9.6.2 Support Vector Machine

Team 16

3/7/2021

Lab 9.6.2 Support Vector Machine

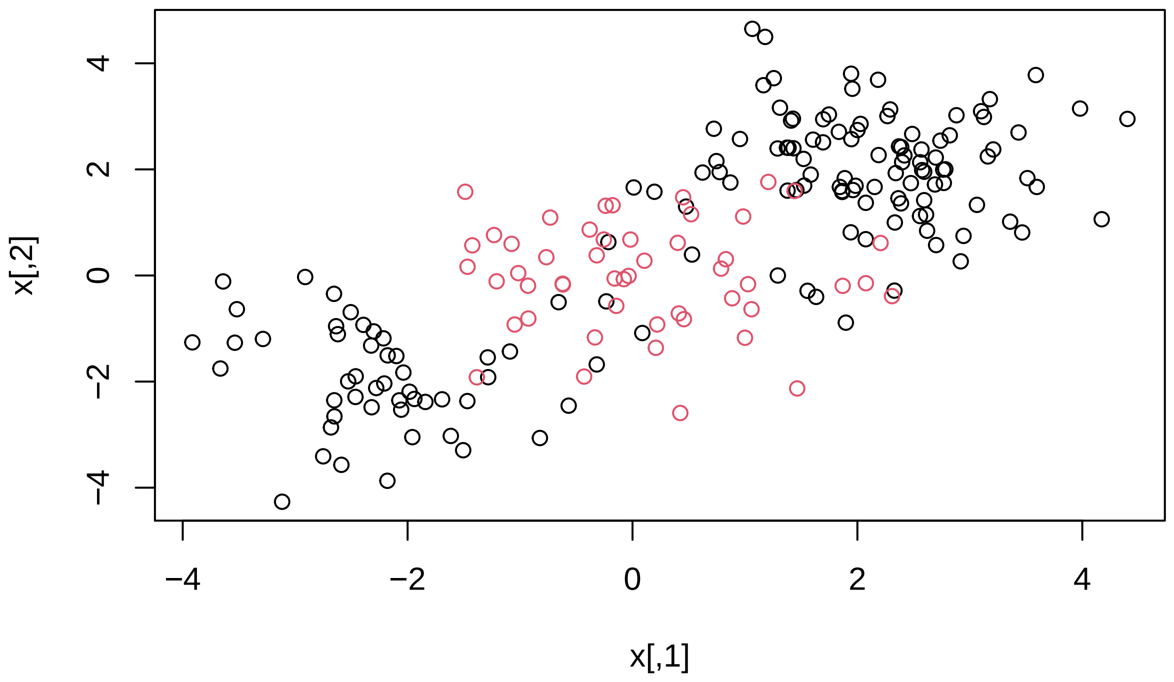
This lab will fit a support vector machine using a nonlinear kernel. This is done simply by changing the parameter 'kernel' within the svm() function. If we want a polynomial kernel, we use kernel='polynomial' and specify the degree for the degree of polynomial. If we want radial, we use kernel='radial' and specify gamma.

To show this in R, we will create a random dataset with a non-linear boundary.

```r
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))
```

Next, we plot the data to ensure we have nonlinear class boundaries.

```r
plot(x, col=y)
```

