Assignment 3

QINGTAN

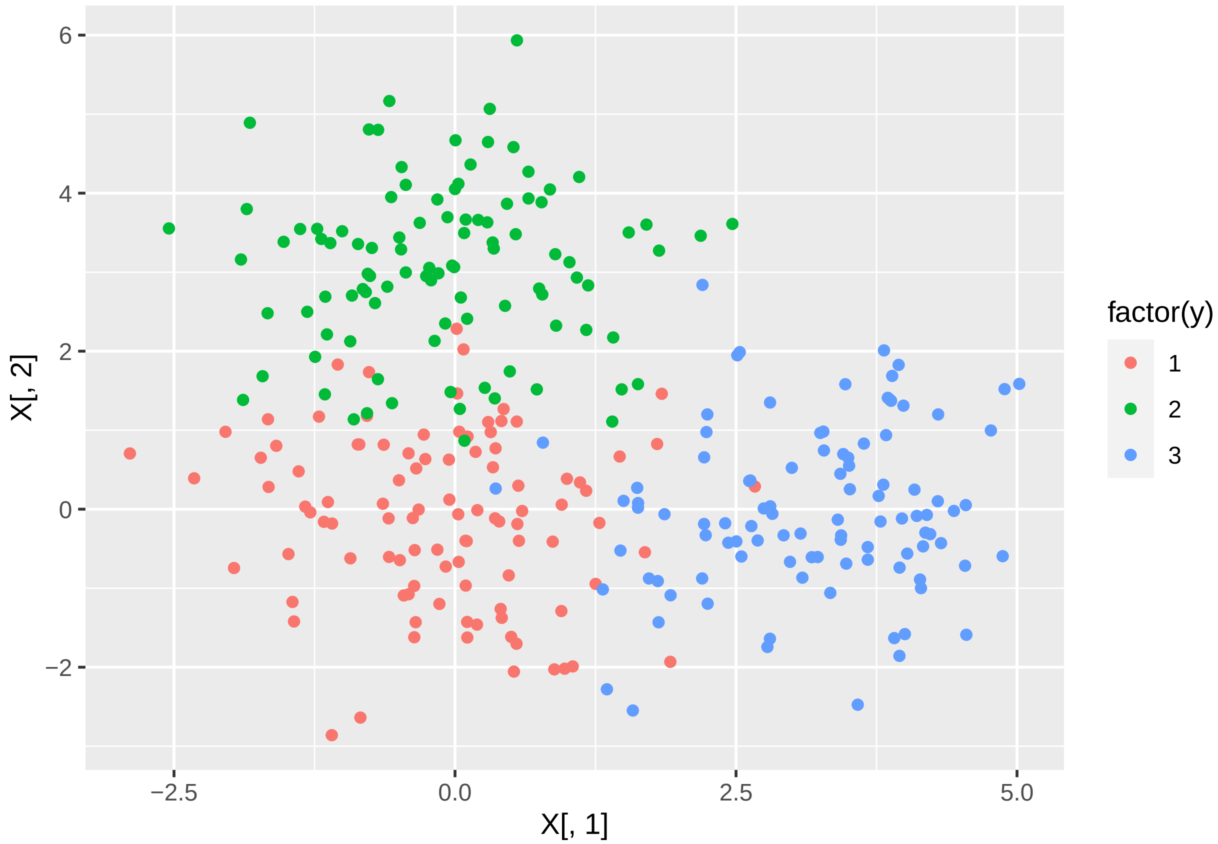
15/10/2022

Q1

1.

Here we draw the plot for these three group points.

