HW4 SML

Jie Li jl5246

March 24, 2019

```
## Warning: package 'e1071' was built under R version 3.5.3
```

Problem 1 Kernelized Nearest Neighbor Classification

1

$$d^2(x, x') = ||x - x'||_2^2 =< (x - x'), (x - x') >=< x, x > -2 < x, x' > + < x', x' >$$

2

$$d_k^2(x, x') =< \phi(x), \phi(x) > -2 < \phi(x), \phi(x') > + < \phi(x'), \phi(x') >$$

3

d_k^2(x, x') is itself a distance measure that map data X into high-dimension feature space by `kernel` function and calculate the distances between data points X in that space.

Problem 2 Subset Selection Methods

First of all, take a look our `Credit` dataset

```
data = read.csv("./Credit.csv")
head(data)
```

```
##   X  Income Limit Rating Cards Age Education Gender Student Married
## 1 1  14.891  3606    283     2  34        11   Male      No     Yes
## 2 2 106.025  6645    483     3  82        15 Female     Yes     Yes
## 3 3 104.593  7075    514     4  71        11   Male      No      No
## 4 4 148.924  9504    681     3  36        11 Female      No      No
## 5 5  55.882  4897    357     2  68        16   Male      No     Yes
## 6 6  80.180  8047    569     4  77        10   Male      No      No
##   Ethnicity Balance
## 1 Caucasian     333
## 2     Asian     903
## 3     Asian     580
## 4     Asian     964
## 5 Caucasian     331
## 6 Caucasian    1151
```

Using `Regsubsets` function to ientify the best model, which minimizes the residual sum-of-squares (RSS).

```
model.set = regsubsets(Balance ~ Income + Limit + Rating + Cards + Age + Education + I(Gender) + I(Studen
t) + I(Ethnicity) + I(Married), data)
summary(model.set)
```

```
## Subset selection object
## Call: regsubsets.formula(Balance ~ Income + Limit + Rating + Cards +
##     Age + Education + I(Gender) + I(Student) + I(Ethnicity) +
##     I(Married), data)
## 11 Variables  (and intercept)
##                        Forced in Forced out
## Income                     FALSE      FALSE
## Limit                      FALSE      FALSE
## Rating                     FALSE      FALSE
## Cards                      FALSE      FALSE
## Age                        FALSE      FALSE
## Education                  FALSE      FALSE
## I(Gender)Female            FALSE      FALSE
## I(Student)Yes              FALSE      FALSE
## I(Ethnicity)Asian          FALSE      FALSE
## I(Ethnicity)Caucasian      FALSE      FALSE
## I(Married)Yes              FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: exhaustive
##          Income Limit Rating Cards Age Education I(Gender)Female
## 1  ( 1 ) " "    " "   "*"    " "   " " " "       " "
## 2  ( 1 ) "*"    " "   "*"    " "   " " " "       " "
## 3  ( 1 ) "*"    " "   "*"    " "   " " " "       " "
## 4  ( 1 ) "*"    "*"   " "    "*"   " " " "       " "
## 5  ( 1 ) "*"    "*"   "*"    "*"   " " " "       " "
## 6  ( 1 ) "*"    "*"   "*"    "*"   "*" " "       " "
## 7  ( 1 ) "*"    "*"   "*"    "*"   "*" " "       "*"
## 8  ( 1 ) "*"    "*"   "*"    "*"   "*" " "       "*"
##          I(Student)Yes I(Ethnicity)Asian I(Ethnicity)Caucasian
## 1  ( 1 ) " "           " "               " "
## 2  ( 1 ) " "           " "               " "
## 3  ( 1 ) "*"           " "               " "
## 4  ( 1 ) "*"           " "               " "
## 5  ( 1 ) "*"           " "               " "
## 6  ( 1 ) "*"           " "               " "
## 7  ( 1 ) "*"           " "               " "
## 8  ( 1 ) "*"           "*"               " "
##          I(Married)Yes
## 1  ( 1 ) " "
## 2  ( 1 ) " "
## 3  ( 1 ) " "
## 4  ( 1 ) " "
## 5  ( 1 ) " "
## 6  ( 1 ) " "
## 7  ( 1 ) " "
## 8  ( 1 ) " "
```

