# Regression Tree 回归树

Microstr...



**本文首发于我的微信公众号里，地址：**Regression Tree 回归树



我的个人**微信公众号**：**Microstrong**
**微信公众号ID：MicrostrongAI**
**公众号介绍：**Microstrong(小强)同学主要研究机器学习、深度学习、计算机视觉、智能对话系统相关内容，分享在学习过程中的读书笔记！期待您的关注，欢迎一起学习交流进步！
**个人博客：**blog.csdn.net/program_d...

---

**目录：**

知乎 首发于 人工智能

## 1. 引言

在全民人工智能时代下，机器学习算法已经成为研究和应用的热点。目前，最流行的两类算法莫过于**神经网络算法（卷积神经网络、循环神经网络、生成式对抗网络和图神经网络）**与**树形算法（随机森林、GBDT、XGBoost和LightGBM）**。树形算法的基础就是**决策树**，由于其易理解、易构建、速度快等特点，被广泛的应用在数据挖掘、机器学习等领域。因此，决策树是经典的机器学习算法，很多复杂的机器学习算法都是由决策树演变而来。对于决策树的学习，是我们机器学习课程中非常重要的一个环节。

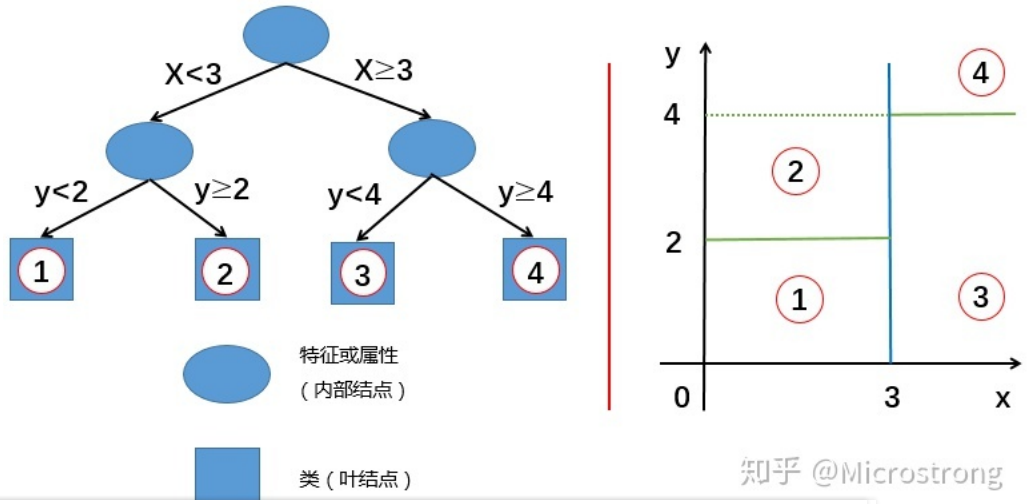
根据处理数据类型的不同，决策树又分为两类：**分类决策树**与**回归决策树**。分类决策树可用于处理离散型数据，回归决策树可用于处理连续型数据。

由于我在学习GBDT算法时，了解到GBDT中的树是回归树，但是在之前的学习中对于回归树了解比较少，这直接影响我对GBDT算法原理的理解。因此，本文首先简单介绍回归树，然后详细介绍CART回归树算法及流程，其次会给出完整的示例以加深理解，最后会讨论ID3、C4.5能不能用来做回归问题，及讨论回归树的研究进展。

## 2. 回归树

### 2.1 决策树简介

决策树是一种基本的分类与回归方法。决策树由结点(node)和有向边(diredcted edge)组成。结点有两种类型：内部结点(internal node)和叶结点(leaf node)。内部结点表示一个特征或属性，叶结点表示一个类别或者某个值。



