中山大学数据科学与计算机学院

移动信息工程专业-人工智能

本科生项目报告

（2017-2018 学年秋季学期）

课程名称：**Artificial Intelligence**

教学班级	**15M1/1518**	专业（方向）	**移动互联网**
组号	**66**	组名	**Geass**
学号	**15352408**	姓名	**张镓伟**

一、 Project 最终结果展示

1. 截止到写报告为止的结果

二元分类 (f1)	多元分类 (平均准确率）	回归 (RMSE)	二元分类排名	多元分类排名	回归排名	总排名
0.91078697422	0.649085005504	40.7048743083	13	2	2	2

2. 组内分工

1.张镓伟：主要负责三个任务的数据集处理以及二元分类的算法，简单尝试了一下多元分类以及回归的算法。

2.蔡荣裕：主要负责多元分类和回归的算法，简单尝试了二元分类的算法。

3. 个人工作

时间	工作
15 周	处理了二元分类的数据集，将数据第二列 typeA~typeE 的字母换成了数字 1-5。另外还有做独热编码的处理
	处理了多元分类的数据集，将原数据集分别转换成了 onehot、TF 以及 TFIDF 矩阵，然后根据特征列与标签列的相关系数分析降维。
	对于回归数据集，去掉了索引列、日期列以及将小时(hr)列改成独热编码
16 周	用 KNN 和 PLA 算法测试了多元分类
	用 BPNN 跑了回归

17 周周一~周五	学习了 Adaboost 算法并且尝试实现了基于单层决策树的 adaboost 算法，然后通过循环一直在调参
17 周周六~18 周周一	实现了 KNN 算法跑二元分类并且调参
18 周周一	做展示 PPT
18 周周二~周日	实现了基于 CART 多层决策树的 adaboost 算法去跑二元分类并且调参一周；实现了加权 KNN 算法
19 周	写报告

二、 工作流程

1. 算法简介

1. K 近邻算法（KNN）。用于分类中时，如果一个测试样本在特征空间中的 K 个距离最近的训练样本中大多数属于某一类别，则该样本也属于这个类别。衡量距离的方法有曼哈顿距离、欧式距离以及余弦距离（余弦值越大越近）。

2. PLA（感知机算法）。用来解决二维及以上的二元分类（+1，-1）的线性可划分问题。在这种问题中 ，我们要在 n 维空间中找到一个超平面将这个空间划成两部分，一部分预测为+1，另一部分为-1，并令错误率最小：

设样本 x={x1,x2,…xd},权重向量 w={w1,w3,…wd}, 样本实际标签 y={y1,y2,…yd},阈值 threshold, 令 w0 = -threshold; x0 = 1 预测方式如下：

$$predict = sign(\sum_{i=0}^{d} wi * xi)$$

我们需要在每一次迭代不断更新权重 W，使得预测准确率提高，更新 W 的方式是每遇到一个预测错误的样本就更新：$w_{t+1} \leftarrow w_t + y_n x_n$

3. 误差反向传播神经网络（BPNN）BPNN 网络模型拓扑结构包括三层， 分别是输入层、隐藏层和输出层，其中隐藏层可以是一层也可以有很多层，另

外两个只有一层。BPNN 的每一次迭代包括前向传播输入以及反向传播误差。在反向传播误差的过程中不断通过梯度下降法更新权重，直到最后损失函数收敛。

4. 决策树。决策树就是带有判决规则（if-then）的一种树，代表对象属性和对象值之间的一种映射关系，由结点和有向边组成。 假设分类样本有 n 种属性，决策树的每一个非叶子结点表示一种特征或者属性， 结点 i 到子结点 j 的边值 val 表示当属性 i 取值 val 时， 下一个要根据属性 j 去分类。决策树的叶子结点表示一个分类类别。 分类预测的过程可以看成从树的根一直往下走的过程。特征选择算法一般有 3 种： ID3、 C4.5 和 CART。

