个人资料

余音丶未散

关注 发私信

访问： 64660次
积分： 1726
等级： BLOG 4
排名： 千里之外

原创： 108篇 转载： 12篇
译文： 0篇 评论： 25条

github地址

https://github.com/lytforgood

文章搜索

文章分类

机器学习 (31)
R (41)
算法 (11)
天池比赛 (6)
机器学习算法MapReduce版 (3)
Hadoop相关 (25)
Spark (15)
linux (22)
Python (8)
爬虫 (6)
深入学习JVM笔记 (3)
Java (12)
神经网络 (11)
面试笔记 (4)

文章存档

xgboost使用调参

2016-12-20 15:14 664人阅读 评论

分类： 机器学习（30）

目录(?) [+]

github：https://github.com/dmlc/xgboost

论文参考：http://www.kaggle.com/blobs/download/forum-message-attachment-files/4087/xgboost-paper.pdf

基本思路及优点

http://blog.csdn.net/q383700092/article/details/60954996

参考http://dataunion.org/15787.html

http://blog.csdn.net/china1000/article/details/51106856

在有监督学习中，我们通常会构造一个目标函数和一个预测函数，使用训练样本对目标函数最小化学习到相关的参数，然后用预

参数来对未知的样本进行分类的标注或者数值的预测。

1. Boosting Tree构造树来拟合残差，而Xgboost引入了二阶导来进行求解，并且引入了节点的数目、参数的L2正则来评估模型

预测函数与目标函数。

2. 在分裂点选择的时候也以目标函数最小化为目标。

优点：

1. 显示的把树模型复杂度作为正则项加到优化目标中。

2. 公式推导中用到了二阶导数，用了二阶泰勒展开。（GBDT用牛顿法貌似也是二阶信息）

3. 实现了分裂点寻找近似**算法**。

4. 利用了特征的稀疏性。

5. 数据事先排序并且以block形式存储，有利于并行计算。

6. 基于分布式通信框架rabit，可以运行在MPI和yarn上。（最新已经不基于rabit了）

7. 实现做了面向体系结构的优化，针对cache和内存做了性能优化。

原理推导及与GBDT区别

http://blog.csdn.net/q383700092/article/details/60954996

参考http://dataunion.org/15787.html

https://www.zhihu.com/question/41354392

2017年03月 (5)
2017年02月 (4)
2017年01月 (5)
2016年12月 (9)
2016年11月 (5)

展开

阅读排行

朴素贝叶斯之MapReduce版 (6422)
R参考卡片 (4752)
阿里音乐流行趋势预测大赛之... (2486)
R可视化绘图三-recharts (2273)
机器学习面试问题汇总 (1684)
Python入门基本语法 (1293)
利用hive完成阿里天池大数据... (1106)
R语言-data.table包使用(方便... (1065)
阿里音乐流行趋势预测大赛-... (1064)
R-k折交叉验证 (1021)

评论排行

R可视化绘图三-recharts (12)
机器学习面试编程题汇总 (7)
R参考卡片 (2)
机器学习与R之朴素贝叶斯分... (2)
GBDT 原理与使用 (1)
机器学习与R笔记之线性回归 (1)
Word2vec原理与应用 (0)
矢量量化神经网络,自组织竞争... (0)
广义回归神经网络GRNN ,竞... (0)
线性神经网络与非线性神经网... (0)

推荐文章

* 云计算的那些事儿之计算虚拟化
* 微服务--分布式事务的实现方法及替代方案
* 你应该知道的 Android 数据库更新策略
* HDFS副本放置节点选择的优化
* CSDN日报20170416 ——《为什么程序员话少钱多死得早？》
* 凡人视角C++之string(上)

最新评论

机器学习面试编程题汇总
余音、未散 ：@u013855234:思路应该没问题，通过率0.0%不知道是不是说该题的通过率

机器学习面试编程题汇总
u013855234 ：按照你的思路，提交之后，怎么运行都是0.0%,大神怎么办

机器学习面试编程题汇总
u013855234 ：我按照你的解法，怎么运行通过率都是0.0%，大神怎么办

机器学习面试编程题汇总
余音、未散 ：@a7760764:就是根据正态分布 随机生成一个数 具体含义不用太深究 正态分布生成数据百度有很多...

