Kaggle 神器 xgboost

在 Kaggle 的很多比赛中，我们可以看到很多 winner 喜欢用 xgboost，而且获得非常好的表现，今天就来看看 xgboost 到底是什么以及如何应用。

本文结构：

- 什么是 xgboost?
- 为什么要用它?
- 怎么应用?
- 学习资源

什么是 xgboost?

XGBoost： eXtreme Gradient Boosting 项目地址：https://github.com/dmlc/xgboost

是由 Tianqi Chen http://homes.cs.washington.edu/~tqchen/ 最初开发的实现可扩展，便携，分布式 gradient boosting (GBDT, GBRT or GBM) 算法的一个库，可以下载安装并应用于 C++，Python，R，Julia，Java，Scala，Hadoop，现在有很多协作者共同开发维护。

XGBoost 所应用的算法就是 gradient boosting decision tree，既可以用于分类也可以用于回归问题中。

那什么是 Gradient Boosting?

Gradient boosting 是 boosting 的其中一种方法

所谓 **Boosting**，就是将弱分离器 f_i(x) 组合起来形成强分类器 F(x) 的一种方法。

所以 **Boosting 有三个要素**：

- A loss function to be optimized： 例如分类问题中用 cross entropy，回归问题用 mean squared error。
- A weak learner to make predictions： 例如决策树。
- An additive model： 将多个弱学习器累加起来组成强学习器，进而使目标损失函数达到极小。

Gradient boosting 就是通过加入新的弱学习器，来努力纠正前面所有弱学习器的残差，最终这样多个学习器相加在一起用来进行最终预测，准确率就会比单独的一个要高。之所以称为 Gradient，是因为在添加新模型时使用了梯度下降算法来最小化的损失。

为什么要用 xgboost?

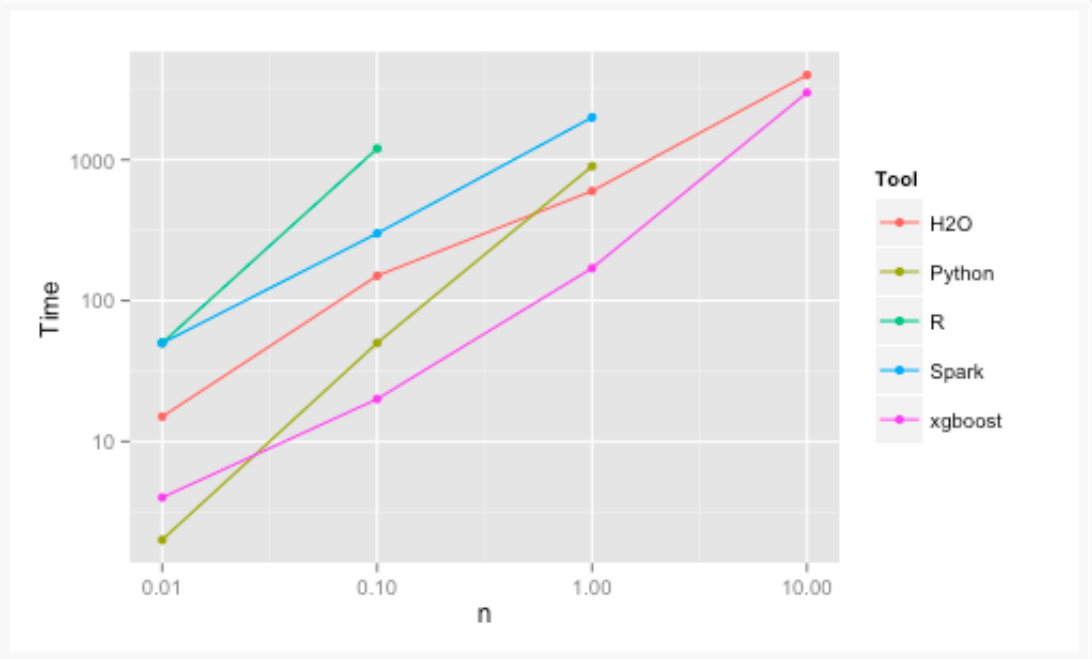
前面已经知道，XGBoost 就是对 gradient boosting decision tree 的实现，但是一般来说，gradient boosting 的实现是比较慢的，因为每次都要先构造出一个树并添加到整个模型序列中。

而 XGBoost 的特点就是**计算速度快，模型表现好**，这两点也正是这个项目的目标。

表现快是因为它具有这样的设计：

- Parallelization： 训练时可以用所有的 CPU 内核来并行化建树。
- Distributed Computing： 用分布式计算来训练非常大的模型。
- Out-of-Core Computing： 对于非常大的数据集还可以进行 Out-of-Core Computing。
- Cache Optimization of data structures and algorithms： 更好地利用硬件。

下图就是 XGBoost 与其它 gradient boosting 和 bagged decision trees 实现的效果比较，可以看出它比 R, Python，Spark，H2O 中的基准配置要更快。



另外一个优点就是在预测问题中**模型表现非常好**，下面是几个 kaggle winner 的赛后采访链接，可以看出 XGBoost 的在实战中的效果。

- Vlad Sandulescu, Mihai Chiru, 1st place of the KDD Cup 2016 competition. Link to the arxiv paper.
- Marios Michailidis, Mathias Müller and HJ van Veen, 1st place of the Dato Truely Native? competition. Link to the Kaggle interview.
- Vlad Mironov, Alexander Guschin, 1st place of the CERN LHCb experiment Flavour of Physics competition. Link to the Kaggle interview.

