GBST Package Review

Concept：GBST package是基于Gradient Boosting Survival Tree论文中的算法，以及XGBoost库当中成熟的多线程优化方法实现的生存分析python库。鉴于在大量的数据分析任务中我们一般会使用sklearn，所以GBST库当中也提供了完善的sklearn方面的接口。

Package结构如下所示。

调用

Python package

编译

Xgboost.dll(windows)

Libxgboost.so(linux)

增删功能后的XGBOOST src

1. Xgboost src源码导览

用以编译得到package中dll库的xgboost源码来自<https://github.com/dmlc/xgboost> 的master branch（1.0.0beta） 版本。Xgboost本身是一个c++库，由于它同时还提供了python/R/java等多种语言的接口，所以我们的GBST模型也能够被很容易地移植到java和R上。Xgboost库使用Apache License 2.0进行开源，所以在我们注明出处的情况下，无需担心代码商用等等问题。

源码中需要关注的文件夹：

dmlc-core：来自dmlc（也就是xgboost的开发者）的一个轻量化的基础功能库，相当于轮子级别。在src中，参数的声明、存储和读取等等使用了这个库。使用Apache License 2.0开源。

rabit：全称为Reliable Allreduce and Broadcast Interface，同样来自dmlc。使用Apache License 2.0开源。Rabit本身是个可运行的library，同时也可以作为Yarn（hadoop）或者是mpi的interface来使用。在本package的实现中直接使用了rabit本身，不过hadoop/mpi相关的分布式处理功能也可以提上开发日程。

Src：xgboost库本体源码的存放位置。为了将GBST移植进xgboost中，以及使得库本身更加轻量化，对其中的部分代码进行了增加、调整和删减。

源码中修改的文件夹和文件：

src/objective：删去了原有的objective，在multiclass\_obj.cu中新增了SurvTreeObj()。该目标函数是基于GBST算法实现的。

objective相当于自定义一个模型的损失函数。对于xgboost库当中的一个objective来说，我们需要定义的有三项：forward pass的输出，和每次计算的一阶导及二阶导。这部分的实现参照了GBST算法推导出的公式。

src/metric: 在xgboost的模型训练和分析的过程中，metric的值不会像objective那样被用来更新权重，而是作为模型的评价指标，用来确定early-stop何时停下，或者是交叉验证时哪一组参数较为优秀。

我们删去了原有的metric（l1loss, classification accuracy等对生存分析任务来说用不上），新增了survival\_metric.cc，其中包含了新定义的，与GBST论文中的主要指标一致的metric “survivalauc”。其计算方法是对于每个时间点的每个样本而言，计算出该样本的累计生存概率，作为模型判断它是否“活着”（class label=1）的置信率；同一时间点上的所有样本构成一组数据集，计算模型在该时刻的auc。最终输出的AUC指标是所有时间点上AUC的平均。

编译库：

首先在项目根目录下建立工作用文件夹：

mkdir build

cd build

Linux：

cmake ..

make -j$(nproc) // 例如需要使用4线程编译库，那么是 make -j4。

Linux环境下生成的库文件是 ./lib/libxgboost.so

Windows：

cmake .. -G"Visual Studio 14 2015 Win64" //根据电脑上安装的visual studio版本会有所不同。例如，若安装的是vs2017，则需要用"Visual Studio 15 2017 Win64"

cmake --build . --config Release

//这一步也可以在VS内部完成。用visual studio打开xgboost.sln，生成“ALL\_BUILD”。

Windows环境下生成的库文件是./lib/xgboost.dll

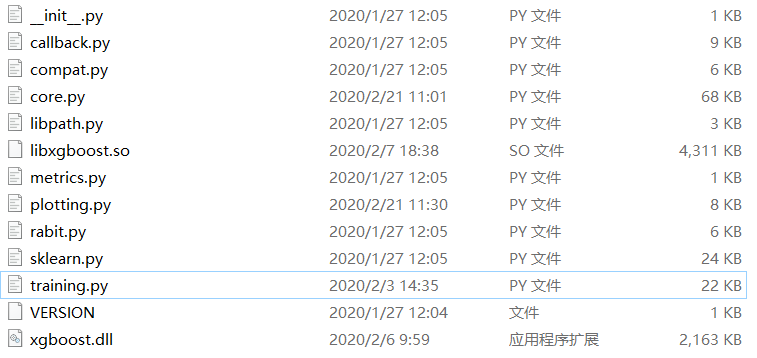
将生成的库文件复制到GBST\_package中，与\_\_init\_\_.py等文件同一目录下即可。

1. Python package源码导览

版本：python 3.5+

需要的库：numpy，scipy，sklearn（可选），matplotlib（可选）

Python方面的所有源码如下所示。



**callback.py**：回调函数。使用这里面的函数来跟踪训练过程，输出过程中的loss，或者进行early-stop等。

**compat.py**：在这里import的是一些可选的库，例如sklearn。若未安装sklearn，依然可以使用train.py当中的函数；但与sklearn相关的如Estimator等部分会变成空的object。

**core.py**：定义了gbst库中的核心对象Booster。一个booster object当中包含了训练、预测eval、模型的保存和读取等等方法，其具体实现则是去调用libxgboost。某种意义上来说，Booster对象的作用是作为libxgboost与python之间的接口。

此外，core.py中还定义了gbst/xgboost所使用的数据格式（DataFrame）的初始化和存储，例如从numpy array构建DataFrame的函数。

**libpath.py**：在原本的xgboost python package中，该部分会搜寻一些系统中常见的路径（例如path），来定位xgboost.dll的位置。gbst package中，为了防止与其它地方安装的xgboost库起冲突，我们只会搜寻当前目录下的dll/so文件。

**metrics.py**：在python中实现的survivalauc实例。其计算的方式和src中定义的metric相同，输出的值也几乎完全一样。

由于在python当中实现一个新的metric要比在src中方便得多（虽然速度会略慢），因此这里的metric作为example进行保留。

**plotting.py**：对某个Booster object或gbstModel(sklearn) object进行特征重要性可视化的库。直接调用plot\_importance()即会返回matplotlib axes。

**rabit.py**：定义了python与rabit之间的接口。训练、存取、callback等等方面都需要调用rabit，把接口部分单独写出来是为了方便起见。

**sklearn.py**：定义了Booster object与sklearn之间的接口。gbstModel这一对象是我们在使用sklearn调用gbst库时所主要使用的模型。

**training.py**：包含了通常的training routine，和cross-validation相关的routine。输入为模型相关的超参数（例如学习率、训练轮数等等）和数据，返回一个训练好的booster对象。

安装python package：

python setup.py install