刘建平Pinard

十年码农，对数学统计学，数据挖掘，机器学习，大数据平台，大数据平台应用开发，大数据可视化感兴趣。

scikit-learn随机森林调参小结

在Bagging与随机森林算法原理小结中，我们对随机森林(Random Forest, 以下简称RF）的原理做了总结。本文就从实践的角度对RF做一个总结。重点讲述scikit-learn中RF的调参注意事项，以及和GBDT调参的异同点。

1. scikit-learn随机森林类库概述

在scikit-learn中，RF的分类类是RandomForestClassifier，回归类是RandomForestRegressor。当然RF的变种Extra Trees也有， 分类类ExtraTreesClassifier，回归类ExtraTreesRegressor。由于RF和Extra Trees的区别较小，调参方法基本相同，本文只关注于RF的调参。

和GBDT的调参类似，RF需要调参的参数也包括两部分，第一部分是Bagging框架的参数，第二部分是CART决策树的参数。下面我们就对这些参数做一个介绍。

2. RF框架参数

首先我们关注于RF的Bagging框架的参数。这里可以和GBDT对比来学习。在scikit-learn 梯度提升树(GBDT)调参小结中我们对GBDT的框架参数做了介绍。GBDT的框架参数比较多，重要的有最大迭代器个数，步长和子采样比例，调参起来比较费力。但是RF则比较简单，这是因为bagging框架里的各个弱学习器之间是没有依赖关系的，这减小的调参的难度。换句话说，达到同样的调参效果，RF调参时间要比GBDT少一些。

下面我来看看RF重要的Bagging框架的参数，由于RandomForestClassifier和RandomForestRegressor参数绝大部分相同，这里会将它们一起讲，不同点会指出。

1) **n_estimators**: 也就是弱学习器的最大迭代次数，或者说最大的弱学习器的个数。一般来说n_estimators太小，容易欠拟合，n_estimators太大，计算量会太大，并且n_estimators到一定的数量后，再增大n_estimators获得的模型提升会很小，所以一般选择一个适中的数值。默认是100。

2) **oob_score** :即是否采用袋外样本来评估模型的好坏。默认识False。个人推荐设置为True，因为袋外分数反应了一个模型拟合后的泛化能力。

3) **criterion:** 即CART树做划分时对特征的评价标准。分类模型和回归模型的损失函数是不一样的。分类RF对应的CART分类树默认是基尼系数gini,另一个可选择的标准是信息增益。回归RF对应的CART回归树默认是均方差mse，另一个可以选择的标准是绝对值差mae。一般来说选择默认的标准就已经很好的。

从上面可以看出， RF重要的框架参数比较少，主要需要关注的是 n_estimators，即RF最大的决策树个数。

3. RF决策树参数

下面我们再来看RF的决策树参数，它要调参的参数基本和GBDT相同，如下：

1) RF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"auto",意味着划分时最多考虑$\sqrt{N}$个特征；如果是"log2"意味着划分时最多考虑log_2N个特征；如果是"sqrt"或者"auto"意味着划分时最多考虑$\sqrt{N}$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。一般我们用默认的"auto"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。

2) 决策树最大深度**max_depth**: 默认可以不输入，如果不输入的话，决策树在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。

3) 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。

4) 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。

5）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。

6) 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。

7) 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(基于基尼系数，均方差)小于这个阈值，则该节点不再生成子节点。即为叶子节点 。一般不推荐改动默认值1e-7。

上面决策树参数中最重要的包括最大特征数max_features， 最大深度max_depth， 内部节点再划分所需最小样本数min_samples_split和叶子节点最少样本数min_samples_leaf。

4.RF调参实例

这里仍然使用GBDT调参时同样的数据集来做RF调参的实例，数据的下载地址在这。本例我们采用袋外分数来评估我们模型的好坏。

完整代码参见我的github:https://github.com/ljpzzz/machinelearning/blob/master/ensemble-learning/random_forest_classifier.ipynb

首先，我们载入需要的类库：

```
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.grid_search import GridSearchCV
from sklearn import cross_validation, metrics

import matplotlib.pylab as plt
%matplotlib inline
```

接着，我们把解压的数据用下面的代码载入，顺便看看数据的类别分布。

```
train = pd.read_csv('train_modified.csv')
target='Disbursed' # Disbursed的值就是二元分类的输出
IDcol = 'ID'
train['Disbursed'].value_counts()
```

可以看到类别输出如下，也就是类别0的占大多数。

```
0    19680
1      320
Name: Disbursed, dtype: int64
```

