서포트벡터머신을 위한 보충 R 코드와 결과

이재용, 임요한

서울대학교
통계학과

Contents

1 준비

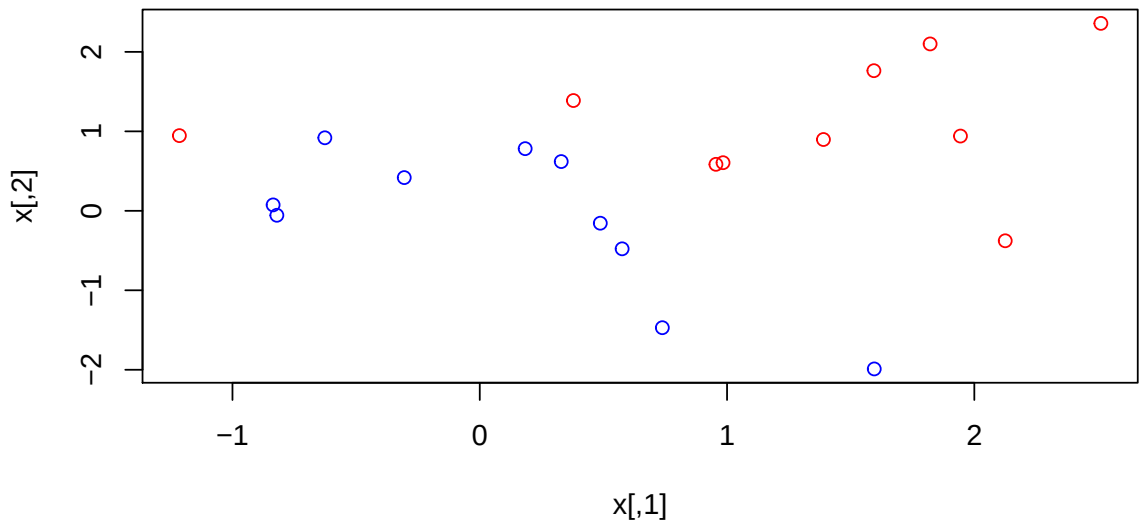
1.1 전역옵션들

```
opts_chunk$set(eval=TRUE, cache=TRUE, fig.width=7, fig.height=4)
```

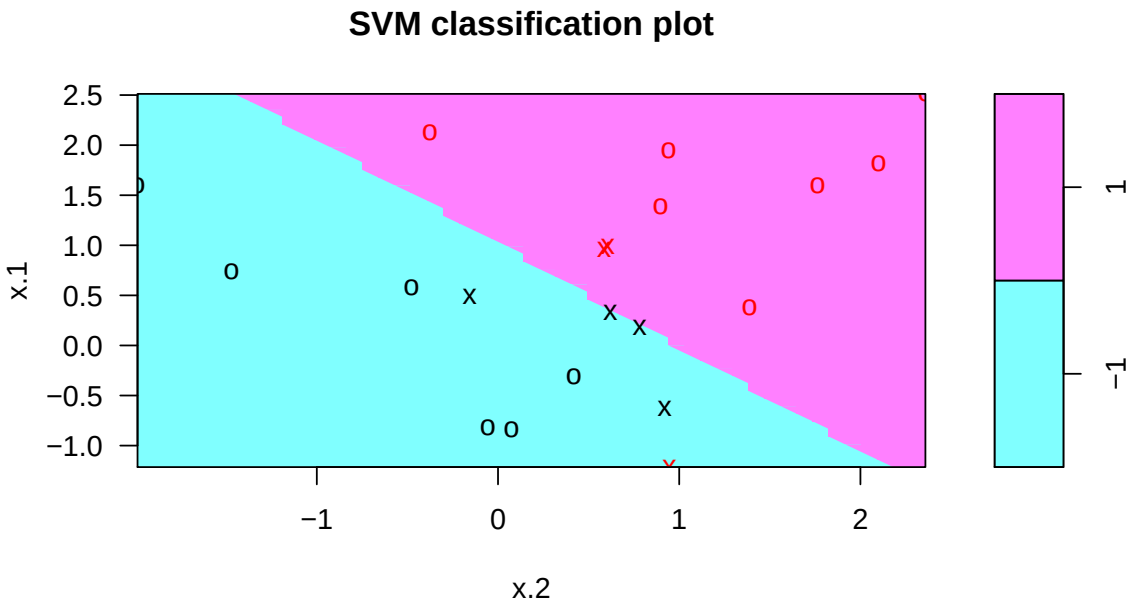
1.2 서포트벡터 분류

```
# Support Vector Classifier


set.seed(1)
x=matrix(rnorm(20*2), ncol=2)
y=c(rep(-1,10), rep(1,10))
```

```r
x[y==1,]=x[y==1,] + 1
plot(x, col=(3-y))
```



