菊安酱的机器学习第8期

菊安酱的直播间：https://live.bilibili.com/14988341

每周一晚8:00 菊安酱和你不见不散哦~(^o^)/~

更新日期：2018-12-24

作者：菊安酱

课件内容说明：

- 本文为作者原创，转载请注明作者和出处
- 如果想获得此课件及完整视频，可扫描下方二维码，回复"k"进群
- 若有任何疑问，请给作者留言。

直播视频及课件：http://www.peixun.net/view/1278.html

完整版视频及课件：http://edu.cda.cn/course/966

12期完整版课纲

直播时间：每周一晚8:00

直播内容：

时间	期数	算法
2018/11/05	第1期	k-近邻算法
2018/11/12	第2期	决策树
2018/11/19	第3期	朴素贝叶斯
2018/11/26	第4期	Logistic回归
2018/12/03	第5期	支持向量机
2018/12/10	第6期	AdaBoost 算法
2018/12/17	第7期	线性回归
2018/12/24	第8期	树回归
2018/12/31	第9期	K-均值聚类算法
2019/01/07	第10期	Apriori 算法
2019/01/14	第11期	FP-growth 算法
2019/01/21	第12期	奇异值分解SVD

树回归

上一期，介绍的线性回归包含了一些强大的方法，但这些方法创建的模型需要拟合所有的样本点（局部加权线性回归除外）。当数据集特征很多并且特征之间关系复杂时，构建全局模型就十分困难了，也略显笨拙。而且，现实生活中，很多问题其实都是非线性的，不可能使用全局线性模型来拟合任何数据。

全局线性建模困难？想简单点？那就把数据集切分呗~

这个大而化小的切分思想与SMO算法(序列最小优化)的思想有点类似，我们在讲解SVM的时候有讲解过SMO算法，它的核心思想就是：把难以求解的大优化问题分解成多个易于求解的小优化问题，然后将小优化问题按照一定的顺序求解，结果与整体求解结果完全一致。

把数据集切分成很多份易于建模的数据，然后再用线性模型来建模应该就会容易多了。如果首次切分后仍然难以拟合线性模型，那就继续切分。在这样的切分方式下，树结构和回归法就相当有用了。

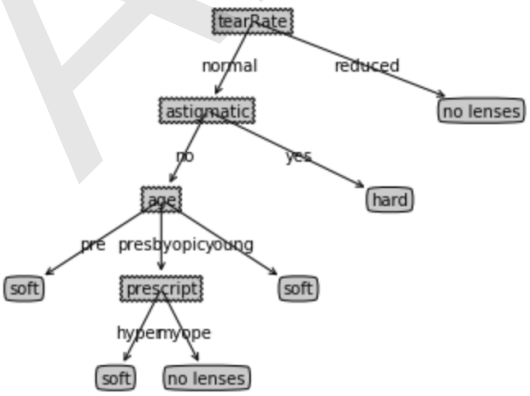
一、回顾决策树（分类）

在第2期，我们讲了如何使用python来构建分类决策树，并利用分类决策树来进行分类。

我们先来回顾一下当时给大家讲的几个的概念：

节点	说明
根节点	没有进边，有出边
中间节点	既有进边也有出边，但进边有且仅有一条，出边也可以有很多条
叶节点	只有进边，没有出边，进边有且仅有一条。**每个叶节点都是一个类别标签**
*父节点和子节点	在两个相连的节点中，更靠近根节点的是父节点，另一个则是子节点。**两者是相对的。**

我们也用了很长的篇幅（6个函数）来讲解决策树的可视化，最后出来的结果是这样的：



当时我们使用的是ID3算法来构建树模型。ID3的做法是：每次选取当前最佳的特征来分割数据，并按照该特征的所有可能取值来切分。也就是说，如果一个特征有4种取值，那么数据将被切分成4份。一旦按某特征切分后，该特征在之后的算法执行过程中将不会再起作用，所以有观点认为这种切分方式过于迅速。

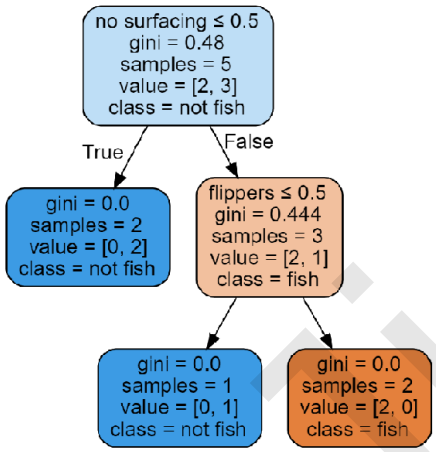
除了切分过于迅速外，ID3算法还存在另一个问题，它不能直接处理连续型特征。只有事先将连续型特征离散化，才能在ID3算法中使用。但这种转换过程会破坏连续型变量的内在特性，也会损失一部分信息。

