Homework 4 (Part 1)

Ying-Yu Huang (yingyu010365@gmail.com)

Question 1 : Generate a Data Set by Simulations

Step 1

```
> A
           [,1]       [,2]        [,3]       [,4]
[1,] -0.5312555 -0.5649478  0.05626874 -0.4443732
[2,] -0.8535393  1.0406786  1.81023079 -1.4159107
[3,] -0.9049987  1.0570002  0.35927503 -1.0292485
[4,] -1.0900022 -0.2154551 -1.05583049 -0.2402299
> B
[1]  1.3336832  0.2670032  0.4160923 -1.9653632
> C
[1] -1.161517
```

Step 2: Keep only 5000 cases of the generated data

```
> head(data)
          x1         x2          x3         x4         U  y
1 -0.3522819  1.1240544 -1.15163178 -0.8311079 -1.929165 -1
2 -1.6018695 -0.6775253 -1.41564195  0.2092816 -2.902350 -1
3  0.5346514  0.7972983  0.02893215  1.2319033 -4.634714 -1
4 -0.8914464  1.8938125  1.20765148  0.7618827  8.594368  1
5 -1.0079545 -1.6822697  1.35814524 -0.3054582 -5.880438 -1
6 -1.6258220  1.8846521 -0.12049491  1.8643487 -1.058891 -1
```

Center and Rescale this data

```
> head(new_data)
          x1         x2          x3         x4  y
1 -0.3069243  0.9422528 -1.00511924 -0.6918174 -1
2 -1.3932685 -0.6158460 -1.23499904  0.2109401 -1
3  0.4641420  0.6596573  0.02282529  1.0982802 -1
4 -0.7756535  1.6079792  1.04916367  0.6904382  1
5 -0.8769412 -1.4848007  1.18020209 -0.2357053 -1
6 -1.4140919  1.6000568 -0.10728434  1.6470601 -1
```

Then Split each class into a training set and a test set, using the proportions 80% and 20%. Then get TRAIN and TEST of resp. sizes 4000 and 1000.

TRAIN	4000 obs. of 5 variables
TEST	1000 obs. of 5 variables

Q2: SVM classification by linear kernel

```
Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  5
Number of Support Vectors:  2800
 ( 1401 1399 )
Number of Classes:  2
Levels:
 -1 1
```

Use the svm() function, for instance cost = 5 and kernel = "linear ", then get S=2800 support vectors, and the ratio s = S/4000=70%

Confusion matrices for the training set:

```
            real
predict   -1    1
     -1 1485  693
     1   515 1307
```

Matrix in frequency of correct predictions within each class on training set:

	Real class: -1	Real class: 1
Predict class: -1	74.25%	34.65%
Predict class: 1	25.75%	65.35%

The correct prediction of PredTrain: 69.8%

The errors of estimation on PredTRAIN:

$\sqrt{69.8\%(1-69.8\%)/4000} = 7.26 \times 10^{-3}$

95% confidence interval:

$69.8\% \pm 1.96 \times 7.26 \times 10^{-3}$ = (68.38%,71.22%)

The errors of estimation on class(-1):

$\sqrt{74.25\%(1-74.25\%)/2000} = 9.78 \times 10^{-3}$

95% confidence interval:

$74.25\% \pm 1.96 \times 9.78 \times 10^{-3}$ = (72.33%,76.17%)

The errors of estimation on class(1):

$\sqrt{65.35\%(1-65.35\%)/2000} = 0.011$

95% confidence interval:

$65.35\% \pm 1.96 \times 0.011$ = (63.19%,67.51%)

Confusion matrices for the test set:

```
     predict
real  -1   1
  -1 358 142
  1  179 321
```

Matrix in frequency of correct predictions within each class on test set:

	Real class: -1	Real class: 1
Predict class: -1	71.6%	35.8%
Predict class: 1	28.4%	64.2%

The correct prediction of PredTest is 67.9%

The errors of estimation on PredTEST:

$\sqrt{67.9\%(1-67.9\%)/1000}$ =0.015

95% confidence interval:

