hw5\_prob3\_CODE\_\_final.R

jrdha

Wed Nov 28 09:09:58 2018

## AUTHOR: Jared Hansen  
## LAST MODIFIED: Sunday, 11/25/2018  
  
  
  
  
#================================================================  
#==== CODE FOR STATISTICAL LEARNING II, hw5 PROBLEM 3 ===========  
#================================================================  
  
  
# Necessary libraries for the script  
library(dplyr)   
library(R.matlab)   
library(geometry)   
library(pracma)   
library(randomForest)  
library(caret)  
library(magrittr)  
library("e1071")  
  
  
# Change the working directory so that the image file can be read in.  
# The line below needs to be changed if you want to run the code.  
setwd("C:/Users/jrdha/OneDrive/Desktop/USU\_Fa2018/Moon\_\_SLDM2/hw5/problem3")  
  
  
# Read in the UN-SHIFTED MNIST data as a dataframe with  
# response "Y" and predictors "X\_1", "X\_2",...  
df <- R.matlab::readMat("mnist\_49\_3000.mat") %>% lapply(t) %>% lapply(as\_tibble)  
colnames(df[[1]]) <- sprintf("X\_%s",seq(1:ncol(df[[1]])))  
colnames(df[[2]]) <- c("Y")  
df <- bind\_cols(df) %>% select(Y, everything())  
  
# Split the data into a training set (first 2000 obs).  
# Add a column of leading 1s for correct dimensions/calculations when dealing  
# with b in theta.  
set.seed(2345)  
trainData <- dplyr::slice(df, 1:2000)  
trainPredictors <- select(trainData, -Y)  
X\_0 <- rep(1, 2000)  
trainPredictors <- cbind(X\_0, trainPredictors)  
trainResponse <- select(trainData, Y)  
trainData$Y <- as.factor(trainData$Y)  
  
# Split the data into a test set too (last 1000 obs).  
# Add a column of leading 1s for correct dimensions/calculations when dealing  
# with b in theta.  
set.seed(2345)  
testData <- dplyr::slice(df, 2001:3000)  
testPredictors <- select(testData, -Y)  
X\_0 <- rep(1000)  
testPredictors <- cbind(X\_0, testPredictors)  
testResponse <- select(testData, Y)  
testData$Y <- as.factor(testData$Y)  
  
# Read in the SHIFTED MNIST data as a dataframe with  
# response "Y" and predictors "X\_1", "X\_2",...  
s\_df <- R.matlab::readMat("mnist\_49\_1000\_shifted.mat") %>% lapply(t) %>% lapply(as\_tibble)  
colnames(s\_df[[1]]) <- sprintf("X\_%s",seq(1:ncol(s\_df[[1]])))  
colnames(s\_df[[2]]) <- c("Y")  
s\_df <- bind\_cols(s\_df) %>% select(Y, everything())  
  
  
  
  
#===============================================================================  
#==== RANDOM FORESTS ===========================================================  
#===============================================================================  
set.seed(2345)  
  
# Fitting the initial model (left default function arguments)  
# ntree = 500  
# mtry = sqrt(785) = 28.01785  
# nodesize = 1  
rf\_model <- randomForest(Y ~., data = trainData)  
  
# Predicting onto the UN-SHIFTED test data  
rf\_prediction <- predict(rf\_model, newdata = testData)  
  
#===== Return the UN-SHIFTED test error FOR RANDOM FORESTS: 0.017 ==============  
1 - mean(rf\_prediction == testData$Y)  
  
# Predicting onto the SHIFTED test data  
rf\_prediction <- predict(rf\_model, newdata = s\_df)  
  
#===== Return the SHIFTED test error for RANDOM FORESTS: 0.477 =================  
1 - mean(rf\_prediction == testData$Y)  
  
  
  
  
#===============================================================================  
#===== SVM =====================================================================  
#===============================================================================  
  
# Tuning LINEAR KERNEL: Round 1 (used this to get a set of values that did well)  
# Ran it by changing cost = 1.digit\*10^(-3:3) where I set digit = 0,1,2,....,9  
set.seed(2345)  
svm\_tune\_linear <- tune.svm(Y ~.,  
 data = trainData,  
 kernel = "linear",   
 cost = 1.0\*10^(-3:2))  
print(svm\_tune\_linear)  
  
