LASSO_NN

Cavalry

10/29/2020

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.4     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0

## Warning: package 'ggplot2' was built under R version 3.6.2

## Warning: package 'tibble' was built under R version 3.6.2

## Warning: package 'tidyr' was built under R version 3.6.2

## Warning: package 'readr' was built under R version 3.6.2

## Warning: package 'purrr' was built under R version 3.6.2

## Warning: package 'dplyr' was built under R version 3.6.2

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

```
library(readr)
library(caret)
```

```
## Loading required package: lattice

## Warning: package 'lattice' was built under R version 3.6.2

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(WeightedROC)
source("./LASSO_NN.R")
```

```
## Warning: package 'glmnet' was built under R version 3.6.2

## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
```

```
##     expand, pack, unpack
```

```
## Loaded glmnet 4.0-2
```

```
##
## Attaching package: 'neuralnet'
```

```
## The following object is masked from 'package:dplyr':
##
##     compute
```

```
## Warning: package 'e1071' was built under R version 3.6.2
```

```
datatrain = read_csv("../data/feature_train_balanced_data.csv")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_double()
## )
## i Use `spec()` for the full column specifications.
```

```
datatest = read_csv("../data/feature_test_data.csv")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_double()
## )
## i Use `spec()` for the full column specifications.
```

```
location = which(colnames(datatrain) == "HEALTH")
trainx = datatrain[,-location]
trainy = datatrain$HEALTH
testx = datatest[, -location]
testy = datatest$HEALTH


weight_test <- rep(NA, length(testy))
for (v in unique(testy)){
  weight_test[testy == v] = 0.5 * length(testy) / length(testy[testy == v])
}
```

LASSO method

```
begin <- Sys.time()
lasso <- lassoReg(trainx, trainy)
summary(lasso)
```

```
##           Length Class     Mode
## a0         1     -none-    numeric
## beta      50     dgCMatrix S4
## df         1     -none-    numeric
## dim        2     -none-    numeric
## lambda     1     -none-    numeric
## dev.ratio  1     -none-    numeric
## nulldev    1     -none-    numeric
## npasses    1     -none-    numeric
## jerr       1      -none-    numeric
## offset     1      -none-    logical
## call       7      -none-    call
## nobs       1      -none-    numeric
```

```
coef <- rbind("(intercept)" = lasso$a0, as.data.frame(as.matrix(lasso$beta))) %>%
  dplyr::arrange(desc(abs(s0))) %>% rename(Coef = s0)

coef
```

```
##                        Coef
## (intercept)   0.5382698332
## FS1_1        -0.1200926654
## fpl           0.0737881831
## FWB2_1       -0.0621141873
## FS2_1        -0.0581033770
## FS1_4        -0.0557699516
## SWB_2         0.0528259504
## FWB2_4       -0.0458344678
## FS1_2        -0.0424773711
## FWB1_1        0.0398995276
## ACT1_1        0.0390377814
## SWB_1         0.0313843533
## SUBKNOWL1     0.0307676681
## FS1_7        -0.0303872325
## FWB1_3       -0.0293556904
## FS1_5        -0.0257466739
## ASK1_1       -0.0228183295
## FSscore       0.0221663563
## FS1_3        -0.0201078089
## FWB1_2        0.0194234243
## SUBNUMERACY1  0.0190647316
## FINGOALS      0.0184990470
## FS2_3         0.0181511710
## MANAGE1_4    -0.0169994161
## PROPPLAN_3    0.0151785494
## PROPPLAN_1   -0.0142079674
## SWB_3         0.0127998433
## ASK1_2        0.0125702918
## FWB1_6       -0.0123974355
## MANAGE1_3    -0.0123160248
## FWBscore     -0.0120046555
## sample       -0.0105703310
## FWB2_3       -0.0098978392
## FWB1_4        0.0092783613
## GOALCONF      0.0091858050
## FS1_6         0.0090480505
## CHANGEABLE   -0.0087448510
## LMscore      -0.0085703329
## MANAGE1_2    -0.0075003280
## FWB2_2        0.0074860885
## MANAGE1_1     0.0070861582
## AUTOMATED_1  -0.0055887392
## ACT1_2        0.0043000740
## FWB1_5       -0.0029075900
## FS2_2        -0.0026010412
## SUBNUMERACY2 -0.0025140813
## SAVEHABIT     0.0020835779
## FRUGALITY    -0.0011212105
## AUTOMATED_2   0.0008323304
## PROPPLAN_4    0.0002998221
## PROPPLAN_2    0.0000000000
```

```
pred <- lassoPred(lasso, testx)
pred <- ifelse(pred > mean(pred), 1, 0)

cat("MSE is", mean((pred - testy)^2),". Accuracy is", mean(pred==testy),".")
```

```
## MSE is 0.3652283 . Accuracy is 0.6347717 .
```

```
confusionMatrix(factor(pred), factor(testy))$byClass
```

```
##          Sensitivity          Specificity       Pos Pred Value
##            0.8016194            0.6042899            0.2701228
##       Neg Pred Value            Precision               Recall
##            0.9434180            0.2701228            0.8016194
##                   F1           Prevalence       Detection Rate
##            0.4040816            0.1544715            0.1238274
## Detection Prevalence    Balanced Accuracy
##            0.4584115            0.7029547
```

```
tpr.fpr <- WeightedROC(pred, testy, weight_test)
auc.log <- WeightedAUC(tpr.fpr)

cat("AUC is", auc.log)
```

```
## AUC is 0.7029547
```

```
end <- Sys.time()
timedif <- end - begin
cat("Time for running","LASSO","is", timedif)
```

```
## Time for running LASSO is 0.531116
```

```
timedif
```

```
## Time difference of 0.531116 secs
```

