P8106 HW5 yz4184

Yunlin Zhou

Contents

```r
library(tidyverse)
library(caret)
library(e1071)
library(kernlab)
library(ISLR)
library(factoextra)
library(gridExtra)
library(corrplot)
library(RColorBrewer)
library(gplots)
library(jpeg)
```

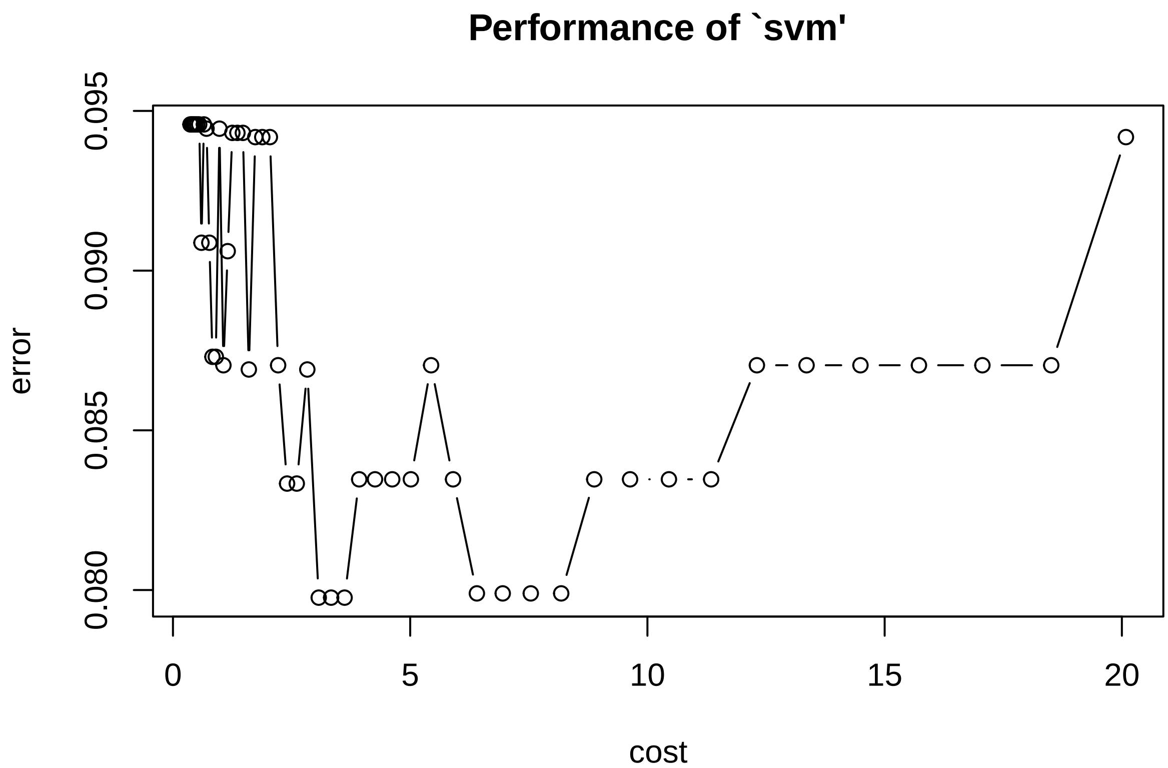
Problem 1

```r
# import data
dat = read.csv("./auto.csv")%>%
  na.omit() %>%
  mutate(
    cylinders = as.factor(cylinders),
         year = as.factor(year),
         origin = as.factor(origin),
    mpg_cat = factor(mpg_cat, levels = c("low", "high")))

# divide data into two parts (training and test)
set.seed(1)
rowTrain <- createDataPartition(y = dat$mpg_cat,
                                p = 0.7,
                                list = FALSE)

train_df = dat[rowTrain,]
test_df = dat[-rowTrain,]
```