根据划分结果，将样本分配到其子结点，这时，每一个子节点对应着该特征的一个取值。如此递归地对样本进行测试并分配，直至到达叶结点。

其实，决策树是将空间用超平面进行划分的一种方法，每次分割的时候，都将当前的空间根据特征的取值进行划分， 这样使得每一个叶子节点都是在空间中的一个不相交的区域，在进行决策的时候，会根据输入样本每一维特征的值，一步一步往下，最后使得样本落入N个区域中的一个（假设有N个叶子节点）。

| 算法 | 支持模型 | | 树结构 | 特征选择 | | 连续值处理 | 缺失值处理 | 剪枝 |
|---|---|---|---|---|---|---|---|---|
| ID3 | 分类 | | 多叉树 | 信息增益 | | 不支持 | 不支持 | 不支持 |
| C4.5 | 分类 | | 多叉数 | 信息增益比 | | 支持 | 支持 | 支持 |
| CART | 分类 | 回归 | 二叉树 | 基尼指数 | 均方差 | 支持 | 支持 | 支持 |



图2：ID3、C4.5和CART算法比较

分类回归树（classification and regression tree, CART）模型由Breiman等人在1984年提出，是应用广泛的决策树学习方法。CART同样由特征选择、树的生成及剪枝组成，既可以用于分类也可以用于回归。图2给出了三种比较常见决策树的一个比较总结，希望可以帮助大家更好的理解。我们这里重点介绍一下CART回归树算法。

### 2.2 理论解释

假设X和Y分别为输入和输出变量，并且Y是连续变量，给定训练数据集 $D=\{(x_1,y_1),(x_2,y_2),\ldots,(x_N,y_N)\}$ 考虑如何生成回归树。

既然是回归树，那么必然会存在以下两个核心问题：

- **如何选择划分点?**
- **如何决定树中叶节点的输出值?**

一个回归树对应着输入空间（即特征空间）的一个划分以及在划分的单元上的输出值。假设已将输入空间划分为M个单元 $R_1,R_2,\ldots,R_M$ ，并且在每个单元 $R_m$ 上有一个固定的输出值 $c_m$ ，于是回归树模型可以表示为：

$$f(x)=\sum_{m=1}^{M}c_m I(x\in R_m)$$

当输入空间的划分确定时，可以用平方误差 $\sum_{x_i\in R_m}(y_i-f(x_i))^2$ 来表示回归树对于训练数据的预测误差，用平方误差最小的准则求解每个单元上的最优输出值。

的均值，即

$$\hat{c}_m = ave(y_i|x_i \in R_m)$$

**（1）问题1：怎样对输入空间进行划分？即如何选择划分点？**

CART回归树采用启发式的方法对输入空间进行划分，选择第j个变量 $x^{(j)}$ 和它取的值s，作为切分变量（splitting variable）和切分点（splitting point），并定义两个区域：

$$R_1(j,s) = \left\{x|x^{(j)} \le s\right\} \text{ 和 } R_2(j,s) = \left\{x|x^{(j)} > s\right\}$$

然后寻找最优切分变量j和最优切分点s。具体地，求解：

$$\min_{j,s}\left[\min_{c_1}\sum_{x_i \in R_1(j,s)}(y_i - c_1)^2 + \min_{c_2}\sum_{x_2 \in R_2(j,s)}(y_i - c_2)^2\right]$$

对固定输入变量j可以找到最优切分点s。

**（2）问题2：如何决定树中叶节点的输出值？**

用选定的最优切分变量j和最优切分点s划分区域并决定相应的输出值：

$$\hat{c}_1 = ave(y_i|x_i \in R_1(j,s)) \text{ 和 } \hat{c}_2 = ave(y_i|x_i \in R_2(j,s))$$

遍历所有输入变量，找到最优的切分变量j，构成一个对（j, s）。依此将输入空间划分为两个区域。接着，对每个区域重复上述划分过程，直到满足停止条件为止。这样就生成一颗回归树。这样的回归树通常称为最小二乘回归树（least squares regression tree）。

如果已将输入空间划分为M个区域 $R_1, R_2, \ldots, R_M$，并且在每个区域 $R_m$ 上有一个固定的输出值 $\hat{c}_m$，于是回归树模型可以表示为：

$$f(x) = \sum_{m=1}^{M}\hat{c}_m I(x \in R_m)$$

### 2.3 算法流程

输出：回归树 $f(x)$ .

