

35 | AdaBoost（下）：如何使用AdaBoost对房价进行预测？

2019-03-04 陈旻

数据分析实战45讲

[进入课程 >](#)



讲述：陈旻

时长 08:37 大小 7.90M



今天我带你用 AdaBoost 算法做一个实战项目。AdaBoost 不仅可以用于分类问题，还可以用于回归分析。


我们先做个简单回忆，什么是分类，什么是回归呢？实际上分类和回归的本质是一样的，都是对未知事物做预测。不同之处在于输出结果的类型，分类输出的是一个离散值，因为物体的分类数有限的，而回归输出的是连续值，也就是在一个区间范围内任何取值都有可能。

这次我们的主要目标是使用 AdaBoost 预测房价，这是一个回归问题。除了对项目进行编码实战外，我希望你能掌握：

1. AdaBoost 工具的使用，包括使用 AdaBoost 进行分类，以及回归分析。
2. 使用其他的回归工具，比如决策树回归，对比 AdaBoost 回归和决策树回归的结果。


如何使用 AdaBoost 工具

我们可以直接在 sklearn 中使用 AdaBoost。如果我们要用 AdaBoost 进行分类，需要在使用前引用代码：

 复制代码

```
1 from sklearn.ensemble import AdaBoostClassifier
```

我们之前讲到过，如果你看到了 Classifier 这个类，一般都会对应着 Regressor 类。AdaBoost 也不例外，回归工具包的引用代码如下：

 复制代码

```
1 from sklearn.ensemble import AdaBoostRegressor
```

我们先看下如何在 sklearn 中创建 AdaBoost 分类器。

我们需要使用 AdaBoostClassifier(base_estimator=None, n_estimators=50, learning_rate=1.0, algorithm='SAMME.R', random_state=None) 这个函数，其中有几个比较主要的参数，我分别来讲解下：

1. base_estimator：代表的是弱分类器。在 AdaBoost 的分类器和回归器中都有这个参数，在 AdaBoost 中默认使用的是决策树，一般我们不需要修改这个参数，当然你也可以指定具体的分类器。
2. n_estimators：算法的最大迭代次数，也是分类器的个数，每一次迭代都会引入一个新的弱分类器来增加原有的分类器的组合能力。默认是 50。
3. learning_rate：代表学习率，取值在 0-1 之间，默认是 1.0。如果学习率较小，就需要比较多的迭代次数才能收敛，也就是说学习率和迭代次数是有相关性的。当你调整 learning_rate 的时候，往往也需要调整 n_estimators 这个参数。
4. algorithm：代表我们要采用哪种 boosting 算法，一共有两种选择：SAMME 和 SAMME.R。默认是 SAMME.R。这两者之间的区别在于对弱分类权重的计算方式不同。
5. random_state：代表随机数种子的设置，默认是 None。随机种子是用来控制随机模式的，当随机种子取了一个值，也就确定了一种随机规则，其他人取这个值可以得到同样的结果。如果不设置随机种子，每次得到的随机数也就不同。

那么如何创建 AdaBoost 回归呢？

我们可以使用 `AdaBoostRegressor(base_estimator=None, n_estimators=50, learning_rate=1.0, loss='linear', random_state=None)` 这个函数。

你能看出来回归和分类的参数基本是一致的，不同点在于回归算法里没有 `algorithm` 这个参数，但多了一个 `loss` 参数。

`loss` 代表损失函数的设置，一共有 3 种选择，分别为 `linear`、`square` 和 `exponential`，它们的含义分别是线性、平方和指数。默认是线性。一般采用线性就可以得到不错的效果。

创建好 AdaBoost 分类器或回归器之后，我们就可以输入训练集对它进行训练。我们使用 `fit` 函数，传入训练集中的样本特征值 `train_X` 和结果 `train_y`，模型会自动拟合。使用 `predict` 函数进行预测，传入测试集中的样本特征值 `test_X`，然后就可以得到预测结果。

