Data Mining II SVM

Allen Rahrooh

1/31/2020

Homework II Problem 2 Code

The error I had with my intial code was not defining the kernel correctly inside of the svm.fit function. The code is now corrected and outputting a 2x2 confusion table instead of a 2x2 confusion table.

```
library(quadprog)
library(Matrix)
library(readr)
rm(list=ls())
# Load in the data

#kernel approach

Housing.df = read.csv("BostonHousing.csv")

#########
#SVM
housing.y= Housing.df[,14]

housing.y[housing.y==0]=-1

housing.x= Housing.df[,c(1,3,5:12)]

X = housing.x

y = housing.y



svm.fit = function(X, y, C=NULL, sigma = NULL)
  {
  n.samples = nrow(X)
  n.features = ncol(X)
  K = matrix(rep(0, n.samples*n.samples), nrow=n.samples)
  for (i in 1:n.samples){
    for (j in 1:n.samples){
      K[i,j] = exp(-sum((unlist(X[i,]) - unlist(X[j,]))^2)*sigma)
    }
  }
  Dmat = outer(y,y) * K
  Dmat = as.matrix(nearPD(Dmat)$mat)
```

```
  dvec = rep(1, n.samples)
  Amat = rbind(y, diag(n.samples), -1*diag(n.samples))
  bvec = c(0, rep(0, n.samples), rep(-C, n.samples))
  res = solve.QP(Dmat,dvec,t(Amat),bvec=bvec, meq=1)
  a = res$solution
  bomega = apply(a*y*X,2,sum)
  return(bomega)
}


standardize = function(z) (z-mean(z))/sd(z)

for(j in 1:dim(housing.x)[2]) X[,j] = standardize(housing.x[,j])

#X = cbind(1,housing.x)

y = housing.y


#for loop takes too long to run so I will omit it
#and just use the tune command from the e1071 package

#C = c(0.01,0.05,0.50)
#Sigma = c(0.5,0.85,0.67)
#acc = matrix(0,3,3)
#for (i in 1:3)
#{
#  for(j in 1:3)
#  {
#  housing.svm.betas = svm.fit(X,y, C = C[i] , sigma = Sigma[j])
#  y_pred = sign(as.matrix(X)%*%matrix(housing.svm.betas,(dim(housing.x)[2]),1))
#  acc[i,j] = sum(y==y_pred)/length(y)
#}

#}

#acc
```

```
#combining x and y to see the support vectors

housing <- cbind2(X,y)
library(e1071)

#default parameters Cost = 1 Gamma = 0.1
svm(formula = y ~., data = housing,  kernel = "radial" )
```

```
##
## Call:
## svm(formula = y ~ ., data = housing, kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  eps-regression
```

```
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.1
##     epsilon:  0.1
##
##
## Number of Support Vectors:  228
```

We see with default parameters of Cost = 1 and Gamma = 0.1 that we get 228 support vectors.

```r
#fitting default parameters to Quadratic Programming Algorithm
housing.svm.betas = svm.fit(X,y, C = 1 , sigma = 0.1)
y_pred = sign(as.matrix(X)%*%matrix(housing.svm.betas,(dim(housing.x)[2]),1))
table(y, pred = y_pred)
```

```
##     pred
## y      -1   1
##   -1 219 203
##   1    5  79
```

```r
acc <- sum(y==y_pred)/length(y) #accuracy
cat("The model accuracy is: ", acc)
```

```
## The model accuracy is:  0.5889328
```

We see a model accuracy of 58% with the default parameters.

We now proceed to tune Cost and Gamma using the tune command.

```r
#tuning cost and gamma
tune(svm, y ~., data = housing, ranges = list(cost = c(10,20,50), gamma = c(1,3,5)))
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##    10     1
##
## - best performance: 0.1789
```

We see that the best parameters are Cost = 10 and gamma = 1. We then fit these values to the quadratic programming function to generate a confusion table and the model accuracy.

```r
#fitting the tuned parameters to the svm.fit function
housing.svm.betas = svm.fit(X,y, C = 10 , sigma = 1)
y_pred = sign(as.matrix(X)%*%matrix(housing.svm.betas,(dim(housing.x)[2]),1))
table(y, pred = y_pred)
```

```
##     pred
## y      -1   1
##   -1 226 196
##   1    5  79
```

```r
acc <- sum(y==y_pred)/length(y) #accuracy
cat("The model accuracy is: ", acc)
```

```
## The model accuracy is:  0.6027668
```

We see the model accuracy has improved to 60%.

We then fit the tuned parameters to the svm command to see the difference in support vectors from the default parameters.

```r
svm(formula = y ~., data = housing,  kernel = "radial", cost = 10, gamma = 1 )
```

```
##
## Call:
## svm(formula = y ~ ., data = housing, kernel = "radial", cost = 10,
##     gamma = 1)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  10
##       gamma:  1
##     epsilon:  0.1
##
##
## Number of Support Vectors:  332
```

We see that after tuning there are 104 more support vectors generated.