# 完善的决策树与随机森林算法

倪杰

2021 年 3 月 24 日

**摘要**

本次随机森林的实现基于上次的决策树算法。随机森林在 bagging 集成的基础上，在决策树中引入随机属性选择并作为基学习器。所谓 bagging，是指在训练每棵树时，并不采用原有的数据集，而是在原有训练集上进行不放回采样生成新的训练集。随机属性选择是指决策树划分节点时搜寻最佳划分准则时不再是遍历全部属性，而是每次随机选取 k 个属性，在这 k 个属性中选取最优值。最后对 n 棵树的预测结果进行投票，得到最终的预测。

# 目录

# 1 决策树算法的完善

上次实现了较为简易的决策树，参数较少。本次新增了是否剪枝，最大深度，最少样本数量等参数，可减少结点数目，缓解训练过程中的过拟合。

## 1.1 剪枝

剪枝需要选取训练集的一部分作为验证集，以此来衡量当前结点是否应该划分。本次实现了预剪枝的算法。验证集选择随机抽取的训练集的 20%。因为决策树当前结点只影响划分到该结点数据的预测结果，故验证集划分可与训练同时进行，验证不需要在所有验证集上进行。

## 1.2 最大深度与最少样本数量

最大深度指的是当决策树深度大于该值时，强制返回当前结点样本中出现最多的标签，停止继续划分结点。最少样本数量指当前结点样本数小于该值时强制停止划分。最少样本数量易于添加，而最大深度在建树的递归函数中加一个深度的参数即可。两者与剪枝同时进行限制可有效防止过拟合。

## 1.3 一些速度上的改进

在属性为连续属性时，每次选择划分属性的划分值时需要遍历几乎所有出现的值，这样效率有点太低。故此次决策树并不遍历所有划分之，而是在划分值备选数量大于样本量 10%时将所有可能值从最大值到最小值划分等块，遍历这些等块即可。在 wine-quality 数据集上效率能提高 40%

# 2 随机森林基本算法实现思路

首先对原有的决策树类进行改进，生成单棵随机树类，即会随机选择属性的决策树类。然后创建随机森林类，给出的参数传入到里面每棵随机树中。随机森林的 fit(X,Y) 将 X，Y 保存，每次 bagging 出不同的训练集传给不同的树进行训练, 储存所有训练出的树随机森林的 predict(test) 对每棵树进行预测，再将所有预测结果投票，返回投票最多的类。

# 3 sklearn 中简单的代码实现

```
from sklearn.datasets import load_iris
from sklearn.model_selection  import cross_val_score
import sklearn.tree as st
from sklearn.ensemble import RandomForestClassifier
import random
import sklearn.utils as us
iris = load_iris()
x = iris.data
y = iris.target
x,y=us.shuffle(x,y,random_state=7)
dt=RandomForestClassifier(criterion='entropy',max_depth=10)
scores = cross_val_score(dt, x, y, cv=10, scoring='accuracy')
print(scores.mean())
```

# 4 结果

本次采用 iris 和 wine-quality 两个数据集作分类任务, 衡量标准为 4 折交叉验证下的预测正确率.

## 4.1 iris 结果

在单纯的决策树模型上, 自编的与 sklearn 的决策树正确率分别为 0.933 和 0.960，两者差别不大。在随机森林模型上, 自编的正确率为 0.980,sklearn 的正确率为 0.967，自编的随机森林表现略优。

## 4.2 wine-quality 结果

在单纯的决策树模型上, 自编的与 sklearn 的决策树正确率分别为 0.548 和 0.424，自编的明显更优。在随机森林模型上, 自编的正确率为 0.68,sklearn 的正确率为 0.52，明显自编的随机森林表现更优。

# 5 不足与日后的改进

wine-quality 是 7 分类任务，直接在上面训练预测不管是自编的还是 sklearn 的决策树效果都不是很好，可以做一个一对一或者一对多的多分类机制。但直接判断效果确实已经优于 sklearn 的随机森林。