67.9%$\pm$1.96 $\times$ 0.015 = (64.96%,70.84%)

The errors of estimation on class(-1):

$\sqrt{71.6\%(1-71.6\%)/500}$ = 0.02

95% confidence interval:

71.6%$\pm$ 1.96 $\times$0.02= (67.68%,75.52%)

The errors of estimation on class(1):

$\sqrt{64.2\%(1-64.2\%)/500}$ = 0.021

95% confidence interval:

64.2% $\pm$ 1.96 $\times$0.021 = (60.08%,68.32%)

Interpretation:

The performance of SVM function with ‘kernel = linear’ and ‘cost = 5’ is not quite good: 95% confidence interval for TRAIN is (68.38%,71.22%) and 95% confidence interval for TEST is (64.96%,70.84%).

Within each class, for both TRAIN and TEST set, the 95% confidence interval are on the left side of 72%.

So, we need to modify the parameters in SVM function.

Q3: optimize the parameter "cost"

```
Parameter tuning of ‘svm’:
- sampling method: 10-fold cross validation
- best parameters:
 cost
 0.01
- best performance: 0.30425
```

```
- Detailed performance results:
   cost   error dispersion
1 1e-03 0.30875 0.02533799
2 1e-02 0.30425 0.02779014
3 1e-01 0.30425 0.02828550
4 1e+00 0.30425 0.02879646
5 5e+00 0.30425 0.02879646
6 1e+01 0.30425 0.02879646
```

Select a list of 6 values for the "cost " parameter: cost=c (0.001,0.01,0.1,1,5,10)
When the cost=0.01, we can get the best performance: error=0.30425

Use the svm() function, for instance cost = 0.01 and kernel = "linear ", then get S=2899 support vectors, and the ratio s = S/4000=72.48%

```
Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  0.01
Number of Support Vectors:  2899
 ( 1450 1449 )
Number of Classes:  2
Levels:
 -1 1
```

Confusion matrices for the training set:

```
           real
predict   -1    1
     -1 1484  694
     1   516 1306
```

Matrix in frequency of correct predictions within each class on training set:

	Real class: -1	Real class: 1
Predict class: -1	74.2%	34.7%
Predict class: 1	25.8%	65.3%

The correct prediction of PredTrain: 69.75%
The errors of estimation on PredTRAIN:

$$\sqrt{69.75\%(1-69.75\%)/4000} = 7.26 \times 10^{-3}$$

95% confidence interval:

$69.75\% \pm 1.96 \times 7.26 \times 10^{-3}$ = (68.33%,71.17%)

The errors of estimation on class(-1):

$\sqrt{74.2\%(1-74.2\%)/2000}$ = 9.78×10^{-3}

95% confidence interval:

$74.2\% \pm 1.96 \times 9.78 \times 10^{-3}$ = (72.28%,76.12%)

The errors of estimation on class(1):

$\sqrt{65.3\%(1-65.3\%)/2000}$ = 0.011

95% confidence interval:

$65.3\% \pm 1.96 \times 0.011$ = (63.14%,67.46%)

Confusion matrices for the test set:

```
      predict
real  -1   1
  -1 357 143
  1  179 321
```

Matrix in frequency of correct predictions within each class on test set:

	Real class: -1	Real class: 1
Predict class: -1	71.4%	35.8%
Predict class: 1	28.6%	64.2%

The correct prediction of PredTest is 67.8%

The errors of estimation on PredTEST:

$\sqrt{67.8\%(1-67.8\%)/1000}$ = 0.015

95% confidence interval:

$67.8\% \pm 1.96 \times 0.015$ = (64.86%,70.74%)

The errors of estimation on class(-1):

$\sqrt{71.4\%(1-71.4\%)/500}$ = 0.02

95% confidence interval:

$71.4\% \pm 1.96 \times 0.02$= (67.48%,75.32%)

The errors of estimation on class(1):

$\sqrt{64.2\%(1-64.2\%)/500}$ = 0.021

95% confidence interval:

$64.2\% \pm 1.96 \times 0.021$ = (60.08%,68.32%)

Interpretation:

After fix the cost = 0.01, 95% confidence interval for TRAIN is (68.33%,71.17%) and 95% confidence interval for TEST is (64.86%,70.74%). Within each class, for both TRAIN and TEST set, the 95% confidence interval are still on the left side of 72%.

