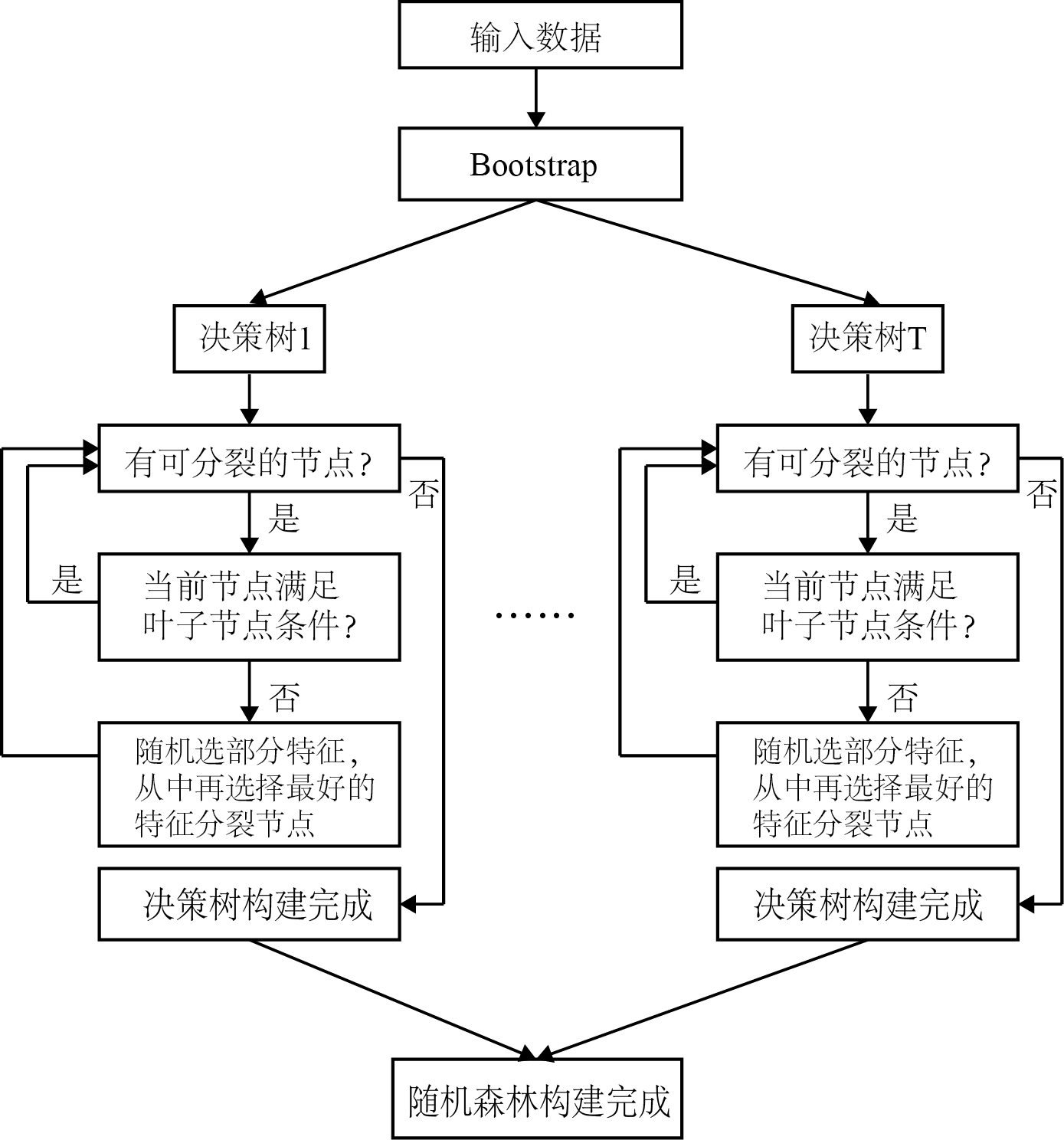
随机森林算法文档

2016-03-28

朱方舟

1. **输出随机森林（和Scikit RF的实现对齐）的算法（分类和回归两种场景）流程图和关键公式，包括Splitting point公式、特征排序公式。定义输入数据格式和输出数据格式。**



随机森林的分类和回归都可对应上面的算法流程图，唯一不同的是如果是回归，下面的不是决策树，是回归树。注意，分类算法支撑二分类和多分类问题。回归算法支持多目标回归。

下面介绍算法细节：

1. 分类
2. 节点何时结束分裂？

Scikit中分裂结束条件一共有3个，分别对应3个参数（max\_depth, min\_samples\_split, min\_samples\_leaf）。

max\_depth：如果当前节点的深度大于等于max\_depth，那么结束分裂，当前节点是叶子节点（根节点深度是0）；

min\_samples\_split：如果当前节点内的样本数小于min\_samples\_split，那么结束分裂，当前节点是叶子节点；

min\_samples\_leaf：如果当前节点内的样本数小于2倍的min\_samples\_leaf，那么结束分裂，当前节点是叶子节点（如果小于了2倍，那么分裂好总有一个子节点的样本数小于min\_samples\_leaf，这样就不满足要求）