# FIT FINAL SVM MODEL, USING THE 10-FOLD CV cost=0.01 FOR LINEAR KERNEL  
svm\_model <- svm(Y ~., trainData, kernel = "linear", cost = 0.01)  
testData$pred <- predict(svm\_model, testData)  
testData$correct <- with(testData, ifelse(testData$Y == pred, 1, 0))  
# Generate the test error  
# Total number of correct classifications  
totalCorrect <- sum(testData$correct)  
testError\_SVM <- 1 - (totalCorrect / nrow(testData))  
testError\_SVM  
# Remove the "correct" column for the SVM portion  
testData <- subset(testData, select = -c(correct, pred))  
#=========== RESULTS: SVM, LINEAR-KERNEL =======================================  
#=========== UN-SHIFTED Test error rate of 0.049 ===============================  
  
# FIT FINAL SVM MODEL, USING THE 10-FOLD CV cost=0.01 FOR LINEAR KERNEL  
s\_df$pred <- predict(svm\_model, s\_df)  
s\_df$correct <- with(s\_df, ifelse(s\_df$Y == pred, 1, 0))  
# Generate the test error  
# Total number of correct classifications  
totalCorrect <- sum(s\_df$correct)  
testError\_SVM <- 1 - (totalCorrect / nrow(s\_df))  
testError\_SVM  
# Remove the "correct" column for the SVM portion  
s\_df <- subset(s\_df, select = -c(correct, pred))  
#=========== RESULTS: SVM, LINEAR-KERNEL =======================================  
#=========== SHIFTED Test error rate of 0.517 ==================================  
  
#========== TUNING GAUSSIAN-KERNEL MODEL =======================================  
# INITIAL PASS (leave base values at 10)  
set.seed(2345)  
svm\_tune\_gaussian <- tune.svm(Y~., data = trainData,  
 cost = 10^(-3:2),  
 gamma = 10^(-5:2))  
print(svm\_tune\_gaussian)  
# Parameter tuning of 'svm':  
# - sampling method: 10-fold cross validation   
# - best parameters:  
# gamma cost  
# 0.01 100  
# - best performance: 0.0185   
  
# THIS FITS A FINAL SVM MODEL, USING THE CROSSVALIDATED TUNING PARAMETERs  
# of C = 100, and gamma = 0.01  
costVal = 100  
gammaVal = 0.01  
svm\_model <- svm(Y ~., trainData, kernel = "radial",  
 cost = costVal, gamma = gammaVal)  
testData$pred <- predict(svm\_model, testData)  
testData$correct <- with(testData, ifelse(testData$Y == pred, 1, 0))  
# Generate the test error  
# Total number of correct classifications  
totalCorrect <- sum(testData$correct)  
testError\_SVM <- 1 - (totalCorrect / nrow(testData))  
testError\_SVM  
# Remove the "correct" column for the SVM portion  
testData <- subset(testData, select = -c(correct, pred))  
#========= RESULTS: SVM, GAUSSIAN-KERNEL =======================================  
#========= Test error UN-SHIFTED: 0.018 ========================================  
  
s\_df$pred <- predict(svm\_model, s\_df)  
s\_df$correct <- with(s\_df, ifelse(s\_df$Y == pred, 1, 0))  
# Generate the test error  
# Total number of correct classifications  
totalCorrect <- sum(s\_df$correct)  
testError\_SVM <- 1 - (totalCorrect / nrow(s\_df))  
testError\_SVM  
# Remove the "correct" column for the SVM portion  
s\_df <- subset(s\_df, select = -c(correct, pred))  
#========= RESULTS: SVM, GAUSSIAN-KERNEL =======================================  
#========= Test error SHIFTED: 0.478 ===========================================  
  
  
  
  
# Re-run this just to be sure we're working with the right data before doing  
# logistic regression  
set.seed(2345)  
trainData <- dplyr::slice(df, 1:2000)  
trainPredictors <- select(trainData, -Y)  
X\_0 <- rep(1, 2000)  
trainPredictors <- cbind(X\_0, trainPredictors)  
trainResponse <- select(trainData, Y)  
trainData$Y <- as.factor(trainData$Y)  
  
# Split the data into a test set too (last 1000 obs).  
# Add a column of leading 1s for correct dimensions/calculations when dealing  
# with b in theta.  
set.seed(2345)  
testData <- dplyr::slice(df, 2001:3000)  
testPredictors <- select(testData, -Y)  
X\_0 <- rep(1000)  
testPredictors <- cbind(X\_0, testPredictors)  
testResponse <- select(testData, Y)  
testData$Y <- as.factor(testData$Y)  
  
