访问： 380567次
积分： 5021
等级： BLOG 6
排名： 第6266名

原创： 73篇 转载： 268篇
译文： 1篇 评论： 33条

文章搜索

文章分类

- 读书 (1)
- C++ (13)
- C (1)
- Java (27)
- php (2)
- Python (52)
- R (4)
- Scala (4)
- Perl (7)
- 调试 (2)
- 面试 (52)
- Linux (28)
- Mac (7)
- Windows (4)
- 前端 (1)

xgboost入门与实战（原理篇）

标签： xgboost

2017-01-13 15:50 2045人阅读 评论(0) 收藏 举报

分类：

Machine Learning（39）

目录(?) [+]

本文转载自:http://blog.csdn.net/sb19931201/article/details/52557382

前言：

> xgboost是大规模并行boosted tree的工具，它是目前最快最好的开源boosted tree工具包，比常见的工具包快10倍以上。在数据科学方面，有大量kaggle选手选用它进行数据挖掘比赛，其中包括两个以上kaggle比赛的夺冠方案。在工业界规模方面，xgboost的分布式版本有广泛的可移植性，支持在YARN, MPI, Sungrid Engine等各个平台上面运行，并且保留了单机并行版本的各种优化，使得它可以很好地解决于工业界规模的问题。

花了几天时间粗略地看完了xgboost原论文和作者的slide讲解，仅仅是入门入门入门笔记。给我的感觉就是xgboost算法比较复杂，针对传统GBDT算法做了很多细节改进，包括损失函数、正则化、切分点查找算法优化、稀疏感知算法、并行化算法设计等等。本文主要介绍xgboost基本原理以及与传统gbdt算法对比总结，后续会基于Python版本做了一些实战调参试验。想详细学习xgboost算法原理建议通读作者原始论文与slide讲解。

相关文献资料：

Xgboost Slides

XGBoost中文版原理介绍

原始论文XGBoost: A Scalable Tree Boosting System

XGBoost Parameters (official guide)

精彩博文：

XGBoost浅入浅出——wepon

xgboost: 速度快效果好的boosting模型

算法 (41)
NLP (20)
推荐 (0)
Data mining (21)
Machine Learning (40)
Deep Learning (15)
软件开发设计 (1)
hadoop (4)
hive (7)
数据库 (20)
NoSQL (2)
面试题 数学 (6)
心路历程 (3)
生活技巧 (1)
日常学习工作 (15)
职场思考 (5)
语言基础 (2)
工具 (1)

文章存档

2017年11月 (1)
2017年10月 (4)
2017年09月 (10)
2017年08月 (15)
2017年07月 (4)

展开

阅读排行

mysql忽略主键冲突、避免重... (11089)
windows 目录表示（上级目录... (8661)
R语言曲线拟合函数（绘图） (7795)
时间序列完全教程（R） (7748)
Word 表格换页自动"续表"方法 (7185)
论文中如何让页眉上自动显示... (6841)
用深度学习（CNN RNN Atten... (5849)
java 将数字转成百分比（%）... (5386)
网络爬虫工作原理分析 (5226)
Python中的replace方法 (5051)

评论排行

2016小米校招笔试题 (4)
使用Python的Swampy程序包... (4)
在windows下安装scala出现错... (4)
vim中按Ctrl+s 终端疑似卡死 (2)
初学Python，Python中的整数... (2)
滴滴全球Di-Tech算法大赛落幕... (2)
即将步入2015年的一些想法 (2)

Complete Guide to Parameter Tuning in XGBoost (with codes in Python)

一、xgboost基本原理介绍

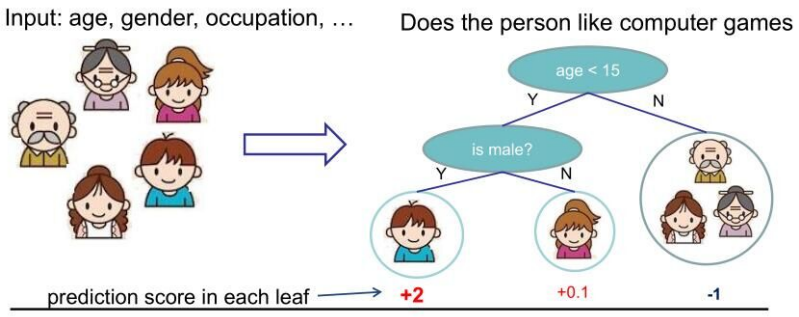
1.提升方法是一种非常有效的机器学习方法，在前几篇笔记中介绍了提升树与GBDT基本原理，xgboost（eXtreme Gradient Boosting）可以说是提升方法的完全加强版本。xgboost算法在各大比赛中展现了强大的威力，引用原论文中的一段描述：

