9_SVM

Park Ju ho

2022 6 15

spam data 불러오기

```
setwd("D:/ /4-1 /  /R")

spam <- read.table('spam.txt', header=T, sep='\t')
str(spam)
```

```
## 'data.frame':    4601 obs. of  59 variables:
##  $ word.freq.make            : num  0 0.21 0.06 0 0 0 0 0 0.15 0.06 ...
##  $ word.freq.address         : num  0.64 0.28 0 0 0 0 0 0 0 0.12 ...
##  $ word.freq.all             : num  0.64 0.5 0.71 0 0 0 0 0 0.46 0.77 ...
##  $ word.freq.3d              : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word.freq.our             : num  0.32 0.14 1.23 0.63 0.63 1.85 1.92 1.88 0.61 0.19 ...
##  $ word.freq.over            : num  0 0.28 0.19 0 0 0 0 0 0 0.32 ...
##  $ word.freq.remove          : num  0 0.21 0.19 0.31 0.31 0 0 0 0.3 0.38 ...
##  $ word.freq.internet        : num  0 0.07 0.12 0.63 0.63 1.85 0 1.88 0 0 ...
##  $ word.freq.order           : num  0 0 0.64 0.31 0.31 0 0 0 0.92 0.06 ...
##  $ word.freq.mail            : num  0 0.94 0.25 0.63 0.63 0 0.64 0 0.76 0 ...
##  $ word.freq.receive         : num  0 0.21 0.38 0.31 0.31 0 0.96 0 0.76 0 ...
##  $ word.freq.will            : num  0.64 0.79 0.45 0.31 0.31 0 1.28 0 0.92 0.64 ...
##  $ word.freq.people          : num  0 0.65 0.12 0.31 0.31 0 0 0 0 0.25 ...
##  $ word.freq.report          : num  0 0.21 0 0 0 0 0 0 0 0 ...
##  $ word.freq.addresses       : num  0 0.14 1.75 0 0 0 0 0 0 0.12 ...
##  $ word.freq.free            : num  0.32 0.14 0.06 0.31 0.31 0 0.96 0 0 0 ...
##  $ word.freq.business        : num  0 0.07 0.06 0 0 0 0 0 0 0 ...
##  $ word.freq.email           : num  1.29 0.28 1.03 0 0 0 0.32 0 0.15 0.12 ...
##  $ word.freq.you             : num  1.93 3.47 1.36 3.18 3.18 0 3.85 0 1.23 1.67 ...
##  $ word.freq.credit          : num  0 0 0.32 0 0 0 0 0 3.53 0.06 ...
##  $ word.freq.your            : num  0.96 1.59 0.51 0.31 0.31 0 0.64 0 2 0.71 ...
##  $ word.freq.font            : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word.freq.000             : num  0 0.43 1.16 0 0 0 0 0 0 0.19 ...
##  $ word.freq.money           : num  0 0.43 0.06 0 0 0 0 0 0.15 0 ...
##  $ word.freq.hp              : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word.freq.hpl             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word.freq.george          : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word.freq.650             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word.freq.lab             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word.freq.labs            : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word.freq.telnet          : num  0 0 0 0 0 0 0 0 0 0 ...
```

```
##  $ word.freq.857             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word.freq.data            : num  0 0 0 0 0 0 0 0 0.15 0 ...
##  $ word.freq.415             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word.freq.85              : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word.freq.technology      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word.freq.1999            : num  0 0.07 0 0 0 0 0 0 0 0 ...
##  $ word.freq.parts           : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word.freq.pm              : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word.freq.direct          : num  0 0 0.06 0 0 0 0 0 0 0 ...
##  $ word.freq.cs              : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word.freq.meeting         : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word.freq.original        : num  0 0 0.12 0 0 0 0 0 0.3 0 ...
##  $ word.freq.project         : num  0 0 0 0 0 0 0 0 0 0.06 ...
##  $ word.freq.re              : num  0 0 0.06 0 0 0 0 0 0 0 ...
##  $ word.freq.edu             : num  0 0 0.06 0 0 0 0 0 0 0 ...
##  $ word.freq.table           : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word.freq.conference      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ char.freq.semi            : num  0 0 0.01 0 0 0 0 0 0 0.04 ...
##  $ char.freq.lparen          : num  0 0.132 0.143 0.137 0.135 0.223 0.054 0.206 0.271 0.03 ...
##  $ char.freq.lbrack          : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ char.freq.bang            : num  0.778 0.372 0.276 0.137 0.135 0 0.164 0 0.181 0.244 ...
##  $ char.freq.dollar          : num  0 0.18 0.184 0 0 0 0.054 0 0.203 0.081 ...
##  $ char.freq.hash            : num  0 0.048 0.01 0 0 0 0 0 0.022 0 ...
##  $ capital.run.length.average: num  3.76 5.11 9.82 3.54 3.54 ...
##  $ capital.run.length.longest: int  61 101 485 40 40 15 4 11 445 43 ...
##  $ capital.run.length.total  : int  278 1028 2259 191 191 54 112 49 1257 749 ...
##  $ spam                      : chr  "spam" "spam" "spam" "spam" ...
##  $ rgroup                    : int  52 91 49 88 73 45 93 10 60 75 ...
```

```
spamTrain <- spam[spam$rgroup>=10,]
spamTest <- spam[spam$rgroup<10,]

spamVars <- setdiff(colnames(spam),list('rgroup','spam'))
```