#===============================================================================  
#===== LOGISTIC REGRESSION W/OUT BAGGING =======================================  
#===============================================================================  
  
# Fit logistic regression model  
logRegr\_model <- glm(Y ~ . ,  
 data = trainData,  
 family = binomial  
 )  
  
# Generates the predictions for our UN-SHIFTED testData, cutoff of 0.5  
testData$pred <- predict(logRegr\_model, newdata = testData, type = "response")  
testData$pred <- ifelse(testData$pred > 0.5, "1", "-1")  
  
# Generates a column that says whether or not a test observation was correctly  
# classified (1==correct, 0==incorrect)  
testData$correct <- with(testData, ifelse(Y == pred, 1, 0))  
  
# Generate the test error  
totalCorrect <- sum(testData$correct)  
testError\_logReg <- 1 - (totalCorrect / nrow(testData))  
testError\_logReg  
#============= RESULTS: LOGISTIC REGRESSION, 1 model ===========================  
#============= UN-SHIFTED Test error of 0.152 ==================================  
  
# Remove the "correct" and "pred" columns so that we'll have the original data   
# to work with for the SVM portion  
testData <- subset(testData, select = -c(correct, pred))  
  
# Generates the predictions for our SHIFTED s\_df, cutoff of 0.5  
s\_df$pred <- predict(logRegr\_model, newdata = s\_df, type = "response")  
s\_df$pred <- ifelse(s\_df$pred > 0.5, "1", "-1")  
  
# Generates a column that says whether or not a test observation was correctly  
# classified (1==correct, 0==incorrect)  
s\_df$correct <- with(s\_df, ifelse(Y == pred, 1, 0))  
  
# Generate the test error  
totalCorrect <- sum(s\_df$correct)  
testError\_logReg <- 1 - (totalCorrect / nrow(s\_df))  
testError\_logReg  
#============= RESULTS: LOGISTIC REGRESSION, 1 model ===========================  
#============= SHIFTED Test error of 0.391 =====================================  
  
# Remove the "correct" and "pred" columns so that we'll have the original data   
# to work with for the SVM portion  
s\_df <- subset(s\_df, select = -c(correct, pred))  
  
  
  
  
#===============================================================================  
#===== LOGISTIC REGRESSION W/BAGGING, 51 SAMPLES ===============================  
#===============================================================================  
  
# First, generate 51 bootstrapped samples  
set.seed(2345)  
  
bootModel <- function(trainData, testData, i){  
   
 loop\_seed = 1000+i  
 set.seed(loop\_seed)  
   
 lengthTrain <- nrow(trainData)  
   
 # Randomly sample indices from 1 - 2000 (observations from training data)  
 btstrp\_ind <- sample(seq\_len(lengthTrain), size = lengthTrain)  
   
 # Make the observations at these indices your new training data  
 bt\_train <- trainData[btstrp\_ind, ]  
   
 # Fit logistic regression model  
 logRegr\_model <- glm(Y ~ . ,  
 data = bt\_train,  
 family = binomial  
 )  
   
 # Generates the predictions for our testData, cutoff of 0.5  
 testData$pred <- predict(logRegr\_model, newdata = testData, type = "response")  
 testData$pred <- ifelse(testData$pred > 0.5, "1", "-1")  
   
 # Generates a column that says whether or not a test observation was correctly  
 # classified (1==correct, 0==incorrect)  
 testData$correct <- with(testData, ifelse(Y == pred, 1, 0))  
   
 # Generate the test error  
 totalCorrect <- sum(testData$correct)  
 testError\_logReg <- 1 - (totalCorrect / nrow(testData))  
 testError\_logReg  
   
 # Store the predictions in here before removing that column from testData  
 predictions <- testData$pred  
   
 # Remove the "correct" and "pred" columns so that we'll have the original data   
 # to work with for the SVM portion  
 testData <- subset(testData, select = -c(correct, pred))  
   
 return(predictions)  
}  
  
# Initialize empty dataframe of predictions  
all\_51\_preds <- data.frame(matrix(NA, nrow = 1000, ncol = 51))  
  
# Generate the predictions from the 51 models fitted on bootstrapped data  
for (i in 1:51){  
 preds <- bootModel(trainData, testData, i)  
 all\_51\_preds[ ,i] <- preds  
}  
  
