

# 题目 集成学习 AdaBoost 和 Random Forest

姓名 吴紫航 学号 171860659 邮箱 [401986905@qq.com](mailto:401986905@qq.com) 联系方式 call: 18956668797

(南京大学 计算机科学与技术系, 南京 210093)

## 1 实验目的

- (1) 实践集成学习的两大经典想法: "Boosting" and "Bagging"
- (2) 理解并掌握 AdaBoost 和 Random Forest 两个算法的原理和实现
- (3) 加强对传统机器学习框架流程的掌握, 如模型构建、交叉验证、超参数训练等过程

## 2 算法理论

本节主要介绍 *AdaBoost* 和 *Random Forest* 算法层面的伪代码

### 2.1 AdaBoost

---

#### 算法 1 AdaBoost 算法

---

输入: 训练集  $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$ ; 基训练算法  $\zeta$ ; 训练轮数  $T$ .

输出:  $H(\mathbf{x}) = \text{sign}(\sum_{t=1}^T \alpha_t h_t(\mathbf{x}))$

```
1: function ADAMBOOST( $D, \zeta, T$ )
2:    $D_1(\mathbf{x}) \leftarrow 1/m$ .
3:   for  $t = 1, 2, \dots, T$  do
4:      $h_t \leftarrow \zeta(D, D_t)$ ;
5:      $\epsilon_t \leftarrow P_{\mathbf{x} \sim D_t}(h_t(\mathbf{x}) \neq f(\mathbf{x}))$ ;
6:     if  $\epsilon > 0.5$  then
7:       break
8:     end if
9:      $\alpha_t \leftarrow \frac{1}{2} \ln(\frac{1-\epsilon_t}{\epsilon_t})$ ;
10:    if  $h_t(\mathbf{x}) = f(\mathbf{x})$  then
11:       $D_{t+1}(\mathbf{x}) \leftarrow \frac{D_t(\mathbf{x})}{Z_t} \times \exp(-\alpha_t)$ 
12:    else
13:       $D_{t+1}(\mathbf{x}) \leftarrow \frac{D_t(\mathbf{x})}{Z_t} \times \exp(\alpha_t)$ 
14:    end if
15:  end for
16: end function
```

---

## 2.2 Random Forest

---

### 算法 2 Random Decision Tree 算法

---

输入: Data set  $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$ ; Feature subset size  $K$ ;

输出: A random decision tree

```

1: function RANDOMDECISIONTREE( $D, K$ )
2:    $N \leftarrow$  create a tree node based on  $D$ ;
3:   if all instances in the same class then
4:     return  $N$ ;
5:   end if
6:    $F \leftarrow$  the set of features that can be split further;
7:   if  $F$  is empty then
8:     return  $N$ ;
9:   end if
10:   $\tilde{F} \leftarrow$  select  $K$  features from  $F$  randomly;
11:   $N.f \leftarrow$  the feature which has the best split point in  $\tilde{F}$ ;
12:   $N.p \leftarrow$  the best split point on  $N.f$ ;
13:   $D_l \leftarrow$  subset of  $D$  with values on  $N.f$  smaller than  $N.p$ ;
14:   $D_r \leftarrow$  subset of  $D$  with values on  $N.f$  no smaller than  $N.p$ ;
15:   $N_l \leftarrow \text{RandomDecisionTree}(D_l, K)$ ;
16:   $N_r \leftarrow \text{RandomDecisionTree}(D_r, K)$ ;
17:  return  $N$ 
18: end function

```

---

### 算法 3 Random Forest 算法

---

输入: 训练集  $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$ ; 特征子集大小  $K$ ; 训练轮数  $T$ .

输出:  $H(\mathbf{x}) = \arg \max_{y \in Y} \sum_{t=1}^T \mathbb{I}(h_t(\mathbf{x}) = y)$

```

1: function RANDOMFOREST( $D, K, T$ )
2:   for  $t = 1, 2, \dots, T$  do
3:     自助采样得到样本子集  $D_{bs} \leftarrow D$ 
4:      $h_t \leftarrow \text{RandomDecisionTree}(D_{bs}, K)$ ;
5:   end for
6: end function

```

---

## 3 实现过程

本节主要介绍代码实现过程主要流程和一些细节

### 3.1 结构配置

文件	内容
AdaBoost.py	AdaBoost 代码
RandomForestMain.py	RandomForest 代码
adult.data	adult 训练集
adult.names	adult 数据集介绍
adult.test	adult 测试集

ML2020_PS5.tex	习题解答 latex 版本
ML2020_PS5.pdf	习题解答 pdf 版本
report.pdf	你正在读的报告

### 3.2 模块介绍

模块/方法	功能
InputData(filePath)	根据路径获取数据, 删除缺失数据的行, 并把离散字符串数据映射到整数的方法
AdaBoostModel	AdaBoost 模型定义, 内置训练、测试以及基于 AUC 的最佳轮次获取接口
RandomForestModel	RandomForest 模型定义, 内置训练、测试以及基于 AUC 的最佳轮次获取接口

