assignment2_wuhao

Group 8

12/10/2021

Assignment 2

```
## Warning: package 'kernlab' was built under R version 4.1.1
```

```
## [1] "err0: 0.0675"
```

```
## [1] "err1: 0.0848938826466916"
```

```
## [1] "err2: 0.0823970037453183"
```

```
## [1] "err3: 0.0212234706616729"
```

Question 1

Filter 0 or filter 1 should be return, since when we find the optimal C, we should still use train as training data, so filter 0 and filter 1 all satisfies this condition.

Question 2

Error 1 should be return to the user. We are training the model in the train data,and using the valid data to find the optimal hyper parameter, so test of generalization error should be done on test data. And only error 1 satisfies this condition.

Question 3

According to the requirement, we find needed parameter from given function and model by using the code below.

`spam_matrix <- as.matrix(spam[,-58])` The original data without label.

`alpha<-alphaindex(filter3)[[1]]` Index of support vector.

`support_x <- as.matrix(spam[alpha,-58])` Support vector data without label.

`coef <- coef(filter3)[[1]]` Coefficient times label of each support vector.

`b <- b(filter3)` Bias of decision formula.

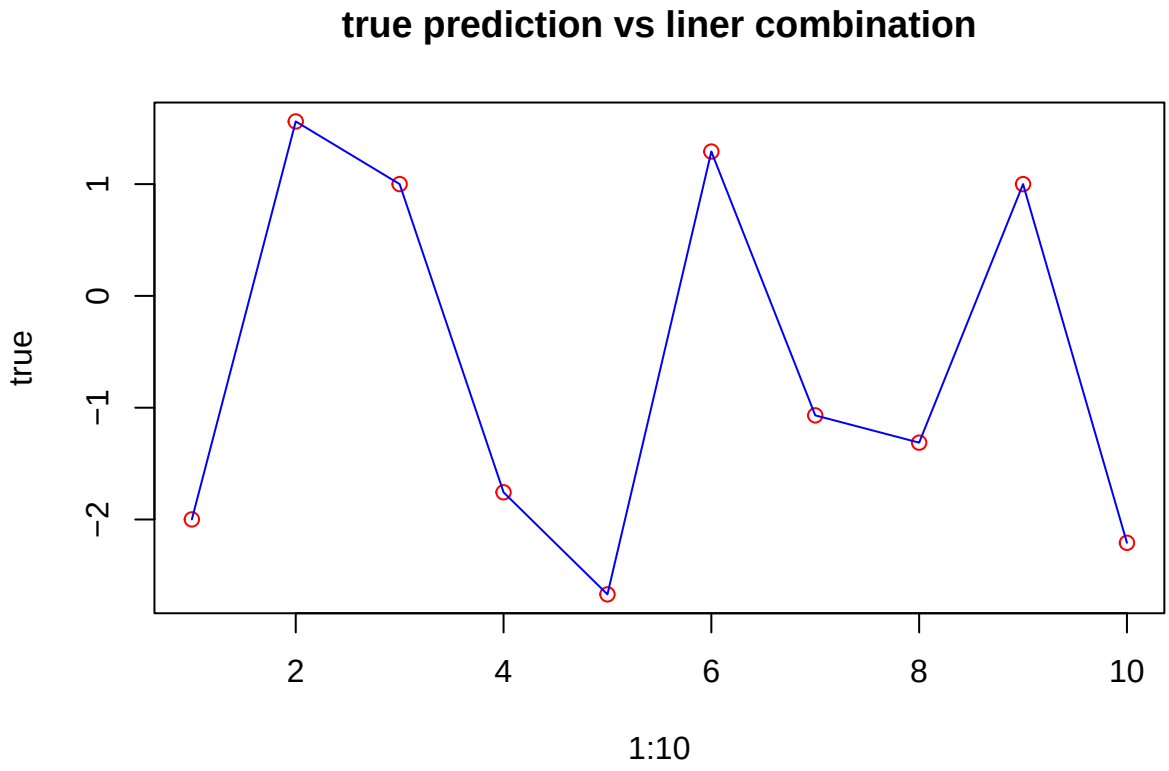
The formula (not decision function) to compute decision value is like :