机器学习面试编程题汇总
a7760764 ：楼主，请问normalRandom()这个函数里面的 double c0 = 2.515517, c...

机器学习面试编程题汇总
a7760764 ：楼主，请问normalRandom()这个函数里面的 double c0 = 2.515517, c...

机器学习面试编程题汇总
a7760764 ：楼主，请问x,y 怎么求出来的，根据u，s么？

GBDT 原理与使用
husthust201 ：最后两行程序的代码是不是写错了？

R可视化绘图三-recharts

参数说明

参考http://blog.csdn.net/han_xiaoyang/article/details/52665396

参数

booster：默认 gbtree效果好 (linear booster很少用到)

gbtree：基于树的模型

gbliner：线性模型

silent[默认0]

nthread[默认值为最大可能的线程数]

eta[默认0.3] 学习率 典型值为0.01-0.2

min_child_weight[默认 1] 决定最小叶子节点样本权重和 值较大，避免过拟合 值过高，会导致欠拟合

max_depth[默认6]

gamma[默认0] 指定了节点分裂所需的最小损失函数下降值。 这个参数的值越大，算法越保守

subsample[默认1] 对于每棵树，随机采样的比例 减小，算法保守，避免过拟合。值设置得过小，它会导致欠拟合 典型值：0.5

colsample_bytree[默认1] 每棵随机采样的列数的占比

colsample_bylevel[默认1] 树的每一级的每一次分裂，对列数的采样的占比

lambda[默认1] 权重的L2正则化项

alpha[默认1] 权重的L1正则化项

scale_pos_weight[默认1] 在各类别样本十分不平衡时，参数设定为一个正值，可以使算法更快收敛

objective[默认reg:linear] 最小化的损失函数

binary:logistic 二分类的逻辑回归，返回预测的概率(不是类别)。 multi:softmax 使用softmax的多分类器，返回预测的类别(不

在这种情况下，你还需要多设一个参数：num_class(类别数目)。 multi:softprob 和multi:softmax参数一样，但是返回的是每

率。

eval_metric[默认值取决于objective参数的取值]

对于回归问题，默认值是rmse，对于分类问题，默认值是error。 典型值有：

rmse 均方根误差 mae 平均绝对误差 logloss 负对数似然函数值

error 二分类错误率 merror 多分类错误率 mlogloss 多分类logloss损失函数 auc 曲线下面积

seed(默认0) 随机数的种子 设置它可以复现随机数据的结果

sklearn包，XGBClassifier会改变的函数名

eta ->learning_rate

lambda->reg_lambda

alpha->reg_alpha

常用调整参数：

第一步：确定学习速率和tree_based 参数调优的估计器数目

树的最大深度一般3-10

max_depth = 5

节点分裂所需的最小损失函数下降值0.1到0.2

gamma = 0

采样

subsample= 0.8,

colsample_bytree = 0.8

比较小的值，适用极不平衡的分类问题

min_child_weight = 1

余音丶未散 ：@zhangxiaozhu123:参考新更新文档，已经改变用法http://blog.csdn.ne...

R可视化绘图三-recharts
zhangxiaozhu123 ：您好，下载echartR.R的链接已经失效，应该怎么才能运用echartr函数呢？现在无法下载到本地...