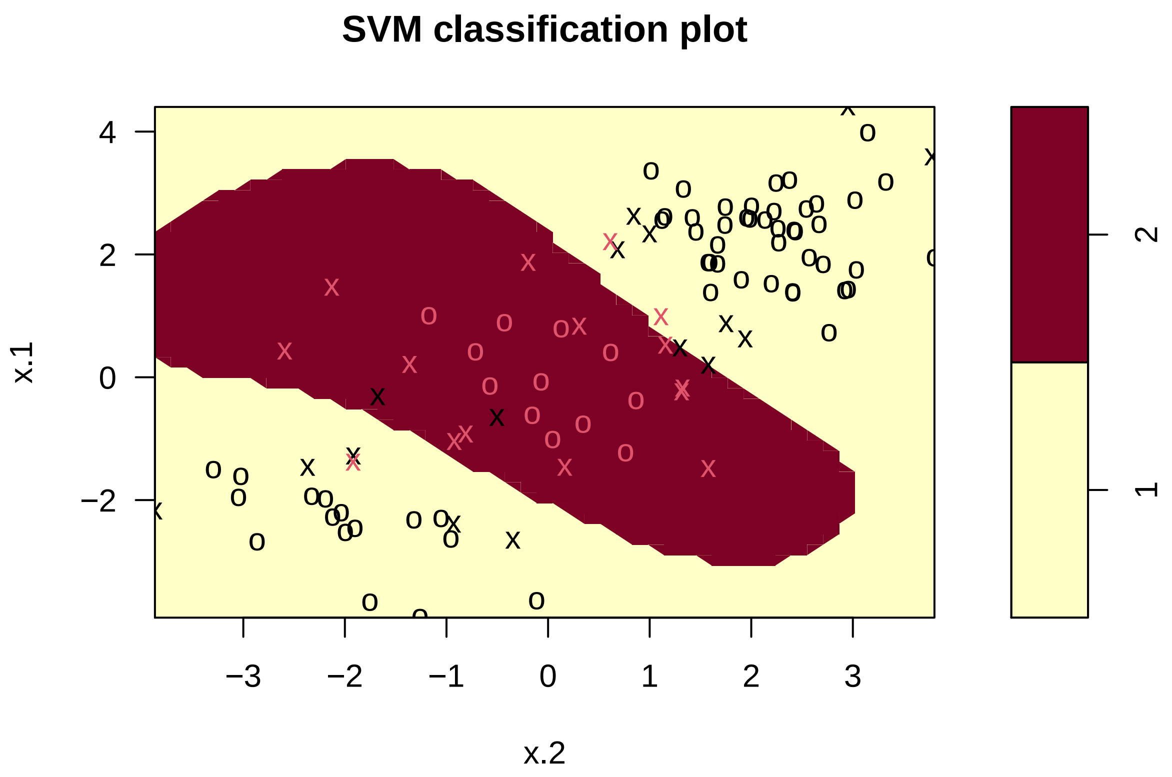


Now we split the data into training and testing groups using R's sample function to get row indices. Then, we can create our svm() model using the training data. We will set the kernel to radial with a gamma of one. We choose radial because the graph above shows that the data cannot be split using a polynomial. If the split appears circular, a radial kernel is the optimal choice. Cost allows us to specify the cost of a violation to the margin.

```
train<- sample(200,100)
library(e1071) # Library that contains svm function
```

```
## Warning: package 'e1071' was built under R version 4.0.3
```

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



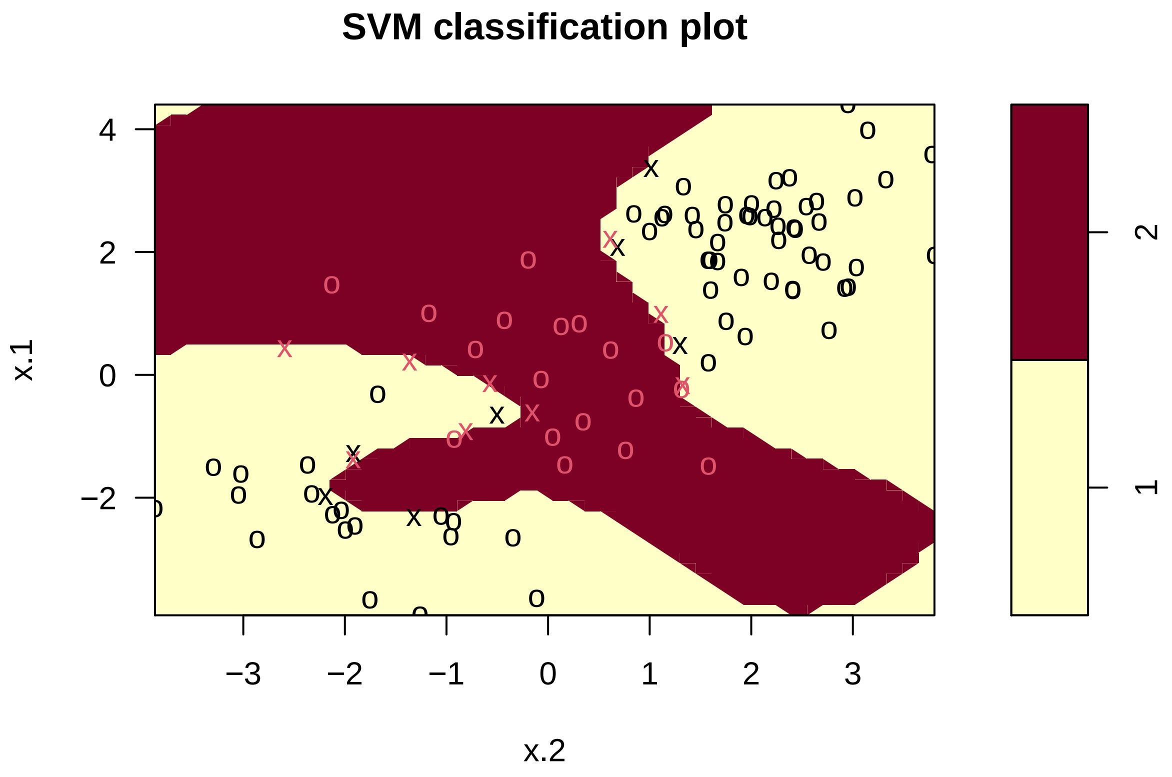
This plot shows that our model has a nonlinear and radial decision boundary. We can use the summary() function to get more information about our model.

```
summary(svmfit)
```

```
##
## Call:
## svm(formula = y ~ ., data = dat[train, ], kernel = "radial", gamma = 1,
##     cost = 1)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  31
##
##  ( 16 15 )
##
##
## Number of Classes:  2
##
## Levels:
##  1 2
```

