M6 Individual Coding Assignment

Leonard Genders

2025-02-16

### Part A. 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.

# read in the data  
Auto <- read.csv("Auto.csv", stringsAsFactors = T)  
  
# view  
View(Auto)  
  
# binary var = 1 for cars above median gas mileage, 0 below  
Auto$mpg\_binary <- as.numeric(Auto$mpg)  
  
# using ifelse conditional  
Auto$mpg\_binary <- ifelse(Auto$mpg > median(Auto$mpg), 1, 0) # if > median, 1, else 0 (< median)  
  
# check  
median(Auto$mpg) # 23

## [1] 23

View(Auto) # only those with mpg > 23 will be classified as 1

### Part B. Use the e1071 library in R. Use the SVM() function and kernel = “linear” and a cost argument to fit a support vector classifier to the data with three different values of cost at cost = .01, 0.1, 5 individually and respectively, in order to predict whether a car gets high or low gas mileage.

library(e1071)

## Warning: package 'e1071' was built under R version 4.3.3

# Use the SVM() function and kernel = "linear" and a cost argument to fit a   
# SVC to the data with three different values of cost at cost = 01, 0.1, 5 individually  
# and respectively, IOT predict whether a car gets high or low gas mileage  
  
# Creating a training set containing a random sample of 80% (397\*.8 = 317.6 = 318)   
# of the obs, using set.seed(1) to replicate easy  
set.seed(1)  
train <- sample(1:nrow(Auto), 318)  
  
# Create a test set containing the remaining obs  
Auto.test <- Auto[-train, ]  
MPG.test <- Auto$mpg\_binary[-train]  
  
# cost = .01 model  
svmfit.01 <- svm(mpg\_binary ~ ., data = Auto, kernel = "linear", cost = 0.01, scale = FALSE)  
summary(svmfit.01)

##   
## Call:  
## svm(formula = mpg\_binary ~ ., data = Auto, kernel = "linear", cost = 0.01,   
## scale = FALSE)  
##   
##   
## Parameters:  
## SVM-Type: eps-regression   
## SVM-Kernel: linear   
## cost: 0.01   
## gamma: 0.002475248   
## epsilon: 0.1   
##   
##   
## Number of Support Vectors: 264

# predict  
mpg.pred.01 <- predict(svmfit.01, Auto.test, type = "response")  
# not coming out binary without adjustment  
mpg.pred.01 <- ifelse(mpg.pred.01 > 0.5, 1, 0) # 1 = High mpg, 0 = low mpg  
  
# prediction results  
table(predict = mpg.pred.01, truth = MPG.test)

## truth  
## predict 0 1  
## 0 39 1  
## 1 6 33

(39+33)/79# 0.9113924

## [1] 0.9113924

(6+1)/79 # 0.08860759

## [1] 0.08860759

**Interpretation:** Our model with a cost argument set to a value of 0.01 indicates there are 264 support vectors in the model. Our prediction results indicate that there are 72 correct mpg classifications and 7 incorrect classifications. Our error for this model is 0.08860759 or ~9%. We correctly predicted 91% of the test observations. This is the best performing model in the cost argument vectors.

# cost = 0.1 model  
svmfit.1 <- svm(mpg\_binary ~ ., data = Auto, kernel = "linear", cost = 0.1, scale = FALSE)  
summary(svmfit.1)

##   
## Call:  
## svm(formula = mpg\_binary ~ ., data = Auto, kernel = "linear", cost = 0.1,   
## scale = FALSE)  
##   
##   
## Parameters:  
## SVM-Type: eps-regression   
## SVM-Kernel: linear   
## cost: 0.1   
## gamma: 0.002475248   
## epsilon: 0.1   
##   
##   
## Number of Support Vectors: 352

mpg.pred.1 <- predict(svmfit.1, Auto.test, type = "response")  
# not coming out binary without adjustment  
mpg.pred.1 <- ifelse(mpg.pred.1 > 0.5, 1, 0) # 1 = High mpg, 0 = low mpg  
  
# prediction results  
table(predict = mpg.pred.1, truth = MPG.test)

## truth  
## predict 0 1  
## 0 33 4  
## 1 12 30

(33+30)/79 # 0.7974684

## [1] 0.7974684

(12+4)/79 # 0.2025316

## [1] 0.2025316

**Interpretation:** Our model with a cost argument set to a value of 0.1 indicates there are 352 support vectors in the model. Our prediction results indicate that there are 63 correct mpg classifications and 16 incorrect classifications. Our error for this model is 0.2025316 or ~20%. We correctly predicted ~80% of the test observations. This is worse than a cost argument set to 0.01 and better than a cost argument set to 5.