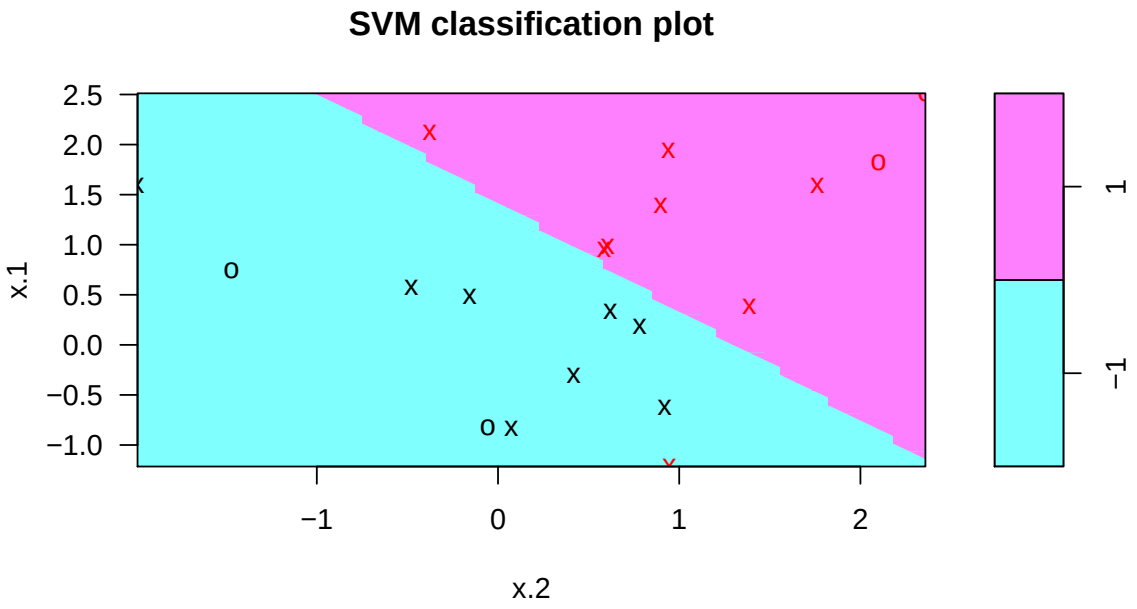

```r
dat=data.frame(x=x, y=as.factor(y))
library(e1071)
svmfit=svm(y~., data=dat, kernel="linear", cost=10,scale=FALSE)
plot(svmfit, dat)
```




```
svmfit$index

## [1]  1  2  5  7 14 16 17

summary(svmfit)

##
## Call:
## svm(formula = y ~ ., data = dat, kernel = "linear", cost = 10,
##     scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  10
##       gamma:  0.5
##
## Number of Support Vectors:  7
##
##  ( 4 3 )
```

```
##
##
## Number of Classes:  2
##
## Levels:
##  -1 1
```

```
svmfit=svm(y~., data=dat, kernel="linear", cost=0.1,scale=FALSE)
plot(svmfit, dat)
```




```
svmfit$index
```

```
##  [1]  1  2  3  4  5  7  9 10 12 13 14 15 16 17 18 20
```

```
set.seed(1)
tune.out=tune(svm,y~.,data=dat,kernel="linear",ranges=list(cost=c(0.001, 0.01, 0.1, 1,5,10,100)))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
```

```
##
## - best parameters:
##  cost
##   0.1
##
## - best performance: 0.1
##
## - Detailed performance results:
##    cost error dispersion
## 1 1e-03  0.70     0.4216
## 2 1e-02  0.70     0.4216
## 3 1e-01  0.10     0.2108
## 4 1e+00  0.15     0.2415
## 5 5e+00  0.15     0.2415
## 6 1e+01  0.15     0.2415
## 7 1e+02  0.15     0.2415
```

