最后的预测数据再将-1替换为0。

## 2.2 决策树桩实现

决策树桩，也称单层决策树，它是一种简单的决策树，通过给定的[阈值](https://so.csdn.net/so/search?q=%E9%98%88%E5%80%BC&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/qq_42148307/article/details/_blank)进行分类，举例如图2.1。

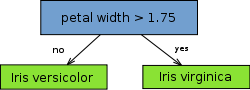


图2.1决策树桩的举例

从树（数据结构）的观点来看，它由根节点（root）与叶子节点（leaves）直接相连。用作分类器（classifier）的 decision stump 的叶子节点也就意味着最终的分类结果。

在自己实现决策树桩类时，构建了两个函数，分别为stumpClassify与fit函数，其中stumpClassify函数用于进行分类，依据所给的值进行分类大于为1，小于为-1。

另一个函数为fit函数，因为决策树的原理是根据一个值进行二分，所给数据有不止一个属性，因此函数的实现应该有三层循环，对每一个属性的每一个值进行判断，计算其错误率，最后选择错误率最小的决策树构建单层决策树，即选择某一个属性的使得错误率最小的值作为阈值，具体的实现见如下伪代码（图2.2），此外决策树桩的分类值为{-1,1}，而所给的标签值为{0,1}，所以将-1的值替换为0，这也方便了adaboost的实现。

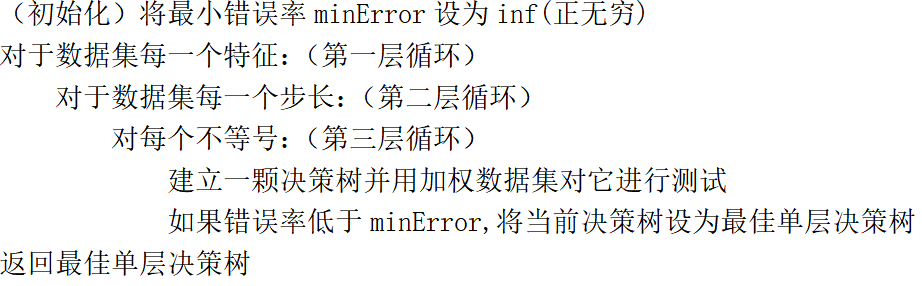


图2.2单层决策树的伪代码表示

实现单层决策树后对所给数据进行测试可以发现，该决策树桩的最小错误率与预测值没有改变，所选作为阈值的属性也没有改变，但是阈值却发生了变化，因为归一化数据改变了绝对大小但是没有改变相对大小。所以归一化对于决策树桩的实现并没有很大的用处。

同时，计算皮尔斯系数进行处理对于其准确率影响也不大，因为决策树桩就是一个比较单一的弱分类器，比较的数据也是相对大小，计算皮尔斯系数是为了选择相关系数最高的属性，可是如果该属性的相关性很高，决策树桩也会选择出该属性，因此进行这种数据处理是意义不大的，可以缩短决策树桩选择的时间但是不会提高准确率，而且计算系数也很耗时间，所以最后也没有采用这种数据处理方式。

综上所述，决策树桩的实现较为简单，对于数据也无需进行很多处理只是将标签值中的0替换为-1即可，但是决策树桩的准确率并不高，这也是其一个缺点。

## 2.3 对数几率回归实现

对数几率回归该模型在设计之初是用来解决0/1二分类问题，虽然它的名字中有回归二字，但只是在其线性部分隐含地做了一个回归，最终目标还是以解决分类问题为主。

线性回归是找到一条直线或超平面尽可能地接近所有的训练数据点（就是用线性方程来拟合数据），而对数几率回归是找到一条直线或超平面尽可能地分开两种不同类别的数据点（就是在公式中的线性部分来做了回归）。

首先，我们要解决的问题是：在线性模型上做二分类（这里不讨论多分类）。

把问题转化为，让模型输出为0或者1，而且在分界处变化很陡。

直接想法是套一个函数来实现一个单位阶跃函数，如下：

也就是把线性模型看作为一个两种类别的分界线。

由于分段函数性质太差，不符合优化问题的目标函数要连续可微的特点。所以我们找一个形似的函数，Sigmoid 函数（S型函数）中的杰出代表——对数几率函数（一个任意阶可导的凸函数，有良好的数学性质，很适合优化问题）。将线性模型代入就得到总的模型为。

对于sigmoid函数，其表达式为，使用python绘制出其图像如图2.3所示。其实，对数几率回归模型就是在拟合 线性模型，使得这条直线尽可能地将原始数据中的两个类别正确的划分开。