5. 独热编码。在这里的独热编码不是文档分类中的判断某单词在某个样本中是否出现过。而是将一些特征列是离散值（假设该列有 n 个离散值）的，改成 n 个 0/1 列，0 表示不是这个特征值，1 表示是这个特征值。在任意时刻，在同一行中，这 n 列都只有一列为 1。那么这样做的好处是：

A. 分类器一般默认数据是连续的，但是离散数据（可以理解为属性分类值）是随机的不连续的。转换之后可以解决分类器不好处理属性数据的问题，在一定程度上也扩充了特征。

B. 假如我们要使用一些基于距离计算的分类方法（如 KNN），那么经过 one-hot 编码后的离散型特征之间的距离计算会更加合理。例如某个属性有三个取值 0,1,2，现有三个样本，该特征取值依次为 x1=0，x2=1，x3=2。采用曼哈顿距离来衡量，d(x1,x2)=1，d(x1,x3) = 2，那么我们会认为 x1 和 x2 更加接近。但实际上这是不科学的，这三者两两之间应该是距离一样近，因

为这些取值只是各代表了一个类别。经过 onehot 处理后，这三个样本在这一个特征上的互相之间的曼哈顿距离都是 1,这样就比较合理。如果是那种通过函数拟合数据变化曲线的，显然只有两个取值要比有多个取值容易拟合得多。本次实验中我使用了 onehot 编码的数据集有：

A. 二元数据集将第二列的 type_A~type_E 转换成 5 列独热编码、将算上标签列在内的倒数第 3 和倒数第 4 列各转成两组两列的独热编码，因为倒数第 4 列的取值只有 0 和 1.，倒数第 3 列的取值只有 0 和 3。

B. 多元分类数据集。就是以前做过的单词的 onehot 矩阵。当然后面又处理成了 TF 和 TFIDF

C. 回归的数据集，去掉了索引列，将日期改为分别按年、月、日、季度、星期几排的独热编码、将小时列和天气列也改成了独热编码。最后，将温度这个连续值按照每 2.5 度一个区间来独热编码。

6. 相关系数分析降维。相关系数常用的是 Pearson 相关系数，是用来反映俩变量之间相似程度的统计量，在机器学习中可以用来计算特征与类别间的相似度，即可判断所提取到的特征和类别是正相关、负相关还是没有相关程度。

Pearson 系数的取值范围为[-1,1]，当值为负时，为负相关，当值为正时，为正相关，绝对值越大，则正/负相关的程度越大。我计算出训练集每一特征列和标签列的相关系数，相关系数绝对值越低，代表这个特征对分类结果的影响越低，那么去掉这一特征列可能算法的分类结果会更好。皮尔森相关系数的定义式如下：

$$\rho_{X,Y}=\frac{\operatorname{cov}(X,Y)}{\sigma_X\sigma_Y}=\frac{E((X-\mu_X)(Y-\mu_Y))}{\sigma_X\sigma_Y}=\frac{E(XY)-E(X)E(Y)}{\sqrt{E(X^2)-E^2(X)}\sqrt{E(Y^2)-E^2(Y)}}$$

我这里主要是帮队友对多元分类数据集的 onehot 矩阵进行了降维处理，多元分类我们转换成了三个二元分类，即分别对 LOW、MID、HIG 做二元分类，看

取哪个的概率高 。那么我们就分别计算每一个特征列跟每一个标签列的相关系数，各取了相关系数最大的（正相关）150 列和相关系数最小的（负相关）150 列，以此达到了降维的目的。

7.我尝试的各算法在各数据集上的最优表现

二元分类：

算法	参数	训练集类型	测试集 F1
基于单层决策树的 adaboost	33 个分类器	随机划分的原训练集的 80%的训练集	0.751544324108
基于 CART 多层决策树的 adaboost	13 个分类器、每棵树深度为 12	全训练集无降维无归一化	0.784198607733
KNN	K=1	全训练集无降维无归一化	0.91078697422