接着我们选择好样本特征和类别输出。

```
x_columns = [x for x in train.columns if x not in [target, IDcol]]
X = train[x_columns]
y = train['Disbursed']
```

不管任何参数，都用默认的，我们拟合下数据看看：

```
rf0 = RandomForestClassifier(oob_score=True, random_state=10)
rf0.fit(X,y)
print rf0.oob_score_
y_predprob = rf0.predict_proba(X)[:,1]
print "AUC Score (Train): %f" % metrics.roc_auc_score(y, y_predprob)
```

输出如下，可见袋外分数已经很高，而且AUC分数也很高。相对于GBDT的默认参数输出，RF的默认参数拟合效果对本例要好一些。

```
0.98005
AUC Score (Train): 0.999833
```

我们首先对n_estimators进行网格搜索：

```
param_test1 = {'n_estimators':range(10,71,10)}
gsearch1 = GridSearchCV(estimator = RandomForestClassifier(min_samples_split=100,
                                  min_samples_leaf=20,max_depth=8,max_features='sqrt'
,random_state=10),
                        param_grid = param_test1, scoring='roc_auc',cv=5)
```

```
gsearch1.fit(X,y)
gsearch1.grid_scores_, gsearch1.best_params_, gsearch1.best_score_
```

输出结果如下：

```
([mean: 0.80681, std: 0.02236, params: {'n_estimators': 10},
  mean: 0.81600, std: 0.03275, params: {'n_estimators': 20},
  mean: 0.81818, std: 0.03136, params: {'n_estimators': 30},
  mean: 0.81838, std: 0.03118, params: {'n_estimators': 40},
  mean: 0.82034, std: 0.03001, params: {'n_estimators': 50},
  mean: 0.82113, std: 0.02966, params: {'n_estimators': 60},
  mean: 0.81992, std: 0.02836, params: {'n_estimators': 70}],
{'n_estimators': 60},
0.8211334476626017)
```

这样我们得到了最佳的弱学习器迭代次数，接着我们对决策树最大深度max_depth和内部节点再划分所需最小样本数min_samples_split进行网格搜索。

```
param_test2 = {'max_depth':range(3,14,2), 'min_samples_split':range(50,201,20)}
gsearch2 = GridSearchCV(estimator = RandomForestClassifier(n_estimators= 60,
                                 min_samples_leaf=20,max_features='sqrt' ,oob_score=True,
random_state=10),
   param_grid = param_test2, scoring='roc_auc',iid=False, cv=5)
gsearch2.fit(X,y)
gsearch2.grid_scores_, gsearch2.best_params_, gsearch2.best_score_
```

