有监督学习-决策树

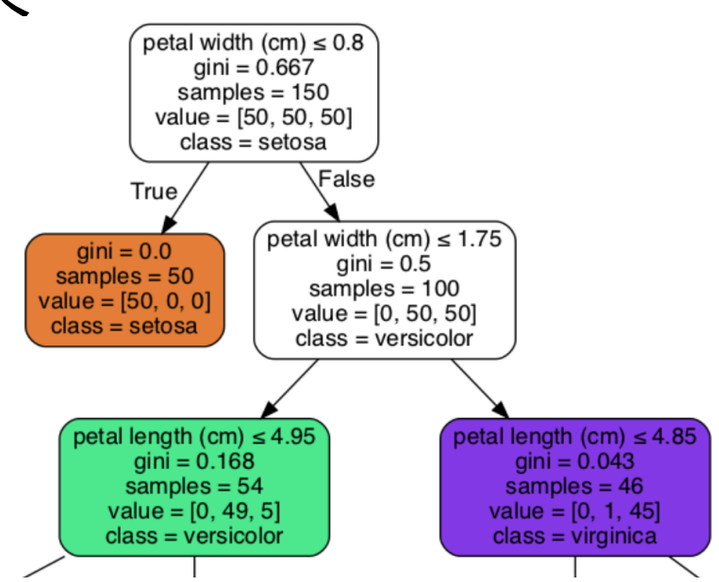
Supervised Learning Decision Tree

- 决策树
- 分类回归树（Classification and Regression Tree, CART）
- 使用决策树模型对鸢尾花种类进行分类

决策树

决策树

- 决策树模型是一个树形结构的模型
  - 每个分支节点都是对某一个树形的决策
  - 每一个分支，则是分支节点的判断结果
- 每一个叶子结点则对一个预测的结果



决策树

- 学习方法：通过对训练数据的学习，来决定如何划分决策树
- 预测过程：从决策树的根结点开始，沿着决策树的分支向叶子节点移动

算法流程

- 决策树采用分治的思想，在每个中间节点都会寻找一个划分
- 停止划分的条件：
  - 当前节点包含的样本属于同一类
  - 属性集为空，或者所有样本在这个属性上的值都一样

算法流程

训练集 T= $\{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$

属性集 A= $\{a_1, a_2, \ldots, a_n\}$

def TreeGrouth(T, A)

生成节点node

if T中的样本都属于同一类别C:
    将node标记为叶子结点，类别为C的叶子结点
    return node

if A为空集 或者 T中的样本在A上取值相同:
    将node节点标记为叶子结点，类别为T中类别最多的类
    return

从A中选取最优的属性节点ai

创建分支节点
根据ai，指定划分规则，为node创建分支，对node进行划分

if 划分后的T的某个子集为空:
    将该分支节点标记为叶子结点，类别为node中最多的类
    return node的分支节点
else
    TreeGrouth(T的子集，A-ai)为node的分支节点

输出：一棵以node为根节点的树

如何划分呢?

- 我们希望决策树的分支节点所包含的样本的类别尽可能的统一，也就是节点的纯度越高越好

- 划分指标
  - 信息增益
  - 基尼指数

信息熵

- 信息熵是衡量信息纯度的常用指标
- 样本T中共有C个类别，每个类别的出现的比例用 p_c 表示
- 则T的信息熵定义为:

$$Ent(T) = -\sum_{i=1}^{|C|} p_c log_2 p_c$$

- Ent(T)的值越小，则T的纯度越高

信息增益

- 对于离散属性a的取值： $a = \{a_1, a_2, \ldots, a_Z\}$
- T_i 定义为：样本T在a上 $a = a_i$ 的集合
- 以属性a对数据集D进行划分所获得的信息增益为：

$$Gain(T, a) = Ent(T) - \sum_{i=1}^{Z} \frac{|T_i|}{|T|} Ent(T_i)$$

- Ent(T) 为划分前的信息熵
- $\frac{|T_i|}{|T|}$ 为第i个分支的权重，样本越多越重要
- $\sum_{i=1}^{Z} \frac{|T_i|}{|T|} Ent(T_i)$ 划分后的信息熵

