

CSDN新首页上线啦，邀请你来立即体验！ (<http://blog.csdn.net/>)

立即体验



博客 (<http://blog.csdn.net/?ref=toolbar>)

学院 (<http://edu.csdn.net?ref=toolbar>)

下载 (<http://download.csdn.net?ref=toolbar>)

更多 ▼



登录 (https://passport.csdn.net/account/login?ref=login_new_github)

注册 (<http://passport.csdn.net/account/mobileregister?ref=toolbar&action=mobileRegister>)

· 实战以及源代码分析 (/activity?utm_source=csdnblog1)

xgboost 实战以及源代码分析

转载

2017年08月23日 11:30:09

706



1.序

距离上一次编辑将近10个月，幸得爱可可老师（微博）推荐，访问量陡增。最近毕业论文与xgboost相关，于是重新写一下这篇文章。

关于xgboost的原理网络上的资源很少，大多数还停留在应用层面，本文通过学习陈天奇博士的PPT、论文、一些网络资源，希望对xgboost原理进行深入理解。（笔者在最后的参考文献中会给出地址）

2.xgboost vs gbdt

说到xgboost，不得不说gbdt，两者都是boosting方法（如图1所示），了解gbdt可以看我这篇文章 地址 (<http://blog.csdn.net/a819825294/article/details/51188740>)。

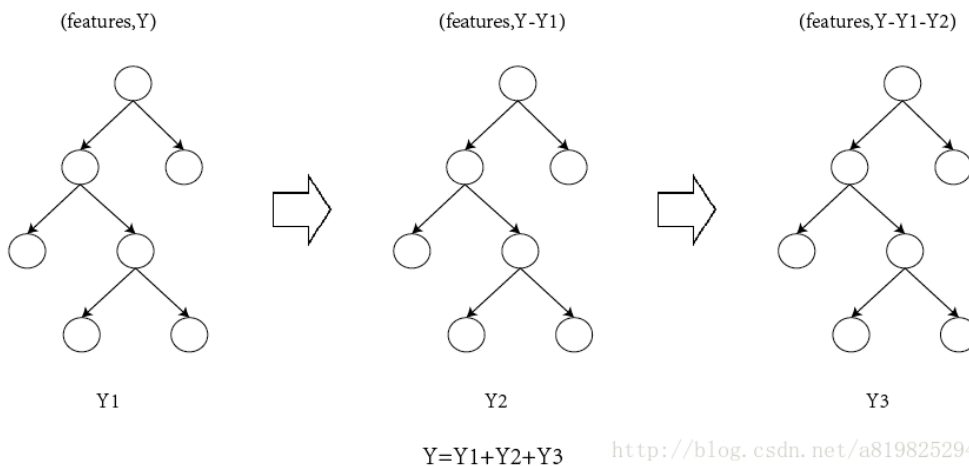


图1

如果不考虑工程实现、解决问题上的一些差异，xgboost与gbdt比较大的不同就是目标函数的定义。



Unable to Conn

The Proxy was unable to connect to the remote site.
responding to requests. If you feel you have reached
please submit a ticket via the link provided below.

URL: <http://pos.baidu.com/s?hei=250&wid=300&di=u...%2Fblog.csdn.net%2Fu010159842%2Farticle%2Fdet>

在线课程



腾讯云容器服务架构实现介绍 0

讲师：董晓杰



内容举报



容器技术在58同城的实践
/July/Courses/series_detail
践 (http://edu.csdn.net/7732utm_source=blog9)
(July/Course

[/series_detail/73?utm_source=blog9\)](#)

他的热门文章

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

• 用泰勒展开来近似我们原来的目标

▪ 泰勒展开: $f(x + \Delta x) \simeq f(x) + f'(x)\Delta x + \frac{1}{2}f''(x)\Delta x^2$

▪ 定义: $g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$, $h_i = \partial_{\hat{y}^{(t-1)}}^2 l(y_i, \hat{y}^{(t-1)})$

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

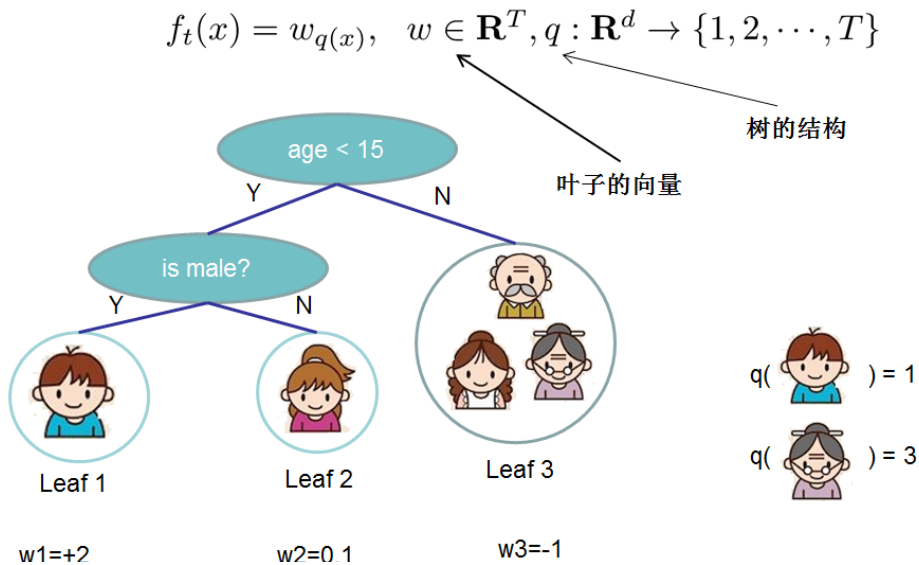
注: 红色箭头指向的 l 即为损失函数; 红色方框为正则项, 包括 $L1$ 、 $L2$; 红色圆圈为常数项。xgboost利用泰勒展开三项, 做一个近似, 我们可以很清晰地看到, 最终的目标函数只依赖于每个数据点的在误差函数上的一阶导数和二阶导数。

3. 原理

对于上面给出的目标函数, 我们可以进一步化简

(1) 定义树的复杂度

对于 f 的定义做一下细化, 把树拆分成结构部分 q 和叶子权重部分 w 。下图是一个具体的例子。结构函数 q 把输入映射到叶子的索引号上面去, 而 w 给定了每个索引号对应的叶子分数是什么。