总结ID3算法缺点：

- 每个特性只能参与一次切分，对后续切分不再起作用
- 不能直接处理连续型特征

构建分类树还有另外一种方法是**二元切分法**，即每次把数据集切成两份。如果数据的某值等于切分所要求的值，那么这些数据就进入树的左子树，反之进入树的右子树。

sklearn就是用这种二元切分法来构建决策树的：



二、CART算法

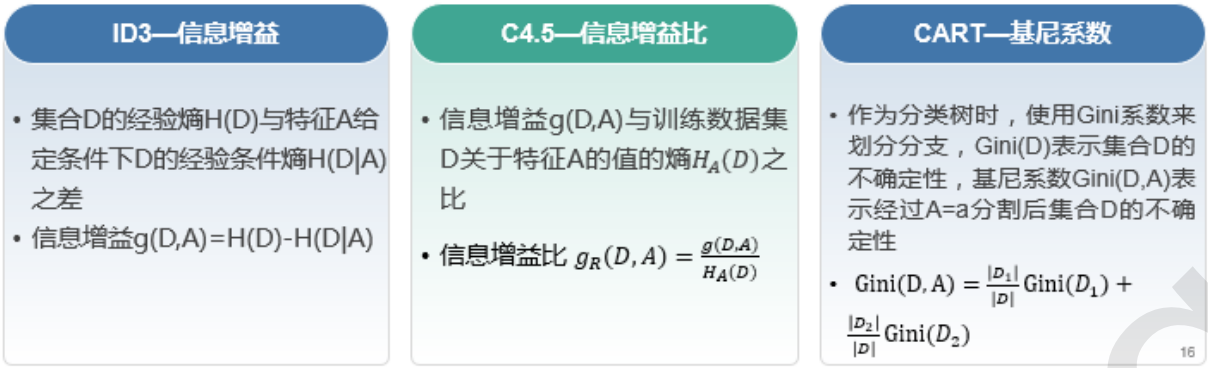
CART是英文**Classification And Regression Tree**的简写，又称为**分类回归树**。从它的名字我们就可以看出，它是一个很强大的算法，既可以用于分类还可以用于回归，所以非常值得我们来学习。

CART算法使用二元切分法来处理连续型变量。而使用二元切分法则易于对树构建过程进行调整以处理连续型特征。具体的处理方法是：如果特征值大于给定值就走左子树，否则就走右子树。

CART算法有两步：

- 决策树生成：递归地构建二叉决策树的过程，基于训练数据集生成决策树，生成的决策树要尽量大；自上而下从根开始建立节点，在每个节点处要选择一个**最好**的属性来分裂，使得子节点中的训练集尽量的纯。
- 决策树剪枝：用验证数据集对已生成的树进行剪枝并选择最优子树，这时损失函数最小作为剪枝的标准。

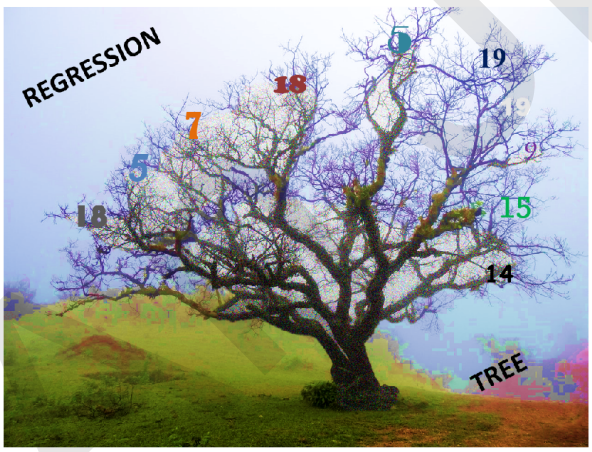
不同的算法使用不同的指标来定义"**最好**"：



【完整版】讲解衡量指标的数学公式

【完整版】CART分类树的python实现

三、CART回归树的python实现



CART树的构建过程：首先找到最佳的列来切分数据集，每次都执行二元切分法，如果特征值大于给定值就走左子树，否则就走右子树，当节点不能再分时就将该节点保存为叶节点。

