

使用scikit-learn解释随机森林算法

发表于 2015-10-09 06:30 | 12748次阅读 | 来源 DataDive | 2 条评论 | 作者 Pedro Fonseca

机器学习 随机森林 算法 scikit-learn treeinterpreter

摘要：机器学习中的随机森林不可被人们忽视，如何将随机森林算法转换为一个“白盒”，就由这篇文章带来深度的讨论。

在以前的一篇博文里，我讨论过如何将随机森林算法转化为一个“白盒”，这样每次预测就能被分解为各项特征的贡献和，即

$$prediction = bias + feature_1 contribution + \dots + feature_n contribution.$$

我多次想找相关的代码。然而，绝大多数的随机森林算法库（包括scikit-learn）不暴露预测过程的树路径（tree paths）。sklearn的实现方法需要一个额外补丁来暴露。庆幸的是，scikit-learn自0.17版起在API中增加了两项功能，使得这个过程相对而言比较容易理解：获取用于预测的所有叶子节点的ID，并存储所有决策树的所有节点的中间值，而不仅仅只存叶子节点的。结合这两步，就可以获取每次独立预测的预测路径，同时根据查看路径来分解预测过程。

代码已经放在github上了，也可以用 pip install treeinterpreter 进行安装。

注意：需要用到仍在开发中的scikit-learn 0.17，你在下面的链接中能找到安装方法<http://scikit-learn.org/stable/install.html#install-bleeding-edge>。

用treeinterpreter分解随机森林预测

我们选一个简单的数据集，训练一个随机森林模型，并用测试集进行预测，然后分解预测过程。

```
[py]
1. from treeinterpreter import treeinterpreter as ti
2. from sklearn.tree import DecisionTreeRegressor
3. from sklearn.ensemble import RandomForestRegressor
4. import numpy as np
5.
6. from sklearn.datasets import load_boston
7. boston = load_boston()
8. rf = RandomForestRegressor()
9. rf.fit(boston.data[:300], boston.target[:300])
```

我们随机挑选两个预测价格不相同的样本。

```
[py]
1. instances = boston.data[[300, 309]]
2. print "Instance 0 prediction:", rf.predict(instances[0])
3. print "Instance 1 prediction:", rf.predict(instances[1])

Instance 0 prediction: [ 30.76]
Instance 1 prediction: [ 22.41]
```

随机森林模型对它们的预测结果迥然不同。这是为什么呢？我们接下来就把预测结果分为偏置项（也就是训练集的平均结果）和单个特征贡献值，以便于观察究竟哪些特征项造成了差异，差异程度有多大。

我们直接调用tree interpreter的predict方法，向其传入模型和数据作为参数。

CSDN官方微信
扫描二维码,向CSDN吐槽
微信号：CSDNnews

程序员移动端订阅下载

每日资讯快速浏览

微博关注

CSDN云计算 北京 朝阳区

加关注

十位值得关注的Java顶级专家<http://t.cn/RyTf1a2>
11月27日 09:17 转发(2) | 评论(2)

[赞][赞][赞]

进击吧程序员：【进击吧！程序员】第2期：作为程序员，GTX 1080、小姐姐、BUG闪避你最喜欢哪个？#程序员# <http://t.cn/RiktEdn>

相关热门文章

热门标签

Hadoop	AWS	移动游戏
Java	Android	iOS
Swift	智能硬件	Docker
OpenStack	VPN	Spark
ERP	IE10	Eclipse
CRM	JavaScript	数据库
Ubuntu	NFC	WAP

下载专辑

javascript电子书

Python 书籍集合

[py]

```
1. prediction, bias, contributions = ti.predict(rf, instances)
```

打印出这些结果：

[py]

```
1. for i in range(len(instances)):
2.     print "Instance", i
3.     print "Bias (trainset mean)", biases[i]
4.     print "Feature contributions:"
5.     for c, feature in sorted(zip(contributions[i],
6.                                 boston.feature_names),
7.                             key=lambda x: -abs(x[0])):
8.         print feature, round(c, 2)
9.     print "-"*20
```