如何用 AdaBoost 对房价进行预测

了解了 AdaBoost 工具包之后，我们看下 `sklearn` 中自带的波士顿房价数据集。

这个数据集一共包括了 506 条房屋信息数据，每一条数据都包括了 13 个指标，以及一个房屋价位。


13 个指标的含义，可以参考下面的表格：

指标	含义
CRIM	城镇人均犯罪率
ZN	住宅用地比例
INDUS	非零售商业用地比例
CHAS	CHAS变量，0或者1
NOX	一氧化氮浓度
RM	每个住宅的平均房间数
AGE	1940年以前自用房屋的比例
DIS	距离五个波士顿就业中心的加权距离
RAD	距离高速公路的便捷指数
TAX	该地区每一万美元的不动产税率
PRTATIO	该地区教师学生比例
B	该地区黑人比例
LSTAT	该地区中低收入阶层比例

这些指标分析得还是挺细的，但实际上，我们不用关心具体的含义，要做的就是如何通过这 13 个指标推导出最终的房价结果。

如果你学习了之前的算法实战，这个数据集的预测并不复杂。

首先加载数据，将数据分割成训练集和测试集，然后创建 AdaBoost 回归模型，传入训练集数据进行拟合，再传入测试集数据进行预测，就可以得到预测结果。最后将预测的结果与实际结果进行对比，得到两者之间的误差。具体代码如下：

 复制代码

```

1 from sklearn.model_selection import train_test_split
2 from sklearn.metrics import mean_squared_error
3 from sklearn.datasets import load_boston
4 from sklearn.ensemble import AdaBoostRegressor
5 # 加载数据
6 data=load_boston()
7 # 分割数据
8 train_x, test_x, train_y, test_y = train_test_split(data.data, data.target, test_size=0
9 # 使用 AdaBoost 回归模型
10 regressor=AdaBoostRegressor()
11 regressor.fit(train_x,train_y)
12 pred_y = regressor.predict(test_x)
13 mse = mean_squared_error(test_y, pred_y)


```

```

14 print(" 房价预测结果 ", pred_y)
15 print(" 均方误差 = ",round(mse,2))

```

运行结果：

 复制代码

```


1 房价预测结果  [20.2          10.4137931  14.63820225 17.80322581 24.58931298 21.25076923
2  27.52222222 17.8372093  31.79642857 20.86428571 27.87431694 31.09142857
3  12.81666667 24.13131313 12.81666667 24.58931298 17.80322581 17.66333333
4  27.83          24.58931298 17.66333333 20.90823529 20.10555556 20.90823529
5  28.20877193 20.10555556 21.16882129 24.58931298 13.27619048 31.09142857
6  17.08095238 26.19217391  9.975          21.03404255 26.74583333 31.09142857
7  25.83960396 11.859375   13.38235294 24.58931298 14.97931034 14.46699029
8  30.12777778 17.66333333 26.19217391 20.10206186 17.70540541 18.45909091
9  26.19217391 20.10555556 17.66333333 33.31025641 14.97931034 17.70540541
10 24.64421053 20.90823529 25.83960396 17.08095238 24.58931298 21.43571429
11 19.31617647 16.33733333 46.04888889 21.25076923 17.08095238 25.83960396
12 24.64421053 11.81470588 17.80322581 27.63636364 23.59731183 17.94444444
13 17.66333333 27.7253886  20.21465517 46.04888889 14.97931034  9.975
14 17.08095238 24.13131313 21.03404255 13.4          11.859375   26.19214286
15 21.25076923 21.03404255 47.11395349 16.33733333 43.21111111 31.65730337
16 30.12777778 20.10555556 17.8372093  18.40833333 14.97931034 33.31025641
17 24.58931298 22.88813559 18.27179487 17.80322581 14.63820225 21.16882129
18 26.91538462 24.64421053 13.05          14.97931034  9.975          26.19217391
19 12.81666667 26.19214286 49.46511628 13.27619048 17.70540541 25.83960396
20 31.09142857 24.13131313 21.25076923 21.03404255 26.91538462 21.03404255
21 21.16882129 17.8372093  12.81666667 21.03404255 21.03404255 17.08095238
22 45.16666667]
23 均方误差 =  18.05

```

这个数据集是比较规范的，我们并不需要在数据清洗，数据规范化上花太多精力，代码编写起来比较简单。

同样，我们可以使用不同的回归分析模型分析这个数据集，比如使用决策树回归和 KNN 回归。

编写代码如下：

 复制代码

```


1 # 使用决策树回归模型

```



```
2 dec_regressor=DecisionTreeRegressor()
3 dec_regressor.fit(train_x,train_y)
4 pred_y = dec_regressor.predict(test_x)
5 mse = mean_squared_error(test_y, pred_y)
6 print(" 决策树均方误差 = ",round(mse,2))
7 # 使用 KNN 回归模型
8 knn_regressor=KNeighborsRegressor()
9 knn_regressor.fit(train_x,train_y)
10 pred_y = knn_regressor.predict(test_x)
11 mse = mean_squared_error(test_y, pred_y)
12 print("KNN 均方误差 = ",round(mse,2))
```