> The impact of the system has been widely recognized in a number of machine lear
> mining challenges. Take the challenges hosted by the machine learning competitio
> for example. Among the 29 challenge winning solutions published at Kaggle's blog
> 17 solutions used XGBoost. Among these solutions, eight solely used XGBoost to tr
> model,while most others combined XGBoost with neural nets in ensembles. For comparison, the
> second most popular method,deep neural nets, was used in 11 solutions. The succe
> system was also witnessed in KDDCup 2015, where XGBoost was used by every winning team in
> the top-10.Moreover, the winning teams reported that ensemble methods outperfo
> configured XGBoost by only a small amount.

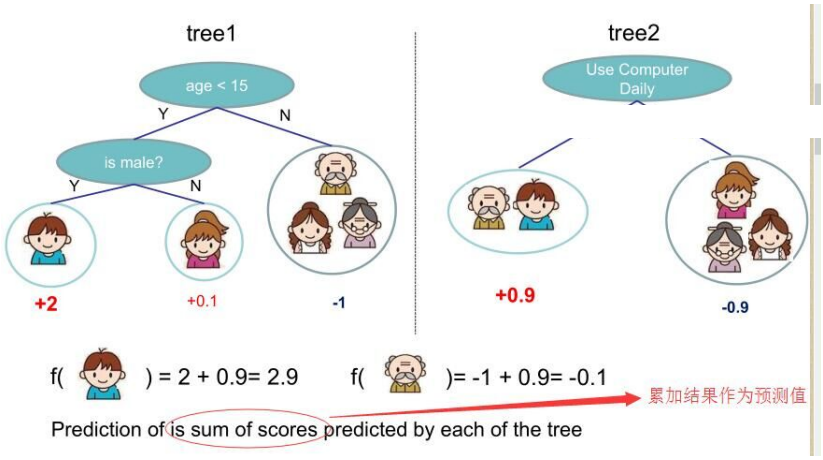
2.Regression Tree and Ensemble (What are we Learning，得到学习目标)

（1）.Regression Tree (CART)回归树

- regression tree (also known as classification and regression tree):
  - Decision rules same as in decision tree
  - Contains one score in each leaf value



（2）.Regression Tree Ensemble 回归树集成



关闭

在上面的例子中，我们用两棵树来进行预测。我们对于每个样本的预测结果就是每棵树预测分数的和。

（3）.Objective for Tree Ensemble 得到学习目标函数

- Model: assuming we have K trees

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in \mathcal{F}$$

- Objective

$$Obj = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

Training loss　　Complexity of the Trees

这里是构造一个目标函数，然后我们要做的就是去尝试优化这个目标函数。读到这里，感觉与gbdt[illegible]

有什么区别，确实如此，不过在后面就能看到他们的不同了（构造（学习）模型参数）。

3.Gradient Boosting (How do we Learn，如何学习)

（1）.So How do we Learn?

目标函数：

$$\sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k} \Omega(f_k), f_k \in \mathcal{F}$$

We can not use methods such as SGD, to find f (since they are trees, instead of just numerical vectors)
Solution: Additive Training (Boosting)
Start from constant prediction, add a new function each time

- Start from constant prediction, add a new function each time

$$\begin{aligned}
\hat{y}_i^{(0)} &= 0 \\
\hat{y}_i^{(1)} &= f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i) \\
\hat{y}_i^{(2)} &= f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i) \\
&\cdots \\
\hat{y}_i^{(t)} &= \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)
\end{aligned}$$

New function

Model at training round t　　Keep functions added in previous round

（2）.Additive Training

关闭

• How do we decide which f to add?
Optimize the objective!! 目标优化

- The prediction at round t is $\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i)$

This is what we need to decide in round t

$$Obj^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^{t} \Omega(f_i)$$
$$= \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t) + constant$$

Goal: find f_t to minimize this

- Consider square loss

$$Obj^{(t)} = \sum_{i=1}^{n} \left(y_i - (\hat{y}_i^{(t-1)} + f_t(x_i))\right)^2 + \Omega(f_t) + const$$
$$= \sum_{i=1}^{n} \left[2(\hat{y}_i^{(t-1)} - y_i) f_t(x_i) + f_t(x_i)^2\right] + \Omega(f_t) + co$$

