STAT 8330 FALL 2015 ASSIGNMENT 7

Peng Shao

October 30, 2015

▶ **Exercises 1.** **Solution.**

The optimal cost is

```
tune.svm.lin$best.parameters$cost
```

```
## [1] 0.05
```

The training error rate is 16.12%, and the test error rate is 18.89%.

▶ **Exercises 2.** **Solution.**

The optimal cost is

```
tune.svm.poly$best.parameters$cost
```

```
## [1] 5
```

The training error rate is 14.88%, and the test error rate is 18.15%.

▶ **Exercises 3.** **Solution.**

The optimal cost and gamma are

```
unlist(tune.svm.rbf$best.parameters)[2:3]
```

```
##  cost gamma
## 10.00  0.05
```

The training error rate is 13.88%, and the test error rate is 18.52%.

▶ **Exercises 4.** **Solution.**

The optimal cost and gamma are

```
unlist(tune.svm.sigm$best.parameters)[2:3]
```

```
##  cost gamma
##  5.00  0.01
```

The training error rate is 16.5%, and the test error rate is 19.26%.

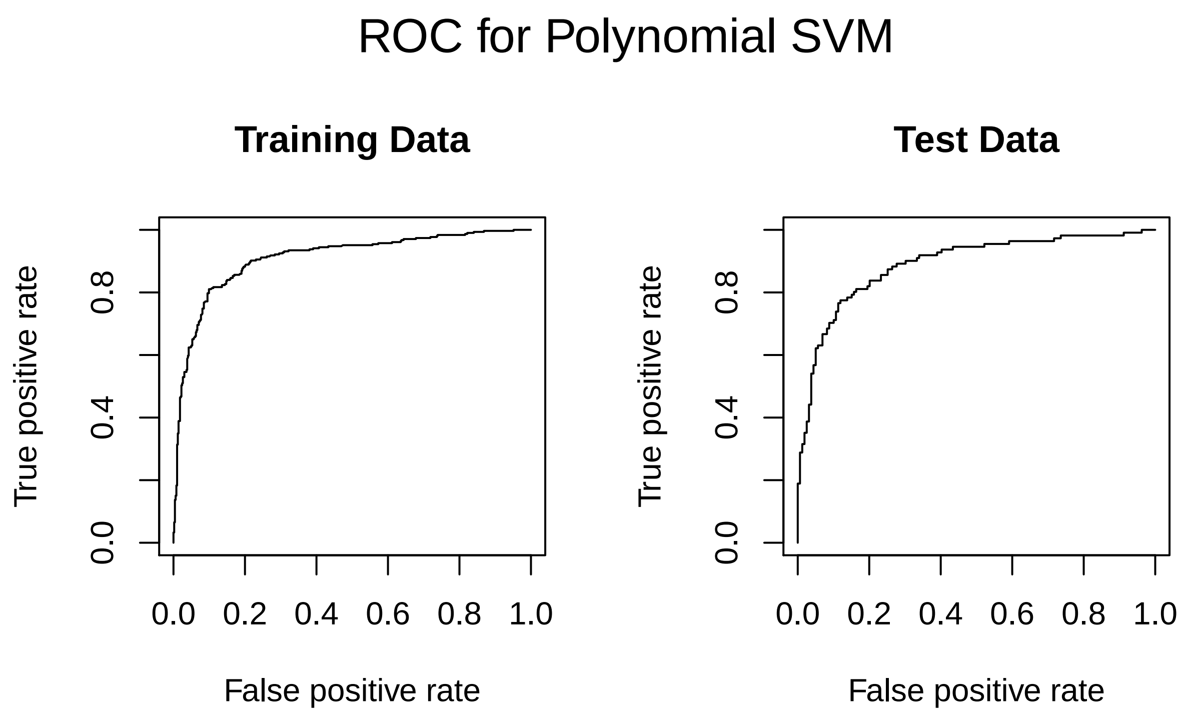
▶ **Exercises 5.** **Solution.**

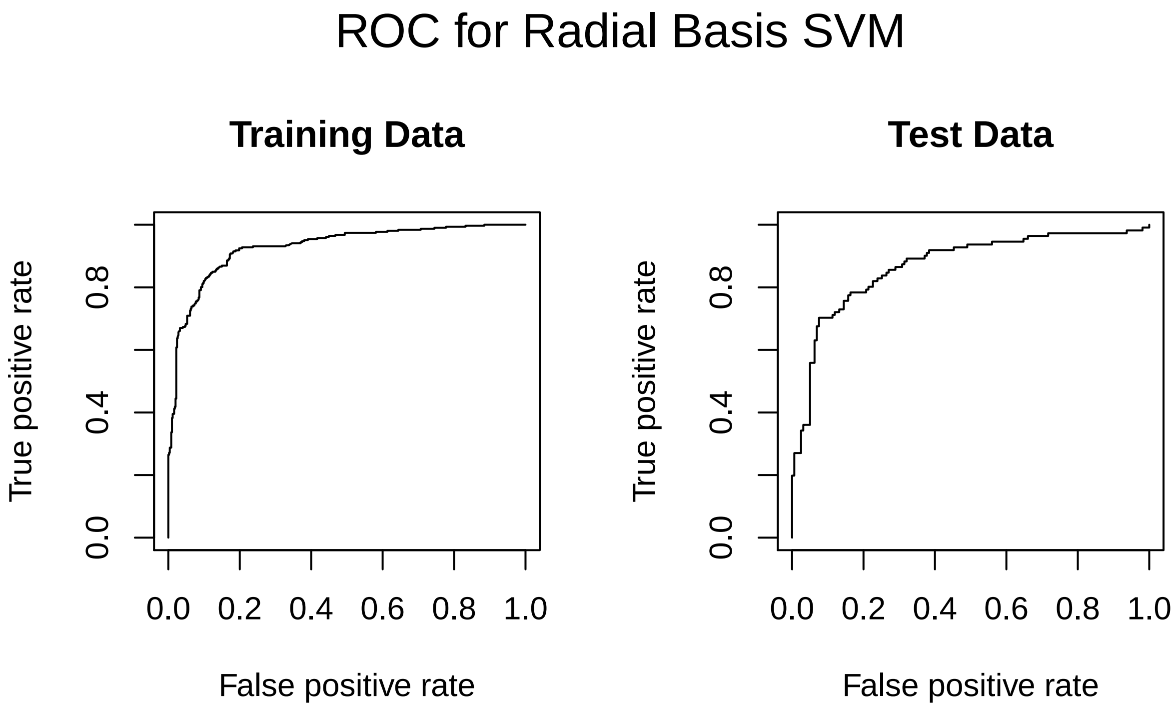
The training error rate is 15.87%, and the test error rate is 19.26%.

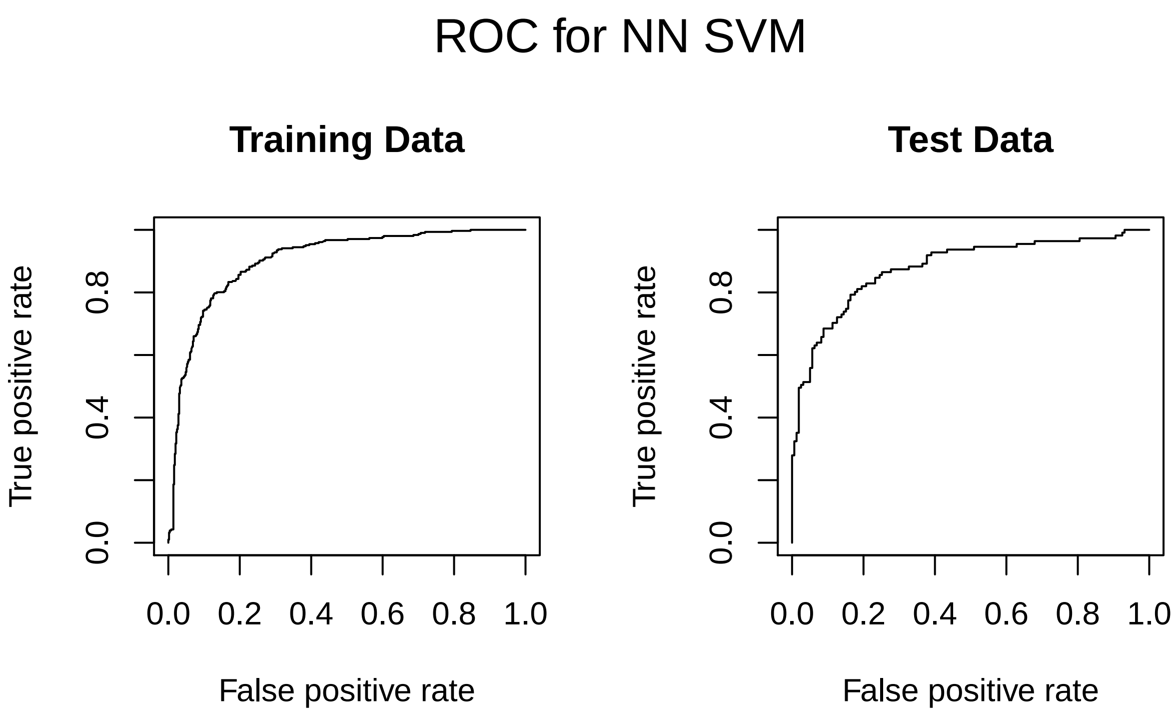
▶ **Exercises 6.** **Solution.**

```
## Loading required package: gplots
##
## Attaching package: 'gplots'
##
## The following object is masked from 'package:stats':
##
##     lowess
```


