Applied Data Mining: Homework #7

Keith Hickman

Due on December 6, 2017

Problem 1

In this problem, you are asked to use SVM to predict whether a given car gets high or low gas mileage based on the Auto data set. The data set can be obtained as follows:

```
##install.packages("ISLR")
library(ISLR)
View(Auto)
```

1.1

Create a binary variable that takes on a 1 for cars with gas mileage above the median, and a 0 for cars with gas mileage below the median. Add this variable to the data as a new variable and name it as `"mpglevel"` (`mpglevel` is the response variable for questions 1.2 and 1.3).

R Code

```
Auto$mpglevel <- as.factor(Auto$mpg >= median(Auto$mpg))
print(Auto$mpglevel)
```

```
##   [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [386] TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## Levels: FALSE TRUE
```

```
summary(Auto)
```

```
##       mpg          cylinders      displacement     horsepower
##  Min.   : 9.00   Min.   :3.000   Min.   : 68.0   Min.   : 46.0
##  1st Qu.:17.00   1st Qu.:4.000   1st Qu.:105.0   1st Qu.: 75.0
##  Median :22.75   Median :4.000   Median :151.0   Median : 93.5
##  Mean   :23.45   Mean   :5.472   Mean   :194.4   Mean   :104.5
##  3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:275.8   3rd Qu.:126.0
##  Max.   :46.60   Max.   :8.000   Max.   :455.0   Max.   :230.0
##
##      weight      acceleration        year           origin
##  Min.   :1613   Min.   : 8.00   Min.   :70.00   Min.   :1.000
##  1st Qu.:2225   1st Qu.:13.78   1st Qu.:73.00   1st Qu.:1.000
##  Median :2804   Median :15.50   Median :76.00   Median :1.000
##  Mean   :2978   Mean   :15.54   Mean   :75.98   Mean   :1.577
```

```
##  3rd Qu.:3615   3rd Qu.:17.02   3rd Qu.:79.00   3rd Qu.:2.000
##  Max.   :5140   Max.   :24.80   Max.   :82.00   Max.   :3.000
##
##                 name      mpglevel
##  amc matador       :  5   FALSE:196
##  ford pinto        :  5   TRUE :196
##  toyota corolla    :  5
##  amc gremlin       :  4
##  amc hornet        :  4
##  chevrolet chevette:  4
##  (Other)           :365
```

1.2

Fit a linear support vector classifier to the data with various values of cost (cost = c(0.01, 0.1, 1, 5,10, 100)), in order to predict whether a car gets high or low gas mileage. Report the cross-validation errors associated with different values of this parameter. Comment on your results, i.e., what is the cost value for the model that has the lowest cross-validation error?

R Code

```
library(e1071)
set.seed(123456)
rndSample <- sample(1:nrow(Auto), 300)
tr <- Auto[rndSample, ]
ts <- Auto[-rndSample, ]

# the default svm () uses radial kernel with constraints violations of cost o
f 1
## ??svm

#Beginning with a cost of .01
s.01 <- svm(mpglevel ~ ., tr,C=.01)
ps.01 <- predict(s.01, ts)
cm.01 <- table(ps.01, ts$mpglevel) #confusion matrix
cm.01

##
## ps.01   FALSE TRUE
##   FALSE    35    1
##   TRUE      3   53

100*(1-sum(diag(cm.01))/sum(cm.01))

## [1] 4.347826

#Cost of .1
s.1 <- svm(mpglevel ~ ., tr,C=.1)
ps.1 <- predict(s.1, ts)
```

```r
cm.1 <- table(ps.1, ts$mpglevel) #confusion matrix
100*(1-sum(diag(cm.1))/sum(cm.1))

## [1] 4.347826

#Default cost of 1
s1 <- svm(mpglevel ~ ., tr,C=1)
ps1 <- predict(s1, ts)
cm1 <- table(ps1, ts$mpglevel) #confusion matrix
100*(1-sum(diag(cm1))/sum(cm1))

## [1] 4.347826

##Cost of 5
s5 <- svm(mpglevel ~ ., tr,C=5)
ps5 <- predict(s5, ts)
cm5 <- table(ps5, ts$mpglevel) #confusion matrix
100*(1-sum(diag(cm5))/sum(cm5))

## [1] 4.347826

## Cost of 10
s10 <- svm(mpglevel ~ ., tr,C=10)
ps10 <- predict(s10, ts)
cm10 <- table(ps10, ts$mpglevel) #confusion matrix
100*(1-sum(diag(cm10))/sum(cm10))

## [1] 4.347826

## Cost of 100
s100 <- svm(mpglevel ~ ., tr,C=100)
ps100 <- predict(s100, ts)
cm100 <- table(ps100, ts$mpglevel) #confusion matrix
100*(1-sum(diag(cm100))/sum(cm100))

## [1] 4.347826
```

Cross-validation Errors and Discussion of the Results

All of my cross-validation errors are the same with the costs from .01 to 100 = 6.52137% error rate.