Part a

Fit a support vector classifier (linear kernel) to the training data.

```r
set.seed(1)
linear.tune <- tune.svm( mpg_cat ~ . ,
data = train_df,
kernel = "linear",
cost = exp(seq(-1,3,len=50)),
scale = TRUE)
plot(linear.tune)
```




```r
best.linear <- linear.tune$best.model
summary(best.linear)
```

```
##
## Call:
## best.svm(x = mpg_cat ~ ., data = train_df, cost = exp(seq(-1, 3,
##     len = 50)), kernel = "linear", scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  3.072369
##
## Number of Support Vectors:  50
##
##  ( 27 23 )
##
##
## Number of Classes:  2
##
## Levels:
##  low high
```

According to the cost-error plot and best model summary above, we can conclude that the best tuning parameter c is 3.072369.

There are 50 support vectors in the optimal support vector classifier with a linear kernel.

Training error rate

```
#train error
pred.linear.train <- predict(best.linear, newdata = train_df)
confusionMatrix(data = pred.linear.train,
                reference = train_df$mpg_cat)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction low high
##       low  134    7
##       high   4  131
##
##                Accuracy : 0.9601
##                  95% CI : (0.9298, 0.9799)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9203
##
##  Mcnemar's Test P-Value : 0.5465
##
##             Sensitivity : 0.9710
##             Specificity : 0.9493
##          Pos Pred Value : 0.9504
##          Neg Pred Value : 0.9704
##              Prevalence : 0.5000
##          Detection Rate : 0.4855
##    Detection Prevalence : 0.5109
##       Balanced Accuracy : 0.9601
##
##        'Positive' Class : low
##
```

According to the confusion Matrix above, the accuracy is 0.9601, so the training error rate is (1-0.9601)*100% = 3.99% .

Test error rate

```
#test error
pred.linear.test <- predict(best.linear, newdata = test_df)
confusionMatrix(data = pred.linear.test,
                reference = test_df$mpg_cat)
```

```
## Confusion Matrix and Statistics
##
```

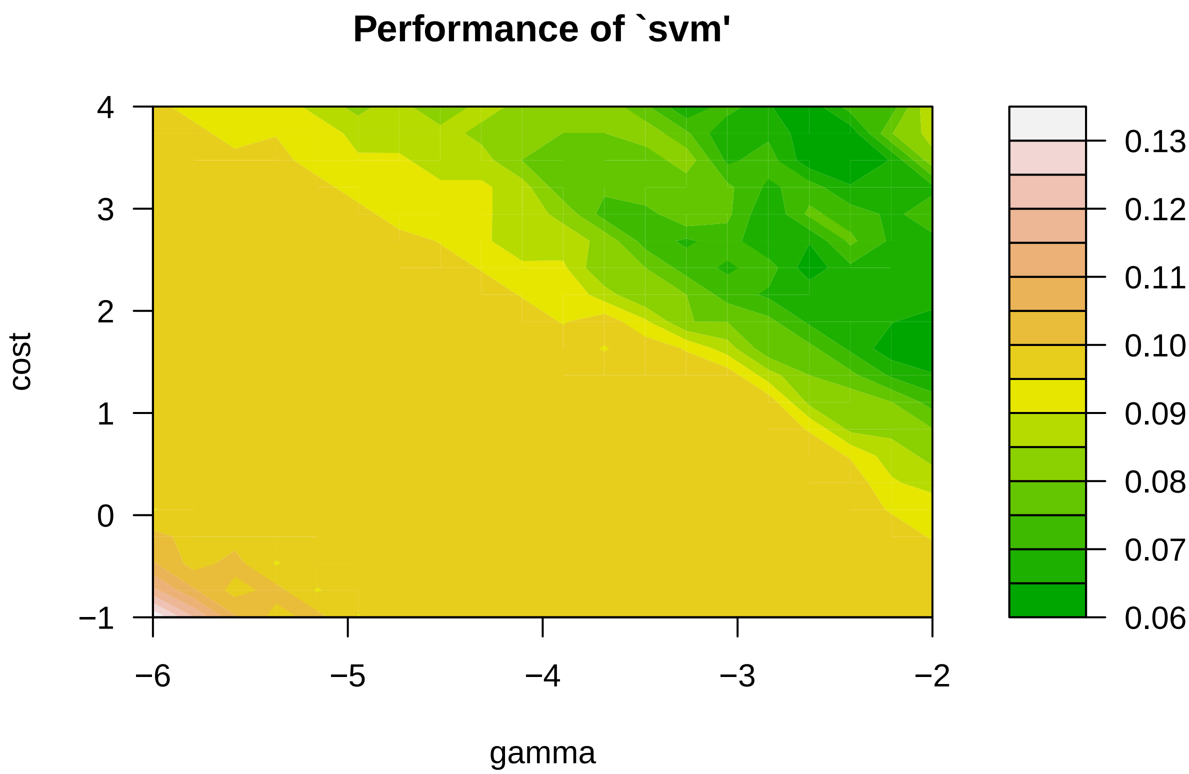
```
##           Reference
## Prediction low high
##       low   50    4
##       high   8   54
##
##                Accuracy : 0.8966
##                  95% CI : (0.8263, 0.9454)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7931
##
##  Mcnemar's Test P-Value : 0.3865
##
##             Sensitivity : 0.8621
##             Specificity : 0.9310
##          Pos Pred Value : 0.9259
##          Neg Pred Value : 0.8710
##              Prevalence : 0.5000
##          Detection Rate : 0.4310
##    Detection Prevalence : 0.4655
##       Balanced Accuracy : 0.8966
##
##        'Positive' Class : low
##
```