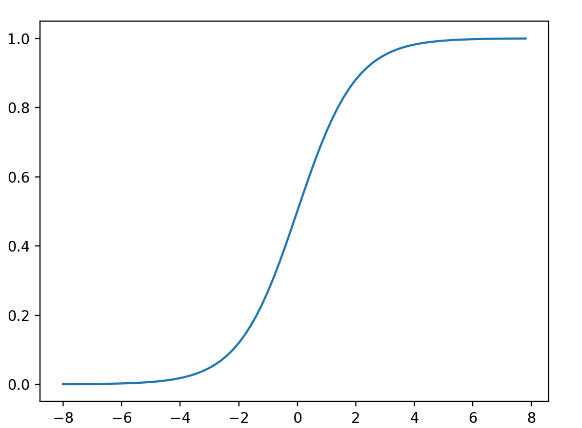


图2.3sigmoid函数图像

因此可以得到 ……（1）

……（2）

将（1）、（2）合并起来可得：

……（3）

（3）式求出了样本标记的分布律，便可以通过极大似然法来估计分布律中的参数*w*。写出极大似然函数：

……（4）

（4）式两边取对数可得：

……（5）

将（5）式在前面乘上负数因子便可以得到对数几率回归的代价函数：

……（6）

通过最小化上述代价函数（6）便可以估计出参数w的值。

最小化有很多方法，我自己分别尝试了梯度下降法、随机梯度下降法以及正则化处理。

### 2.3.1梯度下降法求参数w的值

梯度下降法是一种常用的一阶优化方法，是求解无约束优化问题最简单、最经典的方法之一。

对（6）式函数J（w）求导可得：

……（7）

因此w变为：

……（8）

为学习率，是我们自己设置的值，的不同会导致最终预测结果的不同，我们可以设置一个循环次数，不断地进行梯度下降从而去寻找最合适的w值。

以上为理论知识，具体实现时首先对于所给数据进行处理，因为最后线性回归会有常数项，所以将所给数据扩展一列，变为58列，最后一列填充为1。该算法的伪代码如下：

每个回归系数（w）初始化为1

重复epoches次：

    计算整个数据集的梯度

    使用 alpha \* gradient（dw） 更新回归系数的向量

返回回归系数

使用所给数据取其中386到772作为测试数据进行简单的测试如图2.12所示，得到输出准确率如图2.4所示。



图2.4测试输出的准确率

可以发现，该对数几率回归的还是比较高的，因为我的学习率跟循环次数设置的较为合适，另一个原因是测试数据较小，模型拟合比较好，如果测试数据取全部数据的话，会发现准确率并不是很乐观。同样是上述数据，进行归一化以及选取最相关的30个属性进行预测输出准确率如图2.5所示：



图2.5数据进行处理后的准确率

处理数据后模型的准确率得到了提升，这是因为选取了最相关的属性并且进行了归一化处理，使得数据与模型的拟合效果更好。

### 2.3.2随机梯度下降法求w值

梯度下降法”的不足在于每次更新回归系数时都要遍历整个数据集，计算复杂度高因此，产生了“随机梯度下降法”，即每次仅使用1个样本点数据来更新回归系数。其伪代码如下：

随机梯度下降算法：一次仅用一个样本点来更新回归系数

随机梯度下降伪代码：

每个回归系数初始化为1

对数据集中每个样本：

计算该样本的梯度

使用 alpha \* gradient 更新回归系数的向量

返回回归系数

对该方式下的对数几率回归进行测试，同2.3.1输出其准确率如图2.6所示。

8

图2.6随机梯度下降算法准确率

该准确率与梯度下降的准确率差别并不大，但是能发现程序执行速度明显快了很多，因为随机梯度下降一次只选取一个样本更新数据。但是我们对随机梯度下降算法同样进行多次迭代，发现问题如下：

不足在于回归系数存在周期震荡波动。

因此，产生了“改进的随机梯度下降法”，改进之处有两点：

 1、动态调整步进因子；

1. 随机选取样本来更新回归系数。

### 2.3.3改进的随机梯度下降算法

改进的两点在于动态调整学习率的值并且每次随机选取样本更新回归数学习率的值动态依据步长改变，一定程度上避免了固定学习率导致的最后不收敛问题，而随机的梯度也缩短了运算时间并且避免了偶然性，使得算法更精确。调整为 = 4/(1.0+j+i)+0.01。i为数据数量，j为循环次数。