怎么应用?

先来用 Xgboost 做一个简单的二分类问题，以下面这个数据为例，来判断病人是否会在 5 年内患糖尿病，这个数据前 8 列是变量，最后一列是预测值为 0 或 1。

数据描述： https://archive.ics.uci.edu/ml/datasets/Pima+Indians+Diabetes

下载数据集，并保存为 “pima-indians-diabetes.csv“ 文件：
https://archive.ics.uci.edu/ml/machine-learning-databases/pima-indians-diabetes/pima-indians-diabetes.data

1. 基础应用

引入 xgboost 等包

```
from numpy import loadtxt
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

分出变量和标签

```
dataset = loadtxt('pima-indians-diabetes.csv', delimiter=",")

X = dataset[:,0:8]
Y = dataset[:,8]
```

将数据分为训练集和测试集，测试集用来预测，训练集用来学习模型

```
seed = 7
test_size = 0.33
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=test_size,
random_state=seed)
```

xgboost 有封装好的分类器和回归器，可以直接用 XGBClassifier 建立模型 这里是 XGBClassifier 的文档： http://xgboost.readthedocs.io/en/latest/python/python_api.html#module-xgboost.sklearn

```
model = XGBClassifier()
model.fit(X_train, y_train)
```

xgboost 的结果是每个样本属于第一类的概率，需要用 round 将其转换为 0 1 值

```
y_pred = model.predict(X_test)
predictions = [round(value) for value in y_pred]
```

得到 Accuracy: 77.95%

```
accuracy = accuracy_score(y_test, predictions)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

2. 监控模型表现

xgboost 可以在模型训练时，评价模型在测试集上的表现，也可以输出每一步的分数

只需要将

```
model = XGBClassifier()
model.fit(X_train, y_train)
```

变为：

```
model = XGBClassifier()
eval_set = [(X_test, y_test)]
model.fit(X_train, y_train, early_stopping_rounds=10, eval_metric="logloss",
eval_set=eval_set, verbose=True)
```

那么它会在每加入一颗树后打印出 logloss

```
[31]    validation_0-logloss:0.487867
[32]    validation_0-logloss:0.487297
[33]    validation_0-logloss:0.487562
```

并打印出 Early Stopping 的点：

```
Stopping. Best iteration:
[32]    validation_0-logloss:0.487297
```