注: 运行需要安装的 py 包有: *pandas, numpy, sklearn*

### 3.3 数据读入

- (1) 利用自定义的 inputData 模块读入 adult.data 数据, 数据格式为 DataFrame
- (2) 放弃使用 adult.test 数据, 把 adult.data 按 3:1 随机划分为 trainingDataSet 和 testDataSet
- (3) 把 trainingDataSet 随机分为 5 等份, 方便之后进行五折交叉验证
- (4) DataFrame 数据结构在拆分后, 需要重置索引, 调用 reset\_index 接口即可

### 3.4 训练模型

#### 3.4.1 AdaBoost

在 *AdaBoostModel.train()* 内实现模型的训练算法。主要思想参照 2.1 的算法即可。这里整理一些细节:

- (1) 单个的弱学习器采用的是单层的**决策树(桩)**, 保证其学习能力较弱
- (2) 对于样本的分布, 原本采用的方法是直接在 fit 接口中给参数 sample\_weight 传值, 但模型训练过程经常发生错误率大于 0.5 的情况, 故现在改进为处理样本分布的方法是用 sample 接口进行**等量加权采样**, 这样可以进行重启动(重采样)防止迭代太早结束。
- (3) 对于算法 2.1 的第 9 行, 若错误率为 0 会导致分母为 0, 因此参考**拉普拉斯修正**的思想, 分母修正为  $\max\{e, 1e-16\}$
- (4) 每轮学好一个新的弱学习器后, 都把最新的集成模型在验证集上进行验证, 并保存轮次-AUC 的结果

#### 3.4.2 Random Forest

在 *RandomForestModel.train()* 内实现模型的训练算法。调用了 *DecisionTreeClassifier* 接口。这里整理一些细节:

- (1) 使用**自助采样法**保证单学习器之间的行差异性(同样用 sample 接口, 不加权重, 等概率抽样)
- (2) 使用属性随机抽取保证单学习器之间的列差异性(在 DT 接口中使用参数 **max\_features='log2'**)
- (3) 虽然是并行算法, 但还是串行的训练。训练完第 k 个随机决策树, 把前 k 个随即决策树集成验证即可
- (4) 总的来说, 实现了 **bootstrap 和随机决策树的结合**; 随机决策树内部的实现由调包实现

