Birla Institute of Technology and Science

Foundations of Data Science (CS F320)

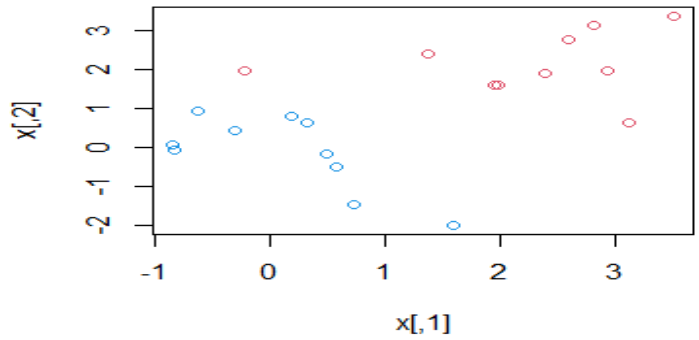
SECOND SEMESTER 2021-2022

Lab Sheet #6: Support Vector Machine

SVM is a supervised machine learning algorithm that helps in classification or regression problems. It aims to find an optimal boundary between the possible outputs.SVM does complex data transformations depending on the selected kernel function and based on that transformations, it tries to maximize the separation boundaries between your data points depending on the labels or classes defined.

#Create 2 dimensional test data

```
set.seed(1)
x <- matrix(rnorm(20*2), ncol=2)
y <- c(rep(-1,10), rep(1,10))
x[y==1,]=x[y==1,] + 1
plot(x,col=(3-y))
```



#install e1071 package which has several statistical learning methods

```
Install.packages("e1071")
library(e1071)
```

#Hard Margin Vs Soft Margin by adjusting the value of cost in svm method()

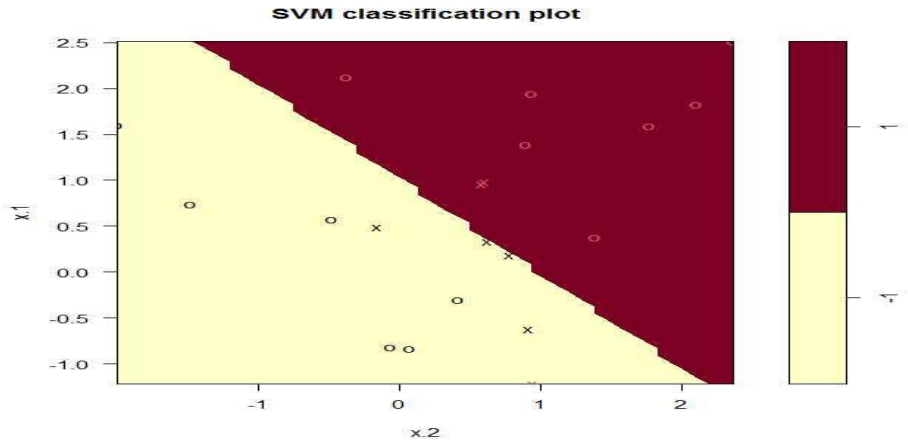
If our data is linearly separable, we go for a hard margin. However, if this is not the case, it won't be feasible to do that. In the presence of the data points that make it impossible to find a linear classifier, we would have to be more lenient and let some of the data points be misclassified. In this case, a soft margin SVM is appropriate.

#To perform classification using Linear SVM classifier:

- Use svm() method to fit a support vector classifier when the argument kernel='linear' is used.
- Specify the response as a factor.
- The cost argument allows us to specify the cost of a violation to the margin. Thus when the cost is small, the margins will be wide and there will be many support vectors.
- The argument scale = FALSE tells the function not to scale each feature.

#Linear SVM

```
dat<- data.frame(x=x, y=as.factor(y))
model1 <- svm(y ~., data=dat, kernel='linear', cost=10, scale=FALSE)
plot(model1, dat)
```



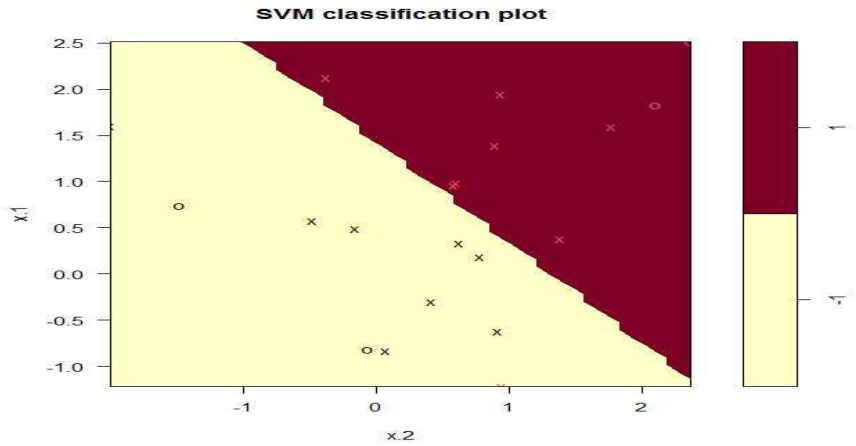

```
summary(model1)
```

Call:
svm(formula = y ~ ., data = dat, kernel = "linear", cost = 10, scale = FALSE)
Parameters:
SVM-Type: C-classification
SVM-Kernel: linear
cost: 10
Number of Support Vectors: 7
(4 3)
Number of Classes: 2
Levels:
-1 1

The summary lets us know there were 7 support vectors, four in the first class and three in the second.

