Assignment 4

Group 11

Chang Zhou 01983512, Qian Zhang 01939418, Yutong Zheng 01895402

2021/5/30

Question 1 First of all, the estimates can be biased if the 10,000 data points are not selected randomly. Moreover, compared to the whole data set (50,000 points), the original estimated parameters using only 10,000 points may not perform well in the new fitting environment. Considering the penalty term $C\sum_{i=1}^{n}\xi_i$, the value of C needs to be adjusted appropriately. As the number of training sets increases, C as a penalty coefficient should be reduced as far as possible to weaken the impact of over fitting, and the value of σ should also relatively decrease the adaptability of the increasing model.

Question 2

(a)

```
library(mlbench)
library(e1071)
library(caret)
```

```
data(BreastCancer)
dim(BreastCancer)
```

```
## [1] 699  11
```

```
data<- BreastCancer[,-1]
data<-na.omit(data)
summary(data)
```

```
##   Cl.thickness   Cell.size     Cell.shape  Marg.adhesion  Epith.c.size
##  1      :139   1      :373   1      :346   1      :393   2      :376
##  5      :128   10     : 67   2      : 58   2      : 58   3      : 71
##  3      :104   3      : 52   10     : 58   3      : 58   4      : 48
##  4      : 79   2      : 45   3      : 53   10     : 55   1      : 44
##  10     : 69   4      : 38   4      : 43   4      : 33   6      : 40
##  2      : 50   5      : 30   5      : 32   8      : 25   5      : 39
##  (Other):114   (Other): 78   (Other): 93   (Other): 61   (Other): 65
##   Bare.nuclei   Bl.cromatin  Normal.nucleoli    Mitoses          Class
##  1      :402   3      :161   1      :432     1      :563   benign   :444
##  10     :132   2      :160   10     : 60     2      : 35   malignant:239
##  2      : 30   1      :150   3      : 42     3      : 33
##  5      : 30   7      : 71   2      : 36     10     : 14
##  3      : 28   4      : 39   8      : 23     4      : 12
##  8      : 21   5      : 34   6      : 22     7      :  9
##  (Other): 40   (Other): 68   (Other): 68     (Other): 17
```

There are 699 samples in total and 16 objects having NA values in the independent variable "Bare.nuclei", therefore leaving a data-set with 683 data-points. Among these 683 samples, there are 444 benign samples

and 239 malignant samples.

(b)

```
set.seed(1)
sub<-sample(1:nrow(data),round(nrow(data)*0.75))
train<-data[sub,]
test<-data[-sub,]
tune.out = tune(svm,Class~.,data=train,kernel="linear",
                ranges=list(cost=c(0.001, 0.01, 0.1, 1,5,10,100)))
# summary(tune.out)
bestmod = tune.out$best.model
summary(bestmod)
```

```
##
## Call:
## best.tune(method = svm, train.x = Class ~ ., data = train, ranges = list(cost = c(0.001,
##     0.01, 0.1, 1, 5, 10, 100)), kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.1
##
## Number of Support Vectors:  88
##
##  ( 40 48 )
##
##
## Number of Classes:  2
##
## Levels:
##  benign malignant
```

The best Cost is 0.1.

```
svmfit = svm(Class~., data=train, kernel="linear", cost=0.1)
ypred = predict(svmfit,test[1:9])
table(predict=ypred, truth=test$Class)
```

```
##            truth
## predict     benign malignant
##   benign       111         4
##   malignant      5        51
```

Thus, with this value of cost, 162 of the test observations are correctly classified. The precision of predicting malignant is $\frac{51}{51+5} = 0.91$, the recall is $\frac{51}{51+4} = 0.927$.

(c)

```
tune.out = tune(svm,Class~.,data=train,kernel="polynomial",degree=2,
                ranges=list(cost=c(0.001, 0.01, 0.1, 1,5,10,100)),
                coef0=c(0, 0.5, 1, 1.5, 2, 2.5))
summary(tune.out) # cost: 100
```

```
##
## Parameter tuning of 'svm':
```

```
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   100
##
## - best performance: 0.0331448
##
## - Detailed performance results:
##    cost      error dispersion
## 1 1e-03 0.35961538 0.07720271
## 2 1e-02 0.35961538 0.07720271
## 3 1e-01 0.35961538 0.07720271
## 4 1e+00 0.35961538 0.07720271
## 5 5e+00 0.06651584 0.06287485
## 6 1e+01 0.05852187 0.03777136
## 7 1e+02 0.03314480 0.02243120
```

```
bestmod = tune.out$best.model
summary(bestmod)
```

```
##
## Call:
## best.tune(method = svm, train.x = Class ~ ., data = train, ranges = list(cost = c(0.001,
##     0.01, 0.1, 1, 5, 10, 100)), kernel = "polynomial", degree = 2,
##     coef0 = c(0, 0.5, 1, 1.5, 2, 2.5))
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  100
##      degree:  2
##      coef.0:  0 0.5 1 1.5 2 2.5
##
## Number of Support Vectors:  184
##
##  ( 62 122 )
##
##
## Number of Classes:  2
##
## Levels:
##  benign malignant
```

```
svmfit = svm(Class~., data=train, kernel="polynomial", cost=100, degree = 2)
ypred = predict(svmfit,test[1:9])
table(predict=ypred, truth=test$Class)
```

```
##            truth
## predict     benign malignant
##   benign       110         4
##   malignant      6        51
```