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

```
##
## Call:
## best.tune(method = svm, train.x = y ~ ., data = dat, 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
##       gamma:  0.5
##
## Number of Support Vectors:  16
##
##  ( 8 8 )
##
```

```
##
## Number of Classes:  2
##
## Levels:
##  -1 1
```

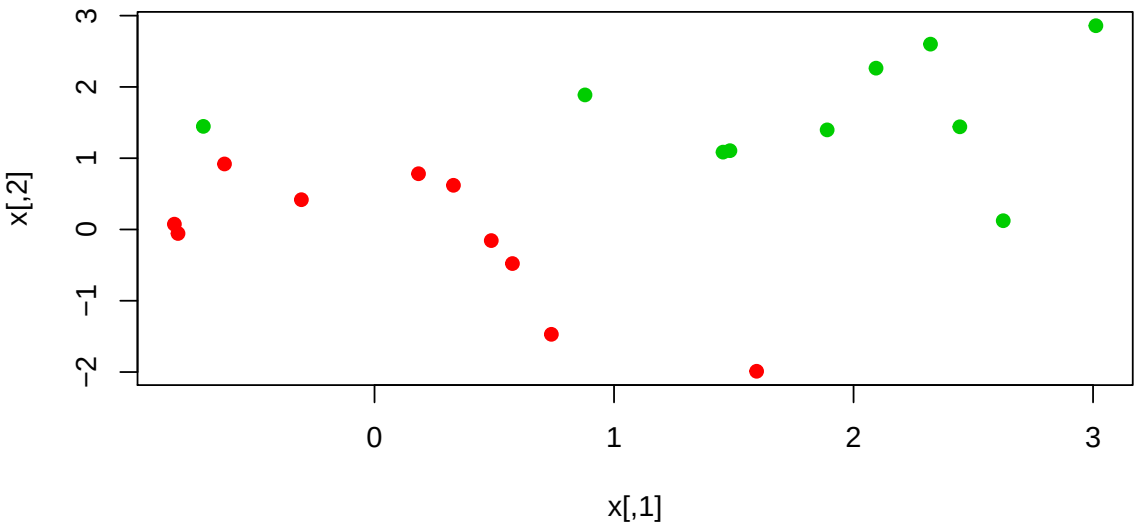
```
xtest=matrix(rnorm(20*2), ncol=2)
ytest=sample(c(-1,1), 20, rep=TRUE)
xtest[ytest==1,]=xtest[ytest==1,] + 1
testdat=data.frame(x=xtest, y=as.factor(ytest))
ypred=predict(bestmod,testdat)
table(predict=ypred, truth=testdat$y)
```

```
##        truth
## predict -1  1
##      -1 11  1
##      1   0  8
```

```
svmfit=svm(y~., data=dat, kernel="linear", cost=.01,scale=FALSE)
ypred=predict(svmfit,testdat)
table(predict=ypred, truth=testdat$y)
```

```
##        truth
## predict -1  1
##      -1 11  2
##      1   0  7
```