What if we used a smaller cost parameter instead?

```
dat<- data.frame(x=x, y=as.factor(y))
model2 <- svm(y ~., data=dat, kernel='linear', cost=0.1, scale=FALSE)
plot(model2, dat)
```




```
summary(model2)
```

Call:
svm(formula = y ~ ., data = dat, kernel = "linear", cost = 0.1, scale = FALSE)
Parameters:
SVM-Type: C-classification
SVM-Kernel: linear
cost: 0.1
Number of Support Vectors: 16
(8 8)
Number of Classes: 2
Levels:
-1 1

Now analyze both plots and summary!!!

What is the cost effect on margin?

With a smaller value of cost we obtain more number of support vectors via the wider margin.

#Linear SVM to build the best model

In the summary you can see that cost = 0.1 results in the lowest error rate. tune() also stores the best model obtained accessed through $best.model, thus we can predict using test data. Here we create a simulated test set.

```
set.seed(1)
tune.out<- tune(svm, y ~., data=dat, kernel='linear',
```

```
            ranges=list(cost=c(0.001,0.01,0.1,1,5,10,100)))
summary(tune.out)

tune.out$best.model
```

#Prediction and Evaluation of Model

Predict the class labels of the test observations from the cross validated results.

```
yhat<- predict(tune.out$best.model, testdat)
library(caret)
#Evaluation of Model
confusionMatrix(yhat, testdat$y)
```

#Non-Linear SVM

Similarly change the kernel parameter of svm method as radial/polynomial/Sigmoid to understand the Non-Linear SVM model

Also, check how the cost changes the margin where cost > 0.

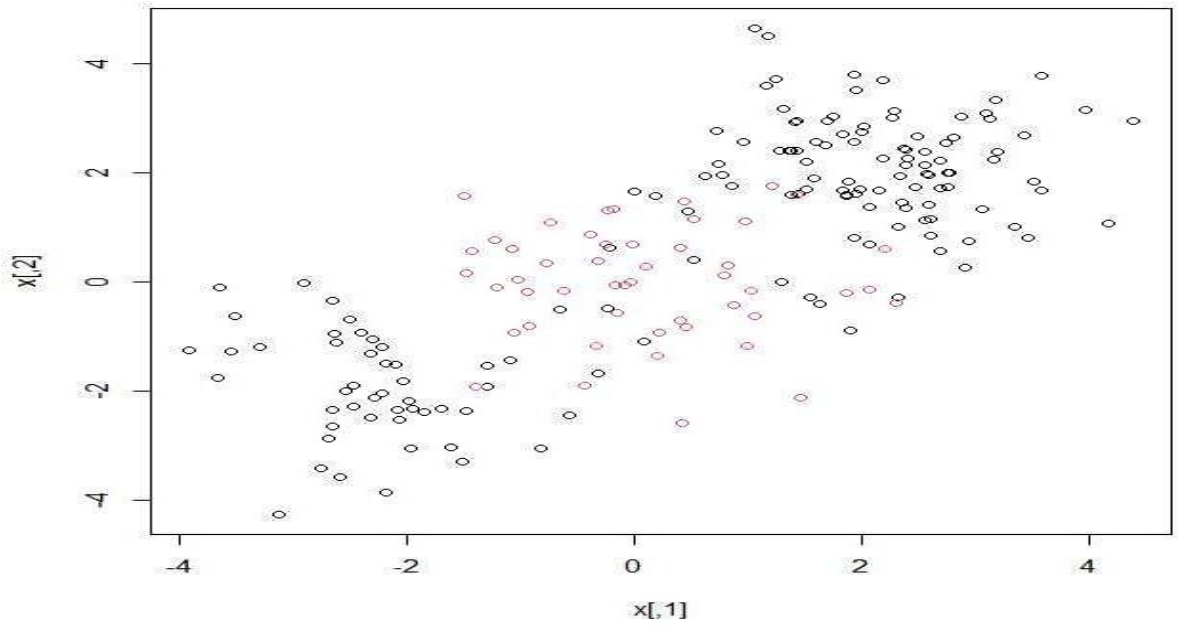
```
#Radial Kernel
model3 <- svm(y ~., data=dat, kernel='radial', cost=10, scale=FALSE)
plot(model3, dat)
summary(model3)

#Polynomial Kernel
model5 <- svm(y ~., data=dat, kernel='polynomial', cost=10, scale=FALSE)
plot(model5, dat)
summary(model5)

#Sigmoid Kernel
model7 <- svm(y ~., data=dat, kernel='sigmoid', cost=10, scale=FALSE)
# Plot the SVC obtained
plot(model7, dat)
summary(model7)
```