残差

This is usually called residual from previous round

（3）.Taylor Expansion Approximation of Loss 泰勒近似展开

- Goal $Obj^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t) + constant$
  - Seems still complicated except for the case of square loss
- Take Taylor expansion of the objective
  - Recall $f(x + \Delta x) \simeq f(x) + f'(x)\Delta x + \frac{1}{2} f''(x)\Delta x^2$

保留二次项

  - Define $g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)}), \quad h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$

$$Obj^{(t)} \simeq \sum_{i=1}^{n} \left[l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)\right] + \Omega(f_t) + constant$$

- *If you are not comfortable with this, think of square loss*

$$g_i = \partial_{\hat{y}^{(t-1)}} (\hat{y}^{(t-1)} - y_i)^2 = 2(\hat{y}^{(t-1)} - y_i) \quad h_i = \partial^2_{\hat{y}^{(t-1)}} (y_i - \hat{y}^{(t-1)})^2 = 2$$

- Compare what we get to previous slide

把平方损失函数的一二次项带入原目标函数，你会发现与之前那张ppt的损失函数是一致的

（4）.Our New Goal 得到新的学习目标函数

- Objective, with constants removed

$$\sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)\right] + \Omega(f_t)$$

  - where $g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)}), \quad h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$

从这里就可以看出xgboost的特点了，目标函数保留了泰勒展开

关闭

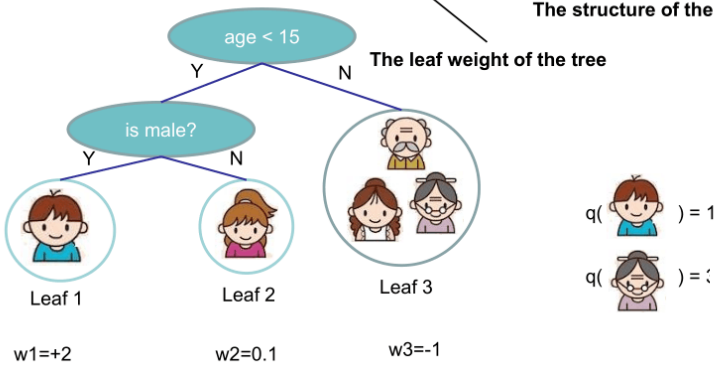
（5）.Refine the definition of tree 重新定义每棵树

• We define tree by a vector of scores in leafs, and a leaf index mapping function that maps an instance to a leaf

$$f_t(x) = w_{q(x)}, \quad w \in \mathbf{R}^T, q : \mathbf{R}^d \to \{1, 2, \cdots, T\}$$

The structure of the tree

The leaf weight of the tree



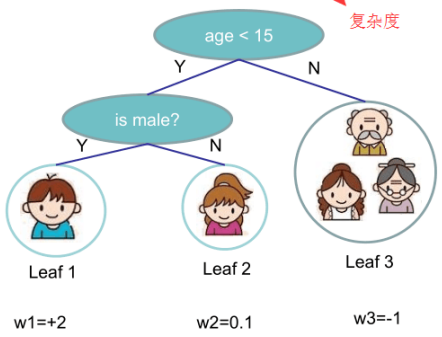
（6）.Define the Complexity of Tree 树的复杂度项

- Define complexity as (this is not the only possible definition)

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

Number of leaves 复杂度

L2 norm of leaf scores 平滑？？？



$$\Omega = \gamma 3 + \frac{1}{2}\lambda(4 + 0.01 + 1)$$

从图中可以看出，xgboost算法中对树的复杂度项增加了一个L2正则化项，针对每个叶结点的得分增加L2平滑，目的也是为了避免过拟合。

（7）.Revisit the Objectives

- Define the instance set in leaf j as $I_j = \{i|q(x_i) = j\}$
- Regroup the objective by each leaf 根据叶结点重新组合目标函数

$$\begin{aligned} Obj^{(t)} &\simeq \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t) \\ &= \sum_{i=1}^{n} \left[g_i w_{q(x_i)} + \frac{1}{2} h_i w_{q(x_i)}^2 \right] + \gamma T + \lambda \frac{1}{2} \sum_{j=1}^{T} w_j^2 \\ &= \sum_{j=1}^{T} \left[(\sum_{i \in I_j} g_i) w_j + \frac{1}{2} (\sum_{i \in I_j} h_i + \lambda) w_j^2 \right] + \gamma T \end{aligned}$$