定义这个复杂度包含了一棵树里面节点的个数, 以及每个树叶子节点上面输出分数的L2模平方。当然这不是唯一的一种定义方式, 不过这一定义方式学习出的树效果一般都比较不错。下图还给出了复杂度计算的一个例子。

Python 数据处理扩展包: pandas 广告
的DataFrame介绍 (创建和基本操作) (<http://blog.csdn.net/u010159842/article/details/52759224>)

15453

ubuntu16.04卸载NVIDIA驱动 (<http://blog.csdn.net/u010159842/article/details/54344583>)

13405

pandas之dataframe移动复制删除列 (<http://blog.csdn.net/u010159842/article/details/53102281>)

13331

如何保存Keras模型 (<http://blog.csdn.net/u010159842/article/details/54407745>)

12513

python删除pandas DataFrame的某一/几列 (<http://blog.csdn.net/u010159842/article/details/52859035>)

12352



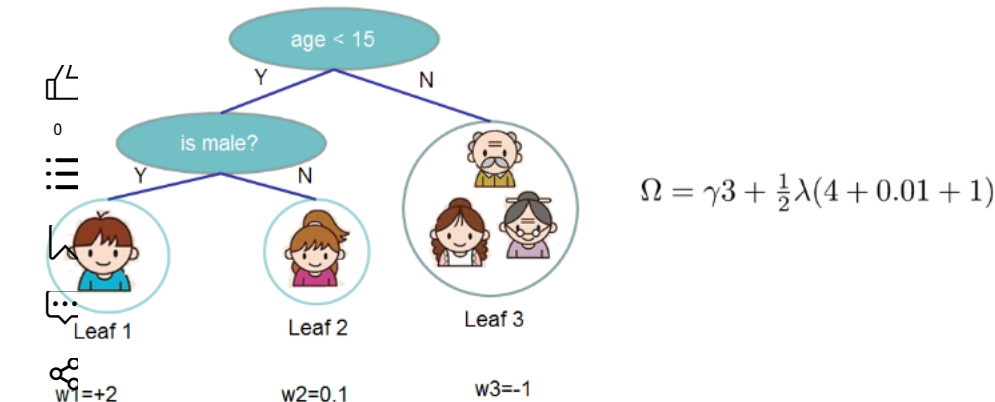
内容举报



返回顶部

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

叶子的个数 w的L2模平方



注：方框部分在最终的模型公式中控制这部分的比重,对应模型参数中的lambda , gamma

在这种新的定义下，我们可以把目标函数进行如下改写，其中I被定义为每个叶子上面样本集合 $I_j = \{i|q(x_i) = j\}$, g是一阶导数，h是二阶导数

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

这一个目标包含了T个相互独立的单变量二次函数。我们可以定义

$$G_j = \sum_{i \in I_j} g_i \quad H_j = \sum_{i \in I_j} h_i$$

最终公式可以化简为

$$\begin{aligned} Obj^{(t)} &= \sum_{j=1}^T \left[\left(\sum_{i \in I_j} g_i \right) w_j + \frac{1}{2} \left(\sum_{i \in I_j} h_i + \lambda \right) w_j^2 \right] + \gamma T \\ &= \sum_{j=1}^T \left[G_j w_j + \frac{1}{2} (H_j + \lambda) w_j^2 \right] + \gamma T \end{aligned}$$

通过对 w_j 求导等于0，可以得到

$$w_j^* = -\frac{G_j}{H_j + \lambda}$$

然后把 w_j 最优解代入得到：

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

(2) 打分函数计算示例

Obj代表了当我们指定一个树的结构的时候，我们在目标上面最多减少多少。我们可以把它叫做结构分数(structure score)








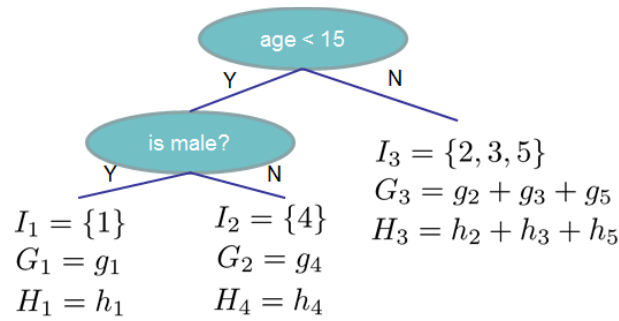
内容举报



返回顶部

广告

样本号	梯度数据
1 	g1, h1
2 	g2, h2
3 	g3, h3
4 	g4, h4
5 	g5, h5



$$Obj = -\frac{1}{2} \sum_j \frac{G_j^2}{H_j + \lambda} + 3\gamma$$

这个分数越小，代表这个树的结构越好

(3) 分裂节点

论文中给出了两种分裂节点的方法

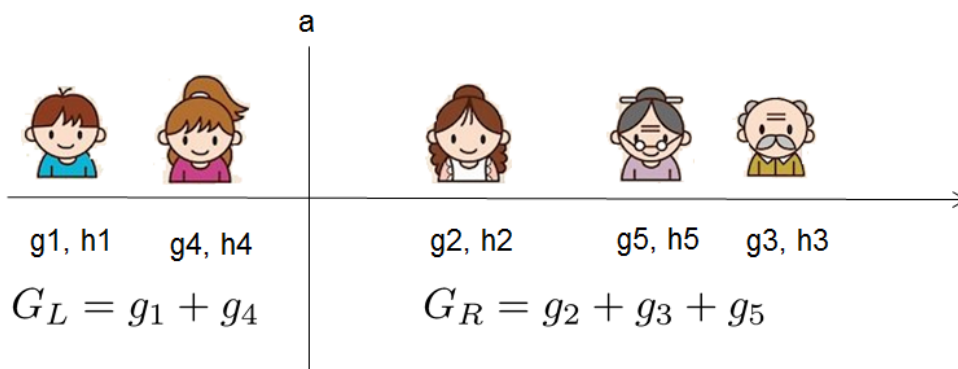
(1) 贪心法：

每一次尝试去对已有的叶子加入一个分割

$$Gain = \frac{1}{2} \left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} \right] - \gamma$$