运行结果：

 复制代码

```
1 决策树均方误差 = 23.84
2 KNN 均方误差 = 27.87
```

你能看到相比之下，AdaBoost 的均方误差更小，也就是结果更优。虽然 AdaBoost 使用了弱分类器，但是通过 50 个甚至更多的弱分类器组合起来而形成的强分类器，在很多情况下结果都优于其他算法。因此 AdaBoost 也是常用的分类和回归算法之一。

AdaBoost 与决策树模型的比较

在 sklearn 中 AdaBoost 默认采用的是决策树模型，我们可以随机生成一些数据，然后对比下 AdaBoost 中的弱分类器（也就是决策树弱分类器）、决策树分类器和 AdaBoost 模型在分类准确率上的表现。

如果想要随机生成数据，我们可以使用 sklearn 中的 `make_hastie_10_2` 函数生成二分类数据。假设我们生成 12000 个数据，取前 2000 个作为测试集，其余作为训练集。

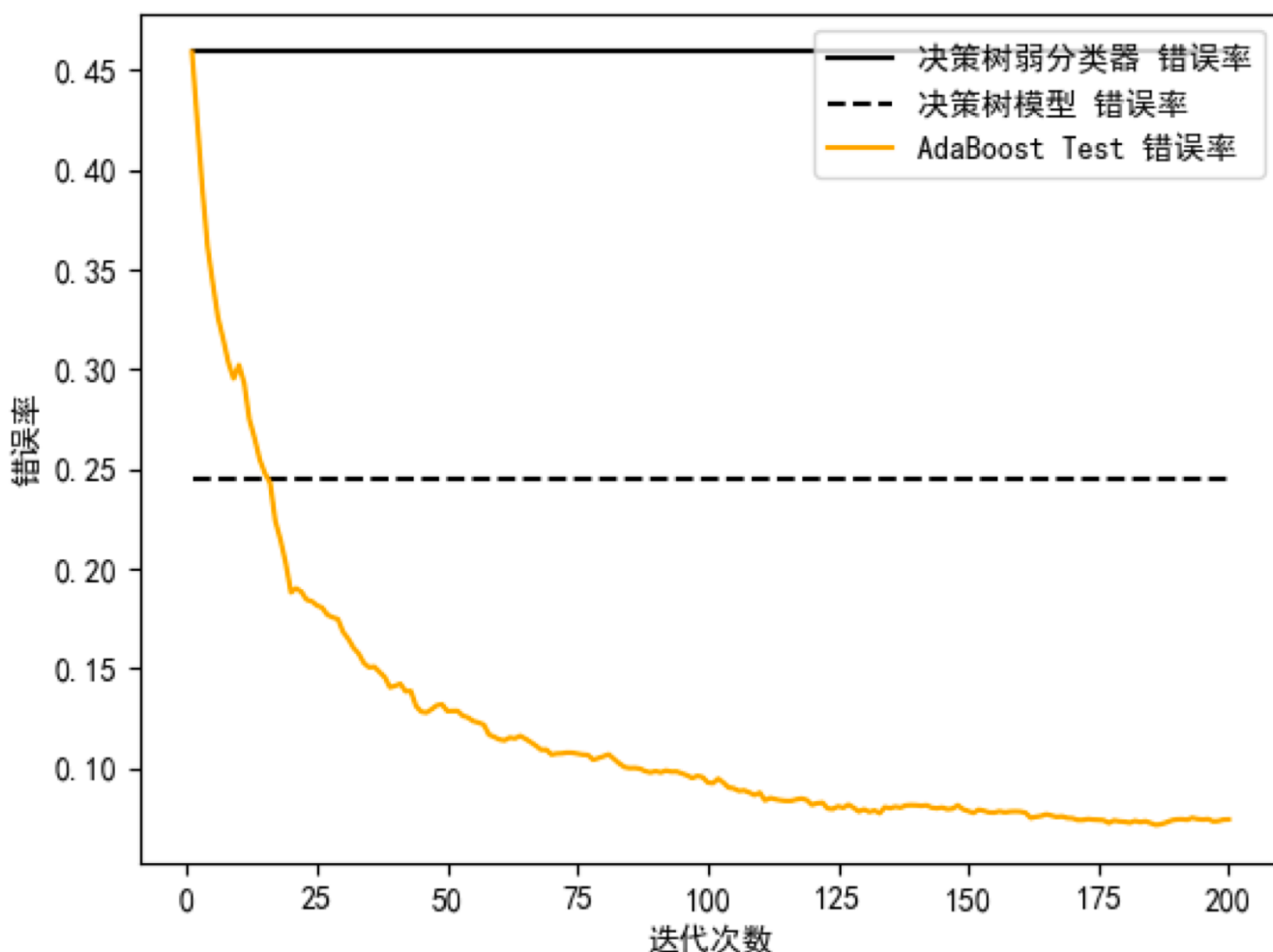
有了数据和训练模型后，我们就可以编写代码。我设置了 AdaBoost 的迭代次数为 200，代表 AdaBoost 由 200 个弱分类器组成。针对训练集，我们用三种模型分别进行训练，然后用测试集进行预测，并将三个分类器的错误率进行可视化对比，可以看到这三者之间的区别：