Performing `forward stepwise selection`

```
model.foward = regsubsets(Balance ~ Income + Limit + Rating + Cards + Age + Education + I(Gender) + I(Stud
ent) + I(Ethnicity) + I(Married), data, method = "forward")
summary(model.foward)
```

```
## Subset selection object
## Call: regsubsets.formula(Balance ~ Income + Limit + Rating + Cards +
##     Age + Education + I(Gender) + I(Student) + I(Ethnicity) +
##     I(Married), data, method = "forward")
## 11 Variables  (and intercept)
##                        Forced in Forced out
## Income                     FALSE      FALSE
## Limit                      FALSE      FALSE
## Rating                     FALSE      FALSE
## Cards                      FALSE      FALSE
## Age                        FALSE      FALSE
## Education                  FALSE      FALSE
## I(Gender)Female            FALSE      FALSE
## I(Student)Yes              FALSE      FALSE
## I(Ethnicity)Asian          FALSE      FALSE
## I(Ethnicity)Caucasian      FALSE      FALSE
## I(Married)Yes              FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: forward
##          Income Limit Rating Cards Age Education I(Gender)Female
## 1  ( 1 ) " "    " "   "*"    " "   " " " "       " "
## 2  ( 1 ) "*"    " "   "*"    " "   " " " "       " "
## 3  ( 1 ) "*"    " "   "*"    " "   " " " "       " "
## 4  ( 1 ) "*"    "*"   "*"    " "   " " " "       " "
## 5  ( 1 ) "*"    "*"   "*"    "*"   " " " "       " "
## 6  ( 1 ) "*"    "*"   "*"    "*"   "*" " "       " "
## 7  ( 1 ) "*"    "*"   "*"    "*"   "*" " "       "*"
## 8  ( 1 ) "*"    "*"   "*"    "*"   "*" " "       "*"
##          I(Student)Yes I(Ethnicity)Asian I(Ethnicity)Caucasian
## 1  ( 1 ) " "           " "               " "
## 2  ( 1 ) " "           " "               " "
## 3  ( 1 ) "*"           " "               " "
## 4  ( 1 ) "*"           " "               " "
## 5  ( 1 ) "*"           " "               " "
## 6  ( 1 ) "*"           " "               " "
## 7  ( 1 ) "*"           " "               " "
## 8  ( 1 ) "*"           "*"               " "
##          I(Married)Yes
## 1  ( 1 ) " "
## 2  ( 1 ) " "
## 3  ( 1 ) " "
## 4  ( 1 ) " "
## 5  ( 1 ) " "
## 6  ( 1 ) " "
## 7  ( 1 ) " "
## 8  ( 1 ) " "
```