这里需要大家思考：

什么样的切分方式才叫最佳？也就是衡量'最佳'的指标什么？

叶节点里面存放的是什么？

在这里，我们使用的数据集储存在 ex00.txt 这样一份文本文件中，先将数据集导进来，方便后续使用。

导入相应的包

```
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline
```

导入数据集，并查看数据分布

```
ex00= pd.read_table('ex00.txt',header=None)
plt.scatter(ex00.iloc[:,0].values,ex00.iloc[:,1].values);
```

1. 找到最佳切分列

如何使用CART算法选择最佳切分特征？我们先来看一下寻找最佳切分列函数的伪代码：

```
对每个特征：
    对每个特征值：
        将数据切分成两份（辅助函数1）
        计算切分的误差（辅助函数2）
        如果当前误差小于最小误差，则将当前切分设定为最佳切分并更新最小误差
 返回最佳切分的特征和阈值
```

这里需要两个辅助函数，我们先来构建辅助函数

辅助函数1：切分数据集函数

```
"""
函数说明:根据特征切分数据集合
参数说明:
    dataSet：原始数据集
    feature：待切分的特征索引
    value：该特征的值
返回:
    mat0：切分的数据集合0
    mat1：切分的数据集合1
"""
def binSplitDataSet(dataSet, feature, value):
    mat0 = dataSet.loc[dataSet.iloc[:,feature] > value,:]
    mat1 = dataSet.loc[dataSet.iloc[:,feature] <= value,:]
    return mat0, mat1
```

使用一个简单数据集验证函数运行效果：

```
testMat = pd.read_csv('data.csv',header=None)
testMat
mat0, mat1 = binSplitDataSet(testMat, 0, 1)
```

辅助函数2：计算切分误差函数

该函数的功能是，在给定数据集上计算目标变量的平方误差。这里使用均方差函数var()来计算目标变量的均方误差，由于我们需要的是总方差，所以要用均方误差乘以样本个数。

```
#计算总方差：均方差*样本数
def errType(dataSet):
    var= dataSet.iloc[:,-1].var() *dataSet.shape[0]
    return var
```

用ex00数据集测试函数：

```
errType(ex00)
```

辅助函数3：生成叶节点函数

当我们的最佳切分函数确定不再对数据进行切分时，将调用该函数来得到叶节点的模型。在回归树中，该模型其实就是目标变量的均值。

```
#生成叶节点
def leafType(dataSet):
    leaf = dataSet.iloc[:,-1].mean()
    return leaf
```

同样用ex00数据集测试函数：

```
leafType(ex00)
```

我们的辅助函数都构建好了，然后我们就可以来构建我们的主函数——最佳寻找最佳切分列函数

```
"""
函数说明:找到数据的最佳二元切分方式函数
参数说明:
    dataSet：原始数据集
    leafType：生成叶结点函数
    errType：误差估计函数
    ops：用户定义的参数构成的元组
返回:
    bestIndex：最佳切分特征
    bestValue：最佳特征值
"""
def chooseBestSplit(dataSet, leafType=leafType, errType=errType, ops = (1,4)):
    #tolS允许的误差下降值,tolN切分的最少样本数
    tolS = ops[0]; tolN = ops[1]
    #如果当前所有值相等,则退出。(根据set的特性)
    if len(set(dataSet.iloc[:,-1].values)) == 1:
        return None, leafType(dataSet)
    #统计数据集合的行m和列n
    m, n = dataSet.shape
```

```
    #默认最后一个特征为最佳切分特征,计算其误差估计
    S = errType(dataSet)
    #分别为最佳误差,最佳特征切分的索引值,最佳特征值
    bestS = np.inf; bestIndex = 0; bestValue = 0
    #遍历所有特征列
    for featIndex in range(n - 1):
        colval= set(dataSet.iloc[:,featIndex].values)
        #遍历所有特征值
        for splitVal in colval:
            #根据特征和特征值切分数据集
            mat0, mat1 = binSplitDataSet(dataSet, featIndex, splitVal)
            #如果数据少于tolN,则退出
            if (mat0.shape[0] < tolN) or (mat1.shape[0] < tolN): continue
            #计算误差估计
            newS = errType(mat0) + errType(mat1)
            #如果误差估计更小,则更新特征索引值和特征值
            if newS < bestS:
                bestIndex = featIndex
                bestValue = splitVal
                bestS = newS
    #如果误差减少不大则退出
    if (S - bestS) < tolS:
        return None, leafType(dataSet)
    #根据最佳的切分特征和特征值切分数据集合
    mat0, mat1 = binSplitDataSet(dataSet, bestIndex, bestValue)
    #如果切分出的数据集很小则退出
    if (mat0.shape[0] < tolN) or (mat1.shape[0] < tolN):
        return None, leafType(dataSet)
    #返回最佳切分特征和特征值
    return bestIndex, bestValue
```