经过测试，该算法的准确率确实优于前两个而且运算时间上也有了一定的缩短，是比较合适的一种实现对数几率回归的方法。

### 2.3.4正则化处理实现对数几率回归

在模型训练中，其实很容易出现过拟合的情况，所以为了缓解这种情况轻易的发生，我们需要对代价函数做一些改变。在模型的训练过程中，没一次训练，都会有一组权重参数产生，我们给所有组的权重参数同时加上一个相对相同的值，其实并不影响谁最后是最小值，同加同减，最后最小的还是最小的。所以我们决定为代价函数加上一点点惩罚，就是当模型出现过拟合而导致代价很低时，我们要加上一个很大的数，将代价值提起来，来告诉模型，这并不是我们想要的。这就是正则化项的功能。对于恰好拟合的参数来说，加入正则化项，其值对代价整体并没有很大的影响。

正则化有多种方式，包括L0（向量中非零元素个数），L1（向量中元素绝对值之和），L2（向量的模）。但是L0范数的求解是个NP完全问题，而L1也能实现稀疏并且比L0有更好的优化求解特性而被广泛应用。L2范数指各元素平方和后开根的值，可令每个元素接近于0，虽然不如L1更彻底地降低模型复杂度，但是由于处处可微降低了计算难度。

L\_0范数：……（9）

L\_1范数： ……（10）

L\_2范数： ……（11）

在这里采用了L2正则化，即为原来的代价函数加上所有参数theta的均方和，因为在发生过拟合的时候，曲线过于弯曲会导致某一项的theta值很大，所以加上theta的均方和，会使即使过拟合的代价不大。对于对数几率回归要将w的公式加上正则化项，即（8）式更正为：

……（12）

更改后使用之前的测试数据进行测试，测试后输出错误率与准确率如图2.7所示：

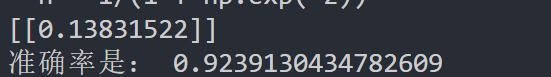


图2.7测试输出数据

这里测试仍然使用的是较小的数据规模，可以看到跟前面的算法预测效果差别不大，但是当数据规模变大后效果明显更优，使用正则化处理后若使用所给的所有数据进行预测，会发现其准确率高达0.8616847826086956（图2.8），而之前没有正则化处理的对数几率回归预测全部数据的准确率只有0.4157608695652174（图2.9）。

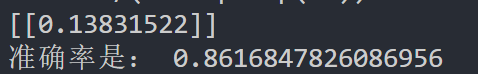


图2.8正则化处理后预测全部数据准确率

5

图2.9未正则化处理预测全部数据准确率

可以明显看到正则化有效的处理了过拟合问题，当数据规模较大时，会导致模型过拟合因此最后的预测效果也并不是很好，而正则化一定程度上削弱了过拟合问题，使得模型与数据拟合较好，最后的预测结果也较为理想。

## 2.4 Adaboost算法实现

Boosting是一族可将弱学习器提升为强学习器的算法，弱学习器是指泛化性能略优于随机猜测的学习器，例如在二分类问题上精度略高于 50% 的分类器。

该族算法的工作机制是先在初始训练集上训练出一个基学习器，再根据基学习器的表现对训练样本的权重分布进行调整，使得先前基学习器预测错的样本在后续受到更多关注，然后基于调整后的权重分布训练下一个基学习器；如此重复进行，直至基学习器数目达到事先指定的值 T，最终将这 T 个基学习器进行加权线性组合。Adaboost是该族算法最著名的代表。一张图解释Adaboost的算法流程如图2.10所示：