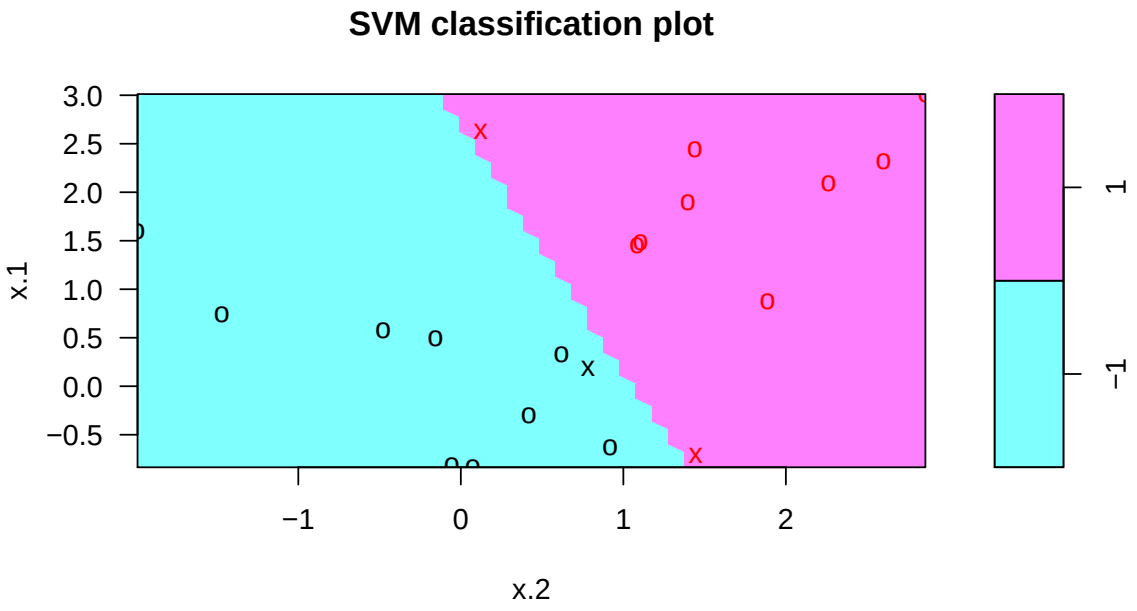
```
x[y==1,]=x[y==1,]+0.5
plot(x, col=(y+5)/2, pch=19)
```




```
dat=data.frame(x=x,y=as.factor(y))
svmfit=svm(y~., data=dat, kernel="linear", cost=1e5)
summary(svmfit)
```

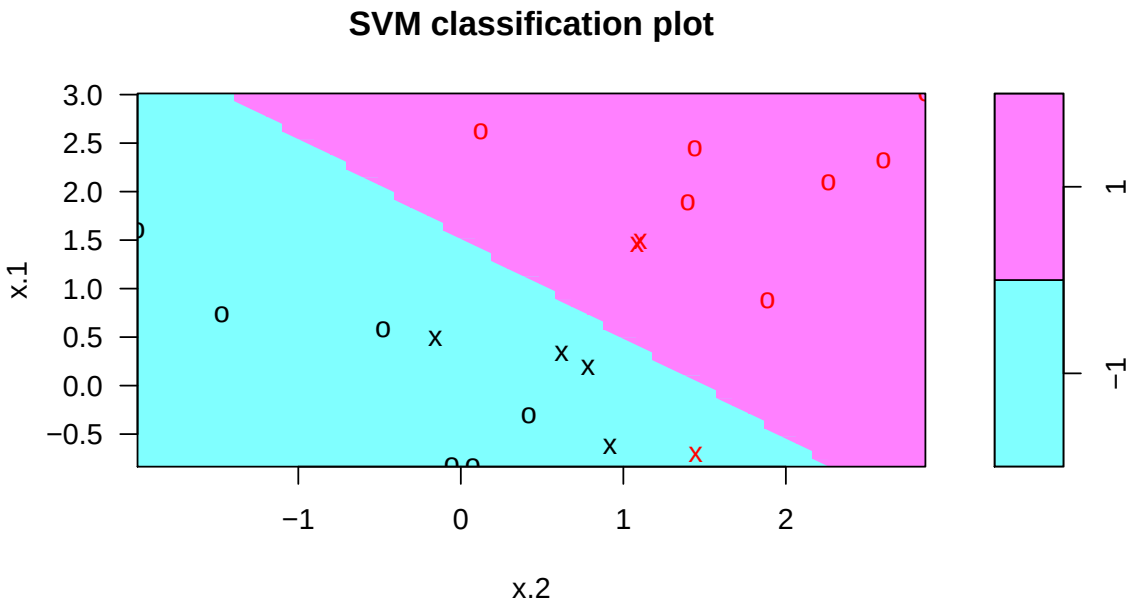
```
##
## Call:
## svm(formula = y ~ ., data = dat, kernel = "linear", cost = 1e+05)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1e+05
##       gamma:  0.5
##
## Number of Support Vectors:  3
##
##  ( 1 2 )
##
##
```

```
## Number of Classes:  2
##
## Levels:
##  -1 1

plot(svmfit, dat)
```




```
svmfit=svm(y~., data=dat, kernel="linear", cost=1)
summary(svmfit)

##
## Call:
## svm(formula = y ~ ., data = dat, kernel = "linear", cost = 1)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##       gamma:  0.5
##
```