Instance 0

Bias (trainset mean) 25.2849333333

Feature contributions:

RM 2.73

LSTAT 1.71

PTRATIO 1.27

ZN 1.04

DIS -0.7

B -0.39

TAX -0.19

CRIM -0.13

RAD 0.11

INDUS 0.06

AGE -0.02

NOX -0.01

CHAS 0.0

Instance 1

Bias (trainset mean) 25.2849333333

Feature contributions:

RM -4.88

LSTAT 2.38

DIS 0.32

AGE -0.28

TAX -0.23

CRIM 0.16

PTRATIO 0.15

B -0.15

INDUS -0.14

CHAS -0.1

ZN -0.05

NOX -0.05

RAD -0.02

特征贡献值按照其绝对值从大到小排序。我们观察到第一个样本的预测结果较高，正贡献值主要来自RM、LSTAT和PTRATIO特征。第二个样本的预测值则低得多，因为RM特征实际上有很大的负面影响，它不会被其它特征的正面影响所抵消，因此使得预测值要低于数据集的平均水平。

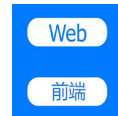
分解的结果真的对吗？很容易检验：偏置和特征贡献值相加应该等于预测值：

[py]

```
1. print prediction
2. print biases + np.sum(contributions, axis=1)
```



ADO.NET书籍整理



Web前端学习电子书



zscat分布式框架

```
[ 30.76 22.41]
[ 30.76 22.41]
```

注意，在把贡献值相加时，我们需要对浮点数进行处理，所以经过四舍五入处理后的值可能略有不同。

比较两个数据集

这个方法的用武之地之一就是比较两个数据集。例如：

- 理解造成两个数据集预测值差异的真正原因，比如是什么因素导致相邻两幢房屋的预测价值差异。
- 调试模型和数据，例如解释为什么新数据的平均预测值和旧数据的不一样。

还是上面这个例子，我们把房价数据的测试集再一分为二，分别计算它们的平均预测价值。

```
[py]
1. ds1 = boston.data[300:400]
2. ds2 = boston.data[400:]
3.
4. print np.mean(rf.predict(ds1))
5. print np.mean(rf.predict(ds2))
```

```
22.1912
18.4773584906
```

我们发现两个数据集的平均预测价值完全不同。现在我们就能细分导致差异的因素：究竟哪些特征项造成了差异，差异程度有多大。

```
[py]
1. prediction1, bias1, contributions1 = ti.predict(rf, ds1)
2. prediction2, bias2, contributions2 = ti.predict(rf, ds2)
```

我们再来计算每一维特征的平均贡献程度。

```
[py]
1. totalc1 = np.mean(contributions1, axis=0)
2. totalc2 = np.mean(contributions2, axis=0)
```

由于两个数据集的偏置项都一样（因为模型的训练集都一样），平均预测价值的差异只能来自于特征的贡献值。换句话说，特征贡献差异的总和应该与平均预测的差异相等，我们很容易验证

```
[py]
1. print np.sum(totalc1 - totalc2)
2. print np.mean(prediction1) - np.mean(prediction2)
```

```
3.71384150943
3.71384150943
```

最后，我们把每一维特征贡献的差异之和显示出来，正好就是平均预测值的差异。

```
[py]
1. for c, feature in sorted(zip(totalc1 - totalc2,
2.                             boston.feature_names), reverse=True):
3.     print feature, round(c, 2)
```

```
LSTAT 2.8
CRIM 0.5
RM 0.5
PTRATIO 0.09
AGE 0.08
NOX 0.03
```

```
B 0.01
CHAS -0.01
ZN -0.02
RAD -0.03
INDUS -0.03
TAX -0.08
DIS -0.14
```

