Homework 5

Cong Zhang

2021-04-28

This is my solution to Homework 5.

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

Import data.

```
data(OJ)
oj_df = OJ %>%
  janitor::clean_names()

set.seed(1)
row_train <- createDataPartition(y = oj_df$purchase, p = 799/1070, list = FALSE)

# training data
x_train <- model.matrix(purchase ~ ., oj_df)[row_train,-1]
y_train <- oj_df$purchase[row_train]
data_train <- subset(oj_df[row_train,])

# test data
x_test <- model.matrix(purchase ~ ., oj_df)[-row_train,-1]
y_test <- oj_df$purchase[-row_train]
data_test <- subset(oj_df[-row_train,])
```

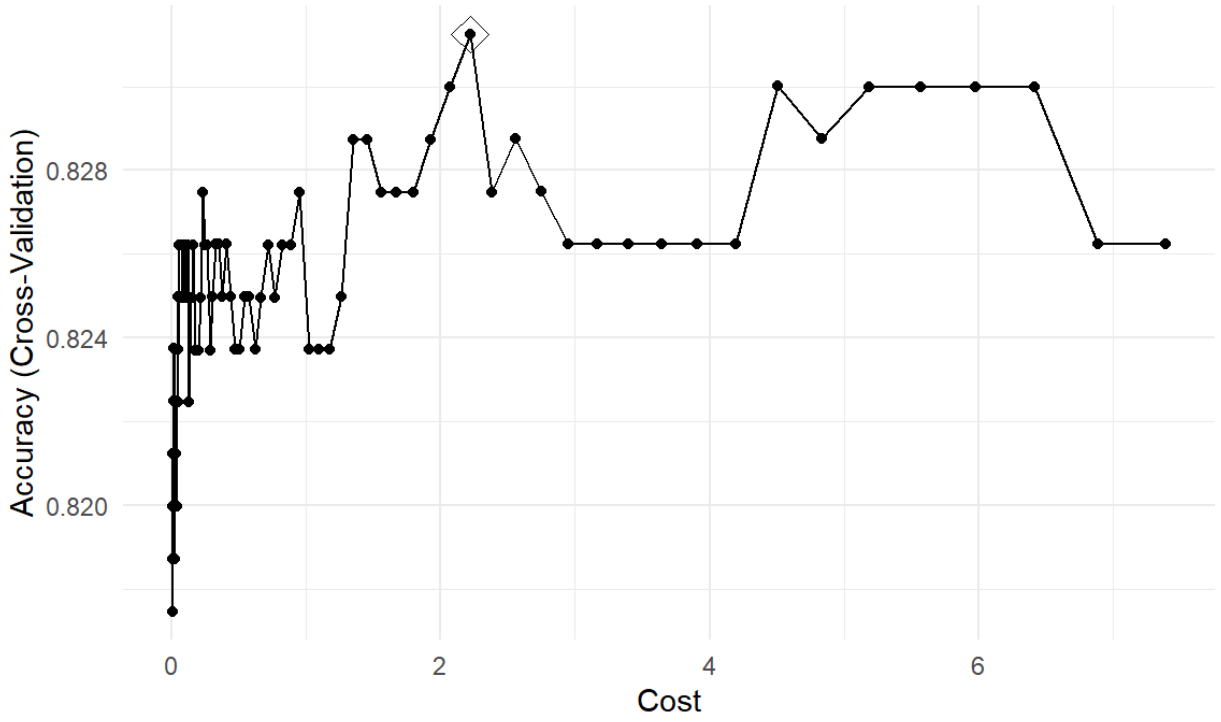
(a) Fit a support vector classifier (linear kernel) to the training data with Purchase as the response and the other variables as predictors. What are the training and test error rates?

Fit a support vector classifier (linear kernel).

```
ctrl = trainControl(method = "cv")

set.seed(1)
svml_fit = train(purchase ~ .,
                 data = data_train,
                 method = "svmLinear",
                 preProcess = c("center", "scale"),
                 tuneGrid = data.frame(C = exp(seq(-5, 2, len = 100))),
                 trControl = ctrl)

ggplot(svml_fit, highlight = TRUE, xTrans = log)
```




```
svml_fit$bestTune
```

```
##           C
## 83 2.221049
```

```
svml_fit$finalModel
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 2.22104942461286
##
## Linear (vanilla) kernel function.
##
## Number of Support Vectors : 345
##
## Objective Function Value : -745.5878
## Training error : 0.16375
```

Calculate the training and test error rates.

```
pred_svml_train = predict(svml_fit)

mean(data_train$purchase != pred_svml_train)
```

```
## [1] 0.16375
```

```
pred_svml_test = predict(svml_fit, newdata = data_test, type = "raw")

confusionMatrix(data = pred_svml_test,
                reference = data_test$purchase)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  CH  MM
##         CH 145  21
##         MM  20  84
##
##                Accuracy : 0.8481
##                  95% CI : (0.7997, 0.8888)
##     No Information Rate : 0.6111
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.68
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8788
##             Specificity : 0.8000
##          Pos Pred Value : 0.8735
##          Neg Pred Value : 0.8077
##              Prevalence : 0.6111
##          Detection Rate : 0.5370
##    Detection Prevalence : 0.6148
##       Balanced Accuracy : 0.8394
##
##        'Positive' Class : CH
##
```

```
mean(data_test$purchase != pred_svml_test)
```

```
## [1] 0.1518519
```

From the result, we could see the training error rate is 16.375%, and the test error rate is 15.1851852%.

(b) Fit a support vector machine with a radial kernel to the training data. What are the training and test error rates?

Fit a support vector machine with a radial kernel.