# Make sure we're getting some different predictions  
# all\_51\_preds[,1] == all\_51\_preds[,3]  
  
# We'll use these two functions to make the data be numeric and not character  
asNumeric <- function(x){  
 as.numeric(as.character(x))  
}  
factorsNumeric <- function(d){  
 modifyList(d, lapply(d[, sapply(d, is.character)], asNumeric))  
}  
  
# Used this during de-bugging to save on computation time  
# all\_51\_preds <- all\_51\_preds[, 1:3]  
  
# Make the whole dataframe numeric so we can sum the rows (easier to get vote)  
all\_51\_preds <- factorsNumeric(all\_51\_preds)  
  
# Sum each of the rows, adding a new column that gives this sum  
all\_51\_preds$sum <- rowSums(all\_51\_preds)  
  
# If the sum of a row is negative then the majority vote is for -1, if the sum  
# of the row is positive then the majority vote is for 1  
all\_51\_preds$vote <- ifelse(all\_51\_preds$sum < 0, -1, 1)  
  
# Add on the true labels to the dataframe  
all\_51\_preds$truth <- testData$Y  
  
# Finally, make a column that says whether or not the majority vote was correct  
all\_51\_preds$correct <- ifelse(all\_51\_preds$vote == all\_51\_preds$truth, 1, 0)  
  
# Generate test error  
# Total number of correct classifications  
totalCorrect <- sum(all\_51\_preds$correct)  
testError\_LR\_51 <- 1 - (totalCorrect / nrow(testData))  
testError\_LR\_51  
#============= RESULTS: LOGISTIC REGRESSION, 51 models VOTING ==================  
#============= UN-SHIFTED Test error of 0.158 ==================================  
  
#==== NOW FOR THE SHIFTED DATA, USING THE SAME 51 PREDICTION MODELS ============  
#===============================================================================  
# Initialize empty dataframe of predictions  
all\_51\_shift <- data.frame(matrix(NA, nrow = 1000, ncol = 51))  
  
# Generate the predictions from the 51 models fitted on bootstrapped data  
for (i in 1:51){  
 preds <- bootModel(trainData, s\_df, i)  
 all\_51\_shift[ ,i] <- preds  
}  
  
# Make the whole dataframe numeric so we can sum the rows (easier to get vote)  
all\_51\_shift <- factorsNumeric(all\_51\_shift)  
  
# Sum each of the rows, adding a new column that gives this sum  
all\_51\_shift$sum <- rowSums(all\_51\_shift)  
  
# If the sum of a row is negative then the majority vote is for -1, if the sum  
# of the row is positive then the majority vote is for 1  
all\_51\_shift$vote <- ifelse(all\_51\_shift$sum < 0, -1, 1)  
  
# Add on the true labels to the dataframe  
all\_51\_shift$truth <- s\_df$Y  
  
# Finally, make a column that says whether or not the majority vote was correct  
all\_51\_shift$correct <- ifelse(all\_51\_shift$vote == all\_51\_shift$truth, 1, 0)  
  
# Generate test error  
# Total number of correct classifications  
totalCorrect <- sum(all\_51\_shift$correct)  
testError\_LR\_51\_SHIFT <- 1 - (totalCorrect / nrow(s\_df))  
testError\_LR\_51\_SHIFT  
#============= RESULTS: LOGISTIC REGRESSION, 51 models VOTING ==================  
#============= SHIFTED Test error of 0.407 =====================================  
  
  
  
  
#===============================================================================  
#===== LOGISTIC REGRESSION W/BAGGING, 101 SAMPLES ==============================  
#===============================================================================  
  
# First, generate 101 bootstrapped samples  
set.seed(2345)  
  
bootModel <- function(trainData, testData, i){  
   
 loop\_seed = 1000+i  
 set.seed(loop\_seed)  
   
 lengthTrain <- nrow(trainData)  
   
 # Randomly sample indices from 1 - 2000 (observations from training data)  
 btstrp\_ind <- sample(seq\_len(lengthTrain), size = lengthTrain)  
   
 # Make the observations at these indices your new training data  
 bt\_train <- trainData[btstrp\_ind, ]  
   
 # Fit logistic regression model  
 logRegr\_model <- glm(Y ~ . ,  
 data = bt\_train,  
 family = binomial  
 )  
   
 # Generates the predictions for our testData, cutoff of 0.5  
 testData$pred <- predict(logRegr\_model, newdata = testData, type = "response")  
 testData$pred <- ifelse(testData$pred > 0.5, "1", "-1")  
   
