5241 HW4P3

Mengjia Huang

2018/4/3

Problem3

```
##Part1 Implement the AdaBoost algorithm
#from every iteration, we get a classifer(its parameters in output) and an alpha, the
 voting weights

AdaBoost<-function(X,y,B){
  n<-nrow(X)
  w<-rep(1/n, n)
  alpha<-rep(0,B)
  allpars<-rep(list(list()),B)

  for(b in 1:B){
    #step 1 train the base classifier
    allpars[[b]]<-train(X,w,y)
    #step 2 compute the error
    misclass<-as.numeric(y!=classify(X,allpars[[b]]))
    error<-(w %*% misclass/sum(w))[1]
    #step 3 computing voting weight
    alpha[b]<-log((1-error)/error)
    #step 4 reassgin the weight
    w<-w*exp(alpha[b]*misclass)
  }
  return(list(allpars=allpars, alpha=alpha))
}

#evaluates the boosting classifier ("aggregated classifier")
#output:the estimated class labels
agg_class<-function(X,alpha,allpars){
  n<-nrow(X)
  B<-length(alpha)
  labels<-matrix(0,nrow=n,ncol=B)

  for(b in 1:B){
    labels[,b]<-classify(X,allpars[[b]])
  }
  #weight classifier response by alpha
  labels<-labels %*% alpha
  est<-sign(labels)
  return(est)
}
```

```
##Part2 Implement the functions train and classify for decision stumps.
#train the weak learner (decision stump)
#X is a matrix the columns of which are the training vectors x(1) , . . . , x(n)
#w and y are vectors containing the weights and class labels
train<-function(X,w,y){
  n<-nrow(X)
  p<-ncol(X)

  mm<-rep(0,p)
  tt<-rep(0,p)
  err<-rep(0,p)
  ?rev

  for (j in 1:p){
    index <- order(X[,j]) #ascending
    x.j<- X[index,j] #sort data points along dim j
    revx.j<-X[rev(index),j] #decending

    w.cum<-cumsum(w[index]*y[index]) #sum of weight on the left side of the threshol
d.
    w.cum[rev(duplicated(revx.j)==1)]<-NA #threshold point should not lie between two
 same elements.

    #find optimal threshold (theta) and classify accordingly
    #θj* is the global extremum of the cumulative sum
    m<-max(abs(w.cum),na.rm = TRUE) #find the max cumsum and remove repeats
    maxindex<-min(which(abs(w.cum)==m)) #find the index that first time reach the max
 cumsum
    mm[j]<-(w.cum[maxindex]<0)*2-1 #1*2-1=1 or 0*2-1=-1 #just for T and F
    tt[j]<-x.j[maxindex]
    class<-((X[,j]>tt[j])*2-1)*mm[j]
    err[j]<-w %*% (class!=y)
  }
  #find optimal dim
  m<-min(err)
  j.opt<-min(which(err==m)) #find the index that first time reach the min error

  pars<-list(j=j.opt,theta=tt[j.opt],mode=mm[j.opt])
  return(pars)
}


#evaluates the weak learner on X using the parametrization pars.
#output: the predicted test point classification
classify<-function(X,pars){
 label<-ifelse(X[,pars$j]>pars$theta, pars$mode,-pars$mode)
 return(label)
}
```

```
##part 3 Run algorithm on USPS data, evaluate results using cross validation
train.3<-read.table("train_3.txt",header = FALSE, sep=",")
train.8<-read.table("train_8.txt",header = FALSE, sep=",")
xtrain<-rbind(as.matrix(train.3),as.matrix(train.8))
ytrain<-as.matrix(rep(c(-1,1),c(nrow(train.3),nrow(train.8))))
dim(xtrain)
```

```
## [1] 1200  256
```

```
test<-as.matrix(read.table("zip_test.txt"))
ytest<-test[,1]
xtest<-test[ytest==3|ytest==8,-1]
ytest<-as.matrix(ytest[ytest==3|ytest==8])
ytest[ytest==3]<--1
ytest[ytest==8]<-1
dim(xtest)
```

```
## [1] 332 256
```

```
XX<-rbind(xtrain,xtest)
YY<-rbind(ytrain,ytest)
n<-nrow(XX)
dim(XX)
```

```
## [1] 1532  256
```

```
dim(YY)
```

```
## [1] 1532    1
```

