Homework 4

Helga Mazyar

October 1, 2016

1.
```
  library("data.table")
library("DescTools")
##
## Attaching package:  'DescTools'
## The following object is masked from 'package:data.table':
##
##    %like%

library("plyr")
#setwd("C:/GoogleDrive/Fall_2016/PSYCH 625/Project2")
setwd("C:/Users/hemaz/Google Drive/Fall_2016/PSYCH 625/Project2")

############## training
train <- function (Y, X) {
  posterior = vector(mode = "list", length = ncol(X))

  for (i in 1:ncol(X)){
    t = table(Y, X[,i] )
    for (j in 1:nrow(t)) t[j,]= t[j,]/sum(t[j,])

    posterior[[i]] = t;
  }

  # calculate priors
  prior = rep(NA, nlevels(Y))
  for(i in 1:nlevels(Y)){
    prior[i] = sum(Y==levels(Y)[i])/length(Y)
  }

  return(classifier = list(prior=prior, posterior=posterior))
}
##########classify
classify <- function (classifier, query) {
  prob = classifier$prior
  for (i in 1:length(query)){
    t = classifier$posterior[[i]]
    prob = prob* t[,query[i]]
  }
  return(prob)


}
```

```r
NB <- function(formula, training_set, query){
  if (!is.data.frame(training_set)) stop("NB can only handle data frames")
  if (!is.vector(query)) stop ("Query needs to be a vector")

  f1.parsed <- ParseFormula(formula,data=training_set[1,])
  Y <- f1.parsed$lhs$vars
  X <- f1.parsed$rhs$vars
  a = training_set[,X]
  #make sure query is in the correct format
  for (i in 1:length(X)){

    b = levels(a[,i]) #loop over X colomns
    if (!query[i] %in% b) stop("Query does not match the model")
  }
  classifier <- train(training_set[,Y], training_set[,X])

  prob = classify(classifier, query)
  ind = which.max(prob)
  cat(levels(training_set[,Y])[ind], prob[ind])
  return(levels(training_set[,Y])[ind])
}
#
data <-read.csv('Tennis.csv')
data = data[,-1]
# # #
# #
formula <- Play ~ Outlook + Temperature + Humidity + Wind
# # #
query <- c("Sunny", "Cool", "High", "Strong")
NB(formula, data, query)

## No 0.02057143
## [1] "No"

query <- c("Sunny", "Cool", "Normal", "Weak")
NB(formula, data, query)

## Yes 0.02116402
## [1] "Yes"

query <- c("Hot", "Cool", "Normal", "Weak")
NB(formula, data, query)

## Error in NB(formula, data, query):  Query does not match the model

NB(formula, c(1,2), query)

## Error in NB(formula, c(1, 2), query):  NB can only handle data frames

query <- data[data$Outlook == "Sunny", ]
NB(formula, data, query)

## Error in NB(formula, data, query):  Query needs to be a vector
```

```
##########House data
houseData <- read.csv("house-votes-84.data",na.strings="?")
# colnames(houseData) <- c("party", "handicapped-infants",
#                          "water-project-cost-sharing",
#                          "adoption-of-the-budget-resolution",
#                          "physician-fee-freeze",
#                          "el-salvador-aid",
#                          "religious-groups-in-schools",
#                          "anti-satellite-test-ban",
#                          "aid-to-nicaraguan-contras",
#                          "mx-missile",
#                          "immigration",
#                          "synfuels-corporation-cutback",
#                          "education-spending",
#                          "superfund-right-to-sue",
#                          "crime", "duty-free-exports",
#                          "export-administration-act-south-africa")
colnames(houseData) = c("party", LETTERS[seq( from = 1, to = 16 )])

mydata <- houseData[complete.cases(houseData),]
formula <- party ~ .
query <- c(as.vector(t(mydata[2,-1])))
NB(formula,mydata[-2,],query)

## republican 0.009148421
## [1] "republican"
```

2. ```
   LOOCV <- function(formula, data){
     f1.parsed <- ParseFormula(formula,data=data[1,])
     Y <- f1.parsed$lhs$vars
     X <- f1.parsed$rhs$vars
     res = rep(NA, dim(data)[1])
     for (i in 1:dim(data)[1]){
       trainig_set = data[-i,]
       test = data[i,X]
       test = as.vector(t(test))
       res[i] = NB(formula, trainig_set, test);
     }
     perf = mean(data[,Y] == res)
     cat('\n', perf)

   }
   data <-read.csv('Tennis.csv')
   data = data[,-1]
   query <- c("Sunny", "Cool", "High", "Strong")
   formula <- Play ~ Outlook + Temperature + Humidity + Wind
    LOOCV(formula, data)

   ## Yes 0.007597341No 0.01442308Yes 0.004507212No 0.01969231Yes 0.01502404Yes 0.01709402Yes 0.009014
   ##  0.5714286

    formula <- Outlook ~ Temperature + Humidity + Wind + Play
    LOOCV(formula,data)
   ```
   ```

```
## Sunny 0.009615385Sunny 0.004807692Sunny 0.02215385Sunny 0.022153850vercast 0.01923077Sunny 0.007
##  0.1428571

 formula <- party ~.
 LOOCV(formula,mydata)

## democrat 5.417018e-07republican 0.009148421democrat 0.001303638democrat 0.001261585democrat 0.00
##  0.9181034
```