在训练数据集所在的输入空间中，递归地将每个区域划分为两个子区域并决定每个子区域上的输出值，构建二叉决策树：

（1）选择最优切分变量 $j$ 与切分点 $s$ ，求解

$$\min_{j,s}\left[\min_{c_1}\sum_{x_i\in R_1(j,s)}(y_i-c_1)^2+\min_{c_2}\sum_{x_i\in R_2(j,s)}(y_i-c_2)^2\right] \tag{5.21}$$

遍历变量 $j$ ，对固定的切分变量 $j$ 扫描切分点 $s$ ，选择使式 (5.21) 达到最小值的对 $(j,s)$ .

（2）用选定的对 $(j,s)$ 划分区域并决定相应的输出值：

$$R_1(j,s)=\{x\,|\,x^{(j)}\leqslant s\}\ ,\quad R_2(j,s)=\{x\,|\,x^{(j)}>s\}$$

$$\hat{c}_m=\frac{1}{N_m}\sum_{x_i\in R_m(j,s)}y_i\ ,\quad x\in R_m\ ,\quad m=1,2$$

（3）继续对两个子区域调用步骤 (1)，(2)，直至满足停止条件.

（4）将输入空间划分为 $M$ 个区域 $R_1,R_2,\cdots,R_M$ ，生成决策树：

$$f(x)=\sum_{m=1}^{M}\hat{c}_m I(x\in R_m)$$



## 3. 回归树示例

本示例来源于李航著的《统计学习方法》第5章决策树习题中的5.2题。已知如图3所示的训练数据，试用平方误差损失准则生成一个二叉回归树。

| $x_i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| $y_i$ | 4.50 | 4.75 | 4.91 | 5.34 | 5.80 | 7.05 | 7.90 | 8.23 | 8.70 | 9.00 |

图3：训练数据表

寻找最优切分变量j和最优切分点s的方法为：

$$\min_{j,s}\left[\min_{c_1}\sum_{x_i\in R_1(j,s)}(y_i-c_1)^2+\min_{c_2}\sum_{x_2\in R_2(j,s)}(y_i-c_2)^2\right]$$

其中，

$c_1=ave(y_i|x_i\in R_1(j,s))$ 和 $c_2=ave(y_i|x_i\in R_2(j,s))$

例如，取s=1。此时 $R_1=\{1\}$ ， $R_2=\{2,3,4,5,6,7,8,9,10\}$ ，这两个区域的输出值分别为：

$$c_2 = \frac{1}{9}(4.75 + 4.91 + 5.34 + 5.80 + 7.05 + 7.90 + 8.23 + 8.70 + 9.00) = 6.85$$

根据上面的计算方法，可以得到下表：

| s | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| $c_1$ | 4.50 | 4.63 | 4.72 | 4.88 | 5.06 | 5.39 | 5.75 | 5.18 | 6.35 | 6.62 |
| $c_2$ | 6.85 | 7.12 | 7.43 | 7.78 | 8.18 | 8.46 | 8.64 | 8.85 | 9.00 | 0.00 |

把 $c_1, c_2$ 的值代入到均方差中，如下：

$$m(1) = 0 + \{(4.75 - 6.85)^2 + (4.91 - 6.85)^2 + (5.34 - 6.85)^2 + (5.80 - 6.85)^2 + (7.05 - 6.85)^2 + (7.90 - 6.85)^2 + (8.23 - 6.85)^2 + (8.70 - 6.85)^2 + (9.00 - 6.85)^2\} = 22.65$$

同理，可以获得下表：

| s | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| m(s) | 22.65 | 17.70 | 12.19 | 7.38 | 3.36 | 5.07 | 10.05 | 15.18 | 21.33 | 27.63 |