## 交叉验证

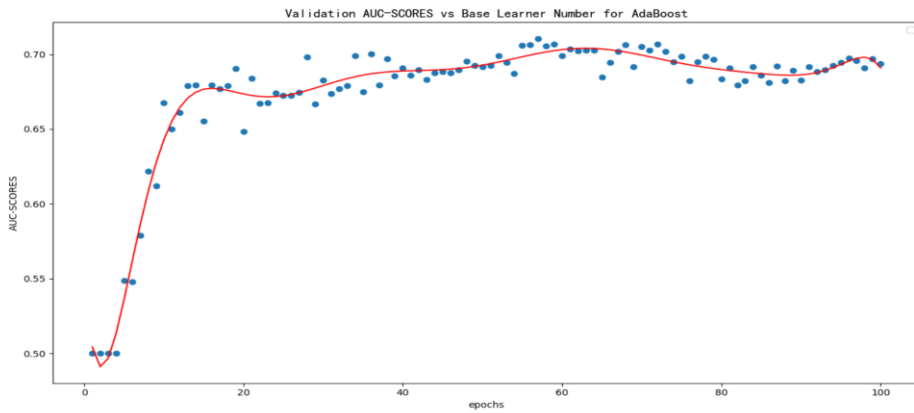


Figure 1 五折交叉验证的 AUC vs Base Learner Number for AdaBoost

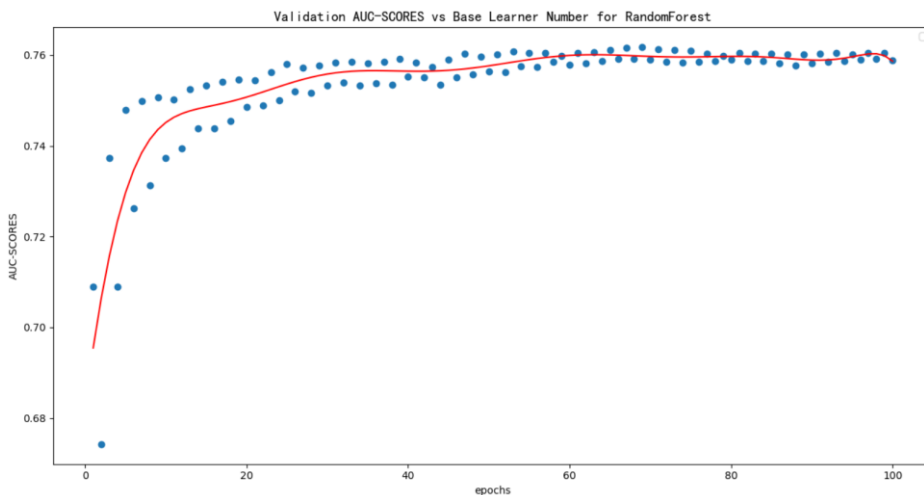


Figure 2 五折交叉验证的 AUC vs Base Learner Number for Random Forest

由五折交叉验证的 AUC-epochs 曲线, 对于两种算法, 最理想的训练轮次分别为 57 和 69

### 3.5 测试结果

以下给出用最佳轮次进行训练后的模型, 在测试集上的运行结果, 并用决策树桩(弱学习器)进行对比  
由于是随机算法, 每次的运行结果可能会有差异

算法	最佳轮次	测试集准确率	测试集 AUC-SCORES
----	------	--------	----------------

决策树桩	\	74.58 %	50.00 %
AdaBoost	57	81.74 %	75.01 %
Random Forest	69	84.64 %	76.83 %

我们对以上结果进行适当分析：

- (1)相比单个的学习器，两种集成方法虽然在准确度指标上提升不大，但是在 AUC 上都有明显的效果提升；
- (2)随机森林的运算速度较快(可并行化)，且实现容易，且效果也略好于 AdaBoost；
- (3)AdaBoost 算法能以较少轮次达到最优轮次；

### 3.6 补充思考：

Adult 数据集的标签分为 0 和 1，即年收入是否大于 50 万。这是一个非常不均匀的样本分布，我们知道年收入大于 50 万的肯定明显较少。事实上也确实如此：adult 数据集中 1 标签的个数只有 0 标签个数的 1/3。因此哪怕一个训练器将所有样本都归类为标签 0 也将获得 75%左右的准确率。但是这没有任何意义，因此本次实验选取的性能指标是 AUC，即考察：集成后的学习器，比较于单个的学习器，是否在 AUC 性能上具有提升。这意味着年收入大于 50 万的人群也能有足够机会被训练器给识别出来。

## 4 参考资料

### [1] KNN 算法对 adult 数据集预测

参考了博客中数据的读取到 DataFrame 的方法，以及预处理方法(删除缺失数据、字符离散值映射到整数)  
[https://blog.csdn.net/qq\\_43745026/article/details/107219361](https://blog.csdn.net/qq_43745026/article/details/107219361)

### [2] pandas dataFraome 重置索引

参考 reset\_index 接口的使用,用于在划分数据集后重置索引  
[https://blog.csdn.net/weixin\\_39223665/article/details/97397404](https://blog.csdn.net/weixin_39223665/article/details/97397404)

### [3] 决策树分类器 scikit-learn 的使用

参考了单决策树的接口使用方法和参数说明  
<https://zhuanlan.zhihu.com/p/40968625>

### [4] Pandas 中 loc 和 iloc 函数用法详解

参考了 iloc 的使用方法，用于分离特征列和标签列  
[https://blog.csdn.net/W\\_weiyang/article/details/81411257](https://blog.csdn.net/W_weiyang/article/details/81411257)

### [5] pandas.DataFrame.sample 随机选取若干行

参考了 sample 接口用于实现抽样(AdaBoost 中加权抽样改分布,Random Forest 中随机抽样)  
<https://blog.csdn.net/zhengxu25689/article/details/87347700>

### [6] sklearn 中决策树重要参数详解

---

参考了 `class_weight` 参数,因为原本想进行类别均衡,但发现弱训练器不需要良好的 AUC(突出集成学习可以改善 AUC),故最后放弃使用该参数

<https://www.cnblogs.com/juanjiang/p/11003369.html>

#### [7] python 中 `train_test_split()`函数划分训练集、测试集数据

参考了 `train_test_split()`方法用于划分训练集和测试集

<https://blog.csdn.net/peachlychee/article/details/102897919>

#### [8]sklearn 中的 `train_test_split` 方法

参考了 `train_test_split()`方法用于划分训练集和测试集

<https://blog.csdn.net/fxlou/article/details/79189106>

#### [9]理解随机森林：基于 Python 的实现和解释

参考了随机森林的原理和 `RandomForestClassifier` 接口的初步使用

<https://zhuanlan.zhihu.com/p/53042043>

#### [10]机器学习 sklearn 中决策树模型参数释义

参考了 `DecisionTreeClassifier` 接口的 `max_features` 参数用法

[https://blog.csdn.net/qq\\_16000815/article/details/80954039](https://blog.csdn.net/qq_16000815/article/details/80954039)