According to the confusion Matrix above, the accuracy is 0.8966, so the test error rate is (1-0.8966)*100% = 10.34% .

Part b

Fit a support vector machine with a radial kernel to the training data.

```
set.seed(1)
radial.tune <- tune.svm( mpg_cat ~ . ,
                          data = train_df,
                        kernel = "radial",
                        cost = exp(seq(-1,4,len=20)),
                        gamma = exp(seq(-6,-2,len=20)))

plot(radial.tune, transform.y = log, transform.x = log,
     color.palette = terrain.colors)
```




```
radial.tune$best.parameters
```

```
##           gamma     cost
## 357 0.07196474 32.25536
```

```
best.radial <- radial.tune$best.model
summary(best.radial)
```

```
##
```

```
## Call:
## best.svm(x = mpg_cat ~ ., data = train_df, gamma = exp(seq(-6, -2,
##     len = 20)), cost = exp(seq(-1, 4, len = 20)), kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  32.25536
##
## Number of Support Vectors:  54
##
##  ( 29 25 )
##
##
## Number of Classes:  2
##
## Levels:
##  low high
```

According to the gamma-cost plot and best parameters summary above, we can conclude that the best tuning parameters, gamma and cost, of the support vector machine are 0.07196474 and 32.25536.

There are 54 support vectors in the optimal support vector classifier with a linear kernel.

Training error rate

```
#train error
pred.radial <- predict(best.radial, newdata = train_df)

confusionMatrix(data = pred.radial,
                reference = train_df$mpg_cat)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction low high
##       low  137    2
##       high   1  136
##
##                Accuracy : 0.9891
##                  95% CI : (0.9686, 0.9978)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9783
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9928
##             Specificity : 0.9855
##          Pos Pred Value : 0.9856
```

```
##          Neg Pred Value : 0.9927
##              Prevalence : 0.5000
##          Detection Rate : 0.4964
##    Detection Prevalence : 0.5036
##       Balanced Accuracy : 0.9891
##
##        'Positive' Class : low
##
```

According to the confusion Matrix above, the accuracy is 0.9891, so the training error rate is (1-0.9891)*100% = 1.09% .