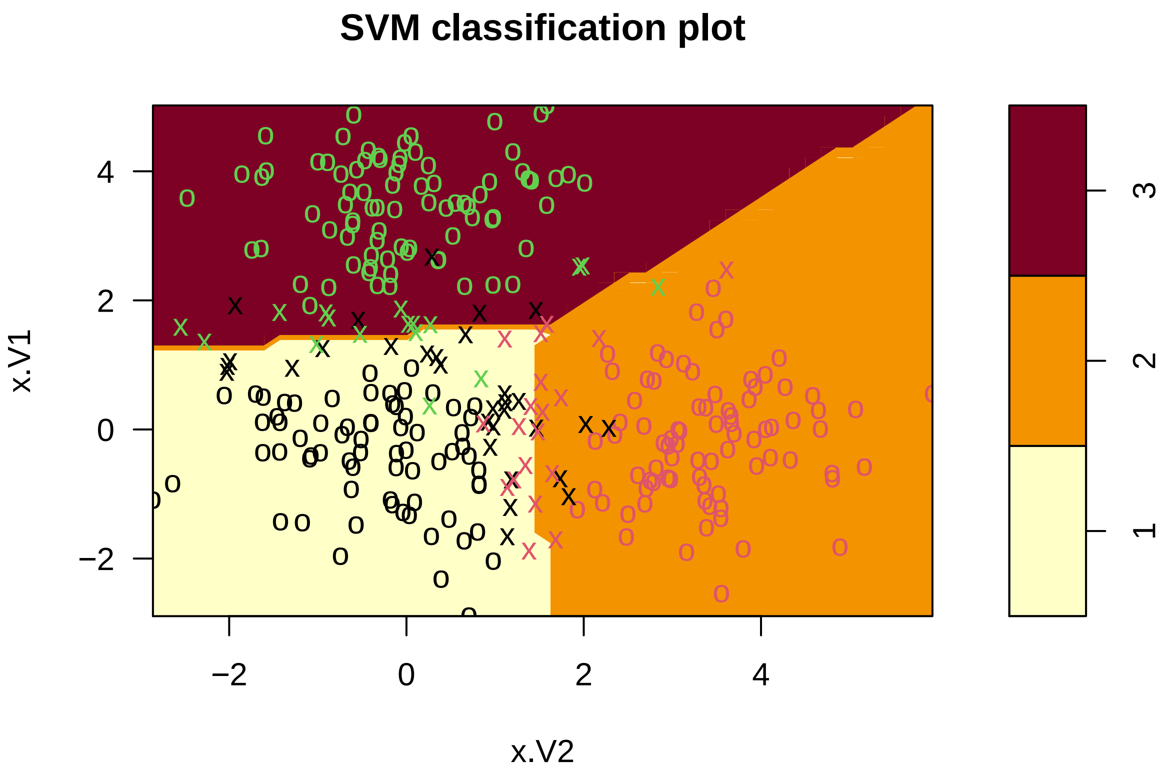
```
library(ggplot2)
C = 3
N = 300
set.seed(50)
X = matrix(rnorm(N*2), ncol=2)
Z = matrix(c(0,0,3,0,3,0), C, 2)
y = c(rep(1,100), rep(2,100), rep(3,100))
for (i in 1:N){
  X[i,] <- X[i,] + Z[y[i], ]
}
X = as.data.frame(X)
ggplot(data=X, aes(x=X[,1], y=X[,2], color=factor(y))) + geom_point()
```



2.

We draw the svm image firstly:

```
library(e1071)
tdata = data.frame(x=X, y=as.factor(y))
svmfit = svm(y~., data=tdata, cost=10, kernel='linear')
plot(svmfit, tdata)
```



Then we use the summary method to get the support vector information:

```
summary(svmfit)
```

```
##
## Call:
## svm(formula = y ~ ., data = tdata, cost = 10, kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  10
##
## Number of Support Vectors:  67
##
##  ( 31 19 17 )
##
##
## Number of Classes:  3
##
## Levels:
##  1 2 3
```

So we can see that there are 31 vectors in class 1, and 19 vectors in class 2, and 17 vectors in class 3. They have 67 vectors in total.

