SVM Assignment

Sujeeth Shetty

2/10/2020

```r
pacman::p_load(e1071, ggplot2, caret, rmarkdown, corrplot)
#search()
theme_set(theme_classic())
options(digits = 3)
```

Read juice.csv

```
## [1] 1000   18
```

```
## 'data.frame':    1000 obs. of  18 variables:
##  $ Purchase      : Factor w/ 2 levels "CH","MM": 2 1 2 1 1 2 1 1 1 1 ...
##  $ WeekofPurchase: int  237 258 242 271 276 240 248 270 266 274 ...
##  $ StoreID       : int  2 7 3 2 2 1 3 1 2 7 ...
##  $ PriceCH       : num  1.75 1.86 1.99 1.86 1.99 1.75 1.99 1.86 1.86 1.86 ...
##  $ PriceMM       : num  1.99 2.18 2.23 2.18 2.18 1.99 2.23 2.18 2.18 2.13 ...
##  $ DiscCH        : num  0 0 0 0 0 0 0 0 0 0.47 ...
##  $ DiscMM        : num  0 0 0 0.06 0 0.3 0 0 0 0.54 ...
##  $ SpecialCH     : int  0 0 0 0 0 0 0 0 0 1 ...
##  $ SpecialMM     : int  0 0 0 0 1 1 0 0 0 0 ...
##  $ LoyalCH       : num  0.4 0.90814 0.00721 0.78839 0.97251 ...
##  $ SalePriceMM   : num  1.99 2.18 2.23 2.12 2.18 1.69 2.23 2.18 2.18 1.59 ...
##  $ SalePriceCH   : num  1.75 1.86 1.99 1.86 1.99 1.75 1.99 1.86 1.86 1.39 ...
##  $ PriceDiff     : num  0.24 0.32 0.24 0.26 0.19 -0.06 0.24 0.32 0.32 0.2 ...
##  $ Store7        : Factor w/ 2 levels "No","Yes": 1 2 1 1 1 1 1 1 1 2 ...
##  $ PctDiscMM     : num  0 0 0 0.0275 0 ...
##  $ PctDiscCH     : num  0 0 0 0 0 ...
##  $ ListPriceDiff : num  0.24 0.32 0.24 0.32 0.19 0.24 0.24 0.32 0.32 0.27 ...
##  $ STORE         : int  2 0 3 2 2 1 3 1 2 0 ...
```

```r
set.seed(123)
trainindex <- createDataPartition(juice.df$Purchase, p=0.8, list= FALSE)
juice_train <- juice.df[trainindex, ]
juice_test <- juice.df[-trainindex, ]
```

```r
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
```

2) SVM with a linear kernel creates 446 support vectors out of 800 training points. Out of these, 224 belong to level CH and remaining 222 belong to level MM

```r
 ## Performance Evaluation train datapoints##
pred_train <- predict(svm_linear, juice_train)

# confusion matrix
conf.matrix.train <- table(Predicted = pred_train, Actual = juice_train$Purchase)
conf.matrix.train
```

```
##          Actual
## Predicted  CH  MM
##        CH 433  81
##        MM  55 231
```

```r
# Training Error
1-(sum(diag(conf.matrix.train))) / sum(conf.matrix.train)
```

```
## [1] 0.17
```

```r
## Performance Evaluation Test datapoints##
pred_test <- predict(svm_linear, juice_test)

# confusion matrix
conf.matrix.test <- table(Predicted = pred_test, Actual = juice_test$Purchase)
conf.matrix.test
```

```
##          Actual
## Predicted  CH  MM
##        CH 108  19
##        MM  14  59
```

```r
# Test Error
1-(sum(diag(conf.matrix.test))) / sum(conf.matrix.test)
```

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

3) Training Error Rate : 17% ; Test Error Rate: 16.5%

```r
set.seed(123)
tune_linear_svm <- tune(svm, Purchase~., data = juice_train,kernel = "linear",
     ranges = list(cost = 10^seq(-2,1, by=0.25)))

summary(tune_linear_svm)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##  3.16
##
## - best performance: 0.167
##
## - Detailed performance results:
##       cost error dispersion
## 1   0.0100 0.184     0.0400
## 2   0.0178 0.181     0.0396
## 3   0.0316 0.180     0.0355
## 4   0.0562 0.173     0.0322
## 5   0.1000 0.172     0.0337
## 6   0.1778 0.174     0.0336
## 7   0.3162 0.174     0.0314
## 8   0.5623 0.176     0.0285
## 9   1.0000 0.174     0.0291
## 10  1.7783 0.170     0.0290
## 11  3.1623 0.167     0.0296
## 12  5.6234 0.172     0.0262
## 13 10.0000 0.175     0.0276
```

