Classification

Anisha Mittal

```
library(adabag)
library(randomForest)
library(kernlab)
```

Data

The given data contains 983 mid-infrared (MIR) spectra collected from different authenticated fruit purees. The purees are either obtained from pure strawberry, or from adulterated strawberries and/or other black and red colored fruits. The sample spectra are thus assigned to one of two classes: strawberry and nonstrawberry. The spectra were acquired from each puree using attenuated total reluctance sampling, with the acquisition order randomized with respect to the sample type. The spectra are measured over 47 wavelenghts. The task is to predict whether a sample is a strawberry/non-strawberry puree on the basis of the MIR data for the purpose of food authenticity recognition.

Since all variable are measured on a same scale, we do not need to scale the data.

```
data_strwbry <- read.csv("D:/Anisha/Projects/Classification 2/data_mir_strawberry.csv", stringsAsFactors

# Number of tress
n_trees <- 100

# set aside test and train data
N <- nrow(data_strwbry)
set.seed(20200649)
test  <- sample(1:N, N*0.3) # 30% for test
train <- setdiff(1:N, test) # 70% for train

# Divide data to test and train data
x_train <- data_strwbry[train,]
y_train <- x_train$class
x_test <- data_strwbry[test,]
y_test <- x_test$class

# Number of train data
N_train <- nrow(x_train)

K <- 5 # number of folds
R <- 10 # number of replicates --- NOTE : could be slow

# we use this function to compute classification accuracy
class_acc <- function(y, yhat) {
```

```
  tab <- table(y, yhat)
  return( sum(diag(tab))/sum(tab) )
  }
```

Task 1. Compare random forests, boosting, and SVM classifiers in application to classification of the puree spectra.

```
# Random Forest

# fit model
fitrf <- randomForest(class ~ ., data = x_train, ntree = n_trees,
                      mtry=ncol(data_strwbry) - 1)

# extract/compute training classification error as function of number of trees
err_train_rf <- fitrf$err.rate[n_trees,1]
acc_train_rf <- 1 - err_train_rf
print(paste("Random forest Training Error: ",round(err_train_rf,3),
            "and Accuracy: ",round(acc_train_rf,3)))
```

```
## [1] "Random forest Training Error:  0.058 and Accuracy:  0.942"
```

We do not need to cross validate the result in the case of random forest as there is no need to worry about over fitting.

```
# Boosting

# fit model
fitboost <- boosting.cv(class ~ ., data = x_train,coeflearn = "Breiman",
                        v = K, mfinal = n_trees, boos = FALSE)
```

```
## i:  1 Mon Jul 24 16:18:48 2023
## i:  2 Mon Jul 24 16:18:56 2023
## i:  3 Mon Jul 24 16:19:03 2023
## i:  4 Mon Jul 24 16:19:13 2023
## i:  5 Mon Jul 24 16:19:18 2023
```

```
# cross validation accuracy
err_val_boost <- fitboost$error
acc_val_boost <- 1 - err_val_boost
print(paste("Boosting Validation Error: ",round(err_val_boost,3),
            "and Accuracy: ",round(acc_val_boost,3)))
```

```
## [1] "Boosting Validation Error:  0.055 and Accuracy:  0.945"
```

```
# Support Vector Machine (SVM)
x <- as.matrix(data_strwbry[,-1])
y <- as.matrix(as.numeric(data_strwbry[,1]))

C <- c(1, 2, 5, 10, 20)
```

```r
sigma <- c(0.010, 0.015, 0.020, 0.025, 0.030)
grid <- expand.grid(C, sigma)
colnames(grid) <- c("C", "sigma")
head(grid,10)
```

```
##     C sigma
## 1   1 0.010
## 2   2 0.010
## 3   5 0.010
## 4  10 0.010
## 5  20 0.010
## 6   1 0.015
## 7   2 0.015
## 8   5 0.015
## 9  10 0.015
## 10 20 0.015
```

```r
n_mod <- nrow(grid)

# store accuracy output
out <- vector("list", R)

for ( r in 1:R ) {
  # accuracy of the classifiers in the K folds
  acc <- matrix(NA, K, n_mod)
  folds <- rep( 1:K, ceiling(N_train/K))
  # random permute
  folds <- sample(folds)
  # ensure we got N_train data points
  folds <- folds[1:N_train]
    for ( k in 1:K ) {
      train_fold <- which(folds != k)
      validation <- setdiff(1:N_train, train_fold)
      # fit SVM on the training data and assess on the validation set
      for ( j in 1:n_mod ) {
        fit <- ksvm(x[train_fold,], y[train_fold], type = "C-svc",
                    kernel = "rbfdot", C = grid$C[j],
                    kpar = list(sigma = grid$sigma[j]))
        pred <- predict(fit, newdata = x[validation,])
        acc[k,j] <- class_acc(pred, y[validation])
      }
    }
  out[[r]] <- acc
}

avg_fold_acc <- t(sapply(out, colMeans))

# estimated mean accuracy
avg_acc <- colMeans(avg_fold_acc)
grid_acc <- cbind(grid, avg_acc)

# Get best accuracy
best <- which.max(grid_acc$avg_acc)
```

```
grid_acc[best,]
```

```
##    C sigma   avg_acc
## 9 10 0.015 0.9764932
```

```
acc_val_svm <- grid_acc[best,3]
err_val_svm <- 1 - acc_val_svm
print(paste("SVM Validation Error: ",round(err_val_svm,3),
            "and Accuracy: ",round(acc_val_svm,3)))
```

```
## [1] "SVM Validation Error:  0.024 and Accuracy:  0.976"
```