1. 节点如何分裂？
2. 从所有特征中随机选取max\_feature个候选特征
3. for f in 候选特征集；
4. 对节点中的样本按照特征f的值从小到大排序
5. 对于排好序的样本，如果相邻的样本标签不同，那么把这两个样本的f特征的值的均值作为切分点（如下图中3.2和4.5中间是切分点，4.5和9.3之间是切分点，但是2.8和3.2由于他们label都是-1 ,所以不作为切分点）
6. 对所有的切分点进行计算增益值（小于切分点值的进入左子节点，否则进入右子节点），一般有两种方法，gini增益和entropy增益。

**gini:**

，其中表示第i类样本所占比例

**entropy:**

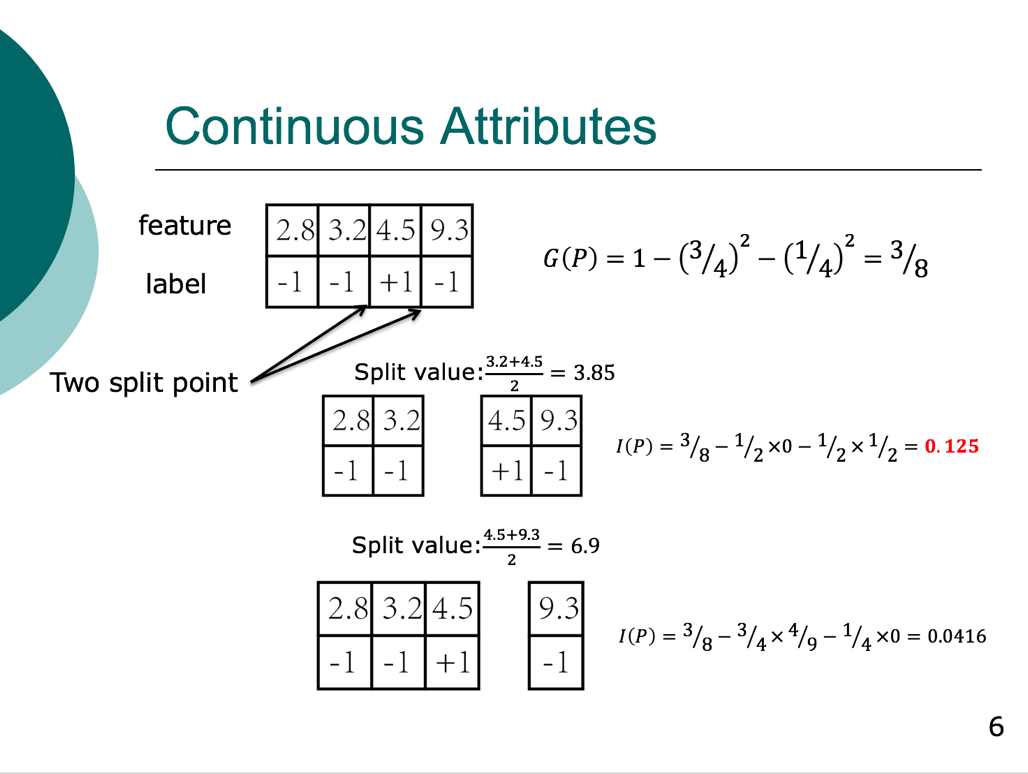
，的含义和上面一样

**增益（以gini为例，entropy一样的）：**

，表示左子 节点的样本数占当前节点样本的比例，同理，表示左子节点。二分类Gini：, .

从所有切分点中选取增益值最大的作为这个特征的最优切分点

1. 从所有特征中选取增益值最大的特征（用那个最优切分点的增益值进行比较）来做为这个节点的分裂特征，用最优切分点作为切分点分裂。

比如下图是一个简单的例子

1. 如何得出预测结果？

对于一棵决策树来说，预测结束就是把测试数据沿着决策树一直走到叶子节点，用叶子节点中训练数据正样本所占比例来作为输出。比如叶子节点中的训练数据有10个，其中8个为正例，2个为负例，那么最后落到这个叶子节点的测试数据预测为正例的概率是0.8。