输出如下：

```
([mean: 0.79379, std: 0.02347, params: {'min_samples_split': 50, 'max_depth': 3},
  mean: 0.79339, std: 0.02410, params: {'min_samples_split': 70, 'max_depth': 3},
  mean: 0.79350, std: 0.02462, params: {'min_samples_split': 90, 'max_depth': 3},
  mean: 0.79367, std: 0.02493, params: {'min_samples_split': 110, 'max_depth': 3},
  mean: 0.79387, std: 0.02521, params: {'min_samples_split': 130, 'max_depth': 3},
  mean: 0.79373, std: 0.02524, params: {'min_samples_split': 150, 'max_depth': 3},
  mean: 0.79378, std: 0.02532, params: {'min_samples_split': 170, 'max_depth': 3},
  mean: 0.79349, std: 0.02542, params: {'min_samples_split': 190, 'max_depth': 3},
  mean: 0.80960, std: 0.02602, params: {'min_samples_split': 50, 'max_depth': 5},
  mean: 0.80920, std: 0.02629, params: {'min_samples_split': 70, 'max_depth': 5},
  mean: 0.80888, std: 0.02522, params: {'min_samples_split': 90, 'max_depth': 5},
  mean: 0.80923, std: 0.02777, params: {'min_samples_split': 110, 'max_depth': 5},
  mean: 0.80823, std: 0.02634, params: {'min_samples_split': 130, 'max_depth': 5},
  mean: 0.80801, std: 0.02637, params: {'min_samples_split': 150, 'max_depth': 5},
  mean: 0.80792, std: 0.02685, params: {'min_samples_split': 170, 'max_depth': 5},
  mean: 0.80771, std: 0.02587, params: {'min_samples_split': 190, 'max_depth': 5},
  mean: 0.81688, std: 0.02996, params: {'min_samples_split': 50, 'max_depth': 7},
  mean: 0.81872, std: 0.02584, params: {'min_samples_split': 70, 'max_depth': 7},
  mean: 0.81501, std: 0.02857, params: {'min_samples_split': 90, 'max_depth': 7},
  mean: 0.81476, std: 0.02552, params: {'min_samples_split': 110, 'max_depth': 7},
  mean: 0.81557, std: 0.02791, params: {'min_samples_split': 130, 'max_depth': 7},
  mean: 0.81459, std: 0.02905, params: {'min_samples_split': 150, 'max_depth': 7},
  mean: 0.81601, std: 0.02808, params: {'min_samples_split': 170, 'max_depth': 7},
  mean: 0.81704, std: 0.02757, params: {'min_samples_split': 190, 'max_depth': 7},
  mean: 0.82090, std: 0.02665, params: {'min_samples_split': 50, 'max_depth': 9},
  mean: 0.81908, std: 0.02527, params: {'min_samples_split': 70, 'max_depth': 9},
  mean: 0.82036, std: 0.02422, params: {'min_samples_split': 90, 'max_depth': 9},
  mean: 0.81889, std: 0.02927, params: {'min_samples_split': 110, 'max_depth': 9},
  mean: 0.81991, std: 0.02868, params: {'min_samples_split': 130, 'max_depth': 9},
  mean: 0.81788, std: 0.02436, params: {'min_samples_split': 150, 'max_depth': 9},
  mean: 0.81898, std: 0.02588, params: {'min_samples_split': 170, 'max_depth': 9},
  mean: 0.81746, std: 0.02716, params: {'min_samples_split': 190, 'max_depth': 9},
  mean: 0.82395, std: 0.02454, params: {'min_samples_split': 50, 'max_depth': 11},
  mean: 0.82380, std: 0.02258, params: {'min_samples_split': 70, 'max_depth': 11},
  mean: 0.81953, std: 0.02552, params: {'min_samples_split': 90, 'max_depth': 11},
  mean: 0.82254, std: 0.02366, params: {'min_samples_split': 110, 'max_depth': 11},
  mean: 0.81950, std: 0.02768, params: {'min_samples_split': 130, 'max_depth': 11},
```

```
 mean: 0.81887, std: 0.02636, params: {'min_samples_split': 150, 'max_depth': 11},
 mean: 0.81910, std: 0.02734, params: {'min_samples_split': 170, 'max_depth': 11},
 mean: 0.81564, std: 0.02622, params: {'min_samples_split': 190, 'max_depth': 11},
 mean: 0.82291, std: 0.02092, params: {'min_samples_split': 50, 'max_depth': 13},
 mean: 0.82177, std: 0.02513, params: {'min_samples_split': 70, 'max_depth': 13},
 mean: 0.82415, std: 0.02480, params: {'min_samples_split': 90, 'max_depth': 13},
 mean: 0.82420, std: 0.02417, params: {'min_samples_split': 110, 'max_depth': 13},
 mean: 0.82209, std: 0.02481, params: {'min_samples_split': 130, 'max_depth': 13},
 mean: 0.81852, std: 0.02227, params: {'min_samples_split': 150, 'max_depth': 13},
 mean: 0.81955, std: 0.02885, params: {'min_samples_split': 170, 'max_depth': 13},
 mean: 0.82092, std: 0.02600, params: {'min_samples_split': 190, 'max_depth': 13}],
{'max_depth': 13, 'min_samples_split': 110},
0.8242016800050813)
```

我们看看我们现在模型的袋外分数：

```
rf1 = RandomForestClassifier(n_estimators= 60, max_depth=13, min_samples_split=110,
                             min_samples_leaf=20,max_features='sqrt' ,oob_score=True,
random_state=10)
rf1.fit(X,y)
print rf1.oob_score_
```

输出结果为：

```
0.984
```

可见此时我们的袋外分数有一定的提高。也就是时候模型的泛化能力增强了。

对于内部节点再划分所需最小样本数min_samples_split，我们暂时不能一起定下来，因为这个还和决策树其他的参数存在关联。下面我们再对内部节点再划分所需最小样本数min_samples_split和叶子节点最少样本数min_samples_leaf一起调参。

```
param_test3 = {'min_samples_split':range(80,150,20), 'min_samples_leaf':range(10,60,10)}
gsearch3 = GridSearchCV(estimator = RandomForestClassifier(n_estimators= 60, max_depth=13,
                             max_features='sqrt' ,oob_score=True, random_state=10),
   param_grid = param_test3, scoring='roc_auc',iid=False, cv=5)
gsearch3.fit(X,y)
gsearch3.grid_scores_, gsearch3.best_params_, gsearch3.best_score_
```

