CSDN
不 止 于 代 码

3.8 XGBoost使用指南 (二)

CSDN学院

2017年11月

XGBoost的使用

CSDN
不 止 于 代 码

- 直接调用XGBoost
  - import xgboost as xgb
- 与scikit-learn一起使用
  - from xgboost import XGBClassifier

与scikit-learn结合使用

CSDN
不 止 于 代 码

- XGBoost提供一个wrapper类，允许模型可以和scikit-learn框架中其他分类器或回归器一样对待
- XGBoost中分类模型为XGBClassifier
  - 回归：
  - 排序：

- 1. 构造学习器实例
  - 模型参数在构造时传递
- 2. 模型训练：fit/GridSearchCV
- 3. 预测

XGBoost接口

CSDN
不 止 于 代 码

http://xgboost.readthedocs.io/en/latest/python/python_api.html#module-xgboost.sklearn

- xgboost.**XGBClassifier**(*max_depth=3, learning_rate=0.1, n_estimators=100, silent=True, objective='binary:logistic', nthread=-1, gamma=0, min_child_weight=1, max_delta_step=0, subsample=1, colsample_bytree=1, colsample_bylevel=1, reg_alpha=0, reg_lambda=1, scale_pos_weight=1, base_score=0.5, random_state=0, seed=None, missing=None , **kwargs*)
- fit(*X, y, sample_weight=None, eval_set=None, eval_metric=None, early_stopping_rounds=None, verbose=True, xgb_model=None*)

参数	说明
max_depth	树的最大深度。树越深通常模型越复杂，更容易过拟合
learning_rate	学习率或收缩因子。学习率和迭代次数 / 弱分类器数目n_estimators相关。 缺省：0.1 （与直接调用xgboost的eta参数含义相同）
n_estimators	弱分类器数目. 缺省:100
slient	参数值为1时，静默模式开启，不输出任何信息
objective	待优化的目标函数，常用值有： binary:logistic 二分类的逻辑回归，返回预测的概率 multi:softmax 使用softmax的多分类器，返回预测的类别(不是概率)。 multi:softprob 和multi:softmax参数一样，但是返回的是每个数据属于各个类别的概率。支持用户自定义目标函数
nthread	用来进行多线程控制。 如果你希望使用CPU全部的核，那就不用缺省值-1，算法会自动检测它。
booster	选择每次迭代的模型，有两种选择： gbtree：基于树的模型，为缺省值。 gbliner：线性模型
gamma	节点分裂所需的最小损失函数下降值
min_child_weight	叶子结点需要的最小样本权重（hessian）和
max_delta_step	允许的树的最大权重

参数	说明
subsample	构造每棵树的所用样本比例（样本采样比例），同GBM
colsample_bytree	构造每棵树的所用特征比例
colsample_bylevel	树在每层每个分裂的所用特征比例
reg_alpha	L1 正则的惩罚系数
reg_lambda	L2 正则的惩罚系数
scale_pos_weight	正负样本的平衡，通常用于不均衡数据
base_score	每个样本的初始估计，全局偏差
random_state	随机种子
seed	随机种子
missing	当数据缺失时的填补值。缺省为np.nan
kwargs	XGBoost Booster的Keyword参数

参数类别

CSDN
不 止 于 代 码

- 通用参数：这部分参数通常我们不需要调整，默认值就好
  - booster、 silent、 *nthread*
- 学习目标参数：与任务有关，定下来后通常也不需要调整
  - *objective*
- booster参数：弱学习器相关参数，需要仔细调整，会影响模型性能

通用参数

CSDN
不 止 于 代 码

- booster：弱学习器类型
  - 可选gbtree（树模型）或gbliner（线性模型）
  - 默认为gbtree（树模型为非线性模型，能处理更复杂的任务）
- silent：是否开启静默模式
  - 1：静默模式开启，不输出任何信息
  - 默认值为0：输出一些中间信息，以助于我们了解模型的状态
- nthread：线程数
  - 默认值为-1，表示使用系统所有CPU核