左子树分数
右子树分数
加入新叶子节点引入的复杂度代价

对于每次扩展，我们还是要枚举所有可能的分割方案，如何高效地枚举所有的分割呢？我假设我们要枚举所有 $x < a$ 这样的条件，对于某个特定的分割 a 我们要计算 a 左边和右边的导数和。



我们可以发现对于所有的 a ，我们只要做一遍从左到右的扫描就可以枚举出所有分割的梯度和 GL 和 GR 。然后用上面的公式计算每个分割方案的分数就可以了。

观察这个目标函数，大家会发现第二个值得注意的事情就是引入分割不一定会使得情况变好，因为我们有一个引入新叶子的惩罚项。优化这个目标对应了树的剪枝，当引入的分割带来的增益小于一个阈值的时候，我们可以剪掉这个分割。大家可以发现，当我们正式地推导目标的时候，像计算分数和剪枝这样的策略都会自然地出现，而不再是一种因为

内容举报

返回顶部

heuristic (启发式) 而进行的操作了。

下面是论文中的算法 (<http://lib.csdn.net/base/datastructure>)

Algorithm 1: Exact Greedy Algorithm for Split Finding

Input: I , instance set of current node
Input: d , feature dimension
 $gain \leftarrow 0$
 $G \leftarrow \sum_{i \in I} g_i, H \leftarrow \sum_{i \in I} h_i$
for $k = 1$ **to** m **do**
 $G_L \leftarrow 0, H_L \leftarrow 0$
 for j **in** $sorted(I, \text{by } x_{jk})$ **do**
 $G_L \leftarrow G_L + g_j, H_L \leftarrow H_L + h_j$
 $G_R \leftarrow G - G_L, H_R \leftarrow H - H_L$
 $score \leftarrow \max(score, \frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{G^2}{H + \lambda})$
 end
end
Output: Split with max score

(2) 近似算法：

主要针对数据太大，不能直接进行计算

Algorithm 2: Approximate Algorithm for Split Finding

for $k = 1$ **to** m **do**
 Propose $S_k = \{s_{k1}, s_{k2}, \dots, s_{kl}\}$ by percentiles on feature k .
 Proposal can be done per tree (global), or per split(local).
end
for $k = 1$ **to** m **do**
 $G_{kv} \leftarrow \sum_{j \in \{j | s_{k,v} \geq x_{jk} > s_{k,v-1}\}} g_j$
 $H_{kv} \leftarrow \sum_{j \in \{j | s_{k,v} \geq x_{jk} > s_{k,v-1}\}} h_j$
end
Follow same step as in previous section to find max score only among proposed splits.

4.自定义损失函数 (指定grad、hess)

(1) 损失函数

Square loss: $l(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2$

Logistic loss: $l(y_i, \hat{y}_i) = y_i \ln(1 + e^{-\hat{y}_i}) + (1 - y_i) \ln(1 + e^{\hat{y}_i})$

(2) grad、hess推导



内容举报



返回顶部

广告

$$\begin{aligned}
 g_i &= \partial_{y_i^{(t-1)}} l(y_i, y_i^{(t-1)}) \\
 &= -y_i \left(1 - \frac{1}{1 + e^{-y_i^{(t-1)}}} \right) + (1 - y_i) \frac{1}{1 + e^{-y_i^{(t-1)}}} \\
 &= \text{Pred} - \text{Label}
 \end{aligned}$$

$$\begin{aligned}
 h_{i_0} &= \partial_{y_i^{(t-1)}}^2 l(y_i, y_i^{(t-1)}) = \frac{e^{-y_i^{(t-1)}}}{(1 + e^{-y_i^{(t-1)}})^2} \\
 &= \text{Pred} * (1 - \text{Pred})
 \end{aligned}$$

(3) 官方代码

```

...
1 #!/usr/bin/python
2 import numpy as np
3 import xgboost as xgb
4 ###
5 # advanced: customized loss function
6 #
7 print ('start running example to used customized objective function')
8
9 dtrain = xgb.DMatrix('./data/agaricus.txt.train')
10 dtest = xgb.DMatrix('./data/agaricus.txt.test')
11
12 # note: for customized objective function, we leave objective as default
13 # note: what we are getting is margin value in prediction
14 # you must know what you are doing
15 param = {'max_depth': 2, 'eta': 1, 'silent': 1}
16 watchlist = [(dtest, 'eval'), (dtrain, 'train')]
17 num_round = 2
18
19 # user define objective function, given prediction, return gradient and second order gradient
20 # this is log likelihood loss
21 def logregobj(preds, dtrain):
22     labels = dtrain.get_label()
23     preds = 1.0 / (1.0 + np.exp(-preds))
24     grad = preds - labels
25     hess = preds * (1.0 - preds)
26     return grad, hess
27
28 # user defined evaluation function, return a pair metric_name, result
29 # NOTE: when you do customized loss function, the default prediction value is margin
30 # this may make builtin evaluation metric not function properly
31 # for example, we are doing logistic loss, the prediction is score before logistic transformation
32 # the builtin evaluation error assumes input is after logistic transformation
33 # Take this in mind when you use the customization, and maybe you need write customized evaluation func
34 def evalerror(preds, dtrain):
35     labels = dtrain.get_label()
36     # return a pair metric_name, result
37     # since preds are margin(before logistic transformation, cutoff at 0)
38     return 'error', float(sum(labels != (preds > 0.0))) / len(labels)
39
40 # training with customized objective, we can also do step by step training
41 # simply look at xgboost.py's implementation of train
42 bst = xgb.train(param, dtrain, num_round, watchlist, logregobj, evalerror)

