ML Exercises - Gradient Boosting

Jan-Philipp Kolb

30 Mai, 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 in this tutorial 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 ‘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

Exercise 1

```r
library(xgboost)
```

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

Solutions boosting exercises - first part

Solution Exercise 2

```
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 <- xgb.DMatrix(X[inTraining,],
                      label=credit$Creditability[inTraining])
dtest <- xgb.DMatrix(X[-inTraining,],
                     label=credit$Creditability[-inTraining])
```

Solutions boosting exercises - second part

Exercise 4

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

```
## [1]  train-error:0.284286
## [2]  train-error:0.300000
## [3]  train-error:0.288571
## [4]  train-error:0.274286
## [5]  train-error:0.265714
## [6]  train-error:0.260000
## [7]  train-error:0.261429
## [8]  train-error:0.264286
## [9]  train-error:0.254286
## [10] train-error:0.250000
## [11] train-error:0.248571
## [12] train-error:0.248571
```

Solutions boosting exercises - third part

Exercise 5

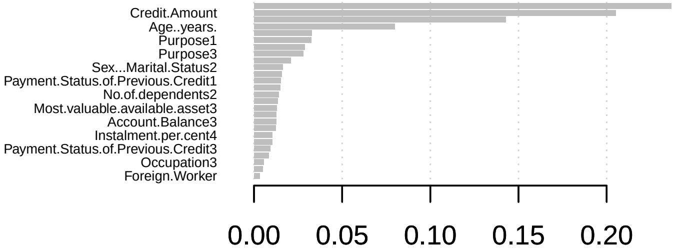
```
err<-mean(round(predict(model,dtest))!=getinfo(dtest,'label'))
print(paste("test-error=", err))
```

```
## [1] "test-error= 0.216666666666667"
```

Exercise 6

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

Importance plot



Exercise 7

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

```
## [1]  train-error:0.284286    test-error:0.276667
## [2]  train-error:0.300000    test-error:0.300000
## [3]  train-error:0.288571    test-error:0.303333
## [4]  train-error:0.274286    test-error:0.263333
## [5]  train-error:0.265714    test-error:0.260000
## [6]  train-error:0.260000    test-error:0.260000
## [7]  train-error:0.261429    test-error:0.260000
## [8]  train-error:0.264286    test-error:0.266667
## [9]  train-error:0.254286    test-error:0.263333
## [10] train-error:0.250000    test-error:0.266667
## [11] train-error:0.248571    test-error:0.263333
## [12] train-error:0.248571    test-error:0.260000
```

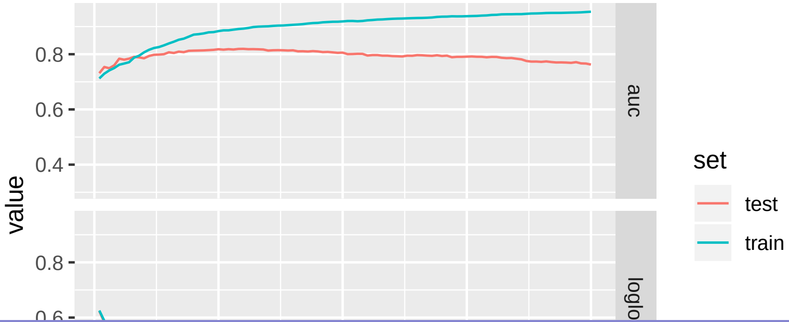
Solution Exercise 8

```
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')
```

```
## [1]  train-auc:0.712245  train-logloss:0.625511  test-auc:0.7
## [2]  train-auc:0.729461  train-logloss:0.588538  test-auc:0.7
## [3]  train-auc:0.741317  train-logloss:0.565714  test-auc:0.7
## [4]  train-auc:0.749869  train-logloss:0.551041  test-auc:0.7
## [5]  train-auc:0.761871  train-logloss:0.539573  test-auc:0.7
## [6]  train-auc:0.766195  train-logloss:0.531495  test-auc:0.7
## [7]  train-auc:0.771035  train-logloss:0.524274  test-auc:0.7
## [8]  train-auc:0.787585  train-logloss:0.515915  test-auc:0.7
## [9]  train-auc:0.794456  train-logloss:0.509156  test-auc:0.7
## [10] train-auc:0.806589  train-logloss:0.501230  test-auc:0.7
```

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~.)
```


value
auc
loglo
set
test
train