输出如下：

```
([mean: 0.82093, std: 0.02287, params: {'min_samples_split': 80, 'min_samples_leaf': 10},
 mean: 0.81913, std: 0.02141, params: {'min_samples_split': 100, 'min_samples_leaf': 10},
 mean: 0.82048, std: 0.02328, params: {'min_samples_split': 120, 'min_samples_leaf': 10},
 mean: 0.81798, std: 0.02099, params: {'min_samples_split': 140, 'min_samples_leaf': 10},
 mean: 0.82094, std: 0.02535, params: {'min_samples_split': 80, 'min_samples_leaf': 20},
 mean: 0.82097, std: 0.02327, params: {'min_samples_split': 100, 'min_samples_leaf': 20},
 mean: 0.82487, std: 0.02110, params: {'min_samples_split': 120, 'min_samples_leaf': 20},
 mean: 0.82169, std: 0.02406, params: {'min_samples_split': 140, 'min_samples_leaf': 20},
 mean: 0.82352, std: 0.02271, params: {'min_samples_split': 80, 'min_samples_leaf': 30},
 mean: 0.82164, std: 0.02381, params: {'min_samples_split': 100, 'min_samples_leaf': 30},
 mean: 0.82070, std: 0.02528, params: {'min_samples_split': 120, 'min_samples_leaf': 30},
 mean: 0.82141, std: 0.02508, params: {'min_samples_split': 140, 'min_samples_leaf': 30},
 mean: 0.82278, std: 0.02294, params: {'min_samples_split': 80, 'min_samples_leaf': 40},
 mean: 0.82141, std: 0.02547, params: {'min_samples_split': 100, 'min_samples_leaf': 40},
 mean: 0.82043, std: 0.02724, params: {'min_samples_split': 120, 'min_samples_leaf': 40},
 mean: 0.82162, std: 0.02348, params: {'min_samples_split': 140, 'min_samples_leaf': 40},
 mean: 0.82225, std: 0.02431, params: {'min_samples_split': 80, 'min_samples_leaf': 50},
 mean: 0.82225, std: 0.02431, params: {'min_samples_split': 100, 'min_samples_leaf': 50},
 mean: 0.81890, std: 0.02458, params: {'min_samples_split': 120, 'min_samples_leaf': 50},
 mean: 0.81917, std: 0.02528, params: {'min_samples_split': 140, 'min_samples_leaf': 50}],
{'min_samples_leaf': 20, 'min_samples_split': 120},
0.8248650279471544)
```

最后我们再对最大特征数max_features做调参：

```
```

```
param_test4 = {'max_features':range(3,11,2)}
gsearch4 = GridSearchCV(estimator = RandomForestClassifier(n_estimators= 60, max_depth=13,
min_samples_split=120,
                                  min_samples_leaf=20 ,oob_score=True, random_state=10),
   param_grid = param_test4, scoring='roc_auc',iid=False, cv=5)
gsearch4.fit(X,y)
gsearch4.grid_scores_, gsearch4.best_params_, gsearch4.best_score_
```

输出如下：

([mean: 0.81981, std: 0.02586, params: {'max_features': 3},
 mean: 0.81639, std: 0.02533, params: {'max_features': 5},
 mean: 0.82487, std: 0.02110, params: {'max_features': 7},
 mean: 0.81704, std: 0.02209, params: {'max_features': 9}],
{'max_features': 7},
0.8248650279471544)

用我们搜索到的最佳参数，我们再看看最终的模型拟合：

```
rf2 = RandomForestClassifier(n_estimators= 60, max_depth=13, min_samples_split=120,
                                  min_samples_leaf=20,max_features=7 ,oob_score=True,
random_state=10)
rf2.fit(X,y)
print rf2.oob_score_
```

此时的输出为：

0.984

可见此时模型的袋外分数基本没有提高，主要原因是0.984已经是一个很高的袋外分数了，如果想进一步需要提高模型的泛化能力，我们需要更多的数据。

以上就是RF调参的一个总结，希望可以帮到朋友们。

（欢迎转载，转载请注明出处。欢迎沟通交流： liujianping-ok@163.com）

分类: 0081. 机器学习

标签: 集成学习

刘建平Pinard
关注 - 15
粉丝 - 2677
+加关注

15 0

« 上一篇：Bagging与随机森林算法原理小结
» 下一篇：K-Means聚类算法原理

posted @ 2016-12-11 21:23 刘建平Pinard 阅读(35776) 评论(45) 编辑 收藏

评论列表

#1楼 2017-04-18 21:19 xiaodongdreams 回复 引用

您好，我尝试使用Python3.6运行了您的代码，有个问题想请教您：
y_predprob = gbm1.predict_proba(X)[:,1]
这条代码中gbm1是什么变量？我发现它没有被定义
运行时显示：
NameError: name 'gbm1' is not defined
请问如何修改？