学习目标参数

CSDN
不 止 于 代 码

- objective: 损失函数
  - 支持分类 / 回归 / 排序
- eval_metric：评价函数
  - fit函数的参数
- seed：随机数的种子
  - 默认为0
  - 设置seed可复现随机数据的结果，也可以用于调整参数

XGBoost支持的目标函数

CSDN
不 止 于 代 码

- Objective： 定义学习任务及相应的学习目标，可选的目标函数如下：
  - “reg:linear” –线性回归。
  - “reg:logistic” –逻辑回归。
  - “binary:logistic” –二分类的逻辑回归问题，输出为概率。
  - “binary:logitraw” –二分类的逻辑回归问题，输出的结果为$\mathbf{w}^T\mathbf{x}$。
  - “count:poisson” –计数问题的poisson回归，输出结果为poisson分布。
  - “multi:softmax” –让XGBoost采用softmax目标函数处理多分类问题
  - “multi:softprob” –和softmax一样，但是输出的是ndata * nclass的向量，可以将该向量reshape成ndata行nclass列的矩阵。没行数据表示样本所属于每个类别的概率。
  - “rank:pairwise” –set XGBoost to do ranking task by minimizing the pairwise loss

XGBoost自定义目标函数

CSDN 不止于代码

- 在GBDT训练过程，当每步训练得到一棵树，要调用目标函数得到其梯度作为下一棵树拟合的目标
- XGBoost在调用obj函数时会传入两个参数：preds和dtrain
  - preds为当前模型完成训练时，所有训练数据的预测值
  - dtrain为训练集，可以通过dtrain.get_label()获取训练样本的label
  - 同时XGBoost规定目标函数需返回当前preds基于训练label的一阶和二阶梯度
- # user define objective function, given prediction, return gradient and second order gradient
- # this is log likelihood loss　　自定义目标函数
- def logregobj(preds, dtrain):　　参数为预测值preds和训练集dtrain（**X**，**y**）
  - labels = dtrain.get_label()
  - preds = 1.0 / (1.0 + np.exp(-preds))
  - grad = preds - labels　　#梯度
  - hess = preds * (1.0-preds)　　#2阶导数
  - return grad, hess
- 参数：obj= 'logregobj'

https://github.com/dmlc/xgboost/blob/master/demo/guide-python/custom_objective.py

XGBoost支持的评价函数

CSDN
不 止 于 代 码

- **'eval_metric' 评估指标:**
  - "rmse": root mean square error
  - "logloss": negative log-likelihood
  - "error": Binary classification error rate. It is calculated as #(wrong cases)/#(all cases).
  - "merror": Multiclass classification error rate. It is calculated as #(wrong cases)/#(all cases).
  - "mlogloss": Multiclass logloss
  - "auc": Area under the curve for ranking evaluation.
  - "ndcg":Normalized Discounted Cumulative Gain
  - "map":Mean average precision
  - "ndcg@n","map@n": n can be assigned as an integer to cut off the top positions in the lists for evaluation.
  - "ndcg-","map-","ndcg@n-","map@n-": In XGBoost, NDCG and MAP will evaluate the score of a list wit hout any positive samples as 1. By adding "-" in the evaluation metric XGBoost will evaluate these score a s 0 to be consistent under some conditions.

XGBoost自定义评价函数

- # user defined evaluation function, return a pair metric_name, result
- # NOTE: when you do customized loss function, the default prediction value is margin
- # this may make builtin evaluation metric not function properly
- # for example, we are doing logistic loss, the prediction is score before logistic transformation
- # the builtin evaluation error assumes input is after logistic transformation
- # Take this in mind when you use the customization, and maybe you need write customized evaluation function

- def evalerror (preds, dtrain):
  - labels = dtrain.get_label()
  - return 'error', float(sum(labels != (preds > 0.0))) / len(labels)

