[STAT 4610] HW - 10

Michael Ghattas

11/14/2022

Chapter 9

Problem - 8

```
library(ISLR)
library(ISLR2)

## Warning: package 'ISLR2' was built under R version 4.1.2

##
## Attaching package: 'ISLR2'

## The following objects are masked from 'package:ISLR':
##
##     Auto, Credit

library(e1071)

## Warning: package 'e1071' was built under R version 4.1.2
```

Part (a)

```
set.seed(123)

train = sample(nrow(OJ), 800)
OJ.train = OJ[train, ]
OJ.test = OJ[-train, ]
```

Part (b)

```
svmLMod <- svm(Purchase ~ . , kernel = "linear", data = OJ.train, cost =
0.01)
summary(svmLMod)

##
## Call:
## svm(formula = Purchase ~ ., data = OJ.train, kernel = "linear", cost =
0.01)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.01
##
## Number of Support Vectors:  442
##
##  ( 220 222 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

-> SV linear classifier created 442 support vectors from 800 training points.
-> 220 belong to level CH.
-> 222 belong to level MM.

Part (c)

```
errorRate <- function(svm_model, dataset, true_classes)
  {
    confusionMatrix <- table(predict(svm_model, dataset), true_classes)
    return(1 - sum(diag(confusionMatrix)) / sum(confusionMatrix))
}

cat("Training Error: ", 100 * errorRate(svmLMod, OJ.train,
OJ.train$Purchase), "%\n")
```

```
## Training Error:  16.5 %
```

```
cat("Test Error: ", 100 * errorRate(svmLMod, OJ.test, OJ.test$Purchase), "%
\n")
```

```
## Test Error:  17.77778 %
```

-> Training error is 16.5%
-> Test error is 17.78%

Part (d)

```r
set.seed(123)

svmTune <- tune(svm, Purchase ~ . , data = OJ, kernel = "linear", ranges =
list(cost = seq(0.01, 10, length = 100)))
summary(svmTune)