支持(0) 反对(0)

#2楼[楼主] 2017-04-19 08:30 刘建平Pinard 回复 引用

@ xiaodongdreams
你好，gbm1那copy的时候搞错了，应该是rf0,已经改正，感谢指出错误

支持(3) 反对(0)

#3楼 2017-04-19 11:51 xiaodongdreams 回复 引用

您好，我在使用Pycharm调试您的基于Python2.7.x的代码时，出现以下问题：
1.%matplotlib inline 为IPython指令，在Pycharm下无法绘图。
2.应加入np.seterr(divide='ignore', invalid='ignore')代码解除由于训练数据中存在零值和Nan值造成的Warning。
3.调试第五段代码（n_estimators）时，出现以下错误：
ValueError: Parameter values for parameter (n_estimators) need to be a sequence.
请问应该如何修改param_test1 = {'n_estimators': range(10, 71, 10)}进行解决?
万望回复！

支持(0) 反对(0)

#4楼[楼主] 2017-04-19 12:08 刘建平Pinard 回复 引用

@ xiaodongdreams
对于1，建议你用Ipython notebook去测试上面的代码，因为有可视化的部分，用pycharm的话我没有试过。不过你可以删除这句“%matplotlib inline”，不影响你测试。
对于2，这个不报错的，你可以不管它。
对于3，我这边没有这个问题啊，你在命令行试一试?

支持(1) 反对(0)

#5楼 2017-04-19 12:22 xiaodongdreams 回复 引用

谢谢您的解答，具体报错如下：
File "C:/Users/Administrator/PycharmProjects/RandomForest/GBDT.py", line 25, in <module>
gsearch1 = GridSearchCV(estimator=RandomForestClassifier(min_samples_split=100,min_samples_leaf=20, max_depth=8, max_features='sqrt', random_state=10),param_grid =param_test1, scoring='roc_auc', cv=5)
File "E:\Anaconda3\lib\site-packages\sklearn\grid_search.py", line 812, in __init__
_check_param_grid(param_grid)
File "E:\Anaconda3\lib\site-packages\sklearn\grid_search.py", line 348, in _check_param_grid
"to be a sequence.".format(name))
ValueError: Parameter values for parameter (n_estimators) need to be a sequence.

支持(0) 反对(0)

#6楼[楼主] 2017-04-19 18:00 刘建平Pinard 回复 引用

@ xiaodongdreams
看错误是param_test1,没有正确初始化，建议print param_test1 看看里面是不是一个序列

支持(1) 反对(0)

#7楼 2017-04-19 21:00 xiaodongdreams 回复 引用

您好,我的问题终于解决了。我是这样处理的：
1.删除语句：param_test1 = {'n_estimators':range(10,71,10)}，改为直接在GridSearchCV参数设置中输入param_grid ={'n_estimators':[10,20,30,40,50,60,70]}
2.修改语句：
gsearch1.grid_scores_, gsearch1.best_params_, gsearch1.best_score_
将其替换为：
print(gsearch1.cv_results_, gsearch1.best_params_, gsearch1.best_score_)
否则不会输出结果，且在scikit-learn 0.18中grid_scores_被替换为cv_results_，如此修改后，可以复现所有的实验结果。
最后，非常谢谢您的这篇文档，作为机器学习的初学者，我解决问题花了很长时间，但是我很享受解决问题的过程。

支持(3) 反对(0)

#8楼 2017-05-04 14:34 xiaodongdreams 回复 引用

博主您好，我想请问您几个问题：
在您的代码中，'Disbursed'列的值为所期望二元分类的标签值，'ID'列的值为序号，利用x_columns = [x for x in train.columns if x not in [target, IDcol]]实现了使除了'Disbursed'和'ID'外的其余列作为输入，进入随机森林算法进行分类来求得输出，再与'Disbursed'列的值比较求得袋外误差率。
请问我的理解是否正确？若想使用您的代码做其他应用，是否只需替换相应CSV文件和'Disbursed'，'ID'两列属性即可？有了袋外误差率，还需要进行交叉验证证明其泛化能力吗?
万望回复！谢谢!

支持(0) 反对(0)

#9楼[楼主] 2017-05-04 15:54 刘建平Pinard 回复 引用

你好。1）你的理解是正确的!
2）要使用这个代码，的确只需要换掉csv和关注于不需要的列与输出列即可。
3）一般还是需要交叉验证的，如果你按照我上面的方法，交叉验证可以在网格搜索时进行，你查文档就可以发现GridSearchCV有cv参数。

支持(1) 反对(0)

#10楼 2017-05-04 19:21 xiaodongdreams 回复 引用

谢谢博主!