对于随机森林来说，就是把所有决策树的预测结果取平均。

1. 特征重要性如何算？

其中，表示以特征f为分裂特征的节点集合，就是上面说过的节点S以特征f分裂最好的增益值。

对每个特征都算出来然后从小到大排序得到最后的排名（值越大说明越重要）。

1. 回归

对于回归算法，只有几个地方不同，下面没列出来的默认和分类一样。

1. 节点分裂

对于回归问题，定义节点的variance为

其中，为S节点的样本数量。

增益的定义和上面分类也类似。

然后同样也是要找最优切分点，再找最优特征。二分类回归. 等价二分类Gini计算。

1. 预测结果

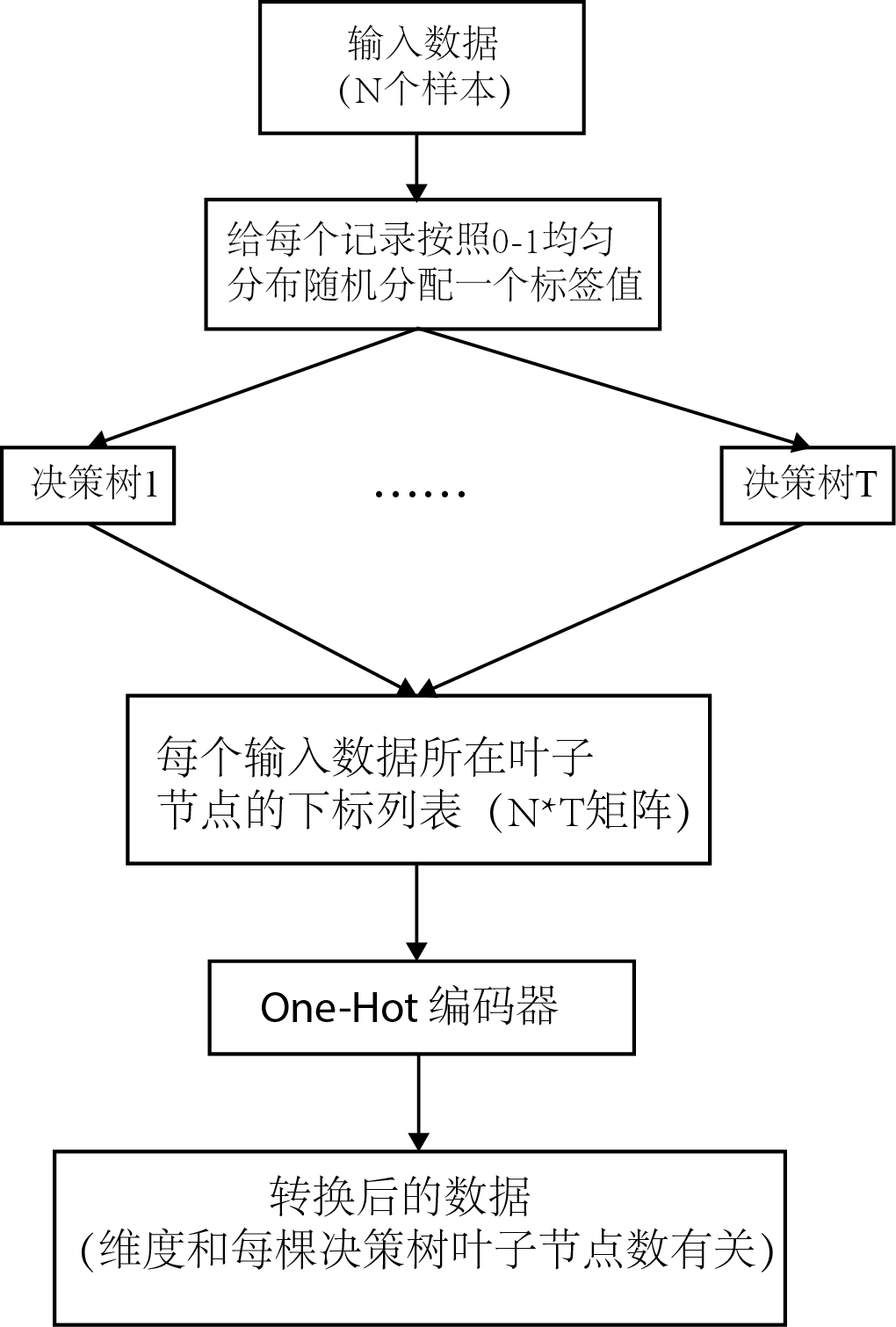
回归的输出就是叶子节点中训练样本的标签值的平均。

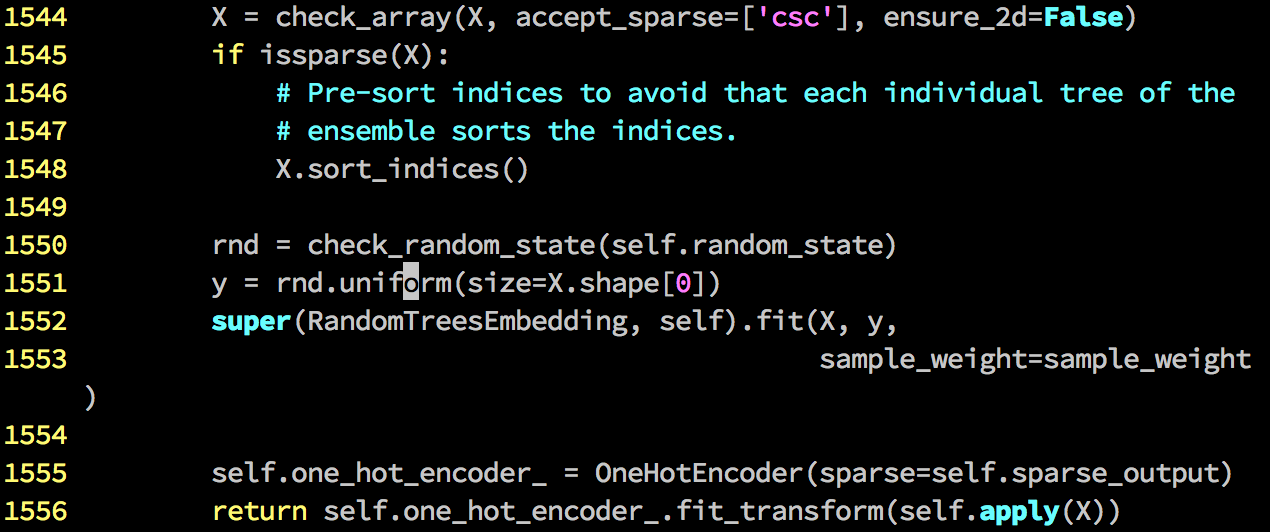
比如叶子节点中有3个样本，他们的标签分别是2，4，6。那么落在这个叶子节点的测试样本的预测结果为4（公式为）。

1. 特征重要性

算法和分类一样，只是把换成这边的。

1. 二类回归为题，如果设置，则和二类分类问题Gini增益等价。
2. **输出RandomTreeEmbedding算法（Unsupervised，和Scikit实现方式对齐）流程图和公式，解释输出结果如何形成聚类样式，并得到类似K-means或者Hierarchical Clustering算法的类似输出结果。简单说就是如何用该算法对一堆数据进行聚类。RandomTreeEmbedding是否就是用随机森林来做聚类？**

流程图：



1. 上图为算法核心代码片段，1551行对应就是对数据中每个样本按照均匀分布随机给一个标 签值，然后1552行按照回归树的方法进行训练，最后1556行中apply的作用是得到X中每个样本在书中叶子节点的下标，然后再用one-hot encoder进行编码得到高维的稀疏矩阵。
2. 这个算法应该不能算是聚类，只能算是一个非监督的算法。把一个数据集转换成一个高维稀疏的表示。它通过随机给数据打标签，把非监督转变成“监督”，这边打引号是因为这个标签没有什么意义。然后用所在叶子节点的下标来做转换后数据的特征。
3. 下面是一个使用这个算法的例子：

import numpy as np

np.random.seed(10)

import matplotlib.pyplot as plt

from sklearn.datasets import make\_classification

from sklearn.ensemble import RandomTreesEmbedding

from sklearn.preprocessing import OneHotEncoder

from sklearn.cross\_validation import train\_test\_split

