



雪城大学



loft公寓



银杏树价格



我的世界游戏



享受日本免税购物

Japan. Tax-free Shop



访问： 380696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)

[登录](#) | [注册](#)

[目录视图](#)

[摘要视图](#)

[RSS](#) [订阅](#)

[图灵赠书——程序员11月书单](#)
[【思考】Python这么厉害的原因竟然是！](#)
[感恩节赠书：《深度学习》等异步社区优秀图书和作者评选启动！](#)
[每周荐书：京东架构、Linux内核、Python全栈](#)

xgboost入门与实战（实战调参篇）

标签： [xgboost](#)

2017-01-13 15:55
1305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/52577592>

xgboost入门与实战（实战调参篇）

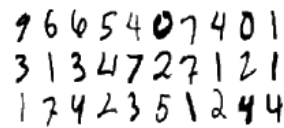
前言

前面几篇博文都在学习原理知识，是时候上数据上模型跑一跑了。本文用的数据来自[kaggle](#)，相信搞机器学习的同学们都知道它，kaggle上有几个老题目一直开放，适合给新手练级，上面还有很多老司机的方案共享以及讨论，非常方便新手入门。这次用的数据是[Classify handwritten digits using the famous MNIST data](#)——手写数字识别，每个样本相当于一个图片像素矩阵（28x28），每个像元就是一个特征啦。用这个数据的好处就是不用做特征工程了，对于上手模型很方便。


xgboost安装看这里：<http://blog.csdn.net/sb19931201/article/details/52236020>

数据集


1.数据介绍：数据采用的是广泛应用于机器学习社区的MNIST数据集：




The data for this competition were taken from the MNIS National Institute of Standards and Technology”) dataset community that has been extensively studied. More deta Learning algorithms that have been tried on it and their at<http://yann.lecun.com/exdb/mnist/index.html>.




宠物焚烧炉



工商管理硕士



怎样软件开发



北美服务器

- 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)

2.训练数据集（共42000个样本）：如下图所示，第一列是label标签，后面共28x28=784个像素特征，特征取值范围为0-255。

	A	B	C	D	E	F	G	H	I	J	K	L
1	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	pixel10
2	1	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0
4	1	0	0	0	0	0	0	0	0	0	0	0
5	4	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0
8	7	0	0	0	0	0	0	0	0	0	0	0
9	3	0	0	0	0	0	0	0	0	0	0	0
...
pixel229	pixel230	pixel231	pixel232	pixel233	pixel234	pixel235	pixel236	pixel237	pixel238	pixel239	pixel240	pixel241
0	0	0	0	0	0	0	0	0	0	0	0	0
0	61	191	254	254	254	254	254	109	83	19		
0	0	0	0	0	0	0	9	254	254	18		
0	0	160	207	6	6	0	0	0	0	0		
0	23	210	253	253	253	248	161	222	222	24		
0	0	0	0	8	211	254	58	0	0	0		
0	122	253	252	253	252	223	243	253	252	25		
0	207	254	254	177	117	39	0	0	56	24		

3.测试数据集（共28000条记录）：就是我们要预测的数据集了，没有标签值，通过train.c模型，然后根据test.csv的特征数据集来预测这28000的类别（0-9的数字标签）。

pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	pixel10	pixel11
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0

4.结果数据样例：两列，一列为ID，一列为预测标签值。

A	B
ImageId	Label
1	1
2	1
3	2
4	3
5	4
6	5
7	6
8	7
9	8
10	9
11	10
12	11
13	12
14	13
15	14
16	15
17	16
18	17
19	18
20	19
21	20

xgboost模型调参、训练（python）

1.导入相关库，读取数据

```
1 import numpy as np
2 import pandas as pd
3 import xgboost as xgb
4 from sklearn.cross_validation import train_test_split
5
6 #记录程序运行时间
7 import time
```

WEB架构师成长之路之一-走正... (1)

谷歌推自然语言理解框架SLIN... (1)

Python 拷贝对象（深拷贝dee... (1)