分类树和森林

同样的方法也能用于分类树，查看特征对某个类别的预测概率值的影响力。

我们在iris数据集上做演示。

```
[py]
1. from sklearn.ensemble import RandomForestClassifier
2. from sklearn.datasets import load_iris
3. iris = load_iris()
4.
5. rf = RandomForestClassifier(max_depth = 4)
6. idx = range(len(iris.target))
7. np.random.shuffle(idx)
8.
9. rf.fit(iris.data[idx][:100], iris.target[idx][:100])
```

接着用一个独立样本做预测。

```
[py]
1. instance = iris.data[idx][100:101]
2. print rf.predict_proba(instance)
```

拆分每一维特征的贡献值：

```
[py]
1. prediction, bias, contributions = ti.predict(rf, instance)
2. print "Prediction", prediction
3. print "Bias (trainset prior)", bias
4. print "Feature contributions:"
5. for c, feature in zip(contributions[0],
6.                        iris.feature_names):
7.     print feature, c
```

```
Prediction [[ 0. 0.9 0.1]]
Bias (trainset prior) [[ 0.36 0.262 0.378]]
Feature contributions:
sepal length (cm) [-0.1228614 0.07971035 0.04315104]
sepal width (cm) [ 0. -0.01352012 0.01352012]
petal length (cm) [-0.11716058 0.24709886 -0.12993828]
petal width (cm) [-0.11997802 0.32471091 -0.20473289]
```

我们看到对第二类预测能力最强的特征是花瓣长度和宽度，它们极大提高了预测的概率值。

总结

让随机森林算法的预测结果具有解释性也很容易，几乎达到了线性模型的解释能力。有了 [treeinterpreter](#)，这个步骤只需几行代码就能搞定。

原文地址：[Random forest interpretation with scikit-learn](#)（译者/赵屹华 校检/刘帝伟、朱正贵、李子健 责编/周建丁）

[赵屹华](#)，计算广告工程师@搜狗，前生物医学工程师，关注推荐算法、机器学习领域。

本文为CSDN编译整理，未经允许不得转载，如需转载请联系[market#csdn.net](#)(#换成@)

顶

9

踩

0

推荐阅读相关主题：[工程师](#)[microsoft](#)[random](#)[测试](#)[lambda](#)[google](#)

- 相关文章
- 最新报道
- [\[访谈\] Olivier Grisel谈scikit-learn和机器学习技术的未来](#)
- [WePay机器学习反欺诈实践：Python+scikit-learn+随机森林](#)
- [随机森林不可思议的有效性](#)
- [拓扑数据分析与机器学习的相互促进](#)
- [机器学习算法汇总：人工神经网络、深度学习及其它](#)
- [收藏！斯坦福Andrew Ng教授“机器学习”26篇教程全译](#)

已有2条评论


还可以再输入500个字



有什么感想，你也来说说吧！

您还没有登录! 请 [登录](#) 或 [注册](#)

发表评论

- 最新评论
- 最热评论
- 

weixin_41041753

2017-11-13 13:19

您好，我想请问有没有用随机森林算法对图像进行分类的完整代码。谢谢

回复

- 
- 兴趣使然430

2017-07-12 08:28
- 太牛了，这绝对绝对的好文

回复

请您注意

· 自觉遵守：爱国、守法、自律、真实、文明的原则

· 尊重网上道德，遵守《全国人大常委会关于维护互联网安全的决定》及中华人民共和国其他各项有关法律法规

· 严禁发表危害国家安全，破坏民族团结、国家宗教政策和社会稳定，含侮辱、诽谤、教唆、淫秽等内容的作品

· 承担一切因您的行为而直接或间接导致的民事或刑事责任

· 您在CSDN新闻评论发表的作品，CSDN有权在网站内保留、转载、引用或者删除

· 参与本评论即表明您已经阅读并接受上述条款