BUAN6357_Homework1_Bhatia

Rishabh Bhatia

09/10/2020

##Loading the data in a data frame

```
pacman::p_load(caret, e1071, reshape2, GGally)
theme_set(theme_classic())

juice.df <- read.csv('juice.csv', header = TRUE)
```

##Creating a training set containing a random sample of 80% of the observations in the data set using createDataPartition(). Test data set contains the remaining observations. Used seed of 123 is used as mentioned in the homework for reproducibility

```
set.seed(123)
train.index <- createDataPartition(juice.df$Purchase, p = 0.8, list = FALSE)
juice.train <- juice.df[train.index, ]
juice.test <- juice.df[-train.index, ]
```

##2. Using SVM model with a linear kernel on the training data using cost=0.01. Purchase is used as response and the other variables as predictors. The summary function describe the results obtained.

```
set.seed(123)
svm.linear <- svm(Purchase ~., data = juice.train, kernel = 'linear', cost = 0.01)

summary(svm.linear)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = juice.train, kernel = "linear",
##     cost = 0.01)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.01
##
## Number of Support Vectors:  446
##
##  ( 224 222 )
##
##
```

```
## Number of Classes:  2
##
## Levels:
##  CH MM
```

##Out of 800 data points, the number of support vectors used to construct the hyperplane are 446. Out of which 224 of class "CH" and 222 of class "MM" is used. We can see that the SVM Type is C-Classification

##The followeing code helps us determine the training and test classification error rates

```
train.pred.svm.linear <- predict(svm.linear, juice.train)
print(paste0("Training classification error rate is ", mean(train.pred.svm.linear != juice.train$Purcha
```

```
## [1] "Training classification error rate is 0.17"
```

```
test.pred.svm.linear <- predict(svm.linear, juice.test)
print(paste0("Test classification error rate is ", mean(test.pred.svm.linear != juice.test$Purchase)))
```

```
## [1] "Test classification error rate is 0.165"
```

##The training classification error rate is 0.17(17%) and the test classification error is 0.165(16.5%)

##We use the tune() function to select an optimal cost. Selecting values in the range 0.01 to 10. Once again seed is 123

```
set.seed(123)
linear.tune.out <- tune(svm, Purchase ~., data = juice.train, kernel = 'linear',
                        ranges = list(cost = c(seq(0.01, 10, 0.01))))

svm.tune.linear <- linear.tune.out$best.model

summary(svm.tune.linear)
```

```
##
## Call:
## best.tune(method = svm, train.x = Purchase ~ ., data = juice.train,
##     ranges = list(cost = c(seq(0.01, 10, 0.01))), kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  2.31
##
## Number of Support Vectors:  342
##
##  ( 170 172 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

##The best model after running the above tune() function has cost = 2.31. The number of support vectors used to construct the hyperplane are 342 in which 170 are of class “CH” and 172 are of class “MM”.

##We compute the training and test error rates using this new cost

```
train.pred.svm.tune.linear <- predict(svm.tune.linear, juice.train)
print(paste0("Training classification error rate is ", mean(train.pred.svm.tune.linear != juice.train$Pu
```

```
## [1] "Training classification error rate is 0.1625"
```

```
test.pred.svm.tune.linear <- predict(svm.tune.linear, juice.test)
print(paste0("Test classification error rate is ", mean(test.pred.svm.tune.linear != juice.test$Purchase
```

```
## [1] "Test classification error rate is 0.175"
```

##The training classification error rate is 0.1625(16.25%) and the test classification error is 0.175(17.5%)

##Now we repeat the above steps using SVM but this time with a radial kernel

```
set.seed(123)
svm.radial <- svm(Purchase ~., data = juice.train, kernel = 'radial', cost = 0.01)

summary(svm.radial)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = juice.train, kernel = "radial",
##     cost = 0.01)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  0.01
##
## Number of Support Vectors:  626
##
##  ( 312 314 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

##Total number of support vectors used are 626. Out of which 312 are of class “CH” and 314 of class “MM”

##Finding the training classification error rate and the test classification error for Radial Kernel

```
train.pred.svm.radial <- predict(svm.radial, juice.train)
print(paste0("Training classification error rate is ", mean(train.pred.svm.radial != juice.train$Purcha
```

```
## [1] "Training classification error rate is 0.39"
```

```r
test.pred.svm.radial <- predict(svm.radial, juice.test)
print(paste0("Test classification error rate is ", mean(test.pred.svm.radial != juice.test$Purchase)))
```

```
## [1] "Test classification error rate is 0.39"
```