显然取s=5时，m(s)最小。因此，第一个最优切分变量为j=x、最优切分点为s=5。

**1）用选定的（j，s）划分区域，并决定输出值：**

两个划分的区域分别是： $R_1 = \{1, 2, 3, 4, 5\}, R_2 = \{6, 7, 8, 9, 10\}$ 。输出值用公式：

$\hat{c}_1 = ave(y_i | x_i \in R_1(j, s))$ 和 $\hat{c}_2 = ave(y_i | x_i \in R_2(j, s))$

得到 $c_1 = 5.06, c_2 = 8.18$ 。

**2）对两个子区域继续调用算法流程中的步骤（1），（2）**

对 $R_1$ 继续进行划分：

| $x_i$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $y_i$ | 4.50 | 4.75 | 4.91 | 5.34 | 5.80 |



取切分点分别为：[1, 2, 3, 4, 5]，则各个区域的输出值c如下表：

| | | | | | |
|---|---|---|---|---|---|
| $c_1$ | 4.50 | 4.63 | 4.72 | 4.88 | 5.06 |
| $c_2$ | 5.20 | 5.35 | 5.57 | 5.80 | 6.00 |



计算m(s)：

| s | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| m(s) | 0.67 | 0.43 | 0.19 | 0.37 | 1.06 |



s=3时，m(3)最小。

之后的递归过程同上，我就不在赘述啦！最后，如图4所示给出完整的二叉回归树。

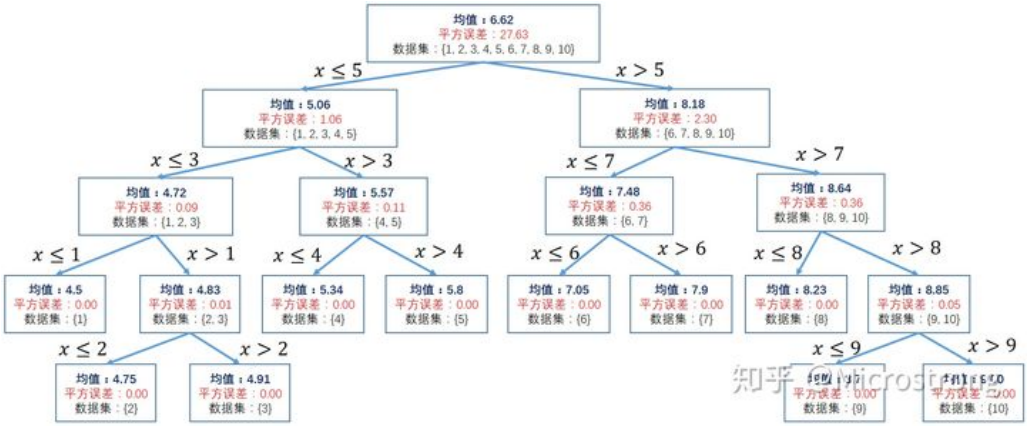





图4：一棵完整的二叉回归树

## 4. 完整的代码示例

本篇文章所有数据集和代码均在我的GitHub中，地址：github.com/Microstrong0...