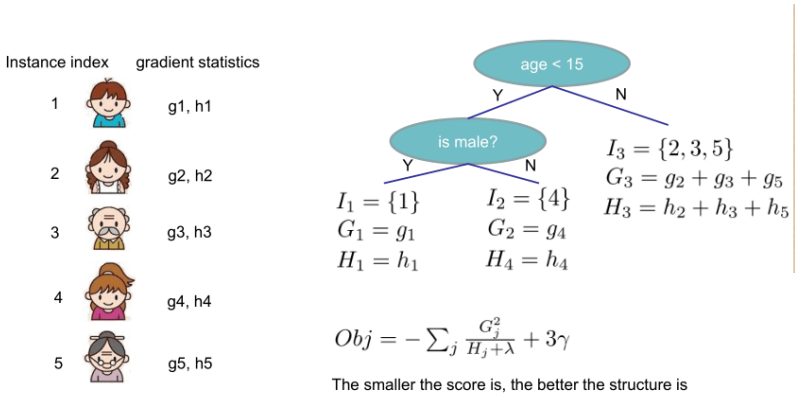
- This is sum of T independent quadratic functions

独立的二次函数的和

（8）.The Structure Score 这个score你可以理解成类似于信息增益的一个指标，在切分点查找算法中用

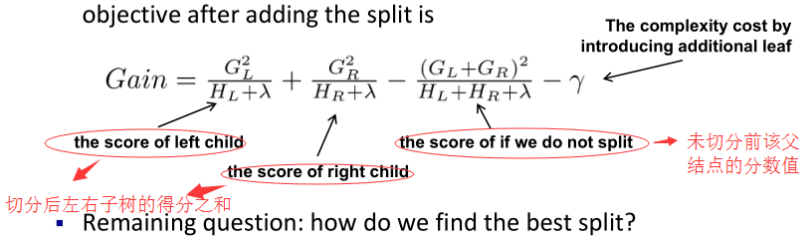
关闭

到。



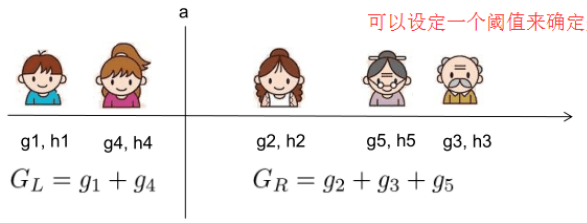
（9）切分点查找算法（贪心算法）

- In practice, we grow the tree greedily
  - Start from tree with depth 0
  - For each leaf node of the tree, try to add a split. The change o objective after adding the split is



  - Remaining question: how do we find the best split?

- What is the gain of a split rule $x_j < a$? Say x_j is age



- All we need is sum of g and h in each side, and calculate

$$Gain = \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{(G_L+G_R)^2}{H_L+H_R+\lambda} - \gamma$$

- Left to right linear scan over sorted instance is enough to decide the best split along the feature

扫描所有排序过的特征实例的得分，得出最优切分点

关闭

上图中G都是各自区域内的gi总和，此外，作者针对算法设计对特征进行了排序，有兴趣的可以阅读原始论文，这里不做详解。

（10）小结一下： Boosted Tree Algorithm

- Add a new tree in each iteration
- Beginning of each iteration, calculate

$$g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)}), \quad h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$$

一阶 二阶

- Use the statistics to greedily grow a tree $f_t(x)$

$$Obj = -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j+\lambda} + \gamma T$$

贪心算法寻找切分点，生产每一轮新的树

- Add $f_t(x)$ to the model $\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i)$
  - Usually, instead we do $y^{(t)} = y^{(t-1)} + \epsilon f_t(x_i)$

称之为：缩减因子
同样为了避

  - ϵ is called step-size or shrinkage, usually set around 0.1
  - This means we do not do full optimization in each step and reserve chance for future rounds, it helps prevent overfitti

二、xgboost特点（与gbdt对比）

说明一下：这部分内容参考了知乎上的一个问答—机器学习算法中GBDT和XGBOOST的区别有哪些？，答主是wepon大神，根据他的总结我自己做了一理解和补充。

1.传统GBDT以CART作为基分类器，xgboost还支持线性分类器，这个时候xgboost相当于带L1和L2正则化项的逻辑斯蒂回归（分类问题）或者线性回归（回归问题）。 —可以通过booster [default=gbtree]设置参数:gbtree: tree-based models/gblinear: linear models

2.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。顺便提一下，xgboost工具支持自定义代价函数，只要函数可一阶和二阶求导。 —对损失函数做了改进（泰勒展开，一阶信息g和二阶信息h,上一章节有做介绍）

