



(<http://lib.csdn.net/base/machinelearning>)

机器学习 (<http://lib.csdn.net/base/machinelearning>) - 资源汇总 (<http://lib.csdn.net/machinelearning/node/23>) - 案例分析 (<http://lib.csdn.net/machinelearning/knowledge/54>)

👁 2833    💬 0

## 王小草【机器学习】笔记--提升之XGBoost工具的应用

作者：sinat\_33761963 ([http://my.csdn.net/sinat\\_33761963](http://my.csdn.net/sinat_33761963))

笔记整理时间：2016年12月29日

整理者：王小草

欢迎关注：

王小草的FM喜马拉雅主播频道：搜索账号名“好吧我真的叫王草”

王小草的个人微信公众号：bigdataML

王小草的CSDN博客地址：[http://my.csdn.net/sinat\\_33761963](http://my.csdn.net/sinat_33761963) ([http://my.csdn.net/sinat\\_33761963](http://my.csdn.net/sinat_33761963))

2016年的最后第三天，终于有阳光，连续一个月的咳嗽终于见好。

这是及其忙碌的一年，忙着适应从学校到社会的血腥，忙着在车水马龙的竞争里立足安身，忙着屈服于又抗争于的生活。

在整理“提升”算法的笔记时，我想每年的自己多像一个基函数，在梯度里不断塑造下一个更好的自己，直到实现目标函数最优才停止迭代。而生活不像函数，至少在百年之前，都不愿停止寻找更优。

## 1. XGBoost介绍

XGBoost的作者是华盛顿大学陈天奇。

XGBoost是使用梯度提升框架实现的高效，灵活，可移植的机器学习库。它的全称是eXtreme Gradient Boosting,是GDBT的一个C++实现。它将树的生成并行完成，从而提高学习速度。

一般而言，XGBoost的速度和性能优于sklearn.ensemble.GradientBoostingClassifier类。

XGBoost提供了python接口，它在机器学习的竞赛中纷纷表现出来优异的成绩，其他学者封装了R和Julia等接口。

XGBoost的官网：

<https://xgboost.readthedocs.io/en/latest/> (<https://xgboost.readthedocs.io/en/latest/>)

github上代码的地址：

<https://github.com/dmlc/xgboost/> (<https://github.com/dmlc/xgboost/>)

## 2. Ubuntu安装XGBoost

在各种系统上的安装官网上有详细介绍：

<https://xgboost.readthedocs.io/en/latest/build.html> (<https://xgboost.readthedocs.io/en/latest/build.html>)

因为我平时学习与工作环境都是在ubuntu上的，本文主要介绍ubuntu系统安装XGBoost。

非常简单，就两个命令：

第一步：在github上把XGBoost工程克隆到本地计算机上，可以创建一个专门的目录，然后进入这个目录运行以下命令：

```
1 git clone --recursive https://github.com/dmlc/xgboost
```

