**Assignment #7: SVM**

**Submit through link: eCampus -> Assignments->Assignment 7 Submission**

**Deadline: December 9 (Sunday) @17:00 pm**

**The filename should have this format: LastName-FirstName-hw07.doc**

**Problem 1**

In this problem, you will use support vector approaches to predict whether a given car gets high or low gas mileage based on the Auto data set in the ISLR package.

(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. Use this variable as response in the following analysis.

Answer: mydata<-Auto

median(Auto$mpg)

= 22.5

mydata$high\_mileage<- rep(0,392)

mydata$high\_mileage[mydata$mpg>= 22.5] <- 1

mydata$high\_mileage <- as.factor(mydata$high\_mileage)

mydata$mpg <- NULL

(b) Fit a support vector classifier to the data with various values of cost, 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.

Answer: library(e1071)

set.seed(1)

tune.out<- tune(svm, high\_mileage~., data= mydata, Kernel= "linear", ranges= list(cost=c(0.001,0.01,0.1,1,5,10,100)) )

summary(tune.out)

- Detailed performance results:

cost error dispersion

1 1e-03 0.52852564 0.10893717

2 1e-02 0.52852564 0.10893717

3 1e-01 0.17115385 0.07281530

4 1e+00 0.10455128 0.05178920

5 5e+00 0.08929487 0.06192162

6 1e+01 0.09692308 0.06483119

7 1e+02 0.08935897 0.05307427

Best model is the model with cost = 5. It has the smallest error rate.

best\_mod= tune.out$best.model

(c) Now repeat (b), this time using SVMs with radial and polynomial kernels, with different values of gamma, degree and cost. Comment on your results.

Answer:

#polyomial kernel

set.seed(1)

tune.out<- tune(svm, high\_mileage~., data= mydata, Kernel= "polynomial", ranges= list(cost=c(0.001,0.01,0.1,1,5,10,100), d= c(1,2,3,4)) )

summary(tune.out)

- best parameters:

cost d

5 1

- best performance: 0.08929487

- Detailed performance results:

cost d error dispersion

1 1e-03 1 0.52852564 0.10893717

2 1e-02 1 0.52852564 0.10893717

3 1e-01 1 0.17115385 0.07281530

4 1e+00 1 0.10455128 0.05178920

5 5e+00 1 0.08929487 0.06192162

6 1e+01 1 0.09692308 0.06483119

7 1e+02 1 0.08935897 0.05307427

8 1e-03 2 0.52852564 0.10893717

9 1e-02 2 0.52852564 0.10893717

10 1e-01 2 0.17115385 0.07281530

11 1e+00 2 0.10455128 0.05178920

12 5e+00 2 0.08929487 0.06192162

13 1e+01 2 0.09692308 0.06483119

14 1e+02 2 0.08935897 0.05307427

15 1e-03 3 0.52852564 0.10893717

16 1e-02 3 0.52852564 0.10893717

17 1e-01 3 0.17115385 0.07281530

18 1e+00 3 0.10455128 0.05178920

19 5e+00 3 0.08929487 0.06192162

20 1e+01 3 0.09692308 0.06483119

21 1e+02 3 0.08935897 0.05307427

22 1e-03 4 0.52852564 0.10893717

23 1e-02 4 0.52852564 0.10893717

24 1e-01 4 0.17115385 0.07281530

25 1e+00 4 0.10455128 0.05178920

26 5e+00 4 0.08929487 0.06192162

27 1e+01 4 0.09692308 0.06483119

28 1e+02 4 0.08935897 0.05307427

So the best model with polynomial kernel is with cost 5 and d=1. Which is again equivalent to linear kernel

or we can say a simple support vector classifier.

#radial kernel

set.seed(1)

tune.out<- tune(svm, high\_mileage~., data= mydata, Kernel= "radial", ranges= list(cost=c(0.001,0.01,0.1,1,5,10,100), gamma= c(0.0001,0.001,0.01,0.1, 1,10)) )

summary(tune.out)

