ML Exercises - Gradient Boosting

Jan-Philipp Kolb

04 Juni, 2019

eXtremely Boost your machine learning Exercises (Part-1)

- eXtreme Gradient Boosting is a machine learning model which became really popular few years ago after winning several Kaggle competitions.
- It is very powerful algorithm that use an ensemble of weak learners to obtain a strong learner.
- Its R implementation is available in xgboost package and it is really worth including into anyone's machine learning portfolio.

Boosting Exercises - first part

Exercise 1

Load `xgboost` library and download German Credit dataset. Your goal will be to predict creditability (the first column in the dataset).

Exercise 2

Convert columns `c(2,4,5,7,8,9,10,11,12,13,15,16,17,18,19,20)` to factors and then encode them as dummy variables. HINT: use the command `model.matrix()`

Exercise 3

Split data into training and test set 700:300. Create `xgb.DMatrix` for both sets with Creditability as label.

Boosting Exercises - second part

Exercise 4

Train `xgboost` with logistic objective and 30 rounds of training and maximal depth 2.

Exercise 5

To check model performance calculate test set classification error.

Exercise 6

Plot predictors importance.

Boosting Exercises - third part

Exercise 7

Use `xgb.train()` instead of `xgboost()` to add both train and test sets as a watchlist. Train model with same parameters, but 100 rounds to see how it performs during training.

Exercise 8

Train model again adding AUC and Log Loss as evaluation metrices.

Exercise 9

Plot how AUC and Log Loss for train and test sets was changing during training process. Use plotting function/library of your choice.

Exercise 10

Check how setting parameter eta to 0.01 influences the AUC and Log Loss curves. image_pdf

Solutions: boosting exercises

Solution Exercise 1 - import dataset

```
library(xgboost)

url <- "http://freakonometrics.free.fr/german_credit.csv"
credit <- read.csv(url, header = TRUE, sep = ",")

head(credit)
```

```
##   Creditability Account.Balance Duration.of.Credit..month.
## 1             1               1                         18
## 2             1               1                          9
## 3             1               2                         12
## 4             1               1                         12
## 5             1               1                         12
## 6             1               1                         10
##   Payment.Status.of.Previous.Credit Purpose Credit.Amount
## 1                                 4       2          1049
## 2                                 4       0          2799
## 3                                 2       9           841
```

Solutions boosting exercises - first part

Solution Exercise 2 - convert columns

```
factor_columns <- c(2,4,5,7,8,9,10,11,12,13,15,16,17,18,19,20)
for(i in factor_columns) credit[,i] <- as.factor(credit[,i])
X <- model.matrix(~ . - Creditability, data=credit)
```

Solution Exercise 3

```
inTraining <- sample(1:nrow(credit),size=700)
dtrain <- xgboost::xgb.DMatrix(X[inTraining,],
                               label=credit$Creditability[inTraining])
dtest <- xgboost::xgb.DMatrix(X[-inTraining,],
                              label=credit$Creditability[-inTraining])
```

Solutions boosting exercises - second part

Solution Exercise 4 - train xgboost model

```
model <- xgboost(data = dtrain,
                 max_depth = 2,
                 nrounds = 30,
                 objective = "binary:logistic")
```

```
## [1]  train-error:0.284286
## [2]  train-error:0.261429
## [3]  train-error:0.261429
## [4]  train-error:0.297143
## [5]  train-error:0.257143
## [6]  train-error:0.257143
## [7]  train-error:0.250000
## [8]  train-error:0.251429
## [9]  train-error:0.248571
## [10] train-error:0.252857
## [11] train-error:0.241429
## [12] train-error:0.225714
```