类别十分不平衡

scale_pos_weight = 1

```
from xgboost import XGBClassifier
xgb1 = XGBClassifier(
 learning_rate =0.1,
 n_estimators=1000,
 max_depth=5,
 min_child_weight=1,
 gamma=0,
 subsample=0.8,
 colsample_bytree=0.8,
 objective= 'binary:logistic',
 nthread=4,
 scale_pos_weight=1,
 seed=27)
```

第二步： max_depth 和 min_weight 参数调优

grid_search参考

http://scikit-learn.org/stable/modules/generated/sklearn.grid_search.GridSearchCV.html

http://blog.csdn.net/abcjennifer/article/details/23884761

网格搜索scoring=’ roc_auc’ 只支持二分类，多分类需要修改scoring(默认支持多分类)

```
param_test1 = {
 'max_depth':range(3,10,2),
 'min_child_weight':range(1,6,2)
}
#param_test2 = {
 'max_depth':[4,5,6],
 'min_child_weight':[4,5,6]
}
from sklearn import svm, grid_search, datasets
from sklearn import grid_search
gsearch1 = grid_search.GridSearchCV(
estimator = XGBClassifier(
learning_rate =0.1,
n_estimators=140, max_depth=5,
min_child_weight=1,
gamma=0,
subsample=0.8,
colsample_bytree=0.8,
objective= 'binary:logistic',
nthread=4,
scale_pos_weight=1,
seed=27),
param_grid = param_test1,
scoring='roc_auc',
n_jobs=4,
iid=False,
cv=5)
gsearch1.fit(train[predictors],train[target])
gsearch1.grid_scores_, gsearch1.best_params_,gsearch1.best_score_
#网格搜索scoring='roc_auc'只支持二分类，多分类需要修改scoring(默认支持多分类)
```

第三步：gamma参数调优

```
param_test3 = {
 'gamma':[i/10.0 for i in range(0,5)]
}
gsearch3 = GridSearchCV(
estimator = XGBClassifier(
learning_rate =0.1,
n_estimators=140,
max_depth=4,
min_child_weight=6,
gamma=0,
subsample=0.8,
colsample_bytree=0.8,
objective= 'binary:logistic',
nthread=4,
scale_pos_weight=1,
```

```
seed=27),
param_grid = param_test3,
scoring='roc_auc',
n_jobs=4,
iid=False,
cv=5)
gsearch3.fit(train[predictors],train[target])
gsearch3.grid_scores_, gsearch3.best_params_, gsearch3.best_score_
```

第四步：调整subsample 和 colsample_bytree 参数

```
#取0.6,0.7,0.8,0.9作为起始值
param_test4 = {
 'subsample':[i/10.0 for i in range(6,10)],
 'colsample_bytree':[i/10.0 for i in range(6,10)]
}

gsearch4 = GridSearchCV(
estimator = XGBClassifier(
learning_rate =0.1,
n_estimators=177,
max_depth=3,
min_child_weight=4,
gamma=0.1,
subsample=0.8,
colsample_bytree=0.8,
objective= 'binary:logistic',
nthread=4,
scale_pos_weight=1,
seed=27),
param_grid = param_test4,
scoring='roc_auc',
n_jobs=4,
iid=False,
cv=5)
gsearch4.fit(train[predictors],train[target])
gsearch4.grid_scores_, gsearch4.best_params_, gsearch4.best_score_
```

第五步：正则化参数调优

```
param_test6 = {
 'reg_alpha':[1e-5, 1e-2, 0.1, 1, 100]
}
gsearch6 = GridSearchCV(
estimator = XGBClassifier(
learning_rate =0.1,
n_estimators=177,
max_depth=4,
min_child_weight=6,
gamma=0.1,
subsample=0.8,
colsample_bytree=0.8,
objective= 'binary:logistic',
nthread=4,
scale_pos_weight=1,
seed=27),
param_grid = param_test6,
scoring='roc_auc',
n_jobs=4,
iid=False,
cv=5)
gsearch6.fit(train[predictors],train[target])
gsearch6.grid_scores_, gsearch6.best_params_, gsearch6.best_score_
```

