DecisionTree And RandomForest

DecisionTree

划分标准

> 我们都知道在ID3，和C4.5决策树中，都是基于信息论来对选取最优特征的，那么这个信息熵的计算公式为什么是这样的？下面介绍信息熵公式的产生

----------信息量----------

信息量是对信息的度量，就跟时间的度量是秒一样，当我们考虑一个离散的随机变量x的时候，当我们观察到的这个变量的一个具体值的时候，我们接收到了多少信息呢?

多少信息用信息量来衡量，**我们接受到的信息量跟具体发生的事件有关。**信息的大小跟随机事件的概率有关。

越小概率的事情发生了产生的信息量越大：湖南产生的地震了

越大概率的事情发生了产生的信息量越小，如太阳从东边升起来了

（是不是很玄学，但是确实很有道理）

结论:一个具体事件的信息量应该是随着其发生概率而递减的，且不能为负。

那么问题又来了，这种减函数这么多为什么要用下面的log公式来度量?

$$info(D) = -\sum p_i * \log_2(p_i)$$

如果我们有俩个不相关的事件x和y，那么我们观察到的俩个事件同时发生时获得的信息应该等于观察到的事件各自发生时获得的信息之和，即:

$$info(x,y) = info(x) + info(y)$$

由于x，y是俩个不相关的事件，那么满足 $p(x,y)=p(x)*p(y)$

根据上面推导，**我们很容易看出info(x)一定与p(x)的对数有关（因为只有对数形式的真数相乘之后，能够对应对数的相加形式，可以试试）**。因此我们有信息量公式如下：

$$info(x)=-\log_2 p(x)$$

下面解决俩个疑问?

（1）为什么有一个负号

其中，负号是为了确保信息一定是正数或者是0，总不能为负数吧！

（2）为什么底数为2

这是因为，我们只需要信息量满足低概率事件x对应于高的信息量。那么对数的选择是任意的。我们只是遵循信息论的普遍传统，使用2作为对数的底！

----------信息熵----------

信息量度量的是一个具体事件发生了所带来的信息，而熵则是在结果出来之前对可能产生的信息量的期望——考虑该随机变量的所有可能取值，即所有可能发生事件所带来的信息量的期望。即

$$H(x)=-\sum(p(x)*info(x))$$

即：

$$Info(x)=-\sum p(x)*log_2 p(x)$$

信息增益度

信息增益度是ID3决策树的选择最优划分特征的判断方法，它的思路是：

$$Gain(D,A)=Entorpy(D)-\sum_{v=1}^{V}\frac{|D^v|}{|D|}*Entropy(D^v)$$

选择A特征之前呈现的信息熵Entropy 与 选择A特征之后呈现的信息熵Entorpy(A) 之差 Gain(A),若Gain(A) 越大，说明选择A特征后得到的信息越多，因此选择A特征为最优划分特征

----------信息增益率----------

信息增益度是C4.5决策树的选择最优划分特征的判断方法，它的思路：

$$IV(a)=-\sum_{v}^{V}\frac{|D^v|}{|D|}log\frac{|D^v|}{|D|}$$

$$Gain\_ratio = \frac{Gain(D, A)}{IV(A)}$$

为什么要引入信息增益率的概念，是因为在ID3决策树中，当一个特征他又很多个划分，每个划分下样本的量很少，这时整个样本纯度很高，选择该特征得到的信息增益度会很高，因此在ID3的判断标准下有选择划分很多的特征的趋势。但是较多取值属性带来的问题是，整个模型的泛化能力很弱。因此引入了IV来减少因为划分数量而带来的信息增益度的降低。