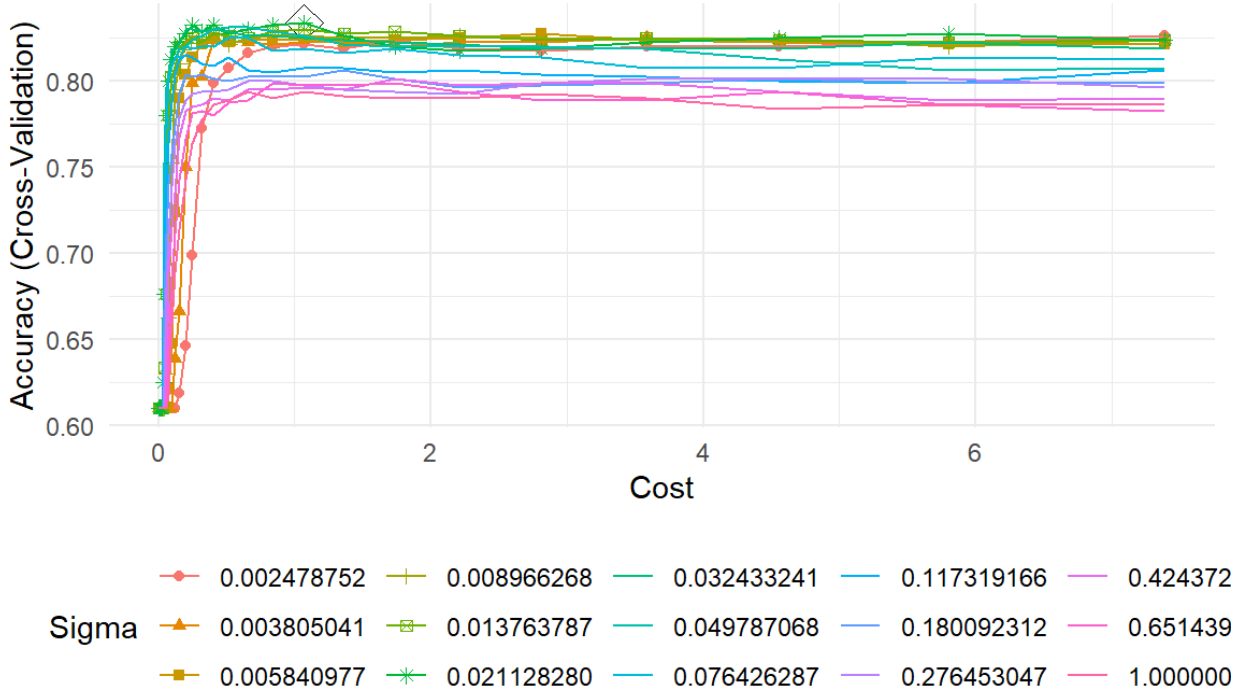
```
svmr_grid = expand.grid(C = exp(seq(-5, 2, len = 30)),
                        sigma = exp(seq(-6, 0, len = 15)))

set.seed(1)
svmr_fit = train(purchase ~ .,
                 data = data_train,
                 method = "svmRadial",
                 preProcess = c("center", "scale"),
                 tuneGrid = svmr_grid,
                 trControl = ctrl)

ggplot(svmr_fit, highlight = TRUE, xTrans = log)
```

```
## Warning: The shape palette can deal with a maximum of 6 discrete values because
## more than 6 becomes difficult to discriminate; you have 15. Consider
## specifying shapes manually if you must have them.
```

```
## Warning: Removed 270 rows containing missing values (geom_point).
```




```
svmr_fit$bestTune
```

```
##          sigma        C
## 321 0.02112828 1.071399
```

```
svmr_fit$finalModel
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 1.07139926370916
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.0211282798811833
##
## Number of Support Vectors : 389
##
## Objective Function Value : -368.0806
## Training error : 0.1575
```

Calculate the training and test error rates.

```
pred_svmr_train = predict(svmr_fit)

mean(data_train$purchase != pred_svmr_train)
```

```
## [1] 0.1575
```

```
pred_svmr_test = predict(svmr_fit, newdata = data_test, type = "raw")

confusionMatrix(data = pred_svmr_test,
                reference = data_test$purchase)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  CH  MM
##         CH 147  24
##         MM  18  81
##
##                Accuracy : 0.8444
##                  95% CI : (0.7956, 0.8855)
##     No Information Rate : 0.6111
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.6693
##
##  Mcnemar's Test P-Value : 0.4404
##
##             Sensitivity : 0.8909
##             Specificity : 0.7714
##          Pos Pred Value : 0.8596
##          Neg Pred Value : 0.8182
##              Prevalence : 0.6111
##          Detection Rate : 0.5444
##    Detection Prevalence : 0.6333
##       Balanced Accuracy : 0.8312
##
##        'Positive' Class : CH
##
```

```
mean(data_test$purchase != pred_svmr_test)
```

```
## [1] 0.1555556
```

From the result, we could see the training error rate is 15.75%, and the test error rate is 15.5555556%.

Compare the two models.

```
resamp = resamples(list(svml = svml_fit,
                        svmr = svmr_fit))
summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: svml, svmr
## Number of resamples: 10
##
## Accuracy
##       Min.   1st Qu.    Median      Mean  3rd Qu.      Max. NA's
## svml 0.775 0.803125 0.8375000 0.8312461 0.859375 0.8765432    0
## svmr 0.800 0.815625 0.8364715 0.8337141 0.850000 0.8641975    0
##
## Kappa
##           Min.    1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## svml 0.5422759 0.5771912 0.6513001 0.6396985 0.6963322 0.7388781    0
## svmr 0.5736176 0.6178730 0.6438386 0.6442212 0.6753376 0.7143315    0
```

```
bwplot(resamp)
```