# 项目报告

## ——分布式随机森林的实现

### 简述

#### 背景描述

此项目为中山大学软件学院2015年春季学期数据挖掘课程第二次课程项目。

题目是一个典型的分类问题，要求学生使用随机森林与任意并行技术来解决。

题目提供的数据集将所有数据分成26类，且每条记录有617个特征值。

此外，题目提供6238条训练数据以及1559条测试数据。

#### 使用程序语言及框架

此项目使用python语言进行编写。虽然python在效率上不如c语言或者java快，但是由于第一次实现决策树，希望将更多的时间花费在理清算法的逻辑上而非实现一些特殊的数据结构上。Python恰好十分符合这个需求，提供大量强大的API，使得我们的思路能够很轻松的转换成代码实现。

另外，由于在后期实现并行化的时候计划使用spark框架。刚好在本学期上的大数据软件工程课程中学习并使用过python版本的spark，因此选择python来实现更加方便我再后期将单机的随机森林程序改成分布式的。

### 程序介绍

分类器：分类决策树CART

树数量：500

每次节点分裂抽取特征比例：0.15

训练每棵树时有放回的抽取数据比例：1

|  |
| --- |
| tree |
| Attributes:  + nodes : list |
| Methods:  + train(records, featurePercent) : tree  + predict(record) |

数据结构：

|  |
| --- |
| Node |
| Attributes:  + reccords : list  + leftChild : Node  + rightChild : Node  + bestFeature : int  + bestVal : float  + isLeaf : boolean  + label : string |
| Methods:  + is\_leaf() : boolean |

### 基础实现

#### 单机随机森林

1. 从本地文件中获取训练数据并以矩阵形式储存。矩阵中的每一行为一条数据，每一列为一个特征，最后一列为标签
2. 从所有数据集中有放回地随机抽取与原数据集相同大小的新数据集
3. 根据数据集训练一个决策树，并压入一个列表中
4. 根据数据集申明一个新节点作为目标树的根节点并储存于一个list中。该list用于储存所有节点
5. 设置start和end两个游标，start指向0，end指向len(records)，start到end之间的节点即为待分裂节点（广度优先搜索策略）
6. 遍历下标处于start到end之间的每一个节点，判断当前节点是否是叶子节点，如果是叶子节点，则跳过此次循环
7. 找到对当前节点而言最优的分割特征以及分割值
8. 根据参数featurePercent从所有特征值中无放回的随机提取一定数量特征值下 标
9. 遍历特征值，对每个特征值找出最好的分割值
10. 将当前节点的所有数据根据当前特征值进行排序
11. 寻找当前特征值的突变点
12. 根据突变点两边的特征值计算分割值
13. 根据分割值将当前节点的数据集分割成两个新的子集
14. 计算两个子集分别的GINI系数
15. 如果当前GINI系数小于最小的GINI系数，则将当前特征定义为最佳特征，当 前分割值定义为最佳分割值
16. 循环步骤4-8直至当前特征值的所有突变点都访问过
17. 循环步骤2-8直至遍历完所有特征都访问过
18. 根据最佳分割特征以及最佳分割值将当前节点的数据集划分成两个子数据集
19. 返回最佳分割特征，最佳分割值以及两个子数据集
20. 根据最优分割特征和分割值对当前节点进行分割。对于每条记录，其对应属性小于等于分割值的为一组，大于的为一组。生成两个新的矩阵
21. 根据新矩阵生成两个新子节点，将两个子节点压入用于储存节点的list中
22. 将两个新节点设置为当前节点的leftchild和rightChild
23. 将最好的分割特征以及最好的分割值记录在当前节点的bestFeature和bestVal属性中
24. 重设start为上一次循环的end
25. 循环c-d直至start到end之间的所有节点都是叶子节点，循环结束
26. 单棵决策树训练结束
27. 循环步骤2-3直至训练得到n棵决策树
28. 从本地文件中获取预测数据并以矩阵形式储存
29. 遍历数据集中的每一条数据，并逐一对数据进行预测。每条数据都用n棵决策树进行预测，并获得n个类别结果，认为在预测结果中出现次数最多的类别为真实类别
30. 预测结束

## 程序优化

#### 单机随机森林

在实现分类器，即分类回归树的时候，我发现完全按照算法的逻辑编写出来的程序效率十分慢，训练一棵树大概需要3个小时，这是十分不合理的，于是我开始寻找程序中最耗时的地方在哪里。不难发现，在CART算法中最耗时的步骤就是处理连续数据时需要遍历整个数据集的所有特征值，才能确定在当前数据集和特征集合中，哪一个特征的哪一个值分割出来的两个子集合才会有最小的GINI系数。

再进一步分析下去，我对每一个可能耗时的步骤都进行了计时，发现程序在遍历一个特征的所有可能分割值的这个过程消耗了十分多的时间，平均每个特征需要15秒去找出最佳的分割点。于是我开始分析其中的原因，一开始我认为是重复的集合分割过程比较耗时。于是我将“确认分割点-划分集合”这个过程改变成“确认分割点-从右边的集合删除小于等于分割点的数据-将被删除数据加入左边集合”，这样就避免了重复的便利整个数据集的过程。

然而这样改进之后，程序的效率仍然不是十分理想，寻找一个特征的最佳分割点还是需要8-10秒左右。因此我再次对可能耗时的过程计时。最后我发现，当我每次根据某个新的分割点将数据集分割成两个子集以后，都要进行计算子集的GINI系数的过程。计算GINI系数包括两个步骤，步骤一是计算数据集中每个label出现的次数，步骤二是根据每个label出现的次数计算出现频率并累加。也就是说每计算一次GINI系数，程序就需要再遍历一次当前节点的整个数据集，这是十分耗时的。于是我再次修改程序逻辑，将一开始模拟切分子集的过程改变成直接累计每个label出现的次数。也就是说，当我开始遍历某个特征的所有可能分割点之前，我先申明left和right两个字典，分别保存分割后左子集每个类别出现的次数以及分割后右子集每个类别出现的次数。那么当分割点改变的时候，我们只需要将右子集中被删除的数据对应类别的累加器减1，左子集加1。那么当分割结束后需要计算GINI系数时，则免去了遍历当前节点整个数据集的过程。通过这样的改进，训练一棵树大概只需要5分钟的时间。

## 五、 总结