实际操作中，通常选取信息增益度TOPN的特征，然后分别计算它们的信息增益率，选择信息增益率最大的为最优划分特征

----------基尼指数----------

基尼指数是CART树的的选择最优划分特征的判断方法，思路如下：

基尼指数是为了信息熵模型中的对数运算，而且保留熵模型的特点(即代表该事物的信息量)而产生的模型。

基尼系数代表了模型的不纯度，基尼系数越小，则不纯度越低，特征越好。这和信息增益(比)是相反的。

$$Gini(p) = \sum_{k=1}^{K} p_k(1 - p_k) = 1 - \sum_{k=1}^{k} p_k^2$$

$$Gini(D, A) = \sum_{v=1}^{V} \frac{|D^v|}{|D|} Gini(D_v)$$

CART树在使用Gini指数时，还额外进行了简化，在CART树下，所有特征只能进行2分类，即使例如特征A有三个划分（A1，A2，A3);在Cart树中，也会将把A分成{A1}和{A2,A3}{A1}和{A2,A3}, {A2}和{A1,A3}{A2}和{A1,A3}, {A3}和{A1,A2}{A3}和{A1,A2}三种情况，找到基尼系数最小的组合，比如{A2}和{A1,A3}{A2}和{A1,A3},然后建立二叉树节点，一个节点是A2对应的样本，另一个节点是{A1,A3}对应的节点。同时，由于这次没有把特征A的取值完全分开，后面我们还有机会在子节点继续选择到特征A来划分A1和A3。

过拟合与剪枝

----------过拟合----------

从直观上来说，只要决策树足够深，划分标准足够细，它在训练集上的表现就能够接近完美；但同时也容易像想象，由于它可能吧训练集的一些"特性"当作所有数据的"共性"来看待，因此它在位置数据上的表现可能比较一般，亦即会出现过拟合的问题。

模型出现过拟合问题一般是因为模型太过复杂。所以决策树解决过拟合问题就是采取适当的"剪枝"。剪枝主要分为"预剪枝（pre-pruning)"和"后剪枝（post-pruning）"。

----------**预剪枝**----------

在建立决策树过程中，其实已经进行了预剪枝的过程，就是我们建树的"停止条件"。通常来说，在建树过程中，我们的停止条件是划分到，当前划分下无其他类别或者已经没有划分特征为止。

当然我们也可以采用交叉验证的方法进行预剪枝，当选出一个划分特征后，我们通过验证集的验证得到在该验证集下的正确率，如果该正确率大于划分前验证集的正确率或者给定的阈值正确率则对其进行划分。否则停止划分。

----------**后剪枝**----------

一般提起的剪枝都是指的后剪枝，它是指在决策树生成完毕后再对其进行修剪，把多余的节点剪掉。换句话说后剪枝是从全局出发，通过某种标准对一些Node进行局部剪枝。从而有效的减少模型的复杂度。通常有两种做法：

- 应用交叉验证的思想，若局部剪枝能够使得模型在测试集上的错误率降低，则进行局部剪枝（预剪枝中也有类似思想，但不同之处是，后剪枝是从低向上，而预剪枝是从顶至下）。
- 应用正则化思想，综合考虑**不确定性**和**模型的复杂程度**来确定一个新的损失，用该损失来作为一个Node是否进行局部剪枝的标准。定义新的损失通常如下

$$C_a(T) = C(T) + a|T|$$

> 其中C(T)即是该Node和不确定性相关的损失，|T|则是该Node下属叶节点的个数,a为惩罚因子。不妨设第t个叶节点含有Nt个样本且这Nt个样本的不确定性为Ht，那么新算是一般可以直接定义为加权不确定性
>
> $$C(T) = \sum_{t=1}^{|T|} N_t * H_t(T)$$

  - C4.5 ，ID3 剪枝方法： 通过比较剪枝前和局部剪枝后的C(T)来决定是否剪枝
  - CART 树剪枝方法有点奇怪： 它是取一系列阈值[C1,C2,C3,.....,Cn]，通过C(T)与阈值比较进行局部剪枝，最后每个阈值C会得到一颗决策树 T，得到一些列决策树[T1,T2,T3.....Tn]，对这一些列决策树进行交叉验证，正确率最高的为最终生成的决策树。

----------**剪枝比较**----------

一般情况下，后剪枝决策树的欠拟合风险很小，泛化性能往往优于预剪枝决策树，但后剪枝过程实在生成完全决策树后进行的，并且要自底向上地对树中的所有非叶节点进行一一考察，因此训练时间开销比未剪枝决策树和预剪枝决策树都要大得多。

Sklearn中的决策树

> 实验源码：
>
> https://github.com/Zrealshadow/MachineLearning/tree/master/LAB2%20DecisionTree

重点：

sklearn中的决策树都是CART树，虽然在参数中有criterion的算法，但是选择"gini"或者"entropy"只是度量方式不同，其结构都是二叉CART树。