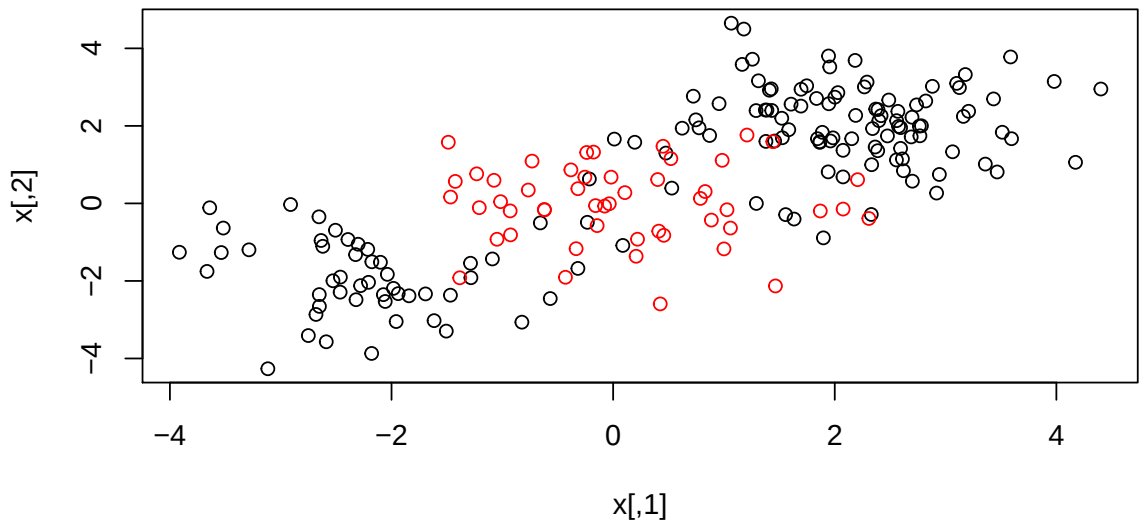
```
## Number of Support Vectors:  7
##
##  ( 4 3 )
##
##
## Number of Classes:  2
##
## Levels:
##  -1 1

plot(svmfit,dat)
```



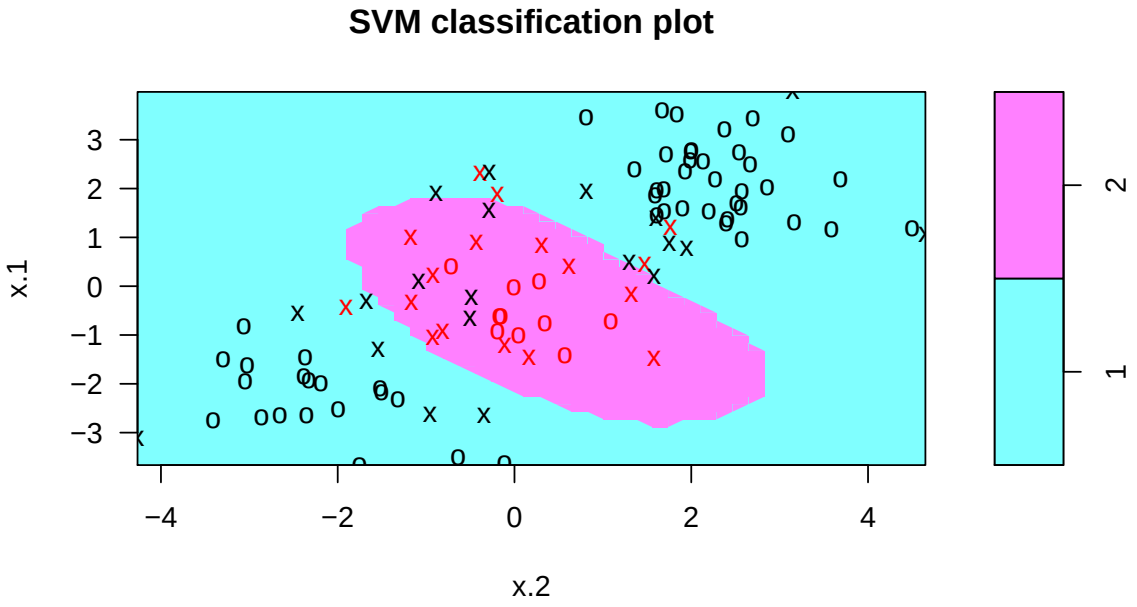
1.3 서포트벡터기계

```
# Support Vector Machine


set.seed(1)
x=matrix(rnorm(200*2), ncol=2)
x[1:100,]=x[1:100,]+2
x[101:150,]=x[101:150,]-2
```

```r
y=c(rep(1,150),rep(2,50))
dat=data.frame(x=x,y=as.factor(y))
plot(x, col=y)
```