### 4.1 根据图3的训练数据用Python3实现二叉回归树

```
from numpy import *
```

```
    fr = open(fileName)
    for line in fr.readlines():
        curLine = line.strip().split('\t')
        # python3不适用：fltLine = map(float,curLine) 修改为：
        fltLine = list(map(float, curLine))  # 将每行映射成浮点数，
        dataMat.append(fltLine)
    return dataMat


# 切分数据集为两个子集
def binSplitDataSet(dataSet, feature, value):  # 数据集 待切分特征
    mat0 = dataSet[nonzero(dataSet[:, feature] > value)[0], :]
    mat1 = dataSet[nonzero(dataSet[:, feature] <= value)[0], :]
    # 下面原书代码报错 index 0 is out of bounds,使用上面两行代码
    # mat0 = dataSet[nonzero(dataSet[:, feature] > value)[0], :
    # mat1 = dataSet[nonzero(dataSet[:, feature] <= value)[0],
    return mat0, mat1


# Tree结点类型：回归树
def regLeaf(dataSet):  # 生成叶结点，在回归树中是目标变量特征的均值
    return mean(dataSet[:, -1])


# 误差计算函数：回归误差
def regErr(dataSet):  # 计算目标的平方误差（均方误差*总样本数）
    return var(dataSet[:, -1]) * shape(dataSet)[0]


# 二元切分
def chooseBestSplit(dataSet, leafType=regLeaf, errType=regErr,
    # 切分特征的参数阈值，用户初始设置好
    tolS = ops[0]  # 允许的误差下降值
    tolN = ops[1]  # 切分的最小样本数
    # 若所有特征值都相同，停止切分
    if len(set(dataSet[:, -1].T.tolist()[0])) == 1:  # 倒数第一列
        return None, leafType(dataSet)  # 如果剩余特征数为1，停止切
        # 找不到好的切分特征，调用regLeaf直接生成叶结点
    m, n = shape(dataSet)
    S = errType(dataSet)  # 最好的特征通过计算平均误差
    bestS = inf
    bestIndex = 0
```

```
            for splitVal in set((dataSet[:, featIndex].T.A.tolist()
                mat0, mat1 = binSplitDataSet(dataSet, featIndex, sp
                if (shape(mat0)[0] < tolN) or (shape(mat1)[0] < to
                newS = errType(mat0) + errType(mat1)
                if newS < bestS:  # 更新为误差最小的特征
                    bestIndex = featIndex
                    bestValue = splitVal
                    bestS = newS
        # 如果切分后误差效果下降不大，则取消切分，直接创建叶结点
        if (S - bestS) < tolS:
            return None, leafType(dataSet)  # 停止切分2
        mat0, mat1 = binSplitDataSet(dataSet, bestIndex, bestValue)
        # 判断切分后子集大小，小于最小允许样本数停止切分3
        if (shape(mat0)[0] < tolN) or (shape(mat1)[0] < tolN):
            return None, leafType(dataSet)
        return bestIndex, bestValue  # 返回特征编号和用于切分的特征值


    # 构建tree
    def createTree(dataSet, leafType=regLeaf, errType=regErr, ops=(
        # 数据集默认NumPy Mat 其他可选参数【结点类型：回归树，误差计算函数，
        feat, val = chooseBestSplit(dataSet, leafType, errType, ops
        if feat == None: return val  # 满足停止条件时返回叶结点值
        # 切分后赋值
        retTree = {}
        retTree['spInd'] = feat
        retTree['spVal'] = val
        # 切分后的左右子树
        lSet, rSet = binSplitDataSet(dataSet, feat, val)
        retTree['left'] = createTree(lSet, leafType, errType, ops)
        retTree['right'] = createTree(rSet, leafType, errType, ops)
        return retTree

    if __name__ == "__main__":
        myDat = mat(loadDataSet('5.2test.txt'))
        print(createTree(myDat))

        # 绘制数据点图
        import matplotlib.pyplot as plt

        plt.plot(myDat[:, 0], myDat[:, 1], 'ro')
        plt.show()
```

{'spInd': 0, 'spVal': 5.0, 'left': {'spInd': 0, 'spVal': 7.0, 'left': {'spInd': 0, 'spVal': 8.0, 'left': {'spInd': 0, 'spVal': 9.0, 'left': 9.0, 'right': 8.7}, 'right': 8.23}, 'right': {'spInd': 0, 'spVal': 6.0, 'left': 7.9, 'right': 7.05}}, 'right': {'spInd': 0, 'spVal': 3.0, 'left': {'spInd': 0, 'spVal': 4.0, 'left': 5.8, 'right': 5.34}, 'right': {'spInd': 0, 'spVal': 1.0, 'left': {'spInd': 0, 'spVal': 2.0, 'left': 4.91, 'right': 4.75}, 'right': 4.5}}}

