第113天： Python XGBoost 算法项目实战

原创 戴景波 Python技术 1月13日

Python 实现机器学习

如果你的机器学习预测模型表现得不尽如人意，那就用XGBoost。XGBoost算法现在已经成为很多数据工程师的重要武器。

XGBoost 算法

说到XGBoost，不得不提GBDT(Gradient Boosting Decision Tree)。

因为XGBoost本质上还是一个GBDT，但是力争把速度和效率发挥到极致，所以叫X (Extreme) GBoosted。

包括前面说过，两者都是boosting方法。XGBoost高效地实现了GBDT算法并进行了算法和工程上的许多改进，被广泛应用在Kaggle竞赛及其他许多机器学习竞赛中并取得了不错的成绩。

这一篇最适合刚刚接触XGBoost的人阅读，通过一个实战项目拆解整个XGBoost算法。

对于第71天爬取到的数据如下：（作为训练集FBP_train.csv） ID列为每一场比赛，从0或1开始递增，ysb、li等代表易胜博、立博等公司的初始主队获胜赔率，最后一列y代表比赛结果，1代表主队获胜，2代表主队不胜。

以下代码构造训练集，爬取新的预测集放到FBP_predict.csv

```
    trainFilePath = 'E:/PythonLearn/pc_ex/AdaBoost_PeiLv/FBP_train.csv'
    testFilePath = 'E:/PythonLearn/pc_ex/AdaBoost_PeiLv/FBP_predict.csv'
    data = pd.read_csv(trainFilePath)
    X_test= pd.read_csv(testFilePath)
###############第二处调参：选择全部特征还是部分特征############################
    X_train=data[[f0,f1,f2,f3,f4,f5,f6,f7,f8,f9,f10]]#全特征
    y_train=data['y']
    trainandTest(X_train, y_train,X_test)
```

将数据加载到对应变量后通过参数传递给trainandTest函数：其中train_test_split函数是sklearn.cross_validation模块中用来按比例划分训练集和测试集，第一个参数X是被划分的样本特征集;

第二个参数y是被划分的样本标签;

第三个参数test_size是样本占比;

第四个参数是随机数的种子，其实就是该组随机数的编号，在需要重复试验的时候，保证得到一组一样的随机数。比如你每次都填1，其他参数一样的情况下你得到的随机数组是一样的。

但填0或不填，每次都会不一样。随机数的产生取决于种子，随机数和种子之间的关系遵从以下两个规则：种子不同，产生不同的随机数；种子相同，即使实例不同也产生相同的随机数。

model=XGBClassifier()为 XGBoost训练过程。此处用无参数的函数。当然也可以自己手动调整里边的参数，但需多次回测才能调整最优参数，建议用无参数的XGBClassifier方法让系统自己选择最优参数。

```
def trainandTest(X, y,X_t):
    X_train, X_test, y_train, y_test=train_test_split(X, y, test_size=0.33, random_state=7)
    #####特征向量化############
    vec=DictVectorizer(sparse=False)
    ###########归一化和标准化##################
    X_train=vec.fit_transform(X_train.to_dict(orient='record'))
    X_test=vec.transform(X_t.to_dict(orient='record'))
    model=XGBClassifier()#无参数
    model.fit(X_train,y_train)
    # 对测试集进行预测
    ans = model.predict(X_test)
    ans_len = len(ans)    id_list = np.arange(5953, 5956)
    data_arr = []
    for row in range(0, ans_len):
        data_arr.append([int(id_list[row]), ans[row]])
        print(ans[row])
    np_data = np.array(data_arr)
    # 写入文件
    pd_data = pd.DataFrame(np_data, columns=['id', 'y'])
    pd_data.to_csv('FBP_submit.csv', index=None)
    plot_importance(model)
```

首先通过train_test_split函数按照33%的比例分割训练集X和对应的结果标签y，得到新的训练集X_train和对应的结果标签y_train;

再经过DictVectorizer将赔率特征转化为向量形式，再经过fit_transform可以理解为先fit拟合后transform标准化，同理对预测集X_t也进行拟合-标准化。

这时就可以新建模型model了，按照拟合标准化后的X_train训练集进行进一步拟合模型，用此模型对标准化的预测集进行预测了。

将返回的预测结果存到列表data_arr，再通过DataFrame存到CSV文件中。

此外通过plot_importance(model)绘制特征重要性。一个已训练的xgboost模型能够自动计算特征重要性，这些重要性得分可以通过成员变量feature_importances_得到。

梯度提升算法是如何计算特征重要性的？使用梯度提升算法的好处是在提升树被创建后，可以相对直接地得到每个属性的重要性得分。

一般来说，重要性分数，衡量了特征在模型中的提升决策树构建中价值。一个属性越多的被用来在模型中构建决策树，它的重要性就相对越高。

属性重要性是通过对数据集中的每个属性进行计算，并进行排序得到。在单个决策书中通过每个属性分裂点改进性能度量的量来计算属性重要性，由节点负责加权和记录次数。

也就说一个属性对分裂点改进性能度量越大（越靠近根节点），权值越大；被越多提升树所选择，属性越重要。

性能度量可以是选择分裂节点的Gini纯度，也可以是其他度量函数。最终将一个属性在所有提升树中的结果进行加权求和后然后平均，得到重要性得分。

注：(github.com.cn/acredjb/FBP有完整机器学习源码)