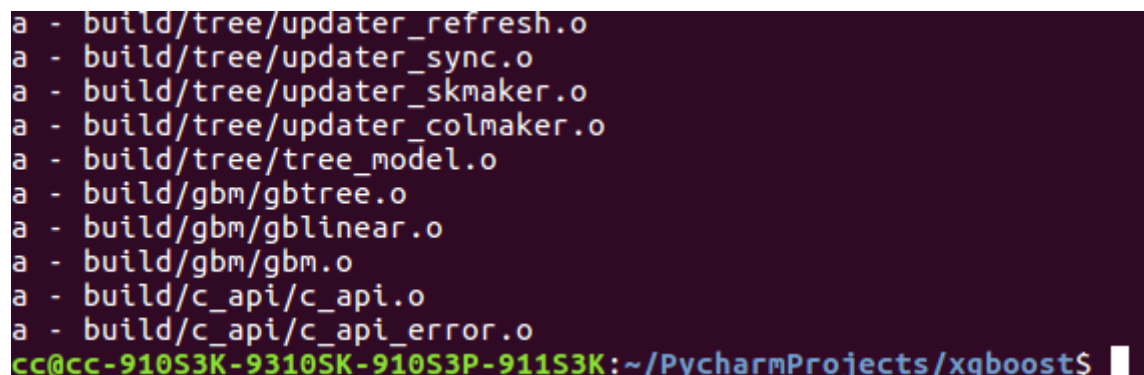
根据网络的好坏，下载需要一点点时间，我大概是花了5分钟。

第二步：进入工程，然后编译。

```
1 cd xgboost
2 make -j4
```

编译的过程大概不到1分钟吧。

编译成功的画面：



```
a - build/tree/updater_refresh.o
a - build/tree/updater_sync.o
a - build/tree/updater_skmaker.o
a - build/tree/updater_colmaker.o
a - build/tree/tree_model.o
a - build/gbm/gbtree.o
a - build/gbm/gblinear.o
a - build/gbm/gbm.o
a - build/c_api/c_api.o
a - build/c_api/c_api_error.o
cc@cc-910S3K-9310SK-910S3P-911S3K:~/PycharmProjects/xgboost$
```

注意：

我机子上有2个python版本，一个是安装anaconda中带着的python,一个是我自己下载python来单独安装的。我的系统默认的是后者，所以以上安装的XGBoost是安装在默认的python中的。

现在我打开pycharm,输入import xgboost as xgb,并没有红色波浪线的报错，表示可以成功使用了！

### 3.XGBoost实践

安装好了之后，我们来尝试着学习与使用这个工具。以下介绍一些从简到繁的案例。

#### 3.1 官网get start小案例

官方文档分别给出了4中语言的小例子，这里只讲解python的。

样本数据说明：

读取的数据是工程里自带的数据，在工程目录下的/demo/data/文件夹下。打开数据文件，格式是这样的：

```
1 3:1 10:1 11:1 21:1 30:1 34:1 36:1 40:1 41:1 53:1 58:1 65:1 69:1 77:1 86:1 88:1 92:1 95:1
0 3:1 10:1 20:1 21:1 23:1 34:1 36:1 39:1 41:1 53:1 56:1 65:1 69:1 77:1 86:1 88:1 92:1 95:1
0 1:1 10:1 19:1 21:1 24:1 34:1 36:1 39:1 42:1 53:1 56:1 65:1 69:1 77:1 86:1 88:1 92:1 95:1
1 3:1 9:1 19:1 21:1 30:1 34:1 36:1 40:1 42:1 53:1 58:1 65:1 69:1 77:1 86:1 88:1 92:1 95:1
0 3:1 10:1 14:1 22:1 29:1 34:1 37:1 39:1 41:1 54:1 58:1 65:1 69:1 77:1 86:1 88:1 92:1 95:1
```

每一行是一个观测样本，第一列是label,后面的列都是特征，一个特征由特征的索引，冒号，特征值组成，列与列之间是用空格隔开的。

在某样本点中没有出现的特征索引，说明该处特征值维0，也就是说，整个数据是一个稀疏的矩阵，所以只存储不为0的数据。

数据读取说明：

直接将path传入xgb.DMatrix()中，会将数据变成DMatrix的格式，这是一个XGBoost自己定义的数据格式（就像numpy中有ndarray, pandas中有dataframe数据格式一样）。这个格式会将第一列作为label,其余的作为features。