Test error rate

```r
#test error
pred.radial <- predict(best.radial, newdata = test_df)

confusionMatrix(data = pred.radial,
                reference = test_df$mpg_cat)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction low high
##       low   49    4
##       high   9   54
##
##                Accuracy : 0.8879
##                  95% CI : (0.816, 0.939)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7759
##
##  Mcnemar's Test P-Value : 0.2673
##
##             Sensitivity : 0.8448
##             Specificity : 0.9310
##          Pos Pred Value : 0.9245
##          Neg Pred Value : 0.8571
##              Prevalence : 0.5000
##          Detection Rate : 0.4224
##    Detection Prevalence : 0.4569
##       Balanced Accuracy : 0.8879
##
##        'Positive' Class : low
##
```

According to the confusion Matrix above, the accuracy is 0.8879, so the test error rate is (1-0.8879)*100% = 11.21% .

Problem 2

```
# import data
data(USArrests)
arrests_df = USArrests %>%
  as.data.frame() %>%
  janitor::clean_names()
```

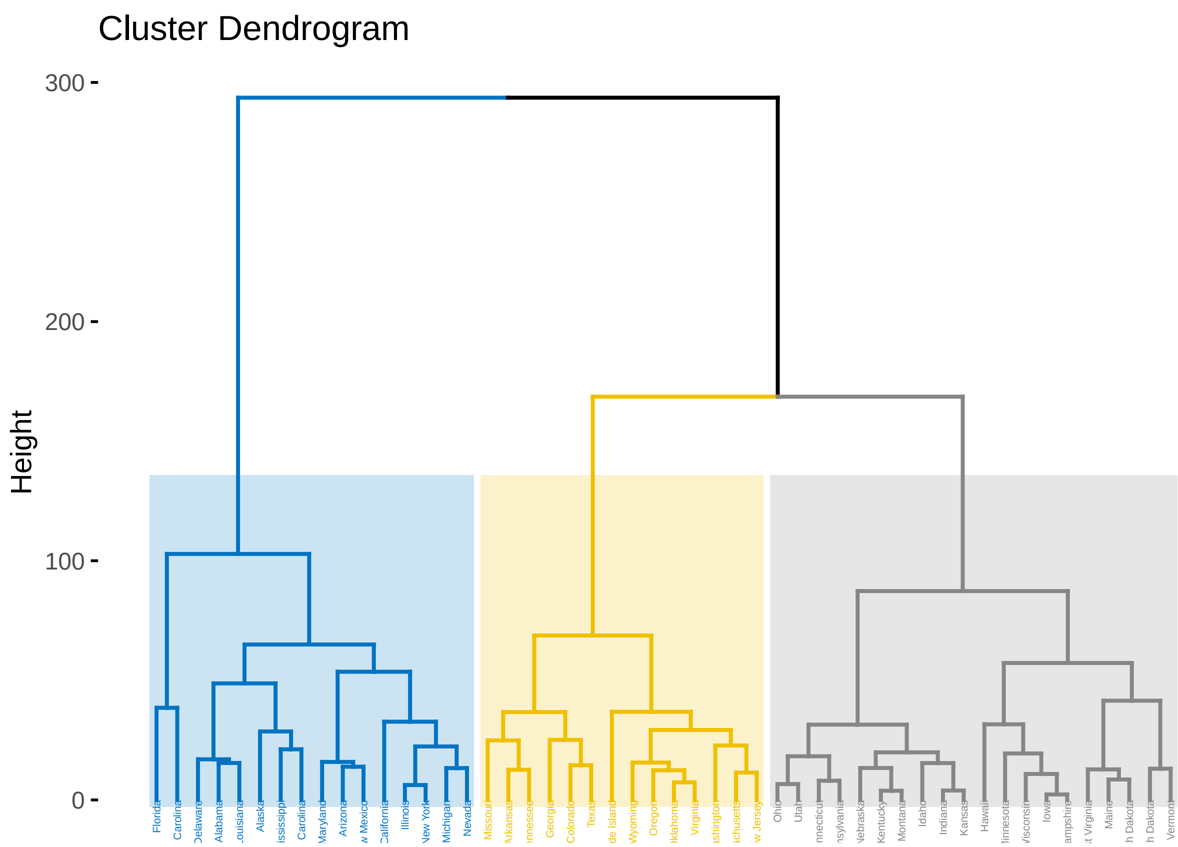
Part a

Using hierarchical clustering with complete linkage and Euclidean distance, cluster the states.

```
hc.complete <- hclust(dist(arrests_df), method = "complete")
```