多元分类：

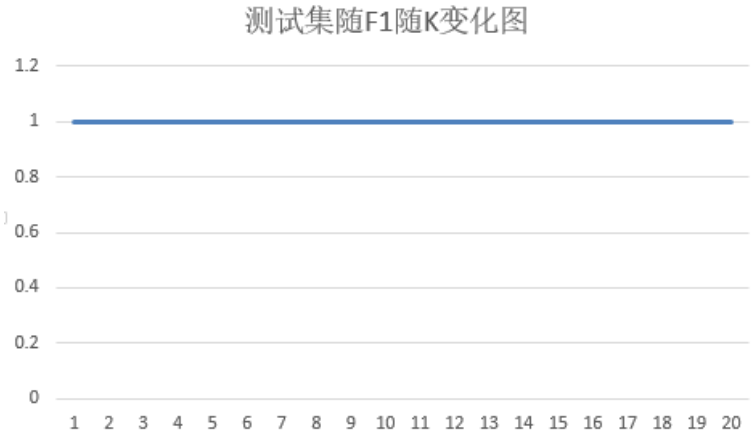
算法	参数	训练集类型	测试集平均准确率
KNN	K=150	Onehot 矩阵无降维无归一化	0.480476605406
PLA	学习率 0.001，迭代 3000 次，权重向量 W 随机初始化(1 以内)	Onehot 矩阵降维后，随机划分了 75%原训练集为训练集，25%为验证集	验证集准确率只有 40%+， 测试集结果的 12000 个样本中预测为 MID 的有 7000 多个，非常不合理，所以没有交上去跑 rank

回归：

算法	参数	训练集类型	测试集 RMSE
BPNN	2 个隐藏层，各 100 个结点，迭代 3000 次，权重 W 为随机初始化，激活函数为 tanh，alpha=0.001	做了降维和独热编码的训练集（如前所述），75%作为训练集，25%作为验证集	96.7782846931

2. 调参过程

对二元分类的 KNN 算法，使用曼哈顿距离，我写了循环将 K 从 1 到 50 试了一遍，将无归一化无降维的训练集随机划分了 20%作为验证集。结果如下：



根据上表，我们可以看出这个数据集还是比较有意思的，居然一直都是准确预测，于是我从小到大选择 K 作为参数最后反馈是 K=1 时 f1 值最高为 0.88。后续作了如下尝试：

1.不划分验证集，用全训练集去预测测试集，结果得到了 0.9 的 f1 值。

2.尝试了 K=1 时的欧式距离，测试集 f1 为 0.897016197783；尝试 K=1 的余弦距离，测试集 f1 为 0.81422057085

3.后续又尝试了 minmax 和 zscore 归一化，这时 K 取了 50，不取 1 是因为归一化后，原本影响小的列的重要程度会上升，取 1 的效果会很差，取 50 是根据经验公式取了样本数开方/2，但最后 K 取了 50 f1 值也只有 77~78，反而 K 取 1 还有 80 出头的 F1 值。

4.尝试用降维或者独热编码后的数据集去重复上面 3 步，都没有取得更好的结果，除了独热编码无归一化的数据集取 K=1 的时候取得了与前面一样的最优结果，其余的 f1 值都在 76~83 之间。

5.尝试了采用倒数加权和高斯函数加权的 KNN。最后采用高斯函数加权，并且 K=3 时得到了更优的 0.910587037665

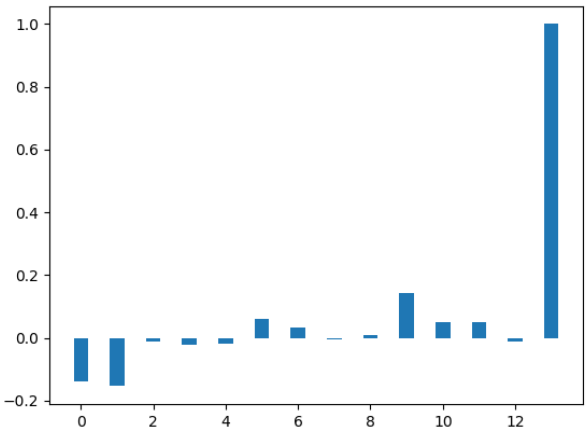
6.之前尝试的降维效果更差，我分析原因可能是因为我降得太多了，于是我少降了一些，将最后 3 维去掉再对 type 列 onehot 编码后，采用高斯加权的 KNN 在 K=3 的时候得到了更优的 0.91078697422