 # Generates a column that says whether or not a test observation was correctly  
 # classified (1==correct, 0==incorrect)  
 testData$correct <- with(testData, ifelse(Y == pred, 1, 0))  
   
 # Generate the test error  
 totalCorrect <- sum(testData$correct)  
 testError\_logReg <- 1 - (totalCorrect / nrow(testData))  
 testError\_logReg  
   
 # Store the predictions in here before removing that column from testData  
 predictions <- testData$pred  
   
 # Remove the "correct" and "pred" columns so that we'll have the original data   
 # to work with for the SVM portion  
 testData <- subset(testData, select = -c(correct, pred))  
   
 return(predictions)  
}  
  
# Initialize empty dataframe of predictions  
all\_101\_preds <- data.frame(matrix(NA, nrow = 1000, ncol = 101))  
  
# Generate the predictions from the 101 models fitted on bootstrapped data  
for (i in 1:101){  
 preds <- bootModel(trainData, testData, i)  
 all\_101\_preds[ ,i] <- preds  
}  
  
# Make sure we're getting some different predictions  
# all\_101\_preds[,1] == all\_101\_preds[,3]  
  
# We'll use these two functions to make the data be numeric and not character  
asNumeric <- function(x){  
 as.numeric(as.character(x))  
}  
factorsNumeric <- function(d){  
 modifyList(d, lapply(d[, sapply(d, is.character)], asNumeric))  
}  
  
# Used this during de-bugging to save on computation time  
# all\_101\_preds <- all\_101\_preds[, 1:3]  
  
# Make the whole dataframe numeric so we can sum the rows (easier to get vote)  
all\_101\_preds <- factorsNumeric(all\_101\_preds)  
  
# Sum each of the rows, adding a new column that gives this sum  
all\_101\_preds$sum <- rowSums(all\_101\_preds)  
  
# If the sum of a row is negative then the majority vote is for -1, if the sum  
# of the row is positive then the majority vote is for 1  
all\_101\_preds$vote <- ifelse(all\_101\_preds$sum < 0, -1, 1)  
  
# Add on the true labels to the dataframe  
all\_101\_preds$truth <- testData$Y  
  
# Finally, make a column that says whether or not the majority vote was correct  
all\_101\_preds$correct <- ifelse(all\_101\_preds$vote == all\_101\_preds$truth, 1, 0)  
  
# Generate test error  
# Total number of correct classifications  
totalCorrect <- sum(all\_101\_preds$correct)  
testError\_LR\_101 <- 1 - (totalCorrect / nrow(testData))  
testError\_LR\_101  
#============= RESULTS: LOGISTIC REGRESSION, 101 models VOTING ==================  
#============= UN-SHIFTED Test error of 0.158 ==================================  
  
#==== NOW FOR THE SHIFTED DATA, USING THE SAME 101 PREDICTION MODELS ============  
#===============================================================================  
# Initialize empty dataframe of predictions  
all\_101\_shift <- data.frame(matrix(NA, nrow = 1000, ncol = 101))  
  
# Generate the predictions from the 101 models fitted on bootstrapped data  
for (i in 1:101){  
 preds <- bootModel(trainData, s\_df, i)  
 all\_101\_shift[ ,i] <- preds  
}  
  
# Make the whole dataframe numeric so we can sum the rows (easier to get vote)  
all\_101\_shift <- factorsNumeric(all\_101\_shift)  
  
# Sum each of the rows, adding a new column that gives this sum  
all\_101\_shift$sum <- rowSums(all\_101\_shift)  
  
# If the sum of a row is negative then the majority vote is for -1, if the sum  
# of the row is positive then the majority vote is for 1  
all\_101\_shift$vote <- ifelse(all\_101\_shift$sum < 0, -1, 1)  
  
# Add on the true labels to the dataframe  
all\_101\_shift$truth <- s\_df$Y  
  
# Finally, make a column that says whether or not the majority vote was correct  
all\_101\_shift$correct <- ifelse(all\_101\_shift$vote == all\_101\_shift$truth, 1, 0)  
  
# Generate test error  
# Total number of correct classifications  
totalCorrect <- sum(all\_101\_shift$correct)  
testError\_LR\_101\_SHIFT <- 1 - (totalCorrect / nrow(s\_df))  
testError\_LR\_101\_SHIFT  
#============= RESULTS: LOGISTIC REGRESSION, 101 models VOTING =================  
#============= SHIFTED Test error of 0.407 =====================================