第六步：降低学习速率

```
xgb4 = XGBClassifier(
 learning_rate =0.01,
 n_estimators=5000,
 max_depth=4,
 min_child_weight=6,
 gamma=0,
 subsample=0.8,
 colsample_bytree=0.8,
 reg_alpha=0.005,
 objective= 'binary:logistic',
```

```
 nthread=4,
 scale_pos_weight=1,
 seed=27)
modelfit(xgb4, train, predictors)
```

python示例

```
import xgboost as xgb
import pandas as pd
#获取数据
from sklearn import cross_validation
from sklearn.datasets import load_iris
iris = load_iris()
#切分数据集
X_train, X_test, y_train, y_test = cross_validation.train_test_split(iris.data, iris.target, test_size=
#设置参数
m_class = xgb.XGBClassifier(
 learning_rate =0.1,
 n_estimators=1000,
 max_depth=5,
 gamma=0,
 subsample=0.8,
 colsample_bytree=0.8,
 objective= 'binary:logistic',
 nthread=4,
 seed=27)
#训练
m_class.fit(X_train, y_train)
test_21 = m_class.predict(X_test)
print "Accuracy : %.2f" % metrics.accuracy_score(y_test, test_21)
#预测概率
#test_2 = m_class.predict_proba(X_test)
#查看AUC评价标准
from sklearn import metrics
print "Accuracy : %.2f" % metrics.accuracy_score(y_test, test_21)
##必须二分类才能计算
##print "AUC Score (Train): %f" % metrics.roc_auc_score(y_test, test_2)
#查看重要程度
feat_imp = pd.Series(m_class.booster().get_fscore()).sort_values(ascending=False)
feat_imp.plot(kind='bar', title='Feature Importances')
import matplotlib.pyplot as plt
plt.show()
#回归
#m_regress = xgb.XGBRegressor(n_estimators=1000,seed=0)
#m_regress.fit(X_train, y_train)
#test_1 = m_regress.predict(X_test)
```

整理

xgb原始

```
from sklearn.model_selection import train_test_split
from sklearn import metrics
from  sklearn.datasets  import  make_hastie_10_2
import xgboost as xgb
#记录程序运行时间
import time
start_time = time.time()
X, y = make_hastie_10_2(random_state=0)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0)##test_size测试
#xgb矩阵赋值
xgb_train = xgb.DMatrix(X_train, label=y_train)
xgb_test = xgb.DMatrix(X_test,label=y_test)
##参数
params={
'booster':'gbtree',
'silent':1 ,#设置成1则没有运行信息输出，最好是设置为0.
#'nthread':7,# cpu 线程数 默认最大
'eta': 0.007, # 如同学习率
'min_child_weight':3,
# 这个参数默认是 1，是每个叶子里面 h 的和至少是多少，对正负样本不均衡时的 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。
'max_depth':6, # 构建树的深度，越大越容易过拟合
'gamma':0.1,  # 树的叶子节点上作进一步分区所需的最小损失减少,越大越保守，一般0.1、0.2这样子。
```

```
'subsample':0.7, # 随机采样训练样本
'colsample_bytree':0.7, # 生成树时进行的列采样
'lambda':2,  # 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。
#'alpha':0, # L1 正则项参数
#'scale_pos_weight':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。
#'objective': 'multi:softmax', #多分类的问题
#'num_class':10, # 类别数，多分类与 multisoftmax 并用
'seed':1000, #随机种子
#'eval_metric': 'auc'
}
plst = list(params.items())
num_rounds = 100 # 迭代次数
watchlist = [(xgb_train, 'train'),(xgb_test, 'val')]

#训练模型并保存
# early_stopping_rounds 当设置的迭代次数较大时，early_stopping_rounds 可在一定的迭代次数内准确率没有提升
model = xgb.train(plst, xgb_train, num_rounds, watchlist,early_stopping_rounds=100,pred_margin=1)
#model.save_model('./model/xgb.model') # 用于存储训练出的模型
print "best best_ntree_limit",model.best_ntree_limit
y_pred = model.predict(xgb_test,ntree_limit=model.best_ntree_limit)
print ('error=%f' % (  sum(1 for i in range(len(y_pred)) if int(y_pred[i]>0.5)!=y_test[i]) /float(len(y
#输出
cost_time = time.time()-start_time
print             cess!",'\n',"cost time:",cost_time,"(s)......"
```

xgb使用sklearn接口(推荐)