运行函数，查看结果：

```
chooseBestSplit(exOO)
```

2. CART算法实现代码

创建函数 createTree() 的伪代码如下：

```
找到最佳的待切分特征：
    如果该节点不能再分，将该节点保存为叶节点
    执行二元切分
    在右子树调用createTree() 方法
    在左子树调用createTree() 方法
```

```
"""
函数功能:树构建函数
参数说明:
    dataSet：原始数据集
```

```
    leafType：建立叶结点的函数
    errType：误差计算函数
    ops：包含树构建所有其他参数的元组
返回：
    retTree：构建的回归树
"""
def createTree(dataSet, leafType = leafType, errType = errType, ops = (1, 4)):
    #选择最佳切分特征和特征值
    col, val = chooseBestSplit(dataSet, leafType, errType, ops)
    #如果没有特征,则返回特征值
    if col == None: return val
    #回归树
    retTree = {}
    retTree['spInd'] = col
    retTree['spVal'] = val
    #分成左数据集和右数据集
    lSet, rSet = binSplitDataSet(dataSet, col, val)
    #创建左子树和右子树
    retTree['left'] = createTree(lSet, leafType, errType, ops)
    retTree['right'] = createTree(rSet, leafType, errType, ops)
    return retTree
```

运行函数，查看函数返回结果：

```
createTree(ex00)
```

从结果中可以看出，这个树只分了一次就结束了。那是因为我们使用的这个ex00数据集比较简单，它只有一个特征。我们换一个稍微复杂点的数据集，再跑一遍看看结果。

```
#导入数据集
ex0 = pd.read_table('ex0.txt',header=None)
ex0.head()

#数据可视化
plt.scatter(ex0.iloc[:,1].values,ex0.iloc[:,2].values);

#创建回归树
mytree = createTree(ex0)
mytree
```

从结果中可以看出，这里生成了5个叶节点。

Sklearn调库实现回归树：

```
from sklearn.tree import DecisionTreeRegressor
from sklearn import linear_model

#用来训练的数据
x = (ex0.iloc[:,1].values).reshape(-1,1)
y = (ex0.iloc[:,-1].values).reshape(-1,1)

# 训练模型
```

```
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)

# 预测
X_test = np.arange(0, 1, 0.01)[:, np.newaxis]
y_1 = model1.predict(X_test)
y_2 = model2.predict(X_test)
y_3 = model3.predict(X_test)

# 可视化结果
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", linewidth=2)
plt.plot(X_test, y_3, color='red', label='liner regression', linewidth=2)
plt.xlabel("data")
plt.ylabel("target")
plt.title("Decision Tree Regression")
plt.legend()
plt.show()
```

【完整版】剪枝

讲解如何剪枝来防止树模型过拟合

【完整版】模型树

构建模型树：回归树，每个叶节点中包含单个值；模型树，每个叶节点中包含一个线性方程

四、使用python的Tkinter库创建GUI

GUI（Graphical User Interface）就是**图形用户界面**，它能够同时支持数据呈现和用户交互。

Tkinter 的GUI由一些小部件(Widget)组成。所谓小部件，指的是文本框(TextBox)、按钮(Button)、标签(Label)、复选按钮(CheckButton)和按钮整数值(IntVar)等对象。