ROC for Linear SVM
Training Data
Test Data
True positive rate
False positive rate
0.0
0.4
0.8
0.0 0.2 0.4 0.6 0.8 1.0


ROC for Polynomial SVM
Training Data
Test Data
True positive rate
False positive rate
0.0
0.4
0.8
0.0 0.2 0.4 0.6 0.8 1.0


ROC for Radial Basis SVM
Training Data
Test Data
True positive rate
0.0
0.4
0.8
0.0 0.2 0.4 0.6 0.8 1.0
False positive rate


ROC for NN SVM
Training Data
Test Data
True positive rate
0.0
0.4
0.8
0.0 0.2 0.4 0.6 0.8 1.0
False positive rate

In terms of AUC,

```
aucs
```

```
##                     Training      Test
## Linear SVM          0.9056720 0.8819763
## Polynomial SVM      0.9094030 0.8897955
## Radial Basis SVM    0.9284353 0.8695110
## NN SVM              0.9032177 0.8794833
## Logistic Regression 0.9088209 0.8862258
```

Radial Basis SVM performs best on training dataset, and Polynomial SVM performs best on test dataset.

▶ **Exercises 7. Solution.**

Using `penalized` package, the penalty parameter search space is

```
penal.grid <- expand.grid(l1 = c(0.01, 0.05, 0.1, 0.5, 1, 5, 10),
                          l2 = c(0.01, 0.05, 0.1, 0.5, 1, 5, 10))
```

The optimal penalty paramters are

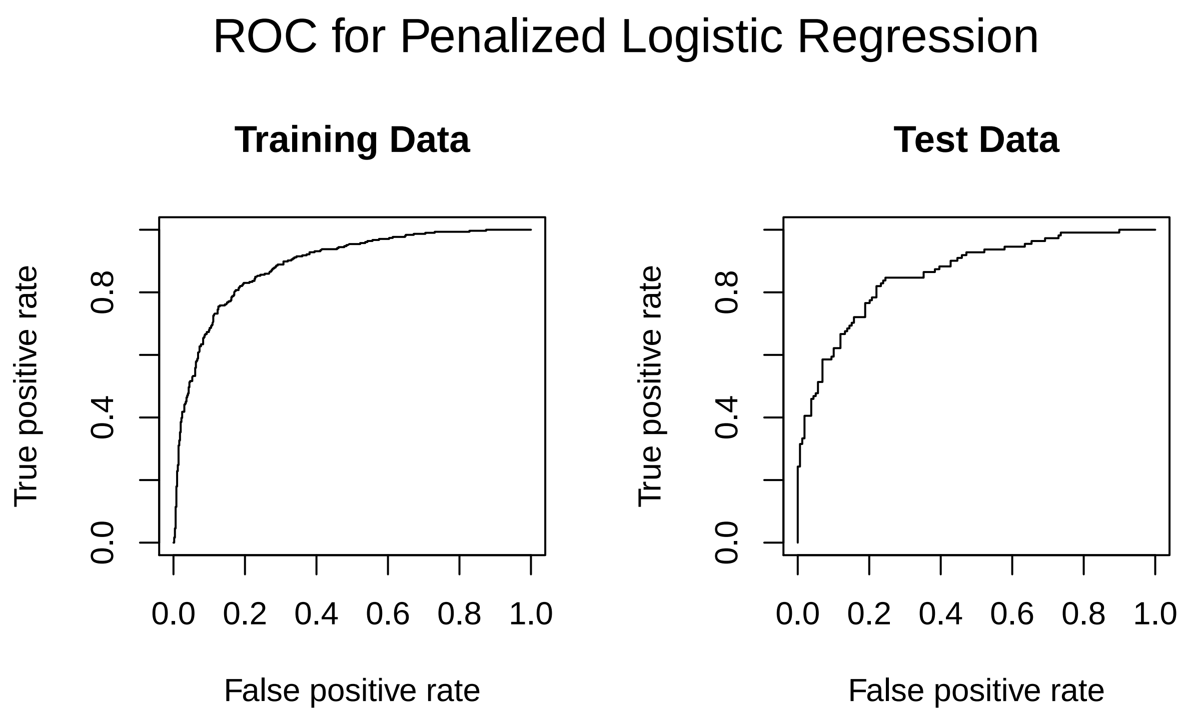
```
penlogis.fit@lambda1
```

```
## [1] 10
```

```
penlogis.fit@lambda2
```

```
## [1] 10
```