4) Tuning shows the optimal cost is 3.1623

```r
## Best SVM Model ##
best_linear_svm <- tune_linear_svm$best.model
summary(best_linear_svm)
```

```
##
## Call:
## best.tune(method = svm, train.x = Purchase ~ ., data = juice_train,
##     ranges = list(cost = 10^seq(-2, 1, by = 0.25)), kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  3.16
##
## Number of Support Vectors:  341
##
```

```
##  ( 171 170 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

```r
#prediction for train datapoint
best_train_pred <- predict(best_linear_svm, juice_train)

# confusion matrix
conf.matrix.train <- table(Predicted = best_train_pred, Actual = juice_train$Purchase)
conf.matrix.train
```

```
##          Actual
## Predicted  CH  MM
##        CH 431  76
##        MM  57 236
```

```r
# Training Error
1-(sum(diag(conf.matrix.train))) / sum(conf.matrix.train)
```

```
## [1] 0.166
```

```r
#prediction for test datapoint
best_test_pred <- predict(best_linear_svm, juice_test)

# confusion matrix
conf.matrix.test <- table(Predicted = best_test_pred, Actual = juice_test$Purchase)
conf.matrix.test
```

```
##          Actual
## Predicted  CH  MM
##        CH 108  21
##        MM  14  57
```

```r
#Test Error
1-(sum(diag(conf.matrix.test))) / sum(conf.matrix.test)
```

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

5) The training error decreases to 16.66% and test error slightly increases to 17.5%

```r
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
```

8) SVM with a radial kernel creates 626 support vectors out of 800 training points. Out of these, 312 belong to level CH and remaining 314 belong to level MM

```r
 ## Performance Evaluation train datapoints##
pred_train <- predict(svm_radial, juice_train)

# confusion matrix
conf.matrix.train <- table(Predicted = pred_train, Actual = juice_train$Purchase)
conf.matrix.train
```

```
##          Actual
## Predicted  CH  MM
##        CH 488 312
##        MM   0   0
```

```r
# Training Error
1-(sum(diag(conf.matrix.train)) / sum(conf.matrix.train))
```

```
## [1] 0.39
```

```r
## Performance Evaluation Test datapoints##
pred_test <- predict(svm_radial, juice_test)

# confusion matrix
conf.matrix.test <- table(Predicted = pred_test, Actual = juice_test$Purchase)
conf.matrix.test
```

```
##          Actual
## Predicted  CH  MM
##        CH 122  78
##        MM   0   0
```

```r
# Test Error
1-(sum(diag(conf.matrix.test))) / sum(conf.matrix.test)
```

```
## [1] 0.39
```

8) Training Error Rate : 39% ; Test Error Rate: 39%

```r
set.seed(123)
tune_radial_svm <- tune(svm, Purchase~., data = juice_train,kernel = "radial",
     ranges = list(cost = 10^seq(-2,1, by=0.25)))

summary(tune_radial_svm)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##   cost
##  0.562
##
## - best performance: 0.181
##
## - Detailed performance results:
##       cost error dispersion
## 1   0.0100 0.390     0.0642
## 2   0.0178 0.390     0.0642
## 3   0.0316 0.379     0.0662
## 4   0.0562 0.217     0.0345
## 5   0.1000 0.195     0.0369
## 6   0.1778 0.188     0.0386
## 7   0.3162 0.186     0.0309
## 8   0.5623 0.181     0.0319
## 9   1.0000 0.184     0.0301
## 10  1.7783 0.188     0.0358
## 11  3.1623 0.184     0.0413
## 12  5.6234 0.186     0.0388
## 13 10.0000 0.195     0.0378
```

8) Tuning shows the optimal cost is 0.5623

```r
## Best SVM Model ##
best_radial_svm <- tune_radial_svm$best.model
summary(best_radial_svm)
```

```
##
## Call:
## best.tune(method = svm, train.x = Purchase ~ ., data = juice_train,
##     ranges = list(cost = 10^seq(-2, 1, by = 0.25)), kernel = "radial")
##
```