参数说明：

模型中可以传入一些列参数，在训练之前，先把这些参数以map的形式定义好。参数有很多，下面的例子中是最基本的。

每个参数的意义在代码中都有注释，再此不赘述。

```
1 # /usr/bin/python
2 # -*- encoding:utf-8 -*-
3
4 import xgboost as xgb
5
6 # 读取数据
7 dtrain = xgb.DMatrix('/home/cc/PycharmProjects/xgboost/demo/data/agaricus.txt.train')
8 dtest = xgb.DMatrix('/home/cc/PycharmProjects/xgboost/demo/data/agaricus.txt.test')
9
10 # 设置参数，参数的格式用map的形式存储
11 param = {'max_depth': 2,                # 树的最大深度
12          'eta': 1,                      # 一个防止过拟合的参数，默认0.3
13          'silent': 1,                  # 打印信息的繁简指标，1表示简，0表示繁
14          'objective': 'binary:logistic'} # 使用的模型，分类的数目
15
16 num_round = 2 # 迭代的次数
17
18 # 看板，每次迭代都可以在控制台打印出训练集与测试集的损失
19 watchlist = [(dtest, 'eval'), (dtrain, 'train')]
20
21 # 训练模型
22 bst = xgb.train(param, dtrain, num_round, evals=watchlist)
23
24 # 做预测
25 preds = bst.predict(dtest)
26
27 # 打印结果
28 y_hat = preds
29 y = dtest.get_label()
30 print y_hat
31 print y
32
33 error_count = sum(y != (y_hat > 0.5))
```

```
34 error_rate = float(error_count) / len(y_hat)
35 print "样本总数:\t", len(y_hat)
36 print "错误数目:\t%4d" % error_count
37 print "错误率:\t%.2f%%" % (100 * error_rate)
```

来看看上面代码的输出：

```
[14:27:00] 6513x127 matrix with 143286 entries loaded from /home/cc/PycharmProjects/xgboost
[14:27:00] 1611x127 matrix with 35442 entries loaded from /home/cc/PycharmProjects/xgboost
[0] eval-error:0.042831 train-error:0.046522
[1] eval-error:0.021726 train-error:0.022263
[ 0.28583017  0.92392391  0.28583017 ...,  0.92392391  0.05169873
 0.92392391]
[ 0.  1.  0. ...,  1.  0.  1.]
样本总数：    1611
错误数目：     35
错误率：      2.17%
```

Process finished with exit code 0

### 3.2 自定义基函数与损失

在做提升的时候我们一般选取决策树这样的弱预测器来作为基函数，也可以使用逻辑回归。逻辑回归其实本身是一个强分类器，提升强分类器不一定会有更好的表现，但也具体问题具体分析。这个基函数 $f$ 其实是可以根据实际需求来调整的，当然也可以自己构造。

下面的案例中我们不适用XGBboost自带的基函数，而是自己定义基函数，然后再传入模型中训练。

```
1 # /usr/bin/python
2 # -*- encoding:utf-8 -*-
3
4 import xgboost as xgb
5 import numpy as np
6
7
8 # 定义f: theta * x
9 def log_reg(y_hat, y):
10     p = 1.0 / (1.0 + np.exp(-y_hat))
11     g = p - y.get_label()
12     h = p * (1.0-p)
13     return g, h
14
15
16 # 定义误差率
17 def error(y_hat, y):
18     return 'error', float(sum(y.get_label() != (y_hat > 0.0))) / len(y_hat)
19
20
21 if __name__ == "__main__":
22     # 读取数据
23     dtrain = xgb.DMatrix('/home/cc/PycharmProjects/xgboost/demo/data/agaricus.txt.
24     dtest = xgb.DMatrix('/home/cc/PycharmProjects/xgboost/demo/data/agaricus.txt.t
25
26     # 设置参数, 参数的格式用map的形式存储
27     param = {'max_depth': 2, # 树的最大深度
28             'eta': 1, # 一个防止过拟合的参数, 默认0.3
29             'silent': 1} # 打印信息的繁简指标, 1表示简, 0表示繁
30
31     num_round = 2 # 迭代的次数
32
33     # 看板, 每次迭代都可以在控制台打印出训练集与测试集的损失
```

```

34     watchlist = [(dtest, 'eval'), (dtrain, 'train')]
35
36     # 训练模型
37     bst = xgb.train(param, dtrain, num_round, evals=watchlist, obj=log_reg, feval=
38
39     # 计算错误率
40     y_hat = bst.predict(dtest)
41     y = dtest.get_label()
42     print y_hat
43     print y
44     error = sum(y != (y_hat > 0))
45     error_rate = float(error) / len(y_hat)
46     print '样本总数:\t', len(y_hat)
47     print '错误数目:\t%d' % error
48     print '错误率:\t%.2f%%' % (100 * error_rate)

