Individual_assignment9

Yuhan_Xu_474154

2019/11/8

Prefix

This problem involves the OJ data set which is part of the ISLR package.

```
library(ISLR)
attach(OJ)
```

(a)

Create a training set containing a random sample of 800 observations, and a test set containing the remaining observations.

```
set.seed(1)
train = sample(1:nrow(OJ), 800)
OJ.train = OJ[train,]
OJ.test = OJ[-train,]
```

(b)

Fit a support vector classifier to the training data using cost=0.01, with Purchase as the response and the other variables as predictors. Use the summary() function to produce summary statistics, and describe the results obtained.

```
library(e1071)
svcfit1 = svm(Purchase~., data = OJ.train, kernel = "linear", cost = 0.01)
summary(svcfit1)
```

```
##
## 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:  435
##
##  ( 219 216 )
##
##
## Number of Classes:  2
```

```
##
## Levels:
##  CH MM
```

435 support vectors are used in this linear model. 219 support vectors belong to CH, indicating that customer purchased Citrus Hill Orange Juice, 216 support vectors belong to MM, indicating that customer purchased Minute Maid Orange Juice.

(c)

What are the training and test error rates?

```
pred.train1 = predict(svcfit1, OJ.train)
mean((pred.train1 != OJ.train$Purchase))
```

```
## [1] 0.175
```

```
pred.test1 = predict(svcfit1, OJ.test)
mean((pred.test1 != OJ.test$Purchase))
```

```
## [1] 0.1777778
```

The training error rate is 0.175. The test error rate is 0.178.

(d)

Use the tune() function to select an optimal cost. Consider values in the range 0.01 to 10.

```
set.seed(1)

tune.out1 = tune(svm, Purchase~., data = OJ.train, kernel = "linear",
                 ranges = list(cost = c(0.01, 0.05, 0.1, 0.5, 1, 5, 10)))
summary(tune.out1)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   0.5
##
## - best performance: 0.16875
##
## - Detailed performance results:
##    cost   error dispersion
## 1  0.01 0.17625 0.02853482
## 2  0.05 0.17625 0.02853482
## 3  0.10 0.17250 0.03162278
```

```
## 4  0.50 0.16875 0.02651650
## 5  1.00 0.17500 0.02946278
## 6  5.00 0.17250 0.03162278
## 7 10.00 0.17375 0.03197764
```

The optimal cost is 0.5.

(e)

Compute the training and test error rates using this new value for cost.

```r
pred.train2 = predict(tune.out1$best.model, OJ.train)
mean((pred.train2 != OJ.train$Purchase))
```

```
## [1] 0.165
```

```r
pred.test2 = predict(tune.out1$best.model, OJ.test)
mean((pred.test2 != OJ.test$Purchase))
```

```
## [1] 0.1555556
```

The training error rate is 0.165. The test error rate is 0.156.

(f)

Repeat parts (b) through (e) using a support vector machine with a radial kernel. Use the default value for gamma.

```r
svcfit2 = svm(Purchase~., data = OJ.train, kernel = "radial")
summary(svcfit2)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = OJ.train, kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  373
##
##  ( 188 185 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

373 support vectors are used in this linear model. 188 support vectors belong to CH, indicating that customer purchased Citrus Hill Orange Juice, 185 support vectors belong to MM, indicating that customer purchased Minute Maid Orange Juice.

```
pred.train3 = predict(svcfit2, OJ.train)
mean((pred.train3 != OJ.train$Purchase))
```

```
## [1] 0.15125
```

```
pred.test3 = predict(svcfit2, OJ.test)
mean((pred.test3 != OJ.test$Purchase))
```

```
## [1] 0.1851852
```

The training error rate is 0.151. The test error rate is 0.185.

```
set.seed(1)

tune.out2 = tune(svm, Purchase~., data = OJ.train, kernel = "radial",
                 ranges = list(gamma = c(0.5, 1, 2, 3, 4),
                               cost = c(0.01, 0.05, 0.1, 0.5, 1, 5, 10)))
summary(tune.out2)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  gamma cost
##    0.5  0.5
##
## - best performance: 0.20875
##
## - Detailed performance results:
##    gamma  cost   error dispersion
## 1    0.5  0.01 0.39375 0.04007372
## 2    1.0  0.01 0.39375 0.04007372
## 3    2.0  0.01 0.39375 0.04007372
## 4    3.0  0.01 0.39375 0.04007372
## 5    4.0  0.01 0.39375 0.04007372
## 6    0.5  0.05 0.39375 0.04007372
## 7    1.0  0.05 0.39375 0.04007372
## 8    2.0  0.05 0.39375 0.04007372
## 9    3.0  0.05 0.39375 0.04007372
## 10   4.0  0.05 0.39375 0.04007372
## 11   0.5  0.10 0.28250 0.05502525
## 12   1.0  0.10 0.34500 0.04937104
## 13   2.0  0.10 0.38625 0.04348132
## 14   3.0  0.10 0.39375 0.04007372
## 15   4.0  0.10 0.39375 0.04007372
## 16   0.5  0.50 0.20875 0.04411554
```

