Individual Assignment 9

Charlie Ling

2021/11/13

Problem #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)
set.seed(1)
train=sample(nrow(OJ),800)
test=-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)

svcfit=svm(Purchase~.,data = OJ[train,], kernel = "linear", cost=0.01)

summary(svcfit)

##
## 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

#There are 435 support vectors, 219 are in CH side and 216 are in MM side
```

(c) What are the training and test error rates?

```
mean(predict(svcfit,OJ[train,])!=OJ[train,]$Purchase)

## [1] 0.175

mean(predict(svcfit,OJ[-train,])!=OJ[-train,]$Purchase)

## [1] 0.1777778

# training error rate is 0.175
# test error rate is 0.1777778
```

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

```
set.seed(1)

tune.out = tune(svm, Purchase~.,data = OJ[train,], kernel = "linear",
ranges=list(cost=c(0.1,1,10)))

summary(tune.out)

##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   0.1
##
## - best performance: 0.1725
##
## - Detailed performance results:
##   cost   error dispersion
## 1  0.1 0.17250 0.03162278
## 2  1.0 0.17500 0.02946278
## 3 10.0 0.17375 0.03197764

bestmod = tune.out$best.model

summary(bestmod)

##
## Call:
## best.tune(method = svm, train.x = Purchase ~ ., data = OJ[train,
##     ], ranges = list(cost = c(0.1, 1, 10)), kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.1
```

```
##
## Number of Support Vectors:  342
##
##  ( 171 171 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

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

```
mean(predict(bestmod,OJ[train,])!=OJ[train,]$Purchase)

## [1] 0.165

mean(predict(bestmod,OJ[-train,])!=OJ[-train,]$Purchase)

## [1] 0.162963

# training error rate is 0.165
# test error rate is 0.162963
```

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

```
svmfit=svm(Purchase~.,data = OJ[train,], kernel = "radial", cost=0.01)
summary(svmfit)

##
## Call:
## svm(formula = Purchase ~ ., data = OJ[train, ], kernel = "radial",
##     cost = 0.01)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  0.01
##
## Number of Support Vectors:  634
##
##  ( 319 315 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM

#There are 634 support vectors, 319 are in CH side and 315 are in MM side
mean(predict(svmfit,OJ[train,])!=OJ[train,]$Purchase)
```

```
## [1] 0.39375

mean(predict(svmfit,OJ[-train,])!=OJ[-train,]$Purchase)

## [1] 0.3777778

# training error rate is 0.39375
# test error rate is 0.3777778

set.seed(1)
tune.out = tune(svm, Purchase~.,data = OJ[train,], kernel = "radial",
ranges=list(cost=c(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  0.1 0.18625 0.02853482
## 2  1.0 0.17125 0.02128673
## 3 10.0 0.18625 0.02853482

bestmod = tune.out$best.model
summary(bestmod)

##
## Call:
## best.tune(method = svm, train.x = Purchase ~ ., data = OJ[train,
##     ], ranges = list(cost = c(0.1, 1, 10)), 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
```

```
mean(predict(bestmod,OJ[train,])!=OJ[train,]$Purchase)
```

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

```
mean(predict(bestmod,OJ[-train,])!=OJ[-train,]$Purchase)
```

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

```
# training error rate is 0.15125
# test error rate is 0.1851852
```

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

```
svmfit=svm(Purchase~.,data = OJ[train,], kernel = "polynomial", cost=0.01,
degree=2)
summary(svmfit)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = OJ[train, ], kernel = "polynomial",
##     cost = 0.01, degree = 2)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  0.01
##      degree:  2
##      coef.0:  0
##
## Number of Support Vectors:  636
##
##  ( 321 315 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

```
#There are 636 support vectors, 321 are in CH side and 315 are in MM side
mean(predict(svmfit,OJ[train,])!=OJ[train,]$Purchase)
```

```
## [1] 0.3725
```

```
mean(predict(svmfit,OJ[-train,])!=OJ[-train,]$Purchase)
```

```
## [1] 0.3666667
```

```r
# training error rate is 0.3725
# test error rate is 0.3666667

set.seed(1)
tune.out = tune(svm, Purchase~.,data = OJ[train,], kernel = "polynomial",
ranges=list(cost=c(0.1,1,10), degree = 2))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost degree
##    10      2
##
## - best performance: 0.18125
##
## - Detailed performance results:
##   cost degree   error dispersion
## 1  0.1      2 0.32125 0.05001736
## 2  1.0      2 0.20250 0.04116363
## 3 10.0      2 0.18125 0.02779513
```

```r
bestmod = tune.out$best.model
summary(bestmod)
```

```
##
## Call:
## best.tune(method = svm, train.x = Purchase ~ ., data = OJ[train,
##     ], ranges = list(cost = c(0.1, 1, 10), degree = 2), kernel =
"polynomial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  10
##      degree:  2
##      coef.0:  0
##
## Number of Support Vectors:  340
##
##  ( 171 169 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

```r
mean(predict(bestmod,OJ[train,])!=OJ[train,]$Purchase)

## [1] 0.15

mean(predict(bestmod,OJ[-train,])!=OJ[-train,]$Purchase)

## [1] 0.1888889

# training error rate is 0.15
# test error rate is 0.1888889
```

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

```r
# support vector classifier (linear kernel) with cost=0.1 seems to give the best results
```