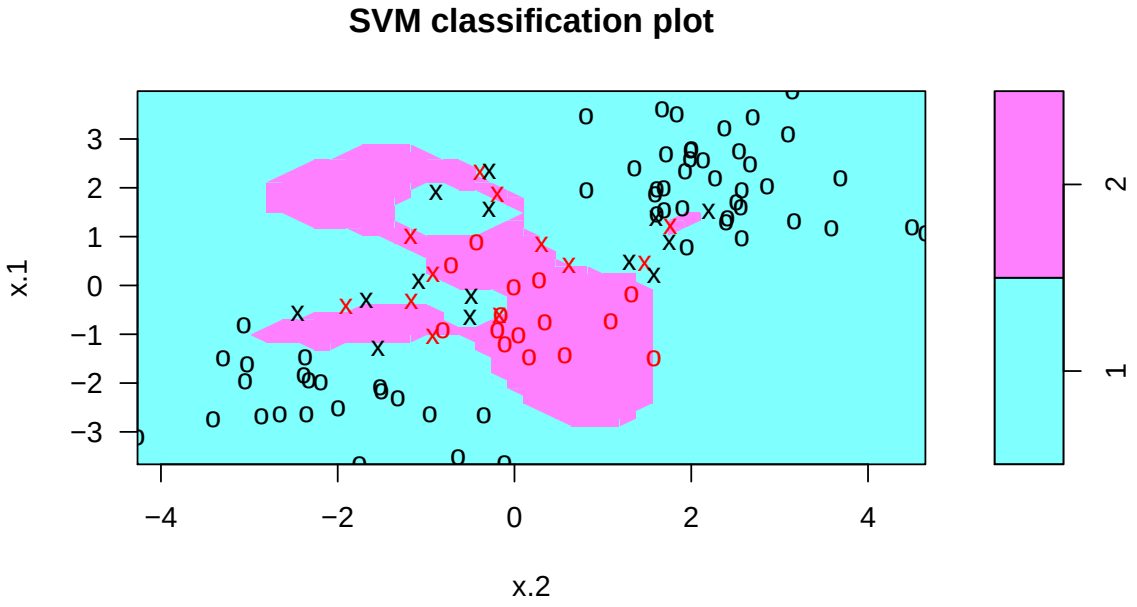



```r
train=sample(200,100)
svmfit=svm(y~., data=dat[train,], kernel="radial",  gamma=1, cost=1)
plot(svmfit, dat[train,])
```




```
summary(svmfit)

##
## Call:
## svm(formula = y ~ ., data = dat[train, ], kernel = "radial",
##     gamma = 1, cost = 1)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  1
##
## Number of Support Vectors:  37
##
##  ( 17 20 )
##
##
## Number of Classes:  2
```

```
##
## Levels:
##  1 2

svmfit=svm(y~., data=dat[train,], kernel="radial",gamma=1,cost=1e5)
plot(svmfit,dat[train,])
```




```
set.seed(1)
tune.out=tune(svm, y~., data=dat[train,], kernel="radial", ranges=list(cost=c(0.1,1,10,100,1000),gamma=
summary(tune.out)

##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##     1     2
##
## - best performance: 0.12
```

```
##
## - Detailed performance results:
##     cost gamma error dispersion
## 1  1e-01   0.5  0.27    0.11595
## 2  1e+00   0.5  0.13    0.08233
## 3  1e+01   0.5  0.15    0.07071
## 4  1e+02   0.5  0.17    0.08233
## 5  1e+03   0.5  0.21    0.09944
## 6  1e-01   1.0  0.25    0.13540
## 7  1e+00   1.0  0.13    0.08233
## 8  1e+01   1.0  0.16    0.06992
## 9  1e+02   1.0  0.20    0.09428
## 10 1e+03   1.0  0.20    0.08165
## 11 1e-01   2.0  0.25    0.12693
## 12 1e+00   2.0  0.12    0.09189
## 13 1e+01   2.0  0.17    0.09487
## 14 1e+02   2.0  0.19    0.09944
## 15 1e+03   2.0  0.20    0.09428
## 16 1e-01   3.0  0.27    0.11595
## 17 1e+00   3.0  0.13    0.09487
## 18 1e+01   3.0  0.18    0.10328
## 19 1e+02   3.0  0.21    0.08756
## 20 1e+03   3.0  0.22    0.10328
## 21 1e-01   4.0  0.27    0.11595
## 22 1e+00   4.0  0.15    0.10801
## 23 1e+01   4.0  0.18    0.11353
## 24 1e+02   4.0  0.21    0.08756
## 25 1e+03   4.0  0.24    0.10750
```

```
table(true=dat[-train,"y"], pred=predict(tune.out$best.model,newx=dat[-train,]))
```

```
##     pred
## true  1  2
##    1 56 21
##    2 18  5
```