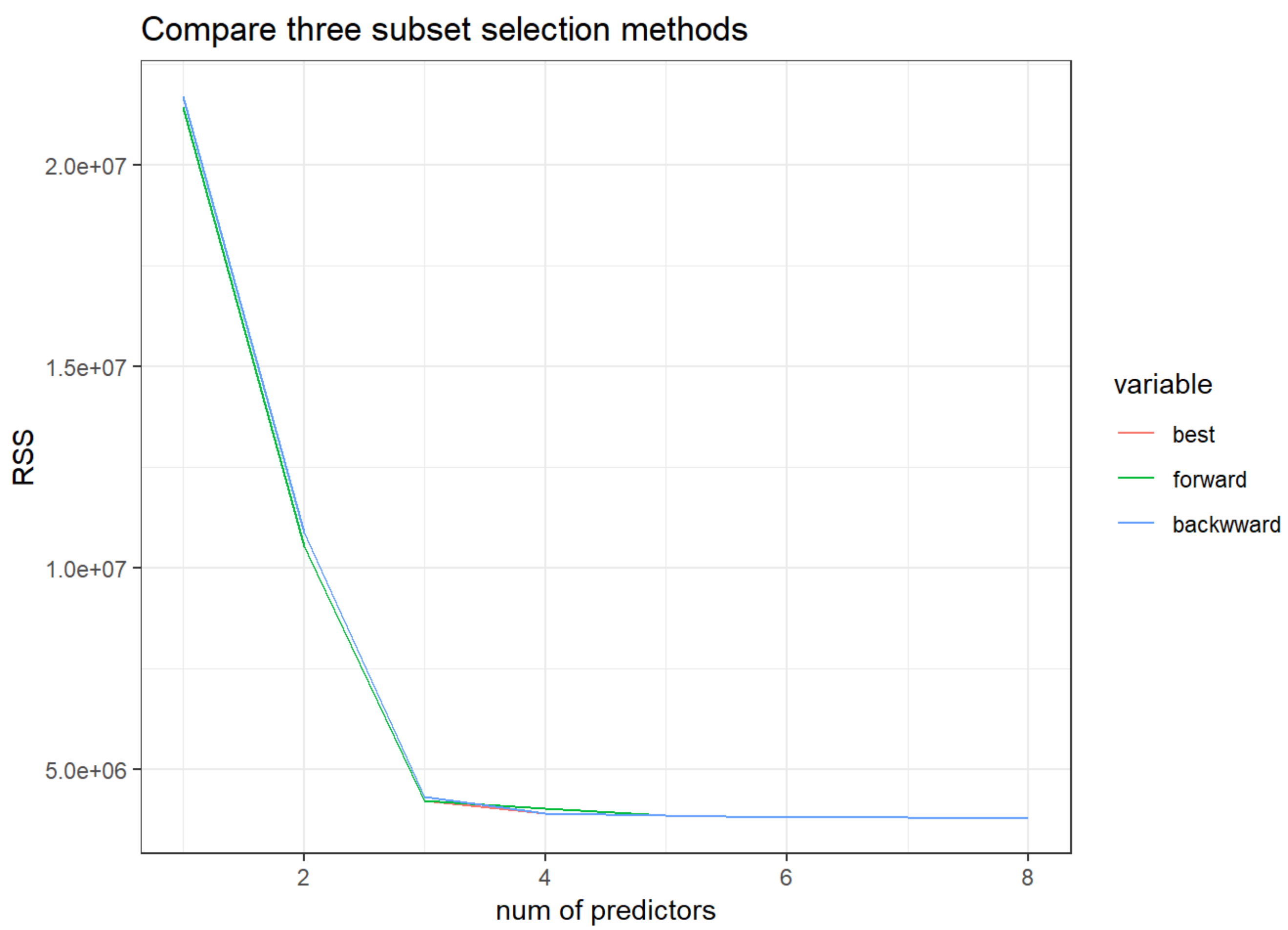
Performing `backward stepwise selection`

```
model.backward = regsubsets(Balance ~ Income + Limit + Rating + Cards + Age + Education + I(Gender) + I(St
udent) + I(Ethnicity) + I(Married), data, method = "backward")
summary(model.backward)
```

```
## Subset selection object
## Call: regsubsets.formula(Balance ~ Income + Limit + Rating + Cards +
##     Age + Education + I(Gender) + I(Student) + I(Ethnicity) +
##     I(Married), data, method = "backward")
## 11 Variables  (and intercept)
##                         Forced in Forced out
## Income                      FALSE      FALSE
## Limit                       FALSE      FALSE
## Rating                      FALSE      FALSE
## Cards                       FALSE      FALSE
## Age                         FALSE      FALSE
## Education                   FALSE      FALSE
## I(Gender)Female             FALSE      FALSE
## I(Student)Yes               FALSE      FALSE
## I(Ethnicity)Asian           FALSE      FALSE
## I(Ethnicity)Caucasian       FALSE      FALSE
## I(Married)Yes               FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: backward
##          Income Limit Rating Cards Age Education I(Gender)Female
## 1  ( 1 ) " "    "*"   " "    " "   " " " "       " "            
## 2  ( 1 ) "*"    "*"   " "    " "   " " " "       " "            
## 3  ( 1 ) "*"    "*"   " "    " "   " " " "       " "            
## 4  ( 1 ) "*"    "*"   " "    "*"   " " " "       " "            
## 5  ( 1 ) "*"    "*"   "*"    "*"   " " " "       " "            
## 6  ( 1 ) "*"    "*"   "*"    "*"   "*" " "       " "            
## 7  ( 1 ) "*"    "*"   "*"    "*"   "*" " "       "*"            
## 8  ( 1 ) "*"    "*"   "*"    "*"   "*" " "       "*"            
##          I(Student)Yes I(Ethnicity)Asian I(Ethnicity)Caucasian
## 1  ( 1 ) " "           " "               " "                  
## 2  ( 1 ) " "           " "               " "                  
## 3  ( 1 ) "*"           " "               " "                  
## 4  ( 1 ) "*"           " "               " "                  
## 5  ( 1 ) "*"           " "               " "                  
## 6  ( 1 ) "*"           " "               " "                  
## 7  ( 1 ) "*"           " "               " "                  
## 8  ( 1 ) "*"           "*"               " "                  
##          I(Married)Yes
## 1  ( 1 ) " "          
## 2  ( 1 ) " "          
## 3  ( 1 ) " "          
## 4  ( 1 ) " "          
## 5  ( 1 ) " "          
## 6  ( 1 ) " "          
## 7  ( 1 ) " "          
## 8  ( 1 ) " "          
```

1

```
RSS.df = data.frame(n = 1:8, best = summary(model.set)$rss, forward = summary(model.foward)$rss, backwward
 = summary(model.backward)$rss)

RSS.df %>% melt(id.var = "n") %>%
ggplot()+
  geom_line(aes(n, value, col = variable))+
  xlab("num of predictors")+
  ylab("RSS")+
  ggtitle("Compare three subset selection methods")+
  theme_bw()
```