Cut the dendrogram at a height that results in three distinct clusters.

```
fviz_dend(hc.complete, k = 3,
cex = 0.3,
palette = "jco",
color_labels_by_k = TRUE,
rect = TRUE, rect_fill = TRUE, rect_border = "jco",
labels_track_height = 2.5)
```



States belong to the first cluster

```
state_clusters = cutree(hc.complete, 3)

row.names(arrests_df[state_clusters == 1,])
```

```
##  [1] "Alabama"        "Alaska"         "Arizona"        "California"
##  [5] "Delaware"       "Florida"        "Illinois"       "Louisiana"
##  [9] "Maryland"       "Michigan"       "Mississippi"    "Nevada"
## [13] "New Mexico"     "New York"       "North Carolina" "South Carolina"
```

States belong to the second cluster

```
row.names(arrests_df[state_clusters == 2,])
```

```
##  [1] "Arkansas"      "Colorado"      "Georgia"       "Massachusetts"
##  [5] "Missouri"      "New Jersey"    "Oklahoma"      "Oregon"
##  [9] "Rhode Island"  "Tennessee"     "Texas"         "Virginia"
## [13] "Washington"    "Wyoming"
```

States belong to the third cluster

```
row.names(arrests_df[state_clusters == 3,])
```

```
##  [1] "Connecticut"   "Hawaii"        "Idaho"         "Indiana"
##  [5] "Iowa"          "Kansas"        "Kentucky"      "Maine"
##  [9] "Minnesota"     "Montana"       "Nebraska"      "New Hampshire"
## [13] "North Dakota"  "Ohio"          "Pennsylvania"  "South Dakota"
## [17] "Utah"          "Vermont"       "West Virginia" "Wisconsin"
```

Part b

Scaling the variables to have standard deviation one.

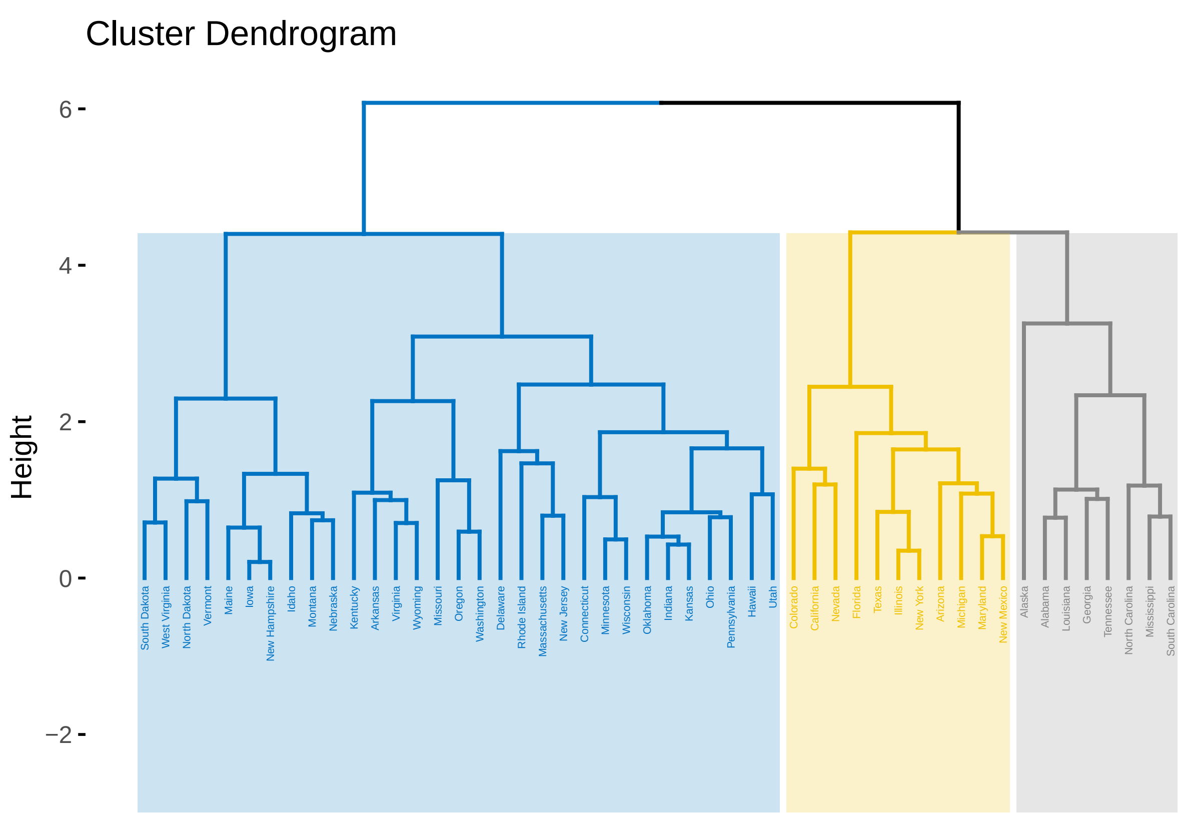
```
arrests_df_scale = scale(arrests_df)
```