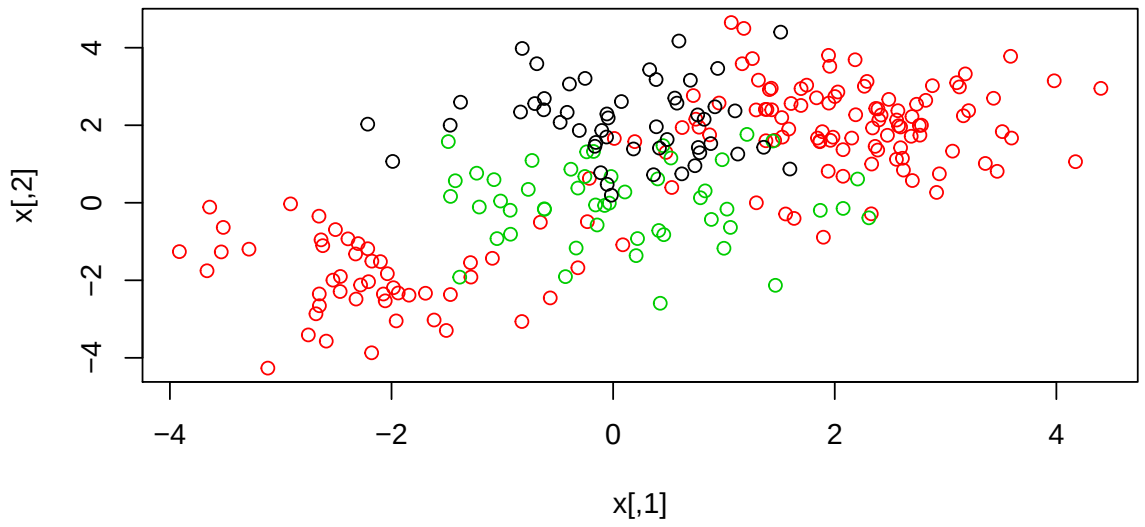
1.4 ROC 커브

```
# ROC Curves

library(ROCR)

rocplot=function(pred, truth, ...){
   predob = prediction(pred, truth)
   perf = performance(predob, "tpr", "fpr")
   plot(perf,...)}
svmfit.opt=svm(y~., data=dat[train,], kernel="radial",gamma=2, cost=1,decision.values=T)
fitted=attributes(predict(svmfit.opt,dat[train,],decision.values=TRUE))$decision.values
par(mfrow=c(1,2))
rocplot(fitted,dat[train,"y"],main="Training Data")

svmfit.flex=svm(y~., data=dat[train,], kernel="radial",gamma=50, cost=1, decision.values=T)
fitted=attributes(predict(svmfit.flex,dat[train,],decision.values=T))$decision.values
rocplot(fitted,dat[train,"y"],add=T,col="red")

fitted=attributes(predict(svmfit.opt,dat[-train,],decision.values=T))$decision.values
rocplot(fitted,dat[-train,"y"],main="Test Data")

fitted=attributes(predict(svmfit.flex,dat[-train,],decision.values=T))$decision.values
rocplot(fitted,dat[-train,"y"],add=T,col="red")

## Error: 함수 "prediction"를 찾을 수 없습니다
```