3.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。

4.shrinkage and column subsampling —还是为了防止过拟合，论文2.3节有介绍，这里答主已概括的非常到位

关闭

> （1）shrinkage缩减类似于学习速率，在每一步tree boosting之后增加了一个参数n（权重），通过这种方式来减小每棵树的影响力，给后面的树提供空间去优化模型。
>
> （2）column subsampling列(特征)抽样，说是从随机森林那边学习来的，防止过拟合的效果比传统的行抽样还好（行抽样功能也有），并且有利于后面提到的并行化处理算法。

5.split finding algorithms(划分点查找算法)：—理解的还不够透彻，需要进一步学习

（1）exact greedy algorithm—贪心算法获取最优切分点

（2）approximate algorithm— 近似算法，提出了候选分割点概念，先通过直方图算法获得候选分割点的

分布情况，然后根据候选分割点将连续的特征信息映射到不同的buckets中，并统计汇总信息。详细见论文

3.3节

（3）Weighted Quantile Sketch—分布式加权直方图算法，论文3.4节

这里的算法（2）、（3）是为了解决数据无法一次载入内存或者在分布式情况下算法（1）效率低的问题，

以下引用的还是wepon大神的总结：

> 可并行的近似直方图算法。树节点在进行分裂时，我们需要计算每个特征的每个分割点对应的增益，即用贪心法枚举所有可能的分割点。当数据无法一次载入内存或者在分布式情况下，贪心算法效率就会变得很低，所以xgboost还提出了一种可并行的近似直方图算法，用于高效地生成候选的分割点。

6.对缺失值的处理。对于特征的值有缺失的样本，xgboost可以自动学习出它的分裂方向。

法，论文3.4节，Algorithm 3: Sparsity-aware Split Finding

7.Built-in Cross-Validation（内置交叉验证)

> XGBoost allows user to run a cross-validation at each iteration of the boosting process, and thus it is easy to get the exact optimum number of boosting iterations in a single run.
> This is unlike GBM where we have to run a grid-search and only a limited values can be tested.

8.continue on Existing Model（接着已有模型学习）

> User can start training an XGBoost model from its last iteration of previous run. This can be of significant advantage in certain specific applications.
> GBM implementation of sklearn also has this feature so they are even on this point.

9.High Flexibility（高灵活性）

> **XGBoost allow users to define custom optimization objectives and evaluation criteria.
> This adds a whole new dimension to the model and there is no limit to what we can do.**

10.并行化处理 —系统设计模块,块结构设计等

> xgboost工具支持并行。boosting不是一种串行的结构吗?怎么并行的？注意xgboost的并行不是tree粒度的并行，xgboost也是一次迭代完才能进行下一次迭代的（第t次迭代的代价函数里包含了前面t-1次迭代的预测值）。xgboost的并行是在特征粒度上的。我们知道，决策树的学习最耗时的一个步骤就是对特征的值进行排序（因为要确定最佳分割点），xgboost在训练之前，预先对数据进行了排序，然后保存为block结构，后面的迭代中重复地使用这个结构，大大减小计算量。这个block结构也使得并行成为了可能，在进行节点的分裂时，需要计算每个特征的增益，最终选增益最大的那个特征去做分裂，那么各个特征的增益计算就可以开多线程进行。

关闭

此外xgboost还设计了高速缓存压缩感知算法，这是系统设计模块的效率提升。

当梯度统计不适合于处理器高速缓存和高速缓存丢失时，会大大减慢切分点查找算法的速度。

（1）针对 exact greedy algorithm采用缓存感知预取算法

（2）针对 approximate algorithms选择合适的块大小

我觉得关于xgboost并行化设计仅仅从论文PPT博客上学习是远远不够的，有时间还要从代码层面去学习分布式 xgboost的设计理念。

三、xgboost参数详解

官方参数介绍看这里：

Parameters (official guide)

General Parameters（常规参数）

1.booster [default=gbtree]：选择基分类器，gbtree: tree-based models/gblinear: linear models

2.silent [default=0]:设置成1则没有运行信息输出，最好是设置为0.