- 参数： feval =‘evalerror’

http://blog.csdn.net/lujiandong1/article/details/52791117

booster参数

CSDN
不 止 于 代 码

- 弱学习器的参数，尽管有两种booster可供选择，这里只介绍gbtree
- 1. learning_rate : 收缩步长 vs. n_estimators：树的数目
  - 较小的学习率通常意味着更多弱分学习器
  - 通常建议学习率较小（ $\eta < 0.1$ ），弱学习器数目n_estimators大

$$f_m(\mathbf{x}_i) = f_{m-1}(\mathbf{x}_i) + \eta\phi_m(\mathbf{x}_i)$$

  - 可以设置较小的学习率，然后用交叉验证确定n_estimators

booster参数

CSDN
不 止 于 代 码

- 2. 行（*subsample*）列（*colsample_bytree*、*colsample_bylevel*）下采样比例
  - 默认值均为1，即不进行下采样，使用所有数据
  - 随机下采样通常比用全部数据的确定性过程效果更好，速度更快
  - 建议值：0.3 - 0.8

booster参数

- 3. 树的最大深度： max_depth
  - max_depth越大，模型越复杂，会学到更具体更局部的样本
  - 需要使用交叉验证进行调优，默认值为6，建议3-10

- 4. min_child_weight ：孩子节点中最小的样本权重和
  - 如果一个叶子节点的样本权重和小于min_child_weight则分裂过程结束
  - min_child_weight：minimum sum of instance weight (hessian) needed in a child

Kaggle竞赛优胜者的建议

CSDN
不 止 于 代 码

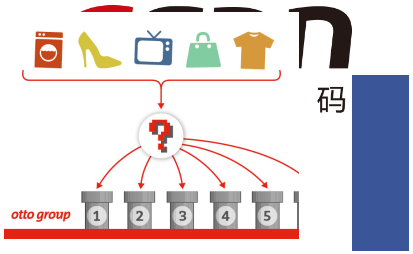
- Tong He（XGBoost R语言版本开发者）： 三个最重要的参数为：树的**数目**、树的**深度**和**学习率**。建议参数调整策略为：
  - 采用默认参数配置试试
  - 如果系统过拟合了，降低学习率
  - 如果系统欠拟合，加大学习率

Kaggle竞赛优胜者的建议（cont.）

CSDN
不 止 于 代 码

- Owen Zhang（常使用XGBoost）建议：
  - n_estimators和learning_rate：固定n_estimators为100（数目不大，因为树的深度较大，每棵树比较复杂），然后调整learning_rate
  - 树的深度max_depth：从6开始，然后逐步加大
  - min_child_weight：$1/sqrt\sqrt{\text{rare\_events}}$，其中rare_events 为稀有事件的数目
  - 列采样colsample_bytree / colsample_bylevel：在[0.3, 0.5]之间进行网格搜索
  - 行采样subsample：固定为1
  - gamma: 固定为0.0

Kaggle竞赛案例分析：
Otto Group Product Classification Challenge



- 竞赛官网：https://www.kaggle.com/c/otto-group-product-classification-challenge
- 电商商品分类：
  - Target：共9个商品类别
  - 93个特征：整数型特征