> scikit-learn uses an optimised version of the CART algorithm; however, scikit-learn implementation does not support categorical variables for now.
>
> ——官方文档

sklearn用的是优化后的CART树，至今不提供对决策树种类选择的接口

----------决策树分类----------

```
sklearn.tree.DecisionTreeClassifer
```

参考：

https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier

https://www.cnblogs.com/pinard/p/6056319.html

参数	DecisionTreeClassifier
特征选择标准criterion	可以使用"gini"或者"entropy"，前者代表基尼系数，后者代表信息增益。一般说使用默认的基尼系数"gini"就可以了。
特征划分点选择标准splitter	可以使用"best"或者"random"。前者在特征的所有划分点中找出最优的划分点。后者是随机的在部分划分点中找局部最优的划分点。默认的"best"适合样本量不大的时候，而如果样本数据量非常大，此时决策树构建推荐"random"
划分时考虑的最大特征数max_features	可以使用很多种类型的值， 默认是"None",意味着划分时考虑所有的特征数； 如果是"log2"意味着划分时最多考虑log2Nlog2N个特征； 如果是"sqrt"或者"auto"意味着划分时最多考虑N‾√N个特征。如果是整数，代表考虑的特征绝对数。 如果是浮点数，代表考虑特征百分比，即考虑（百分比xN）取整后的特征数。其中N为样本总特征数。 一般来说，如果样本特征数不多，比如小于50，我们用默认的"None"就可以了，如果特征数非常多，我们可以灵活使用刚才描述的其他取值来控制划分时考虑的最大特征数，以控制决策树的生成时间。
	决策树的最大深度，默认可以不输入，如果不输入的话，决策树在

决策树最大深max_depth	建立子树的时候不会限制子树的深度。一般来说，数据少或者特征少的时候可以不管这个值。如果模型样本量多，特征也多的情况下，推荐限制这个最大深度，具体的取值取决于数据的分布。常用的可以取值10-100之间。
内部节点再划分所需最小样本数min_samples_split	这个值限制了子树继续划分的条件，如果某节点的样本数少于min_samples_split，则不会继续再尝试选择最优特征来进行划分。默认是2.如果样本量不大，不需要管这个值。如果样本量数量级非常大，则推荐增大这个值。
叶子节点最少样本数min_samples_leaf	这个值限制了叶子节点最少的样本数，如果某叶子节点数目小于样本数，则会和兄弟节点一起被剪枝。 默认是1,可以输入最少的样本数的整数，或者最少样本数占样本总数的百分比。如果样本量不大，不需要管这个值。如果样本量数量级非常大，则推荐增大这个值。之前的10万样本项目使用min_samples_leaf的值为5，仅供参考。
叶子节点最小的样本权重和min_weight_fraction_leaf	这个值限制了叶子节点所有样本权重和的最小值，如果小于这个值，则会和兄弟节点一起被剪枝。 默认是0，就是不考虑权重问题。一般来说，如果我们有较多样本有缺失值，或者分类树样本的分布类别偏差很大，就会引入样本权重，这时我们就要注意这个值了。
最大叶子节点数max_leaf_nodes	通过限制最大叶子节点数，可以防止过拟合，默认是"None”，即不限制最大的叶子节点数。如果加了限制，算法会建立在最大叶子节点数内最优的决策树。如果特征不多，可以不考虑这个值，但是如果特征分成多的话，可以加以限制，具体的值可以通过交叉验证得到。
类别权重class_weight	指定样本各类别的的权重，主要是为了防止训练集某些类别的样本过多，导致训练的决策树过于偏向这些类别。这里可以自己指定各个样本的权重，或者用“balanced”，如果使用“balanced”，则算法会自己计算权重，样本量少的类别所对应的样本权重会高。当然，如果你的样本类别分布没有明显的偏倚，则可以不管这个参数，选择默认的"None"
节点划分最小不纯度min_impurity_split	这个值限制了决策树的增长，如果某节点的不纯度(基尼系数，信息增益，均方差，绝对差)小于这个阈值，则该节点不再生成子节点。即为叶子节点 。
数据是否预排序presort	这个值是布尔值，默认是False不排序。一般来说，如果样本量少或者限制了一个深度很小的决策树，设置为true可以让划分点选择更加快，决策树建立的更加快。如果样本量太大的话，反而没有什么好处。问题是样本量少的时候，我速度本来就不慢。所以这个值一般懒得理它就可以了。