3. 输出特征重要度

gradient boosting 还有一个优点是可以给出训练好的模型的特征重要性， 这样就可以知道哪些变量需要被保留，哪些可以舍弃

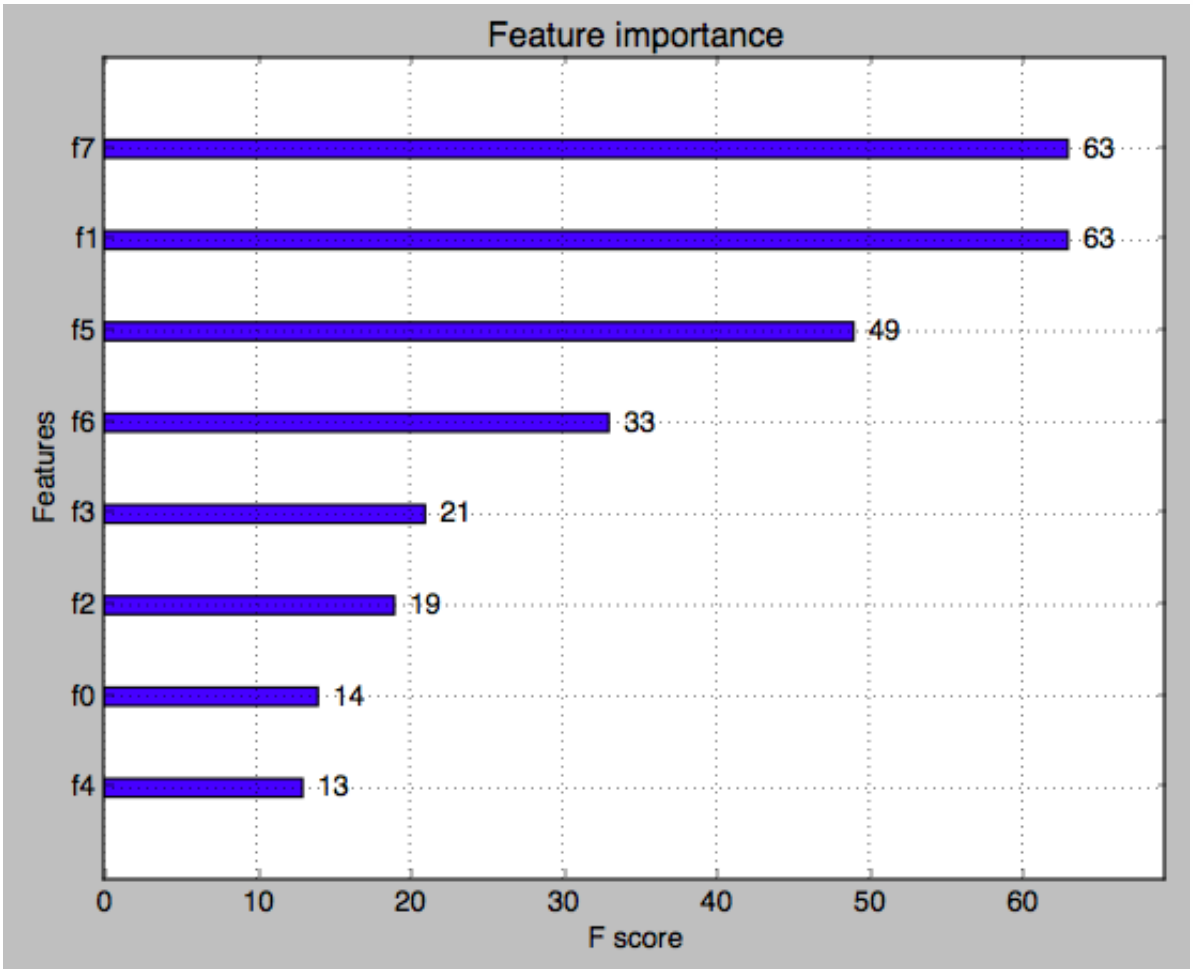
需要引入下面两个类

```
from xgboost import plot_importance
from matplotlib import pyplot
```

和前面的代码相比，就是在 fit 后面加入两行画出特征的重要性

```
model.fit(X, y)

plot_importance(model)
pyplot.show()
```



4. 调参

如何调参呢，下面是三个超参数的一般实践最佳值，可以先将它们设定为这个范围，然后画出 learning curves，再调解参数找到最佳模型：

- learning_rate = 0.1 或更小，越小就需要多加入弱学习器；
- tree_depth = 2～8；
- subsample = 训练集的 30%～80%；

接下来我们用 **GridSearchCV** 来进行调参会更方便一些：

可以调的超参数组合有：

树的个数和大小 `(n_estimators and max_depth)`. 学习率和树的个数 `(learning_rate and n_estimators)`. 行列的 subsampling rates `(subsample, colsample_bytree and colsample_bylevel)`.

下面以学习率为例：

先引入这两个类

```
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import StratifiedKFold
```

设定要调节的 learning_rate = [0.0001, 0.001, 0.01, 0.1, 0.2, 0.3] 和原代码相比就是在 model 后面加上 grid search 这几行：

```
model = XGBClassifier()
learning_rate = [0.0001, 0.001, 0.01, 0.1, 0.2, 0.3]
param_grid = dict(learning_rate=learning_rate)
kfold = StratifiedKFold(n_splits=10, shuffle=True, random_state=7)
grid_search = GridSearchCV(model, param_grid, scoring="neg_log_loss", n_jobs=-1,
cv=kfold)
grid_result = grid_search.fit(X, Y)
```

最后会给出最佳的学习率为 0.1 Best: -0.483013 using {'learning_rate': 0.1}

```
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
```

我们还可以用下面的代码打印出每一个学习率对应的分数：

```
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))
```

```
-0.689650 (0.000242) with: {'learning_rate': 0.0001}
-0.661274 (0.001954) with: {'learning_rate': 0.001}
-0.530747 (0.022961) with: {'learning_rate': 0.01}
-0.483013 (0.060755) with: {'learning_rate': 0.1}
-0.515440 (0.068974) with: {'learning_rate': 0.2}
-0.557315 (0.081738) with: {'learning_rate': 0.3}
```

前面就是关于 xgboost 的一些基础概念和应用实例，下面还有一些学习资源供参考：

学习资源：

Tianqi Chen 的讲座： https://www.youtube.com/watch?v=Vly8xGnNiWs&feature=youtu.be 讲义： https://speakerdeck.com/datasciencela/tianqi-chen-xgboost-overview-and-latest-news-la-meetup-talk

入门教程： https://xgboost.readthedocs.io/en/latest/

安装教程： http://xgboost.readthedocs.io/en/latest/build.html

应用示例： https://github.com/dmlc/xgboost/tree/master/demo

最好的资源当然就是项目的 Github 主页： https://github.com/dmlc/xgboost

参考： http://machinelearningmastery.com/develop-first-xgboost-model-python-scikit-learn/ https://www.zhihu.com/question/37683881