参数调优的一般方法

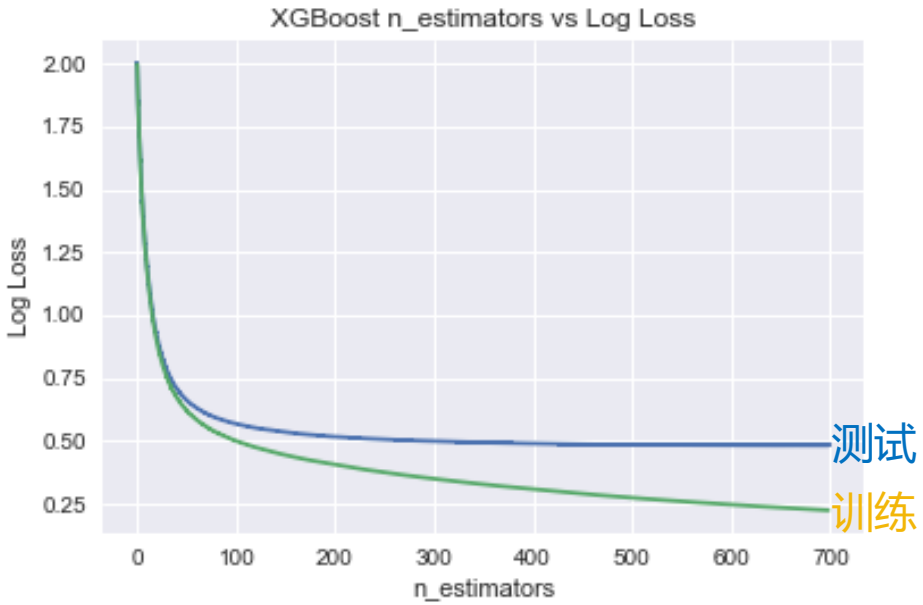
CSDN
不 止 于 代 码

- 1. 选择较高的学习率(learning rate)，并选择对应于此学习率的理想的树的数量
  - 学习率以工具包默认值为0.1。
  - XGBoost直接引用函数“cv”可以在每一次迭代中使用交叉验证，并返回理想的树数量（因为交叉验证很慢，所以可以import两种XGBoost：直接引用xgboost（用“cv”函数调整树的数目）和XGBClassifier —xgboost的sklearn包（用GridSearchCV调整其他参数））。
- 2. 对于给定的学习率和树数量，进行树参数调优(max_depth, min_child_weight, gamma, subsample, colsample_bytree, colsample_bylevel)
- 3. xgboost的正则化参数(lambda, alpha)的调优
- 4. 降低学习率，确定树的数目参数

可参考：http://blog.csdn.net/u010657489/article/details/51952785

CSDN
不 止 于 代 码

- 1. 采用缺省参数,此时learning_rate =0.1（较大）， 观察n_estimators的合适范围
  - 参数设为1000，earlystop = 50
  - cv函数在n_estimators =699时停止
  - cv测试误差为0.482744

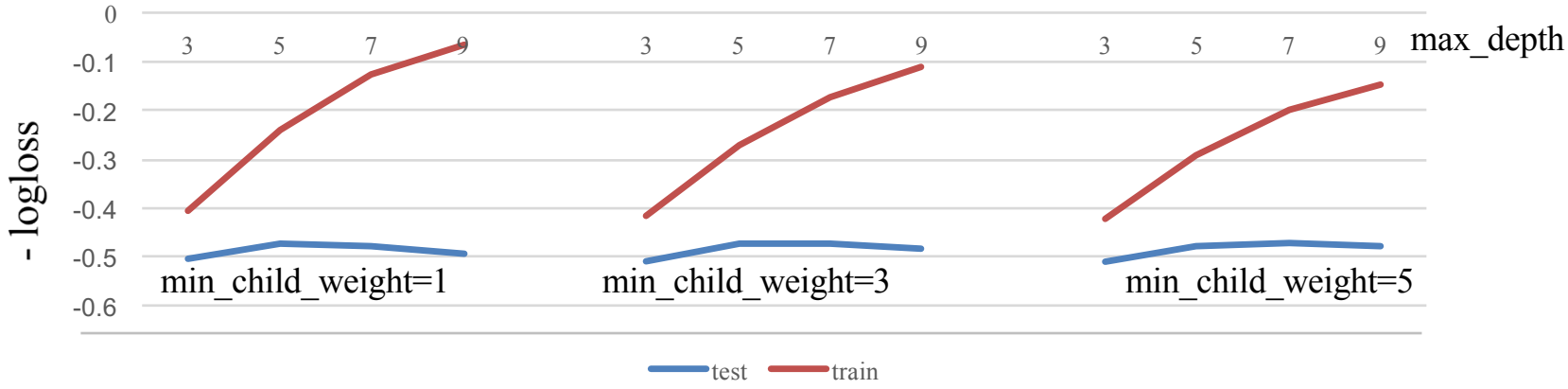



```
max_depth=5,
min_child_weight=1,
gamma=0,
subsample=0.3,
colsample_bytree=0.8,
colsample_bylevel=0.7,
```