官方

会改变的函数名是：

eta -> learning_rate

lambda -> reg_lambda

alpha -> reg_alpha

```
from sklearn.model_selection import train_test_split
from sklearn import metrics
from  sklearn.datasets  import  make_hastie_10_2
from xgboost.sklearn import XGBClassifier
X, y = make_hastie_10_2(random_state=0)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0)##test_size测试
clf = XGBClassifier(
silent=0 ,#设置成1则没有运行信息输出，最好是设置为0.是否在运行升级时打印消息。
#nthread=4,# cpu 线程数 默认最大
learning_rate= 0.3, # 如同学习率
min_child_weight=1,
# 这个参数默认是 1，是每个叶子里面 h 的和至少是多少，对正负样本不均衡时的 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。
max_depth=6, # 构建树的深度，越大越容易过拟合
gamma=0,  # 树的叶子节点上作进一步分区所需的最小损失减少,越大越保守，一般0.1、0.2这样子。
subsample=1, # 随机采样训练样本 训练实例的子采样比
max_delta_step=0,#最大增量步长，我们允许每个树的权重估计。
colsample_bytree=1, # 生成树时进行的列采样
reg_lambda=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。
#reg_alpha=0, # L1 正则项参数
#scale_pos_weight=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。平衡正负权重
#objective= 'multi:softmax', #多分类的问题 指定学习任务和相应的学习目标
#num_class=10, # 类别数，多分类与 multisoftmax 并用
n_estimators=100, #树的个数
seed=1000 #随机种子
#eval_metric= 'auc'
)
clf.fit(X_train,y_train,eval_metric='auc')
#设置验证集合 verbose=False不打印过程
clf.fit(X_train, y_train,eval_set=[(X_train, y_train), (X_val, y_val)],eval_metric='auc',verbose=False)
#获取验证集合结果
evals_result = clf.evals_result()
y_true, y_pred = y_test, clf.predict(X_test)
print"Accuracy : %.4g" % metrics.accuracy_score(y_true, y_pred)
#回归
#m_regress = xgb.XGBRegressor(n_estimators=1000,seed=0)
```

网格搜索

可以先固定一个参数 最优化后继续调整

第一步：确定学习速率和tree_based 给个常见初始值 根据是否类别不平衡调节

max_depth,min_child_weight,gamma,subsample,scale_pos_weight

max_depth=3 起始值在4-6之间都是不错的选择。

min_child_weight比较小的值解决极不平衡的分类问题eg:1

subsample, colsample_bytree = 0.8: 这个是最常见的初始值了

scale_pos_weight = 1: 这个值是因为类别十分不平衡。

第二步： max_depth 和 min_weight 对最终结果有很大的影响

'max_depth' :range(3,10,2),

'min_child_weight' :range(1,6,2)

先大范围地粗调参数，然后再小范围地微调。

第三步：gamma参数调优

'gamma' :[i/10.0 for i in range(0,5)]

第四步：调整subsample 和 colsample_bytree 参数

'subsample' :[i/100.0 for i in range(75,90,5)],

'colsample_bytree' :[i/100.0 for i in range(75,90,5)]

第五步：正则化参数调优

'reg_alpha' :[1e-5, 1e-2, 0.1, 1, 100]

'reg_lambda'