```



内容举报



返回顶部

5.Xgboost调参

由于xgboost的参数过多，这里介绍三种思路

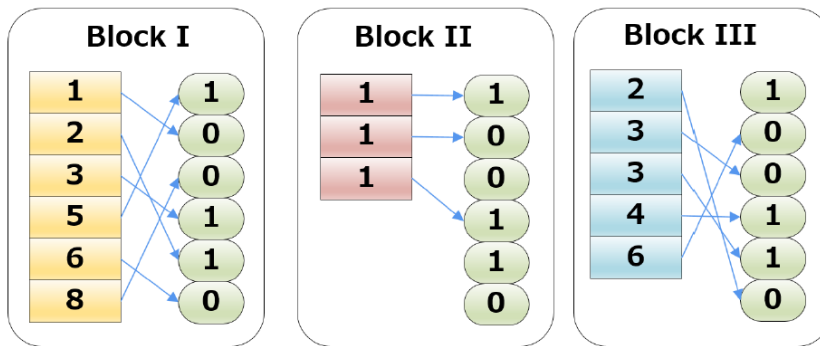
(1) GridSearch (<https://www.kaggle.com/tanitter/introducing-kaggle-scripts/grid-search-xgboost-with-scikit-learn/run/23363>)

(2) Hyperopt (<http://blog.csdn.net/a819825294/article/details/51775418>)

(3) 另外写的一篇文章，操作性比较强，推荐学习一下。地址 (<http://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/>)

6. 工程实现优化

(1) Column Blocks and Parallelization



- Feature values are sorted.
- A block contains one or more feature values.
- Instance indices are stored in blocks.
- Missing features are not stored.
- With column blocks, a **parallel** split finding algorithm is easy to design.

(2) Cache Aware Access

- A thread pre-fetches data from non-continuous memory into a continuous buffer.
- The main thread accumulates gradients statistics in the continuous buffer.

(3) System Tricks

- Block pre-fetching.
- Utilize multiple disks to parallelize disk operations.
- LZ4 compression (popular recent years for outstanding performance).
- Unrolling loops.
- OpenMP



内容举报



返回顶部

7. 代码走读

这块非常感谢杨军老师的无私奉献【4】

个人看代码用的是SourceInsight，由于xgboost有些文件是cc后缀名，可以通过以下命令修改下（默认的认识不了）

```
1 find ./ -name "*.cc" | awk -F "." '{print $2}' | xargs -i -t mv ./{}.cc ./{}.cpp
```

实际^L上，对XGBoost的源码进行走读分析之后，能够看到下面的主流程：

```
0
1 cli_main.cc:
2 main()
3   -> CLIRunTask()
4   -> CLITrain()
5   -> DMatrix::Load()
6   -> learner = Learner::Create()
7   -> learner->Configure()
8   -> learner->InitModel()
9   -> for (i = 0; i < param.num_round; ++i)
10      -> learner->UpdateOneIter()
11      -> learner->Save()
12 learner.cc:
13 Create()
14   -> new LearnerImpl()
15 Configure()
16 InitModel()
17   -> LazyInitModel()
18   -> obj_ = ObjFunction::Create()
19   -> objective.cc
20       Create()
21       -> SoftmaxMultiClassObj(multiclass_obj.cc)/
22       LambdaRankObj(rank_obj.cc)/
23       RegLossObj(regression_obj.cc)/
24       PoissonRegression(regression_obj.cc)
25   -> gbm_ = GradientBooster::Create()
26   -> gbm.cc
27       Create()
28       -> GBTree(gbtree.cc)/
29       GBLinear(gblinear.cc)
30   -> obj_->Configure()
31   -> gbm_->Configure()
32 UpdateOneIter()
33   -> PredictRaw()
34   -> obj_->GetGradient()
35   -> gbm_->DoBoost()
36
37 gbtree.cc:
38 Configure()
39   -> for (up in updaters)
40     -> up->Init()
41 DoBoost()
42   -> BoostNewTrees()
43   -> new_tree = new RegTree()
44   -> for (up in updaters)
45     -> up->Update(new_tree)
46
47 tree_updater.cc:
48 Create()
49   -> ColMaker/DistColMaker(updater_colmaker.cc)/
50   SketchMaker(updater_skmaker.cc)/
51   TreeRefresher(updater_refresh.cc)/
52   TreePruner(updater_prune.cc)/
53   HistMaker/CQHistMaker/
54   GlobalProposalHistMaker/
55   QuantileHistMaker(updater_hismaker.cc)/
56   TreeSyncher(updater_sync.cc)
```



内容举报



返回顶部

从上面的代码主流程可以看到，在XGBoost的实现中，对算法进行了模块化的拆解，几个重要的部分分别是：

广告

- I. ObjFunction：对应于不同的Loss Function，可以完成一阶和二阶导数的计算。
- II. GradientBooster：用于管理Boost方法生成的Model，注意，这里的Booster Model既可以对应于线性Booster Model，也可以对应于Tree Booster Model。
- III. Updater：用于建树，根据具体的建树策略不同，也会有多种Updater。比如，在XGBoost里为了性能优化，既提供了单机多线程并行加速，也支持多机分布式加速。也就提供了若干种不同的并行建树的updater实现，按并行策略的不同，包括：
 - I. inter-feature exact parallelism（特征级精确并行）
 - II. inter-feature approximate parallelism（特征级近似并行，基于特征分bin计算，减少了枚举所有特征分裂点的开销）
 - III. intra-feature parallelism（特征内并行）

此外，为了避免overfit，还提供了一个用于对树进行剪枝的updater(TreePruner)，以及一个用于在分布式场景下完成结点模型参数信息通信的updater(TreeSyncher)，这样设计，关于建树的主要操作都可以通过Updater链的方式串接起来，比较一致干净，算是Decorator设计模式[4]的一种应用。