1.3

Now repeat (1.2), this time using SVMs with radial and polynomial basis kernels, with different values of gamma (c(0.01, 0.1, 1, 5, 10, 100)) and degree (c(2, 3, 4)) and cost (c(0.1, 1, 5, 10)). Use the cost and degree parameters values for polynomial kernels. The cost and gamma parameters values are given for radial basis kernels. Comment on your results, i.e., what are the parameters values (cost, degree, gamma) for the model that has the lowest cross-validation error?

R Code

```
#Low Cost, Gamma, and Degree
svm1 <- svm(mpglevel ~ ., tr,C=1, degree=1, gamma=1)
ps111 <- predict(svm1, ts)
cm111 <- table(ps111, ts$mpglevel) #confusion matrix
100*(1-sum(diag(cm111))/sum(cm111))

## [1] 3.26087

#High Cost, Gamma, Degree
s100 <- svm(mpglevel ~ ., tr,C=100, degree=3, gamma=10)
ps100 <- predict(s100, ts)
cm100 <- table(ps100, ts$mpglevel) #confusion matrix
100*(1-sum(diag(cm100))/sum(cm100))

## [1] 55.43478
```

Discussion of Results

I find that modifying the cost doesn't change the overall CV error rate, but that lower gamma and degree parameters has a significant impact on the CV error rate.

Problem 2

```
##install.packages("dplyr")
##library(dplyr)
View(Caravan)
```

2.1

Create a training set consisting of the first 1,000 observations, and a test set consisting of the remaining observations. The class variable is "Purchase" whose values are "No" and "Yes". Transform "No" to 0 "Yes" to 1. Place the R code below.

R Code

```
Caravan$Purchase <- as.character(ifelse(Caravan$Purchase=="Yes", 1, 0))
train <- Caravan[1:1000,]
test <- Caravan[1001:5822,]
## View(Caravan)
typeof(Caravan$Purchase)

## [1] "character"
```

I kept getting "Nan" values when evaluating my gbm model summary. I tried converting the `Purchase` variable to a factor, character, and integer. After looking through the text and lecture notes, I went to stack overflow, but the suggestions there didn't help either.

2.2

Fit a boosting model to the training set with Purchase as the response and the other variables as predictors. Use 1,000 trees, and a shrinkage value of 0.01. Which predictors appear to be the most important?