多元分类和**回归**我只是过了一遍算法让小组有个初始成绩，调参优化由队友负责。

3. 数据集分析

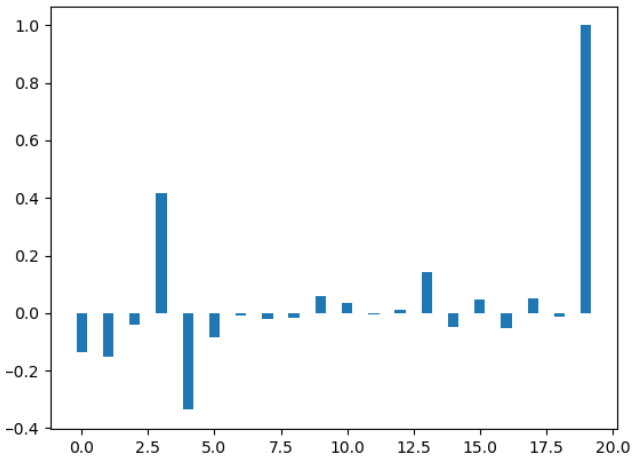
二元分类：

这个数据集没有给出每一列特征的意义，从数字上看，只知道第二列是一个离散的分类值 type，倒数第 3 列和倒数第 4 列也是分类值，因为这两类值的取值分别只有（0 和 3） 和 （0 和 1）。对原训练集的特征列与标签列做相关性分析得(type 列已改用数字 1-5 替换掉 typeA-typeE)：



由图可知，第 2、3、4、7、8、12 列的相关系绝对值非常地，说明对分类的影响比较小，剩余各列的影响比较大。

对经过 onehot 编码（原数据集 type 列，倒数第 3 第 4 列该用 onehot 编码）的数据集做相关性分析得：



由图（2、3、4、5、6 列为 type 的独热编码，倒数第 5、6 列是原倒数第 4 列，倒数第 3,4 列是原倒数第 3 列）可以看出第 2、6、7、8、11、12、18 列相关性非常低。可是按照这种分析降维后却没有跑到一个好的结果，测试集的 f1 在 0.76~0.78 徘徊。不过最后我少降了几维(见调参部分)却取得了更好的结果。

多元分类：

训练集是一个包含 622522 行文本的的数据集，每一行文本有多个句子，大概有 100 多个单词一行，整个训练集的不重复单词数约有 90048 个。这个数据集就是要求我们根据每个文本的单词做文本分类。我仔细看了一下，文本中还有大量的非法单词，比如 didn't 变成了 did 和 n't 两个词，还有的单词是中间某个字母重复了很多遍，这些我现在没办法通过代码好好区分出来，我只能对 90048 个词与每个标签列计算相关系数，去掉相关系数低的来降维。

回归：

这个数据集特征列有索引列、日期、小时、天气（三种）、温度、体感温度、

湿度、以及风速。我们需要根据这些特征值预测自行车的使用数量。很显然索引列是没有用的可以去掉。另外小时和天气两列是离散值、它们可以转化成独热编码以提高模型的分类准确性。另外最重要一环，我们将日期这一列拆成了年、月、日、季度、星期几这 5 列，很明显这 5 列都是离散特征列，于是我将它们都改成了 onehot 编码。之所以想到这么做一个是因为队友调参已经到达了瓶颈，需要找其他突破口，另一个是参考了 BPNN 实验中对数据集的说明 readme 文件，里面也是有这几列的，我只是通过对日期列的计算把它加回来再改成了 onehot，这样令特征更多，可能会对回归拟合起到帮助，事实证明也的确起到了作用。最后为了能提升拟合能力，我将温度列的温度按照每 2.5 度一个区间离散化后独热编码。另外，这次的数据联系特征可能有缺失值，对于缺失值的处理，我是取了前一个样本和后一个样本的平均值填充上去。还有，在每一个月份的样本结束之后会有一行统计该月的总数的无效行，要去掉这一行。

4. 集成学习方法(AdaBoost)

