Team 17 Applied Exercise

Team 17

3/22/2021

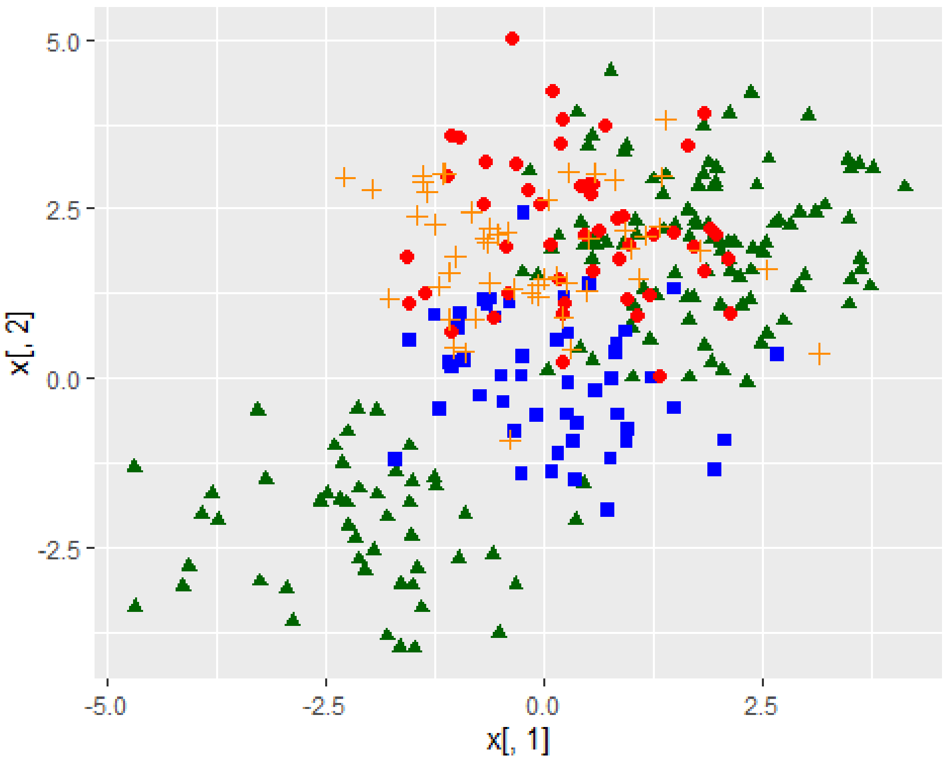
Team 17 Exercise

We combined data from 9.6.2 and 9.6.4, and added an extra 4th class to create our dataframe

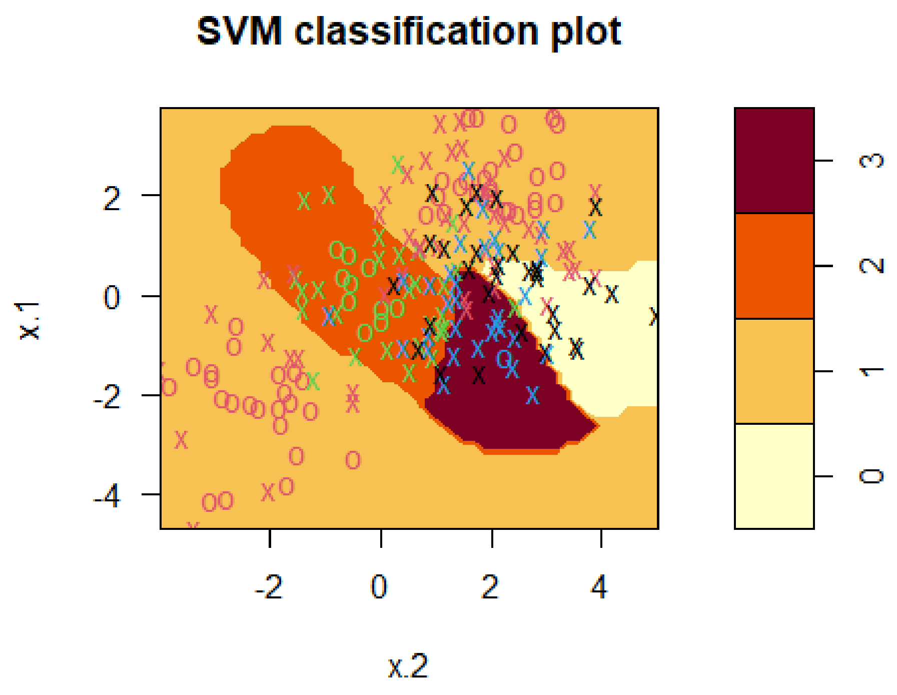
```r
set.seed(2021)
x = matrix(rnorm (200 * 2), ncol = 2)
x[1:100,] = x[1:100,] + 2
x[101:150,] = x[101:150,] - 2
y = c(rep(1,150), rep(2,50))

x = rbind(x, matrix(rnorm (50 * 2), ncol = 2))
y = c(y, rep(0, 50))
x[y == 0, 2] = x[y == 0, 2] + 2
dat = data.frame(x = x, y = as.factor(y))

x = rbind(x, matrix(rnorm (50 * 2), ncol = 2))
y = c(y, rep(3, 50))
x[y == 3, 2] = x[y == 3, 2] + 2
dat = data.frame(x = x, y = as.factor(y))
```