信息增益

- 所以，我们可以选择信息增益最大的属性作为节点划分的属性

基尼不纯度

- 反映了从T中随机抽取两个样例，其类别标记不一致的概率
- Gini 为基尼指数，Gini(T)越小，数据集T的纯度越高，Pc为出现第k类的概率

$$Gini(T) = \sum_{i=1}^{|C|} p_c(1-p_c) = 1 - \sum_{i=1}^{|C|} p_c^2$$

- 其中满足 $\sum_{k=1}^{|C|} p_c = 1$
- 对于数据集T来说，Gini指数为

$$Gini(T) = 1 - \sum_{i=1}^{|C|} \frac{|C_i|^2}{|T|}$$

基尼不纯度

- 在样本T中，根据特征A的某一个属性值a对样本T进行划分，把T划分T1和T2两部分，则在特征A的条件下，T的基尼表达式为：

$$Gini_index(T,a)=\frac{|T_1|}{T}Gini(|T_1|)+\frac{|T_2|}{|T|}Gini(T_2)$$

- 选择基尼指数最小的属性进行划分

分类回归树（Classification and Regression Tree，CART）

分类回归树（Classification and Regression Tree，CART）

- CART即可以用于回归问题也可以用于分类问题
  - 如果是分类问题：
    - 叶子结点中最多的类别，则是叶子结点预测的类别
  - 如果是回归问题：
    - 叶子结点中输出类别的平均值就是CART的预测结果

CART优点

- 结果易于理解与解释，形象化
- 可以筛选出较为重要的变量
- 可以高速运转
- 因为CART是一棵二叉树，逻辑简单（if…else…）

CART算法基础

- CART是一棵二叉树，也就是说每一个非叶子结点之多只能有两个分支
    - 如果某些有多个值的离散变量，最好提前做好预处理
    - 使用基尼不纯度进行特征选取
    - 有剪枝优化策略（先剪枝和后剪枝）

先剪枝

- 深度达到用户所要的深度
- 节点中样本个数少于用户指定个数
- 不纯度指标下降的最大幅度小于指定的幅度

后剪枝

- 后剪枝则是通过在完全生长的树上剪去分枝实现的，通过删除节点的分支来剪去树节点，可以使用的后剪枝方法有多种，比如：代价复杂性剪枝、最小误差剪枝、悲观误差剪枝等等。

实践

鸢尾花数据

列名	说明	类型
SepalLength	花萼长度	float
SepalWidth	花萼宽度	float
PetalLength	花瓣长度	float
PetalWidth	花瓣宽度	float
Class	类别变量 0:山鸢尾　1:变色鸢尾 2:维吉尼亚鸢尾	int

DecisionTreeClassifier

```
DecisionTreeClassifier(
                       criterion='gini', # 划分标准，'gini'或'entropy'
                       splitter='best',  # 选择split一个node的策略，如果=best，
                                         # 则选择能降低criterion的最好的feature 及 splitting value。
                                         # 如果=random，从几个随机的选项中，选择split效果最好的一个。
                       max_depth=None,   # 设定tree的最大深度
                       min_samples_split=2, # 要split一个node，其必须满足的最小样本量
                       min_samples_leaf=1,  # 一个叶子节点必须拥有的最小样本量
                       max_features=None, # 在寻找最优的split时，所考虑的feature量
                                          # 如果是None，则用全部特征数
                                          # 如果是int，则使用指定的数字
                                          # 如果是float，则使用int(max_features * n_features)
                                          # 如果是sqrt或auto，则使用sqrt(n_features)
                                          # 如果是log2，则使用log2 (n_features)
                       max_leaf_nodes=None,
                       min_impurity_decrease=0.0, # node split前后，criterion差值>该值，则split
                       )
```

鸢尾花分类

- 使用决策树对鸢尾花的类型进行分类
- 要求:
    - 打印模型的准确度
    - 使用不同的参数训练模型
    - 打印树形结构

鸢尾花分类

训练一个决策树分类模型，对鸢尾花的类别进行自动分类

1. 引入必须的包

[26]:
```
import numpy as np
import pydotplus
import matplotlib.pyplot as plt

# 从Ipython.display引入Image，用来显示图片
# 可以用于显示pillow的图片
from IPython.display import Image
from sklearn import tree
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
```

2. 加载数据，对鸢尾花数据进行2，8分

[27]:
```
# Load data
iris = load_iris()
print(iris.data.shape)
X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target,
        test_size = 0.2)
print(X_train.shape)
print(X_test.shape)
```

```
(150, 4)
(120, 4)
(30, 4)
```

鸢尾花分类

3. 训练模型

[28]:

```
clf = DecisionTreeClassifier().fit(X_train, y_train)
predict_test = clf.predict(X_test)

accuracy = clf.score(X_test, y_test.reshape(-1, 1))
print('Accuracy is ', accuracy)

dot_data = tree.export_graphviz(clf, out_file=None,
                                feature_names=iris.feature_names,
                                class_names=iris.target_names,
                                filled=True, rounded=True,
                                special_characters=True)
graph = pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
```

```
Accuracy is  0.9666666666666667
```

[28]: