Math 5365

Data Mining 1

Homework 16

Mary Barker

1. Load the fgl data set in the MASS library.

   Split the data into 70% training and 30% testing data. Predict the glass types in the test data using the following three methods, and compare the classification accuracies of these methods.

   (a) Build an SVM and use the predict command.

   Using the basic SVM command with no altered parameters, the accuracy is 73.4375%

   (b) Write a script implementing the one-against-rest approach with SVM's (much of the code you need is in the 'programming one-against-rest' section of chapter5.R)

   The confusion matrix for this case is shown below.

		predy			
		Con	Head	WinF	WinNF
	WinF	0	0	21	5
	WinNF	0	0	3	18
y	Veh	0	0	3	0
	Con	2	0	0	1
	Tabl	0	0	1	0
	Head	0	9	1	0

   (c) Write a script implementing the one-against-one approach with SVM's.

   The confusion matrix for this test is shown below.

		y					
		WinF	WinNF	Veh	Con	Tabl	Head
	WinF	16	10	0	0	0	0
	WinNF	2	19	0	0	0	0
predy	Veh	2	1	0	0	0	0
	Con	0	1	0	2	0	0
	Tabl	0	0	0	0	1	0
	Head	1	0	0	0	0	9

```
#Load the fgl data set in the MASS library.

library(e1071)
data(fgl,package='MASS')
source('~/Dropbox/Tarleton/data_mining/class_notes/extras.R')
source('~/Dropbox/Tarleton/data_mining/generic_functions/dataset_ops.R')

# Split the data into 70\% training and 30\% testing data.
# Predict the glass types in the test data using the following
# three methods, and compare the classification accuracies of
# these methods.

splitset <- splitdata(fgl,0.7,F)
train <- splitset$train

#(a) Build an SVM and use the predict command.

a_model <- svm(type~., fgl[train,])
a_pred <- predict(a_model,fgl[-train,])
```

```
a_acc <- confmatrix(fgl$type[-train],a_pred)

#(b) Write a script implementing the one-against-rest approach
#   with SVM’s (much of the code you need is in the ’programming
#   one-against-rest’ section of chapter5.R)

one_against_rest <- function(dset, v_idx, train, my_pred){
  tmpname <- names(dset)[v_idx]
  names(dset)[v_idx] <- ’type’
  levels_t <- levels(dset$type)

  tmpdset <- dset[train,]
  levels(tmpdset$type) <- c(levels_t, ’other’)
  tmptype <- dset$type[train]
  preds <- list()
  for(idx in 1:length(levels_t) ){
          name <- levels_t[idx]

    index_v <- (tmpdset$type != name) * 1
    tmpdset$type[index_v == 1] <- ’other’

         model <- svm(type~., tmpdset)
         tmppred <- predict(model, dset[-train,])
         preds[[idx]] <- tmppred
         tmpdset$type <- tmptype
    levels(tmpdset$type) <- c(levels_t,’other’)
  }
```

```

  votes <- matrix(0,
                nrow = (nrow(dset[-train,])),
                ncol = length(levels_t) )

  for(idx in 1:length(levels_t)){
    name <- levels_t[idx]
    votes[,idx] <- (preds[[idx]] == name)
    for(i in 1:length(levels_t)){
      if(i != idx){
        votes[,idx] <- votes[,idx] + ( preds[[i]] == 'other' )
      }
    }
  }

  one_against_rest_pred <- levels_t[apply(votes, 1 ,which.max)]
  return(table(my_pred, one_against_rest_pred))
}

tab1 <- one_against_rest(fgl, ncol(fgl), train, fgl$type[-train])

#(c) Write a script implementing the one-against-one approach
#   with SVM's.

comp_preds <- function(idx1, idx2, var, train, dset){
  tmpname <- names(dset)[var]
  names(dset)[var] <- 'my_var_name'
```

```
  myvar <- dset[,var]

  n1 <- levels(myvar)[idx1]
  n2 <- levels(myvar)[idx2]

  idx <- (((myvar == n1) | (myvar == n2) )) * 1
  idx[-train] = 0

  v1_v2_train <- dset[idx == 1,]
  v1_v2_model <- svm(my_var_name~., v1_v2_train)
  v1_v2_pred <- predict(v1_v2_model, dset[-train,], type='class')

  names(dset)[var] <- tmpname

  return(v1_v2_pred)
}

comp_votes <- function(preds, dset, v_idx){

  nr = length(preds[[1]][[1]])
  nc = length(levels(dset[,v_idx]))

  votes <- matrix(0,nrow=nr, ncol=nc)
  for(k in 1:nc){
    name1 <- levels(dset[,v_idx])[k]

    for(i in 1:nc){
```

```
      for(j in 1:nc){
        if(j != i){
          votes[,k] <- votes[,k] + (preds[[i]][[j]] == name1) * 1
        }
      }
    }
  }
  return(votes)
}

one_against_one <- function(v_idx, dset, train, predicted){
  tmpname <- (names(dset))[v_idx]
  varname <- 'mytempname'
  names(dset)[v_idx] <- 'mytempname'

  list_of_preds <- list()
  tmplist <- list()
  acc <- matrix(-1, nrow = ncol(dset), ncol = ncol(dset))
  len <- length( levels(dset$mytempname) )

  for(i in 1:len ){
    for(j in 1:len ){
      if(i != j){
        tmplist[[j]] <- comp_preds(i, j, v_idx, train, dset)
      }else{
              tmplist[[j]] = rep(-1, nrow(dset) - length(train))
      }
```

```
    }
    list_of_preds[[i]] <- tmplist
  }
  votes <- comp_votes(list_of_preds, dset, v_idx)
  one_against_one_pred <- (levels(dset$mytempname))[apply(votes,1,which.max)]
  return(table(predicted, one_against_one_pred))
}

data(fgl,package='MASS')
tab2 <- one_against_one(ncol(fgl), fgl, train, fgl$type[-train])
```