```

打印的结果：

```

[14:53:48] 6513x127 matrix with 143286 entries loaded from /home/cc/PycharmProjects/xgboost
[14:53:48] 1611x127 matrix with 35442 entries loaded from /home/cc/PycharmProjects/xgboost
[0] eval-error:0.042831 train-error:0.046522
[1] eval-error:0.021726 train-error:0.022263
[-1.04997814  2.57504988 -1.04997814 ...,  2.57504988 -3.01916885
 2.57504988]
[ 0.  1.  0. ...,  1.  0.  1.]
样本总数：    1611
错误数目：      35
错误率：      2.17%

Process finished with exit code 0

```

显然，模型并不可观，仍然有35个错误的样本点。于是我们需要去调整参数优化模型，比如，将迭代的次数改成3，其他的不变，输出的结果如下，错误率降低到了0.62%



```
[14:54:55] 6513x127 matrix with 143286 entries loaded from /home/cc/PycharmProjects/xgboost
[14:54:55] 1611x127 matrix with 35442 entries loaded from /home/cc/PycharmProjects/xgboost
[0] eval-error:0.042831 train-error:0.046522
[1] eval-error:0.021726 train-error:0.022263
[2] eval-error:0.006207 train-error:0.007063
[-1.83141637  1.79361176 -1.83141637 ...,  3.24044585 -3.8006072
 3.24044585]
[ 0.  1.  0. ...,  1.  0.  1.]
样本总数：    1611
错误数目：      10
错误率：      0.62%
```

Process finished with exit code 0

再比如，其他不变，只将最大深度max\_depth改称3，结果如下，亮瞎眼睛，测试集的误差居然为0了！

```
[14:57:11] 6513x127 matrix with 143286 entries loaded from /home/cc/PycharmProjects/xgboost
[14:57:11] 1611x127 matrix with 35442 entries loaded from /home/cc/PycharmProjects/xgboost
[0] eval-error:0.016139 train-error:0.014433
[1] eval-error:0      train-error:0.001228
[-3.03464127  3.02548742 -3.03464127 ...,  3.02548742 -3.26926422
 3.02548742]
[ 0.  1.  0. ...,  1.  0.  1.]
样本总数：    1611
错误数目：      0
错误率：      0.00%
```

Process finished with exit code 0

### 3.3 softmax多分类问题

下面案例介绍一个用softmax做多分类的问题。还是使用那个家喻户晓的iris数据集，通过一系列特征去预测花的种类，总共有3类。

看一眼数据集的格式是酱紫的，总共有5列，前4列是4个特征，最后一列是label。

可见这个Label是string类型，在输入模型前，我们需要把它转换成数字作为标识。

```
5.1,3.5,1.4,0.2,Iris-setosa  
4.9,3.0,1.4,0.2,Iris-setosa  
7.0,3.2,4.7,1.4,Iris-versicolor  
6.4,3.2,4.5,1.5,Iris-versicolor  
6.3,3.3,6.0,2.5,Iris-virginica  
5.8,2.7,5.1,1.9,Iris-virginica
```