The performance of 'linear' kernel doesn't change much. So, the 'linear' kernel won't do a good job.

Q4: SVM classification by radial kernel

```
Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  0.01
Number of Support Vectors:  3783
 ( 1891 1892 )
Number of Classes:  2
Levels:
 -1 1
```

Use the svm() function, for instance cost = 0.01 and kernel = "radial", then get S=3783 support vectors, and the ratio s = S/4000= 94.58%

Confusion matrices for the training set:

```
    predict
real   -1    1
  -1 1957   43
  1   193 1807
```

Matrix in frequency of correct predictions within each class on training set:

	Real class: -1	Real class: 1
Predict class: -1	97.85%	9.65%
Predict class: 1	2.15%	90.35%

The correct prediction of PredTrain: 94.1%

The errors of estimation on PredTRAIN:

$\sqrt{94.1\%(1-94.1\%)/4000} = 3.73 \times 10^{-3}$

95% confidence interval:

$94.1\% \pm 1.96 \times 3.73 \times 10^{-3}$ = (93.37%,94.83%)

The errors of estimation on class(-1):

$\sqrt{97.85\%(1-97.85\%)/2000} = 3.24 \times 10^{-3}$

95% confidence interval:

$97.85\% \pm 1.96 \times 3.24 \times 10^{-3}$ = (97.21%,98.49%)

The errors of estimation on class(1):

$\sqrt{90.35\%(1-90.35\%)/2000} = 6.6 \times 10^{-3}$

95% confidence interval:

$90.35\% \pm 1.96 \times 6.6 \times 10^{-3}$ = (89.06%,91.64%)

Confusion matrices for the test set:

```
     predict
real  -1   1
  -1 483  17
  1   49 451
```

Matrix in frequency of correct predictions within each class on test set:

	Real class: -1	Real class: 1
Predict class: -1	96.6%	9.8%
Predict class: 1	3.4%	90.2%

The correct prediction of PredTest is 93.4%

The errors of estimation on PredTEST:

$\sqrt{93.4\%(1-93.4\%)/1000}$ = 7.85×10^{-3}

95% confidence interval:

$93.4\% \pm 1.96 \times 7.85 \times 10^{-3}$ = (91.86%,94.94%)

The errors of estimation on class(-1):

$\sqrt{96.6\%(1-96.6\%)/500}$ =8.1×10^{-3}

95% confidence interval:

$96.6\% \pm 1.96 \times 8.1 \times 10^{-3}$ = (95.01%,98.19%)

The errors of estimation on class(1):

$\sqrt{90.2\%(1-90.2\%)/500}$ =0.013

95% confidence interval:

$90.2\% \pm 1.96 \times 0.013$ = (87.65%,92.75%)

Interpretation:

The performance of SVM function with ‘kernel = radial’ and ‘cost = 0.01’ and ‘gamma=1’ is quite good: 95% confidence interval for TRAIN is (93.37%,94.83%) and 95% confidence interval for TEST is (91.86%,94.94%).

Within each class, for both TRAIN and TEST set, the 95% confidence interval are on the right side of 91%.

So, radial kernel is doing a better job than linear kernel.

Question 5: optimize the parameter "cost" and "gamma"

Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation
- best parameters:

```
  cost  gamma
  1000    0.1
- best performance: 0.009
- Detailed performance results:
     cost  gamma    error   dispersion
1    1e-01  1e-02  0.27100  0.021285102
2    1e+00  1e-02  0.15675  0.017242148
3    1e+01  1e-02  0.05650  0.012812754
4    1e+02  1e-02  0.02100  0.009294562
5    1e+03  1e-02  0.01000  0.003726780
6    1e-01  1e-01  0.07000  0.014092945
7    1e+00  1e-01  0.02700  0.009264628
8    1e+01  1e-01  0.01375  0.004894725
9    1e+02  1e-01  0.01150  0.004743416
10   1e+03  1e-01  0.00900  0.004743416
11   1e-01  1e+00  0.03525  0.008775756
12   1e+00  1e+00  0.02800  0.008147938
13   1e+01  1e+00  0.02250  0.009354143
14   1e+02  1e+00  0.02350  0.010013879
15   1e+03  1e+00  0.02400  0.008913161
16   1e-01  1e+01  0.47600  0.134170290
17   1e+00  1e+01  0.04625  0.011008204
18   1e+01  1e+01  0.04700  0.011105554
19   1e+02  1e+01  0.04700  0.011105554
20   1e+03  1e+01  0.04700  0.011105554
21   1e-01  1e+02  0.50750  0.037006006
22   1e+00  1e+02  0.38475  0.064919287
23   1e+01  1e+02  0.36800  0.062303023
24   1e+02  1e+02  0.36800  0.062303023
25   1e+03  1e+02  0.36800  0.062303023
```

Select a list of 5 values for the "cost " parameter and a list of 5 values for the parameter "gamma"

cost=c(0.1,1,10,100,1000),gamma=c(0.01,0.1,1,10,100)

When the cost=1000, gamma=0.1, we can get the best performance: error= 0.009

Use the svm() function, for instance cost = 1000, gamma=0.1 and kernel = "radial", then get S=115 support vectors, and the ratio s = S/4000=2.88%

```
Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1000
Number of Support Vectors:  115
 ( 57 58 )
Number of Classes:  2
Levels:
 -1 1
```

Confusion matrices for the training set:

```
       predict
real   -1    1
  -1 1997    3
  1     3 1997
```

Matrix in frequency of correct predictions within each class on training set:

	Real class: -1	Real class: 1
Predict class: -1	99.85%	0.15%
Predict class: 1	0.15%	99.85%

The correct prediction of PredTrain: 99.85%

The errors of estimation on PredTRAIN:

$\sqrt{99.85\%(1-99.85\%)/4000} = 6.12 \times 10^{-4}$

95% confidence interval:

$99.85\% \pm 1.96 \times 6.12 \times 10^{-4}$ = (99.73%,99.97%)

The errors of estimation on class(-1):

$\sqrt{99.85\%(1-99.85\%)/2000} = 8.65 \times 10^{-4}$

95% confidence interval:

$99.85\% \pm 1.96 \times 8.65 \times 10^{-4}$ = (99.68%,100%)

The errors of estimation on class(1):

$\sqrt{99.85\%(1-99.85\%)/2000} = 8.65 \times 10^{-4}$

95% confidence interval:

$99.85\% \pm 1.96 \times 8.65 \times 10^{-4}$ = (99.68%,100%)

Confusion matrices for the test set:

```
predict
real  -1   1
  -1 494   6
  1    0 500
```

Matrix in frequency of correct predictions within each class on test set:

	Real class: -1	Real class: 1
Predict class: -1	98.8%	0%
Predict class: 1	1.2%	100%

The correct prediction of PredTest is 99.4%

The errors of estimation on PredTEST:

$\sqrt{99.4\%(1-99.4\%)/1000} = 2.44 \times 10^{-3}$

95% confidence interval:

$99.4\% \pm 1.96 \times 2.44 \times 10^{-3}$= (98.92%,99.88%)

The errors of estimation on class(-1):

$\sqrt{98.8\%(1-98.8\%)/500}$ =4.87×10^{-3}

95% confidence interval:

$98.8\% \pm 1.96 \times 4.87 \times 10^{-3}$= (97.85%,99.75%)

The errors of estimation on class(1):

$\sqrt{100\%(1-100\%)/500} = 0$

95% confidence interval:

$100\% \pm 1.96 \times 0$ = (100%,100%)

Interpretation:

After fix the cost = 1000, gamma=0.1, 95% confidence interval for TRAIN is (99.73%,99.97%) and 95% confidence interval for TEST is (98.92%,99.88%). Within each class, for both TRAIN and TEST set, the 95% confidence interval are on the right side of 98%.