##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##       cost
##  0.3127273
##
## - best performance: 0.164486
##
## - Detailed performance results:
##            cost     error dispersion
## 1    0.0100000 0.1710280 0.03713573
## 2    0.1109091 0.1691589 0.03339296
## 3    0.2118182 0.1663551 0.02948825
## 4    0.3127273 0.1644860 0.02929012
## 5    0.4136364 0.1654206 0.02753090
## 6    0.5145455 0.1654206 0.02753090
## 7    0.6154545 0.1654206 0.02753090
## 8    0.7163636 0.1672897 0.02870442
## 9    0.8172727 0.1672897 0.02870442
## 10   0.9181818 0.1672897 0.02870442
## 11   1.0190909 0.1672897 0.03034786
## 12   1.1200000 0.1672897 0.03034786
## 13   1.2209091 0.1682243 0.03176959
## 14   1.3218182 0.1672897 0.03034786
## 15   1.4227273 0.1672897 0.03190676
## 16   1.5236364 0.1682243 0.03296886
## 17   1.6245455 0.1682243 0.03296886
## 18   1.7254545 0.1682243 0.03296886
## 19   1.8263636 0.1682243 0.03296886
## 20   1.9272727 0.1691589 0.03310106
## 21   2.0281818 0.1682243 0.03412602
## 22   2.1290909 0.1682243 0.03412602
## 23   2.2300000 0.1672897 0.03425375
## 24   2.3309091 0.1672897 0.03425375
## 25   2.4318182 0.1672897 0.03425375
## 26   2.5327273 0.1672897 0.03425375
## 27   2.6336364 0.1672897 0.03425375
## 28   2.7345455 0.1672897 0.03425375
## 29   2.8354545 0.1672897 0.03425375
## 30   2.9363636 0.1672897 0.03425375
## 31   3.0372727 0.1663551 0.03546480
## 32   3.1381818 0.1663551 0.03546480
## 33   3.2390909 0.1663551 0.03546480
## 34   3.3400000 0.1663551 0.03546480
## 35   3.4409091 0.1663551 0.03519008
## 36   3.5418182 0.1654206 0.03580522
## 37   3.6427273 0.1654206 0.03580522
## 38   3.7436364 0.1654206 0.03580522
## 39   3.8445455 0.1654206 0.03580522
## 40   3.9454545 0.1654206 0.03580522
## 41   4.0463636 0.1654206 0.03580522
## 42   4.1472727 0.1663551 0.03519008
## 43   4.2481818 0.1663551 0.03519008
## 44   4.3490909 0.1654206 0.03525896
## 45   4.4500000 0.1654206 0.03525896
## 46   4.5509091 0.1654206 0.03525896
## 47   4.6518182 0.1654206 0.03525896
## 48   4.7527273 0.1644860 0.03584586
## 49   4.8536364 0.1644860 0.03584586
## 50   4.9545455 0.1644860 0.03584586
## 51   5.0554545 0.1644860 0.03584586
## 52   5.1563636 0.1644860 0.03584586
## 53   5.2572727 0.1654206 0.03580522
## 54   5.3581818 0.1654206 0.03580522
## 55   5.4590909 0.1654206 0.03580522
## 56   5.5600000 0.1654206 0.03580522
## 57   5.6609091 0.1654206 0.03580522
## 58   5.7618182 0.1654206 0.03580522
## 59   5.8627273 0.1654206 0.03580522
## 60   5.9636364 0.1654206 0.03580522
## 61   6.0645455 0.1654206 0.03580522
## 62   6.1654545 0.1644860 0.03446559
## 63   6.2663636 0.1644860 0.03446559
## 64   6.3672727 0.1644860 0.03446559
## 65   6.4681818 0.1644860 0.03446559
## 66   6.5690909 0.1644860 0.03446559
## 67   6.6700000 0.1644860 0.03446559
## 68   6.7709091 0.1644860 0.03446559
## 69   6.8718182 0.1644860 0.03446559
## 70   6.9727273 0.1644860 0.03446559
## 71   7.0736364 0.1644860 0.03446559
## 72   7.1745455 0.1644860 0.03446559
## 73   7.2754545 0.1644860 0.03446559
## 74   7.3763636 0.1644860 0.03446559
## 75   7.4772727 0.1644860 0.03446559
## 76   7.5781818 0.1644860 0.03446559
## 77   7.6790909 0.1644860 0.03446559
## 78   7.7800000 0.1644860 0.03446559
## 79   7.8809091 0.1644860 0.03446559
## 80   7.9818182 0.1644860 0.03446559
## 81   8.0827273 0.1644860 0.03446559
## 82   8.1836364 0.1644860 0.03446559
## 83   8.2845455 0.1644860 0.03446559
## 84   8.3854545 0.1644860 0.03446559
## 85   8.4863636 0.1644860 0.03446559
## 86   8.5872727 0.1654206 0.03385477
## 87   8.6881818 0.1654206 0.03385477
## 88   8.7890909 0.1654206 0.03385477
## 89   8.8900000 0.1654206 0.03385477
## 90   8.9909091 0.1654206 0.03385477
## 91   9.0918182 0.1654206 0.03385477
## 92   9.1927273 0.1654206 0.03385477
## 93   9.2936364 0.1654206 0.03385477
## 94   9.3945455 0.1654206 0.03385477
## 95   9.4954545 0.1663551 0.03406909
## 96   9.5963636 0.1663551 0.03406909
## 97   9.6972727 0.1663551 0.03406909
## 98   9.7981818 0.1663551 0.03406909
## 99   9.8990909 0.1663551 0.03406909
## 100 10.0000000 0.1654206 0.03298358
```

-> Tuning indicates optimal cost = 0.3127273

Part (e)

```
svmTLM <- svm(Purchase ~ . , kernel = "linear", data = OJ.train, cost =
svmTune$best.parameters$cost)

cat("Training Error: ", 100 * errorRate(svmTLM, OJ.train, OJ.train$Purchase),
"%\n")

## Training Error:  16.25 %

cat("Test Error: ", 100 * errorRate(svmTLM, OJ.test, OJ.test$Purchase), "%
\n")