完整代码：

自己写了一个方法iris\_type，目的是将string类型的花名，转换成Float类型的数字标识。

与前面的案例不同，这次我们需要自己去分训练集与测试集，可直接调用方法train\_test\_split即可。

另外一点与以上案例不同的是系数中'silent'设置为0，表示打印出更多信息，在下面输出的结果中，后面的一大段信息就是这个导致的。

另外，案例中还调用了逻辑回归模型来做对比，发现正确率低于前者。

```
1 # /usr/bin/python
2 # -*- encoding:utf-8 -*-
3
4 import xgboost as xgb
5 import numpy as np
6 from sklearn.cross_validation import train_test_split
7
8
9 def iris_type(s):
10     it = {'Iris-setosa': 0, 'Iris-versicolor': 1, 'Iris-virginica': 2}
11     return it[s]
12
13
14 if __name__ == "__main__":
15     # 读数据
16     path = "/home/cc/下载/深度学习笔记/提升/8.数据/4.iris.data"
17     data = np.loadtxt(path, dtype=float, delimiter=',', converters={4: iris_type})
18
19     # 将数据集拆分成feature与label两部分
20     x, y = np.split(data, (4, ), axis=1)
21
22     # 将特征与标签数据分别拆分成训练集与测试集
23     x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=1, test
24
25     # 将训练集与测试集变成DMatrix的数据格式
26     train = xgb.DMatrix(x_train, label=y_train)
27     test = xgb.DMatrix(x_test, label=y_test)
28
29     watch_list = [(test, 'eval'), (train, 'train')]
30
31     # 设置参数
32     param = {'max_depth': 3,
33             'eta': 1,
```

```
34         'silent': 0,
35         'objective': 'multi:softmax',
36         'num_class': 3}
37
38     num_round = 3
39
40     # 训练模型
41     bsg = xgb.train(param, train, num_round, evals=watch_list)
42
43     # 预测模型
44     y_hat = bsg.predict(test)
45
46     # 计算误差
47     result = y_test.reshape(1, -1) == y_hat
48     print 'XGBoost正确率:\t', float(np.sum(result)) / len(y_hat)
49     print 'END.....\n'
50
51     # =====#
52     # 逻辑回归
53     lr = LogisticRegression(penalty='l2')
54     lr.fit(x_train, y_train.ravel())
55     y_hat2 = lr.predict(x_test)
56
57     # 计算误差
58     result2 = y_test.reshape(1, -1) == y_hat2
59     print '逻辑回归正确率:\t', float(np.sum(result2)) / len(y_hat2)
60     print 'END.....\n'
```

输出的结果：

```
[0] eval-merror:0.02    train-merror:0.02
[1] eval-merror:0.02    train-merror:0.02
[2] eval-merror:0.04    train-merror:0.01
正确率:    0.96
END.....

[15:57:15] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 0 pruned
[15:57:15] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 8 extra nodes, 0 pruned
[15:57:15] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 4 extra nodes, 0 pruned
[15:57:15] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 0 pruned
[15:57:15] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 8 extra nodes, 0 pruned
[15:57:15] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned
[15:57:15] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 2 extra nodes, 0 pruned
[15:57:15] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned
[15:57:15] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 8 extra nodes, 0 pruned

逻辑回归正确率:    0.88
END.....
```

更多案例请参看：

<https://github.com/dmlc/xgboost/tree/master/demo> (<https://github.com/dmlc/xgboost/tree/master/demo>)

[查看原文>> \(http://blog.csdn.net/sinat\\_33761963/article/details/53930021\)](http://blog.csdn.net/sinat_33761963/article/details/53930021)



### 看过本文的人也看了：

- 机器学习知识结构图  
(<http://lib.csdn.net/base/machinelearning/structure>)
- 推荐系统的基本原理  
(<http://lib.csdn.net/article/machinelearning/45255>)
- 决策树算法介绍及应用  
(<http://lib.csdn.net/article/machinelearning/38073>)
- 简单易学的机器学习算法——协同过滤...  
(<http://lib.csdn.net/article/machinelearning/45110>)
- EM算法 - 2 - EM算法在高斯混合模型...  
(<http://lib.csdn.net/article/machinelearning/45108>)
- Learning to Rank 简介  
(<http://lib.csdn.net/article/machinelearning/48968>)

### 发表评论

输入评论内容

发表

0个评论

公司简介 (<http://www.csdn.net/company/about.html>) | 招贤纳士 (<http://www.csdn.net/company/recruit.html>) | 广告服务 (<http://www.csdn.net/company/marketing.html>) | 联系方式 (<http://www.csdn.net/company/contact.html>) | 版权声明 (<http://www.csdn.net/company/statement.html>) | 法律顾问 (<http://www.csdn.net/company/layer.html>) | 问题报告 (<mailto:webmaster@csdn.net>) | 合作伙伴 (<http://www.csdn.net/friendlink.html>) | 论坛反馈 (<http://bbs.csdn.net/forums/Service>)

网站客服 杂志客服 (<http://wpa.qq.com/msgrd?v=3&uin=2251809102&site=qq&menu=yes>) 微博客服 (<http://e.weibo.com/csdnsupport/profile>) [webmaster@csdn.net](mailto:webmaster@csdn.net) (<mailto:webmaster@csdn.net>)

400-660-0108 | 北京创新乐知信息技术有限公司 版权所有 | 江苏知之为计算机有限公司 | 江苏乐知网络技术有限公司