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 from sklearn import datasets
4 from sklearn.metrics import zero_one_loss
5 from sklearn.tree import DecisionTreeClassifier
6 from sklearn.ensemble import AdaBoostClassifier
7 # 设置 AdaBoost 迭代次数
8 n_estimators=200
9 # 使用
10 X,y=datasets.make_hastie_10_2(n_samples=12000,random_state=1)
11 # 从 12000 个数据中取前 2000 行作为测试集，其余作为训练集
12 test_x, test_y = X[2000:],y[2000:]
13 train_x, train_y = X[:2000],y[:2000]
14 # 弱分类器
15 dt_stump = DecisionTreeClassifier(max_depth=1,min_samples_leaf=1)
16 dt_stump.fit(train_x, train_y)
17 dt_stump_err = 1.0-dt_stump.score(test_x, test_y)
18 # 决策树分类器
19 dt = DecisionTreeClassifier()
20 dt.fit(train_x, train_y)
21 dt_err = 1.0-dt.score(test_x, test_y)
22 # AdaBoost 分类器
23 ada = AdaBoostClassifier(base_estimator=dt_stump,n_estimators=n_estimators)
24 ada.fit(train_x, train_y)
25 # 三个分类器的错误率可视化
26 fig = plt.figure()
27 # 设置 plt 正确显示中文
28 plt.rcParams['font.sans-serif'] = ['SimHei']
29 ax = fig.add_subplot(111)
30 ax.plot([1,n_estimators],[dt_stump_err]*2, 'k-', label=u'决策树弱分类器 错误率')
31 ax.plot([1,n_estimators],[dt_err]*2,'k--', label=u'决策树模型 错误率')
32 ada_err = np.zeros((n_estimators,))
33 # 遍历每次迭代的结果 i 为迭代次数, pred_y 为预测结果
34 for i,pred_y in enumerate(ada.staged_predict(test_x)):
35     # 统计错误率
36     ada_err[i]=zero_one_loss(pred_y, test_y)
37 # 绘制每次迭代的 AdaBoost 错误率
38 ax.plot(np.arange(n_estimators)+1, ada_err, label='AdaBoost Test 错误率', color='orange')
39 ax.set_xlabel('迭代次数')
40 ax.set_ylabel('错误率')
41 leg=ax.legend(loc='upper right',fancybox=True)
42 plt.show()

```

运行结果：



从图中你能看出来，弱分类器的错误率最高，只比随机分类结果略好，准确率稍微大于50%。决策树模型的错误率明显要低很多。而 AdaBoost 模型在迭代次数超过 25 次之后，错误率有了明显下降，经过 125 次迭代之后错误率的变化形势趋于平缓。

因此我们能看出，虽然单独的一个决策树弱分类器效果不好，但是多个决策树弱分类器组合起来形成的 AdaBoost 分类器，分类效果要好于决策树模型。

总结

今天我带你用 AdaBoost 回归分析对波士顿房价进行了预测。因为这是个回归分析的问题，我们直接使用 sklearn 中的 AdaBoostRegressor 即可。如果是分类，我们使用 AdaBoostClassifier。

另外我们将 AdaBoost 分类器、弱分类器和决策树分类器做了对比，可以看出经过多个弱分类器组合形成的 AdaBoost 强分类器，准确率要明显高于决策树算法。所以 AdaBoost 的优势在于框架本身，它通过一种迭代机制让原本性能不强的分类器组合起来，形成一个强分类器。

其实在现实工作中，我们也能找到类似的案例。IBM 服务器追求的是单个服务器性能的强大，比如打造超级服务器。而 Google 在创建集群的时候，利用了很多 PC 级的服务器，将它们组成集群，整体性能远比一个超级服务器的性能强大。

再比如我们讲的“三个臭皮匠，顶个诸葛亮”，也就是 AdaBoost 的价值所在。



今天我们用 AdaBoost 分类器与决策树分类做对比的时候，使用到了 sklearn 中的 make_hastie_10_2 函数生成数据。实际上在[第 19 篇](#)，我们对泰坦尼克号的乘客做生存预测的时候，也讲到了决策树工具的使用。你能不能编写代码，使用 AdaBoost 算法对泰坦尼克号乘客的生存做预测，看看它和决策树模型，谁的准确率更高？

你也可以把这篇文章分享给你的朋友或者同事，一起切磋一下。

数据分析实战 45 讲

即学即用的数据分析入门课

陈旻

清华大学计算机博士



新版升级：点击「👤 请朋友读」，10位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 34 | AdaBoost（上）：如何使用AdaBoost提升分类器性能？

下一篇 36 | 数据分析算法篇答疑

精选留言 (11)

写留言



Destroy、

2019-03-05

源代码中：

从 12000 个数据中取前 2000 行作为测试集，其余作为训练集