支持(0) 反对(0)

#11楼 2017-05-05 10:56 xiaodongdreams 回复 引用

博主，您好！
gsearch3.grid_scores_, gsearch2.best_params_, gsearch2.best_score_
写错了，应该是：
gsearch3.grid_scores_, gsearch3.best_params_, gsearch3.best_score_ ？

支持(0) 反对(0)

#12楼[楼主] 2017-05-05 11:48 刘建平Pinard 回复 引用

@ xiaodongdreams
感谢指正，的确当时写错了，已修改。

支持(0) 反对(0)

#13楼 2017-05-05 15:42 xiaodongdreams 回复 引用

请问博主，除了oob_score和AUC Score之外，随机森林还有没有其他评估性能的标准？比如准确率和召回率之类的？

支持(0) 反对(0)

#14楼[楼主] 2017-05-05 17:55 刘建平Pinard 回复 引用

@ xiaodongdreams
你好，这些都有的，你去看看sklearn文档的sklearn.metrics包就可以找到了

支持(0) 反对(0)

#15楼 2017-05-25 17:53 xbycz168 回复 引用

楼主，您好，有个问题想请教您，您首先对n_estimators进行网格搜索调参的时候采用的min_samples_split=100, min_samples_leaf=20,max_depth=8等参数都不是默认值，这些值是怎么定的？当参数取这些值时调的n_estimators参数，还适用于去其他值吗？烦请解疑，多谢！！

支持(0) 反对(0)

#16楼[楼主] 2017-05-25 18:01 刘建平Pinard 回复 引用

@ xbycz168
你好，这里取这些值都只是经验值。这些值与你训练样本的量和分布有关，如果样本量很大，最小划分数和叶子样本数最好都增大。如果样本分布比较均匀，没有怎么隔开，那么最大深度也也增加。具体多少要自己去摸索。
由于参数多，经常需要固定一些参数，来调其他参数，这个思路是这样的，估计固定哪些调哪些，也需要自己摸索。

支持(0) 反对(0)

#17楼 2017-05-25 19:09 mmxx 回复 引用

%matplotlib inline
这个错了吧！

支持(0) 反对(0)

#18楼[楼主] 2017-05-25 19:37 刘建平Pinard 回复 引用

@ mmxx
如果你用ipython notebook来跑上面的程序，这句可以帮你把图绘在页面里。如果你不用notebook的话，可以不用这一句。我博客里所有的程序都是用notebook跑的。

支持(1) 反对(0)

#19楼 2017-07-27 09:26 秋风梧 回复 引用

楼主，关于n_estimators太大会过拟合的问题，我看过其他的文章，似乎都是说“树数目越大越好，但会增加计算开销”， 没有涉及到过拟合的问题

支持(0) 反对(0)

#20楼[楼主] 2017-07-27 10:33 刘建平Pinard 回复 引用

@ 秋风梧
感谢指出错误，这里的确写的时候欠思考了。
内容已经改成了下面这样：
“一般来说n_estimators太小，容易欠拟合，n_estimators太大，计算量会太大，并且n_estimators到一定的数量后，再增大n_estimators获得的模型提升会很小，所以一般选择一个适中的数值。”

支持(0) 反对(0)

#21楼 2017-08-14 22:54 活不明白 回复 引用

您好，想请教一个问题，如果不考虑bagging，即用的是所有的样本数据，决策树生成的随机森林会有数量上限吗，比如现在有5个特征，那么是不是只能生成31棵数？(5个里面选1个特征+5个里面选2个特征+5个里面选3个特征+5个里面选4个特征+5个里面选5个特征=31)

支持(0) 反对(0)

#22楼[楼主] 2017-08-15 17:47 刘建平Pinard 回复 引用

@ 活不明白
你好，你说得对，如果没有bagging,那么决策树的棵数是有上限的。bagging可以让决策树的棵数更加灵活，样本量达到一定数量后，决策树的棵数可以非常大。

支持(0) 反对(0)

#23楼 2017-09-09 20:20 zzysli 回复 引用

@ 活不明白
是最多只能生成这么多树吧
因为在生成随机森林的时候，不是有随机选择一部分特征进行，这部分特征应该有对应的最优值在里面，所以觉得，通常情况下，应该到不了最多的决策树个数，如5个特征时31棵树

支持(0) 反对(0)

#24楼 2017-11-16 13:23 zihsin 回复 引用

"在实际调参的过程中，我们常常将n_estimators和learning_rate一起考虑。"

随机森林并没有learning_rate这个参数吧，它的原理也没涉及到梯度下降等优化算法

支持(0) 反对(0)

#25楼[楼主] 2017-11-16 13:35 刘建平Pinard 回复 引用

@ zihsin
感谢指出错误，这段话之前复制了我之前的文章中的n_estimators的内容，已经修改。

支持(0) 反对(0)

#26楼 2018-04-09 22:47 扶瑶直上 回复 引用

你好，请问袋外分数是指什么?