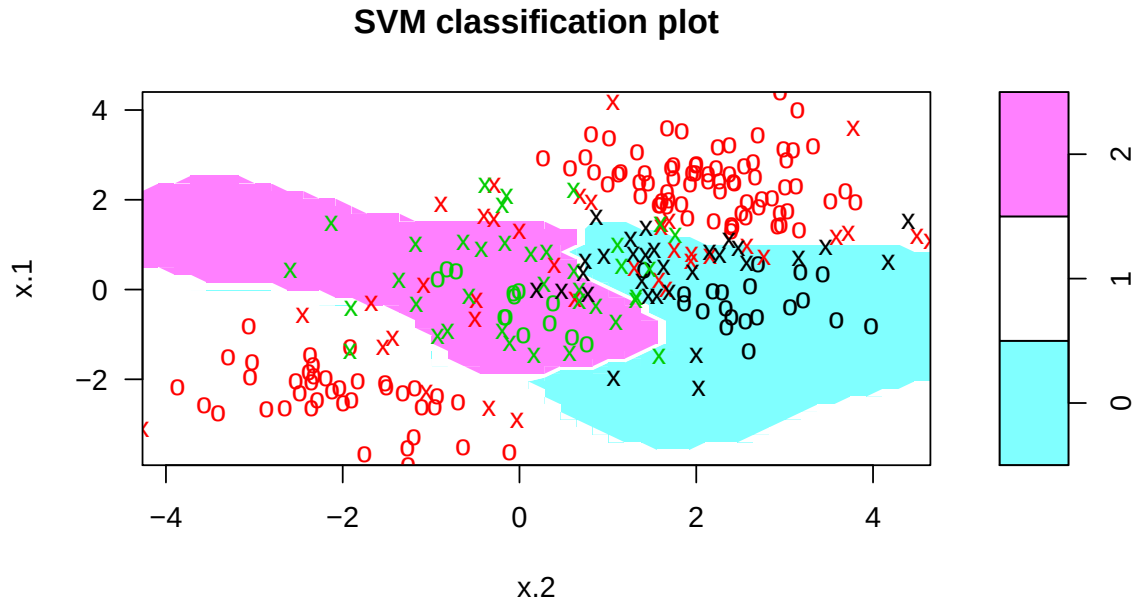
1.5 2개 이상의 클래스가 있는 경우의 서포트벡터머신

```
# SVM with Multiple Classes

set.seed(1)
x=rbind(x, matrix(rnorm(50*2), ncol=2))
y=c(y, rep(0,50))
x[y==0,2]=x[y==0,2]+2
dat=data.frame(x=x, y=as.factor(y))
par(mfrow=c(1,1))
```

```
plot(x,col=(y+1))
```




```
svmfit=svm(y~., data=dat, kernel="radial", cost=10, gamma=1)
plot(svmfit, dat)
```



1.6 유전자 발현 자료에 적용

```
# Application to Gene Expression Data

library(ISLR)
names(Khan)

## [1] "xtrain" "xtest"  "ytrain" "ytest"

dim(Khan$xtrain)

## [1]   63 2308

dim(Khan$xtest)

## [1]   20 2308

length(Khan$ytrain)

## [1] 63

length(Khan$ytest)

## [1] 20

table(Khan$ytrain)

##
##  1  2  3  4
##  8 23 12 20

table(Khan$ytest)

##
## 1 2 3 4
## 3 6 6 5

dat=data.frame(x=Khan$xtrain, y=as.factor(Khan$ytrain))
out=svm(y~., data=dat, kernel="linear",cost=10)
summary(out)
```

```
##
## Call:
## svm(formula = y ~ ., data = dat, kernel = "linear", cost = 10)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  10
##       gamma:  0.0004333
##
## Number of Support Vectors:  58
##
##  ( 20 20 11 7 )
##
##
## Number of Classes:  4
##
## Levels:
##  1 2 3 4
```

```
table(out$fitted, dat$y)
```

```
##
##      1  2  3  4
##   1  8  0  0  0
##   2  0 23  0  0
##   3  0  0 12  0
##   4  0  0  0 20
```

```
dat.te=data.frame(x=Khan$xtest, y=as.factor(Khan$ytest))
pred.te=predict(out, newdata=dat.te)
table(pred.te, dat.te$y)
```

```
##
## pred.te 1 2 3 4
##       1 3 0 0 0
```

```
##       2 0 6 2 0
##       3 0 0 4 0
##       4 0 0 0 5
```