Ensemble techniques

Yixin Sun,

10-23-2022

load the data

```
adult <- read.csv("C:/Users/Yixin Sun/Documents/Assignment3/adult.csv", header = T)
str(adult)
```

```
## 'data.frame':    32561 obs. of  15 variables:
##  $ age           : int  39 50 38 53 28 37 49 52 31 42 ...
##  $ workclass     : chr  " State-gov" " Self-emp-not-inc" " Private" " Private" ...
##  $ fnlwgt        : int  77516 83311 215646 234721 338409 284582 160187 209642 45781 159449
...
##  $ education     : chr  " Bachelors" " Bachelors" " HS-grad" " 11th" ...
##  $ education.num : int  13 13 9 7 13 14 5 9 14 13 ...
##  $ marital.status: chr  " Never-married" " Married-civ-spouse" " Divorced" " Married-civ-spou
se" ...
##  $ occupation    : chr  " Adm-clerical" " Exec-managerial" " Handlers-cleaners" " Handlers-cl
eaners" ...
##  $ relationship  : chr  " Not-in-family" " Husband" " Not-in-family" " Husband" ...
##  $ race          : chr  " White" " White" " White" " Black" ...
##  $ sex           : chr  " Male" " Male" " Male" " Male" ...
##  $ capital.gain  : int  2174 0 0 0 0 0 0 0 14084 5178 ...
##  $ capital.loss  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ hoursperweek  : int  40 13 40 40 40 40 16 45 50 40 ...
##  $ nativecountry : chr  " United-States" " United-States" " United-States" " United-States"
...
##  $ salary        : chr  " <=50K" " <=50K" " <=50K" " <=50K" ...
```

data cleaning and divide into train and test

```
adult <- adult[,c(1,5,10,13,15)]
str(adult)
```

```
## 'data.frame':    32561 obs. of  5 variables:
##  $ age          : int  39 50 38 53 28 37 49 52 31 42 ...
##  $ education.num: int  13 13 9 7 13 14 5 9 14 13 ...
##  $ sex          : chr  " Male" " Male" " Male" " Male" ...
##  $ hoursperweek : int  40 13 40 40 40 40 16 45 50 40 ...
##  $ salary       : chr  " <=50K" " <=50K" " <=50K" " <=50K" ...
```

```
set.seed(1234)
adult$sex <- as.factor(adult$sex)
adult$salary <- as.factor(adult$salary)
i <- sample(1:nrow(adult), 0.8*nrow(adult), replace = F)
train <- adult[i,]
test <- adult[-i,]
str(train)
```

```
## 'data.frame':    26048 obs. of  5 variables:
##  $ age          : int  17 34 24 67 25 24 65 44 45 23 ...
##  $ education.num: int  7 9 10 9 8 13 5 10 9 9 ...
##  $ sex          : Factor w/ 2 levels " Female"," Male": 1 2 2 2 2 2 1 2 1 1 ...
##  $ hoursperweek : int  20 55 40 20 40 39 24 40 40 40 ...
##  $ salary       : Factor w/ 2 levels " <=50K"," >50K": 1 2 1 1 1 1 1 1 1 1 ...
```

Random forest

```
library(randomForest)
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
set.seed(1234)
rf <- randomForest(salary~., data=train, importance=TRUE)
rf
```

```
## 
## Call:
##  randomForest(formula = salary ~ ., data = train, importance = TRUE) 
##               Type of random forest: classification
##                     Number of trees: 500
## No. of variables tried at each split: 2
## 
##         OOB estimate of  error rate: 19.71%
## Confusion matrix:
##         <=50K  >50K class.error
##  <=50K  18280  1518  0.07667441
##  >50K    3617  2633  0.57872000
```

pred

```
library(mltools)
pred <- predict(rf, newdata=test, type="response")
acc_rf <- mean(pred==test$salary)
mcc_rf <- mcc(factor(pred), test$salary)
print(paste("accuracy=", acc_rf))
```

```
## [1] "accuracy= 0.808383233532934"
```

```
print(paste("mcc=", mcc_rf))
```

```
## [1] "mcc= 0.421657927572985"
```

Adabeg

```
library(adabag)
```

```
## Loading required package: rpart
```

```
## Loading required package: caret
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:randomForest':
##
##     margin
```

```
## Loading required package: lattice
```

```
## Loading required package: foreach
```

```
## Loading required package: doParallel
```

```
## Loading required package: iterators
```

```
## Loading required package: parallel
```

```
adab1 <- boosting(salary~., data=train, boos=TRUE, mfinal=20, coeflearn='Breiman')
summary(adab1)
```

```
##            Length Class   Mode
## formula        3  formula call
## trees         20  -none-  list
## weights       20  -none-  numeric
## votes      16582  -none-  numeric
## prob       16582  -none-  numeric
## class       8291  -none-  character
## importance    30  -none-  numeric
## terms          3  terms   call
## call           6  -none-  call
```

```
pred <- predict(adab1, newdata=test, type="response")
acc_adabag <- mean(pred$class==test$salary)
mcc_adabag <- mcc(factor(pred$class), test$salary)
print(paste("accuracy=", acc_adabag))
```

```
## [1] "accuracy= 0.74862518089725"
```

```
print(paste("mcc=", mcc_adabag))
```

```
## [1] "mcc= 0.558811377251421"
```

fastAdaboost

```
library(fastAdaboost)
set.seed(1234)
fadab <- adaboost(salary~., train, 10)
summary(fadab)
```

```
##                    Length Class   Mode
## formula             3     formula call
## trees              10     -none-  list
## weights            10     -none-  numeric
## classnames          2     -none-  character
## dependent_variable  1     -none-  character
## call                4     -none-  call
```

```
pred <- predict(fadab, newdata=test, type="response")

acc_fadab <- mean(pred$class==test$salary)
mcc_fadab <- mcc(pred$class, test$salary)
print(paste("accuracy=", acc_fadab))
```

```
## [1] "accuracy= 0.779517887302318"
```

```
print(paste("mcc=", mcc_fadab))
```

```
## [1] "mcc= 0.403089886563709"
```

