Custom Xgboost reg : squared-log-error

kipédène COULIBALY

19/04/2023

Analytical formula

The analytical formula of Mean Squared Log Error (MSLE) is :

$$MSLE = \frac{1}{n}\sum_{i=1}^{N}\Big[log(Y_i+1) - log(\hat{Y}_i+1)\Big]^2$$

- **Objective function** :

$$f(pred, label) = \frac{1}{2}\Big[log(pred+1) - log(label+1)\Big]^2$$
$$Grad = \frac{1}{(pred+1)}\Big[log(pred+1) - log(label+1)\Big]$$
$$Hess = \frac{1}{(pred+1)^2}\Big[1 - log(pred+1) + log(label+1)\Big]$$

NB: With this function, all input labels are required to be greater than -1.

- **Evaluation metrics** :

Here we use two evaluation metrics: first, the Root Mean Square Log Error (RMSLE) which is simply the square root of the MSLE :

$$RMSLE = \sqrt{\frac{1}{n}\sum_{i=1}^{N}\Big[log(Y_i+1) - log(\hat{Y}_i+1)\Big]^2}$$

Then for a robust verification we use Mean Absolute Error (MAE) without implementing it (you should be able to do it without any problem) :

$$MAE = \frac{1}{n}\sum_{i=1}^{N}|Y_i - \hat{Y}_i|$$

Implementation with R

```r
library(ISLR)
library(xgboost)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
## v ggplot2 3.4.2     v purrr   0.3.4
## v tibble  3.1.8     v dplyr   1.0.8
## v tidyr   1.2.1     v stringr 1.5.0
## v readr   2.1.3     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x dplyr::slice()  masks xgboost::slice()
```

```
library(Metrics)

# Data #
df = ISLR::Hitters %>% select(Salary, AtBat, Hits, HmRun, Runs, RBI, Walks,
                              Years, CAtBat, CHits, CHmRun, CRuns, CRBI, CWalks,
                              PutOuts, Assists, Errors)
df = df[complete.cases(df),]
train = df[1:150,]
test = df[151:nrow(df),]

# XGBoost Matrix
dtrain <- xgb.DMatrix(data = as.matrix(train[,-1]),label = as.matrix(train[,1]))
dtest <- xgb.DMatrix(data = as.matrix(test[,-1]),label = as.matrix(test[,1]))
watchlist <- list(eval = dtest)

# Custom objective function (squared log error)
myobjective <- function(preds, dtrain) {
  labels <- getinfo(dtrain, "label")
  grad <- 1/(preds + 1)*(log(preds + 1) - log(labels + 1))
  hess <- 1/(preds + 1)^2*(1 - log(preds + 1) + log(labels + 1))
  return(list(grad = grad, hess = hess))
}

# Custom Metric
evalerror <- function(preds, dtrain) {
  labels <- getinfo(dtrain, "label")
  err <- (log(preds + 1) - log(labels + 1))^2
  return(list(metric = "MyError", value = sqrt(mean(err))))
}

# Custom Model
param1 <- list(booster = 'gbtree', learning_rate = 0.1, objective = myobjective,
               eval_metric = evalerror, set.seed = 2020)

xgb1 <- xgb.train(params = param1, data = dtrain, nrounds = 500, watchlist,
                  maximize = FALSE, early_stopping_rounds = 5)
```

```
## [21:27:49] WARNING: amalgamation/../src/learner.cc:627:
## Parameters: { "set_seed" } might not be used.
##
##   This could be a false alarm, with some parameters getting used by language bindings but
##   then being mistakenly passed down to XGBoost core, or some parameter actually being used
##   but getting flagged wrongly here. Please open an issue if you find any such cases.
##
##
```

```