推荐文章

* 【2017年11月27日】CSDN博客更新周报

* 【CSDN】邀请您来GitChat赚钱啦！

* 【GitChat】精选——JavaScript进阶指南

* 改做人工智能之前，90%的人都没能给自己定位

* TensorFlow 人脸识别网络与对抗网络搭建

* Vue 移动端项目生产环境优化

* 面试必考的计算机网络知识点梳理

最新评论

Python 拷贝对象（深拷贝deepcopy与浅...
lang_mumo : 你好 其余能知道 知道 为什么 C里面没有5呀

python opencv去图片水印
Johnny_Xiu : 挺垃圾的一个处理，核心就是颜色替换，呵呵了

谷歌推自然语言理解框架SLING，看文本...
wonderwhy20 : 有没有相应的学习资料

97.5%准确率的深度学习中文分词（字嵌...
taonie6673 : 请问：使用word2vec对字进行嵌入，指定维度50的向量，具体生成后什么样子呢？

vim中按Ctrl+s 终端疑似卡死
chvalrous : 哈哈哈哈哈，同感。

vim中按Ctrl+s 终端疑似卡死
李_柱 : 随意查了一下，嘛的，++..... 我的眼泪也留下来了。不知道×了多少窗口了

卷积神经网络CNN
qq_36118352 : 请问下博主，为什么是四个数据集，一个训练集x和一个测试集x不可以吗？新手哈

用深度学习（CNN RNN Attention）解决...
chvalrous : 我现在的场景是要对文章标题这种短文本进行分类，分类的类别为40+个，我使用textCNN进行试验，目...

网络爬虫工作原理分析
qq_36423458 : Python基础与爬虫技术 课程学习地址：http://www.xuetuwuyou.com/cou...

在windows下安装scala出现错误：找不到...
疯狂的鱈江 : @vermouthlove:应该是空格的问题，不要安装在C:\Program Files下面，同时s...

```
8 start_time = time.time()
9
10 #读入数据
11 train = pd.read_csv("Digit_Recognizer/train.csv")
12 tests = pd.read_csv("Digit_Recognizer/test.csv")
```

2.划分数据集

```
1 #用sklearn.cross_validation进行训练数据集划分，这里训练集和交叉验证集比例为7：3，可以自
2 train_xy,val = train_test_split(train, test_size = 0.3,random_state=1)
3
4 y = train_xy.label
5 X = train_xy.drop(['label'],axis=1)
6 val_y = val.label
7 val_X = val.drop(['label'],axis=1)
8
9 #xgb矩阵赋值
10 xgb_val = xgb.DMatrix(val_X,label=val_y)
11 xgb_train = xgb.DMatrix(X, label=y)
12 xgb_test = xgb.DMatrix(tests)
```

3.xgboost模型

```
1 params={
2 'booster':'gbtree',
3 'objective':'multi:softmax', #多分类的问题
4 'num_class':10, # 类别数，与 multisoftmax 并用
5 'gamma':0.1, # 用于控制是否后剪枝的参数,越大越保守，一般0.1、0.2这样子。
6 'max_depth':12, # 构建树的深度，越大越容易过拟合
7 '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。
8 'subsample':0.7, # 随机采样训练样本
9 'colsample_bytree':0.7, # 生成树时进行的列采样
10 'min_child_weight':3,
11 # 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
12 #，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。
13 #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 overfitti
14 'silent':0, #设置成1则没有运行信息输出，最好是设置为0。
15 'eta': 0.007, # 如同学习率
16 'seed':1000,
17 'nthread':7, # cpu 线程数
18 '#eval_metric': 'auc'
19 }
20 plst = list(params.items())
21 num_rounds = 5000 # 迭代次数
22 watchlist = [(xgb_train, 'train'),(xgb_val, 'val')]
23
24 #训练模型并保存
25 # 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
26 model = xgb.train(plst, xgb_train, num_rounds, watchlist, early_stopping_rounds=100)
27 model.save_model('./model/xgb.model') # 用于存储，而非训练模型
28 print "best best_ntree_limit",model.best_ntree_limit
```

关闭

4.预测并保存

```
1 preds = model.predict(xgb_test,ntree_limit=model.best_ntree_limit)
2
3 np.savetxt('xgb_submission.csv',np.c_[range(1,len(tests)+1),preds],delimiter=',',header='ImageId
4
5 #输出运行时长
6 cost_time = time.time()-start_time
7 print "xgboost success!",'\n',"cost time:",cost_time,"(s)....."
```