The training error rate is 18.37%, and the test error rate is 21.48%. The training AUC is 0.8889 and the test AUC is 0.862. So the penalized logistic regression is worse than any other methods in terms of AUC, but not that much. The ROC plot is



I also try this kind of regression using `glmnet` package, the penalty parameter search space is

```
alpha <- seq(0, 1, 0.01)
lambda <- seq(0, 1, 0.01)
```

The optimal penalty paramters are

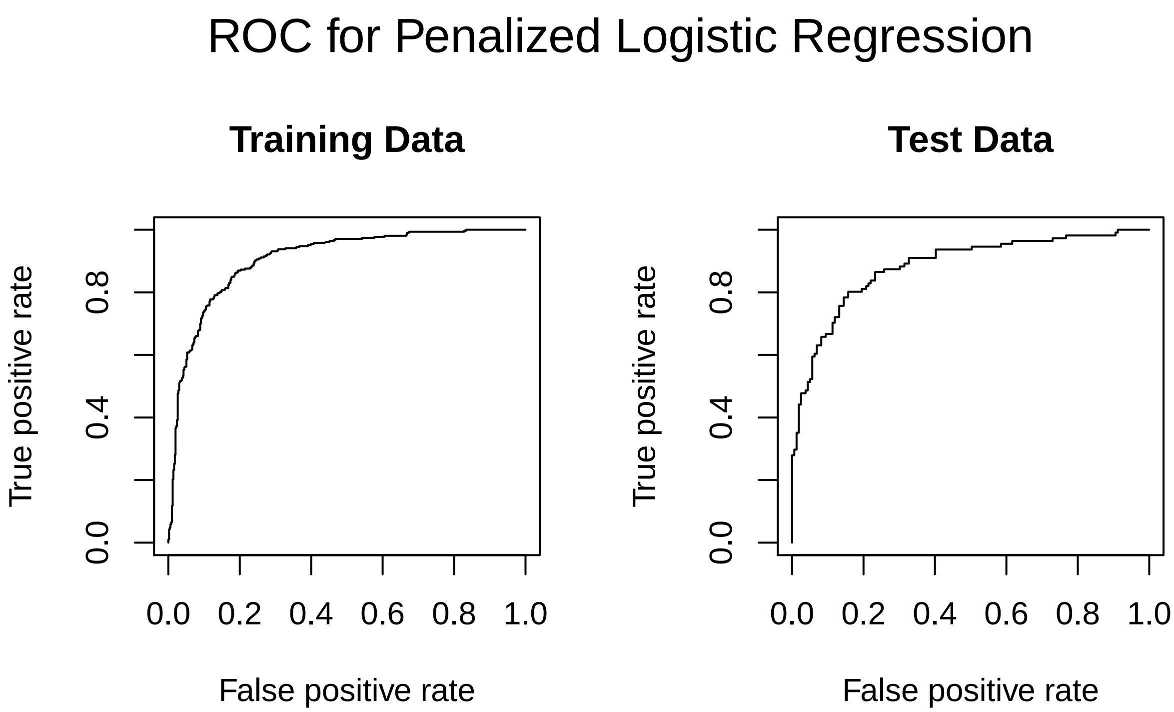
```
lambda
```

```
## [1] 0.03
```

```
a
```

```
## [1] 0.06
```