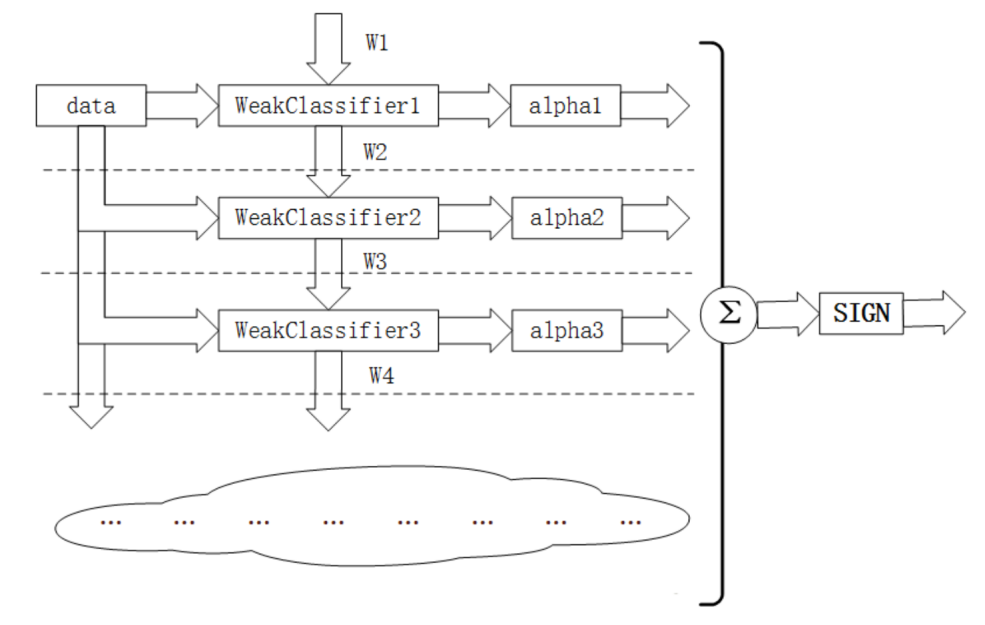


图2.10Adaboost算法流程

自己在实现Adaboost算法时，分别实现了决策树桩为基以及对数几率回归为基的Adaboost，具体的实现是将基学习器“套进”Adaboost的模板中，值得注意的是所给数据的标签值是{0,1}将0替换为-1，方便后续的计算，基学习器具体的实现在前面已经介绍，在这一部分主要是将两者结合起来。

# 3. 实验环境与平台

Python版本为3.10.5，使用vscode编译运行代码。

# 4. 结果与分析

实现以决策树桩为基的Adaboost算法后进行十折交叉验证输出的结果如图2.11所示：

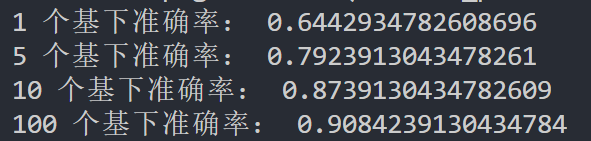


图2.11决策树桩为基的Adaboost算法不同基下十折交叉验证准确率

实现以对数几率回归为基的Adaboost算法后进行十折交叉验证输出的结果如图2.12所示：

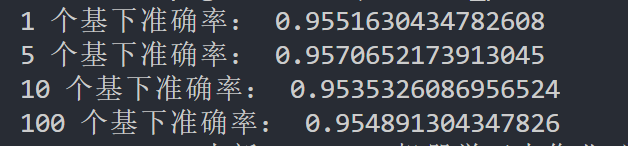


图2.12对数几率回归为基的Adaboost算法不同基下十折交叉验证准确率

能明显发现对于以决策树桩为基的Adaboost而言其准确率随着基的数目增加而逐渐增加至收敛，可是以对数几率回归为基的Adaboost其准确率随着基的数目增加并不会有明显的提升，四个基的预测结果相差不大。