下面我们尝试构建一个简单的GUI

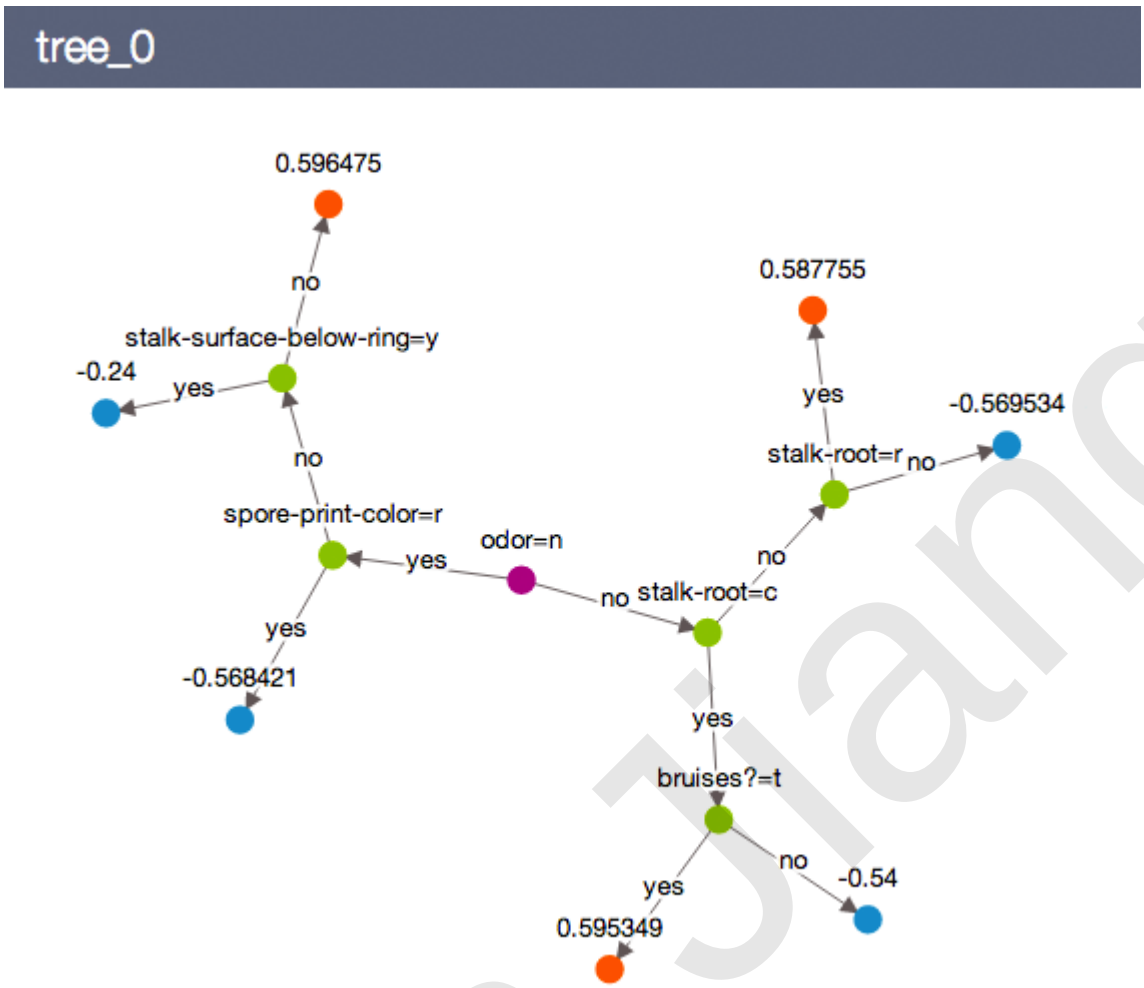
```
import matplotlib
matplotlib.use('TkAgg') #matplotlib后端
import matplotlib.pyplot as plt
from tkinter import *

#占位用，内容后续补充
```

```
def reDraw(tolS,tolN):
    pass

#占位用，内容后续补充
def drawNewTree():
    pass

#实例化一个窗口对象
root = Tk()
#窗口的标题
root.title('回归树调参')
#指定主框体大小
root.geometry('300x200')
Label(root,text='Plot Place Holder').grid(row = 0, columnspan = 3)
#tolN
Label(root,text = 'tolN').grid(row = 1, column = 0)
tolNentry = Entry(root)  #Entry：单行文本输入框
tolNentry.grid(row=1, column = 1)
tolNentry.insert(0,'10')     #默认值为10
#tolS
Label(root,text = 'tolS').grid(row = 2, column = 0)
tolSentry = Entry(root)
tolSentry.grid(row=2, column = 1)
tolSentry.insert(0,'1.0')     #默认值为1.0
#按钮
Button(root,text = 'ReDraw',command = drawNewTree).grid(row = 1, column = 2,rowspan =
3)
#按钮整数值
chkBtnVar = IntVar()
#复选按钮
chkBtn = Checkbutton(root,text = 'Model Tree',variable = chkBtnVar)
chkBtn.grid(row = 3, column = 0,columnspan = 2)
```

【完整版】集成Matplotlib和Tkinter创建GUI

【完整版】回归树的可视化

(Turi上 `graphlab-create` Python包)



【完整版】案例：预测链家二手房价格

（链家网站上爬出来的数据）

其他

- 菊安酱的直播间：https://live.bilibili.com/14988341
- 下周一（2018/12/31）将讲解**Kmeans聚类算法**，欢迎各位进入菊安酱的直播间观看直播
- 如有问题，可以给我留言哦~