By observing the plot of our model, we can also see that we misclassified some of the training samples. We could reduce the number of errors by increasing cost, but this could result in overfitting

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



To find the best value for gamma and cost, we use cross validation. In R, we do this using the tune() function.

```
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)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##     1   0.5
##
## - best performance: 0.07
##
## - Detailed performance results:
##     cost gamma error dispersion
## 1  1e-01   0.5  0.26 0.15776213
## 2  1e+00   0.5  0.07 0.08232726
## 3  1e+01   0.5  0.07 0.08232726
## 4  1e+02   0.5  0.14 0.15055453
## 5  1e+03   0.5  0.11 0.07378648
## 6  1e-01   1.0  0.22 0.16193277
## 7  1e+00   1.0  0.07 0.08232726
## 8  1e+01   1.0  0.09 0.07378648
## 9  1e+02   1.0  0.12 0.12292726
## 10 1e+03   1.0  0.11 0.11005049
## 11 1e-01   2.0  0.27 0.15670212
## 12 1e+00   2.0  0.07 0.08232726
## 13 1e+01   2.0  0.11 0.07378648
## 14 1e+02   2.0  0.12 0.13165612
## 15 1e+03   2.0  0.16 0.13498971
## 16 1e-01   3.0  0.27 0.15670212
## 17 1e+00   3.0  0.07 0.08232726
## 18 1e+01   3.0  0.08 0.07888106
## 19 1e+02   3.0  0.13 0.14181365
## 20 1e+03   3.0  0.15 0.13540064
## 21 1e-01   4.0  0.27 0.15670212
## 22 1e+00   4.0  0.07 0.08232726
## 23 1e+01   4.0  0.09 0.07378648
## 24 1e+02   4.0  0.13 0.14181365
## 25 1e+03   4.0  0.15 0.13540064
```

According to the cross validation results, our best cost is 1 and best gamma is 0.5. Gammas of 2 and 4 will also produce the same results. Finally, we can use the model to predict on our test data using predict(). Since our train variable are indices, we get our test data by using -train as an index set.

We can use the caret package to get more info on our predictions.

```r
conf.mat<- table(true=dat[-train,'y'], pred=predict(tune.out$best.model, newdata=dat[-train,]))
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.0.3

## Loading required package: lattice

## Loading required package: ggplot2
```

```r
confusionMatrix(conf.mat)
```

```
## Confusion Matrix and Statistics
##
##     pred
## true  1  2
##    1 67 10
##    2  2 21
##
##                Accuracy : 0.88
##                  95% CI : (0.7998, 0.9364)
##     No Information Rate : 0.69
##     P-Value [Acc > NIR] : 7.669e-06
##
##                   Kappa : 0.698
##
##  Mcnemar's Test P-Value : 0.04331
##
```

```
##             Sensitivity : 0.9710
##             Specificity : 0.6774
##          Pos Pred Value : 0.8701
##          Neg Pred Value : 0.9130
##              Prevalence : 0.6900
##          Detection Rate : 0.6700
##    Detection Prevalence : 0.7700
##       Balanced Accuracy : 0.8242
##
##        'Positive' Class : 1
##
```

Our accuracy on the test set is 88%