5. 变量信息

Name	Type	Size	Value
y	Series	(29400L,)	32141 8 10895 6
watchlist	list	2	[(<xgboost...28C5C748>, 'train'), (<xgboost...1B4B1710>, 'val')]
val_y	Series	(12600L,)	29633 1 345 5
val_X	DataFrame	(12600, 784)	Column names: pixel0, pixel1, pixel2, pixel3, pixel4, pixel5, pixel6, pixel7, pi ...
val	DataFrame	(12600, 785)	Column names: label, pixel0, pixel1, pixel2, pixel3, pixel4, pixel5, pixel6, pix ...
train_xy	DataFrame	(29400, 785)	Column names: label, pixel0, pixel1, pixel2, pixel3, pixel4, pixel5, pixel6, pix ...
train	DataFrame	(42000, 785)	Column names: label, pixel0, pixel1, pixel2, pixel3, pixel4, pixel5, pixel6, pix ...
tests	DataFrame	(28000, 784)	Column names: pixel0, pixel1, pixel2, pixel3, pixel4, pixel5, pixel6, pixel7, pi ...
test	int64	(28000L, 784L)	Min: 0 Max: 255
start_time	float	1	1474199127.27
plst	list	13	[('seed', 1000), ('colsample_bytree', 0.7), ('booster', 'gbtree'), ('num_class', ...
params	dict	13	{'booster': 'gbtree', 'colsample_bytree': 0.7, 'eta': 0.1, 'gamma': 0.1, 'lambda ...
num_rounds	int	1	5000
X	DataFrame	(29400, 784)	Column names: pixel0, pixel1, pixel2, pixel3, pixel4, pixel5, pixel6, pixel7, pi ...

预测结果评价

将预测得到的xgb_submission.csv文件上传到kaggle，看系统评分。

由于迭代次数和树的深度设置的都比较大，程序还在训练中，等这次跑完在将这两个参数调整小一些，看看运行时间已经预测精度的变化。

```
[21:05:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 42 extra nodes, 42 pruned nodes, max_depth=8
[21:05:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 52 extra nodes, 46 pruned nodes, max_depth=10
[1616]    train-merror:0          val-merror:0.030397
[21:05:27] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 20 extra nodes, 24 pruned nodes, max_depth=6
[21:05:27] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 20 extra nodes, 16 pruned nodes, max_depth=5
[21:05:27] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 30 extra nodes, 36 pruned nodes, max_depth=6
[21:05:27] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 42 extra nodes, 34 pruned nodes, max_depth=9
[21:05:28] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 32 extra nodes, 40 pruned nodes, max_depth=8
[21:05:28] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 34 extra nodes, 32 pruned nodes, max_depth=9
[21:05:28] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 20 extra nodes, 24 pruned nodes, max_depth=5
[21:05:28] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 36 extra nodes, 30 pruned nodes, max_depth=7
[21:05:29] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 34 extra nodes, 52 pruned nodes, max_depth=8
```

Python 1

```
[21:17:41] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 24 extra nodes, 32 pruned nodes, max_depth=6
[21:17:41] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 26 extra nodes, 26 pruned nodes, max_depth=5
[21:17:41] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 20 extra nodes, 22 pruned nodes, max_depth=5
[21:17:42] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 16 extra nodes, 38 pruned nodes, max_depth=4
[21:17:42] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 30 extra nodes, 32 pruned nodes, max_depth=6
[21:17:42] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 26 extra nodes, 52 pruned nodes, max_depth=6
[1919]    train-merror:0          val-merror:0.029206
Stopping. Best iteration:
[1819]    train-merror:0          val-merror:0.029206
```

1819-1919不再提升，故提前跳出迭代过程

跑到这里了save_model

关闭

此处先留个坑。。。不同版本的运行结果我会陆续贴出来。

版本：1 运行时长：中间程序报错了一下，大概是2500s

参数：

```
params={
'booster':'gbtree',
'objective': 'multi:softmax', #多分类的问题
'num_class':10, # 类别数, 与 multisoftmax 并用
'gamma':0.1, # 用于控制是否后剪枝的参数,越大越保守,一般0.1、0.2这样子。
'max_depth':12, # 构建树的深度, 越大越容易过拟合
'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。
'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
'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。
'silent':0, #设置成1则没有运行信息输出, 最好是设置为0.
'eta': 0.007, # 如同学习率
'seed':1000,
'nthread':7, # cpu 线程数
#'eval_metric': 'auc'
}
```

版本：1

成绩：

638 new 我曾经被山河大海跨过0.966861Sun, 18 Sep 2016 13:34:09

Your Best Entry ↑
Congratulations on making your first submission!

Tweet this!