XGBoost

```
library(xgboost)
train_label <- ifelse(train$salary==1, 1, 0)
train_matrix <- data.matrix(train[, -31])
model <- xgboost(data=train_matrix, label=train_label,
                 nrounds=100, objective='binary:logistic')
```

```
## [1]  train-logloss:0.437521
## [2]  train-logloss:0.296323
## [3]  train-logloss:0.207351
## [4]  train-logloss:0.147824
## [5]  train-logloss:0.106631
## [6]  train-logloss:0.077520
## [7]  train-logloss:0.056661
## [8]  train-logloss:0.041572
## [9]  train-logloss:0.030586
## [10] train-logloss:0.022548
## [11] train-logloss:0.016648
## [12] train-logloss:0.012307
## [13] train-logloss:0.009107
## [14] train-logloss:0.006745
## [15] train-logloss:0.004999
## [16] train-logloss:0.003709
## [17] train-logloss:0.002755
## [18] train-logloss:0.002048
## [19] train-logloss:0.001525
## [20] train-logloss:0.001138
## [21] train-logloss:0.000851
## [22] train-logloss:0.000639
## [23] train-logloss:0.000481
## [24] train-logloss:0.000364
## [25] train-logloss:0.000278
## [26] train-logloss:0.000213
## [27] train-logloss:0.000166
## [28] train-logloss:0.000130
## [29] train-logloss:0.000103
## [30] train-logloss:0.000083
## [31] train-logloss:0.000067
## [32] train-logloss:0.000056
## [33] train-logloss:0.000047
## [34] train-logloss:0.000040
## [35] train-logloss:0.000034
## [36] train-logloss:0.000034
## [37] train-logloss:0.000034
## [38] train-logloss:0.000034
## [39] train-logloss:0.000034
## [40] train-logloss:0.000034
## [41] train-logloss:0.000034
## [42] train-logloss:0.000034
## [43] train-logloss:0.000034
## [44] train-logloss:0.000034
## [45] train-logloss:0.000034
## [46] train-logloss:0.000034
## [47] train-logloss:0.000034
## [48] train-logloss:0.000034
## [49] train-logloss:0.000034
## [50] train-logloss:0.000034
## [51] train-logloss:0.000034
## [52] train-logloss:0.000034
## [53] train-logloss:0.000034
## [54] train-logloss:0.000034
## [55] train-logloss:0.000034
## [56] train-logloss:0.000034
## [57] train-logloss:0.000034
## [58] train-logloss:0.000034
## [59] train-logloss:0.000034
## [60] train-logloss:0.000034
## [61] train-logloss:0.000034
## [62] train-logloss:0.000034
## [63] train-logloss:0.000034
## [64] train-logloss:0.000034
## [65] train-logloss:0.000034
## [66] train-logloss:0.000034
## [67] train-logloss:0.000034
## [68] train-logloss:0.000034
## [69] train-logloss:0.000034
## [70] train-logloss:0.000034
## [71] train-logloss:0.000034
## [72] train-logloss:0.000034
## [73] train-logloss:0.000034
## [74] train-logloss:0.000034
## [75] train-logloss:0.000034
## [76] train-logloss:0.000034
## [77] train-logloss:0.000034
## [78] train-logloss:0.000034
## [79] train-logloss:0.000034
## [80] train-logloss:0.000034
## [81] train-logloss:0.000034
## [82] train-logloss:0.000034
## [83] train-logloss:0.000034
## [84] train-logloss:0.000034
## [85] train-logloss:0.000034
## [86] train-logloss:0.000034
## [87] train-logloss:0.000034
## [88] train-logloss:0.000034
## [89] train-logloss:0.000034
## [90] train-logloss:0.000034
## [91] train-logloss:0.000034
## [92] train-logloss:0.000034
## [93] train-logloss:0.000034
## [94] train-logloss:0.000034
## [95] train-logloss:0.000034
## [96] train-logloss:0.000034
## [97] train-logloss:0.000034
## [98] train-logloss:0.000034
## [99] train-logloss:0.000034
## [100]    train-logloss:0.000034
```

```
test_label <- ifelse(test$salary==1, 1, 0)
test_matrix <- data.matrix(test[, -31])
probs <- predict(model, test_matrix)
pred <- ifelse(probs>0.5, 1, 0)
acc_xg <- mean(pred==test_label)
mcc_xg <- mcc(pred, test_label)
print(paste("accuracy=", acc_xg))
```

```
## [1] "accuracy= 1"
```

```
print(paste("mcc=", mcc_xg))
```

```
## [1] "mcc= 0"
```

analysis

In terms of accuracy, XGBoost is better than random forest and better than adaboost.

In terms of the number of runs, XGBoost runs 100 times, random forest runs 500 times, and adaboost runs 10-20 times.

From the metric comparison, the mcc of random forest is 0.42, the mcc of XGBoost is 0 (may be caused by data overfitting), and the mcc of adaboost is 0.55