首先需要说明一下 adaboost，我参考了一篇博客，它的说明更加完整详细。

Adaboost 简介：adaboost 是一种迭代算法，核心思想是针对同一个训练集训练不同的弱分类器，然后将这些弱分类器加权组合成一个强分类器以达到一个很好的预测效果。我们下面的描述全部基于二元分类，且标签为 -1 和 1

在 adaboost 算法流程中，对每个训练样本给了一个权重 W，初始化为 1/N，每次迭代，样本数据不变，当前弱分类器的错误率会影响样本的权重和分类器的权重，错误率$\mathrm{err} = \frac{\text{分类错误的样本数}}{\text{总样本数}}$　　加权错误率：$\mathrm{Err} = \frac{\sum \text{分类错误的样本的权重} w}{\text{样本数目}}$

分类器的权重：$\alpha = \frac{1}{2}\ln(\frac{1-Err}{Err})$

由上式可以看出，当 Err > 0.5 时，α<0；当 0<Err <0.5 时，α>0；

对每个弱分类器，我们在它分类完成后根据错误率计算它的权重α，之后我们再调整下一个分类器的样本权重。假设当前是第 t 个分类器，其被分类正确的样本 i 在第 t+1 个分类器中权重由下式获得:

$$W_i^{t+1} = \frac{W_i^t e^{-\alpha}}{Sum(W)}$$

在第 t 个分类器中被分类错误的样本在第 t+1 个分类器中的权重为：

$$W_i^{t+1} = \frac{W_i^t e^{\alpha}}{Sum(W)}$$

综合得：

$$W_i^{t+1} = \frac{W_i^t e^{-\text{predictResult[i]}*\text{label[i]}*\alpha}}{Sum(W)}$$

其中 predictResult[i]和 label[i]分别表示样本 i 的预测标签和实际标签。由公式我们可以看出，当样本被分错且α>0（错误率低且分类结果可信）时，增大其权重，反之减小权重。当样本被分类正确且α>0（错误率低且分类结果可信），减小权重，反之增大权重。

单层决策树：不采用 ID3、CART、C4.5 这样的特征选择算法，它只基于单个特征做决策，方法是这个特征的取值中找一个中间值划分成两段，一段预测为+1，另一端预测为-1。然后看哪种划分方式的错误率最小。单层决策树又称树桩（Stump），尤其特点可以知道每一棵树的准确率在 50%左右。

Adaboost 和单层决策树合并后的训练算法伪代码：

Input: trainSet, trainLabels, MaxNumOfClassifiers
Output: 存储分类器的列表 classifierList

Initial: 样本权重向量 W=[1/N,1/N,…1/N]

classifierList = []

最终分类结果 result = [0,0,0…,0]

for step = 1,2,3…MaxNumOfClassifiers

找到满足 min($\sum_i Wi * isError[i]$)加权错误率最小的单层决策树 nowTree 作为一个弱分类器

NowResult, NowError = predict(trainSet, trainLabels, nowTree)

S = (1.0 - NowError) / NowError

alpha = 0.5*log(s)

for i=1,2…，len(trainSet)

Wi = Wi * exp(-alpha * NowResult [i]*trainLabels[i])

Wi = Wi/sum(W)

result[i] += alpha * NowResult[i]

end for

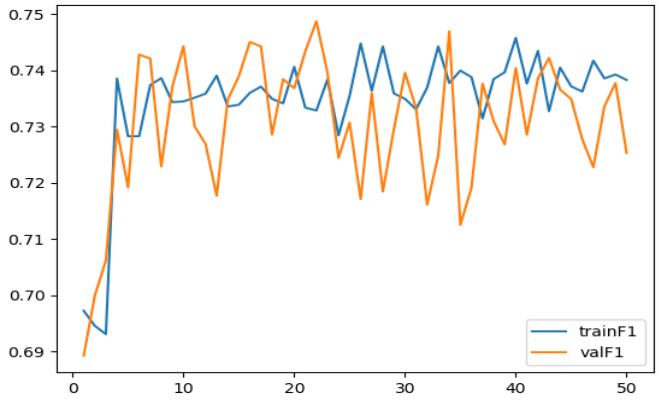
classifierList.append(nowTree)