版本：2 运行时长:2275s

参数：

```
params={
'booster':'gbtree',
'objective': 'multi:softmax', #多分类的问题
'num_class':10, # 类别数, 与 multisoftmax 并用
'gamma':0.2, # 用于控制是否后剪枝的参数,越大越保守,一般0.1、0.2这样子。
'max_depth':8, # 构建树的深度, 越大越容易过拟合
'lambda':3, # 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。
'subsample':0.7, # 随机采样训练样本
'colsample_bytree':0.5, # 生成树时进行的列采样
'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。
'silent':0, #设置成1则没有运行信息输出, 最好是设置为0.
'eta': 0.01, # 如同学习率
'seed':1000,
'nthread':8, # cpu 线程数
#'eval_metric': 'auc'
}
```

版本：2

关闭

成绩：

622 new 我曾经被山河大海跨过0.967432Sun, 18 Sep 2016 14:27:56

Your Best Entry ↑
You improved on your best score by 0.00057.

You just moved up 17 positions on the leaderboard. Tweet this!

·
结
果
图
片
·
·
·

附：完整代码

```
#coding=utf-8
"""
Created on 2016/09/17
By 我曾经被山河大海跨过
"""

import numpy as np
import pandas as pd
import xgboost as xgb
from sklearn.cross_validation import train_test_split

#from xgboost.sklearn import XGBClassifier
#from sklearn import cross_validation, metrics #Additional sklearn functions
#from sklearn.grid_search import GridSearchCV #Perforing grid search
#
#import matplotlib.pyplot as plt
#from matplotlib.pyplot import rcParams

#记录程序运行时间
import time
start_time = time.time()

#读入数据
train = pd.read_csv("Digit_Recognizer/train.csv")
tests = pd.read_csv("Digit_Recognizer/test.csv")

params={
    'booster':'gbtree',
    'objective': 'multi:softmax', #多分类的问题
    'num_class':10, # 类别数，与 multisoftmax 并用
    'gamma':0.1, # 用于控制是否后剪枝的参数,越大越保守，一般0.1、0.2这样子。
    'max_depth':12, # 构建树的深度，越大越容易过拟合
    '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，模型越简单，模型越简单越不容易过拟合。
    '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
    'min_child_weight':3,
    # 这个参数默认是 1，是每个叶子节点 h 的和至少是多少，对正负样本不平衡时的 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 overfitti
    'silent':0, #设置成1则没有运行信息输出，最好是设置为0。
    'eta': 0.007, # 如同学习率
    'seed':1000,
    'nthread':7, # cpu 线程数
    'eval_metric': 'auc'
```

关闭

```
}

plst = list(params.items())
num_rounds = 5000 # 迭代次数

train_xy, val = train_test_split(train, test_size = 0.3, random_state=1)
# random_state is of big influence for val-auc
y = train_xy.label
X = train_xy.drop(['label'], axis=1)
val_y = val.label
val_X = val.drop(['label'], axis=1)

xgb_val = xgb.DMatrix(val_X, label=val_y)
xgb_train = xgb.DMatrix(X, label=y)
xgb_test = xgb.DMatrix(tests)

watchlist = [(xgb_train, 'train'), (xgb_val, 'val')]

# training model
# early_stopping_rounds 当设置的迭代次数较大时，early_stopping_rounds 可设为0
model = xgb.train(plst, xgb_train, num_rounds, watchlist, early_stopping_rounds=100)

model.save_model('./model/xgb.model') # 用于存储训练出的模型
print "best_ntree_limit", model.best_ntree_limit

print "跑到这里了model.predict"
preds = model.predict(xgb_test, ntree_limit=model.best_ntree_limit)

np.savetxt('xgb_submission.csv', np.c_[range(1, len(tests)+1), preds], delimiter=',', header='ImageId')

# 输出运行时长
cost_time = time.time() - start_time
print "xgboost success!", "\n", "cost time:", cost_time, "(s)"
```

顶 踩
2 0

- [上一篇](#) [xgboost入门与实战（原理篇）](#)
- [下一篇](#) [Hbase shell 常用命令（1）](#)

关闭

相关文章推荐

- XGBoost-Python完全调参指南-参数解释篇
- 腾讯云容器服务架构实现介绍-董晓杰
- 机器学习系列(12)_XGBoost参数调优完全指南（附...
- 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使用调参
- MySQL深入浅出

- xgboost 调参经验
- Python可以这样学（第三季：多线程与多进程编程...
- XGBoost：参数解释
- 华为工程师，带你实战C++
- XGBoost-Python完全调参指南-介绍篇
- xgboost 调参经验
- xgboost调参
- XGBoost简易调参指南



查看评论

暂无评论

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