The performance of 'radial' kernel gets improved. So, after optimizing the parameters, 'radial' kernel give a very good performance.

Question 6 : SVM classification using a polynomial kernel

```
Parameter tuning of 'svm':
- sampling method: 10-fold cross validation
- best parameters:
 coef0 cost
   100   10
```

```
- best performance: 0.00225
- Detailed performance results:
   coef0  cost   error  dispersion
1  1e-02 1e-01 0.12350 0.015284342
2  1e-01 1e-01 0.04750 0.014433757
3  1e+00 1e-01 0.01950 0.007888106
4  1e+01 1e-01 0.00975 0.007307720
5  1e+02 1e-01 0.00500 0.002886751
6  1e-02 1e+00 0.05875 0.012318121
7  1e-01 1e+00 0.02975 0.010570530
8  1e+00 1e+00 0.01350 0.007283924
9  1e+01 1e+00 0.00650 0.004887626
10 1e+02 1e+00 0.00300 0.003291403
11 1e-02 1e+01 0.05050 0.007888106
12 1e-01 1e+01 0.01575 0.008901467
13 1e+00 1e+01 0.00925 0.005533986
14 1e+01 1e+01 0.00550 0.004684490
15 1e+02 1e+01 0.00225 0.002486072
16 1e-02 1e+02 0.03900 0.009874209
17 1e-01 1e+02 0.01100 0.004887626
18 1e+00 1e+02 0.00575 0.004721405
19 1e+01 1e+02 0.00300 0.003073181
20 1e+02 1e+02 0.00425 0.002371708
21 1e-02 1e+03 0.01875 0.010622957
22 1e-01 1e+03 0.00825 0.004571956
23 1e+00 1e+03 0.00325 0.003545341
24 1e+01 1e+03 0.00350 0.002934469
25 1e+02 1e+03 0.00550 0.004216370
```

Select a list of 5 values for the "cost " parameter and a list of 5 values for the parameter "coef0"

cost=c(0.1,1,10,100,1000), coef0=c(0.01,0.1,1,10,100)

When the cost=10, coef0=100, we can get the best performance: error=0.00225

Use the svm() function, for instance cost = 10, coef0=100, and kernel = " polynomial ", then get S=27 support vectors, and the ratio s = S/4000=0.68%

```
Parameters:
```

```
   SVM-Type:  C-classification
 SVM-Kernel:  polynomial
       cost:  10
     degree:  4
     coef.0:  100
Number of Support Vectors:  27
 ( 15 12 )
Number of Classes:  2
Levels:
 -1 1
```

Confusion matrices for the training set:

```
    predict
real   -1    1
  -1 1992    8
  1     5 1995
```

Matrix in frequency of correct predictions within each class on training set:

	Real class: -1	Real class: 1
Predict class: -1	99.6%	0.25%
Predict class: 1	0.4%	99.75%

The correct prediction of PredTrain: 99.68%

The errors of estimation on PredTRAIN:

$\sqrt{99.68\%(1-99.68\%)/4000} = 8.93 \times 10^{-4}$

95% confidence interval:

$99.68\% \pm 1.96 \times 8.93 \times 10^{-4}$ = (99.5%,99.86%)

The errors of estimation on class(-1):

$\sqrt{99.6\%(1-99.6\%)/2000} = 1.41 \times 10^{-3}$

95% confidence interval:

$99.6\% \pm 1.96 \times 1.41 \times 10^{-3}$ = (99.32%,99.88%)

The errors of estimation on class(1):

$\sqrt{99.75\%(1-99.75\%)/2000} = 1.12 \times 10^{-3}$

95% confidence interval:

$99.75\% \pm 1.96 \times 1.12 \times 10^{-3}$ = (99.53%,99.97%)

Confusion matrices for the test set:

```
    predict
```

```
real  -1   1
  -1 493   7
  1    1 499
```

Matrix in frequency of correct predictions within each class on test set:

	Real class: -1	Real class: 1
Predict class: -1	98.6%	0.2%
Predict class: 1	1.4%	99.8%

The correct prediction of PredTest is 99.2%

The errors of estimation on PredTEST:

$\sqrt{99.2\%(1-99.2\%)/1000} = 2.82 \times 10^{-3}$

95% confidence interval:

$99.2\% \pm 1.96 \times 2.82 \times 10^{-3}$ = (98.65%,99.75%)

The errors of estimation on class(-1):

$\sqrt{98.6\%(1-98.6\%)/500} = 5.25\times 10^{-3}$

95% confidence interval:

$98.6\% \pm 1.96 \times 5.25\times 10^{-3}$= (97.57%,99.63%)

The errors of estimation on class(1):

$\sqrt{99.8\%(1-99.8\%)/500} = 2\times 10^{-3}$

95% confidence interval:

$99.8\% \pm 1.96 \times 2\times 10^{-3}$ = (99.4%,100%)

Interpretation:

After fix the cost = 10, coef0=100, 95% confidence interval for TRAIN is (99.5%,99.86%) and 95% confidence interval for TEST is (98.65%,99.75%). Within each class, for both TRAIN and TEST set, the 95% confidence interval are on the right side of 98%.

So, after optimizing the parameters, 'polynomial' kernel give a very good performance.

