Homework 4

Thanh Hung Duong

November 11, 2019

HW4 Part1 : SVM classification for Simulated Data

Question 1 : Generate a Data Set by Simulations

We seek to generate 5000 cases x1 … x5000 in R4 each case x = [x1 x2 x3 x4] has 4 numerical features

Step 1

Using random sampling of uniform distribution over the interval [-2, +2] to create:

- Aij 4x4 matrix
- Bi 1x4 matrix
- C

And define the polynomial of degree 2 in the 4 variables x1 x2 x3 x4 as follows

$$Pol(x) = \sum_i \sum_j A_{ij} x_i x_j + \sum_i B_i x_i + c/20$$

```
## [1] "Aij 4x4 matrix:"
```

```
##         [,1]   [,2]   [,3]    [,4]
## [1,] -1.0243  0.933  0.224  1.6710
## [2,]  0.0518  0.651 -0.704 -0.0622
## [3,] -1.8455  1.913  1.792 -1.2401
## [4,] -1.3353 -0.468 -0.373  0.1972
```

```
## [1] "Bi 4x1 matrix:"
```

```
##        [,1]
## [1,] -1.393
## [2,] -1.710
## [3,]  1.317
## [4,]  0.916
```

```
## [1] "C =  1.852"
```

```
## [1] "Test the polynomial function"
```

```
## [1] "If x = "
```

```
## [1] -0.381  0.885 -0.766  0.924
```

```
## [1] ",then Pol(x)= -0.5049"
```

If x= ,then Pol(x)= 3.5853095

Step 2

Using random sampling of uniform distribution over the interval [-2, 2] select 10000 vectors x1 … x10000 in R4 each such vector xn has 4 randomly chosen coordinates with values in [-2, 2]] for each selected xn compute U(n) = Pol(xn) and y(n) = sign[U(n)] define two classes by CL(1) = class1= set of all xn such that y(n) = +1 CL(-1) = class1= set of all xn such that y(n) = -1 keep only 2500 cases in CL(1) and 2500 cases in CL(-1), Center and Rescale this data set of size 5000 so that the standardized data set will have mean = 0 and dispersion =1 Then Split each class into a training set and a test set , using the proportions 80% and 20% this defines a training set TRAIN and a test set TEST of resp. sizes 4000 and 1000

Table 1: Head of TRAIN set

	X1	X2	X3	X4	Yn
4379	0.693	0.119	0.234	1.284	-1
4725	1.469	0.542	-0.766	-1.065	-1
46	-0.416	-0.214	1.630	-0.139	1
913	0.058	1.492	1.134	0.649	1
2369	-1.590	-1.647	-1.484	-0.238	1
4997	-1.308	1.493	-0.930	1.666	-1

Table 2: Head of TEST set

	X1	X2	X3	X4	Yn
1444	-1.309	0.190	0.492	-1.592	1
2827	-0.618	0.614	-0.680	-0.086	-1
596	0.805	1.377	1.357	-0.695	1
2628	0.483	-0.014	-0.394	-0.397	-1
4519	0.456	-0.742	-0.903	-0.842	-1
277	-0.959	-0.850	0.280	1.229	1

Question 2: SVM classification by linear kernel

- Fix arbitrarily the "cost" parameter in the svm() function, for instance cost = 5
- Select the kernel parameter kernel = "linear"
- Run the svm() function on the set TRAIN
- Compute the number S of support vectors and the ratio s = S/4000
- Compute the percentages of correct prediction PredTrain and PredTest on the sets TRAIN and TEST
- Compute two confusion matrices (one for the set TRAIN and one for the test set. Confusion matrices must be converted in terms of frequencies of correct predictions within each class
- Compute the errors of estimation on PredTRAIN, PredTEST, and on the terms of the confusion matrices interpret your results

```
################ A function to calculate the error of estimation
std_p = function(acc,n) {sqrt(acc*(1-acc)/n)*100}
```

```
############## A function to generate a nice confusion matrix
generate_cfm=function(predi,truec,cap){
  cfm=table(truec, predi)
  #colnames(cfm)=c('Pred_CL-1','Pred_CL1')
  #rownames(cfm)=c('True_CL-1','True_CL1')
  #print(cfm,caption = cap)
  nocc=rowSums(cfm)
  pocp=cfm/nocc
  kable(pocp,caption = paste('Confustion matrix for the set ',cap),digits = 3)
  acc=(cfm[1,1]+cfm[2,2])/length(truec)
  std=std_p(acc,length(truec))
```

```
  std_1=std_p(pocp[1,1],nocc[1])
  std_2=std_p(pocp[2,2],nocc[2])
  ac=c(acc,pocp[1,1],pocp[2,2])
  st=c(std,std_1,std_2)
  pred_table=data.frame(ac,st)
  rownames(pred_table) = c('Global accuracy','CL-1','CL1')
  colnames(pred_table) = c('Accuracy','Standard deviation')
  kable(pred_table,caption=paste('Prediction table for the set ',cap),digits = 3 )
}



N=5000
#x=TRAIN[,1:4]
#y=TRAIN$Yn
model =  svm(Yn~., cost=5, kernel='linear',type='C-classification',scale=FALSE,data=TRAIN)
#Alternative way to use svm without specify the type is to conver Y to factor
#dat = data.frame(x=x, y=as.factor(y)) # must convert to factor to avoid the svm regression
#model =  svm(y~., cost=5, kernel='linear',scale=FALSE,data=dat)
summary(model)
```

Call: svm(formula = Yn ~ ., data = TRAIN, cost = 5, kernel = “linear”, type = “C-classification”, scale = FALSE)

Parameters: SVM-Type: C-classification SVM-Kernel: linear cost: 5

Number of Support Vectors: 2498

(1250 1248)

Number of Classes: 2

Levels: -1 1

```
S=sum(model$nSV)
s=S/N
print(paste('Number S of support vectors is ',S))
```

[1] “Number S of support vectors is 2498”

```
print(paste('And the ratio s = ',s))
```

[1] “And the ratio s = 0.4996”

```
PredTRAIN =predict(model,TRAIN[,1:4])
PredTEST =predict(model,TEST[,1:4])
# Check accuracy:
generate_cfm(PredTRAIN, TRAIN$Yn,'Confusion matrix for the set TRAIN')
```

Table 3: Prediction table for the set Confusion matrix for the set TRAIN

	Accuracy	Standard deviation
Global accuracy	0.731	0.701
CL-1	0.756	0.960
CL1	0.705	1.020

```r
generate_cfm(PredTEST, TEST$Yn,'Confusion matrix for the set TEST')
```

Table 4: Prediction table for the set Confusion matrix for the set TEST

	Accuracy	Standard deviation
Global accuracy	0.752	1.366
CL-1	0.774	1.870
CL1	0.730	1.985