Laboration report in Machine Learning

Computer lab 3 block 1

732A99

Simge Cinar
Duc Tran
William Wiik

Division of Statistics and Machine Learning
Department of Computer Science
Linköping University

09 December 2023

Contents

1 Assignment 2. Support Vector Machines

In this assignment, the data set "spam" from the R-package kernlab is used. The dataset consists of 4601 observations, 1 response variable labelling the observation spam or not spam, and 57 numerical variables.

1.1 Question 2.1

Question: Which filter do you return to the user? *filter0*, *filter1*, *filter2* or *filter3*? Why?

Answer: In this task data was divided into four different data sets:

- Train (3000 observations)
- Validation (800 observations)
- Test (801 observations)
- Train+Validation (3800 observations)

where the last dataset is training and validation combined. The code used to train each filter is as follows:

```
# All the following code is supplied by the task.
#-----------------------------------------------------------------#
library(kernlab)
set.seed(1234567890)

data(spam)
foo <- sample(nrow(spam))
spam <- spam[foo,]
spam[,-58]<-scale(spam[,-58])
tr <- spam[1:3000, ]
va <- spam[3001:3800, ]
trva <- spam[1:3800, ]
te <- spam[3801:4601, ]


# Finds the best value of C by using validation data
by <- 0.3
err_va <- NULL
for(i in seq(by,5,by)){
  filter <- ksvm(type~.,data=tr,kernel="rbfdot",kpar=list(sigma=0.05),C=i,scaled=FALSE)
  mailtype <- predict(filter,va[,-58])
  t <- table(mailtype,va[,58])
  err_va <-c(err_va,(t[1,2]+t[2,1])/sum(t))
}

filter0 <- ksvm(type~.,data=tr,kernel="rbfdot",kpar=list(sigma=0.05),C=which.min(err_va)*by,scaled=FALSE
mailtype <- predict(filter0,va[,-58])
t <- table(mailtype,va[,58])
err0 <- (t[1,2]+t[2,1])/sum(t)
```

```
filter1 <- ksvm(type~.,data=tr,kernel="rbfdot",kpar=list(sigma=0.05),C=which.min(err_va)*by,scaled=FALSE
mailtype <- predict(filter1,te[,-58])
t <- table(mailtype,te[,58])
err1 <- (t[1,2]+t[2,1])/sum(t)

filter2 <- ksvm(type~.,data=trva,kernel="rbfdot",kpar=list(sigma=0.05),C=which.min(err_va)*by,scaled=FAL
mailtype <- predict(filter2,te[,-58])
t <- table(mailtype,te[,58])
err2 <- (t[1,2]+t[2,1])/sum(t)

filter3 <- ksvm(type~.,data=spam,kernel="rbfdot",kpar=list(sigma=0.05),C=which.min(err_va)*by,scaled=FAL
mailtype <- predict(filter3,te[,-58])
t <- table(mailtype,te[,58])
err3 <- (t[1,2]+t[2,1])/sum(t)
#-----------------------------------------------------------#
```

From the code, the value of C for each filter was found by using the validation data set. In table X, the data the filters were trained on, the data used to estimate of the generalization error, and the error are presented.

Table 1: Summarized information about filters

	Training data	Testing data	Error on testing data
Filter 0	training	validation	0.0675
Filter 1	training	test	0.0849
Filter 2	train+val	test	0.0824
Filter 3	all data	test	0.0212

From table X, filter 0 used training data to train, and tested the filter on validation data. Filter 0 can be returned but we will not be able to give an unbiased estimate of the generalization error.
Filter 1 used training data to train, and tested the filter on test data, filter 1 only used each observation of data once and the estimate of the generalization error will be unbiased.
Filter 2 used training and validation data to train. Since validation data was used to find the parameter c, filter 2 will be a bit more overfitted on the training data compared to filter 0 and 1.
Filter 3 used all data to train, this means the model is a overfitted on training data and we will not be able to give an unbiased estimate of the generalization error.

In conclusion, the best filter to return is filter 1 since this is the only filter that only used each observation once.

1.2 Question 2.2

Question: What is the estimate of the generalization error of the filter returned to the user? *err0*, *err1*, *err2* or *err3*? Why?

Answer: In question 2.1, we concluded that filter 0, filter 2, and filter 3 used each observation more than once, and therefore the filter can not give an unbiased estimate of the generalization error. The estimate of the generalization error of the filter should always be the error estimate of the filter we return. In this case filter 1 was chosen to be returned in question 2.1. The generalization error we returned is therefore 0.0849, which means that around 8.49% of future emails will be wrongly classified by this filter.

1.3 Question 2.3

Question: Once a SVM has been fitted to the training data, a new point is essentially classified according to the sign of a linear combination of the kernel function values between the support vectors and the new point. You are asked to implement this linear combination for *filter3*. You should make use of the functions *alphaindex*, *coef* and *b* that return the indexes of the support vectors, the linear coefficients for the support vectors, and the negative intercept of the linear combination. See the help file of the *kernlab* package for more information. You can check if your results are correct by comparing them with the output of the function predict where you set type = "decision". Do so for the first 10 points in the *spam* dataset. Feel free to use the template provided in the *Lab3Block1 2021 SVMs St.R file*.

Answer: In SVM a new observation ($x_\star$) is predicted as:

$$\hat{y}(\mathbf{x}_\star) = \hat{\alpha}^T \mathbf{K}(\mathbf{X}, \mathbf{x}_\star) \tag{1}$$

where $\hat{\alpha}$ are the coefficients for the support vectors and $\mathbf{K}(\mathbf{X}, \mathbf{x}_\star)$ is the kernel values between the support vectors and the observation $x_\star$. The kernel function used for *filter 3* is the Gaussian RBF kernel which is:

$$K(x, x_\star) = exp(-\sigma||x - x_\star||^2) \tag{2}$$

where $\sigma = 0.05$ was used.

```
# Support vectors
sv<-alphaindex(filter3)[[1]]
# Coefficients for the support vectors
co<-coef(filter3)[[1]]
# negative intercept
intercept <- - b(filter3)

# Support vector observations with response variable removed
x <- spam[sv, -58] # Remove y value.

# The 10 observations we need to predict
x_stars <- spam[1:10, -58]

# The Gaussian RBF kernel function
rbfkernel <- rbfdot(sigma = 0.05)
```

```r
k<-c()
for(i in 1:10){ # We produce predictions for just the first 10 points in the dataset.
  k2<-NULL
  for(j in 1:length(sv)){
    k2[j] <- co[j]*rbfkernel(unlist(x[j,]), unlist(x_stars[i,]))
  }
  prediction <- intercept + sum(k2)
  prediction
  k[i]<- prediction
}
```

```r
manual_pred <- k
function_pred <- predict(filter3,spam[1:10,-58], type = "decision")

table_data <- data.frame(manual_pred, function_pred)
colnames(table_data) <- c("Manual prediction", "Predict function")
kable(table_data, caption = "Comparision between manual prediction and the predict function for the firs
```

Table 2: Comparision between manual prediction and the predict function for the first 10 observations in the spam data set.

Manual prediction	Predict function
-1.998999	-1.998999
1.560584	1.560584
1.000278	1.000278
-1.756815	-1.756815
-2.669577	-2.669577
1.291312	1.291312
-1.068444	-1.068444
-1.312493	-1.312493
1.000183	1.000183
-2.208639	-2.208639

Comparing the values from manual prediction and the predict function we get the same value for the prediction of the first 10 observations.