3. One way to avoid long runtime, is to build the classifer one time and use it for classifying all the queries (instead of building it per query). NB and LOOCV were modified to achive this.

```
 library("tm")

## Loading required package:  NLP

  train_fast <- function (Y, X) {
  print("Training the classifier")
  prob = array(NA, c(nlevels(Y),nlevels(as.factor(X)) , ncol(X)))
  eps = 1/ncol(X)
  for (i in 1:ncol(X)){
    t = table(Y, X[,i] )
    t = t+eps # add-one smoothing
    for (j in 1:nrow(t)) t[j,]= t[j,]/sum(t[j,])
    prob[,,i] =t
  }

  # calculate priors
  prior = rep(NA, nlevels(Y))
  for(i in 1:nlevels(Y)){
    prior[i] = sum(Y==levels(Y)[i])/length(Y)
  }
  classifier = list(prior=prior, posterior=prob)
  return(classifier)
}

######################
NB_fast <- function(classifer, Y, X, training_set, query){
  #classify
  prob = classifer$prior
  for (i in 1:length(query)){
    prob = prob* classifer$posterior[,query[i],i]
  }
  ind = which.max(prob)
  return (levels(training_set[,Y])[ind])
}
###########
cleanup <- function(text){
  docs <- Corpus(VectorSource(text))
  # Twitter tags
  tt<-function(x) gsub("RT |via", "", x)
  docs<- tm_map(docs, content_transformer(tt))
```

```
  # Twitter Usernames
  tun<-function(x) gsub("(^|[^@\\w])@(\\w{1,15})\\b", "", x)
  docs<- tm_map(docs, content_transformer(tun))


  # URLs
  urlPat<-function(x) gsub("(ftp|http)(s?)://.*\\b", "", x)
  docs <- tm_map(docs, content_transformer(urlPat))

  # Convert the text to lower case
  docs <- tm_map(docs, content_transformer(tolower))

  # Remove numbers
  docs <- tm_map(docs, removeNumbers)
  # Remove english common stopwords
  docs <- tm_map(docs, removeWords, stopwords("english"))
  # Remove your own stop word
  # specify your stopwords as a character vector
  docs <- tm_map(docs, removeWords, c("blabla1", "blabla2"))
  # Remove punctuations
  docs <- tm_map(docs, removePunctuation)
  # Eliminate extra white spaces
  docs <- tm_map(docs, stripWhitespace)

  return(docs)
}
LOOCV_fast <- function(classifier, Y, X, data, n){

  if (!is.data.frame(data)) stop("NB can only handle data frames")
  print("Running LOOCV")

  res = rep(NA, dim(data)[1])

  for (i in 1:dim(data)[1]){
    t0 = Sys.time()
    trainig_set = data[-i,]
    test = factor(as.character(data[i,X]))
    levels(test) = c(1,2)
    res[i] = NB_fast(classifier, Y, X, trainig_set, test);
    t1 = Sys.time() - t0;
    if (i%%1000==1){
      cat("trial#", i, "time remaining = ", t1*(dim(data)[1]-i)/60, "min")
      print(i)
    }

  }
  perf = mean(data[,Y] == res)
  cat('\n perf=', perf)

}
#################
sentiment <-function(data){
```

```
    text = data[,2]
    TYPE = data[,1]
    docs = cleanup(text)

    #Build a term-document matrix
    dtm = TermDocumentMatrix(docs)
    #remove sparse terms
    dtm =removeSparseTerms(dtm, sparse=0.99)

    dtm = weightBin(dtm)
    m <- as.matrix(dtm)
    data = data.frame(t(m), TYPE)
    #build the calssifier
    classifier = train_fast(as.factor(TYPE), t(m))
    X = colnames(data[,-ncol(data)])
    Y = "TYPE"
    a = LOOCV_fast(classifier, Y, X, data, n)

}

load("sentiment")
data <- subset(r,select=c("sentiment","text"))
sentiment(data)

## [1] "Training the classifier"
## [1] "Running LOOCV"
## trial# 1 time remaining =  3.013271 min[1] 1
## trial# 1001 time remaining =  3.007692 min[1] 1001
## trial# 2001 time remaining =  3.47136 min[1] 2001
## trial# 3001 time remaining =  2.63381 min[1] 3001
## trial# 4001 time remaining =  1.896829 min[1] 4001
## trial# 5001 time remaining =  2.511363 min[1] 5001
## trial# 6001 time remaining =  1.840922 min[1] 6001
## trial# 7001 time remaining =  1.54976 min[1] 7001
## trial# 8001 time remaining =  1.512785 min[1] 8001
## trial# 9001 time remaining =  1.180082 min[1] 9001
## trial# 10001 time remaining =  0.8730705 min[1] 10001
## trial# 11001 time remaining =  0.8605727 min[1] 11001
## trial# 12001 time remaining =  0.4395415 min[1] 12001
## trial# 13001 time remaining =  0.2035384 min[1] 13001
##
##  perf= 0.5787614

DT <- as.data.frame(
  lapply(subset(r, candidate=="Donald Trump"),
         function(x) if(is.factor(x)) factor(x) else x
  )
)
data <- subset(DT,select=c("sentiment","text"))
sentiment(data)

## [1] "Training the classifier"
## [1] "Running LOOCV"
```

```
## trial# 1 time remaining =  0.1671278 min[1] 1
## trial# 1001 time remaining =  0.1646551 min[1] 1001
## trial# 2001 time remaining =  0.04750198 min[1] 2001
##
##  perf= 0.6565944

JK <- as.data.frame(
  lapply(subset(r, candidate=="John Kasich"),
         function(x) if(is.factor(x)) factor(x) else x
  )
)
data <- subset(JK,select=c("sentiment","text"))
sentiment(data)

## [1] "Training the classifier"
## [1] "Running LOOCV"
## trial# 1 time remaining =  0.01208552 min[1] 1
##
##  perf= 0.661157
```