----------决策树回归----------

```
sklearn.tree.DecisionTreeRegressor
```

在回归树中，API 参数于分类树大致相同，不同点有如下两种

参数	DecisionTreeRegression
特征选择标准 criterion	可以使用"mse"或者"mae"，前者是均方差，后者是和均值之差的绝对值之和。推荐使用默认的"mse"。
类别权重 class_weight	不适用于回归树

----------决策树可视化----------

决策树可视化用到了`graphviz` `pydotplus`两个包

```
#决策树回归建模并评估
regression_tree=DecisionTreeRegressor(max_depth=3,random_state=1)
regression_tree.fit(x_train,y_train)
print("2 feature selection:\n")
evalue(regression_tree,y_test,x_test)


#决策树可视化
dot_data=export_graphviz(regression_tree,out_file=None,\
                         feature_names=
[features[k[0]],features[k[1]]],rounded=True,filled=True)
'''
export_graphviz(decision_tree,out_file,max_depth,feature_names,class_name,
label,filled,leaves_paralled,impurity,node_ids)
@param decision_tree: 可视化的决策树模型
@param out_file: 命名输出文件，默认为None
@param max_depth:最大深度，默认为None 所有可视化
@param feature_names: 特征名称list
@param class_name: 每个特征下的分类的名称 list
@param label: 是否在节点显示不纯度 [all, None,root]
@param filled: True显示分类的主要类，特别是回归的端点值，有多输出的节点的纯度
@param leaves_paralled: True将所有叶子节点展示在树的底端
@param impurity: True 显示每个node的不纯度
@param node_ids: True 显示每个node的编号
@return : dot_data :String
'''
graph=pydotplus.graph_from_dot_data(dot_data)
```

```
graph.write_png('./regressiontreeOf2Features.png')
```

----------决策树算法优缺点----------

优点

- 模型简单直观，适用于展示
- 基本不需要预处理
- 既可以处理离散值，也可以处理连续值
- 可以处理多分类问题
- 可以交叉验证进行剪枝，提高泛化能力
- 对异常点的容忍性好，健壮性高

缺点

- 容易过拟合
- 随着样本的变动而变动
- 难以寻找最优决策树
- 对于复杂的关系，决策树难以学习

RandomForest

理论介绍

随机森林是集成学习的一种算法，它是Bagging算法中的一种，下面将先介绍Bagging算法的步骤，再介绍随机森林的过程。集成学习的思想在大数据竞赛中是十分常见的，因为通常集成模型比单个模型的性能更加优越。Bagging算法的主要思路：**将相同类型但参数不同的弱分类器进行提升**

----------Bootstrap----------

Bootstrap是数理统计中非常重要的一个理论，这里将不会详细介绍其原理，只介绍其过程：

Bootstrap做法：

1. 从X(一个大的样本集)中随机抽出一个样本(每个样本的几率相同)

2. 将该样本的拷贝放入数据集X_j
3. 将该样本放回X中
4. 以上三个步骤重复N次，从而使得X_j中有N个样本。这个过程将对j=1,2,3,.....M都进行一遍，从而我们最终能得到M个含有N个样本的数据集$X_1, X_2, X_3 \ldots\ldots X_M$

----------**Bagging**----------

Bagging 算法的全程是 **Bootstrap Aggregating** ，其思想非常简单

1. 用Bootstrap 生成除M个数据集
2. 用着M个数据集训练出M个弱分类器
3. 最终模型即位这M个弱分类器的简单组合

所谓简单组合：

1. 对于分类问题使用简单的投票表决
2. 对于回归问题则进行简单的取平均

Bagging 算法的特点：

对于"不稳定"（对训练集敏感：若训练样本变化，结果会发生较大变化）的分类器，Bagging算法能够显著的进行提升，这是被大量实验进行证明的。**而且，弱分类器之间的"差异"越大，提升效果更为明显**