##The training classification error rate is 0.39(39%) and the test classification error is 0.39(39%) for Radial kernel

##We use the tune() function to select an optimal cost. Selecting values in the range 0.01 to 10. Once again seed is 123

```r
set.seed(123)
radial.tune.out <- tune(svm, Purchase ~., data = juice.train, kernel = 'radial',
                        ranges = list(cost = c(seq(0.01, 10, 0.01))))

svm.tune.radial <- radial.tune.out$best.model

summary(svm.tune.radial)
```

```
##
## Call:
## best.tune(method = svm, train.x = Purchase ~ ., data = juice.train,
##     ranges = list(cost = c(seq(0.01, 10, 0.01))), kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  0.6
##
## Number of Support Vectors:  405
##
##  ( 200 205 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

##The best model after running the above tune() function has cost = 0.6. The number of support vectors used to construct the hyperplane are 405 in which 200 are of class "CH" and 205 are of class "MM".

##We compute the training and test classification error rates using this new cost

```r
train.pred.svm.tune.radial <- predict(svm.tune.radial, juice.train)
print(paste0("Training classification error rate is ", mean(train.pred.svm.tune.radial != juice.train$Pu
```

```
## [1] "Training classification error rate is 0.155"
```

```r
test.pred.svm.tune.radial <- predict(svm.tune.radial, juice.test)
print(paste0("Test classification error rate is ", mean(test.pred.svm.tune.radial != juice.test$Purchase
```

```
## [1] "Test classification error rate is 0.15"
```

##The training classification error rate using new cost is 0.155(15.5%) and test classification error is 0.15(15%)

##Now we repeat the above steps using SVM but this time with a polynomial kernel, degree is set to 2

```r
set.seed(123)
svm.poly <- svm(Purchase ~., data = juice.train, kernel = 'polynomial', cost = 0.01, degree = 2)

summary(svm.poly)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = juice.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:  629
##
##  ( 312 317 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

##Out of 800 data points, the number of support vectors used to construct the hyperplane are 629. Out of which 312 of class "CH" and 317 of class "MM" is used.

##Finding the training classification error rate and the test classification error for Polynomial Kernel

```r
train.pred.svm.poly <- predict(svm.poly, juice.train)
print(paste0("Training classification error rate is ", mean(train.pred.svm.poly != juice.train$Purchase)
```

```
## [1] "Training classification error rate is 0.39"
```

```r
test.pred.svm.poly <- predict(svm.poly, juice.test)
print(paste0("Test classification error rate is ", mean(test.pred.svm.poly != juice.test$Purchase)))
```

```
## [1] "Test classification error rate is 0.39"
```

##The training classification error rate is 0.39(39%) and test classificationerror is 0.39(39%)

##We use the tune() function to select an optimal cost. Selecting values in the range 0.01 to 10. Once again seed is 123

```r
set.seed(123)
poly.tune.out <- tune(svm, Purchase ~., data = juice.train, kernel = 'polynomial', degree = 2,
                      ranges = list(cost = c(seq(0.01, 10, 0.01))))

svm.tune.poly <- poly.tune.out$best.model

summary(svm.tune.poly)
```

```
##
## Call:
## best.tune(method = svm, train.x = Purchase ~ ., data = juice.train,
##     ranges = list(cost = c(seq(0.01, 10, 0.01))), kernel = "polynomial",
##     degree = 2)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  8.65
##      degree:  2
##      coef.0:  0
##
## Number of Support Vectors:  354
##
##  ( 174 180 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

##The best model selected from tuning has cost = 8.65. Out of 800 data points, the number of support vectors used to construct the hyperplane are 354 in which 174 are of class "CH" and 180 are of class "MM".

##We compute the training and test classification error rates using this new cost

```r
train.pred.svm.tune.poly <- predict(svm.tune.poly, juice.train)
print(paste0("Training classification rate is ", mean(train.pred.svm.tune.poly != juice.train$Purchase)
```

```
## [1] "Training classification rate is 0.16125"
```

```r
test.pred.svm.tune.poly <- predict(svm.tune.poly, juice.test)
print(paste0("Test classification error rate is ", mean(test.pred.svm.tune.poly != juice.test$Purchase)
```

```
## [1] "Test classification error rate is 0.17"
```

##The training classification error rate using new cost is 0.16125(16.125%) and test classification error is 0.17(17%)

##Overall, which approach seems to give the best results on this data? ##Overall SVM with Radial kernel that has cost = 0.6 gives us the best results on this data. The training and test classification error rates are the lowest with 15.5% and 15% respectively.