XGBoost的实现中，最重要的就是建树环节，而建树对应的代码中，最主要的也是Updater的实现。所以我们会以Updater的实现作为介绍的入手点。

以ColMaker（单机版的inter-feature parallelism，实现了精确建树的策略）为例，其建树操作大致如下：

广告

```

• 1 updater_colmaker.cc:
• 2 ColMaker::Update()
• 3 -> Builder builder;
• 4 -> builder.Update()
• 5 -> InitData()
• 6 -> InitNewNode() // 为可用于split的树结点 (即叶子结点, 初始情况下只有一个
• 7 // 叶结点, 也就是根结点) 计算统计量, 包括gain/weight等
• 8 -> for (depth = 0; depth < 树的最大深度; ++depth)
• 9 -> FindSplit()
• 10 -> for (each feature) // 通过OpenMP获取
• 11 // inter-feature parallelism
• 12 -> UpdateSolution()
• 13 -> EnumerateSplit() // 每个执行线程处理一个特征,
• 14 // 选出每个特征的
• 15 // 最优split point
• 16 -> ParallelFindSplit()
• 17 // 多个执行线程同时处理一个特征, 选出该特征
• 18 // 的最优split point;
• 19 // 在每个线程里汇总各个线程内分配到的数据样
• 20 // 本的统计量(grad/hess);
• 21 // aggregate所有线程的样本统计(grad/hess),
• 22 // 计算出每个线程分配到的样本集合的边界特征值作为
• 23 // split point的最优分割点;
• 24 // 在每个线程分配到的样本集合对应的特征值集合进
• 25 // 行枚举作为split point, 选出最优分割点
• 26 -> SyncBestSolution()
• 27 // 上面的UpdateSolution()/ParallelFindSplit()
• 28 // 会为所有待扩展分割的叶结点找到特征维度的最优split
• 29 // point, 比如对于叶结点A, OpenMP线程1会找到特征F1
• 30 // 的最优split point, OpenMP线程2会找到特征F2的最
• 31 // 优split point, 所以需要进行全局sync, 找到叶结点A
• 32 // 的最优split point。
• 33 -> 为需要进行分割的叶结点创建孩子结点
• 34 -> ResetPosition()
• 35 // 根据上一步的分割动作, 更新样本到树结点的映射关系
• 36 // Missing Value(i.e. default)和非Missing Value(i.e.
• 37 // non-default)分别处理
• 38 -> UpdateQueueExpand()
• 39 // 将待扩展分割的叶子结点用于替换qexpand_, 作为下一轮split的
• 40 // 起始基础
• 41 -> InitNewNode() // 为可用于split的树结点计算统计量

```

8.python、R对于xgboost的简单使用

任务：二分类，存在样本不平衡问题（scale_pos_weight可以一定程度上解读此问题）

【Python (<http://lib.csdn.net/base/python>)】



内容举报



返回顶部

广告

```
109 def xgboost_predict():
110     import xgboost as xgb
111     #xgboost start here
112     dtest = xgb.DMatrix(test_x)
113     dval = xgb.DMatrix(val_X, label=val_y)
114     dtrain = xgb.DMatrix(X, label=y)
115     params={
116         'booster':'gbtree',
117         'objective': 'binary:logistic',
118         'early_stopping_rounds':100,
119         'scale_pos_weight': weight,
120         'eval_metric': 'auc',
121         'gamma': '0.1',
122         'max_depth':8,
123         'lambda':550,
124         'subsample':0.7,
125         'colsample_bytree':0.4,
126         'min_child_weight':3,
127         'eta': 0.02,
128         'seed':random_seed,
129         'nthread':7
130     }
131     watchlist = [(dval,'val'), (dtrain,'train')]
132     xgboost_model = xgb.train(params,dtrain,num_boost_round=3000,evals=watchlist)
133     #xgboost_model.save_model('./model/xgb.model')
134
135     #predict test set (from the best iteration)
136     xgboost_predict_y = xgboost_model.predict(dtest,ntree_limit=xgboost_model.best_ntree_limit)
```

【R】

```
124 # fit xgboost model
125 dtrain=xgb.DMatrix(data=train.new[,-1],label=1-train.y$y) #train.new[,-1]表示去掉第一列，即uid列
126 dtest= xgb.DMatrix(data=test.new[,-1])
127 # dunlabeled = xgb.DMatrix(data=train_unlabeled.new[,-1])
128
129 p = nrow(train.y[train.y[,2] == 1,])
130 n = nrow(train.y[train.y[,2] == 0,])
131 weight = p/n
132
133 model_xgboost=xgb.train(
134     booster='gbtree',
135     objective='binary:logistic',
136     scale_pos_weight=weight,
137     gamma=0,
138     lambda=700,
139     subsample=0.7,
140     colsample_bytree=0.3,
141     min_child_weight=5,
142     max_depth=8,
143     eta=0.02,
144     data=dtrain,
145     nrounds=1520,
146     metrics='auc',
147     nthread=2
148 )
149
150 # predict probabilities
151 predict_xgboost=1-predict(model_xgboost,dtest)
```

9.xgboost中比较重要的参数介绍

(1) objective [default=reg:linear] 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wTx。
- “count:poisson” –计数问题的poisson回归，输出结果为poisson分布。在poisson回归中，max_delta_step的缺省值为0.7。(used to safeguard optimization)
- “multi:softmax” –让XGBoost采用softmax目标函数处理多分类问题，同时需要设置参数num_class（类别个数）
- “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



内容举报



返回顶部

广告

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

(2) ‘eval_metric’ The choices are listed below , 评估指标:

- “rmse”: root mean square error
- “logloss”: negative log-likelihood
- “error”: Binary classification error rate. It is calculated as $\#(\text{wrong cases})/\#(\text{all cases})$. For the predictions, the evaluation will regard the instances with prediction value larger than 0.5 as positive instances, and the others as negative instances.
- “merror”: Multiclass classification error rate. It is calculated as $\#(\text{wrong cases})/\#(\text{all cases})$.
- “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 without any positive samples as 1. By adding “-” in the evaluation metric XGBoost will evaluate these score as 0 to be consistent under some conditions.

(3) lambda [default=0] L2 正则的惩罚系数

(4) alpha [default=0] L1 正则的惩罚系数

(5) lambda_bias 在偏置上的L2正则。缺省值为0 (在L1上没有偏置项的正则, 因为L1时偏置不重要)

(6) eta [default=0.3]

为了防止过拟合, 更新过程中用到的收缩步长。在每次提升计算之后, 算法会直接获得新特征的权重。eta通过缩减特征的权重使提升计算过程更加保守。缺省值为0.3

取值范围为: [0,1]

(7) max_depth [default=6] 数的最大深度。缺省值为6, 取值范围为: [1, ∞]

(8) min_child_weight [default=1]

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则拆分过程结束。在现行回归模型中, 这个参数是指建立每个模型所需要的最小样本数。该成熟越大算法越conservative

取值范围为: [0, ∞]