## [1]  eval-MyError:5.468007
## Will train until eval_MyError hasn't improved in 5 rounds.
##
## [2]  eval-MyError:5.387989
## [3]  eval-MyError:5.308218
## [4]  eval-MyError:5.228705
## [5]  eval-MyError:5.149464
## [6]  eval-MyError:5.070510
## [7]  eval-MyError:4.991862
## [8]  eval-MyError:4.913538
## [9]  eval-MyError:4.835561
## [10] eval-MyError:4.757955
## [11] eval-MyError:4.680749
## [12] eval-MyError:4.603975
## [13] eval-MyError:4.527668
## [14] eval-MyError:4.451869
## [15] eval-MyError:4.376624
## [16] eval-MyError:4.301984
## [17] eval-MyError:4.228007
## [18] eval-MyError:4.154755
## [19] eval-MyError:4.082299
## [20] eval-MyError:4.010717
## [21] eval-MyError:3.940091
## [22] eval-MyError:3.870512
## [23] eval-MyError:3.802074
## [24] eval-MyError:3.734877
## [25] eval-MyError:3.669024
## [26] eval-MyError:3.604616
## [27] eval-MyError:3.541754
## [28] eval-MyError:3.480534
## [29] eval-MyError:3.421045
## [30] eval-MyError:3.363365
## [31] eval-MyError:3.307560
## [32] eval-MyError:3.253683
## [33] eval-MyError:3.201767
## [34] eval-MyError:3.151831
## [35] eval-MyError:3.103878
## [36] eval-MyError:3.057893
## [37] eval-MyError:3.013847
## [38] eval-MyError:2.971699
## [39] eval-MyError:2.931396
## [40] eval-MyError:2.892876
## [41] eval-MyError:2.856073
## [42] eval-MyError:2.820913
## [43] eval-MyError:2.820913
## [44] eval-MyError:2.820913
## [45] eval-MyError:2.820913
## [46] eval-MyError:2.820913
## [47] eval-MyError:2.820913
## Stopping. Best iteration:
## [42] eval-MyError:2.820913
```

```r
pred1 = predict(xgb1, dtest)
mae1 = mae(test$Salary, pred1)
```

```
## Normal Model
param2 <- list(booster = 'gbtree', learning_rate = 0.1,
               objective = "reg:squaredlogerror", set.seed = 2020)

xgb2 <- xgb.train(params = param2, data = dtrain, nrounds = 500, watchlist,
                  maximize = FALSE, early_stopping_rounds = 5)
```

```
## [21:27:49] WARNING: amalgamation/../src/learner.cc:627:
## Parameters: { "set_seed" } might not be used.
##
##   This could be a false alarm, with some parameters getting used by language bindings but
##   then being mistakenly passed down to XGBoost core, or some parameter actually being used
##   but getting flagged wrongly here. Please open an issue if you find any such cases.
##
##
## [1]  eval-rmsle:5.468007
## Will train until eval_rmsle hasn't improved in 5 rounds.
##
## [2]  eval-rmsle:5.387990
## [3]  eval-rmsle:5.308218
## [4]  eval-rmsle:5.228705
## [5]  eval-rmsle:5.149464
## [6]  eval-rmsle:5.070510
## [7]  eval-rmsle:4.991862
## [8]  eval-rmsle:4.913538
## [9]  eval-rmsle:4.835561
## [10] eval-rmsle:4.757955
## [11] eval-rmsle:4.680749
## [12] eval-rmsle:4.603975
## [13] eval-rmsle:4.527668
## [14] eval-rmsle:4.451869
## [15] eval-rmsle:4.376624
## [16] eval-rmsle:4.301984
## [17] eval-rmsle:4.228007
## [18] eval-rmsle:4.154755
## [19] eval-rmsle:4.082299
## [20] eval-rmsle:4.010717
## [21] eval-rmsle:3.940091
## [22] eval-rmsle:3.870512
## [23] eval-rmsle:3.802074
## [24] eval-rmsle:3.734878
## [25] eval-rmsle:3.669024
## [26] eval-rmsle:3.604616
## [27] eval-rmsle:3.541754
## [28] eval-rmsle:3.480534
## [29] eval-rmsle:3.421045
## [30] eval-rmsle:3.363365
## [31] eval-rmsle:3.307560
## [32] eval-rmsle:3.253683
## [33] eval-rmsle:3.201767
## [34] eval-rmsle:3.151831
## [35] eval-rmsle:3.103878
## [36] eval-rmsle:3.057893
## [37] eval-rmsle:3.013847
## [38] eval-rmsle:2.971699
## [39] eval-rmsle:2.931396
## [40] eval-rmsle:2.892876
## [41] eval-rmsle:2.856073
## [42] eval-rmsle:2.820913
## [43] eval-rmsle:2.820913
## [44] eval-rmsle:2.820913
## [45] eval-rmsle:2.820913
## [46] eval-rmsle:2.820913
## [47] eval-rmsle:2.820913
## Stopping. Best iteration:
## [42] eval-rmsle:2.820913
```

```
pred2 = predict(xgb2, dtest)
mae2 = mae(test$Salary, pred2)

# comparaison
print(list(xgb1$evaluation_log$eval_MyError[xgb1$best_iteration],
           xgb2$evaluation_log$eval_rmsle[xgb2$best_iteration]))
```

```
## [[1]]
## [1] 2.820913
##
## [[2]]
## [1] 2.820913
```

```
print(list(mae1, mae2))
```

```
## [[1]]
## [1] 474.5879
##
## [[2]]
## [1] 474.5879
```