SVM

SVM Tune

```
begin <- Sys.time()
source("./svm.R")
# Tune SVM
set.seed(2020)
opt.svm <- svm_tune(scale(as.matrix(trainx)), trainy)
bestgamma = opt.svm$best.parameters$gamma
bestcost = opt.svm$best.parameters$cost

# tune svm is very time consuming, takes about 5 hours to run
# tuned result is gamma=0.01, cost = 0.21
bestgamma; bestcost
```

```
## [1] 0.001

## [1] 0.21
```

```
end <- Sys.time()
timedif <- end - begin
cat("Time for tuning","SVM","is", timedif)
```

```
## Time for tuning SVM is 21.75782
```

```
timedif
```

```
## Time difference of 21.75782 mins
```

SVM Train

```
begin <- Sys.time()
svm_fit <- svm_train(scale(as.matrix(trainx)), trainy, bestgamma, bestcost)
#svm_fit <- svm_train(scale(as.matrix(trainx)), trainy, 0.01, 0.11)
```

Step 5.1(b): Test SVM with with tuning parameters

```
svm_pred <- svm_test(svm_fit, scale(as.matrix(testx)))

svm.pred <- ifelse(svm_pred > mean(svm_pred), 1, 0)

# Calculate Accuracy
cat("MSE is",mean((svm.pred-testy)^2),
    "and accuracy is",mean(svm.pred==testy))
```

```
## MSE is 0.355222 and accuracy is 0.644778
```

```
confusionMatrix(factor(svm.pred), factor(testy))$byClass
```

```
##          Sensitivity          Specificity       Pos Pred Value
##            0.7773279            0.6205621            0.2723404
##       Neg Pred Value            Precision               Recall
##            0.9384787            0.2723404            0.7773279
##                   F1           Prevalence       Detection Rate
##            0.4033613            0.1544715            0.1200750
## Detection Prevalence    Balanced Accuracy
##            0.4409006            0.6989450
```

```
tpr.fpr <- WeightedROC(svm.pred, testy, weight_test)
auc.svm <- WeightedAUC(tpr.fpr)
cat("AUC is",auc.svm)
```

```
## AUC is 0.698945
```

```
end <- Sys.time()
timedif <- end - begin
cat("Time for running","SVM","is", timedif)
```

```
## Time for running SVM is 2.897044
```

```
timedif
```

```
## Time difference of 2.897044 mins
```