支持(0) 反对(0)

#27楼[楼主] 2018-04-10 11:50 刘建平Pinard 回复 引用

@ 扶瑶直上
你好，由于放回采样理论上会有大约有36.8%的数据没有被采样集采集中(参看原理篇)，这部分没有参与模型拟合，所以对这些数据的预测分数称为袋外分数。

支持(0) 反对(0)

#28楼 2018-04-20 03:18 一心只读圣贤书 回复 引用

您好，博主
我有个问题：设置参数那里，为什么要设置 iid=False?
iid应该是独立同分布的意思，但是在这里交叉验证这里设置不太明白

顺带发现一处错误：决策树参数那里，RF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；
应该是，随机森林max_features的默认值是auto，即'sqrt'，而不是None；不然的话，随机森林的特征随机性就好像体现不出来。决策树的max_features的默认值才是None；

支持(0) 反对(0)

#29楼[楼主] 2018-04-20 12:39 刘建平Pinard 回复 引用

@ 一心只读圣贤书
这儿复制了之前决策树的部分，的确有差异，是我马虎了，感谢指出错误，已经修改。

设置iid为false，这个其实我没有太关注这个参数，之前参考了另一个例子，里面设置了这个值而已。idd应该不是独立同分布，而是说各个交叉验证folder的数据分布是否一致。true就是一致的。

支持(0) 反对(0)

#30楼 2018-04-24 17:02 shaoguang234 回复 引用

博主你好，有点疑问想请问一下。例子中数据集负样本19680，正样本320。正负样本极不均衡。可以得到如果将所有样本都预测成负样本0，那么正确率为19680/20000=0.984。也就是我们通过调参得到最终模型的准确率。那么，我们有怎么知道模型对这个分类结果有多少贡献呢，也许最终就将绝大多数甚至所有样本都预测成了0.

支持(0) 反对(0)

#31楼[楼主] 2018-04-24 22:26 刘建平Pinard 回复 引用

@ shaoguang234
你好，上面的仅仅是示例，没有考虑类别平衡问题。你可以加上样本权重，类别权重后再做分类。

比如sklearn的RandomForestClassifier也有class_weight参数，还有在fit的时候可以加sample_weight

http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier

支持(0) 反对(0)

#32楼 2018-06-17 13:23 杨鑫Youthy 回复 引用

GridSearchCV(estimator = RandomForestClassifier(min_samples_split=100,
min_samples_leaf=20,max_depth=8,max_features='sqrt' ,random_state=10),请问博主，随机森林和决策树

模型中，有随机变量部分吗，为什么需要random_state这个参数，没这个随机种子参数，每次得到的参数都不一样

支持(0) 反对(0)

#33楼[楼主] 2018-06-17 13:40 刘建平Pinard 回复 引用

@ 杨鑫Youthy
你好，随机森林的每颗决策树的训练集是通过采样得到的，所以如果你不限定随机种子，那么每次用于拟合每一颗决策树的训练集都是不同的，这样自然拟合出的模型参数也不一样了。

支持(0) 反对(0)

#34楼 2018-06-17 14:59 杨鑫Youthy 回复 引用

@ 刘建平Pinard
那决策树呢？它有random_state参数吗，我在网格搜索参数时，
param_gird = {"max_depth":range(5,30,5),"max_features":[1,2,3,4],"min_samples_split":range(2,10,2)}
grid_search = GridSearchCV(DecisionTreeClassifier(),param_grid=param_gird,cv=5))
grid_search.fit(X, y)
每次运行，得到的best_params都是变化的，无法确定最优参数，是每次运行cv集划分的不同导致的吗

支持(0) 反对(0)

#35楼[楼主] 2018-06-17 16:30 刘建平Pinard 回复 引用

@ 杨鑫Youthy
你好，我之前做过不少决策树的模型，定了参数和数据集后模型都是唯一的。
这个我觉得的确如你所说，是用了网格搜索的交叉验证导致的。不设置随机种子，每次K folder的划分得到的训练集各不相同，所以训练出的最优参数也不同。

支持(0) 反对(0)

#36楼 2018-07-01 21:01 我是张三啊 回复 引用

博主，你好。有几个小问题想请教一下。
1.你这样的一部分参数一部分参数的调参得到的是否有局部最优解的可能?
2.这样做的原因是全部一起调参速度很慢吗？这种每次选部分参数，选参数时有什么顺序要求吗?
3.其他别的模型也是这样这样部分参数部分参数调的吗?

