Support Vector Machines

Mowniesh Asokan(mowas455)

12/12/2020

Support Vector Machines

svm is a supervised learning algorithm, which is widely used for classification algorithm.SVM is applicable for data that are linearly separable and for non-linear data it uses kernel method for classification.

- It classifies the two classes using hyperplane,The hyperplane should have the largest margin in a high dimensional space to separate a given data into a classes.The margin between the two classes represent the longest distance between the closest data point of the classes.
- Based on the number of the input features/variables, the decision boundary can be a line (if we had only 2 features) or a hyperplane. A hyperplane is an (N-1)-dimensional subspace for an N-dimensional space.
- SVM pick the decision boundary that maximizes the distance to the support vectors. The decision boundary is drawn in a way that the distance to support vectors are maximized. If the decision boundary is too close to the support vectors then, it will be sensitive to noise and not generalize well.

```
data(spam)

index <- sample(1:4601)
tr <- spam[index[1:3000], ]
va <- spam[index[3001:3800], ]
trva <- spam[index[1:3800], ]
te <- spam[index[3801:4601], ]
```

Error

Error= (c)Classification Error + Margin Error(soft margin)

- Margin help to increase the distance of the decision boundary to the support vectors
- Maximize the number of points that are correctly classified in the training set.

C-parameter

C is small, the penalty for misclassified data point is low so decision boundary with large margin is choosen at the expense of the great number of misclassifications. If c is large ,SVM tries to minimize the number of misclassifed samples and results in decision boundary with smaller margin.

Error value of Filter0

```
## [1] 0.07
```

```
##
## mailtype  nonspam spam
##   nonspam     479   35
##   spam         21  265
```

Error Value of Filter1

```
## [1] 0.08489388
```

```
##
## mailtype  nonspam spam
##   nonspam     446   50
##   spam         18  287
```

Error value of Filter2

```
## [1] 0.08364544
```

```
##
## mailtype  nonspam spam
##   nonspam     446   49
##   spam         18  288
```

Error Value of Filter3

```
## [1] 0.03370787
```

```
##
## mailtype  nonspam spam
##   nonspam     457   20
##   spam          7  317
```

Questions

1.

Which filter do we return to the user ? filter0, filter1, filter2 or filter3 ? Why ?

Filter2 is performing better compare with all the other filter in the list, because this filter is trained by trva data and it is tested by the te data that is unseen to the model.

In the filter0 ,the error value is very low compare with all other filters but the filter is trained by using the tr data and tested by using the validation va data.The c parameter in this filter choose by using the va data in the previous model.So only it performs better than the other filters.Even filter3 also gives good error value but the problem is filter3 is trained by original data and is tested by the data that is taken from the original one.

2.

What is the estimate of the generalization error of the filter returned ? err0, err1, err2 or err3 ? Why ?

Error1 is the generalized error of the all filters, because that filter is trained by the training data and while testing it is generalized to give a prediction and it is trained by unseen data at those stages.Even the best c parameter for this data is estimated by the filter is trained by using the training data.

In the RBFDOT kernel that decrease the distance between the two classes and project the values in the hyper plane. Distance between the class of data is separated by the euclidean distance identifier formula.

Appendix

```
knitr::opts_chunk$set(echo = TRUE)
library(kernlab)
set.seed(1234567890)

data(spam)

index <- sample(1:4601)
tr <- spam[index[1:3000], ]
va <- spam[index[3001:3800], ]
trva <- spam[index[1:3800], ]
te <- spam[index[3801:4601], ]

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)
  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)
mailtype <- predict(filter0,va[,-58])
t <- table(mailtype,va[,58])
err0 <- (t[1,2]+t[2,1])/sum(t)
err0
t
filter1 <- ksvm(type~.,data=tr,kernel="rbfdot",kpar=list(sigma=0.05),C=which.min(err_va)*by)
mailtype <- predict(filter1,te[,-58])
t <- table(mailtype,te[,58])
err1 <- (t[1,2]+t[2,1])/sum(t)
err1
t
filter2 <- ksvm(type~.,data=trva,kernel="rbfdot",kpar=list(sigma=0.05),C=which.min(err_va)*by)
mailtype <- predict(filter2,te[,-58])
t <- table(mailtype,te[,58])
err2 <- (t[1,2]+t[2,1])/sum(t)
err2
t
```

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