Thus, with the cost 100, 161 of the test observations are correctly classified. The precision of predicting

malignant is $\frac{51}{51+6} = 0.89$, the recall is $\frac{51}{51+4} = 0.927$.

(d)

```
tune.out = tune(svm,Class~.,data=train,kernel="polynomial",degree=3,
                ranges=list(cost=c(1,5,10,100, 110, 120, 150, 160, 170, 180)),
                coef0=c(0, 0.5, 1, 1.5, 2, 2.5))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   150
##
## - best performance: 0.03702866
##
## - Detailed performance results:
##    cost      error dispersion
## 1     1 0.35923831 0.09319585
## 2     5 0.35923831 0.09319585
## 3    10 0.35923831 0.09319585
## 4   100 0.27707391 0.13934453
## 5   110 0.20674962 0.14963686
## 6   120 0.13058069 0.11911982
## 7   150 0.03702866 0.02948194
## 8   160 0.04679487 0.02918554
## 9   170 0.05071644 0.02772469
## 10  180 0.04879336 0.02472780
```

```
bestmod = tune.out$best.model
summary(bestmod)
```

```
##
## Call:
## best.tune(method = svm, train.x = Class ~ ., data = train, ranges = list(cost = c(1,
##     5, 10, 100, 110, 120, 150, 160, 170, 180)), kernel = "polynomial",
##     degree = 3, coef0 = c(0, 0.5, 1, 1.5, 2, 2.5))
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  150
##      degree:  3
##      coef.0:  0 0.5 1 1.5 2 2.5
##
## Number of Support Vectors:  366
##
##  ( 184 182 )
##
##
## Number of Classes:  2
```

```
##
## Levels:
##  benign malignant
```

```
svmfit = svm(Class~., data=train, kernel="polynomial", cost=150, degree = 3)
ypred = predict(svmfit,test[1:9])
table(predict=ypred, truth=test$Class)
```

```
##            truth
## predict     benign malignant
##   benign       104         1
##   malignant     12        54
```

Thus, with the cost 150, 157 of the test observations are correctly classified. The precision of predicting malignant is $\frac{54}{54+13} = 0.81$, the recall is $\frac{54}{54+1} = 0.98$. If our objective is to detect the malignant samples as many as possible, then using polynomial kernel of degree 3 with cost 150 is a better choice regarding to its high recall.

(e)

```
tune.out=tune(svm, Class~., data=train, kernel="radial",
              ranges=list(cost=c(0.1,1,10,20,30,100),
                          gamma=c(0.5,1,2,3,4)))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##    10   0.5
##
## - best performance: 0.04494721
##
## - Detailed performance results:
##     cost gamma      error dispersion
## 1    0.1   0.5 0.11915535 0.06291600
## 2    1.0   0.5 0.05275264 0.03210149
## 3   10.0   0.5 0.04494721 0.03211840
## 4   20.0   0.5 0.04494721 0.03211840
## 5   30.0   0.5 0.04494721 0.03211840
## 6  100.0   0.5 0.04494721 0.03211840
## 7    0.1   1.0 0.35923831 0.05519610
## 8    1.0   1.0 0.11715686 0.05452860
## 9   10.0   1.0 0.10350679 0.05540613
## 10  20.0   1.0 0.10350679 0.05540613
## 11  30.0   1.0 0.10350679 0.05540613
## 12 100.0   1.0 0.10350679 0.05540613
## 13   0.1   2.0 0.35923831 0.05519610
## 14   1.0   2.0 0.18759427 0.09079807
## 15  10.0   2.0 0.17779035 0.07370383
## 16  20.0   2.0 0.17779035 0.07370383
## 17  30.0   2.0 0.17779035 0.07370383
## 18 100.0   2.0 0.17779035 0.07370383
```

```
## 19   0.1   3.0 0.35923831 0.05519610
## 20   1.0   3.0 0.24030920 0.07908491
## 21  10.0   3.0 0.23442685 0.07623404
## 22  20.0   3.0 0.23442685 0.07623404
## 23  30.0   3.0 0.23442685 0.07623404
## 24 100.0   3.0 0.23442685 0.07623404
## 25   0.1   4.0 0.35923831 0.05519610
## 26   1.0   4.0 0.24419306 0.07586825
## 27  10.0   4.0 0.24419306 0.07586825
## 28  20.0   4.0 0.24419306 0.07586825
## 29  30.0   4.0 0.24419306 0.07586825
## 30 100.0   4.0 0.24419306 0.07586825
```

```
bestmod = tune.out$best.model
summary(bestmod)
```

```
##
## Call:
## best.tune(method = svm, train.x = Class ~ ., data = train, ranges = list(cost = c(0.1,
##     1, 10, 20, 30, 100), gamma = c(0.5, 1, 2, 3, 4)), kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  10
##
## Number of Support Vectors:  285
##
##  ( 102 183 )
##
##
## Number of Classes:  2
##
## Levels:
##  benign malignant
```

```
svmfit = svm(Class~., data=train, kernel="radial", cost=10, degree = 0.5)
ypred = predict(svmfit,test[1:9])
table(predict=ypred, truth=test$Class)
```

```
##           truth
## predict    benign malignant
##   benign      112         3
##   malignant     4        52
```