We plotted our data to determine if they are linearly separable or if the plot could show where the boundaries could be.

```r
library(ggplot2)
ggplot(data = dat, aes(x = x[,1], y = x[,2], color = y, shape = y)) +
  geom_point(size = 2) +
  scale_color_manual(values=c("red","dark green","blue","dark orange")) +
  theme(legend.position = "none")
```



```
train <- sample(300,200) # 300 for the total samples, 200 for training the
dataset.
library(e1071) # Library that contains svm function
svmfit <- svm(y~., data=dat[train,], kernel='radial', gamma = 1, cost = 1)
plot(svmfit, dat[train,])
```



```
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:  138
##
##  ( 29 46 32 31 )
##
##
## Number of Classes:  4
##
## Levels:
##  0 1 2 3

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
##    10     3
##
## - best performance: 0.32
##
## - Detailed performance results:
##     cost gamma error dispersion
## 1  1e-01   0.5 0.465 0.09442810
## 2  1e+00   0.5 0.345 0.06851602
## 3  1e+01   0.5 0.360 0.08096639
## 4  1e+02   0.5 0.345 0.09264628
## 5  1e+03   0.5 0.345 0.10124228
## 6  1e-01   1.0 0.410 0.06992059
## 7  1e+00   1.0 0.370 0.08232726
## 8  1e+01   1.0 0.350 0.08498366
## 9  1e+02   1.0 0.340 0.10219806
## 10 1e+03   1.0 0.390 0.10219806
## 11 1e-01   2.0 0.440 0.07378648
## 12 1e+00   2.0 0.355 0.07619420
## 13 1e+01   2.0 0.330 0.11105554
## 14 1e+02   2.0 0.400 0.13123346
## 15 1e+03   2.0 0.415 0.13133926
## 16 1e-01   3.0 0.505 0.10124228
## 17 1e+00   3.0 0.355 0.08644202
## 18 1e+01   3.0 0.320 0.12516656
## 19 1e+02   3.0 0.405 0.12122064
## 20 1e+03   3.0 0.410 0.14102797
## 21 1e-01   4.0 0.515 0.10013879
## 22 1e+00   4.0 0.330 0.10327956
## 23 1e+01   4.0 0.365 0.11796892
## 24 1e+02   4.0 0.405 0.12122064
## 25 1e+03   4.0 0.450 0.10540926
```

From the output, it was determined the best model when cost = 1e+02, and gamma = 1.0
Based upon the model, let's test it with a testing set.

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

## Loading required package: lattice

confusionMatrix(conf.mat)
```

```
## Confusion Matrix and Statistics
##
##     pred
## true  0  1  2  3
##    0  3  7  3  6
##    1  6 43  2  2
##    2  0  2  9  0
##    3  3  1  3 10
##
## Overall Statistics
##
##                Accuracy : 0.65
##                  95% CI : (0.5482, 0.7427)
##     No Information Rate : 0.53
##     P-Value [Acc > NIR] : 0.01013
##
##                   Kappa : 0.459
##
##  Mcnemar's Test P-Value : 0.28457
##
## Statistics by Class:
##
##                      Class: 0 Class: 1 Class: 2 Class: 3
## Sensitivity            0.2500   0.8113   0.5294   0.5556
## Specificity            0.8182   0.7872   0.9759   0.9146
## Pos Pred Value         0.1579   0.8113   0.8182   0.5882
## Neg Pred Value         0.8889   0.7872   0.9101   0.9036
## Prevalence             0.1200   0.5300   0.1700   0.1800
## Detection Rate         0.0300   0.4300   0.0900   0.1000
## Detection Prevalence   0.1900   0.5300   0.1100   0.1700
## Balanced Accuracy      0.5341   0.7993   0.7527   0.7351
```

From our training and tunning method, our best model has a 65% accuracy rate.