Using hierarchical clustering with complete linkage and Euclidean distance, cluster the states.

```
hc.complete.2 <- hclust(dist(arrests_df_scale), method = "complete")
```

Cut the dendrogram at a height that results in three distinct clusters.

```
fviz_dend(hc.complete.2, k = 3,
cex = 0.3,
palette = "jco",
color_labels_by_k = TRUE,
rect = TRUE, rect_fill = TRUE, rect_border = "jco",
labels_track_height = 2.5)
```



States belong to the first cluster

```
state_clusters_2 = cutree(hc.complete.2, 3)

row.names(arrests_df[state_clusters_2 == 1,])
```

```
## [1] "Alabama"        "Alaska"         "Georgia"        "Louisiana"
## [5] "Mississippi"    "North Carolina" "South Carolina" "Tennessee"
```

States belong to the second cluster

```
row.names(arrests_df[state_clusters_2 == 2,])
```

```
##  [1] "Arizona"    "California" "Colorado"   "Florida"    "Illinois"
##  [6] "Maryland"   "Michigan"   "Nevada"     "New Mexico" "New York"
## [11] "Texas"
```

States belong to the third cluster

```
row.names(arrests_df[state_clusters_2 == 3,])
```

```
##  [1] "Arkansas"      "Connecticut"   "Delaware"      "Hawaii"
##  [5] "Idaho"         "Indiana"       "Iowa"          "Kansas"
##  [9] "Kentucky"      "Maine"         "Massachusetts" "Minnesota"
## [13] "Missouri"      "Montana"       "Nebraska"      "New Hampshire"
## [17] "New Jersey"    "North Dakota"  "Ohio"          "Oklahoma"
## [21] "Oregon"        "Pennsylvania"  "Rhode Island"  "South Dakota"
## [25] "Utah"          "Vermont"       "Virginia"      "Washington"
## [29] "West Virginia" "Wisconsin"     "Wyoming"
```

Part c

Scaling the variables changed the clustering results.

Since many clustering algorithms require some definition of distance, if you do not scale and center your data, you may give attributes which have larger magnitudes more importance.

If one of your features has a range of values much larger than the others, clustering will be completely dominated by that one feature.

In this problem, we are using Euclidean distance. In this data set, the variable urban_pop has an incomparable units to other variables.

In my opinion, the variables should be scaled before the inter-observation dissimilarities are computed. So that our variables are in comparable units and the algorithm could assign equal weight to the variables.