## Test Error:  15.92593 %
```

-> Training error is 16.25%
-> Test error is 15.93%

Part (f)

```
set.seed(123)
```

```
svmRadial <- svm(Purchase ~ . , data = OJ.train, kernel = "radial")
summary(svmRadial)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = OJ.train, kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  367
##
##  ( 181 186 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

```
cat("Training Error: ", 100 * errorRate(svmRadial, OJ.train,
OJ.train$Purchase), "%\n")
```

```
## Training Error:  13.875 %
```

```
cat("Test Error: ", 100 * errorRate(svmRadial, OJ.test, OJ.test$Purchase), "%
\n")
```

```
## Test Error:  18.88889 %
```

```
svmTune <- tune(svm, Purchase ~ . , data = OJ.train, kernel = "radial",
ranges = list(cost = seq(0.01, 10, length = 100)))
summary(svmTune)
```

```
##
## Parameter tuning of 'svm':
##
```

```
## - sampling method: 10-fold cross validation
##
## - best parameters:
##      cost
##  2.431818
##
## - best performance: 0.15875
##
## - Detailed performance results:
##            cost   error dispersion
## 1    0.0100000 0.39125 0.04411554
## 2    0.1109091 0.17625 0.05905800
## 3    0.2118182 0.17875 0.06347845
## 4    0.3127273 0.17125 0.05104804
## 5    0.4136364 0.17000 0.05210833
## 6    0.5145455 0.16750 0.05439056
## 7    0.6154545 0.16500 0.05062114
## 8    0.7163636 0.16125 0.04945888
## 9    0.8172727 0.16125 0.04945888
## 10   0.9181818 0.16250 0.05103104
## 11   1.0190909 0.16250 0.04823265
## 12   1.1200000 0.16250 0.04823265
## 13   1.2209091 0.16375 0.04387878
## 14   1.3218182 0.16375 0.04387878
## 15   1.4227273 0.16375 0.04387878
## 16   1.5236364 0.16375 0.04387878
## 17   1.6245455 0.16250 0.04249183
## 18   1.7254545 0.16250 0.04564355
## 19   1.8263636 0.16000 0.04706674
## 20   1.9272727 0.16250 0.04823265
## 21   2.0281818 0.16125 0.04875178
## 22   2.1290909 0.16125 0.04910660
## 23   2.2300000 0.16000 0.04669642
## 24   2.3309091 0.16000 0.04669642
## 25   2.4318182 0.15875 0.04788949
## 26   2.5327273 0.15875 0.04788949
## 27   2.6336364 0.15875 0.04489571
## 28   2.7345455 0.16000 0.04440971
## 29   2.8354545 0.16000 0.04440971
## 30   2.9363636 0.16125 0.04348132
## 31   3.0372727 0.16250 0.04249183
## 32   3.1381818 0.16250 0.04249183
## 33   3.2390909 0.16375 0.04226652
## 34   3.3400000 0.16250 0.03996526
## 35   3.4409091 0.16500 0.04158325
## 36   3.5418182 0.16625 0.04291869
## 37   3.6427273 0.16500 0.04116363
## 38   3.7436364 0.16500 0.04116363
## 39   3.8445455 0.16500 0.04116363
## 40   3.9454545 0.16500 0.04116363
## 41   4.0463636 0.16500 0.04031129
## 42   4.1472727 0.16375 0.03793727
## 43   4.2481818 0.16375 0.03793727
## 44   4.3490909 0.16375 0.03793727
## 45   4.4500000 0.16375 0.03793727
## 46   4.5509091 0.16375 0.03793727
## 47   4.6518182 0.16375 0.03793727
## 48   4.7527273 0.16500 0.03717451
## 49   4.8536364 0.16500 0.03717451
## 50   4.9545455 0.16500 0.03717451
## 51   5.0554545 0.16500 0.03717451
## 52   5.1563636 0.16500 0.03717451
## 53   5.2572727 0.16500 0.03717451
## 54   5.3581818 0.16500 0.03717451
## 55   5.4590909 0.16500 0.03717451
## 56   5.5600000 0.16500 0.03717451
## 57   5.6609091 0.16375 0.03884174
## 58   5.7618182 0.16500 0.03899786
## 59   5.8627273 0.16500 0.03899786
## 60   5.9636364 0.16625 0.04126894
## 61   6.0645455 0.16625 0.04126894
## 62   6.1654545 0.16750 0.03961621
## 63   6.2663636 0.16750 0.03961621
## 64   6.3672727 0.16750 0.03961621
## 65   6.4681818 0.16750 0.03961621
## 66   6.5690909 0.16750 0.03961621
## 67   6.6700000 0.16625 0.04084609
## 68   6.7709091 0.16750 0.04257347
## 69   6.8718182 0.16750 0.04257347
## 70   6.9727273 0.16875 0.04419417
## 71   7.0736364 0.16750 0.04571956
## 72   7.1745455 0.16750 0.04571956
## 73   7.2754545 0.16750 0.04571956
## 74   7.3763636 0.16750 0.04571956
## 75   7.4772727 0.16750 0.04571956
## 76   7.5781818 0.16750 0.04571956
## 77   7.6790909 0.16750 0.04571956
## 78   7.7800000 0.16875 0.04458528
## 79   7.8809091 0.16875 0.04458528
## 80   7.9818182 0.16875 0.04458528
## 81   8.0827273 0.16875 0.04458528
## 82   8.1836364 0.16750 0.04257347
## 83   8.2845455 0.16750 0.04257347
## 84   8.3854545 0.16750 0.04257347
## 85   8.4863636 0.16875 0.04458528
## 86   8.5872727 0.16875 0.04458528
## 87   8.6881818 0.16875 0.04458528
## 88   8.7890909 0.17000 0.04684490
## 89   8.8900000 0.17000 0.04684490
## 90   8.9909091 0.17000 0.04684490
## 91   9.0918182 0.17000 0.04684490
## 92   9.1927273 0.17000 0.04684490
## 93   9.2936364 0.17000 0.04684490
## 94   9.3945455 0.17125 0.04931827
## 95   9.4954545 0.17000 0.04794383
## 96   9.5963636 0.17000 0.04794383
## 97   9.6972727 0.17000 0.04794383
## 98   9.7981818 0.17000 0.04794383
## 99   9.8990909 0.17000 0.04794383
## 100 10.0000000 0.17000 0.04794383
```

```
svmRadial <- svm(Purchase ~ . , data = OJ.train, kernel = "radial", cost =
svmTune$best.parameters$cost)
cat("Training Error: ", 100 * errorRate(svmRadial, OJ.train,
OJ.train$Purchase), "%\n")
```

```
## Training Error:  13.625 %
```

```
cat("Test Error:", 100 * errorRate(svmRadial, OJ.test, OJ.test$Purchase), "%
\n")
```

```
## Test Error: 18.51852 %
```