$$res = [\sum_{i=1}^{Ns} \alpha_i y_i K(x_i, x^*)] + b$$

where Ns is the number of support vector, α_i is the coefficient, y_i is label, K is kernel function,x_i is data(without label) of support vector, x^* is the point we want to predict. To be more specific, the kernel function is :

$$\exp(-\sigma \|x - x^*\|^2)$$

where σ is the hyper parameter, $||x - x^*||^2$ is the L2 distance of two given points.



hints : *red point is from predict() function and blue line is from our liner combination*

From the plot above, we can see that the liner combination has the same function of `predict()`, which means our result is correct.

APPENDIX

```
library(kernlab)
set.seed(1234567890)
data(spam)
foo <- sample(nrow(spam))
spam <- spam[foo,]
```

```
spam[,-58]<-scale(spam[,-58])
train <- spam[1:3000, ]
valid <- spam[3001:3800, ]
train_valid <- spam[1:3800, ]
test <- spam[3801:4601, ]

by <- 0.3
err_va <- NULL
for(i in seq(by,5,by)){
  filter <- ksvm(type~.,data=train,kernel="rbfdot",kpar=list(sigma=0.05),C=i,scaled=FALSE)
  mailtype <- predict(filter,valid[,-58])
  t <- table(mailtype,valid[,58])
  err_va <-c(err_va,(t[1,2]+t[2,1])/sum(t))
}

filter0 <- ksvm(type~.,data=train,kernel="rbfdot",kpar=list(sigma=0.05),C=which.min(err_va)*by,scaled=FA
mailtype <- predict(filter0,valid[,-58])
t <- table(mailtype,valid[,58])
err0 <- (t[1,2]+t[2,1])/sum(t)
print(paste0('err0: ',err0))

filter1 <- ksvm(type~.,data=train,kernel="rbfdot",kpar=list(sigma=0.05),C=which.min(err_va)*by,scaled=FA
mailtype <- predict(filter1,test[,-58])
t <- table(mailtype,test[,58])
err1 <- (t[1,2]+t[2,1])/sum(t)
print(paste0('err1: ',err1))

filter2 <- ksvm(type~.,data=train_valid,kernel="rbfdot",kpar=list(sigma=0.05),C=which.min(err_va)*by,sca
mailtype <- predict(filter2,test[,-58])
t <- table(mailtype,test[,58])
err2 <- (t[1,2]+t[2,1])/sum(t)
print(paste0('err2: ',err2))

filter3 <- ksvm(type~.,data=spam,kernel="rbfdot",kpar=list(sigma=0.05),C=which.min(err_va)*by,scaled=FAL
mailtype <- predict(filter3,test[,-58])
t <- table(mailtype,test[,58])
err3 <- (t[1,2]+t[2,1])/sum(t)
print(paste0('err3: ',err3))
##################################################
#q3 liner combination
##################################################
spam_matrix <- as.matrix(spam[,-58])
alpha=alphaindex(filter3)[[1]]
support_x <- as.matrix(spam[alpha,-58])
coef <- coef(filter3)[[1]]
b <- b(filter3)

pre_svm <- function(i)
{
  temp <- t(apply(support_x,1,function(x) x-spam_matrix[i,-58]))
  temp <- exp(-0.05*apply(temp**2,1,sum))
  res <- sum(coef*temp)-b
  return(res)
```

```
}
my_res <- c()
for (i in 1:10)
{
  my_res <- c(my_res,pre_svm(i))
}
true <- predict(filter3,spam[1:10,-58],type='decision')
plot(1:10,true,col='red')
lines(1:10,my_res,col='blue')
title('true prediction vs liner combination')
```