2

Each subset selection method results in a set of models. For each approach, choose a single optimal model by using Cp and BIC statistics respectively.

```
cat("The lowest Cp and BIC in the Best subset selection: ", min(summary(model.set)$cp), min(summary(model.
set)$bic), "\n")
```

```
## The lowest Cp and BIC in the Best subset selection:  5.574883 -1198.053
```

```
cat("The lowest Cp and BIC in the foward subset selection: ", min(summary(model.foward)$cp), min(summary(m
odel.foward)$bic), "\n")
```

```
## The lowest Cp and BIC in the foward subset selection:  5.574883 -1197.096
```

```
cat("The lowest Cp and BIC in the backward subset selection: ", min(summary(model.backward)$cp), min(summa
ry(model.backward)$bic), "\n")
```

```
## The lowest Cp and BIC in the backward subset selection:  5.574883 -1198.053
```

I am going to choose `BIC` statistcs from the `Best subset selection` because it provides the lowest and simplest model.

```
cat("Following is the given number of predictors \n")
```

```
## Following is the given number of predictors
```

```
summary(model.set)$outmat[which.min(summary(model.set)$bic),]
```

```
##               Income              Limit             Rating
##                  "*"                "*"                " "
##                Cards                Age          Education
##                  "*"                " "                " "
##       I(Gender)Female     I(Student)Yes I(Ethnicity)Asian
##                  " "                "*"                " "
## I(Ethnicity)Caucasian       I(Married)Yes
##                  " "                " "
```