R Code

```
##install.packages("gbm")
library(gbm)

## Loading required package: survival

## Loading required package: lattice

## Loading required package: splines

## Loading required package: parallel

## Loaded gbm 2.1.3

model <- gbm(Purchase ~ ., data=test,n.trees = 1000,shrinkage = .01)

## Distribution not specified, assuming bernoulli ...

summary(model, plotit = FALSE)

##               var     rel.inf
## PPERSAUT PPERSAUT 25.07662675
## PPLEZIER PPLEZIER 14.08425902
## PBRAND     PBRAND 11.16089257
## MOPLLAAG MOPLLAAG  5.21533664
## MINKGEM   MINKGEM  4.69512214
## ALEVEN     ALEVEN  4.18820749
## APERSAUT APERSAUT  3.25526666
## PBYSTAND PBYSTAND  3.07208210
## MBERMIDD MBERMIDD  2.30144136
## MOSTYPE   MOSTYPE  2.20101843
## MBERHOOG MBERHOOG  1.81425545
## MBERARBG MBERARBG  1.68668522
## MAUT1       MAUT1  1.58667411
## MKOOPKLA MKOOPKLA  1.52103448
## PWAPART   PWAPART  1.45513613
## MGODOV     MGODOV  1.29436395
## MINK7512 MINK7512  1.29145049
## AFIETS     AFIETS  1.26061847
## MINKM30   MINKM30  1.05005438
## PGEZONG   PGEZONG  1.03874797
## MSKC         MSKC  1.00708097
## PFIETS     PFIETS  0.98458131
## MOSHOOFD MOSHOOFD  0.96505929
## MOPLMIDD MOPLMIDD  0.81288729
```

```
## MINK3045 MINK3045  0.70162814
## MOPLHOOG MOPLHOOG  0.62782018
## MGODGE     MGODGE  0.56303116
## MRELGE     MRELGE  0.53573276
## MHHUUR     MHHUUR  0.53067741
## MSKA         MSKA  0.50750504
## PLEVEN     PLEVEN  0.40247868
## MGODPR     MGODPR  0.39419935
## MINK4575 MINK4575  0.35472250
## MAUT0       MAUT0  0.23626821
## MBERBOER MBERBOER  0.21297093
## MINK123M MINK123M  0.18901388
## MFWEKIND MFWEKIND  0.18129730
## MSKB1       MSKB1  0.17605477
## MGODRK     MGODRK  0.17445922
## MZPART     MZPART  0.17139814
## MRELSA     MRELSA  0.14721546
## MHKOOP     MHKOOP  0.14248855
## MSKD         MSKD  0.13100341
## MZFONDS   MZFONDS  0.11782199
## PINBOED   PINBOED  0.11175351
## MBERZELF MBERZELF  0.10728261
## MFGEKIND MFGEKIND  0.09740308
## MGEMLEEF MGEMLEEF  0.07722697
## MAUT2       MAUT2  0.05428475
## PWALAND   PWALAND  0.03537936
## MAANTHUI MAANTHUI  0.00000000
## MGEMOMV   MGEMOMV  0.00000000
## MRELOV     MRELOV  0.00000000
## MFALLEEN MFALLEEN  0.00000000
## MBERARBO MBERARBO  0.00000000
## MSKB2       MSKB2  0.00000000
## PWABEDR   PWABEDR  0.00000000
## PBESAUT   PBESAUT  0.00000000
## PMOTSCO   PMOTSCO  0.00000000
## PVRAAUT   PVRAAUT  0.00000000
## PAANHANG PAANHANG  0.00000000
## PTRACTOR PTRACTOR  0.00000000
## PWERKT     PWERKT  0.00000000
## PBROM       PBROM  0.00000000
## PPERSONG PPERSONG  0.00000000
## PWAOREG   PWAOREG  0.00000000
## PZEILPL   PZEILPL  0.00000000
## AWAPART   AWAPART  0.00000000
## AWABEDR   AWABEDR  0.00000000
## AWALAND   AWALAND  0.00000000
## ABESAUT   ABESAUT  0.00000000
## AMOTSCO   AMOTSCO  0.00000000
## AVRAAUT   AVRAAUT  0.00000000
## AAANHANG AAANHANG  0.00000000
```

```
## ATRACTOR ATRACTOR  0.00000000
## AWERKT     AWERKT  0.00000000
## ABROM       ABROM  0.00000000
## APERSONG APERSONG  0.00000000
## AGEZONG   AGEZONG  0.00000000
## AWAOREG   AWAOREG  0.00000000
## ABRAND     ABRAND  0.00000000
## AZEILPL   AZEILPL  0.00000000
## APLEZIER APLEZIER  0.00000000
## AINBOED   AINBOED  0.00000000
## ABYSTAND ABYSTAND  0.00000000
```

Problem 3

```r
library(data.table)
library("curl")
mydata <- fread("https://archive.ics.uci.edu/ml/machine-learning-databases/io
nosphere/ionosphere.data")
mydata <- as.data.frame(mydata)
mydata <- mydata[,-2] #remove the second variable
```

3.1

Create a training data set containing a random sample of 300 data points and a test set containing the remaining observations. Name the training data and test data as mydata.training and mydata.testing, respectively. Place the R code below. You will use mydata.training and mydata.testing to answer rest of the questions. Thus, create them once and use mydata.training to train the models (classifiers) and mydata.testing to test the models. The last variable variable (35th variable in the data) is the response and the other variables are predictors.