CSDN
不 止 于 代 码

- 2.1 max_depth 和 min_child_weight 参数调整
  - 这两个参数对结果影响很大
  - 我们先大范围地粗调参数（步长为2），再小范围微调
  - max_depth = range(3,10,2)
  - min_child_weight = range(1,6,2)

最小交叉验证测试误差： -0.471956
max_depth: 7
min_child_weight: 5



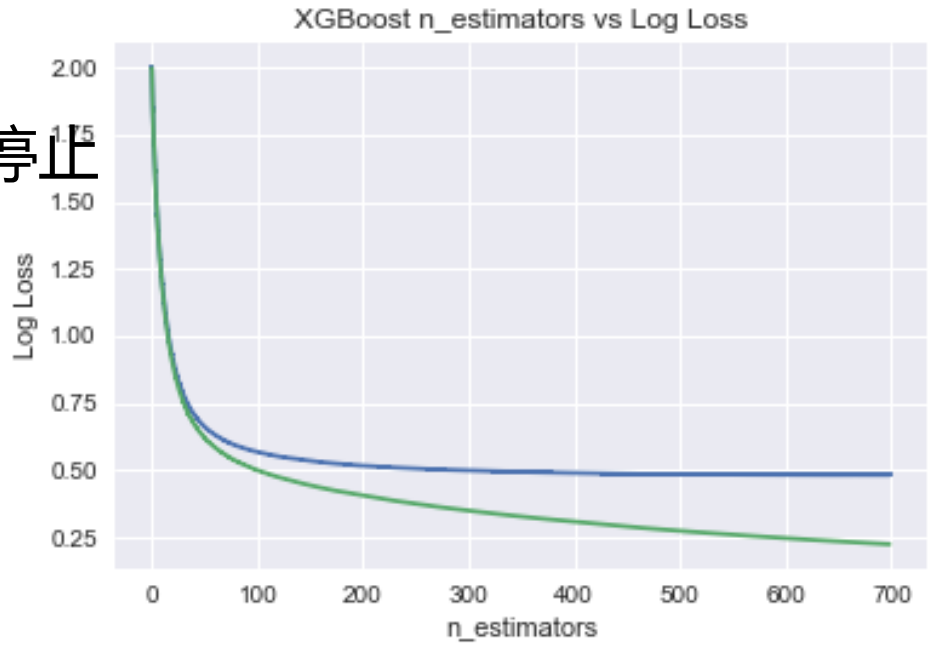
CSDN
不 止 于 代 码

- 2.2 max_depth 和 min_child_weight 参数微调
  - 在max_depth=7和min_child_weight=5周围微调
  - max_depth = [6,7,8]
  - min_child_weight = [4,5,6]

  - 最小交叉验证测试误差： -0.471302
  - max_depth: 6
  - min_child_weight: 4

CSDN
不 止 于 代 码

- 2.3 调整max_depth=6 和 min_child_weight=4后，再次调整 n_estimators
  - 参数设为1000，earlystop = 50
  - cv函数在n_estimators =645时停止