- best parameters:

cost gamma

5 0.1

- best performance: 0.07891026

- Detailed performance results:

cost gamma error dispersion

1 1e-03 1e-04 0.58423077 0.03735370

2 1e-02 1e-04 0.58423077 0.03735370

3 1e-01 1e-04 0.58423077 0.03735370

4 1e+00 1e-04 0.58423077 0.03735370

5 5e+00 1e-04 0.13262821 0.04898567

6 1e+01 1e-04 0.11724359 0.06549795

7 1e+02 1e-04 0.09173077 0.04819260

8 1e-03 1e-03 0.58423077 0.03735370

9 1e-02 1e-03 0.58423077 0.03735370

10 1e-01 1e-03 0.58423077 0.03735370

11 1e+00 1e-03 0.11724359 0.06549795

12 5e+00 1e-03 0.08916667 0.04972572

13 1e+01 1e-03 0.09173077 0.04819260

14 1e+02 1e-03 0.09435897 0.04804681

15 1e-03 1e-02 0.58423077 0.03735370

16 1e-02 1e-02 0.58423077 0.03735370

17 1e-01 1e-02 0.11980769 0.06536118

18 1e+00 1e-02 0.09429487 0.04949748

19 5e+00 1e-02 0.09435897 0.04804681

20 1e+01 1e-02 0.08673077 0.05111087

21 1e+02 1e-02 0.09160256 0.05718713

22 1e-03 1e-01 0.58423077 0.03735370

23 1e-02 1e-01 0.23698718 0.08926820

24 1e-01 1e-01 0.08923077 0.04828946

25 1e+00 1e-01 0.08929487 0.04382379

26 5e+00 1e-01 0.07891026 0.04193095

27 1e+01 1e-01 0.08153846 0.04555423

28 1e+02 1e-01 0.11980769 0.03771165

29 1e-03 1e+00 0.58423077 0.03735370

30 1e-02 1e+00 0.58423077 0.03735370

31 1e-01 1e+00 0.58423077 0.03735370

32 1e+00 1e+00 0.08660256 0.04627087

33 5e+00 1e+00 0.08653846 0.04460032

34 1e+01 1e+00 0.08653846 0.04460032

So the least training error model of all the three is SVM with radial kernel with gamma =0.1 and cost= 5

Radial kernel > Linear > polynomial

But again these are on training set, their order can change if checked on test set

**Problem 2**

This problem uses the OJ data set in the ISLR package.

(a) Create a training set containing a random sample of 800 observations, and a test set containing the remaining observations.

Answer:

mydata <- OJ

subset<-sample(1:1070,size = 800,replace = FALSE)

mydata.train <-mydata[subset,]

mydata.test <- mydata[-subset,]

(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.

Answer:

svm.fit <- svm(Purchase~. , data= mydata.train, kernel="linear", cost=0.01)

summary(svm.fit)

the number of support vectors for this model is 447 which seem to be very high

this suggest that the cost=0.01 is too low if we would increase it, the number of support vectors may reduce.

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

Answer:

pred.train<-predict(svm.fit,newdata = mydata.train)

training\_error = mean(pred.train != mydata.train$Purchase)

=0.17

pred.test<- predict(svm.fit, newdata = mydata.test)

test-error = mean(pred.test != mydata.test$Purchase)

=0.162963

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

Answer:

set.seed(1)

tune.out <- tune(svm, Purchase~. , data= mydata, kernel="linear", ranges= list(cost=c(0.01,0.1,1,10)))

summary(tune.out)

|  |
| --- |
| - best parameters:  cost  10  - best performance: 0.1672897  - Detailed performance results:  cost error dispersion  1 0.01 0.1719626 0.02929012  2 0.10 0.1719626 0.02380682  3 1.00 0.1757009 0.02561346  4 10.00 0.1672897 0.02767155 |
|  |
| |  | | --- | | Best cost is 10 as it gives lowest cross validation error. | |

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

Answer:

svm.fit <- svm(Purchase~. , data= mydata.train, kernel="linear", cost=10)

summary(svm.fit)

pred.train<-predict(svm.fit,newdata = mydata.train)

mean(pred.train != mydata.train$Purchase)

=0.1625

pred.test<- predict(svm.fit, newdata = mydata.test)

mean(pred.test != mydata.test$Purchase)

=0.1555556

(f) Repeat parts (b) through (e) using a support vector machine with a radial kernel. Use the tune() function to select an optimal cost and gamma.