Non Linear SVM

Again we use the svm() function, however now we can experiment with non-linear kernels. For polynomial kernels we use the parameter degree to adjust the polynomial order. For radial kernels we use the gamma parameter to adjust the y value.

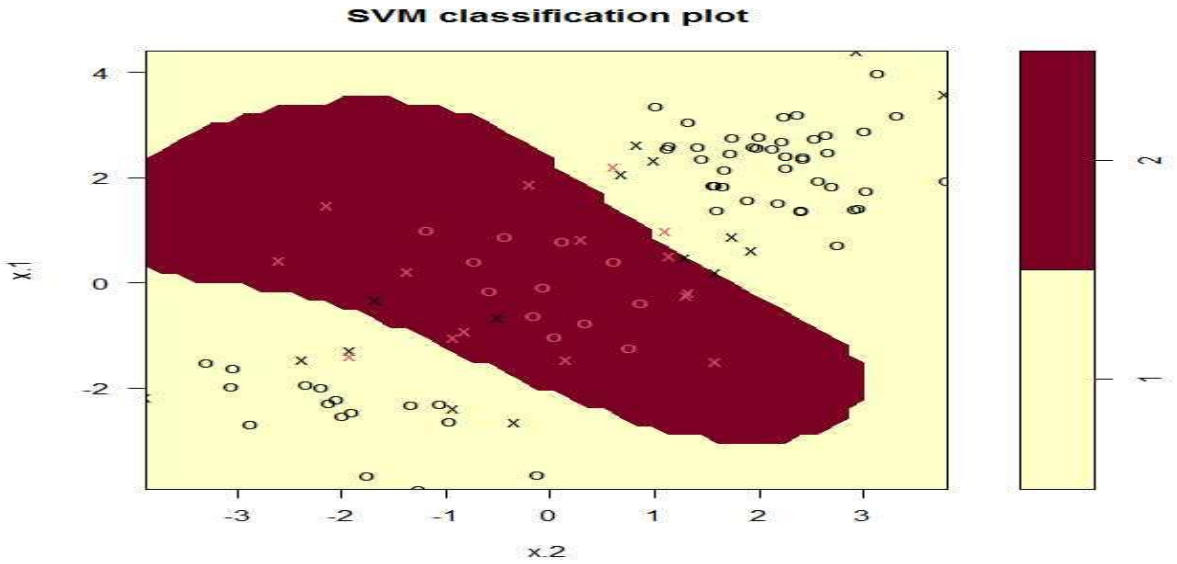
```
# Generate some test data
```

```
set.seed (1)
x <- matrix(rnorm(200*2), ncol=2)
x[1:100,]=x[1:100,]+2
x[101:150,]=x[101:150,]-2

y <- c(rep(1,150),rep(2,50))
dat<- data.frame(x=x,y=as.factor(y))

plot(x, col=y)
```



Randomly split the data into training and testing groups and fit a radial kernel.

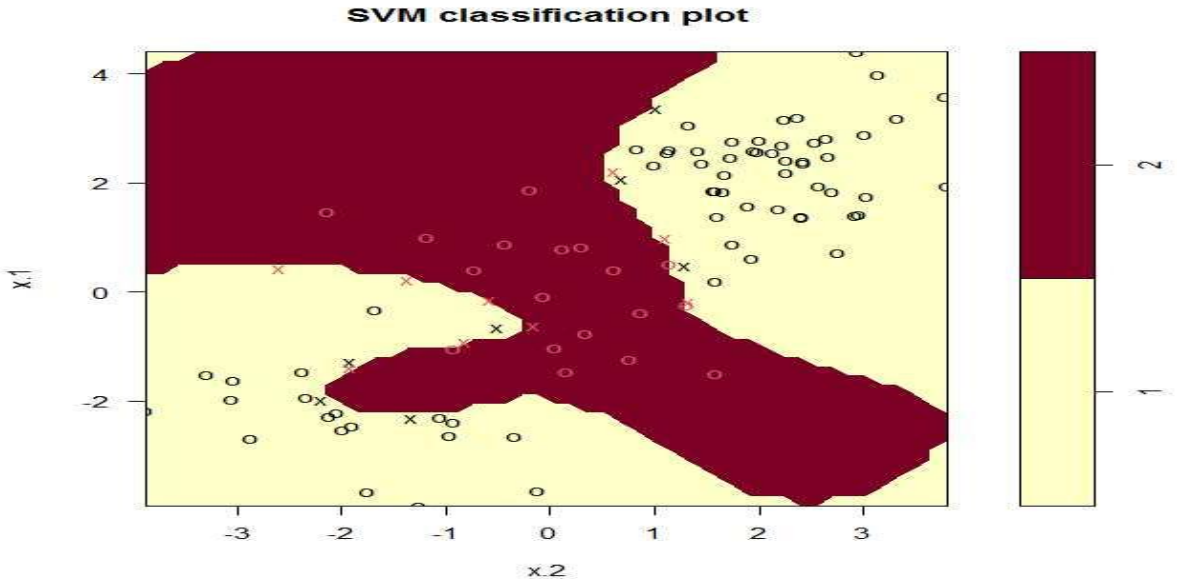
```
train <- sample(200, 100)
svm.fit<- svm(y ~., data=dat[train,], kernel='radial', gamma=1, cost=1)
plot(svm.fit, dat[train,])
```




```
summary(svm.fit)

library(caret)
yhat<- predict(svm.fit, dat[-train,]) # Predict test data
confusionMatrix(yhat, dat[-train,'y'])
```