Task 2. Evaluate and discuss the predictive performance of the best model at classifying new observations.

```
all_val <- c(RF = acc_train_rf, Boost = acc_val_boost, SVM = acc_val_svm)
all_val
```

```
##    RF.OOB     Boost       SVM
## 0.9419448 0.9448476 0.9764932
```

```
best_val <- which.max(all_val)
best_val
```

```
## SVM
##   3
```

```
if (best_val==1){
  # RF testing
  mdlrf <- randomForest(class ~ ., data = x_test, ntree = n_trees,mtry=3)
  pred_rf <- predict(fit_rf,x_test[,-1])
  rf_mat <- table(x_test[,-1],pred_rf)
  # extract/compute training classification error as function of number of trees
  err_test_rf <- mdlrf$err.rate[n_trees,1]
  acc_test_rf <- 1 - err_test_rf
  print(paste("Random Forest test Error: ",round(err_test_rf,3),
              " Accuracy: ",round(acc_test_rf,3)))
  print(rf_mat)

  } else if (best_val==2){
    # boosting testing
    mdlboost <- boosting(class ~ ., data = data_strwbry[train,],
                         coeflearn = "Breiman" ,mfinal = n_trees,boos = FALSE)
    pred_boost <- predict(fit_boost_final,x_test)
    # cross validation accuracy
    err_test_boost <- errorevol(mdlboost, x_test)$error
    acc_test_boost <- 1 - err_test_boost
    print(paste("Boosting test Error: ",round(err_test_boost[100],3),
```

```
                " Accuracy: ",round(acc_test_boost[100],3)))
    print(pred_boost$confusion)

    } else if (best_val==3){
      # SVM testing
      mdlsvm <- ksvm(x[train,], y[train], type = "C-svc",
                     kernel = "rbfdot",C = grid_acc[best,1],
                     kpar = list(sigma = grid_acc[best,2]))
      pred_svm <- predict(mdlsvm, newdata = x[test,])
      svm_mat <- table(pred_svm,y[test,1])
      acc_test_svm <- class_acc(pred_svm, y[test])
      err_test_svm <- 1 - acc_test_svm
      print(paste("SVM test Error: ",round(err_test_svm,3),
                  " Accuracy: ",round(acc_test_svm,3)))
      print(svm_mat)
      }
```

```
## [1] "SVM test Error:  0.01  Accuracy:  0.99"
##
## pred_svm   1   2
##        1 188   1
##        2   2 103
```

Task 3.

For real-world application of the model, a false positive is considered worst than a false negative. For an example, if someone is tested for cancer and the result shows they have no signs of cancer (while they actually have cancer) it could lead to problems for the person due to lack of medical attention. However, if results shows signs of cancer (while they don't have cancer) then this would not lead to a serious problem. Since eventually actual result will be found when they get further medical attention.

As seen above we get that SVM is best model as selected by highest validation accuracy. We also see we have lesser False positive sample classified than False negative. Which is good. So our selected model is good.