10.DART

核心理想就是将dropout引入XGBoost

示例代码



内容举报



返回顶部

广告

```
1 import xgboost as xgb
2 # read in data
3 dtrain = xgb.DMatrix('demo/data/agaricus.txt.train')
4 dtest = xgb.DMatrix('demo/data/agaricus.txt.test')
5 # specify parameters via map
6 param = {'booster': 'dart',
7         'max_depth': 5, 'learning_rate': 0.1,
8         'objective': 'binary:logistic', 'silent': True,
9         'sample_type': 'uniform',
10        'normalize_type': 'tree',
11        'rate_drop': 0.1,
12        'skip_drop': 0.5}
13 num_round = 50
14 bst = xgb.train(param, dtrain, num_round)
15 # make prediction
16 # ntree_limit must not be 0
17 preds = bst.predict(dtest, ntree_limit=num_round)
```

更多细节可以阅读参考文献5

11.csr_matrix训练XGBoost

当数据规模比较大、较多列比较稀疏时，可以使用csr_matrix训练XGBoost模型，从而节约内存。

下面是Kaggle比赛中TalkingData开源的代码，可以学习一下，详见参考文献6。



内容举报



返回顶部

广告

```

1 import pandas as pd
2 import numpy as np
3 import seaborn as sns
4 import matplotlib.pyplot as plt
5 import os
6 from sklearn.preprocessing import LabelEncoder
7 from scipy.sparse import csr_matrix, hstack
8 import xgboost as xgb
9 from sklearn.cross_validation import StratifiedKFold
10 from sklearn.metrics import log_loss
11
12 datadir = '../input'
13 gatrain = pd.read_csv(os.path.join(datadir, 'gender_age_train.csv'),
14                       index_col='device_id')
15 gatest = pd.read_csv(os.path.join(datadir, 'gender_age_test.csv'),
16                       index_col='device_id')
17 phone = pd.read_csv(os.path.join(datadir, 'phone_brand_device_model.csv'))
18 # Get rid of duplicate device ids in phone
19 phone = phone.drop_duplicates('device_id', keep='first').set_index('device_id')
20 events = pd.read_csv(os.path.join(datadir, 'events.csv'),
21                       parse_dates=['timestamp'], index_col='event_id')
22 appevents = pd.read_csv(os.path.join(datadir, 'app_events.csv'),
23                           usecols=['event_id', 'app_id', 'is_active'],
24                           dtype={'is_active': bool})
25 applabels = pd.read_csv(os.path.join(datadir, 'app_labels.csv'))
26
27 gatrain['trainrow'] = np.arange(gatrain.shape[0])
28 gatest['testrow'] = np.arange(gatest.shape[0])
29
30 brandencoder = LabelEncoder().fit(phone.phone_brand)
31 phone['brand'] = brandencoder.transform(phone['phone_brand'])
32 gatrain['brand'] = phone['brand']
33 gatest['brand'] = phone['brand']
34 Xtr_brand = csr_matrix((np.ones(gatrain.shape[0]),
35                           (gatrain.trainrow, gatrain.brand)))
36 Xte_brand = csr_matrix((np.ones(gatest.shape[0]),
37                           (gatest.testrow, gatest.brand)))
38 print('Brand features: train shape {}, test shape {}'.format(Xtr_brand.shape, Xte_brand.shape))
39
40 m = phone.phone_brand.str.cat(phone.device_model)
41 modelencoder = LabelEncoder().fit(m)
42 phone['model'] = modelencoder.transform(m)
43 gatrain['model'] = phone['model']
44 gatest['model'] = phone['model']
45 Xtr_model = csr_matrix((np.ones(gatrain.shape[0]),
46                           (gatrain.trainrow, gatrain.model)))
47 Xte_model = csr_matrix((np.ones(gatest.shape[0]),
48                           (gatest.testrow, gatest.model)))
49 print('Model features: train shape {}, test shape {}'.format(Xtr_model.shape, Xte_model.shape))
50
51 appencoder = LabelEncoder().fit(appevents.app_id)
52 appevents['app'] = appencoder.transform(appevents.app_id)
53 napps = len(appencoder.classes_)
54 deviceapps = (appevents.merge(events[['device_id']], how='left', left_on='event_id', right_index=True)
55               .groupby(['device_id', 'app'])['app'].agg(['size'])
56               .merge(gatrain[['trainrow']], how='left', left_index=True, right_index=True)
57               .merge(gatest[['testrow']], how='left', left_index=True, right_index=True)
58               .reset_index())
59
60 d = deviceapps.dropna(subset=['trainrow'])
61 Xtr_app = csr_matrix((np.ones(d.shape[0]), (d.trainrow, d.app)),
62                       shape=(gatrain.shape[0], napps))
63 d = deviceapps.dropna(subset=['testrow'])
64 Xte_app = csr_matrix((np.ones(d.shape[0]), (d.testrow, d.app)),
65                       shape=(gatest.shape[0], napps))
66 print('Apps data: train shape {}, test shape {}'.format(Xtr_app.shape, Xte_app.shape))