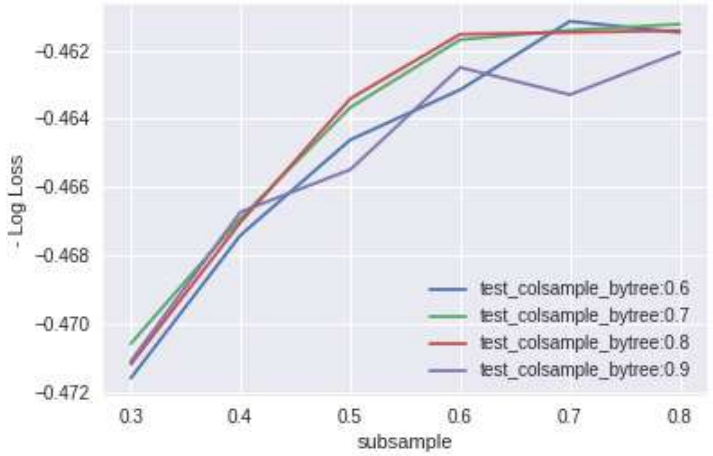


- 3. gamma参数调整
  - 通常缺省值（0）表现还不错，如计算资源允许，可以调整（略）

CSDN
不 止 于 代 码

- 4. 行列采样参数： subsample和colsample_bytree
  - 这两个参数可分别对样本和特征进行采样，从而增加模型的鲁棒性
  - 现在[0.3-1.0]之间调整，粗调时步长0.1
  - 微调时步长为0.05（略）

Best: -0.461137
subsample: 0.7,
colsample_bytree: 0.6

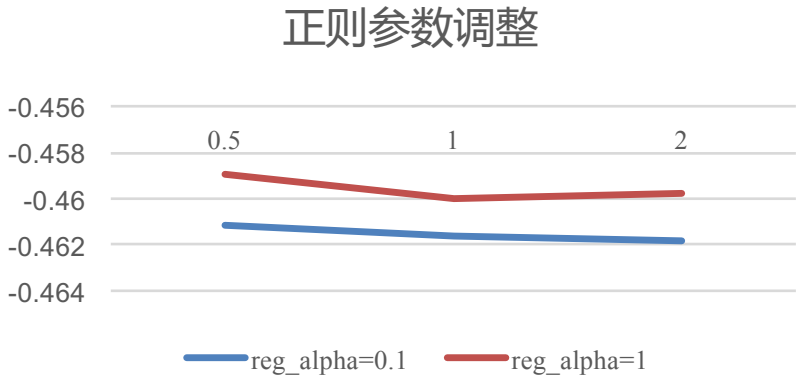


  - 注： colsample_bylevel通常0.7-0.8可以得到较好的结果。如计算资源允许，也可以进一步调整

$$\Omega(\phi(\mathbf{x})) = \gamma T + \frac{1}{2}\lambda \sum_{t=1}^{T} w_t^2$$

CSDN 不 止 于 代 码

- 5. 正则参数调整： reg_alpha（L2）和reg_lambda(L0)
  - reg_alpha = [0.1, 1]　# L1, default = 0
  - reg_lambda = [0.5, 1, 2]　　# L2, default = 1
  - Best: -0.458950 using {'reg_alpha': 1, 'reg_lambda': 0.5)



reg_lambda似乎越小越好（叶子结点的分数）
reg_alpha似乎越大越好（叶子结点数目）

如计算资源允许，可进一步增大reg_alpha，减小reg_lambda

CSDN
不 止 于 代 码

- 6. 降低学习率，调整树的数目
  - 调用xgboost的cv函数
    - 0.1:　669棵树收敛，cv测试误差为0.469456
    - 0.01：2000棵树还没有收敛，cv测试误差为0.501644
    - 0.05:　1386棵树收敛，cv测试误差为0.465813

```
xgb6 = XGBClassifier( learning_rate =0.05,
                      n_estimators=2000,
                      max_depth=6,
                      min_child_weight=4,
                      subsample=0.7,
                      colsample_bytree=0.6,
                      colsample_bylevel=0.7,
                      reg_alpha = 1,
                      reg_lambda = 0.5,
                      objective= 'multi:softprob',       seed=3)
```

XGBoost总结

CSDN
不 止 于 代 码

- XGBoost是一个用于监督学习的非参数模型
  - 目标函数（损失函数、正则项）
  - 参数（树的每个分支分裂特征及阈值）
  - 优化：梯度下降

- 参数优化
  - 决定模型复杂度的重要参数：learning_rate, n_estimators, max_depth, min_child_weight, gamma, reg_alpha, reg_lamba
  - 随机采样参数也影响模型的推广性： subsample, colsample_bytree, colsample_bylevel

CSDN
不 止 于 代 码

THANK YOU

AI100