3.

```
set.seed(50)
cross_svm = tune(method=svm, y~., data=tdata, kernel='linear', ranges=list(cost=c(0.001,0.01,0.1,1,5,10
summary(cross_svm)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     1
##
## - best performance: 0.08
##
## - Detailed performance results:
##    cost      error dispersion
## 1 1e-03 0.77000000 0.04288946
## 2 1e-02 0.10000000 0.04969040
## 3 1e-01 0.10666667 0.07503086
## 4 1e+00 0.08000000 0.05921294
## 5 5e+00 0.09000000 0.05889937
## 6 1e+01 0.09000000 0.05889937
## 7 1e+02 0.08333333 0.05270463
```

After we get the cost, error and dispersion for each cost value, then we can find our best model:

```
bestmod = cross_svm$best.model
summary(bestmod)
```

```
##
## Call:
## best.tune(method = svm, train.x = y ~ ., data = tdata, ranges = list(cost = c(0.001,
##     0.01, 0.1, 1, 5, 10, 100)), kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  81
##
##  ( 37 22 22 )
##
##
## Number of Classes:  3
##
## Levels:
##  1 2 3
```

The cost value is 1, we get the best model with the lowest error, which is 0.08. The total number for support vectors is 81 in this best model, which is greater than 67 support vectors when cost=10. Also there are 37 support vectors in class 1, also 22 support vectors in class 2, and 22 support vectors in class 3. The number of support vectors in each class of our best model is also greater than the model we created when cost=10.

4.

```r
set.seed(100)
xtest = matrix(rnorm(N*2), ncol=2)
set.seed(100)
ytest = sample(c(1,2,3), 300, replace=TRUE)
for (i in 1:N){
  xtest[i,] <- xtest[i,] + Z[ytest[i],]
}
testdata = data.frame(x=xtest, y=as.factor(ytest))
names(testdata) <- c('x.V1', 'x.V2')
predict_class = as.matrix(predict(bestmod, testdata))
wrong_class = sum(ytest != predict_class)
wrong_class
```

```
## [1] 27
```

So we have 27 observations in total which have a wrong prediction. Then we obtain the result table:

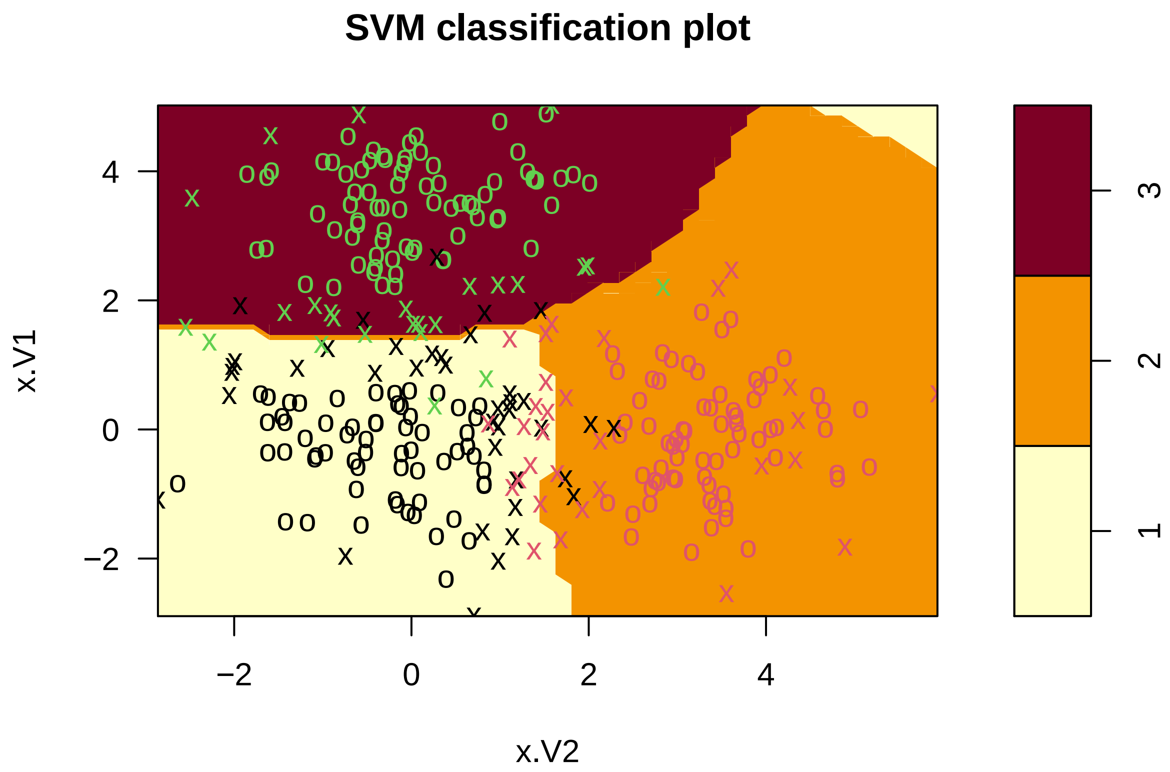
```r
table(predict_class, ytest)
```

```
##              ytest
## predict_class   1   2   3
##             1  74   9   4
##             2   8  90   1
##             3   4   1 109
```