支持(0) 反对(0)

#37楼[楼主] 2018-07-01 22:43 刘建平Pinard 回复 引用

@ 我是张三啊
你好!
1. 非常有可能，主要是要找全局最优解的话实在太难了，非常耗时还不一定可以得到，有时候一个局部最优解也能满足我们的产品需要。
2. 是的，一起网格搜索的话，这个搜索区域实在是太大的。每次选部分参数，选参数时主要靠经验（直觉），并没有说先选择某些参数调一定好。
3.其他模型的调参原理也是类似的。尤其是要调的参数很多的时候。如果较少，比如2-3个参数，则可以一起网格搜索，尝试找到全局最优的参数。

支持(0) 反对(0)

#38楼 2018-07-08 01:01 海之蓝啊 回复 引用

请问博主有针对多分类的RF算法么，是变成多个一对一还是？谢谢博主

支持(0) 反对(0)

#39楼[楼主] 2018-07-08 22:37 刘建平Pinard 回复 引用

@ 海之蓝啊
你好，多分类的RF算法一般是和softmax回归类似的思路，不是多个一对一，而是一起综合考虑。目前所有的RF算法都支持类似于softmax的多分类的。

支持(0) 反对(0)

#40楼 2018-07-08 23:08 海之蓝啊 回复 引用

@ 刘建平Pinard
那请问如果样本数据很多的情况下 我对其中20%甚至50%的进行多次调参得到较优的准确率时的参数 这组参数放到100%的样本训练数据时跑出来的准确率也会是较优的么

支持(0) 反对(0)

#41楼[楼主] 2018-07-09 10:07 刘建平Pinard 回复 引用

@ 海之蓝啊
你好，这样做大部分时候放到100%样本训练准确率不是最优的。因为本质上数据的重要性要大于模型的重要性，所以你少了一半以上的数据，再怎么调优，也难以达到更多训练数据的模型效果。

当然也不绝对，比如你的训练数据很多都是相互冗余的，这时少一点数据对建模的参数影响不大，可以用部分数据训练。不过这样做似乎没有必要。

支持(1) 反对(0)

#42楼 2018-08-10 13:59 雾霭 回复 引用

请问随机数种子的选取有讲究吗？一般取什么值呢？

支持(0) 反对(0)

#43楼[楼主] 2018-08-12 20:48 刘建平Pinard 回复 引用

@ 雾霭
一般没有讲究。只是如果你需要算法可以重现，并且结果可以比较的话，每次记得使用同一个随机种子即可。

支持(0) 反对(0)

#44楼 2018-09-29 14:26 qianxianyang 回复 引用

您好，麻烦问下：线性相关的特征对于GBDT和randomForest模型有什么负面影响么，进一步两个线性相关性的特征其特征重要性评分都很高，是否需要进一步优化特征？还有是否需要进一步根据模型返回的特征重要性评分去除评分较低的特征？谢谢。

支持(0) 反对(0)

#45楼[楼主] 2018-09-30 09:58 刘建平Pinard 回复 引用

@ qianxianyang
你好，我觉得影响不是很大。线性相关的特征一般对聚类影响会很大，但是对分类和回归的影响并不大。所以完全可以都留下使用，甚至可以进一步根据相关性制造新的出二级特征。

既然两个特征都重要，也没有必要去据模型返回的特征重要性评分去除评分较低的两个特征中的一个。

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

发表评论

昵称: 李威威

评论内容:

提交评论 退出 订阅评论

[Ctrl+Enter快捷键提交]

相关博文:
· scikit-learn 梯度提升树(GBDT)调参小结
· scikit-learn 支持向量机算法库使用小结
· rf调参小结
· Sklearn-RandomForest
· scikit-learn决策树算法类库使用小结

最新新闻:
· 出门问问：以技术之名 应对市场尴尬
· 瓜子二手车被罚1250万 背后暗藏瓜子、人人销量之争
· 谷歌将面临第四起诉讼 过去17个月已被罚77亿美元
· 霍尼韦尔将总部从新泽西搬到北卡罗来纳州的夏洛特
· 富士康究竟裁员多少？苹果破坏力压向整条供应链
» 更多新闻...