Labsheet - 5

Support Vector Machine

Machine Learning

BITS F464

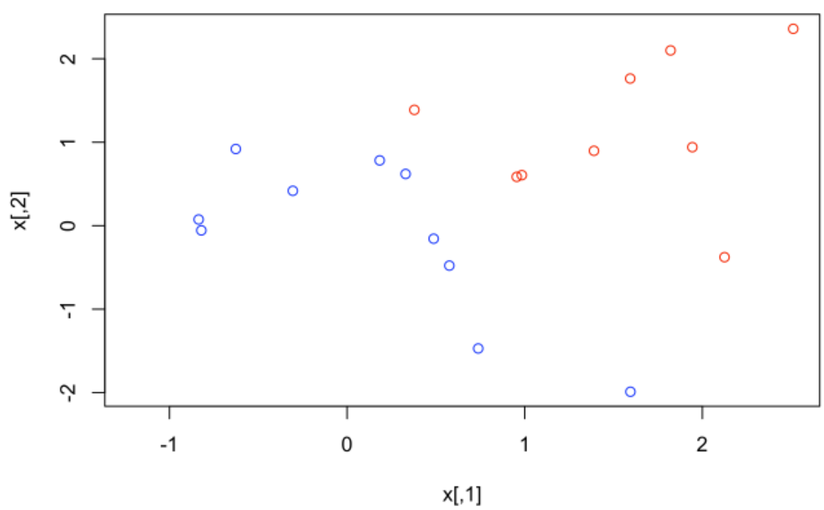
I Semester 2021-22

The e1071 library in R has implemented several statistical learning methods. The svm() method can fit a support vector classifier when the argument kernel='linear' is used. 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.

```
set.seed(1)
#Create our own test data
x <- matrix(rnorm(20*2), ncol=2)
y <- c(rep(-1,10), rep(1,10))
x[y==1,]=x[y==1,] + 1
```

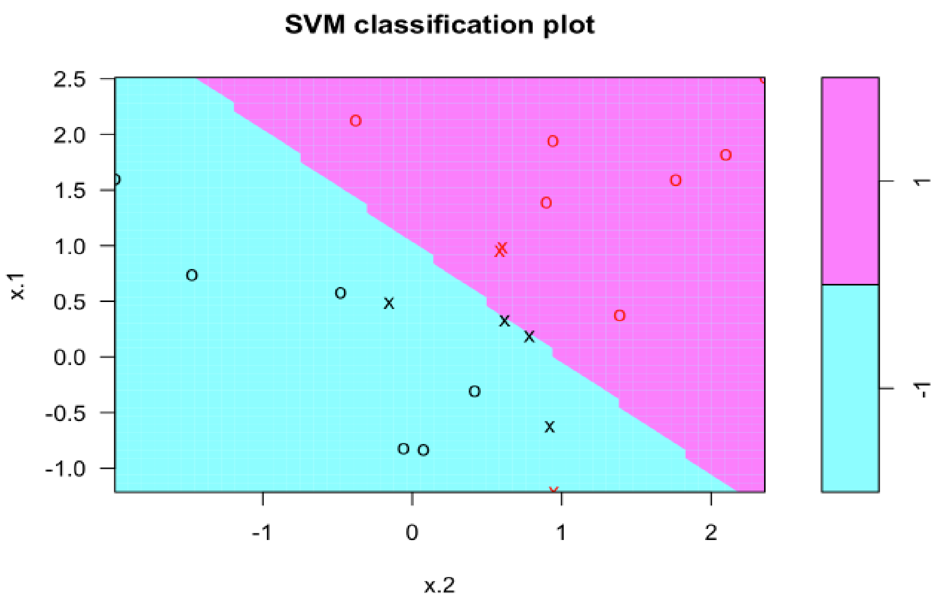
Let's take a look at these data

```
plot(x, col=(3-y))
```



To perform classification, we have to specify the response as a factor. The argument scale = FALSE tells the function not to scale each feature. Sometimes we may want to do this.

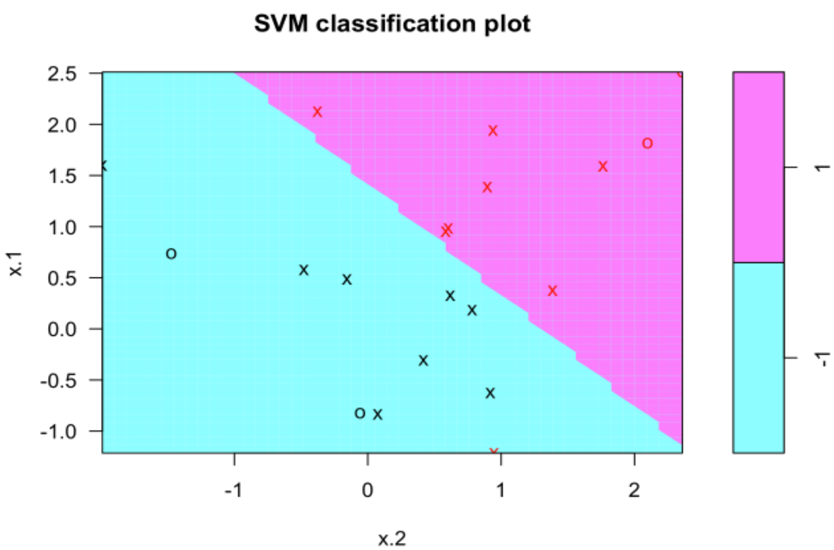
```
library(e1071)

dat <- data.frame(x=x, y=as.factor(y))
svm.fit <- svm(y ~., data=dat, kernel='linear', cost=10, scale=FALSE)
# Plot the SVC obtained
plot(svm.fit, dat)
```




```
summary(svm.fit)
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?