The training error rate is 18.37%, and the test error rate is 21.48%. The training AUC is 0.9063 and the test AUC is 0.8836. The ROC plot is



The main code is

```
train.input <- model.matrix(Purchase ~ ., data = train)
test.input <- model.matrix(Purchase ~ ., data = test)
penal.grid <- expand.grid(l1 = c(0.01, 0.05, 0.1, 0.5, 1, 5, 10),
                          l2 = c(0.01, 0.05, 0.1, 0.5, 1, 5, 10))
cver <- numeric(length(penal.grid))
for (i in 1:nrow(penal.grid)){
        cver[i] <- cvl(response = train$Purchase,
                       penalized = train.input,
                       lambda1 = penal.grid[i, 1],
                       lambda2 = penal.grid[i, 2],
                       model = "logistic",
                       trace = FALSE,
                       fold = 10)$cvl
}
ind <- which.min(cver)
penlogis.fit <- penalized(response = train$Purchase,
                          penalized = train.input,
                          model = "logistic",
                          lambda1 = penal.grid[ind, 1],
                          lambda2 = penal.grid[ind, 1])
penlogis.pred.tr <- predict(penlogis.fit,
                            penalized = train.input)
cm.penlogis.tr <- confusionMatrix(as.factor(ifelse(penlogis.pred.tr >= 0.5,
                                                   "MM",
                                                   "CH")),
                                  train$Purchase)
penlogis.pred.te <- predict(penlogis.fit,
                            penalized = test.input)
cm.penlogis.te <- confusionMatrix(as.factor(ifelse(penlogis.pred.te >= 0.5,
                                                   "MM",
```

```
                                                    "CH")),
                                   test$Purchase)
rocplot(penlogis.pred.tr,
        train$Purchase,
        main="Training Data")
rocplot(penlogis.pred.te,
        test$Purchase,
        main="Test Data")
mtext("ROC for Penalized Logistic Regression", outer = TRUE, cex = 1.5)
auc.penlogis.tr <- colAUC(penlogis.pred.tr, train$Purchase)
auc.penlogis.te <- colAUC(penlogis.pred.te, test$Purchase)


# glmnet
a <- seq(0, 1, 0.01)
l <- seq(0, 1, 0.01)
cverr <- numeric(length(a))
lambda <- numeric(length(a))
train.input <- model.matrix(Purchase ~ ., data = train)
test.input <- model.matrix(Purchase ~ ., data = test)
for (i in 1:length(a)){
        cvglm <- cv.glmnet(train.input,
                           train$Purchase,
                           family = "binomial",
                           type.measure = "class",
                           lambda = l,
                           alpha = a[i],
                           parallel = TRUE)
        lambda[i] <- cvglm$lambda.min
        cverr[i] <- min(cvglm$cvm)
}
lambda <- lambda[which.min(cverr)]
a <- a[which.min(cverr)]
glmnet.fit <- glmnet(train.input,
                     train$Purchase,
                     family = "binomial",
                     lambda = lambda,
                     alpha = a)
glmnet.pred.tr <- predict(glmnet.fit,
                          train.input,
                          type = "response")
cm.glmnet.tr <- confusionMatrix(as.factor(ifelse(penlogis.pred.tr >= 0.5,
                                                    "MM",
                                                    "CH")),
                                   train$Purchase)
glmnet.pred.te <- predict(glmnet.fit,
                          test.input,
                          type = "response")
cm.glmnet.te <- confusionMatrix(as.factor(ifelse(penlogis.pred.te >= 0.5,
                                                  "MM",
                                                  "CH")),
                                 test$Purchase)
rocplot(glmnet.pred.tr,
```

```
        train$Purchase,
        main="Training Data")
rocplot(glmnet.pred.te,
        test$Purchase,
        main="Test Data")
mtext("ROC for Penalized Logistic Regression", outer = TRUE, cex = 1.5)
auc.glmnet.tr <- colAUC(glmnet.pred.tr, train$Purchase)
auc.glmnet.te <- colAUC(glmnet.pred.te, test$Purchase)
```