以上对XGBoost算法在实际一个项目上做了应用分析，接下来从宏观层面介绍一下另外两个算法AdaBoost和GBDT算法。

首先作为一种常用的统计学习方法，提升方法是将多个弱学习器提升（boost）为一个强学习器的算法。

其工作机制是通过一个弱学习算法，从初始训练集中训练出一个弱学习器，再根据弱学习器的表现对训练样本分布进行调整，

使得先前弱学习器做错的训练样本在后续受到更多的关注，然后基于调整后的样本分布来训练下一个弱学习器。

如此反复学习 ，得到一系列的弱学习器，然后 组合这些弱学习器，构成一个强学习器。

提升方法生成的弱学习器之间存在强依赖关系，必须串行生成一系列的弱学习器。

目前提升方法主要有 AdaBoost,GBDT,XGBoost等算法。而通常我们所知的Bagging，随机森林等算法生成的弱学习器之间不存在强依赖关系，可同时生成并行化的方法。

AdaBoost 算法

通常对提升方法来说，有两个问题需要回答：一是如何在每一轮改变训练样本的分布，二是如何将弱分类器组合成一个强分类器。

对于第一个问题，AdaBoost算法的做法是提高被前一轮弱分类器错误分类样本的权值，而降低被正确分类样本的权值。

这样，没有得到正确分类的数据由于权值被加大而受到后轮的弱分类器更大关注。

对于第二个问题，AdaBoost算法的做法是AdaBoosting采取 加权多数表决的方法。

具体就是加大分类误差小的弱分类器的权值，使其在表决中起较大的作用，减小分类误差率大的弱分类器的权值。

GBDT 梯度提升树

对于提升树，当损失函数是平方误差损失函数和指数损失函数时，每一步优化比较简单，但是对于一般损失函数，往往每一步优化没那么容易。

通常优化损失函数，是希望损失函数能够不断的减小，而且是损失函数能够尽可能快的减小。每轮迭代的时候，都去拟合损失函数在当前模型下的负梯度。

这样每轮训练的时候都能够让损失函数尽可能快的减小，尽快的收敛达到局部最优解或者全局最优解。

Adaboost 与 GBDT 算法比较

Adaboost算法的模型是一个弱学习器线性组合，特点是通过迭代，每一轮学习一个弱学习器，在每次迭代中，提高那些被前一轮分类器错误分类的数据的权值，降低正确分类的数据的权值。

最后，将弱分类器的线性组合作为强分类器，给分类误差小的基本分类器大的权值。每一次迭代都可以减少在训练集上的分类误差率。

AdaBoost能够有效的降低偏差，能够在泛化性能非常弱的学习器上构建成很强的集成。缺点是对噪声敏感。

GBDT与Adaboost的主要差别为，Adaboost每轮学习的一个基本学习器是通过改变样本的权值，关注上轮分类错误的样本的权值，以逐步减少在训练集上的分类误差率。

而GBDT每轮学习一个基本学习器是通过改变输出值，每轮拟合的值为真实值与已有的加法模型的差值（即残差）。

GBDT无论是进行分类还是回归问题，都用的CART树。对于分类问题用二叉分类树，回归问题用二叉回归树。

GBDT 与 XGboost 比较

传统GBDT在优化时只用到一阶导数信息，xgboost则对代价函数进行了二阶泰勒展开，同时用到了一阶和二阶导数。xgboost工具支持自定义代价函数，只要函数可一阶和二阶求导。

Xgboost在代价函数里加入了正则项，用于控制模型的复杂度。正则项里包含了树的叶子节点个数、每个叶子节点上输出的score的L2模的平方和。

从Bias-variance tradeoff角度来讲，正则项降低了模型variance，使学习出来的模型更加简单，防止过拟合，这也是xgboost优于传统GBDT的一个特性。

正则化包括了两个部分，都是为了防止过拟合，剪枝是都有的，叶子结点输出L2平滑是新增的。还可以进行列抽样，从而降低过拟合的风险。

XGboost可进行并行计算，但是这种并行计算并不是像随机森林那样是在tree粒度上去进行，而是在特征粒度上并行。

总结

我们以一个实战项目为依托，从零开始，深入浅出，让读者能够实践Python机器学习的整个过程，此外介绍了AdaBoost，GBDT，XGBoost等算法并对之进行了详细比较。

代码地址

> 示例代码：https://github.com/JustDoPython/python-100-day/tree/master/day-113

系列文章

第 112 天：机器学习算法之蒙特卡洛
第 111 天：Python 垃圾回收机制
从 0 学习 Python 0 - 110 大合集总结

PS：公号内回复：Python，即可进入Python 新手学习交流群，一起**100天计划！**

-END-

Python 技术

关于 Python 都在这里