currentError = $\frac{\sum_i sign(result[i]!=trainLables[i])}{N}$

if currentError ==0 then break

end for

效果：对原数据集随机划分 20%作为验证集，使用分类器从 1 个到 50 个的 F1 值曲线如下：



由图可以看到，分类器个数在 5 之前，F1 一直在上升，之后训练集和验证集的 f1 就一直在 0.71~0.75 之间震荡。提交跑 rank 之后得到的结果也是测试集 F1 徘徊于 70~75

基于多层 cart 决策树的 adaboost 算法：单层决策树 adaboost 算法的结果表现虽然很稳定，但是不是很好，这应该是由于单层决策树作为弱分类器准确率只有 50%还是太低的原因，由此受到启发，如果我将多层决策树作为弱分类器，效果应该会更好。我选择采用 CART 决策树，之所以选择 CART 是因为它可以解决特征列是连续值的情况，单层决策树是通过将样本权重 W 加权错误率来影响决策，但是多层 CART 显然不能这么做，因为它是通过 GINI 系数来选择特征列而不是通过错误率选择。回忆到实验课 PPT 上说的重复采样，我得出的解决方案是根据权重比例来扩增数据集，比如样本权重向量中的最小值是 minW，那么一个样本 i，由原来的一个样本，复制成 int(W[i]/minW) 个，也就是说通过增加/减少 分类错误/正确的样本来影响决策结果。其他地方就跟单层决策树是一样的。这样我们的思路就清晰了。伪代码如下：

Input: trainSet, trainLabels, MaxNumOfClassifiers, 决策树深度 depth

Output: 存储分类器的列表 classifierList

Initial: 样本权重向量 W=[1/N,1/N,⋯.1/N]

classifierList, newtrainSet, newtrainLabels = []

最终分类结果 result = [0,0,0…,0]

for step = 1,2,3…MaxNumOfClassifiers

minW = min(W)

for i=1,2…，len(trainSet)

for j = 1,2…,int(W[i]/minW)

newtrainSet.append(trainSet[i])

newLabels.append(trainLabels[i])

NowTree= getCARTtree(newtrainSet, newtrainLabels, depth)

NowResult, NowError = predict(trainSet, trainLabels, nowTree)