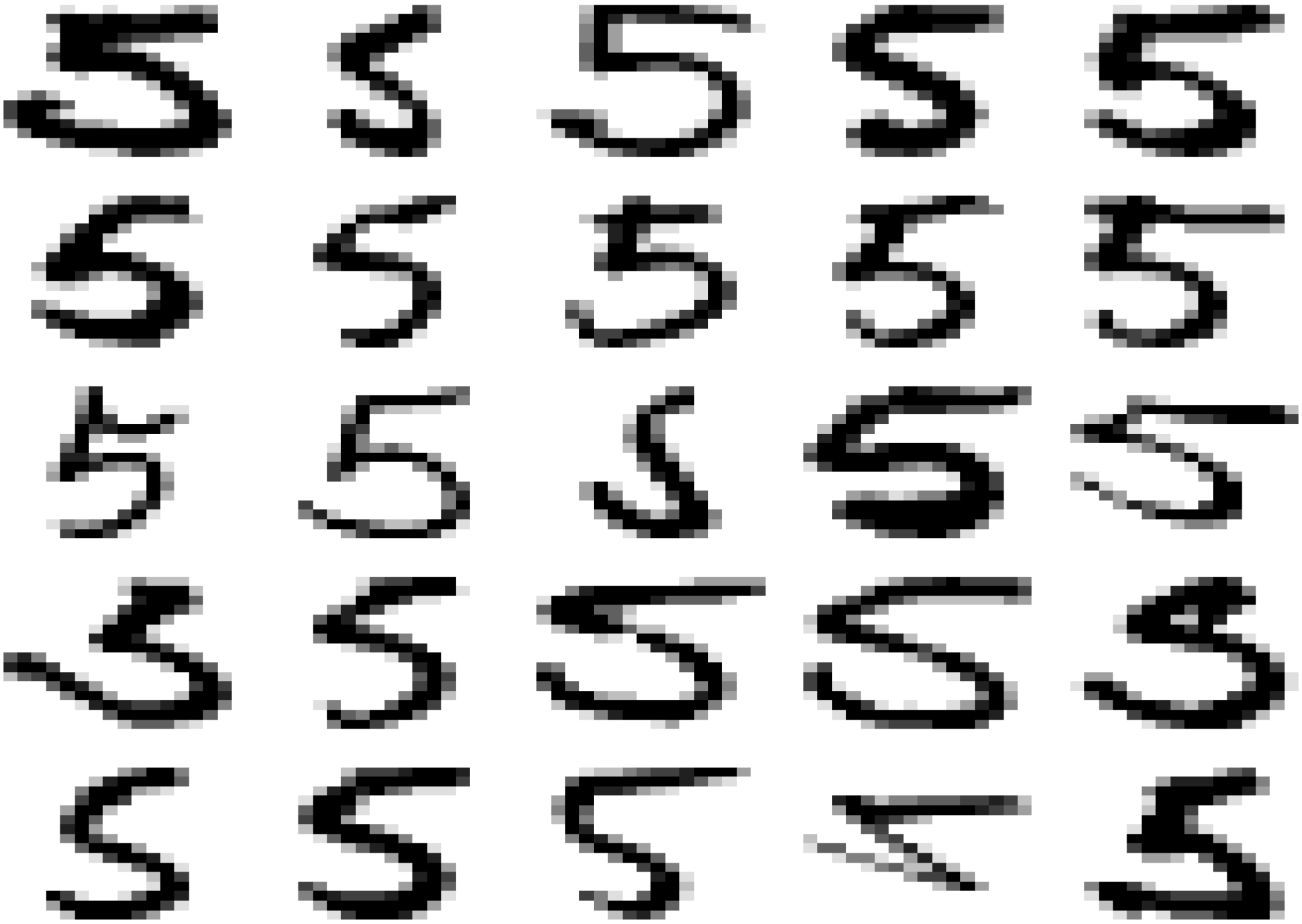
Problem 3 SVM

Loading the data set

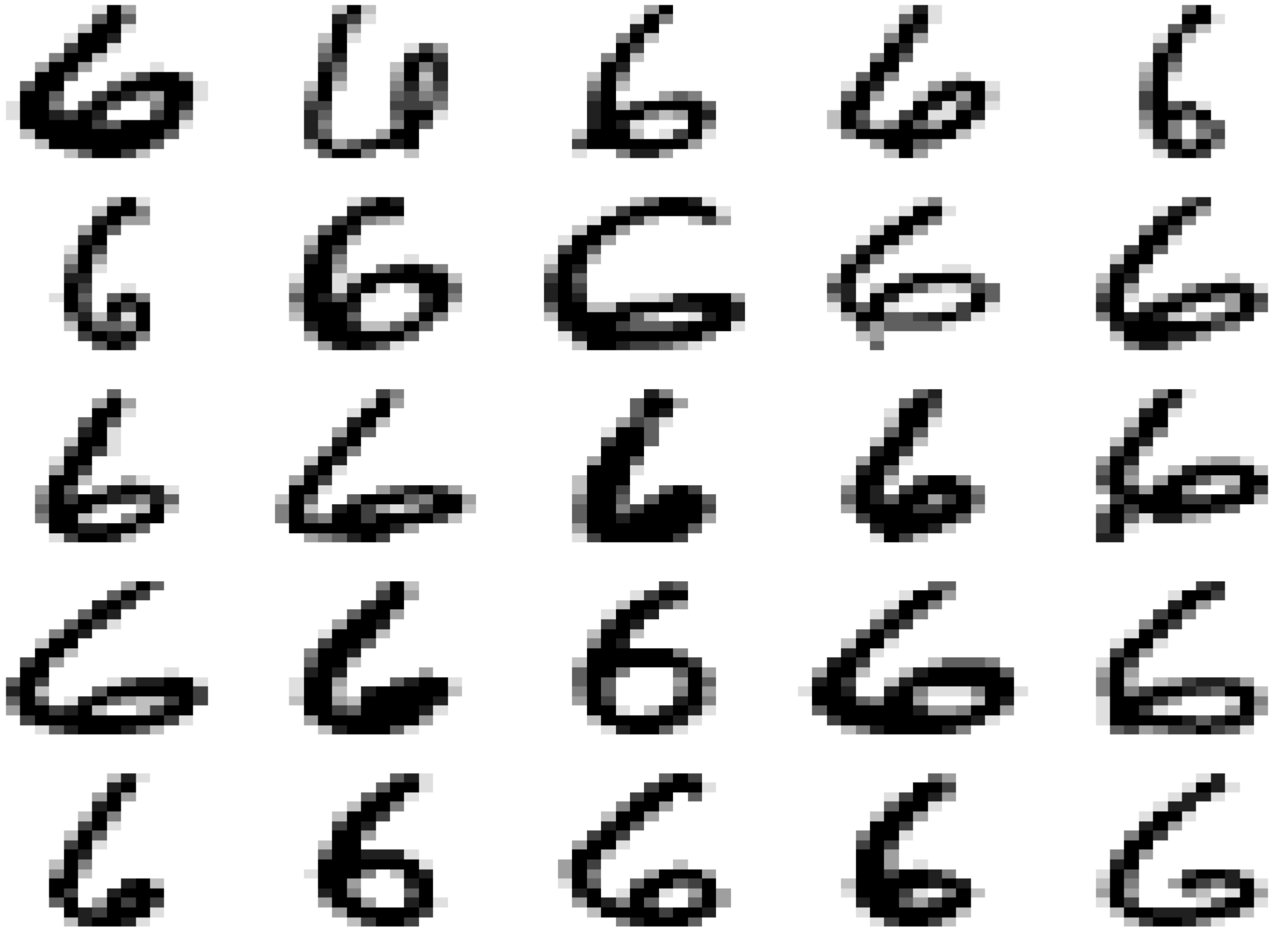
```
### all images corresponding to digit "5"
zip.5 = read.table("train.5-1.txt", header = FALSE, sep = ",")
zip.5 = cbind(zip.5, y=rep(-1, dim(zip.5)[1])) %>% data.frame()

### all images corresponding to digit "6"
zip.6 = read.table("train.6.txt", header = FALSE, sep = ",")
zip.6 = cbind(zip.6, y=rep(1, dim(zip.6)[1])) %>% data.frame()

### combine two data sets together
data = rbind(zip.5, zip.6)

# function of visualizing the image
output.image = function(data)
{
  # Transfer dataframe to vector then convert to matrix
    digit = matrix(as.numeric(data), nrow = 16, ncol = 16)

    # Set index backwards
    index = seq(from = 16, to  = 1, by = -1)
    sym_digit = digit[,index]
    image(sym_digit, col = gray((8:0)/8), axes = FALSE)
}
```