3.nthread [default to maximum number of threads available if not set]：线程数

Booster Parameters（模型参数）

1.eta [default=0.3]:shrinkage参数，用于更新叶子节点权重时，乘以该系数，避免步长过大，

越可能无法收敛。把学习率 eta 设置的小一些，小学习率可以使得后面的学习更加仔细。

2.min_child_weight [default=1]:这个参数默认是 1，是每个叶子里面 h 的和至少是多少，对 衡

时的 0-1 分类而言，假设 h 在 0.01 附近，min_child_weight 为 1 意味着叶子节点中最少需要包含 100 个样本。这个参数非常影响结果，控制叶子节点中二阶导的和的最小值，该参数值越小，越容易 overfitting。

3.max_depth [default=6]: 每颗树的最大深度，树高越深，越容易过拟合。

4.max_leaf_nodes:最大叶结点数，与max_depth作用有点重合。

5.gamma [default=0]：后剪枝时，用于控制是否后剪枝的参数。

6.max_delta_step [default=0]：这个参数在更新步骤中起作用，如果取0表示没有约束，如果取正值则使得更新步骤更加保守。可以防止做太大的更新步子，使更新更加平缓。

7.subsample [default=1]：样本随机采样，较低的值使得算法更加保守，防止过拟合，但是太小的值也会造成欠拟合。

8.colsample_bytree [default=1]：列采样，对每棵树的生成用的特征进行列采样.一般设置为： 0.5-1

9.lambda [default=1]：控制模型复杂度的权重值的L2正则化项参数，参数越大，模型越不容易过拟合。

10.alpha [default=0]:控制模型复杂程度的权重值的 L1 正则项参数，参数值越大，模型越不容易过拟合。

11.scale_pos_weight [default=1]：如果取值大于0的话，在类别样本不平衡的情况下有助于快速收敛。

Learning Task Parameters（学习任务参数）

1.objective [default=reg:linear]：定义最小化损失函数类型，常用参数：

binary:logistic –logistic regression for binary classification, re 关闭

multi:softmax –multiclass classification using the softmax objective, returns predicted class (not probabilities)

you also need to set an additional num_class (number of classes) parameter defining the number of unique classes

multi:softprob –same as softmax, but returns predicted probability of each data point belonging to each class.

2.eval_metric [default according to objective]：

The metric to be used for validation data.

The default values are rmse for regression and error for classification.

Typical values are:

rmse – root mean square error

mae – mean absolute error

logloss – negative log-likelihood

error – Binary classification error rate (0.5 threshold)

merror – Multiclass classification error rate

mlogloss – Multiclass logloss

auc: Area under the curve

3.seed [default=0]：

The random number seed. 随机种子，用于产生可复现的结果

Can be used for generating reproducible results and also for parameter tuning.

注意: python sklearn style参数名会有所变化

eta –> learning_rate

lambda –> reg_lambda

alpha –> reg_alpha

四、实战

官方样例：

XGBoost Python API Reference (official guide)

XGBoost Demo Codes (xgboost GitHub repository)

xgboost参数设置代码示例：

```
# 划分数据集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.01, random_state=1729)
print(X_train.shape, X_test.shape)

#模型参数设置
xlf = xgb.XGBRegressor(max_depth=10,
                       learning_rate=0.1,
                       n_estimators=10,
                       silent=True,
                       objective='reg:linear',
                       nthread=-1,
                       gamma=0,
                       min_child_weight=1,
                       max_delta_step=0,
                       subsample=0.85,
                       colsample_bytree=0.7,
                       colsample_bylevel=1,
                       reg_alpha=0,
                       reg_lambda=1,
```

关闭

```
                scale_pos_weight=1,
                seed=1440,
                missing=None)

xlf.fit(X_train, y_train, eval_metric='rmse', verbose = True, eval_set = [(X_test, y_test)],early_stopp

# 计算 auc 分数、预测
preds = xlf.predict(X_test)
```

实战待续...

顶 0 踩 0

- 上一篇 BM算法 Boyer-Moore高质量实现代码详解与算法详解
- 下一篇 xgboost入门与实战（实战调参篇）

相关文章推荐

- XGBoost：在Python中使用XGBoost
- 腾讯云容器服务架构实现介绍--董晓杰
- xgboost的使用简析
- 容器技术在58同城的实践--姚远
- xgboost入门与实战（实战调参篇）
- Tensorflow项目实战-文本分类
- xgboost入门与实战（原理篇）
- MySQL深入浅出
- 揭秘Kaggle神器xgboost
- Python可以这样学（第三季：多线程与多进程编程...
- xgboost使用案例二
- 华为工程师，带你实战C++
- Kaggle 神器：XGBoost 从基础到实战
- xgboost快速入门
- Kaggle入门实例-预测房价
- xgboost原理

查看评论

暂无评论

关闭

您还没有登录,请[登录]或[注册]

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

关闭