```r
#================================================================
#==== 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]])))
```

```r
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))
```

```r
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)
```

```r
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")
```

```r
# 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){
```

```r
  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)
```

```r
# 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)

# 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 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
```

```r
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 ======================================
```