-> SV radial classifier created 367 support vectors from 800 training points.
-> 181 belong to level CH.
-> 186 belong to level MM.
-> Training error is 13.63%
-> Test error is 18.52%
-> Tuning indicates optimal cost = 2.431818

Part (g)

```
set.seed(123)

svmPoly <- svm(Purchase ~ . , data = OJ.train, kernel = "poly", degree = 2)
summary(svmRadial)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = OJ.train, kernel = "radial", cost =
svmTune$best.parameters$cost)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  2.431818
```

```
##
## Number of Support Vectors:  342
##
##  ( 168 174 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

```
cat("Training Error: ", 100 * errorRate(svmPoly, OJ.train, OJ.train$Purchase), "%\n")
```

```
## Training Error:  17.25 %
```

```
cat("Test Error: ", 100 * errorRate(svmPoly, OJ.test, OJ.test$Purchase), "%\n")
```

```
## Test Error:  22.22222 %
```

```
svmTune <- tune(svm, Purchase ~ . , data = OJ.train, kernel = "poly", degree = 2, ranges = list(cost = seq(0.01, 10, length = 100)))
summary(svmTune)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##  3.34
##
## - best performance: 0.16375
##
## - Detailed performance results:
##            cost   error dispersion
## 1    0.0100000 0.39000 0.04281744
## 2    0.1109091 0.30125 0.05084358
## 3    0.2118182 0.21875 0.06802012
## 4    0.3127273 0.21250 0.07430231
## 5    0.4136364 0.20375 0.07169815
## 6    0.5145455 0.20125 0.06192794
## 7    0.6154545 0.19875 0.06136469
## 8    0.7163636 0.19625 0.06068189
## 9    0.8172727 0.19625 0.06509875
## 10   0.9181818 0.19375 0.06568284
## 11   1.0190909 0.19125 0.06456317
## 12   1.1200000 0.18875 0.06547105
## 13   1.2209091 0.19250 0.06297045
## 14   1.3218182 0.18625 0.06520534
## 15   1.4227273 0.18250 0.06566963
## 16   1.5236364 0.17750 0.06556379
## 17   1.6245455 0.17875 0.06667969
## 18   1.7254545 0.17750 0.06476453
## 19   1.8263636 0.17750 0.06661456
## 20   1.9272727 0.17625 0.06520534
## 21   2.0281818 0.17625 0.06573569
## 22   2.1290909 0.17875 0.06536489
## 23   2.2300000 0.17875 0.06265259
## 24   2.3309091 0.17875 0.06265259
## 25   2.4318182 0.17750 0.06089609
## 26   2.5327273 0.17875 0.06010696
## 27   2.6336364 0.17750 0.06118052
## 28   2.7345455 0.17500 0.05773503
## 29   2.8354545 0.17250 0.05614960
## 30   2.9363636 0.17000 0.05749396
## 31   3.0372727 0.16750 0.05596378
## 32   3.1381818 0.16625 0.05529278
## 33   3.2390909 0.16625 0.05529278
## 34   3.3400000 0.16375 0.05726704
## 35   3.4409091 0.16375 0.05726704
## 36   3.5418182 0.16500 0.05676462
## 37   3.6427273 0.16625 0.05591723
## 38   3.7436364 0.16625 0.05591723
## 39   3.8445455 0.16500 0.05706965
## 40   3.9454545 0.16500 0.05706965
## 41   4.0463636 0.16625 0.05864500
## 42   4.1472727 0.16625 0.05653477
## 43   4.2481818 0.16625 0.05653477
## 44   4.3490909 0.16375 0.05756940
## 