```
test_x, test_y = X[2000:], y[2000:]
```

```
train_x, train_y = X[:2000], y[:2000]
```

...

展开 ▾

7



third

2019-03-04

结果仍然为AdaBoost算法最优。

2

个人发现，前两个分类器出结果很快

分析最优：

- 1.AdaBoost算法经过了更多运算，特别是在迭代弱分类器和组合上
- 2.良好组合起来的个体，能够创造更大的价值。...

展开 ▾



王彬成

2019-03-04

👍 1

由于乘客测试集缺失真实值，采用 K 折交叉验证准确率

运行结果：

决策树弱分类器准确率为 0.7867

决策树分类器准确率为 0.7813...

展开 ▾



滢

2019-04-21

👍

得到结果：

CART决策树K折交叉验证准确率: 0.39480897860892333

AdaBoostK折交叉验证准确率: 0.4376641797318339

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

展开 ▾



滨滨

2019-04-21

👍

分类和回归都是做预测，分类是离散值，回归是连续值

展开 ▾



hlz-123

2019-03-27

👍

老师，在AdaBoost 与决策树模型的比较的例子中，弱分类器

```
dt_stump = DecisionTreeClassifier(max_depth=1,min_samples_leaf=1)
```

为什么两个参数都设置为1，相当于只有1个根节点，2个叶节点？

而普通的决策树分类器，没有设置参数，这是什么原因？



叮当猫

2019-03-19



fit_transform数据统一处理，求问什么时候需要？
在我同时没有进行fit_transform的情况下，准确率：
决策树弱分类器的准确率是0.7867
决策树分类器的准确率是0.7734
AdaBoost分类器的准确率是0.8161...

展开 ▾



JingZ

2019-03-05



AdaBoost

一开始竟然蓦然惯性用了AdaBoostRegressor，得到0.33的准确率，最后看了小伙伴代码，立马修正

感觉算法代码不复杂，关键要自己从空白开始写，还需多实战...

展开 ▾



FORWARD-M...

2019-03-05



老师，房价预测这个算法，50个弱分类器是怎么来的？

展开 ▾



梁林松

2019-03-04



跑第二块代码是需要引入两个模块
`from sklearn.tree import DecisionTreeRegressor`
`from sklearn.neighbors import KNeighborsRegressor`

展开 ▾



佳佳的爸

2019-03-04



你好老师，完整的源代码在哪里可以下载到？我说的是每节课里边的源代码。

展开 ▾