----------**RandomForest**----------

上文中介绍了决策树模型的缺点：很容易受到样本变化带来的影响，因此我们用Bagging算法对其进行提升便成为了随机森林算法。很容易猜到在随机森林中，Bagging中的弱分类器M就是一个CART树。算法流程：

- 用Bootstrap生成M个数据集
- 用这M个数据集训练出M课**不进行后剪枝**的决策树，且在每颗决策树的生成过程中，对Node进行划分时，都从可选特征D个中随机挑选出K个特征，然后依着划分特征的一句从K个特征中选出最优特征作为划分标准
- 最终模型即为这M个弱分类器的简单组合。

> 随机森林除了像Bagging算法那样对样本进行随机采样以外，随机森林还**对特征进行了某种意义上的随机采样**。这样做的意义是通过对特征引入随机扰动，可以使个体模型之间的差异进一步增加，从而提升最终模型的泛化能力。

Sklearn中的随机森林

> 实验源码：
>
> [https://github.com/Zrealshadow/MachineLearning/tree/master/LAB2%20DecisionTree](

```
class sklearn.ensemble.RandomForestClassifier(n_estimators='warn',
criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1,
min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True,
oob_score=False, n_jobs=None, random_state=None, verbose=0,
warm_start=False, class_weight=None)

'''

---Paramter---

@param: n_estimators: integer (default=100)
森林中CART树的个数，弱分类器个数

@param: criterion: string (default="gini") ["gini","entropy"]
选择使用gini不纯度来度量还是信 息熵来度量

@param: max_depth: integer or None (default=None)
选择树的最大深度，否则深度无限制

@param: min_sample_split: int,float,(default=2)
限制子树继续划分，某节点样本数小于么min_sample_split则停止划分

@param: min_samples_leaf : int, float, optional (default=1)
叶子节点最少样本数目

@param: min_weight_fraction_leaf : float, optional (default=0.)
叶子节点最小样本权重，小于这个值则被剪枝

@param: max_features : int, float, string or None, optional
(default="auto")
        - If int, then consider `max_features` features at each split.
        - If float, then `max_features` is a fraction and
          `int(max_features * n_features)` features are considered at each
          split.
        - If "auto", then `max_features=sqrt(n_features)`.
        - If "sqrt", then `max_features=sqrt(n_features)` (same as
"auto").
        - If "log2", then `max_features=log2(n_features)`.
        - If None, then `max_features=n_features`.
        最大特征数

@param: max_leaf_nodes : int or None, optional (default=None)
最多叶子节点数，超出的叶子节点将按照不纯度进行剪枝

@param: min_impurity_decrease : float, optional (default=0.)
最小减少不纯度阈值，如果划分节点的不纯度的减少量大于等于该值则进行划分
```

```
41
42  @param: min_impurity_split : float, (default=1e-7)
43  最小不纯度阈值，如果一个节点要进行分裂，其不纯度要大于min_impurity_split
44
45  ------------------RF框架参数-------------------
46  @param: bootstrap : boolean, optional (default=True)
47  是否使用bootstrap，如果否，利用所有数据集进行训练
48
49  @param: oob_score : bool (default=False)
50  是否采用袋外样本来评估模型的好坏,带外样本可以增加模型泛化能力
51
52  @param: n_jobs : int or None, optional (default=None)
53  使用几个事件，感觉应该和分布式有关
54
55
56  ---attribute---
57
58  @attribute: estimators_ : list of DecisionTreeClassifier
59  返回所有子树的列表
60
61  @attribute:n_classes_ : int or list
62  分类个数
63
64  @attribute:n_features_ : int
65  特征个数
66
67  @attribute:feature_importances_: array of shape = [n_features]
68  特征重要程度的array，数值越高，特征越重要
69
70  @attribute:oob_score_ : float
71  采用打底外样本评估模型的分数，当param中oob_score=true时存在
72
73  '''
74
75  class sklearn.ensemble.RandomForestRegressor(n_estimators='warn',
    criterion='mse', max_depth=None, min_samples_split=2, min_samples_leaf=1,
    min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None,
    min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True,
    oob_score=False, n_jobs=None, random_state=None, verbose=0,
    warm_start=False)
76
77  '''
78  大部分参数与上述相同:
79  @param: criterion: string, optional (default="mse") ["mse","mae"]
80  选择标准，方差还是绝对误差
81  '''
82
```