45   4.4500000 0.16375 0.05756940
## 46   4.5509091 0.16500 0.05583955
## 47   4.6518182 0.16625 0.05775006
## 48   4.7527273 0.16750 0.05749396
## 49   4.8536364 0.16750 0.05749396
## 50   4.9545455 0.16750 0.05749396
## 51   5.0554545 0.16875 0.05958479
## 52   5.1563636 0.16750 0.05839283
## 53   5.2572727 0.16750 0.05839283
## 54   5.3581818 0.16750 0.05839283
## 55   5.4590909 0.16750 0.05839283
## 56   5.5600000 0.16750 0.05839283
## 57   5.6609091 0.16750 0.05839283
## 58   5.7618182 0.16750 0.05839283
## 59   5.8627273 0.16750 0.05839283
## 60   5.9636364 0.16875 0.05720638
## 61   6.0645455 0.16875 0.05720638
## 62   6.1654545 0.16750 0.05565269
## 63   6.2663636 0.16875 0.05810969
## 64   6.3672727 0.16875 0.05810969
## 65   6.4681818 0.16875 0.05810969
## 66   6.5690909 0.16875 0.05810969
## 67   6.6700000 0.16875 0.05810969
## 68   6.7709091 0.16875 0.05810969
## 69   6.8718182 0.16750 0.05898446
## 70   6.9727273 0.16750 0.05898446
## 71   7.0736364 0.16750 0.05898446
## 72   7.1745455 0.16750 0.05898446
## 73   7.2754545 0.16750 0.05898446
## 74   7.3763636 0.16875 0.05899918
## 75   7.4772727 0.16875 0.05899918
## 76   7.5781818 0.16750 0.05898446
## 77   7.6790909 0.16750 0.05898446
## 78   7.7800000 0.16750 0.05898446
## 79   7.8809091 0.16875 0.06159061
## 80   7.9818182 0.16750 0.05898446
## 81   8.0827273 0.16750 0.06297045
## 82   8.1836364 0.16875 0.06298424
## 83   8.2845455 0.16750 0.06043821
## 84   8.3854545 0.16750 0.06043821
## 85   8.4863636 0.16750 0.06043821
## 86   8.5872727 0.16750 0.06043821
## 87   8.6881818 0.16750 0.06043821
## 88   8.7890909 0.16875 0.05958479
## 89   8.8900000 0.16875 0.05958479
## 90   8.9909091 0.16750 0.05719120
## 91   9.0918182 0.16750 0.05719120
## 92   9.1927273 0.16750 0.05719120
## 93   9.2936364 0.16750 0.05719120
## 94   9.3945455 0.16750 0.05719120
## 95   9.4954545 0.16875 0.05899918
## 96   9.5963636 0.16875 0.05899918
## 97   9.6972727 0.16875 0.05899918
## 98   9.7981818 0.17000 0.06129392
## 99   9.8990909 0.17000 0.06129392
## 100 10.0000000 0.17125 0.06010696
```

```
svmPoly <- svm(Purchase ~ . , data = OJ.train, kernel = "poly", degree = 2,
cost = svmTune$best.parameters$cost)
cat("Training Error: ", 100 * errorRate(svmPoly, OJ.train,
OJ.train$Purchase), "%\n")
```

```
## Training Error:  15.125 %
```

```
cat("Test Error:", 100 * errorRate(svmPoly, OJ.test, OJ.test$Purchase), "%\n")

## Test Error: 20 %
```

-> SV poly classifier created 342 support vectors from 800 training points.
-> 168 belong to level CH.
-> 174 belong to level MM.
-> Training error is 17.25%
-> Test error is 22.22%
-> Tuning indicates optimal cost = 3.34

Part (h)

-> Best performance on training set belongs to radial kernel.
-> Best performance on testing set belongs to linear kernel.
-> Overall best performance seems to belong to the linear and radial models.

End.