Code;

```
#Q1
#step1
A = matrix(runif(16,-2,2), 4, 4)
A
B = runif(4,-2,2)
```

```
B
C = runif(1,-2,2)
C
#Pol(x) = Σi Σj Aij xi xj + Σ i Bi xi + c/20
pol <- function(x){
   sum_A = 0
   sum_B = 0
   for (i in 1:4){
      for (j in 1:4){
         sum_A = sum_A + A[i,j] * x[i] * x[j]
      }
   }
   for (i in 1:4){
      sum_B = sum_B + B[i] * x[i]
   }
   result = sum_A + sum_B + C/20
   return(result)
}

#step2
#10, 000 vectors, each vector has 4 randomly chosen coordinates with values in [-2,
2]
row=10000
col=4
x = matrix(runif(row*col,-2,2), row, col)
#U(n) = Pol(xn)
U <- function(n){
   U_result = pol(x[n,])
   return(U_result)
}

#y(n) = sign[U(n)]
y <- function(n){
   y_result= sign(U(n))
   return(y_result)
}

#keep only 2500 cases in CL(1) and 2500 cases in CL(-1)
```

```
select=2500
select_up=0
select_down=0
data = matrix(NA,select*2, col+2)
for(i in 1:row){
   select_total = select_down + select_up + 1
   if(y(i)==1 && select_up<select) {
      data[select_total,1:col]=x[i,]
      data[select_total,col+1]=U(i)
      data[select_total,col+2]=y(i)
      select_up = select_up+1;
   }else if(y(i)==-1 && select_down<select) {
      data[select_total,1:col]=x[i,]
      data[select_total,col+1]=U(i)
      data[select_total,col+2]=y(i)
      select_down=select_down+1
   }
   if(select_up==select && select_down==select) break
}


dimnames(data) <- list(c(1:(select*2)),c("X1","X2","X3","X4","U","y"))
data = na.omit(data)
data=data.frame(data)
head(data)
#Center and Rescale this data set of size 5000 so that the standardized data set will
have mean = 0 and dispersion =1
library(scales)
cen_data <- scale(data[,1:4])
cen_data <- data.frame(cen_data)
#cen_data <- rescale(data[,1:4], mean = 0, sd = 1)
y = as.factor(data[,6])
new_data <- cbind(cen_data,y)
new_data = data.frame(new_data)
head(new_data)
# Split training and test set using an 80:20 ratio
#defines a training set TRAIN and a test set TEST of resp. sizes 4000 and 1000
CL_PO = subset(new_data,y==1)
```

```
CL_NE = subset(new_data,y==-1)
train = sample(2500,2000)

newtrain=merge(CL_PO[train,],CL_NE[train,],all=T)
TRAIN_data= newtrain[c(1:4)]
TRAIN_labels=newtrain[5]

newtest = merge(CL_PO[-train,],CL_NE[-train,],all=T)
TEST_data = newtest[c(1:4)]
TEST_labels=newtest[5]

TRAIN = cbind(TRAIN_data,TRAIN_labels)
Summary(TRAIN)
TEST = cbind(TEST_data,TEST_labels)

#Question 2: SVM classification by linear kernel
#Run the svm() function on the set TRAIN,kernel = "linear ", cost = 5
library(e1071)
TRAIN_labels$y=as.factor(TRAIN_labels$y)
svmfit=svm(TRAIN_labels$y~ .,data=TRAIN,kernel="linear",cost=5,scale=FALSE)

#compute the number S of support vectors and the ratio s = S/4000
#compute the percentages of correct prediction PredTrain and PredTest on the sets
TRAIN and TEST
train_pred = predict(svmfit, TRAIN)
test_pred = predict(svmfit, TEST)

#confusion matrices must be converted in terms of frequencies of correct predictions
within each class
#confusion matrices for TRAIN
TRAIN_confus.matrix = table(predict=train_pred,real=TRAIN$y)
TRAIN_confus.matrix
#confusion matrices for TEST
TEST_confus.matrix = table(real=TEST$y, predict=test_pred)
TEST_confus.matrix

#compute the errors of estimation on PredTRAIN, PredTEST, and on the terms of the
confusion matrices
```

```
sum(diag(TRAIN_confus.matrix))/sum(TRAIN_confus.matrix)

sum(diag(TEST_confus.matrix))/sum(TEST_confus.matrix)
#Q3 : optimize the parameter "cost"
#Select a list of 6 values for the "cost " parameter
#Run the tuning function tune() for the linear svm() to identify the best value of "cost"

set.seed (1)
tune.out=tune(svm,y~.,data=TRAIN,kernel ="linear",ranges
=list(cost=c(0.001,0.01,0.1,1,5,10)))
summary (tune.out)    #best value of "cost"= 0.01

#Evaluate the performance characteristics of the "best" linear svm as in question 2
svmfit_1=svm(TRAIN_labels$y~ .,data=TRAIN,kernel="linear",cost=0.01,scale=FA
LSE)
summary(svmfit_1)
train_pred_1 = predict(svmfit_1, TRAIN)
test_pred_1 = predict(svmfit_1, TEST)

#confusion matrices must be converted in terms of frequencies of correct predictions
within each class
#confusion matrices for TRAIN
TRAIN_confus.matrix_1 = table(predict=train_pred_1,real=TRAIN$y)
TRAIN_confus.matrix_1
#confusion matrices for TEST
TEST_confus.matrix_1 = table(real=TEST$y, predict=test_pred_1)
TEST_confus.matrix_1

#compute the errors of estimation on PredTRAIN, PredTEST, and on the terms of the
confusion matrices
sum(diag(TRAIN_confus.matrix_1))/sum(TRAIN_confus.matrix_1)

sum(diag(TEST_confus.matrix_1))/sum(TEST_confus.matrix_1)

#Q4: SVM classification by radial kernel
#Fix the "cost" parameter in the svm() function to the best cost value identified in
question 3
#Select the kernel parameter kernel = "radial " which means that the kernel ks given
```

by the formula
#Select arbitrarily the gamma parameter "gamma" = 1
#Run the svm() function on the set TRAIN
svmfit1=svm(TRAIN_labels$y~ .,data=TRAIN,kernel="radial",gamma =1,cost=0.01,scale=FALSE)
summary(svmfit1)
#as in question 2 compute the number S and the ratio s = S/4000

#the percentages of correct predictions PredTrain and PredTest and the two confusion matrices
train_pred1 = predict(svmfit1, TRAIN)
test_pred1 = predict(svmfit1, TEST)