### 4.2 用sklearn实现二叉回归树

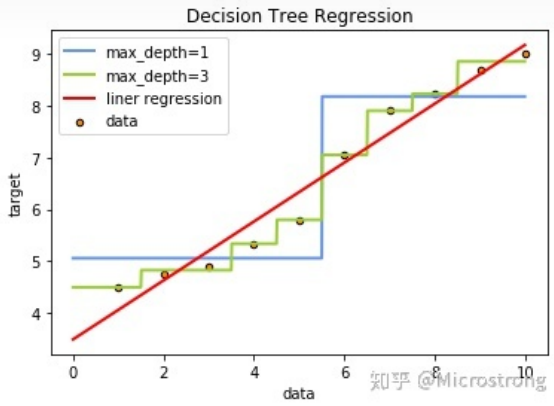
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeRegressor
from sklearn import linear_model

# Data set
x = np.array(list(range(1, 11))).reshape(-1, 1)
y = np.array([4.50, 4.75, 4.91, 5.34, 5.80, 7.05, 7.90, 8.23, 8

# Fit regression model
model1 = DecisionTreeRegressor(max_depth=1)
model2 = DecisionTreeRegressor(max_depth=3)
model3 = linear_model.LinearRegression()
model1.fit(x, y)
model2.fit(x, y)
model3.fit(x, y)

# Predict
X_test = np.arange(0.0, 10.0, 0.01)[:, np.newaxis]
y_1 = model1.predict(X_test)
y_2 = model2.predict(X_test)
y_3 = model3.predict(X_test)