n\_estimator = 10

X, y = make\_classification(n\_samples=80)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.5)

# It is important to train the ensemble of trees on a different subset

# of the training data than the linear regression model to avoid

# overfitting, in particular if the total number of leaves is

# similar to the number of training samples

X\_train, X\_train\_lr, y\_train, y\_train\_lr = train\_test\_split(X\_train, y\_train, test\_size=0.5)

# Unsupervised transformation based on totally random trees

rt = RandomTreesEmbedding(max\_depth=3, n\_estimators=n\_estimator)

X\_train\_transformed = rt.fit\_transform(X\_train)

print X\_train\_transformed.toarray()

print '----------'

print X\_train\_transformed.shape

print '----------'

print np.sum(X\_train\_transformed.toarray(), axis=1)

结果如下：

[[ 0. 0. 0. ..., 0. 0. 0.]

[ 0. 0. 0. ..., 0. 1. 0.]

[ 0. 0. 0. ..., 0. 1. 0.]

...,

[ 0. 0. 0. ..., 0. 0. 0.]

[ 0. 0. 0. ..., 0. 0. 0.]

[ 0. 0. 1. ..., 0. 0. 0.]]

----------

(20, 62)

----------

[ 10. 10. 10. 10. 10. 10. 10. 10. 10. 10. 10. 10. 10. 10. 10.

10. 10. 10. 10. 10.]

可以看到我们设置了10棵树，所以最后得到的数据的每行和都是10（第3个结果）。然后第2个结果中的20表示我们转换了20个样本，62表示10棵树一共有62个叶子节点。