xgboost: eXtreme Gradient Boosting

Tianqi Chen, Tong He

Package Version: 0.71.1

May 15, 2018

1 Introduction

This is an introductory document of using the `xgboost` package in R.

`xgboost` is short for eXtreme Gradient Boosting package. It is an efficient and scalable implementation of gradient boosting framework by (Friedman, 2001) (Friedman *et al.*, 2000). The package includes efficient linear model solver and tree learning algorithm. It supports various objective functions, including regression, classification and ranking. The package is made to be extendible, so that users are also allowed to define their own objectives easily. It has several features:

1. Speed: `xgboost` can automatically do parallel computation on Windows and Linux, with openmp. It is generally over 10 times faster than `gbm`.
2. Input Type: `xgboost` takes several types of input data:
   - Dense Matrix: R's dense matrix, i.e. `matrix`
   - Sparse Matrix: R's sparse matrix `Matrix::dgCMatrix`
   - Data File: Local data files
   - xgb.DMatrix: `xgboost`'s own class. Recommended.
3. Sparsity: `xgboost` accepts sparse input for both tree booster and linear booster, and is optimized for sparse input.
4. Customization: `xgboost` supports customized objective function and evaluation function
5. Performance: `xgboost` has better performance on several different datasets.

2 Example with Mushroom data

In this section, we will illustrate some common usage of `xgboost`. The Mushroom data is cited from UCI Machine Learning Repository. (Bache and Lichman, 2013)

```
library(xgboost)
data(agaricus.train, package='xgboost')
data(agaricus.test, package='xgboost')
train <- agaricus.train
test <- agaricus.test
bst <- xgboost(data = train$data, label = train$label, max_depth = 2, eta = 1,
               nrounds = 2, objective = "binary:logistic")

## [1] train-error:0.046522
## [2] train-error:0.022263

xgb.save(bst, 'model.save')

## [1] TRUE

bst = xgb.load('model.save')
pred <- predict(bst, test$data)
```

`xgboost` is the main function to train a `Booster`, i.e. a model. `predict` does prediction on the model.

Here we can save the model to a binary local file, and load it when needed. We can't inspect the trees inside. However we have another function to save the model in plain text.

```
xgb.dump(bst, 'model.dump')

## [1] TRUE
```

The output looks like

```
booster[0]:
0:[f28<1.00001] yes=1,no=2,missing=2
  1:[f108<1.00001] yes=3,no=4,missing=4
    3:leaf=1.85965
    4:leaf=-1.94071
  2:[f55<1.00001] yes=5,no=6,missing=6
    5:leaf=-1.70044
    6:leaf=1.71218
booster[1]:
0:[f59<1.00001] yes=1,no=2,missing=2
  1:leaf=-6.23624
  2:[f28<1.00001] yes=3,no=4,missing=4
    3:leaf=-0.96853
    4:leaf=0.784718
```

It is important to know `xgboost`'s own data type: `xgb.DMatrix`. It speeds up `xgboost`, and is needed for advanced features such as training from initial prediction value, weighted training instance.

We can use `xgb.DMatrix` to construct an `xgb.DMatrix` object:

```
dtrain <- xgb.DMatrix(train$data, label = train$label)
class(dtrain)

## [1] "xgb.DMatrix"

head(getinfo(dtrain,'label'))

## [1] 1 0 0 1 0 0
```

We can also save the matrix to a binary file. Then load it simply with `xgb.DMatrix`

```
xgb.DMatrix.save(dtrain, 'xgb.DMatrix')

## [1] TRUE

dtrain = xgb.DMatrix('xgb.DMatrix')

## [23:02:36] 6513x126 matrix with 143286 entries loaded from xgb.DMatrix
```

3 Advanced Examples

The function `xgboost` is a simple function with less parameter, in order to be R-friendly. The core training function is wrapped in `xgb.train`. It is more flexible than `xgboost`, but it requires users to read the document a bit more carefully.

`xgb.train` only accept a `xgb.DMatrix` object as its input, while it supports advanced features as custom objective and evaluation functions.

```
logregobj <- function(preds, dtrain) {
   labels <- getinfo(dtrain, "label")
   preds <- 1/(1 + exp(-preds))
   grad <- preds - labels
   hess <- preds * (1 - preds)
   return(list(grad = grad, hess = hess))
}

evalerror <- function(preds, dtrain) {
  labels <- getinfo(dtrain, "label")
  err <- sqrt(mean((preds-labels)^2))
```

```
  return(list(metric = "MSE", value = err))
}

dtest <- xgb.DMatrix(test$data, label = test$label)
watchlist <- list(eval = dtest, train = dtrain)
param <- list(max_depth = 2, eta = 1, silent = 1)

bst <- xgb.train(param, dtrain, nrounds = 2, watchlist, logregobj, evalerror, maximize = FALSE)

## [1] eval-MSE:1.592293 train-MSE:1.595967
## [2] eval-MSE:2.405194 train-MSE:2.409772
```

The gradient and second order gradient is required for the output of customized objective function.

We also have `slice` for row extraction. It is useful in cross-validation.

For a walkthrough demo, please see `R-package/demo/` for further details.

4 The Higgs Boson competition

We have made a demo for the Higgs Boson Machine Learning Challenge.

Here are the instructions to make a submission

1. Download the datasets and extract them to `data/`.
2. Run scripts under `xgboost/demo/kaggle-higgs/`: higgs-train.R and higgs-pred.R. The computation will take less than a minute on Intel i7.
3. Go to the submission page and submit your result.

We provide a script to compare the time cost on the higgs dataset with `gbm` and `xgboost`. The training set contains 350000 records and 30 features.

`xgboost` can automatically do parallel computation. On a machine with Intel i7-4700MQ and 24GB memories, we found that `xgboost` costs about 35 seconds, which is about 20 times faster than `gbm`. When we limited `xgboost` to use only one thread, it was still about two times faster than `gbm`.

Meanwhile, the result from `xgboost` reaches 3.60@AMS with a single model. This results stands in the top 30% of the competition.

References

Bache K, Lichman M (2013). "UCI Machine Learning Repository." URL `http://archive.ics.uci.edu/ml`.

Friedman J, Hastie T, Tibshirani R, *et al.* (2000). "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)." *The annals of statistics*, **28**(2), 337–407.

Friedman JH (2001). "Greedy function approximation: a gradient boosting machine." *Annals of Statistics*, pp. 1189–1232.