```
svm.fit2 <- svm(y ~., data=dat, kernel = 'linear', cost=0.1, scale=FAL
SE)
plot(svm.fit2, dat)
```



Now analyze both plots?

```
summary(svm.fit2)
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
```

What is the cost effect on margin?

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

```
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)
```

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.

```
xtest=matrix(rnorm(20*2), ncol=2)
ytest=sample(c(-1,1), 20, rep=TRUE)
xtest [ ytest ==1 ,]= xtest [ ytest ==1 ,] + 1
testdat=data.frame(x=xtest, y=as.factor(ytest))
```

Then predict the class labels of the test observations from the cross validated results.

```
yhat <- predict(tune.out$best.model, testdat)
library(caret)

confusionMatrix(yhat, testdat$y)
```

Non Linear SVM

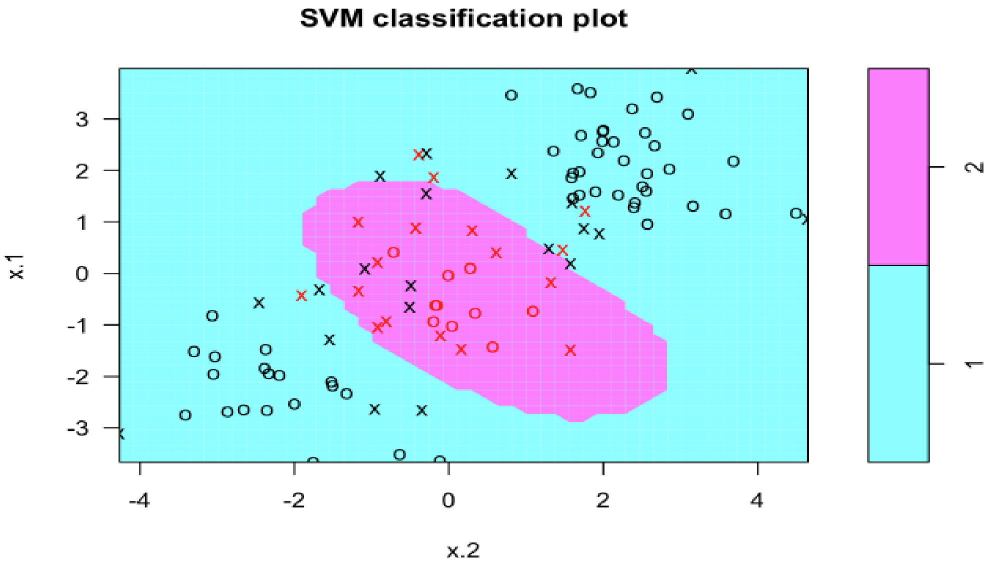
Again we use the svm() function, however now we can experiment with non-linear kernels. For polynomial kernels we use the parameter degree to adjust the polynomial order. For radial kernels we use the gamma parameter to adjust the y value.

```
# 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)
```

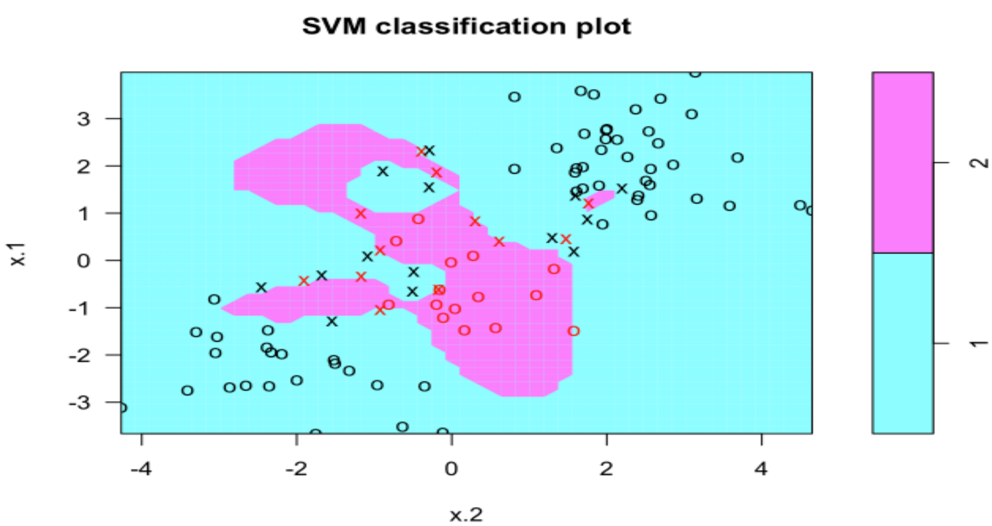
```
plot(svm.fit, dat[train,])
```




```
summary(svm.fit)
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'])
```

ROC Curves

Another way to choose between models is to use and ROC curve. We can do this using the ROCR package.

```
library(ROCR)
# function to handle the different models
rocplot <- function(pred, truth, ...){
  predob =  prediction(pred, truth)
  perf = performance(predob, 'tpr', 'fpr')
  plot(perf, ...)

}
```

Now, when we rebuild the SVMs we set decision.values=TRUE to obtain the fitted values.

```
svm.opt <- svm(y ~., data=dat[train,], kernel='radial',
               gamma=2, cost=1, decision.values=T)

fitted <- attributes(predict(svm.opt, dat[train,], decision.values=T))
$decision.values

rocplot(fitted, dat[train,'y'], main='Training Data')
```

Exercise

- Try linear and polynomial 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](https://cran.r-project.org/web/packages/kernlab/kernlab.pdf)