Solutions boosting exercises - third part

Solution Exercise 5

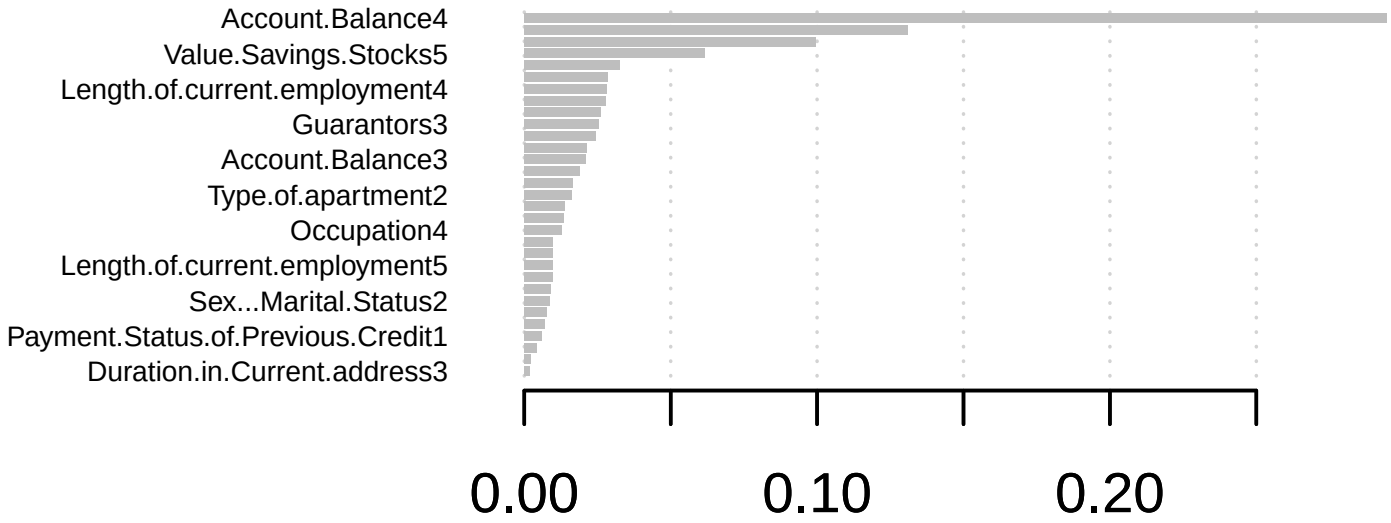
```
err<-mean(round(predict(model,dtest))!=getinfo(dtest,'label'))
print(paste("test-error=", err))
```

```
## [1] "test-error= 0.253333333333333"
```

Solution Exercise 6

```
importance.matrix <- xgb.importance(model = model,
                                    feature_names = colnames(X))
xgb.plot.importance(importance.matrix)
```

Importance plot



Solution Exercise 7

```
model_watchlist <- xgb.train(data = dtrain,
                             max_depth = 2,nrounds = 100,
                             objective = "binary:logistic",
                             watchlist = list(train=dtrain,
                                              test=dtest))
```

```
## [1]  train-error:0.284286    test-error:0.290000
## [2]  train-error:0.261429    test-error:0.286667
## [3]  train-error:0.261429    test-error:0.286667
## [4]  train-error:0.297143    test-error:0.306667
## [5]  train-error:0.257143    test-error:0.290000
## [6]  train-error:0.257143    test-error:0.290000
## [7]  train-error:0.250000    test-error:0.270000
## [8]  train-error:0.251429    test-error:0.263333
## [9]  train-error:0.248571    test-error:0.260000
## [10] train-error:0.252857    test-error:0.266667
## [11] train-error:0.241429    test-error:0.260000
## [12] train-error:0.225714    test-error:0.256667
```

Solution Exercise 8

```
model_auc<-xgb.train(data = dtrain,max_depth = 2,
        nrounds = 100,objective = "binary:logistic",
        watchlist = list(train=dtrain,test=dtest),
        eval_metric = 'auc',eval_metric = 'logloss')
```