```
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  0.562
##
## Number of Support Vectors:  408
##
##  ( 202 206 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

```r
#prediction for train datapoint
best_train_pred <- predict(best_radial_svm, juice_train)

# confusion matrix
conf.matrix.train <- table(Predicted = best_train_pred, Actual = juice_train$Purchase)
conf.matrix.train
```

```
##          Actual
## Predicted  CH  MM
##        CH 446  83
##        MM  42 229
```

```r
# Training Error
1-(sum(diag(conf.matrix.train))) / sum(conf.matrix.train)
```

```
## [1] 0.156
```

```r
#prediction for test datapoint
best_test_pred <- predict(best_radial_svm, juice_test)

# confusion matrix
conf.matrix.test <- table(Predicted = best_test_pred, Actual = juice_test$Purchase)
conf.matrix.test
```

```
##          Actual
## Predicted  CH  MM
##        CH 113  21
##        MM   9  57
```

```r
#Test Error
1-(sum(diag(conf.matrix.test))) / sum(conf.matrix.test)
```

```
## [1] 0.15
```

8) The training error decreases to 15.6% and test error slightly increases to 15% which is better than linear kernel

---

```r
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
```

9) SVM with a polynomial kernel creates 629 support vectors out of 800 training points. Out of these, 312 belong to level CH and remaining 317 belong to level MM

```r
 ## Performance Evaluation train datapoints##
pred_train <- predict(svm_poly, juice_train)

# confusion matrix
conf.matrix.train <- table(Predicted = pred_train, Actual = juice_train$Purchase)
conf.matrix.train
```

```
##          Actual
## Predicted  CH  MM
##        CH 488 312
##        MM   0   0
```

```r
# Training Error
1-(sum(diag(conf.matrix.train))) / sum(conf.matrix.train)
```

```
## [1] 0.39
```

```r
## Performance Evaluation Test datapoints##
pred_test <- predict(svm_poly, juice_test)

# confusion matrix
conf.matrix.test <- table(Predicted = pred_test, Actual = juice_test$Purchase)
conf.matrix.test
```

```
##          Actual
## Predicted  CH  MM
##        CH 122  78
##        MM   0   0
```

```r
# Test Error
1-(sum(diag(conf.matrix.test)) / sum(conf.matrix.test)
```

```
## [1] 0.39
```

9) Training Error Rate : 39% ; Test Error Rate: 39%

```r
set.seed(123)
tune_poly_svm <- tune(svm, Purchase~., data = juice_train,kernel = "polynomial", degree=2,
     ranges = list(cost = 10^seq(-2,1, by=0.25)))

summary(tune_poly_svm)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     1
##
## - best performance: 0.201
##
## - Detailed performance results:
##       cost error dispersion
## 1   0.0100 0.390     0.0642
## 2   0.0178 0.376     0.0676
## 3   0.0316 0.365     0.0597
## 4   0.0562 0.334     0.0441
## 5   0.1000 0.316     0.0559
## 6   0.1778 0.264     0.0605
## 7   0.3162 0.220     0.0369
## 8   0.5623 0.205     0.0284
## 9   1.0000 0.201     0.0346
## 10  1.7783 0.205     0.0405
## 11  3.1623 0.201     0.0491
## 12  5.6234 0.201     0.0393
## 13 10.0000 0.203     0.0372
```

9) Tuning shows the optimal cost is 1

```r
## Best SVM Model ##
best_poly_svm <- tune_poly_svm$best.model
summary(best_poly_svm)
```

```
##
## Call:
## best.tune(method = svm, train.x = Purchase ~ ., data = juice_train,
##     ranges = list(cost = 10^seq(-2, 1, by = 0.25)), kernel = "polynomial",
##     degree = 2)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  1
##      degree:  2
##      coef.0:  0
##
## Number of Support Vectors:  456
##
##  ( 223 233 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

```r
#prediction for train datapoint
best_train_pred <- predict(best_poly_svm, juice_train)

# confusion matrix
conf.matrix.train <- table(Predicted = best_train_pred, Actual = juice_train$Purchase)
conf.matrix.train
```

```
##          Actual
## Predicted  CH  MM
##        CH 452 109
##        MM  36 203
```

```r
# Training Error
1-(sum(diag(conf.matrix.train)) / sum(conf.matrix.train)
```

```
## [1] 0.181
```

```r
#prediction for test datapoint
best_test_pred <- predict(best_poly_svm, juice_test)

# confusion matrix
conf.matrix.test <- table(Predicted = best_test_pred, Actual = juice_test$Purchase)
conf.matrix.test
```

```
##          Actual
## Predicted  CH  MM
##        CH 116  31
##        MM   6  47
```

```r
#Test Error
1-(sum(diag(conf.matrix.test))) / sum(conf.matrix.test)
```

```
## [1] 0.185
```

9) The training error decreases to 18.1% and test error slightly increases to 18.5% which is worse than radial & linear kernel

10) Overall, radial basis kernel produced minimum misclassification error on both train and test data.