Answer:

svm.fit <- svm(Purchase~. , data= mydata.train, kernel="radial", cost=0.01, gamma= 0.01)

summary(svm.fit)

Number of Support Vectors: 635

pred.train<-predict(svm.fit,newdata = mydata.train)

mean(pred.train != mydata.train$Purchase)

training error= 0.39625

pred.test<- predict(svm.fit, newdata = mydata.test)

mean(pred.test != mydata.test$Purchase)

test error = 0.3703

set.seed(1)

tune.out <- tune(svm, Purchase~. , data= mydata, kernel="radial", ranges= list(cost=c(0.01,0.1,1,10), gamma= c(0.001,0.01,0.1,1,10)))

summary(tune.out)

- best parameters:

cost gamma

1 0.01

- best performance: 0.1691589

- Detailed performance results:

cost gamma error dispersion

1 0.01 1e-03 0.3897196 0.03867197

2 0.10 1e-03 0.3897196 0.03867197

3 1.00 1e-03 0.1878505 0.03220949

4 10.00 1e-03 0.1738318 0.02650722

5 0.01 1e-02 0.3897196 0.03867197

6 0.10 1e-02 0.1841121 0.03178486

7 1.00 1e-02 0.1691589 0.02659860

8 10.00 1e-02 0.1813084 0.02613854

9 0.01 1e-01 0.3897196 0.03867197

10 0.10 1e-01 0.1831776 0.03243468

11 1.00 1e-01 0.1794393 0.03378303

12 10.00 1e-01 0.1925234 0.02895688

13 0.01 1e+00 0.3897196 0.03867197

14 0.10 1e+00 0.2981308 0.04954153

15 1.00 1e+00 0.2168224 0.02322910

16 10.00 1e+00 0.2233645 0.02802007

17 0.01 1e+01 0.3897196 0.03867197

18 0.10 1e+01 0.3841121 0.03697860

19 1.00 1e+01 0.2401869 0.04340180

svm.fit <- svm(Purchase~. , data= mydata.train, kernel="radial", cost=1, gamma= 0.01)

summary(svm.fit)

pred.train<-predict(svm.fit,newdata = mydata.train)

mean(pred.train != mydata.train$Purchase)

= 0.16625

pred.test<- predict(svm.fit, newdata = mydata.test)

mean(pred.test != mydata.test$Purchase)

= 0.1444

(g) Repeat parts (b) through (e) using a support vector machine with a polynomial kernel. Set degree=2. Use the tune() function to select an optimal cost.

Answer:

svm.fit <- svm(Purchase~. , data= mydata.train, kernel="polynomial", cost=1, d= 2)

summary(svm.fit)

again the number of support vectors is too high i.e 463

pred.train<-predict(svm.fit,newdata = mydata.train)

mean(pred.train != mydata.train$Purchase)

training error= 0.18

pred.test<- predict(svm.fit, newdata = mydata.test)

mean(pred.test != mydata.test$Purchase)

test error = 0.1703

set.seed(1)

tune.out <- tune(svm, Purchase~. , data= mydata, kernel="polynomial", ranges= list(cost=c(0.01,0.1,1,10), d= c(1,2,3,4)))

summary(tune.out)

- best parameters:

cost d

0.1 1

- best performance: 0.1719626

- Detailed performance results:

cost d error dispersion

1 0.01 1 0.3738318 0.04061808

2 0.10 1 0.1719626 0.02460862

3 1.00 1 0.1738318 0.02613854

4 10.00 1 0.1757009 0.02708669

5 0.01 2 0.3719626 0.03784758

6 0.10 2 0.3000000 0.04609118

7 1.00 2 0.1953271 0.03339296

8 10.00 2 0.1757009 0.03170844

9 0.01 3 0.3691589 0.04019778

10 0.10 3 0.2644860 0.03553314

11 1.00 3 0.1869159 0.02492212

12 10.00 3 0.1859813 0.02509674

13 0.01 4 0.3691589 0.04019778

14 0.10 4 0.3102804 0.03733122

15 1.00 4 0.2149533 0.03469012

16 10.00 4 0.1962617 0.04225751

svm.fit <- svm(Purchase~. , data= mydata.train, kernel="polynomial", cost=0.1, d= 1)

summary(svm.fit)

pred.train<-predict(svm.fit,newdata = mydata.train)

mean(pred.train != mydata.train$Purchase)

training error= 0.17

pred.test<- predict(svm.fit, newdata = mydata.test)

mean(pred.test != mydata.test$Purchase)

test.error = 0.162

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

Answer:

Best model on the basis of lowest test error(0.1444) is the SVM with radial kernel having cost= 2 and gamma= 0.01.