Neural Net

```
source("./LASSO_NN.R")
begin <- Sys.time()
nndata <- data.frame(
  scale(as.matrix(trainx)), HEALTH = trainy)

nnfit <- nnReg(nndata, hd = c(35, 25, 18, 13, 8, 5))

plot(nnfit)

end <- Sys.time()
timedif <- end - begin
cat("Time for training","neuralnet","is", timedif)
```

```
## Time for training neuralnet is 27.45296
```

```
timedif
```

```
## Time difference of 27.45296 secs
```

```
begin <- Sys.time()

pred_nn <- predict(nnfit, testx)
summary(pred_nn)
```

```
##        V1                  V2
##  Min.   :0.0000008   Min.   :0.0000000
##  1st Qu.:0.0000083   1st Qu.:0.0000000
##  Median :0.9991259   Median :0.0004897
##  Mean   :0.5476543   Mean   :0.4543838
##  3rd Qu.:0.9999998   3rd Qu.:0.9999977
##  Max.   :1.0000000   Max.   :0.9999998
```

```
nn.prediction <- rep(0, length(testy))

for(i in 1:length(nn.prediction)){
  if(pred_nn[i,1] < pred_nn[i,2]){
    nn.prediction[i] <- 0
  } else{
    nn.prediction[i] <- 1
  }
}
summary(nn.prediction)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  1.0000  0.5453  1.0000  1.0000
```

```
confusionMatrix(factor(nn.prediction), factor(testy))$byClass
```

```
##          Sensitivity          Specificity       Pos Pred Value
##           0.42105263           0.53920118           0.14305365
##       Neg Pred Value            Precision               Recall
##           0.83600917           0.14305365           0.42105263
##                   F1           Prevalence       Detection Rate
##           0.21355236           0.15447154           0.06504065
## Detection Prevalence    Balanced Accuracy
##           0.45465916           0.48012691
```

```
tpr.fpr <- WeightedROC(nn.prediction, testy, weight_test)
auc.nn <- WeightedAUC(tpr.fpr)

cat("AUC is", auc.nn)
```

```
## AUC is 0.4801269
```

```
end <- Sys.time()
timedif <- end - begin
cat("Time for predicting using","neuralnet","is", timedif)
```

```
## Time for predicting using neuralnet is 0.03086591
```

```
timedif
```

```
## Time difference of 0.03086591 secs
```

Naive Bayies

```
begin <- Sys.time()


nbfit = nbReg(datatrain)
nbpred = predict(nbfit, testx)

mean((as.numeric(nbpred)-1 - testy)^2)
```

```
## [1] 0.3327079
```

```
confusionMatrix(nbpred, factor(testy))$byClass
```

```
##          Sensitivity          Specificity       Pos Pred Value
##            0.6761134            0.6656805            0.2697900
##       Neg Pred Value            Precision               Recall
##            0.9183673            0.2697900            0.6761134
##                   F1           Prevalence       Detection Rate
##            0.3856813            0.1544715            0.1044403
## Detection Prevalence    Balanced Accuracy
##            0.3871169            0.6708969
```

```
tpr.fpr <- WeightedROC(as.numeric(nbpred)-1, testy, weight_test)
auc.nb <- WeightedAUC(tpr.fpr)

cat("AUC is", auc.nb)
```

```
## AUC is 0.6708969
```

```
end <- Sys.time()
timedif <- end - begin
cat("Time for running","Naive Bayies","is", timedif)
```

```
## Time for running Naive Bayies is 0.8138459
```

```
timedif
```

```
## Time difference of 0.8138459 secs
```