Visualizing the part of dataset

```
# Visualize digital 5
par(mfrow = c(5,5),mai = c(0.1,0.1,0.1,0.1))
for(i in 1:25)
{
    output.image(zip.5[i,-257])
}
```



```r
# Visualize digital 6
par(mfrow = c(5,5),mai = c(0.1,0.1,0.1,0.1))
for(i in 1:25)
{
    output.image(zip.6[i,-257])
}
```



Spliting train and test set, repectively 80-20

```
set.seed(123)
index = sample(1:dim(data)[1], dim(data)[1]*0.2)
test = data[index,]
train = data[-index,]

cat("Dimension of Training set: ", dim(train), "\n")
```

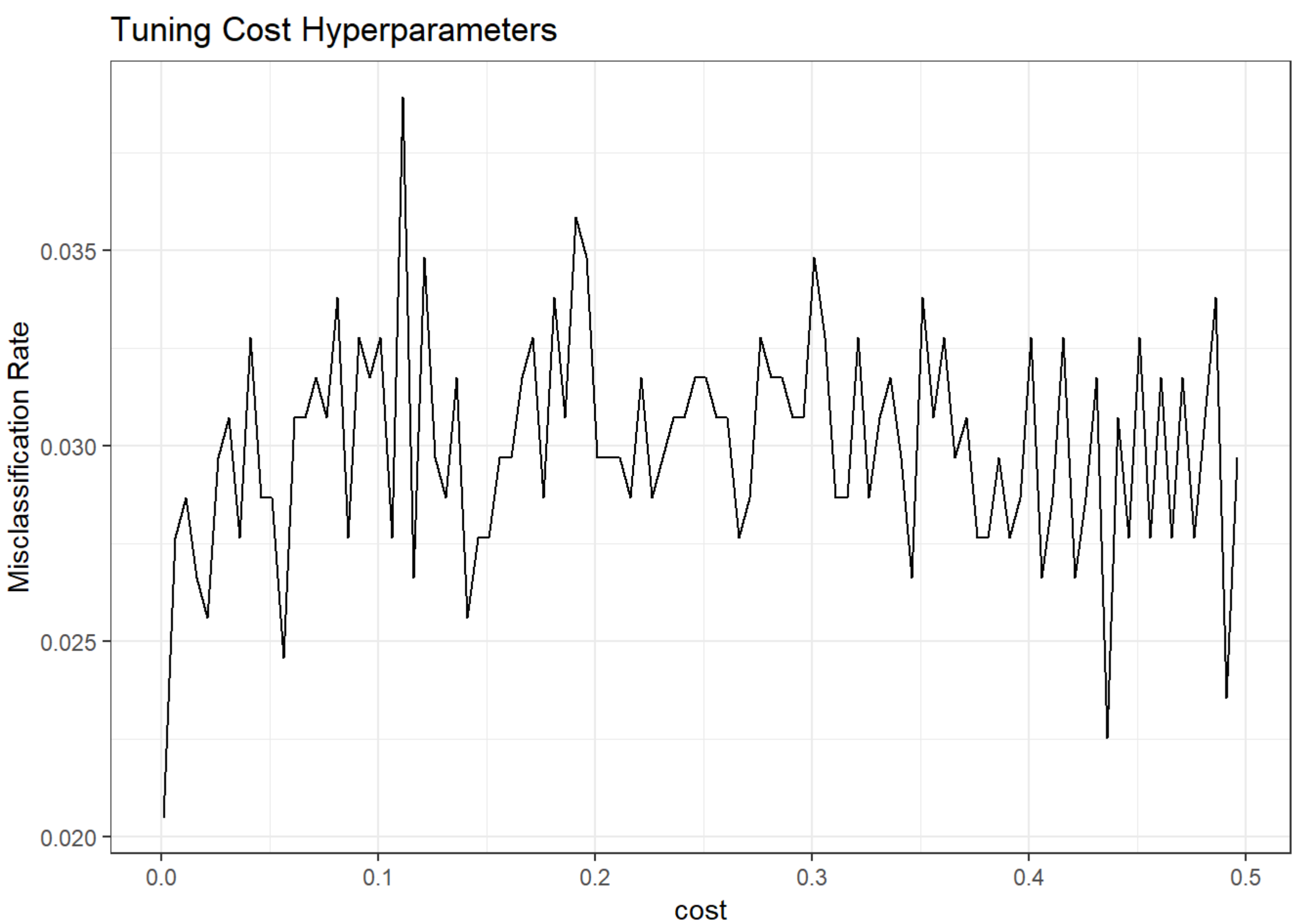
```
## Dimension of Training set:  976 257
```

```
cat("Dimension of Test set: ", dim(test))
```

```
## Dimension of Test set:  244 257
```