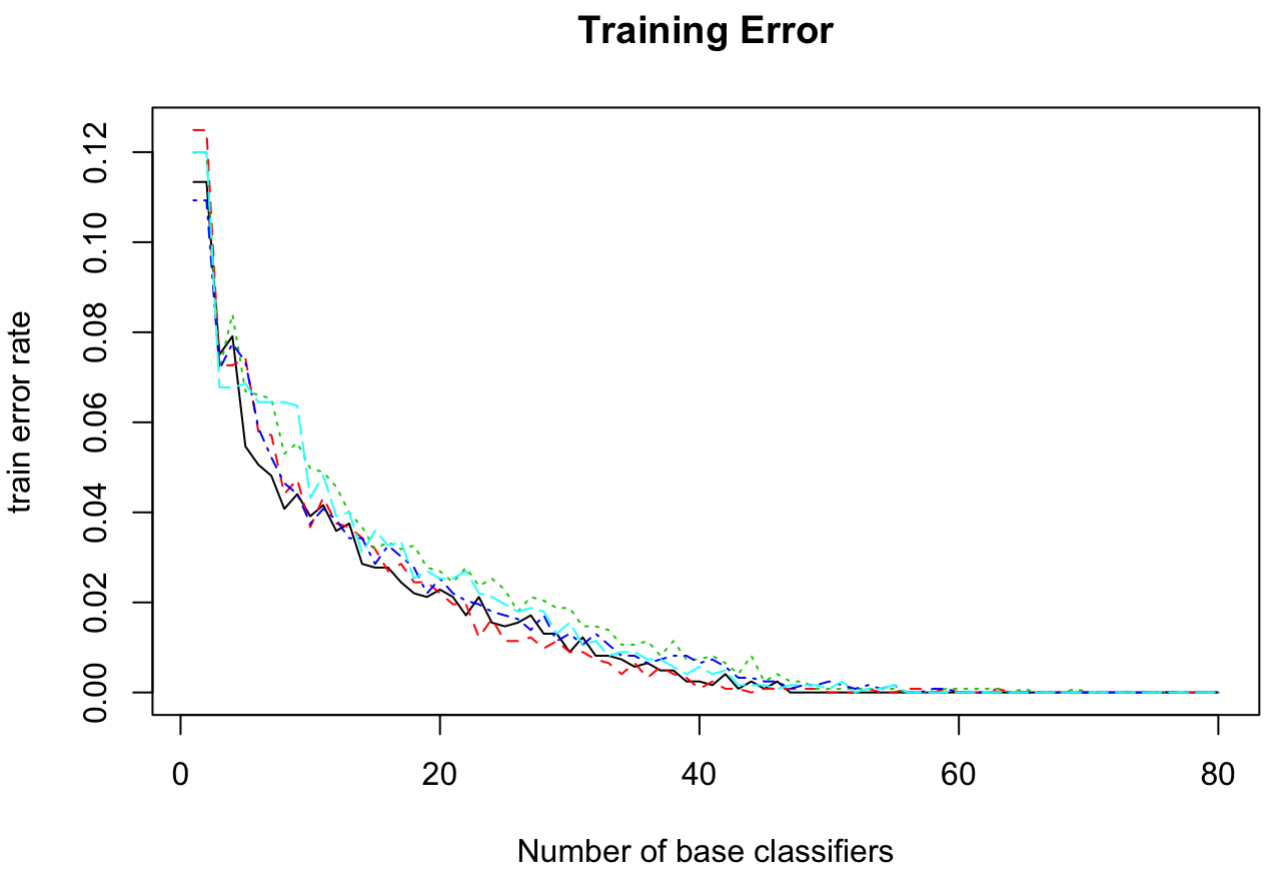
```
set.seed(1098)
Bmax<-80
K<-5

testerror<-matrix(0,nrow=Bmax,ncol=K)
trainerror<-matrix(0,nrow=Bmax,ncol=K)

nums<-rep(1:K,each=round(n/K))
fold<-sample(nums) #shuffle
for(i in 1:K){
  xtest<-XX[fold==i, ]
  xtrain<-XX[fold!=i, ]
  ytest<-YY[fold==i]
  ytrain<-YY[fold!=i]

  #use train data to train the agg classifer
  ada<-AdaBoost(xtrain,ytrain,Bmax)
  allpars<-ada$allpars
  alpha<-ada$alpha

  for(B in 1:Bmax){
    testesterror<-agg_class(xtest,alpha[1:B],allpars[1:B])
    testerror[B,i]<-mean(ytest!=testesterror)
    trainesterror<-agg_class(xtrain,alpha[1:B],allpars[1:B])
    trainerror[B,i]<-mean(ytrain!=trainesterror)
  }
}
head(trainerror)
```

```
##            [,1]       [,2]       [,3]       [,4]       [,5]
## [1,] 0.11337684 0.12489796 0.11990212 0.10929853 0.12000000
## [2,] 0.11337684 0.12489796 0.11990212 0.10929853 0.12000000
## [3,] 0.07504078 0.07265306 0.07177814 0.07177814 0.06775510
## [4,] 0.07911909 0.07265306 0.08401305 0.07748777 0.06775510
## [5,] 0.05464927 0.07428571 0.06688418 0.07340946 0.06857143
## [6,] 0.05057096 0.05795918 0.06606852 0.05872757 0.06448980
```

```
head(testerror)
```

```
##            [,1]       [,2]       [,3]       [,4]       [,5]
## [1,] 0.13398693 0.08794788 0.10784314 0.15032680 0.10749186
## [2,] 0.13398693 0.08794788 0.10784314 0.15032680 0.10749186
## [3,] 0.08169935 0.05537459 0.06862745 0.06862745 0.08469055
## [4,] 0.10130719 0.05537459 0.06862745 0.08823529 0.08469055
## [5,] 0.08496732 0.06188925 0.06535948 0.08169935 0.08143322
## [6,] 0.08823529 0.05537459 0.05555556 0.09803922 0.07817590
```

```
##Part4 Plot the training error and the test error as a function of b.
#matplot:plot the columns of one matrix
matplot(trainerror,type="l",col=1:K,main="Training Error",xlab="Number of base classi
fiers",ylab="train error rate")
```




```
matplot(testerror,type="l",col=1:K,main="Test Error",xlab="Number of base classifier
s",ylab="test error rate")
```

As expected, the training error reach 0 as the number of the weak classifer increase. The test error becomes stable after b=40, correspondly with training error.