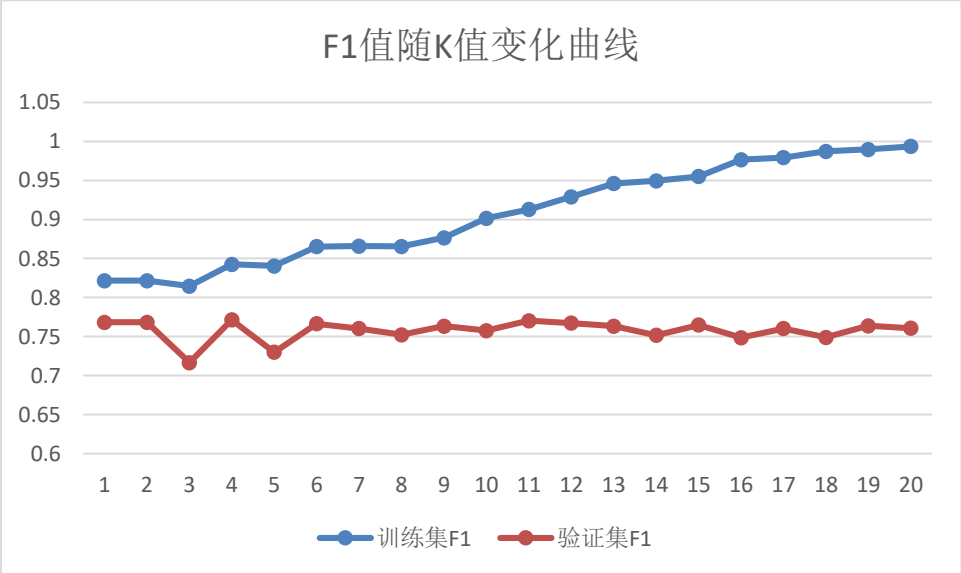
根据公式计算 S、alpha、更新权重 W，加权预测结果 result

currentError = $\frac{\sum_i sign(result[i]!=trainLables[i])}{N}$

if currentError ==0 then break

end for

效果：对原数据集随机划分 20%作为验证集，决策树深度为 10，分类器个数从 1 个到 20 个的 F1 值曲线如下：



从中选择表现较好的提交跑 rank 得

分类器个数	决策树层数	测试集 F1
4	**10**	**0.775985365042**
11	**10**	**0.769716088328**

这个结果仍不够好，我加大或减小了深度再尝试得到的部分结果如下：

分类器个数	决策树层数	测试集 F1
12	**13**	**0.778873474845**
12	**12**	**0.778990104035**
13	**12**	**0.784198607733**
15	**8**	**0.7757125154894672**
14	**8**	**0.778873474845**
1	**13**	**0.764513295536**
4	**13**	**0.77422010042**
15	**9**	**0.777458309373**
17	**9**	**0.77714755446**

从表中数据可以看到，F1 始终在 0.76~0.79 之间徘徊，非常稳定，但是结果却不够好。时间关系，我没办法再尝试多几种情况，因为层数多的时候迭代时间非常长。Adaboost 的决策树算法的优点是预测结果稳定、从上表我们也可以看出来，虽然 f1 值不够高，但是很稳定，缺点是随着层数增加，所需的时间会大幅增加，我在尝试到 10 层以上且分类器个数到 10 以上的时候，1 天只能跑出三四个结果。不够时间继续调参尝试。感觉是二元的训练集数据集可以用这个算法拟合到 1 的准确率，但是却无法同样在测试集上有优秀的表现。据说这个数据集已经是降过维的，KNN 在处理这种低维的数据集上的确会有比较优秀的表现。

三、 引用

[1]. Adaboost 原理与推导详细学习

http://blog.csdn.net/v_july_v/article/details/40718799

[2]. 基于单层决策树的 adaboost 算法参考

http://blog.csdn.net/jiaqiangbandongg/article/details/52903655

四、 课程总结

饶阳辉老师的 AI 课教的是机器学习，这是我以前从来没有接触过的领域，兴趣比较浓厚。通过一个学期的学习，我收获良多。

我们所学习的机器学习的算法，主要分为有监督学习和无监督学习。有监督

学习就是对给定的训练样本的训练得到一个最优模型，再利用这个模型去预测未知样本的标签（分类）或者未知样本所有类别的概率（回归）。而无监督学习则是没有训练样本、我们需要直接对测试数据进行建模、如 K-means 聚类。实验课主要学习的是有监督学习、包括了 KNN、朴素贝叶斯、逻辑回归、感知机、BP 神经网络、决策树。在学习这些算法的过程中，我感觉最困难的不是代码的实现，而是原理的推导，每次我都需要花费一番功夫才能搞懂一个算法的原理推导，而这也是 AI 课比其他课要更加辛苦的地方。但一旦搞懂了原理，就会感觉一切都豁然开朗，代码写起来也更加流畅。

在课程方面，理论课我觉得还是比较好的，课件做的还比较详细，我经常课后需要重新看着课件学习算法，而这往往都比较顺利。至于实验课，课件有时候比较详细，有时候比较乱。例如 BPNN 的课件，中间有两三种风格的表达式截图、会让人看晕，这次实验我是对着理论课的课件去写的代码。希望之后课件能够统一一下风格，这样比较清晰有条理。另外期末 project 我建议不要刷 rank，因为大三上大家都巨多课，像我这学期 8 门课有 7 门实验大作业，剩下一门计网也是要考试，而且全都堆在最后这两三个星期布置，要刷 rank 就要每天不停地跑 AI 的代码，而这还很吃内存，比如多元分类的代码，经常卡着电脑做其他作业都不好做或者没多少时间做，我的电脑还因为很多天跑代码不关机死机了几次，所以还是建议取消最后这个比赛制吧。