# Plot the results
plt.figure()
plt.scatter(x, y, s=20, edgecolor="black",
            c="darkorange", label="data")
plt.plot(X_test, y_1, color="cornflowerblue",
         label="max_depth=1", linewidth=2)
plt.plot(X_test, y_2, color="yellowgreen", label="max_depth=3",
plt.plot(X_test, y_3, color='red', label='liner regression', li
plt.xlabel("data")
plt.ylabel("target")
plt.title("Decision Tree Regression")
plt.legend()
plt.show()
```




## 5. 关于回归树的若干问题

### （1）CART实现分类树与回归树的区别?

CART分类树是一种二分递归分割的技术，分割方法采用基于最小距离的基尼指数估计函数，将当前的样本集分为两个子样本集，使得生成的的每个非叶子节点都有两个分支。因此，CART算法生成的决策树是结构简洁的二叉树。

CART分类树是针对目标变量是离散型变量，通过二叉树将数据进行分割成离散类的方法。而回归树则是针对目标变量是连续性的变量，通过选取最优分割特征的某个值，然后数据根据大于或者小于这个值进行划分进行树分裂最终生成回归树。

### （2）树形结构为什么不需要归一化?

因为数值缩放不影响分裂点位置，对树模型的结构不造成影响。 按照特征值进行排序的，排序的顺序不变，那么所属的分支以及分裂点就不会有不同。而且，树模型是不能进行梯度下降的，因为构建树模型（回归树）寻找最优点时是通过寻找最优分裂点完成的，因此树模型是阶跃的，阶跃点是不可导的，并且求导没意义，也就不需要归一化。

既然树形结构（如决策树、RF）不需要归一化，那为何非树形结构比如Adaboost、SVM、LR、KNN、K-Means之类则需要归一化?

对于线性模型，特征值差别很大时，运用梯度下降的时候，损失等高线是椭圆形，需要进行多次迭代才能到达最优点。 但是如果进行了归一化，那么等高线就是圆形的，促使SGD往原点迭代，从而导致需要的迭代次数较少。

- **预剪枝**：其中的核心思想就是，在每一次实际对结点进行进一步划分之前，先采用验证集的数据来验证如果划分是否能提高划分的准确性。如果不能，就把结点标记为叶结点并退出进一步划分；如果可以就继续递归生成节点。
- **后剪枝**：后剪枝则是先从训练集生成一颗完整的决策树，然后自底向上地对非叶结点进行考察，若将该结点对应的子树替换为叶结点能带来泛化性能提升，则将该子树替换为叶结点。

在第3节回归树的示例中，我没有对生成的二叉回归树进行剪枝，感兴趣的同学可以自己尝试实现预剪枝和后剪枝，来避免生成的二叉回归树过拟合。

### （4）树分裂的终止条件?

有了选取分割特征和最佳分割点的方法，树便可以依此进行分裂，但是分裂的终止条件是什么呢?

- **节点中所有目标变量的值相同**, 既然都已经是相同的值了自然没有必要再分裂了，直接返回这个值就好了。
- 树的深度达到了预先指定的最大值。
- **不纯度的减小量小于预先定好的阈值，**也就是指进一步的分割数据并不能更好的降低数据不纯度的时候就可以停止树分裂了。
- 节点的数据量小于预先定好的阈值。

### （5）ID3和C4.5能不能用来回归?

CART 是一棵二叉树，那么只要回归树不是一棵二叉树，那么就不是 CART 树了。

在分类问题中，ID3、C4.5 和 CART 的区别就在于划分子节点的策略不同，信息增益、信息增益比、基尼指数；而在回归问题中，用平方误差最小的准则求解每个特征上的最优输出值，这种情况下，分类时的 ID3、C4.5、CART 之间的区别就没了，那么就是每个父节点划分成多少个子节点的问题了，如果还是二叉树，那么就认为是 CART 回归树，否则就不是了。

如果同一个时刻对某一个特征 $x^{(j)}$ 选择两个切分点 $s_1$ 和 $s_2$ 来划分父节点，那么将产生三个区间 $R_1\{j,s_1\},R_2\{j,s_1,s_2\},R_3\{j,s_2\}$ ，这种做法无疑增大了遍历的难度，如果选择更多个切分点，那么遍历的难度会指数上升。如果我们想要细分多个区域，让 CART 回归树更深即可，这样遍历的难度会小很多。所以，固然可以构建非 CART 回归树，但是不如 CART 回归树来的更简单。

实际上，回归树总体流程类似于分类树，分枝时穷举每一个特征的每一个阈值，来寻找最优切分特征和最优切分点，衡量的方法是平方误差最小化。分枝直到达到预设的终止条件为止。

当然，在处理具体的实际问题时，使用单一的回归树肯定是不够的。我们可以利用集成学习中的boosting框架，对回归树进行改良升级，得到的新模型就是提升树（Boosting Decision Tree），在进一步改造，就可以得到梯度提升树（Gradient Boosting Decision Tree，GBDT），再进一步可以升级为XGBoost或者LightGBM。我们在学习这些模型时，可以把它们归为一个学习系列，这样便于我们系统理解模型进展。

## 7. Reference

【1】《统计学习方法》，李航著，P55-P75。

【2】《机器学习实战》，Peter Harrington著，李锐、李鹏、曲亚东、王斌译，P159-P182。

【3】Regression Tree 回归树，地址：blog.csdn.net/weixin_40...

【4】机器学习笔记十二：分类与回归树CART，地址：blog.csdn.net/xierhacke...

【5】回归树（Regression Tree），地址：cnblogs.com/wuliytTaota...

【6】机器学习算法实践-树回归 - 少整酱的文章 - 知乎 zhuanlan.zhihu.com/p/30...

【7】sklearn实现决策树，地址：siyuanblog.com/?...

【8】决策树学习笔记（三）：CART算法，决策树总结，地址：ask.hellobi.com/blog/ai...

【9】决策树(ID3 & C4.5 & CART)及正则剪枝， 地址：mayexia.com/%E6%9C%BA%E...

【10】李航 统计学习方法 第五章 决策树 课后 习题 答案，地址：blog.csdn.net/familyshi...