Thus, with the cost as 10 and gamma as 0.5, 164 of the test observations are correctly classified. The precision of predicting malignant is $\frac{52}{52+4} = 0.93$, the recall is $\frac{52}{52+3} = 0.945$. The Gaussian kernel seems to have the highest accuracy and its ability to classify malignant samples is also good.

Question 4

The ridge regression solves the problem:

$$\min_{\beta}\{\frac{1}{2}\|y - x\beta\|_2^2 + \frac{\lambda}{2}\beta^T\beta\}$$

and it has the optimal solution:

$\widehat{\beta_R} = (X^TX + \lambda I_d)^{-1}X^Ty$

$= \ (\phi^T(X)\phi(X) + \lambda I_M)^{-1}\phi^T(X)y \qquad (\phi(X) \in R^{n\times M}, X \in R^{n\times d})$

Here we've replaced $x_i\ (d \times 1)$ with the feature vector $\phi(x_i)\ (M \times 1),\ i = 1, \dots, n$

Let $B = \phi(X), P = \ (\lambda I_M)^{-1}, R = \ I_n$

$\widehat{\beta_R}$

$= (\phi^T(X)\phi(X) + \lambda I_M)^{-1}\phi^T(X)y$

$= \ (\phi^T(X)I_n\phi(X) + \lambda I_M)^{-1}\phi^T(X)I_ny \qquad (AI = A)$

$= \ (B^TRB + P^{-1})^{-1}B^TRy$

$= \ (B^TRB + P^{-1})^{-1}B^TR^{-1}y \qquad (I^{-1} = I)$

$= \ PB^T(BPB^T + R)^{-1}y \qquad ((P^{-1} + B^TR^{-1}B)^{-1}B^TR^{-1} = PB^T(BPB^T + R)^{-1})$

$= \ (\lambda I_M)^{-1}\phi^T(X)(\phi(X)(\lambda I_M)^{-1}\phi^T(X) + I_n)^{-1}y$

$= \ \frac{1}{\lambda}I_M\phi^T(X)\left(\phi(X)\frac{1}{\lambda}I_M\phi^T(X) + I_n\right)^{-1}y \qquad ((\lambda I_M)^{-1} = \ (\lambda)^{-1}(I_M)^{-1} = \frac{1}{\lambda}I_M)$

$= \ \frac{1}{\lambda}\phi^T(X)\left(\phi(X)\frac{1}{\lambda}I_M\phi^T(X) + I_n\right)^{-1}y \qquad (IA = A)$

$= \ \phi^T(X)(\phi(X)\phi^T(X) + \lambda I_n)^{-1}y \qquad (\widehat{\beta_R} \in R^{M\times 1})$

Given x_{new},

y_{new}

$= \ \widehat{\beta_R}^{\mathrm{T}}\ \phi(x_{new})$

$= \ y^T(\phi(X)\phi^T(X) + \lambda I_n)^{-1}\phi^T(X)\phi(x_{new})$

Define the Gram matrix K to be the $n \times n$ matrix with

$K_{ij} := \ \phi(x_i)^T\phi(x_j)$

$:= \ k(x_i, x_j)$

y_{new} will then equal to $y^T(K+\lambda I_n)^{-1}\phi^T(X)\phi(x_{new})$

Since $\phi^T(X)\phi(x_{new})$ produces $(n \times 1)$ vector with the $i - th$ element equal to $k(x_i, x^{new})$, we just need the kernel function but not the feature vector to estimate y_{new}.