我认为对数几率回归是一种“较强的”分类器，所以他很快收敛，基的数目影响并不是很大。使用自己的模型也绘制出了决策树的基分类器数目与准确率的关系如图2.13所示：

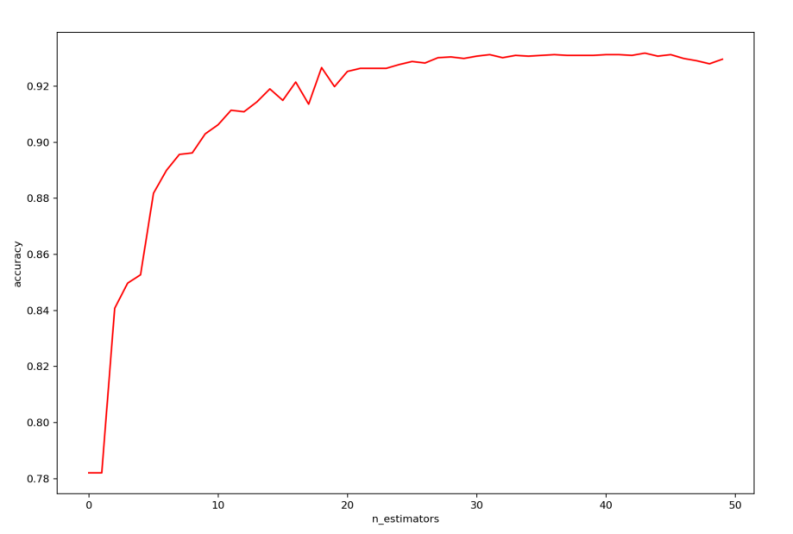


图2.13决策树桩为基的基分类器数目与准确率关系图

可以发现自己所实现的决策树桩为基的Adaboost算法确实准确度、收敛速度上都较差，而且迭代过程中还有抖动，在基数目为30左右开始收敛。

关于学习率的问题，我发现并不是学习率越小越好或者越大越好，要依据数据而定，自己实现了不同数目基分类器下学习率对于对数几率回归准确率的影响如图2.14所示。

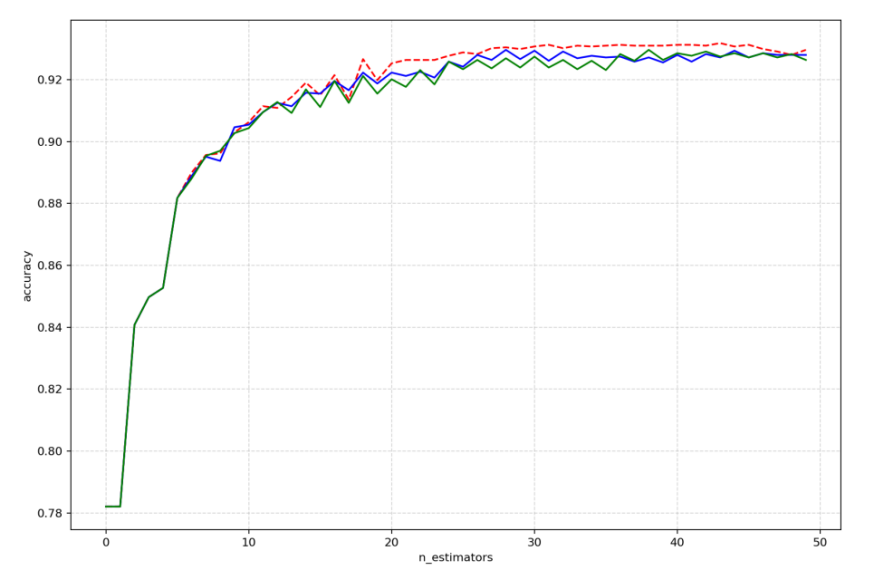


图2.14不同基数目下不同学习率对对数几率回归实现的Adaboost准确率的影响

最后回答问题老师所抛出的问题，通过此次实验，自己算是对于Adaboost算法有了初步的了解，也粗略明白了其思想，发现Adaboost算法颇有“三个臭皮匠顶个诸葛亮”的韵味，该算法对于基分类器的选择没有要求，并且经过该算法后可以避免过拟合，构成了一个强分类器，基分类器类型若是弱分类器就会发现数目越多准确率越高并且收敛较慢，较强的分类器会发现收敛较快而且拟合较好，超参数的的不同会导致准确率的不同，另外超参数不是单一的越大或者越小越好，是需要依据模型以及数据去调整的，上述中自己发现了学习率对于准确率的影响，此外决策树桩的选择步长也有一定的影响，总而言之参数不同准确率不同，至于参数的选择，自己没有发现好的方法，都是不断调整去寻找较为合适的参数的。

# 5. 个人体会

自己在做此次实验时遇到的第一个问题也是最基础的问题就是对于python的语法掌握不太熟练，虽然平时也有做头歌的实验，可是实验中只是很短的代码而且很多是调库实现，自己还没有真正的去实现一个用python语言写的项目，这对于我来说是一种考验，我去看些python语法的视频，另外在写实验时出现报错或者bug就及时上网搜，磕磕绊绊地去实现实验；第二个问题是时间的问题，因为6月底7月初有很多考试，而对于初学者此次作业并不是一天两天能完成的，所以时间不充裕是一大难题，所幸老师善解人意将交作业时间推迟到了7月6号，此乃英明之举让我们能安心备考也能认真完成作业，感谢老师！第三个问题是自己对于机器学习是初学者，不掉库去实现决策树以及对数几率回归两个基分类器时经常出现bug，很多矩阵以及数组或者列表的转换自己搞不明白就晕住了，自己还是多上网搜解决这一问题；最后一个问题是对数几率回归那里卡了好久，最后与助教以及同学沟通交流才知道是模型过拟合的问题，使用正则化处理后解决了这一问题，感谢帮助过我的助教和同学们！

对于课程没有什么建议，个人认为课程较为合理，带领我初步了解了机器学习这个领域的知识，但是如果上课也有一些演示的话感觉效果可能更好些。