Chapter 9 Exercise

rachel sabol

April 9, 2018

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

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

```
library(ISLR)
library (e1071)
set.seed(42)
train=sample(nrow(OJ),800)
OJtrain=OJ[train,]
OJtest=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.

```
svmfit=svm(Purchase~.,data=OJtrain,kernel="linear",cost=0.01)
summary(svmfit)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = OJtrain, kernel = "linear",
##     cost = 0.01)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.01
##       gamma:  0.05555556
##
## Number of Support Vectors:  439
##
##  ( 219 220 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

The support vector classifier results in a model with two classes and 429 suppot vectors. 219 belong to level CH, 220 belong to level MM.

c) What are the training and test error rates?

```
pred.train=predict(svmfit,OJtrain)
table(OJtrain$Purchase,pred.train)
```

```
##     pred.train
##       CH  MM
##   CH 428  54
##   MM  78 240
```

```
(78+54)/800
```

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

```
pred.test=predict(svmfit,OJtest)
table(OJtest$Purchase,pred.test)
```

```
##     pred.test
##       CH  MM
##   CH 150  21
##   MM  29  70
```

```
(29+21)/(150+21+29+70)
```

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

The training error rate is 16.5% and the testing error rate is about 18.5%.

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

```
tune.out=tune(svm,Purchase~.,data=OJtrain,kernel="linear",ranges=list(cost=c(0.01,0.1,1,10)))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     1
##
## - best performance: 0.17125
##
## - Detailed performance results:
##    cost   error dispersion
## 1  0.01 0.17375 0.04910660
## 2  0.10 0.17375 0.05219155
## 3  1.00 0.17125 0.05172376
## 4 10.00 0.17500 0.05137012
```

The cost with the lowest error is 1.

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

```
svmfit=svm(Purchase~.,data=OJtrain,kernel="linear",cost=1)
summary(svmfit)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = OJtrain, kernel = "linear",
##     cost = 1)
```

```
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##       gamma:  0.05555556
##
## Number of Support Vectors:  335
##
##  ( 167 168 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

```r
pred.train=predict(svmfit,OJtrain)
table(OJtrain$Purchase,pred.train)
```

```
##     pred.train
##       CH  MM
##   CH 425  57
##   MM  73 245
```

```r
(73+57)/800
```

```
## [1] 0.1625
```

```r
pred.test=predict(svmfit,OJtest)
table(OJtest$Purchase,pred.test)
```

```
##     pred.test
##       CH  MM
##   CH 149  22
##   MM  25  74
```

```r
(25+22)/(149+74+25+22)
```

```
## [1] 0.1740741
```

The new training error is 16.25% and the testing error is 17.4%, a slight improvement over the original cost.

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

```r
svmfit=svm(Purchase~.,data=OJtrain,kernel="radial",cost=0.01)
summary(svmfit)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = OJtrain, kernel = "radial",
##     cost = 0.01)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
```

```
##        cost:  0.01
##       gamma:  0.05555556
##
## Number of Support Vectors:  638
##
##  ( 318 320 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

The support vector classifier results in a model with two classes and 638 suppot vectors. 318 belong to level CH, 320 belong to level MM.

c) What are the training and test error rates?

```r
pred.train=predict(svmfit,OJtrain)
table(OJtrain$Purchase,pred.train)
```

```
##     pred.train
##       CH  MM
##   CH 482   0
##   MM 318   0
```

```r
(318)/800
```

```
## [1] 0.3975
```

```r
pred.test=predict(svmfit,OJtest)
table(OJtest$Purchase,pred.test)
```

```
##     pred.test
##       CH  MM
##   CH 171   0
##   MM  99   0
```

```r
99/200
```

```
## [1] 0.495
```