In class 3, we can find there are $4+1+109 = 114$ observations, which is greater than 100, it is because ytest is labeled randomly with replacement. So it has the probability that one class has more than 100 samples. We can also see that 4 samples in class 3 are misclassified to class 1, also one sample is misclassified to class 2.

5.

```r
set.seed(50)
rad_svm = svm(y~., data=tdata, cost=1, kernel='radial', gamma=1)
plot(rad_svm, tdata)
```




```r
radsvm_tune = tune(svm, y~., data=tdata, ranges=list(cost=c(0.1,1,10,100,1000), gamma=c(0.5,1,2,3,4)), l
summary(radsvm_tune)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##   100     2
##
## - best performance: 0.07333333
##
## - Detailed performance results:
##      cost gamma      error dispersion
## 1  1e-01   0.5 0.10000000 0.06478835
## 2  1e+00   0.5 0.08666667 0.06324555
## 3  1e+01   0.5 0.07666667 0.05454639
## 4  1e+02   0.5 0.08000000 0.04766136
## 5  1e+03   0.5 0.09000000 0.04981447
## 6  1e-01   1.0 0.10666667 0.07503086
## 7  1e+00   1.0 0.09000000 0.06295207
## 8  1e+01   1.0 0.08000000 0.04216370
## 9  1e+02   1.0 0.08000000 0.03583226
```

```
## 10 1e+03   1.0 0.07666667 0.05223404
## 11 1e-01   2.0 0.09333333 0.06440612
## 12 1e+00   2.0 0.08333333 0.05719795
## 13 1e+01   2.0 0.07666667 0.04727122
## 14 1e+02   2.0 0.07333333 0.04097575
## 15 1e+03   2.0 0.10333333 0.05973191
## 16 1e-01   3.0 0.09000000 0.06295207
## 17 1e+00   3.0 0.07666667 0.05454639
## 18 1e+01   3.0 0.07666667 0.04458312
## 19 1e+02   3.0 0.08666667 0.04766136
## 20 1e+03   3.0 0.12000000 0.06126244
## 21 1e-01   4.0 0.09333333 0.06813204
## 22 1e+00   4.0 0.08000000 0.04766136
## 23 1e+01   4.0 0.07333333 0.04388537
## 24 1e+02   4.0 0.10333333 0.05544433
## 25 1e+03   4.0 0.10666667 0.06992059
```

```
bestmod_rad = radsvm_tune$best.model
summary(bestmod_rad)
```

```
##
## Call:
## best.tune(method = svm, train.x = y ~ ., data = tdata, ranges = list(cost = c(0.1,
##     1, 10, 100, 1000), gamma = c(0.5, 1, 2, 3, 4)), kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  100
##
## Number of Support Vectors:  88
##
##  ( 32 30 26 )
##
##
## Number of Classes:  3
##
## Levels:
##  1 2 3
```

```
names(testdata) <- c('x.V1', 'x.V2')
predict_class_rad = as.matrix(predict(bestmod_rad, testdata))
wrong_class_rad = sum(ytest != predict_class_rad)
wrong_class_rad
```

```
## [1] 36
```

```
table(predict_class_rad, ytest)
```

```
##                    ytest
## predict_class_rad   1   2   3
```

```
##                           1  68   5   9
##                           2  14  92   1
##                           3   4   3 104
```

Conclusion: When using radial kernel, we get the best model when cost=100, gamma = 2. There are 36 misclassfied samples in this best radial model, which is greater than 27 misclassfied samples when using linear kernel. Because the percentage for correct prediction is lower, so these samples might be linearly separated and it is useless to use radial kernel. Then we choose linear kernel.