Training linear SVM, and tuning hyperparameters by 10-fold cross validation

```
cost = seq(0.001,0.5, 0.005)
mis.rate = c()

for(i in cost)
{
  model1 = svm(y ~., train, type = "C-classification", kernel = "linear", cost = i, cross = 10)
  mis.rate = c(mis.rate, 1-summary(model1)$"tot.accuracy"/100)
}

ggplot()+
  geom_line(aes(cost, mis.rate), size = 0.5)+
  ylab("Misclassification Rate")+
  ggtitle("Tuning Cost Hyperparameters")+
  theme_bw()
```




```
cat("The best hyperparameter of cost is:", cost[which.min(mis.rate)])
```

```
## The best hyperparameter of cost is: 0.001
```

Using the optimal parameters to refit SVM, and calcating misclassification rate

```
model1.tune = svm(y ~., train, type = "C-classification", kernel = "linear", cost = 0.001)
summary(model1.tune)
```

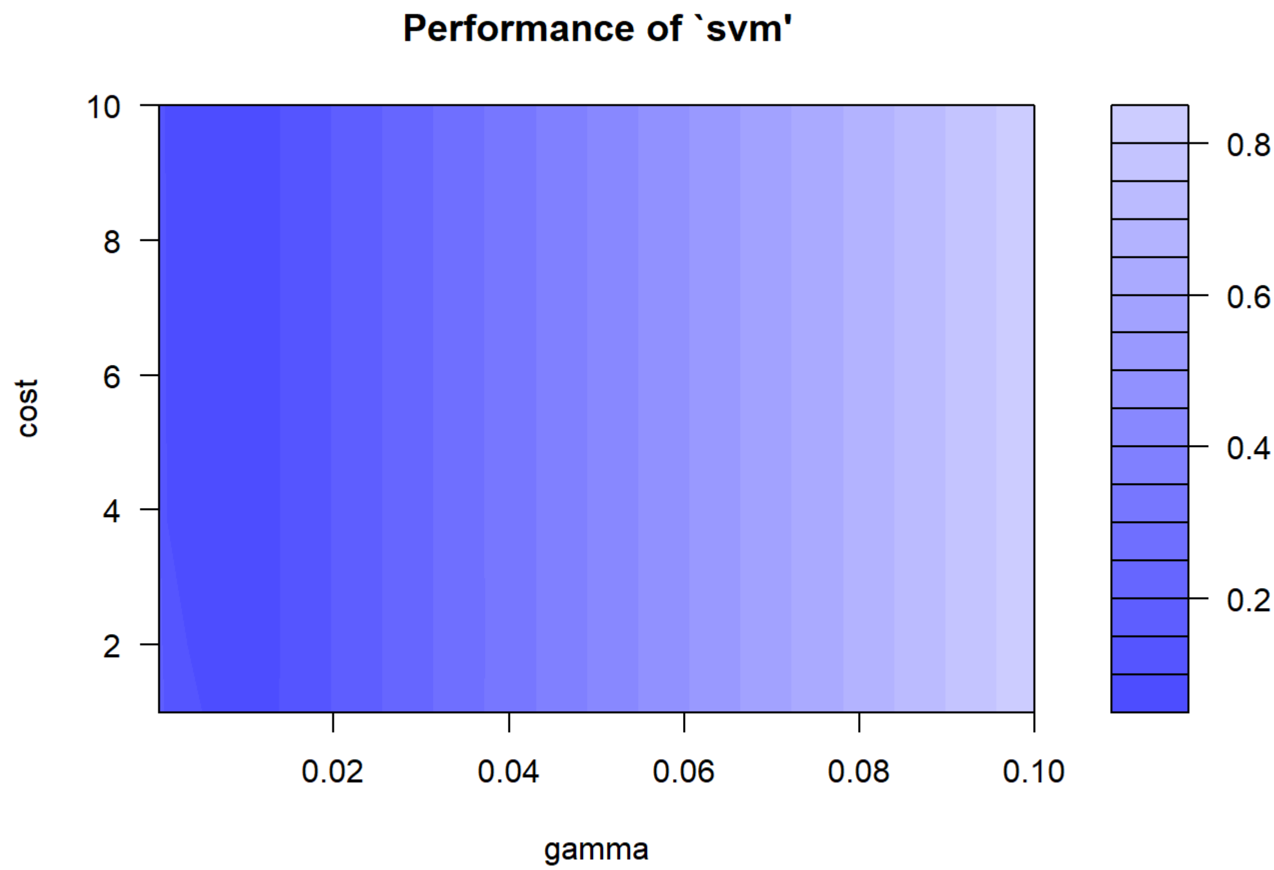
```
##
## Call:
## svm(formula = y ~ ., data = train, type = "C-classification",
##     kernel = "linear", cost = 0.001)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.001
##       gamma:  0.00390625
##
## Number of Support Vectors:  227
##
##  ( 112 115 )
##
##
## Number of Classes:  2
##
## Levels:
##  -1 1
```

```
y.pred = predict(model1.tune, test)
cat("The misclassification rate on the test set:", mean(test$y != y.pred))
```

```
## The misclassification rate on the test set: 0.01639344
```

