Application of XGBOOST for classification task in the enhancer prediction data.

Shaurya Jauhari (Email: shauryajauhari@gzhmu.edu.cn)

2019-11-05

Contents

List of Figures

Introduction

Extreme Gradient Boosting (XGBOOST) is another method for exploring the gradient boosting technique, but is reportedly faster than other methods. The underlying dogma stays the same that the sequential boosting is performed.

Package Installation

The package in question is "xgboost".

```
install.packages("xgboost", dependencies = TRUE, verbose = TRUE,
                 repos = "https://mirrors.tuna.tsinghua.edu.cn/CRAN/")
```

```
## Installing package into '/Users/mei/Library/R/3.6/library'
## (as 'lib' is unspecified)
```

```
##
## The downloaded binary packages are in
##  /var/folders/hm/c3_fjypn62v5xh5b5ygv267m0000gn/T//Rtmpi3s1sv/downloaded_packages
```

Dataset

We'll be employing the enhancer prediction dataset to exemplify an application of gradient boosting algorithm. The details for the dataset could be reached here.

```
epdata <- readRDS("../Machine_Learning_Deep_Learning/data/ep_data_sample.rds")
rownames(epdata) <- c()


set.seed(001)
data_partition <- sample(3, nrow(epdata), replace = TRUE, prob = c(0.64,0.16,0.2))
train <- epdata[data_partition==1,]
test <- epdata[data_partition==2,]
holdout <- epdata[data_partition==3,]
```

```r
head(train)
```

```
##    peaks_h3k27ac peaks_h3k4me3 peaks_h3k4me2 peaks_h3k4me1 class
## 1       0.734144       0.00000      0.000000             0     1
## 2       0.000000       0.00000      0.000000             0     0
## 3       1.468290       1.13595      0.865892             0     1
## 5       0.000000       1.13595      0.000000             0     1
## 9       3.670720       0.00000      0.865892             0     1
## 10      0.000000       0.00000      0.000000             0     1
```

XGBOOST works on numeric vectors. The class labels and feature data are so available. The feature data has to be compiled as a data matrix though.

One-Hot Encoding

```r
library(keras)
train_labels <- to_categorical(train$class)
test_labels <- to_categorical(test$class)
```

Implementation

```r
library(xgboost)
xgbmodel1 <- xgboost(data = as.matrix(train[,1:4]),
                     label = train$class,
                     max_depth = 2,
                     eta = 1,
                     nthread = 2,
                     nrounds = 2,
                     objective = "binary:logistic") # binary classification , i.e. enhancer or non-enha
```

```
## [1]  train-error:0.361729
## [2]  train-error:0.361729
```

It is usual to have data with more zero entries. The sparse data format of a matrix relieves the memory for storing such cells, thereby reducing the data storage size. However, that shouldn't make any difference in the results.

XGBOOST also provides a format *DMatrix* to convert a data table into a matrix, whether sparse or dense. It is recommended in use as well.

```r
train_data <- xgb.DMatrix(data = as.matrix(train[,1:4]), label= train$class)
test_data <- xgb.DMatrix(data = as.matrix(test[,1:4]), label= test$class)


xgbcv1 <- xgb.cv( data = train_data,
                  nrounds = 100,
                  nfold = 5,
                  showsd = T,
                  stratified = T,
                  print.every.n = 10,
                  early.stop.round = 20,
                  maximize = F)
```

```
## Warning: 'print.every.n' is deprecated.
## Use 'print_every_n' instead.
```

```
## See help("Deprecated") and help("xgboost-deprecated").
```

```
## Warning: 'early.stop.round' is deprecated.
## Use 'early_stopping_rounds' instead.
## See help("Deprecated") and help("xgboost-deprecated").
```

```
## [1]  train-rmse:0.488575+0.000540    test-rmse:0.490261+0.001188
## Multiple eval metrics are present. Will use test_rmse for early stopping.
## Will train until test_rmse hasn't improved in 20 rounds.
##
## [11] train-rmse:0.471381+0.001049    test-rmse:0.483429+0.003899
## [21] train-rmse:0.467721+0.001259    test-rmse:0.485998+0.004805
## Stopping. Best iteration:
## [6]  train-rmse:0.474629+0.000586    test-rmse:0.482373+0.003515
```

Accuracy and Prediction

```
cat("The accuracy is", 100 - (min(xgbcv1$evaluation_log$test_rmse_mean)*100), ". Since we are using the
```

```
## The accuracy is 51.76274 . Since we are using the corss-validation scheme, the test/validation data
```

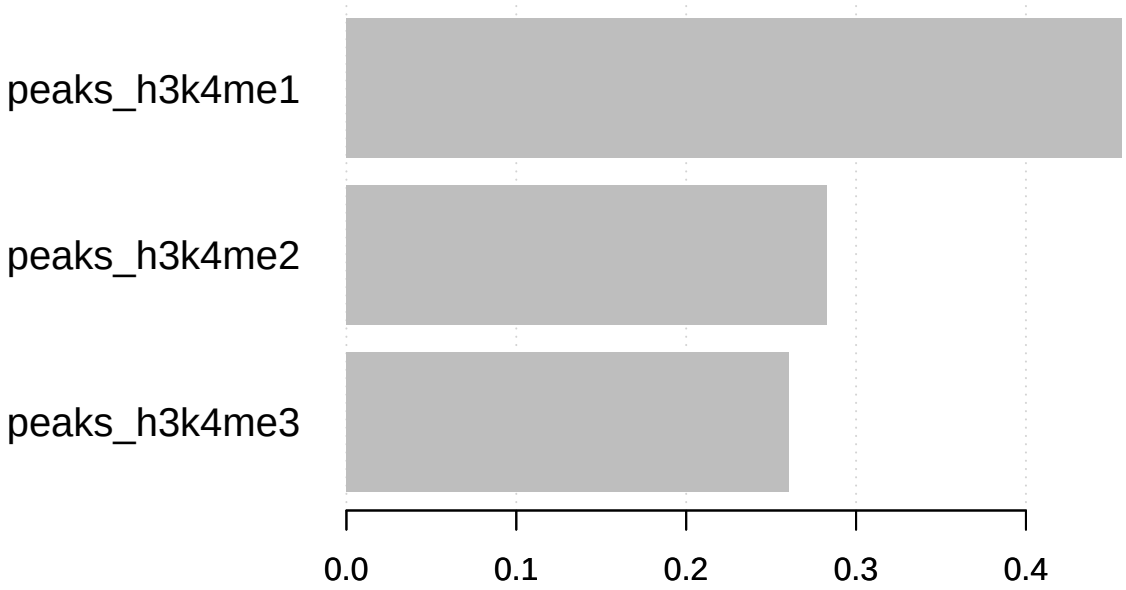
```
#model prediction
xgbpred <- predict (xgbmodel1,test_data)
xgbpred <- ifelse (xgbpred > 0.5,1,0) # 1:Non-Enhancers, 0: Enhancers

table(xgbpred)
```

```
## xgbpred
##    0    1
##    1 1667
```

```
var_imp <- xgb.importance (feature_names = colnames(train[,1:4]),model = xgbmodel1)
var_imp
```

```
##          Feature      Gain     Cover Frequency
## 1: peaks_h3k4me1 0.4565521 0.4812320       0.6
## 2: peaks_h3k4me2 0.2828799 0.2592011       0.2
## 3: peaks_h3k4me3 0.2605681 0.2595669       0.2
```

```
xgb.plot.importance (importance_matrix = var_imp)
```


peaks_h3k4me1
peaks_h3k4me2
peaks_h3k4me3
0.0
0.1
0.2
0.3
0.4