R Code

```r
set.seed(1234)
rndSample <- sample(1:nrow(mydata), 300)
mydata.training <- mydata[rndSample,]
mydata.testing <- mydata[-rndSample,]
```

3.2

Train a naive bayes classifier using 10-fold cross-validation over mydata.training. Use this model to predict the observations in mydata.testing. Form a confusion matrix and report the error rate of the classifier over mydata.testing.

```r
##install.packages("lme4", dependencies = TRUE)
##library(lme4)
##methods(sigma)
##install.packages("pbkrtest", dependencies = TRUE)
##install.packages("DEoptimR")
##install.packages("caret", dependencies = TRUE)
```

```
## library(caret)
##library(e1071)
head(mydata)
```

```
##   V1      V3       V4       V5       V6       V7       V8      V9      V10
## 1  1 0.99539 -0.05889  0.85243  0.02306  0.83398 -0.37708 1.00000  0.03760
## 2  1 1.00000 -0.18829  0.93035 -0.36156 -0.10868 -0.93597 1.00000 -0.04549
## 3  1 1.00000 -0.03365  1.00000  0.00485  1.00000 -0.12062 0.88965  0.01198
## 4  1 1.00000 -0.45161  1.00000  1.00000  0.71216 -1.00000 0.00000  0.00000
## 5  1 1.00000 -0.02401  0.94140  0.06531  0.92106 -0.23255 0.77152 -0.16399
## 6  1 0.02337 -0.00592 -0.09924 -0.11949 -0.00763 -0.11824 0.14706  0.06637
##       V11      V12     V13      V14      V15      V16      V17      V18
## 1 0.85243 -0.17755 0.59755 -0.44945  0.60536 -0.38223  0.84356 -0.38542
## 2 0.50874 -0.67743 0.34432 -0.69707 -0.51685 -0.97515  0.05499 -0.62237
## 3 0.73082  0.05346 0.85443  0.00827  0.54591  0.00299  0.83775 -0.13644
## 4 0.00000  0.00000 0.00000  0.00000 -1.00000  0.14516  0.54094 -0.39330
## 5 0.52798 -0.20275 0.56409 -0.00712  0.34395 -0.27457  0.52940 -0.21780
## 6 0.03786 -0.06302 0.00000  0.00000 -0.04572 -0.15540 -0.00343 -0.10196
##        V19      V20      V21      V22      V23      V24      V25      V26
## 1  0.58212 -0.32192  0.56971 -0.29674  0.36946 -0.47357  0.56811 -0.51171
## 2  0.33109 -1.00000 -0.13151 -0.45300 -0.18056 -0.35734 -0.20332 -0.26569
## 3  0.75535 -0.08540  0.70887 -0.27502  0.43385 -0.12062  0.57528 -0.40220
## 4 -1.00000 -0.54467 -0.69975  1.00000  0.00000  0.00000  1.00000  0.90695
## 5  0.45107 -0.17813  0.05982 -0.35575  0.02309 -0.52879  0.03286 -0.65158
## 6 -0.11575 -0.05414  0.01838  0.03669  0.01519  0.00888  0.03513 -0.01535
##        V27      V28      V29      V30      V31      V32      V33      V34
## 1  0.41078 -0.46168  0.21266 -0.34090  0.42267 -0.54487  0.18641 -0.45300
## 2 -0.20468 -0.18401 -0.19040 -0.11593 -0.16626 -0.06288 -0.13738 -0.02447
## 3  0.58984 -0.22145  0.43100 -0.17365  0.60436 -0.24180  0.56045 -0.38238
## 4  0.51613  1.00000  1.00000 -0.20099  0.25682  1.00000 -0.32382  1.00000
## 5  0.13290 -0.53206  0.02431 -0.62197 -0.05707 -0.59573 -0.04608 -0.65697
## 6 -0.03240  0.09223 -0.07859  0.00732  0.00000  0.00000 -0.00039  0.12011
##   V35
## 1   g
## 2   b
## 3   g
## 4   b
## 5   g
## 6   b
```

```
##model = train(mydata.training,'nb',trControl=trainControl(method='cv',numbe
r=10))
##model <- NaiveBayes(mydata.training$ ~ ., data = tr)
##predict(model,mydata.testing)
##table(predict(model,mydata.testing)
##plot(model)
```

Unfortunately, I could not load the library caret. I tried several fixes, including uninstalling and reinstalling the source files, updating R (current version is 3.3.3), installing several other packages ahead of the caret, and it continually gives a namespace error and will not install. Therefore I can’t use the train() function.

I am going to re-attempt fixes tomorrow.