# cost = 5 model  
svmfit5 <- svm(mpg\_binary ~ ., data = Auto, kernel = "linear", cost = 5, scale = FALSE)  
summary(svmfit5)

##   
## Call:  
## svm(formula = mpg\_binary ~ ., data = Auto, kernel = "linear", cost = 5,   
## scale = FALSE)  
##   
##   
## Parameters:  
## SVM-Type: eps-regression   
## SVM-Kernel: linear   
## cost: 5   
## gamma: 0.002475248   
## epsilon: 0.1   
##   
##   
## Number of Support Vectors: 387

mpg.pred.5 <- predict(svmfit5, Auto.test, type = "response")  
# not coming out binary without adjustment  
mpg.pred.5 <- ifelse(mpg.pred.5 > 0.5, 1, 0) # 1 = High mpg, 0 = low mpg  
  
# prediction results  
table(predict = mpg.pred.5, truth = MPG.test)

## truth  
## predict 0 1  
## 0 3 8  
## 1 42 26

(3+26)/79 # 0.3670886

## [1] 0.3670886

(42+8)/79 # 0.6329114

## [1] 0.6329114

**Interpretation:** Our model with a cost argument set to a value of 5 indicates there are 387 support vectors in the model. Our prediction results indicate that there are 29 correct mpg classifications and 50 incorrect classifications. Our error for this model is 0.6329114 or ~63%. We correctly predicted ~37% of the test observations. This is the worst performing model in the cost argument vectors.

### Part C.Use the tune() function and kernel = “linear” to report the cross-validation errors associated with different values of cost, use cost=c(0.01, 0.1, 5).

# use the tune() function and kernel = "linear" to report the cross-validation  
# errors associated with the different cost values cost=c(0.01, 0.1, 5)  
set.seed(1)  
tune.out <- tune(svm, mpg\_binary ~., data = Auto, kernel = "linear",  
 range = list(cost = c(0.01, 0.1, 5)))  
summary(tune.out)

##   
## Parameter tuning of 'svm':  
##   
## - sampling method: 10-fold cross validation   
##   
## - best parameters:  
## cost  
## 0.1  
##   
## - best performance: 0.07700521   
##   
## - Detailed performance results:  
## cost error dispersion  
## 1 0.01 0.08245518 0.02331372  
## 2 0.10 0.07700521 0.02606576  
## 3 5.00 0.09593061 0.01748426

**Interpretation:** Comment on your results: From the results of tune.out we can see that the lowest cross-validation error rate is with a cost value of 0.10 and an associated error of 0.07700521. The next lowest cross-validation error rate is with a cost value of 0.01 and this had an associated error of 0.08245518. The highest cross-validation error rate was with a cost value of 5 and had an error of 0.09593061.

### Part D.Now repeat (c) with tune(). This time using SVMs with radial kernels and polynomial kernels. For radial kernels, use gamma=c(2, 5) and cost=c(0.01, 0.1, 5), For polynomial kernels, use degree=c(2, 3) and cost=c(0.01, 0.1, 5). Comment on your results.

# Repeat (c) with tune() use SVM with radial kernels, gamma=c(2,5), cost=c(.01, .1, 5)  
set.seed(1)  
tune.out.radial <- tune(svm, mpg\_binary ~., data = Auto, kernel = "radial",  
 ranges = list(  
 gamma = c(2,5),   
 cost = c(0.01, 0.1, 5)))  
summary(tune.out.radial)

##   
## Parameter tuning of 'svm':  
##   
## - sampling method: 10-fold cross validation   
##   
## - best parameters:  
## gamma cost  
## 2 5  
##   
## - best performance: 0.2351545   
##   
## - Detailed performance results:  
## gamma cost error dispersion  
## 1 2 0.01 0.4309265 0.064056008  
## 2 5 0.01 0.4311362 0.064005041  
## 3 2 0.10 0.3940180 0.058994418  
## 4 5 0.10 0.3961374 0.058877118  
## 5 2 5.00 0.2351545 0.007473750  
## 6 5 5.00 0.2437632 0.006547979

**Interpretation:** Comment on your results: The tune() function sweeps over all variations of the gamma and cost values, providing results for each combination of gamma and cost to predict whether a car’s mpg will be high or low. The lowest error of the SVM with radial kernels called tune.out.radial is for a gamma value set to 2 and a cost value set to 5. This combination results in an error of 0.2359513.

# Repeat (c) with tune() use SVM with polynomial kernels, degree=c(2, 3), cost=c(0.01, 0.1, 5)  
tune.out.poly <- tune(svm, mpg\_binary ~., data = Auto, kernel = "polynomial",  
 ranges = list(  
 degree=c(2,3),   
 cost=c(0.01, 0.1, 5)))  
summary(tune.out.poly)

##   
## Parameter tuning of 'svm':  
##   
## - sampling method: 10-fold cross validation   
##   
## - best parameters:  
## degree cost  
## 2 5  
##   
## - best performance: 0.4106599   
##   
## - Detailed performance results:  
## degree cost error dispersion  
## 1 2 0.01 0.4355240 0.04148753  
## 2 3 0.01 0.4355544 0.04147568  
## 3 2 0.10 0.4351084 0.04159793  
## 4 3 0.10 0.4354120 0.04147965  
## 5 2 5.00 0.4106599 0.04725919  
## 6 3 5.00 0.4276271 0.04175236

**Interpretation:** Comment on your results: Moving to the polynomial kernels model, named tune.out.poly we can see that the tune() function again sweeps over all combinations of the degree argument and the cost argument values. We learn that the lowest error here is with a degree value of 2 and a cost value of 5. This results in an error rate of 0.4100906.