Craig Van Vliet & Ryan Thomas

ECO 7100 – Econometrics 1

April 15th, 2019

Lab: Support Vector Machines

Assignment Submission

1) This problem involves the OJ data set which is part of the ISLR package.

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

```
library(ISLR)
set.seed(9004)
summary(OJ)

train_ind=sample(seq_len(nrow(OJ)),800)
train=OJ[train_ind,]
test=OJ[-train_ind,]
```

(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.

```
library(e1071)
svmfit.linear=svm(Purchase~.,data=train,kernel="linear",cost=0.01)
summary(svmfit.linear)
```

```
> summary(svmfit.linear)

Call:
svm(formula = Purchase ~ ., data = train, kernel = "linear", cost = 0.01)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  0.01
      gamma:  0.05555556

Number of Support Vectors:  432

 ( 217 215 )


Number of Classes:  2

Levels:
 CH MM
```

The linear svm model uses 432 points as support vectors to determine the hyperplane for classification of the two levels (CH and MM) for the variable Purchase on the training data set.

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

```
table(true=train$Purchase, pred=predict(svmfit.linear))
table(true=test$Purchase, pred=predict(svmfit.linear,newdata=test))
```

```
> table(true=train$Purchase, pred=predict(svmfit.linear))
    pred
true  CH  MM
  CH 439  53
  MM  82 226
> table(true=test$Purchase, pred=predict(svmfit.linear,newdata=test))
    pred
true  CH  MM
  CH 142  19
  MM  29  80
```

Training error rate:

- CH: 53/492 = 10.77%
- MM: 82/308 = 26.62%

Test error rate:

- CH: 19/161 = 11.80%
- MM: 29/109 = 26.61%

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

```
tune.out.linear=tune(svm,
Purchase~.,data=train,kernel="linear",ranges=list(cost=c(0.01, 0.1,
seq(0.25,10, 0.25))))
summary(tune.out.linear)
tune.out.linear$best.model
```

```
> summary(tune.out.linear)

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost
 2.75

- best performance: 0.16

- Detailed performance results:
    cost   error dispersion
1   0.01 0.17375 0.04693746
2   0.10 0.16500 0.04241004
3   0.25 0.16500 0.04241004
4   0.50 0.16125 0.04226652
5   0.75 0.16250 0.04082483
6   1.00 0.16375 0.04059026
7   1.25 0.16375 0.04267529
8   1.50 0.16375 0.04267529
9   1.75 0.16375 0.04267529
10  2.00 0.16375 0.04267529
11  2.25 0.16500 0.04241004
12  2.50 0.16375 0.04267529
13  2.75 0.16000 0.04322101
```

Optimal cost is 2.75 with an error rate of 16%

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

```
svmfit.best.linear=svm(Purchase~.,data=train,kernel="linear",cost=2.75)
table(true=train$Purchase, pred=predict(svmfit.best.linear))
table(true=test$Purchase, pred=predict(svmfit.best.linear,newdata=test))
```

```
> table(true=train$Purchase, pred=predict(svmfit.best.linear))
    pred
true  CH  MM
  CH 436  56
  MM  72 236
> table(true=test$Purchase, pred=predict(svmfit.best.linear,newdata=test))
    pred
true  CH  MM
  CH 141  20
  MM  29  80
```

Training error rate:

- CH: 56/492 = 11.38%
- MM: 72/308 = 23.38%

Test error rate:

- CH: 20/161 = 12.42%
- MM: 29/109 = 26.61%

(f) Repeat parts (b) through (e) using a support vector machine with a radial kernel. Use the default value for gamma.

```
svmfit.radial=svm(Purchase~.,data=train,kernel="radial",cost=0.01)
summary(svmfit.radial)
```

```
> summary(svmfit.radial)

Call:
svm(formula = Purchase ~ ., data = train, kernel = "radial", cost = 0.01)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  0.01
      gamma:  0.05555556

Number of Support Vectors:  620

 ( 312 308 )


Number of Classes:  2

Levels:
 CH MM
```

The radial svm model uses 620 points as support vectors to determine the hyperplane for classification of the two levels (CH and MM) for the variable Purchase on the training data set.

```
table(true=train$Purchase, pred=predict(svmfit.radial))
table(true=test$Purchase, pred=predict(svmfit.radial,newdata=test))
```

```
> table(true=train$Purchase, pred=predict(svmfit.radial))
     pred
true  CH  MM
  CH 492   0
  MM 308   0
> table(true=test$Purchase, pred=predict(svmfit.radial,newdata=test))
     pred
true  CH  MM
  CH 161   0
  MM 109   0
```

Training error rate:

- CH: 0/492 = 0%
- MM: 308/308 = 100%

Test error rate:

- CH: 0/161 = 0%
- MM: 109/109 = 100%

```
tune.out.radial=tune(svm,
Purchase~.,data=train,kernel="radial",ranges=list(cost=c(0.01, 0.1,
seq(0.25,10, 0.25))))
summary(tune.out.radial)
tune.out.radial$best.model
```

```
> summary(tune.out.radial)

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost
  0.5

- best performance: 0.17

- Detailed performance results:
    cost   error dispersion
1   0.01 0.38500 0.05945353
2   0.10 0.18250 0.05779514
3   0.25 0.17750 0.05767485
4   0.50 0.17000 0.05898446
```

```
svmfit.best.radial=svm(Purchase~.,data=train,kernel="radial",cost=0.5)
table(true=train$Purchase, pred=predict(svmfit.best.radial))
table(true=test$Purchase, pred=predict(svmfit.best.radial,newdata=test))
```

```
> table(true=train$Purchase, pred=predict(svmfit.best.radial))
    pred
true  CH  MM
  CH 452  40
  MM  77 231
> table(true=test$Purchase, pred=predict(svmfit.best.radial,newdata=test))
    pred
true  CH  MM
  CH 147  14
  MM  28  81
```

Training error rate:

- CH: 40/492 = 8.13%
- MM: 77/308 = 25%

Test error rate:

- CH: 14/161 = 8.70%
- MM: 28/109 = 25.67%

(g) Repeat parts (b) through (e) using a support vector machine with a polynomial kernel. Set degree=2.

```
svmfit.poly=svm(Purchase~.,data=train,kernel="polynomial",degree=2,cost=0.01)
summary(svmfit.poly)
```

```
> summary(svmfit.poly)

Call:
svm(formula = Purchase ~ ., data = train, kernel = "polynomial", degree = 2, cost = 0.01)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  polynomial
       cost:  0.01
     degree:  2
      gamma:  0.05555556
     coef.0:  0

Number of Support Vectors:  624

 ( 316 308 )


Number of Classes:  2

Levels:
 CH MM
```

The polynomial svm model uses 624 points as support vectors to determine the hyperplane for classification of the two levels (CH and MM) for the variable Purchase on the training data set.

```
table(true=train$Purchase, pred=predict(svmfit.poly))
table(true=test$Purchase, pred=predict(svmfit.poly,newdata=test))
```

```
> table(true=train$Purchase, pred=predict(svmfit.poly))
    pred
true  CH  MM
  CH 492   0
  MM 307   1
> table(true=test$Purchase, pred=predict(svmfit.poly,newdata=test))
    pred
true  CH  MM
  CH 161   0
  MM 109   0
```

Training error rate:

- CH: 0/492 = 0%
- MM: 307/308 = 99.68%

Test error rate:

- CH: 0/161 = 0%
- MM: 109/109 = 100%

```
tune.out.poly=tune(svm,
Purchase~.,data=train,kernel="polynomial",degree=2,ranges=list(cost=c(0.01,
0.1, seq(0.25,10, 0.25))))
summary(tune.out.poly)
tune.out.poly$best.model
```

```
> summary(tune.out.poly)

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost
    5

- best performance: 0.1775

- Detailed performance results:
    cost   error dispersion
1   0.01 0.38500 0.05163978
2   0.10 0.31000 0.06118052
3   0.25 0.23375 0.04788949
4   0.50 0.20500 0.04794383
5   0.75 0.20750 0.05041494
6   1.00 0.20250 0.04556741
7   1.25 0.19875 0.04980866
8   1.50 0.19125 0.05529278
9   1.75 0.18750 0.05464532
10  2.00 0.19000 0.04779877
11  2.25 0.18625 0.04730589
12  2.50 0.18250 0.04794383
13  2.75 0.18375 0.04715886
14  3.00 0.18125 0.04832256
15  3.25 0.18500 0.05130248
16  3.50 0.18500 0.05130248
17  3.75 0.18375 0.04896498
18  4.00 0.18375 0.04752558
19  4.25 0.18250 0.05006940
20  4.50 0.18125 0.05179085
21  4.75 0.17875 0.05272110
22  5.00 0.17750 0.05096295
```

```
svmfit.best.poly=svm(Purchase~.,data=train,kernel="polynomial",degree=2,cost=
5)
table(true=train$Purchase, pred=predict(svmfit.best.poly))
table(true=test$Purchase, pred=predict(svmfit.best.poly,newdata=test))
```

```
> table(true=train$Purchase, pred=predict(svmfit.best.poly))
     pred
true  CH  MM
  CH 454  38
  MM  85 223
> table(true=test$Purchase, pred=predict(svmfit.best.poly,newdata=test))
     pred
true  CH  MM
  CH 148  13
  MM  34  75
```

Training error rate:

- CH: 38/492 = 7.72%
- MM: 85/308 = 27.60%

Test error rate:

- CH: 13/161 = 8.07%
- MM: 34/109 = 31.19%

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

Recap of results after tuning:

Linear

- Training error rate:
  - CH: 56/492 = 11.38%
  - MM: 72/308 = 23.38%
- Test error rate:
  - CH: 20/161 = 12.42%
  - MM: 29/109 = 26.61%

Polynomial

- Training error rate:
  - CH: 38/492 = 7.72%
  - MM: 85/308 = 27.60%
- Test error rate:
  - CH: 13/161 = 8.07%
  - MM: 34/109 = 31.19%

Radial

- Training error rate:
  - CH: 40/492 = 8.13%
  - MM: 77/308 = 25%
- Test error rate:
  - CH: 14/161 = 8.70%
  - MM: 28/109 = 25.67%

Radial svm performs best compared to other approaches. Compared to the linear model, the radial model performs significantly better classifying CH and performs comparably well with classifying MM. The radial approach performs slightly worse than polynomial when classifying CH, but it classifies MM significantly better than the polynomial approach.