```



内容举报



返回顶部

广告

```

67
68applabels = applabels.loc[applabels.app_id.isin(appevents.app_id.unique())]
69applabels['app'] = appencoder.transform(applabels.app_id)
70labelencoder = LabelEncoder().fit(applabels.label_id)
71applabels['label'] = labelencoder.transform(applabels.label_id)
72nlabels = len(labelencoder.classes_)
73
74devicelabels = (deviceapps[['device_id', 'app']]
75                    .merge(applabels[['app', 'label']])
76                    .groupby(['device_id', 'label'])['app'].agg(['size'])
77                    .merge(gatrain[['trainrow']], how='left', left_index=True, right_index=True)
78                    .merge(gatest[['testrow']], how='left', left_index=True, right_index=True)
79                    .reset_index())
80devicelabels.head()
81
82d = devicelabels.dropna(subset=['trainrow'])
83Xtr_label = csr_matrix((np.ones(d.shape[0]), (d.trainrow, d.label)),
84                        shape=(gatrain.shape[0], nlabels))
85d = devicelabels.dropna(subset=['testrow'])
86Xte_label = csr_matrix((np.ones(d.shape[0]), (d.testrow, d.label)),
87                        shape=(gatest.shape[0], nlabels))
88print('Labels data: train shape {}, test shape {}'.format(Xtr_label.shape, Xte_label.shape))
89
90Xtrain = hstack((Xtr_brand, Xtr_model, Xtr_app, Xtr_label), format='csr')
91Xtest = hstack((Xte_brand, Xte_model, Xte_app, Xte_label), format='csr')
92print('All features: train shape {}, test shape {}'.format(Xtrain.shape, Xtest.shape))
93
94targetencoder = LabelEncoder().fit(gatrain.group)
95y = targetencoder.transform(gatrain.group)
96
97##### XGBOOST #####
98
99params = {}
100params['booster'] = 'gblinear'
101params['objective'] = "multi:softprob"
102params['eval_metric'] = 'mlogloss'
103params['eta'] = 0.005
104params['num_class'] = 12
105params['lambda'] = 3
106params['alpha'] = 2
107
108# Random 10% for validation
109kf = list(StratifiedKFold(y, n_folds=10, shuffle=True, random_state=4242))[0]
110
111Xtr, Xte = Xtrain[kf[0], :], Xtrain[kf[1], :]
112ytr, yte = y[kf[0]], y[kf[1]]
113
114print('Training set: ' + str(Xtr.shape))
115print('Validation set: ' + str(Xte.shape))
116
117d_train = xgb.DMatrix(Xtr, label=ytr)
118d_valid = xgb.DMatrix(Xte, label=yte)
119
120watchlist = [(d_train, 'train'), (d_valid, 'eval')]
121
122f = xgb.train(params, d_train, 1000, watchlist, early_stopping_rounds=25)
123
124pred = clf.predict(xgb.DMatrix(Xtest))
125
126pred = pd.DataFrame(pred, index = gatest.index, columns=targetencoder.classes_)
127pred.head()
128pred.to_csv('sparse_xgb.csv', index=True)
129
130params['lambda'] = 1
131for alpha in [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]:
132    params['alpha'] = alpha

```



内容举报



返回顶部

- 13# clf = xgb.train(params, d_train, 1000, watchlist, early_stopping_rounds=25)
- 13# print('\n' + str(alpha))

广告

12.Tip

- (1) 含有缺失进行训练

```
dtrain= xgb.DMatrix(x_train, y_train, missing=np.nan )
```

13.参考文献

- (1) xgboost导读和实战 (<http://pan.baidu.com/s/1gfA6FK3>)
- (2) xgboost (<http://pan.baidu.com/s/1eR7ADge>)
- (3) 自定义目标函数 (https://github.com/dmlc/xgboost/blob/master/demo/guide-python/custom_objective.py)
- (4) 机器学习算法中GBDT和XGBOOST的区别有哪些? (<https://www.zhihu.com/question/41354392>)
- (5) DART (<http://xgboost.readthedocs.io/en/latest/tutorials/dart.html>)
- (6) <https://www.kaggle.com/anokas/sparse-xgboost-starter-2-26857/code/code> (<https://www.kaggle.com/anokas/sparse-xgboost-starter-2-26857/code/code>)
- (7) XGBoost: Reliable Large-scale Tree Boosting System (http://learningsys.org/papers/LearningSys_2015_paper_32.pdf)
- (8) XGBoost: A Scalable Tree Boosting System (http://delivery.acm.org/10.1145/2940000/2939785/p785-chen.pdf?ip=202.118.228.100&id=2939785&acc=ACTIVE%20SERVICE&key=BF85BBA5741FDC6E.5C4511229FC427D6.4D4702B0C3E38B35.4D4702B0C3E38B35&CFID=905733202&CFTOKEN=53852884&_acm_=1488265641_ffdebf36cef2b1bf7f3f76abf6bfe426)



相关文章推荐

XGBoost源码阅读笔记(2)--树构造之Exact Greedy Algorithm (<http://blog.csdn.net/flydre...>)

在上一篇《XGBoost源码阅读笔记(1)--代码逻辑结构》中向大家介绍了XGBoost源码的逻辑结构，同时也简单介绍了XGBoost的基本情况。本篇将继续向大家介绍XGBoost源码是如何构造一颗回...

flydreamforever (<http://blog.csdn.net/flydreamforever>) 2017年07月27日 20:47 674

xgboost原理 (<http://blog.csdn.net/a819825294/article/details/51206410>)





内容举报





返回顶部

文章内容可能会相对比较多，读者可以点击上方目录，直接阅读自己感兴趣的章节。**1.序** 距离上一次编辑将近**10**个月，幸得爱可可老师（微博）推荐，访问量陡增。最近毕业论文与**xgboost**相关，于是重新写一下...

广告

 **a819825294** (<http://blog.csdn.net/a819825294>) 2016年04月21日 10:15  86736



XGBoost设计思路与数学推导 (<http://blog.csdn.net/chedan541300521/article/details/546...>)

 **chedan541300521** (<http://blog.csdn.net/chedan541300521>) 2017年01月21日 23:50  1539

XGBoost是由陈天奇大神设计的一套基于**gbdt**的可并行计算的机器学习工具。在**kaggle**、天池等大数据竞赛有着广泛的应用。通过阅读论文和代码，受益良多，并总结了包括公式推导、并行化设计和源码剖析等...

xgboost代码示例 (<http://blog.csdn.net/zhuqihui/article/details/72584376>)

之前写过很久了，怕新更新的**xgboost**不再适用，重新调试了一下代码，可运行，但数据得换成自己的，**xgboost**，都应该知道它的威力了，这里不再多说，欢迎一起讨论！# coding=utf-8 im...