#confusion matrices for TRAIN
TRAIN_confus.matrix1 = table(real=TRAIN$y, predict=train_pred1)
TRAIN_confus.matrix1
#confusion matrices for TEST
TEST_confus.matrix1 = table(real=TEST$y, predict=test_pred1)
TEST_confus.matrix1

sum(diag(TRAIN_confus.matrix1))/sum(TRAIN_confus.matrix1) # 0.94575
sum(diag(TEST_confus.matrix1))/sum(TEST_confus.matrix1) #0.931
#Q5 : optimize the parameter "cost"and "gamma"
#Select a list of 5 values for the "cost " parameter and a list of 5 values for the parameter "gamma"
#On the TRAIN set , run the tuning function tune() for the radial svm() to identify the best value of the pair ("cost", "gamma") among the 25 values you have listed
set.seed (1)
tune.out1=tune(svm ,y~ .,data=TRAIN,kernel ="radial",ranges =list(cost=c(0.1 ,1 ,10 ,100 ,1000),gamma=c(0.01,0.1,1,10,100)))
summary (tune.out1)

#Evaluate the performance characteristics of the "best" radial svm as in question 2
#Interpret your results
svmfit1_1=svm(TRAIN_labels$y~ .,data=TRAIN,kernel="radial",gamma =0.1,cost=1000,scale=FALSE)
summary(svmfit1_1)
train_pred1_1 = predict(svmfit1_1, TRAIN)

```
test_pred1_1 = predict(svmfit1_1, TEST)

#confusion matrices for TRAIN
TRAIN_confus.matrix1_1 = table(real=TRAIN$y, predict=train_pred1_1)
TRAIN_confus.matrix1_1
#confusion matrices for TEST
TEST_confus.matrix1_1 = table(real=TEST$y, predict=test_pred1_1)
TEST_confus.matrix1_1

sum(diag(TRAIN_confus.matrix1_1))/sum(TRAIN_confus.matrix1_1)
sum(diag(TEST_confus.matrix1_1))/sum(TEST_confus.matrix1_1)

#Q6 : SVM classification using a polynomial kernel
#Implement the steps of question 4 and 5 for the svm() function based on the
polynomial kernel
#K(x,y) = (a + <x,y>)^4
#You will have to optimize the choice of the two parameters "a" >0 and "cost"
svmfit2=svm(TRAIN_labels$y~ .,data=TRAIN,kernel="polynomial",coef0=1,degree
=4,cost=0.01,scale=FALSE)
summary(svmfit2)

train_pred2 = predict(svmfit2, TRAIN)
test_pred2 = predict(svmfit2, TEST)

#confusion matrices for TRAIN
TRAIN_confus.matrix2 = table(real=TRAIN$y, predict=train_pred2)
TRAIN_confus.matrix2
#confusion matrices for TEST
TEST_confus.matrix2 = table(real=TEST$y, predict=test_pred2)
TEST_confus.matrix2

sum(diag(TRAIN_confus.matrix2))/sum(TRAIN_confus.matrix2)
sum(diag(TEST_confus.matrix2))/sum(TEST_confus.matrix2)

set.seed (1)
tune.out2=tune(svm ,y~ .,data=TRAIN,kernel ="polynomial",ranges
=list(coef0=c(0.01,0.1,1,10,100),cost=c(0.1 ,1 ,10 ,100 ,1000)))
summary (tune.out2)
```

```
svmfit2_1=svm(TRAIN_labels$y~ .,data=TRAIN,kernel="polynomial",coef0=100,de
gree=4,cost=10,scale=FALSE)
summary(svmfit2_1)

train_pred2_1 = predict(svmfit2_1, TRAIN)
test_pred2_1 = predict(svmfit2_1, TEST)

#confusion matrices for TRAIN
TRAIN_confus.matrix2_1 = table(real=TRAIN$y, predict=train_pred2_1)
TRAIN_confus.matrix2_1
#confusion matrices for TEST
TEST_confus.matrix2_1 = table(real=TEST$y, predict=test_pred2_1)
TEST_confus.matrix2_1

sum(diag(TRAIN_confus.matrix2_1))/sum(TRAIN_confus.matrix2_1)
sum(diag(TEST_confus.matrix2_1))/sum(TEST_confus.matrix2_1)
```