The training error rate is about 40% and the testing error rate is 49.5%.

```r
tune.out=tune(svm,Purchase~.,data=OJtrain,kernel="radial",ranges=list(cost=c(0.01,0.1,1,10)))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     1
##
## - best performance: 0.1775
##
## - Detailed performance results:
```

```
##    cost   error dispersion
## 1  0.01 0.39750 0.04993051
## 2  0.10 0.18125 0.05212498
## 3  1.00 0.17750 0.05062114
## 4 10.00 0.18250 0.04794383
```

The cost with the lowest error is 1.

```r
svmfit=svm(Purchase~.,data=OJtrain,kernel="radial",cost=1)
summary(svmfit)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = OJtrain, kernel = "radial",
##     cost = 1)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.05555556
##
## Number of Support Vectors:  382
##
##  ( 191 191 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

```r
pred.train=predict(svmfit,OJtrain)
table(OJtrain$Purchase,pred.train)
```

```
##     pred.train
##       CH  MM
##   CH 443  39
##   MM  78 240
```

```r
(78+39)/800
```

```
## [1] 0.14625
```

```r
pred.test=predict(svmfit,OJtest)
table(OJtest$Purchase,pred.test)
```

```
##     pred.test
##       CH  MM
##   CH 152  19
##   MM  24  75
```

```r
(24+19)/200
```

```
## [1] 0.215
```

The new training error is about 14.6% and the testing error is 21.5%.

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

```
svmfit=svm(Purchase~.,data=OJtrain,kernel="polynomial",degree=2,cost=0.01)
summary(svmfit)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = OJtrain, kernel = "polynomial",
##     degree = 2, cost = 0.01)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  0.01
##      degree:  2
##       gamma:  0.05555556
##      coef.0:  0
##
## Number of Support Vectors:  642
##
##  ( 318 324 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

The support vector classifier results in a model with two classes and 642 suppot vectors. 318 belong to level CH, 324 belong to level MM.

```
pred.train=predict(svmfit,OJtrain)
table(OJtrain$Purchase,pred.train)
```

```
##     pred.train
##       CH  MM
##   CH 482   0
##   MM 318   0
```

```
318/800
```

```
## [1] 0.3975
```

```
pred.test=predict(svmfit,OJtest)
table(OJtest$Purchase,pred.test)
```

```
##     pred.test
##       CH  MM
##   CH 171   0
##   MM  99   0
```

```
99/200
```

```
## [1] 0.495
```

The training error rate is about 40% and the testing error rate is 49.5%.

```r
tune.out=tune(svm,Purchase~.,data=OJtrain,kernel="polynomial",degree=2,ranges=list(cost=c(0.01,0.1,1,10
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##    10
##
## - best performance: 0.19375
##
## - Detailed performance results:
##    cost   error dispersion
## 1  0.01 0.39750 0.06635343
## 2  0.10 0.33125 0.06827487
## 3  1.00 0.21125 0.04059026
## 4 10.00 0.19375 0.03294039
```

The cost with the lowest error is 10.

```r
svmfit=svm(Purchase~.,data=OJtrain,kernel="polynomial",degree=2,cost=10)
summary(svmfit)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = OJtrain, kernel = "polynomial",
##     degree = 2, cost = 10)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  10
##      degree:  2
##       gamma:  0.05555556
##      coef.0:  0
##
## Number of Support Vectors:  350
##
##  ( 170 180 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

```r
pred.train=predict(svmfit,OJtrain)
table(OJtrain$Purchase,pred.train)
```

```
##     pred.train
##       CH  MM
##   CH 441  41
```

```
##   MM  80 238
```

```r
(80+41)/800
```

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

```r
pred.test=predict(svmfit,OJtest)
table(OJtest$Purchase,pred.test)
```

```
##     pred.test
##       CH  MM
##   CH 152  19
##   MM  29  70
```

```r
(29+19)/200
```

```
## [1] 0.24
```

h) Overall, which approach seems to give the best results on this data?

While the radial and polynomial approaches fit the training data better, the linear fit has the lowest testing error. The radial and polynomial models are likely overfitting. Therefore, the linear model has the best results.