Training `RBF` kernel SVM, and tuning hyperparameters by 10-fold cross validation

```
cost = seq(1,10,1)
gamma = c(10^(-1:-4))

svm_tune = tune.svm(y ~., data = train, kernel="radial", scale=F, cost = cost, gamma = gamma)

plot(svm_tune)
```




```
summary(svm_tune)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  gamma cost
##   0.01    2
##
## - best performance: 0.06648839
##
## - Detailed performance results:
##    gamma cost      error dispersion
## 1  1e-01    1 0.83758137 0.03407352
## 2  1e-02    1 0.06816863 0.02363817
## 3  1e-03    1 0.12546282 0.03844079
## 4  1e-04    1 0.20235822 0.04187338
## 5  1e-01    2 0.83758120 0.03407309
## 6  1e-02    2 0.06648839 0.02313457
## 7  1e-03    2 0.11132736 0.03674286
## 8  1e-04    2 0.17556925 0.04072827
## 9  1e-01    3 0.83758120 0.03407309
## 10 1e-02    3 0.06669279 0.02320744
## 11 1e-03    3 0.10439686 0.03655420
## 12 1e-04    3 0.16535357 0.04045214
## 13 1e-01    4 0.83758120 0.03407309
## 14 1e-02    4 0.06677812 0.02318854
## 15 1e-03    4 0.09903875 0.03612435
## 16 1e-04    4 0.16047787 0.04096380
## 17 1e-01    5 0.83758120 0.03407309
## 18 1e-02    5 0.06677812 0.02318854
## 19 1e-03    5 0.09525116 0.03563447
## 20 1e-04    5 0.15758079 0.04199609
## 21 1e-01    6 0.83758120 0.03407309
## 22 1e-02    6 0.06677812 0.02318854
## 23 1e-03    6 0.09253450 0.03504611
## 24 1e-04    6 0.15575222 0.04173929
## 25 1e-01    7 0.83758120 0.03407309
## 26 1e-02    7 0.06677812 0.02318854
## 27 1e-03    7 0.09078426 0.03446791
## 28 1e-04    7 0.15434630 0.04153936
## 29 1e-01    8 0.83758120 0.03407309
## 30 1e-02    8 0.06677812 0.02318854
## 31 1e-03    8 0.08945378 0.03390678
## 32 1e-04    8 0.15311475 0.04164349
## 33 1e-01    9 0.83758120 0.03407309
## 34 1e-02    9 0.06677812 0.02318854
## 35 1e-03    9 0.08830707 0.03351104
## 36 1e-04    9 0.15205801 0.04175446
## 37 1e-01   10 0.83758120 0.03407309
## 38 1e-02   10 0.06677812 0.02318854
## 39 1e-03   10 0.08725219 0.03301034
## 40 1e-04   10 0.15123158 0.04184559
```

```
cat("The best hyperparameter of cost =", svm_tune$best.parameters[1,1], "Gamma = ", svm_tune$best.paramete
rs[1,2])
```

```
## The best hyperparameter of cost = 0.01 Gamma =  2
```

Using the optimal parameters to refit SVM, and calcating misclassification rate

```
model.tune = svm(y ~., train, type = "C-classification", kernel = "radial", cost = 2, gamma = 0.01)
summary(model.tune)
```

```
##
## Call:
## svm(formula = y ~ ., data = train, type = "C-classification",
##     kernel = "radial", cost = 2, gamma = 0.01)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  2
##       gamma:  0.01
##
## Number of Support Vectors:  514
##
##  ( 299 215 )
##
##
## Number of Classes:  2
##
## Levels:
##  -1 1
```

```
y.pred = predict(model.tune, test)
cat("The misclassification rate on the test set: ", mean(test$y != y.pred))
```

```
## The misclassification rate on the test set:  0.01229508
```

For conclusion, the `RBF` kernel SVM has better performance on the test set, which misclassification rate is 0.0123. And the hpyerparameter I selected is $Cost = 2, Gamma = 0.01$