Output model_auc

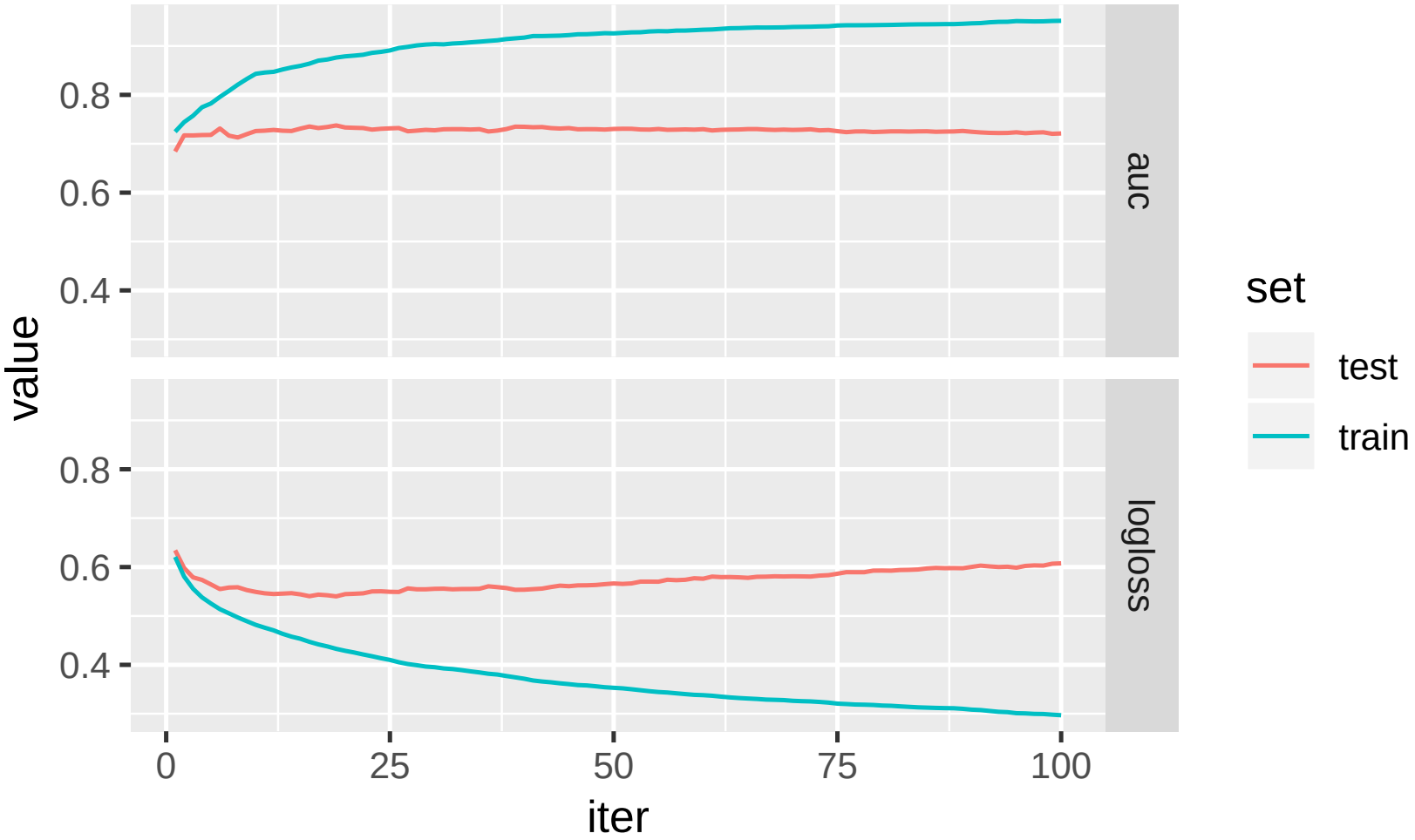
```
model_auc
```

```
## ##### xgb.Booster
## Handle is invalid! Suggest using xgb.Booster.complete
## raw: 39.3 Kb
## call:
##   xgb.train(data = dtrain, nrounds = 100, watchlist = list(tr
##     test = dtest), max_depth = 2, objective = "binary:logisti
##     eval_metric = "auc", eval_metric = "logloss")
## params (as set within xgb.train):
##   max_depth = "2", objective = "binary:logistic", eval_metric
## callbacks:
##   cb.print.evaluation(period = print_every_n)
##   cb.evaluation.log()
## # of features: 55
## niter: 100
## nfeatures : 55
## evaluation_log:
```

Solution Exercise 9

```
library(tidyverse)
model_auc$evaluation_log %>%
  gather(metric, value, -iter) %>%
  separate(metric, c('set','metric')) %>%
  ggplot(aes(iter, value, color = set)) +
  geom_line() +
  facet_grid(metric~.)
```

Evaluation plot



Solution Exercise 10

```
model_eta<-xgb.train(data=dtrain,max_depth = 2,eta = 0.05,
          nrounds = 100,objective = "binary:logistic",
          watchlist = list(train=dtrain, test=dtest),
          eval_metric = 'auc',eval_metric = 'logloss')
```

Output of model eta

```
model_eta
```

```
## ##### xgb.Booster
## Handle is invalid! Suggest using xgb.Booster.complete
## raw: 39.9 Kb
## call:
##   xgb.train(data = dtrain, nrounds = 100, watchlist = list(tr
##     test = dtest), max_depth = 2, eta = 0.05, objective = "bi
##     eval_metric = "auc", eval_metric = "logloss")
## params (as set within xgb.train):
##   max_depth = "2", eta = "0.05", objective = "binary:logistic
## callbacks:
##   cb.print.evaluation(period = print_every_n)
##   cb.evaluation.log()
## # of features: 55
## niter: 100
## nfeatures : 55
## evaluation_log:
```