第六步：降低学习速率

learning_rate =0.01,

```
from sklearn.model_selection import GridSearchCV
tuned_parameters= [{'n_estimators':[100,200,500],
                    'max_depth':[3,5,7], ##range(3,10,2)
                    'learning_rate':[0.5, 1.0],
                    'subsample':[0.75,0.8,0.85,0.9]
                    }]
tuned_parameters= [{'n_estimators':[100,200,500,1000]
                    }]
clf = GridSearchCV(XGBClassifier(silent=0,nthread=4,learning_rate= 0.5,min_child_weight=1, max_depth=3,
clf.fit(X_train, y_train)
##clf.grid_scores_, clf.best_params_, clf.best_score_
print(clf.best_params_)
y_true, y_pred = y_test, clf.predict(X_test)
print"Accuracy : %.4g" % metrics.accuracy_score(y_true, y_pred)
y_proba=clf.predict_proba(X_test)[:,1]
print "AUC Score (Train): %f" % metrics.roc_auc_score(y_true, y_proba)
```

```
from sklearn.model_selection import GridSearchCV
parameters= [{'learning_rate':[0.01,0.1,0.3],'n_estimators':[1000,1200,1500,2000,2500]}]
clf = GridSearchCV(XGBClassifier(
             max_depth=3,
             min_child_weight=1,
             gamma=0.5,
             subsample=0.6,
             colsample_bytree=0.6,
             objective= 'binary:logistic', #逻辑回归损失函数
             scale_pos_weight=1,
             reg_alpha=0,
             reg_lambda=1,
             seed=27
            ),
           param_grid=parameters,scoring='roc_auc')
clf.fit(X_train, y_train)
print(clf.best_params_)
y_pre= clf.predict(X_test)
y_pro= clf.predict_proba(X_test)[:,1]
print "AUC Score : %f" % metrics.roc_auc_score(y_test, y_pro)
print"Accuracy : %.4g" % metrics.accuracy_score(y_test, y_pre)
```

输出特征重要性

```
import pandas as pd
import matplotlib.pylab as plt
feat_imp = pd.Series(clf.booster().get_fscore()).sort_values(ascending=False)
feat_imp.plot(kind='bar', title='Feature Importances')
plt.ylabel('Feature Importance Score')
plt.show()
```


- 上一篇 GBDT 原理与使用
- 下一篇 并发和并行

我的同类文章

机器学习（30）

- Word2vec原理与应用 2017-03-22 阅读 72
- XGBoost原理与应用 2017-03-09 阅读 182
- 机器学习面试准备(持续更新) 2017-02-20 阅读 414
- sklearn-数据预处理-特征变换 2017-01-16 阅读 162
- 数据处理与模型选择的一些注释 2016-12-26 阅读 198
- gcForest算法理解 2017-03-
- 机器学习面试问题汇总 2017-02-
- 聚类与距离学习笔记 2017-02-
- 结合Scikit-learn介绍几种常用的特征选... 2016-12-
- 对机器学习与数据竞赛的一些总结 2016-12-

更多文章

参考知识库

猜你在找

Python数据分析实战：泰坦尼克…
Python算法实战视频课程--二叉树
Python算法实战视频课程--图
Python算法实战视频课程--队列…
使用决策树算法对测试数据进行…
windows下在Java中使用xgboost …
xgboost在windows下的安装与使用
在Python中使用XGBoost
python+xgboost在windows上的安…
使用g++编译CC++程序链接时出现…

查看评论

暂无评论

核心技术类目

全部主题 Hadoop AWS 移动游戏 Java Android iOS Swift 智能硬件 Docker OpenStack VP
IE10 Eclipse CRM JavaScript 数据库 Ubuntu NFC WAP jQuery BI HTML5 Spring Apa
HTML SDK IIS Fedora XML LBS Unity Splashtop UML components Windows Mobile Rai
Cassandra CloudStack FTC coremail OPhone CouchBase 云计算 iOS6 Rackspace Web App S
Compuware 大数据 aptech Perl Tornado Ruby Hibernate ThinkPHP HBase Pure Solr An
Cloud Foundry Redis Scala Django Bootstrap