```
## 17   1.0   0.50 0.22125 0.04084609
## 18   2.0   0.50 0.22750 0.04073969
## 19   3.0   0.50 0.23625 0.04543387
## 20   4.0   0.50 0.25750 0.05210833
## 21   0.5   1.00 0.21375 0.03701070
## 22   1.0   1.00 0.22625 0.04466309
## 23   2.0   1.00 0.22750 0.04281744
## 24   3.0   1.00 0.22625 0.03304563
## 25   4.0   1.00 0.22750 0.03322900
## 26   0.5   5.00 0.21375 0.03928617
## 27   1.0   5.00 0.22500 0.04487637
## 28   2.0   5.00 0.23375 0.04825065
## 29   3.0   5.00 0.24625 0.03120831
## 30   4.0   5.00 0.24875 0.03606033
## 31   0.5  10.00 0.21250 0.03632416
## 32   1.0  10.00 0.23000 0.04684490
## 33   2.0  10.00 0.24000 0.04158325
## 34   3.0  10.00 0.25375 0.03335936
## 35   4.0  10.00 0.25500 0.03496029
```

The optimal gamma is 0.5. The optimal cost is 0.5.

```
pred.train4 = predict(tune.out2$best.model, OJ.train)
mean((pred.train4 != OJ.train$Purchase))
```

```
## [1] 0.1375
```

```
pred.test4 = predict(tune.out2$best.model, OJ.test)
mean((pred.test4 != OJ.test$Purchase))
```

```
## [1] 0.1962963
```

The training error rate is 0.138. The test error rate is 0.196.

(g)

Repeat parts (b) through (e) using a support vector machine with a polynomial kernel. Set degree=2.

```
svcfit3 = svm(Purchase~., data = OJ.train, kernel = "poly", degree = 2)
summary(svcfit3)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = OJ.train, kernel = "poly",
##     degree = 2)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  1
```

```
##      degree:  2
##      coef.0:  0
##
## Number of Support Vectors:  447
##
##  ( 225 222 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

447 support vectors are used in this linear model. 225 support vectors belong to CH, indicating that customer purchased Citrus Hill Orange Juice, 222 support vectors belong to MM, indicating that customer purchased Minute Maid Orange Juice.

```
pred.train5 = predict(svcfit3, OJ.train)
mean((pred.train5 != OJ.train$Purchase))
```

```
## [1] 0.1825
```

```
pred.test5 = predict(svcfit3, OJ.test)
mean((pred.test5 != OJ.test$Purchase))
```

```
## [1] 0.2222222
```

The training error rate is 0.183. The test error rate is 0.222.

```
set.seed(1)

tune.out3 = tune(svm, Purchase~., data = OJ.train, kernel = "poly", degree = 2,
                 ranges = list(gamma = c(0.5, 1, 2, 3, 4),
                               cost = c(0.01, 0.05, 0.1, 0.5, 1, 5, 10)))
summary(tune.out3)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  gamma cost
##      1 0.05
##
## - best performance: 0.175
##
## - Detailed performance results:
##    gamma  cost   error dispersion
## 1    0.5  0.01 0.20375 0.04251225
## 2    1.0  0.01 0.18000 0.03545341
```

```
## 3     2.0   0.01 0.18000 0.03073181
## 4     3.0   0.01 0.18000 0.02776389
## 5     4.0   0.01 0.18250 0.02776389
## 6     0.5   0.05 0.18375 0.03438447
## 7     1.0   0.05 0.17500 0.02763854
## 8     2.0   0.05 0.18250 0.02371708
## 9     3.0   0.05 0.18500 0.02024160
## 10    4.0   0.05 0.19000 0.02024160
## 11    0.5   0.10 0.18125 0.03240906
## 12    1.0   0.10 0.18000 0.03016160
## 13    2.0   0.10 0.18500 0.02188988
## 14    3.0   0.10 0.19250 0.02443813
## 15    4.0   0.10 0.19250 0.02513851
## 16    0.5   0.50 0.18250 0.02776389
## 17    1.0   0.50 0.18750 0.02124591
## 18    2.0   0.50 0.18875 0.02389938
## 19    3.0   0.50 0.19250 0.02581989
## 20    4.0   0.50 0.19625 0.02638523
## 21    0.5   1.00 0.18250 0.02513851
## 22    1.0   1.00 0.19250 0.02581989
## 23    2.0   1.00 0.19875 0.02316157
## 24    3.0   1.00 0.20000 0.02282177
## 25    4.0   1.00 0.20375 0.03007514
## 26    0.5   5.00 0.19000 0.02486072
## 27    1.0   5.00 0.19375 0.02585349
## 28    2.0   5.00 0.20375 0.03175973
## 29    3.0   5.00 0.21000 0.04322101
## 30    4.0   5.00 0.20375 0.04041881
## 31    0.5  10.00 0.18875 0.02389938
## 32    1.0  10.00 0.20125 0.02461509
## 33    2.0  10.00 0.20875 0.04332131
## 34    3.0  10.00 0.20375 0.04041881
## 35    4.0  10.00 0.20875 0.04126894
```

The optimal gamma is 1. The optimal cost is 0.05.

```r
pred.train6 = predict(tune.out3$best.model, OJ.train)
mean((pred.train6 != OJ.train$Purchase))
```

```
## [1] 0.1475
```

```r
pred.test6 = predict(tune.out3$best.model, OJ.test)
mean((pred.test6 != OJ.test$Purchase))
```

```
## [1] 0.1962963
```

The training error rate is 0.148. The test error rate is 0.196.

(h)

Q: Overall, which approach seems to give the best results on this data?
A: The linear model with optimal cost of 0.5 gives the best results on this data.