 **zhuqihui** (<http://blog.csdn.net/zhuqihui>) 2017年05月20日 19:28  822



Delphi7高级应用开发随书源码 (<http://download.csdn.net/detail/chexh...>)

<http://download.csdn.net/detail/chexh...> 2003年04月30日 00:00 676KB [下载](#)



AI 工程师职业指南

我们请来商汤、杜邦、声智、希为、58同城、爱因互动、中科视拓、鲁朗软件等公司 AI 技术一线的专家，请他们从实践的角度来解析 AI 领域各技术岗位的合格工程师都是怎样炼成的。

(http://www.baidu.com/cb.php?c=lgF_pyfqhHmknjzrj00lZ0qnfK9ujYzP1f4Pjnd0Aw-5Hc4nj6vPjm0TAq15Hf4rjn1n1b0T1Y3njfkuWFbuhN-nWF9nARY0AwY5HDdnHcsnHDYnWc0lgF_5y9YIZ0lQzqMpgwBUvqoQhP8QvIGIAPCmgfEmvq_lyd8Q1R4uhF-rA7Wuj0YmhP9PARvujmYmH0vm1qdlAdxTvqdThP-5HDknWF9mhkEusKzujY4rHb0mhYqn0KsTWYs0ZNGujYkPHTYn1mk0AqGujYkn10snj10APGujYLnWm4n1c0ULI85H00TZbqnW0v0APzm1YdPjcYns)



基于xgboost的贷款风险预测 (<http://blog.csdn.net/luoganttc/article/details/77435064>)

现在我们用传说中的**xgboost**对这个数据集进行计算#!/usr/bin/env python3 #-*- coding: utf-8 -*- Created on Sat Aug 19 ...

 **luoganttc** (<http://blog.csdn.net/luoganttc>) 2017年08月20日 22:09  373

xgboost算法原理与实战 (<http://blog.csdn.net/JasonZhangOO/article/details/73061060>)

xgboost算法原理与实战之前一直有听说GBM，GBDT（Gradient Boost Decision Tree）渐进梯度决策树 GBRT（Gradient Boost RegressionTr...

 **JasonZhangOO** (<http://blog.csdn.net/JasonZhangOO>) 2017年06月11日 19:12  1173



xgboost导读和实战_王超&陈帅华 (<http://download.csdn.net/detai...>)

<http://download.csdn.net/detai...> 2015年06月07日 17:15 868KB [下载](#)



xgboost源代码python (http://download.csdn.net/detail/sinat_29817779/...)

http://download.csdn.net/detail/sinat_29817779/... 2017年11月27日 12:19 2KB [下载](#)



内容举报




返回顶部

广告

xgboost入门与实战（原理篇） (<http://blog.csdn.net/sb19931201/article/details/52557382>)

xgboost入门与实战（原理篇）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...

 sb19931201 (<http://blog.csdn.net/sb19931201>) 2016年09月16日 20:26 44675




xgboost源代码 (<http://download.csdn.net/detail/jingyi130705008/9836887>)

<http://download.csdn.net/detail/jingyi130705008/9836887> 2017年05月08日 22:32 23.66MB 下载()


机器学习xgboost实战—手写数字识别 (http://blog.csdn.net/Eddy_zheng/article/details/501845632)

1、xgboost 安装安装问题这里就不再做赘述，可参考前面写的博文：http://blog.csdn.net/eddy_zheng/article/details/501845632

 Eddy_zheng (http://blog.csdn.net/Eddy_zheng) 2016年01月11日 12:13 12682


XGBoost实战与调优 (http://blog.csdn.net/weixin_38569817/article/details/76354004)

首先，python和Anaconda都没有自带xgboost。windows下安装xgboost非常方便。在前面的文章中，提供了下载地址和详细的安装步骤。你可以在python中，输入i...

 weixin_38569817 (http://blog.csdn.net/weixin_38569817) 2017年07月29日 21:13 358


史上最详细的XGBoost实战（下） (<http://blog.csdn.net/SzM21C11U68n04vdcLmJ/article/details/76354004>)

作者：章华燕编辑：田旭四 XGBoost 参数详解在运行XGboost之前，必须设置三种类型成熟：general parameters, booster parameters...

 SzM21C11U68n04vdcLmJ (<http://blog.csdn.net/SzM21C11U68n04vdcLmJ>) 2017年11月12日 00:00 34


Xgboost参数调优的完整指南及实战 (<http://blog.csdn.net/u010665216/article/details/785...>)

引言Xgboost是一种高度复杂的算法可以处理各种各样的数据。相信每个用过Xgboost的人都有过这样的感受：利用Xgboost构建模型十分简单，但是用Xgboost来调参提升模型就很难了。该算法使用...

 u010665216 (<http://blog.csdn.net/u010665216>) 2017年11月14日 17:09 337

史上最详细的XGBoost实战（上） (<http://blog.csdn.net/SzM21C11U68n04vdcLmJ/article/details/76354004>)

作者：章华燕编辑：祝鑫泉零环境介绍：· Python版本：3.6.2 · 操作系统：Windows · 集成开发环境：PyCharm ...

 SzM21C11U68n04vdcLmJ (<http://blog.csdn.net/SzM21C11U68n04vdcLmJ>) 2017年10月31日 00:00 102

实战from GBDT to Xgboost (http://blog.csdn.net/BD_Jiang/article/details/73351336)

这一系列主要是对DT、RF的简单介绍，以及对GBDT源码（Python）分析，然后成功搭建Xgboost工具，最后通过简单demo实例熟悉Xgboost建模过程。1. Decision Tree ...

 BD_Jiang (http://blog.csdn.net/BD_Jiang) 2017年06月16日 22:00 284



内容举报




返回顶部

广告


实战xgboost与sklearn与pandas训练模型 (http://blog.csdn.net/qq_36330643/article/deta...

import cPickle import xgboost as xgb import numpy as np from sklearn.model_selection import KFold, t...

 qq_36330643 (http://blog.csdn.net/qq_36330643) 2017年11月09日 16:32 63


xgboost入门与实战（原理篇） (http://blog.csdn.net/chivalrousli/article/details/54409316)

本文转载自:http://blog.csdn.net/sb19931201/article/details/52557382 前言: xgboost是大规模并行boosted tr...

 chivalrousli (http://blog.csdn.net/chivalrousli) 2017年01月13日 15:50 2042

R语言实战：机器学习与数据分析源代码2 (http://blog.csdn.net/baimafujinji/article/detai...

本文辑录了《R语言实战——机器学习与数据分析》一书第4章至第5章之代码。整合R语言深藏不露的强大威力，决胜数据分析之巅。且听我将统计学之精髓娓娓道来，助你砥砺大数据时代的掘金技法。探寻数据挖掘之术， ...

 baimafujinji (http://blog.csdn.net/baimafujinji) 2016年06月12日 17:29 2069



内容举报



返回顶部