If we increase the value of cost, we can reduce the training errors, but we risk overfitting the data.

```
svm.fit<- svm(y ~., dat[train,], kernel='radial', gamma=1, cost=1e5)
plot(svm.fit, dat[train,])
```



This certainly is very irregular, and probably would over fit the test data. Let's try cross validating these parameters instead.

```
set.seed(1)
tune.out<- tune(svm, y ~., data=dat[train,], kernel='radial', ranges = list(cost=c(0.1,1,10,100,1000),
            gamma=c(0.5, 1,2,3,4)))

summary(tune.out)

yhat<- predict(tune.out$best.model, dat[-train,])
confusionMatrix(yhat, dat[-train, 'y'])
```

Finally, if you are starting from scratch, it is perhaps best to begin with the SVM functions in the kernlab package. An interesting advantage of this package is that it was developed natively in R rather than C or C++, which allows it to be easily customized; none of the internals are hidden behind the scenes. Perhaps even more importantly, unlike the other options, kernlab can be used with the caret package, which allows SVM models to be trained and evaluated using a variety of automated methods

Train and Test SVM model for Classification on high Dimension Data

```
# import packages
#install.packages("kernlab")
library(kernlab)
#install.packages("hmeasure")
library(hmeasure)

# Load the Data
InputDataFileName="C:/Users/abc/Downloads/breastCancer.csv"
InputDataFileName

dataset <- read.csv(InputDataFileName)      # Read the datafile
dataset <- dataset[sample(nrow(dataset)),]  # Shuffle the data row wise.
head(dataset)   # Show Top 6 records
nrow(dataset)   # Show number of records
names(dataset)  # Show fields names or columns names

# Count total number of observations/rows.
cat("\nStep 4: Counting dataset")
totalDataset <- nrow(dataset)
totalDataset

# Partition the data into Train/Test set
training = 50      # Defining Training Percentage; Testing = 100 – Training

# Choose Target variable
target  <- names(dataset)[10]   # i.e. Cancer
```

```
target

# Choose inputs Variables
inputs <- setdiff(names(dataset),target)
inputs
length(inputs)

#Select Training Data Set
trainDataset <- dataset[1:(totalDataset * training/100),c(inputs, target)]
head(trainDataset)    # Show Top 6 records
nrow(trainDataset)    # Show number of train Dataset

#Select Testing Data Set
testDataset <- dataset[(totalDataset * training/100):totalDataset,c(inputs, target)]
head(testDataset)
nrow(testDataset)

# Model Building (Training)
formula <- as.formula(paste(target, "~", paste(c(inputs), collapse = "+")))
formula
model  <- ksvm(formula, trainDataset, kernel="rbfdot", prob.model=TRUE)
model

# Prediction (Testing)

Predicted <- round(as.numeric(predict(model, testDataset)))
head(Predicted)
PredictedProb <- predict(model, testDataset)
head(PredictedProb)

#Extracting Actual
Actual <- as.double(unlist(testDataset[target]))
head(Actual)

# Evaluate on: Accuracy.

# Model Evaluation
# Confusion Matrix
ConfusionMatrix <- misclassCounts(Predicted,Actual)$conf.matrix
ConfusionMatrix

accuracy <- round(mean(Actual==Predicted) *100,2)
accuracy
```

Exercise

- Try linear and non-linear SVM model and predict whether a given car gets high or low gas mileage based on the Auto dataset. And analyze the difference between them.
- Learn about kernlab from:
  # https://cran.r-project.org/web/packages/kernlab/kernlab.pdf