logistic

```
spamFormula <- as.formula(paste('spam=="spam"',
                                paste(spamVars,collapse=' + '),sep=' ~ '))
spamModel <- glm(spamFormula,family=binomial(link='logit'),data=spamTrain)

spamTest$pred <- predict(spamModel,newdata=spamTest, type='response')

print(with(spamTest,table(y=spam,glPred=pred>=0.5)))
```

```
##           glPred
## y          FALSE TRUE
##   non-spam   264   14
##   spam        22  158
```

svm

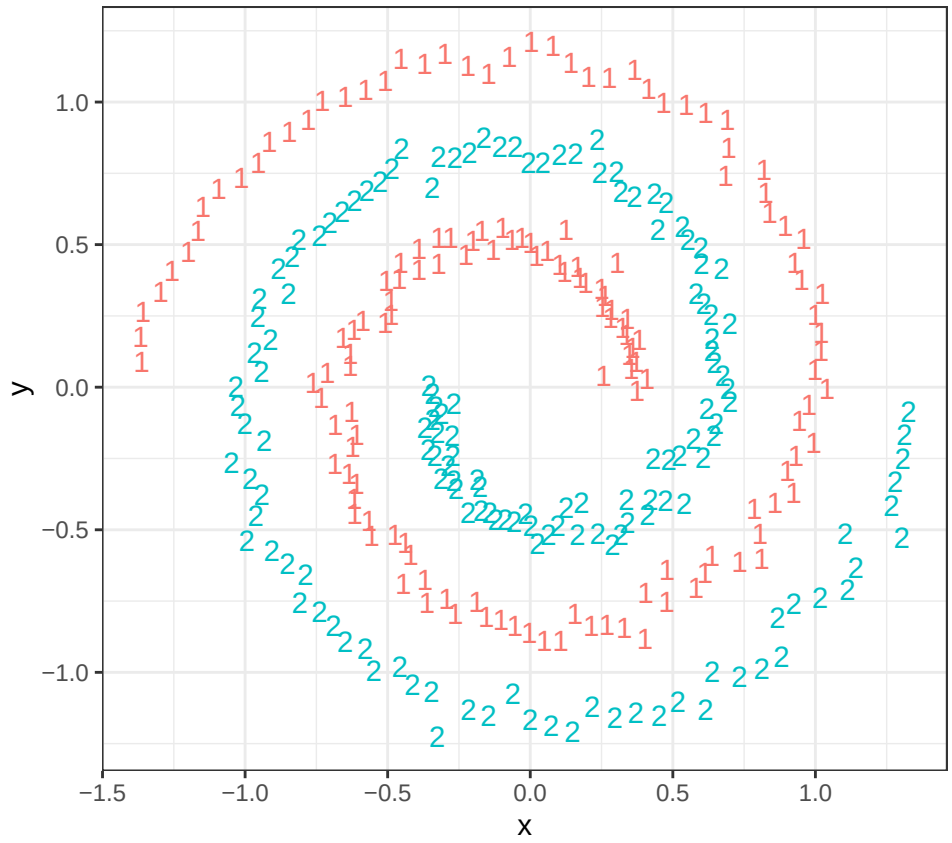
svm은 통계적 모형이라고 보기 힘들기에 결과를 보면 Support vector의 수 정도만 출력된다 kernel = 말 그대로 kernel 종류, vanilladot = 특별한 변환 없이 내적을 계산, rdfdot = Gaussian cost = 제약 위배의 비용(작을 수록 시간이 오래 걸림) prob.model = 분류를 위한 확률 cross = cross validation의 k 수

이유를 모르겠으나 학습이 진행되질 않음…..

```
library(kernlab)

library(kernlab)
spamFormulaV <- as.formula(paste('spam',paste(spamVars,collapse=' + '),sep=' ~ '))
svmM <- ksvm(spamFormulaV,data=spamTrain, kernel='rbfdot',C=10, prob.model=T, cross=5)
spamTest$svmPred <- predict(svmM,newdata=spamTest,type='response')
print(with(spamTest,table(y=spam,svmPred=svmPred)))
print(svmM)

head(predict(svmM,spamTest,type='probabilities' ))
```

비선형 svm

```
#
library(kernlab)
data(spirals)

set.seed(1)
sc <- specc(spirals, centers = 2)
s <- data.frame(x=spirals[,1],y=spirals[,2],class=as.factor(sc))

library('ggplot2')
ggplot(data=s) + geom_text(aes(x=x,y=y,label=class,color=class)) +
   coord_fixed() + theme_bw() + theme(legend.position='none')
```




```
set.seed(123)
s$group <- sample.int(100,size=dim(s)[[1]],replace=T)
sTrain <- subset(s,group>10)
sTest <- subset(s,group<=10)
```

선형 커널을 사용한 SVM으로 데이터를 분류

```
library(e1071)
mSVMV <- svm(class~x+y,data=sTrain,kernel='linear',type='C-classification')

mSVMV
```

```
##
## Call:
## svm(formula = class ~ x + y, data = sTrain, kernel = "linear", type = "C-classification")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  231
```

```
sTest$predSVMV <- predict(mSVMV,newdata=sTest,type='response')
print(with(sTest,table(y=class,svmPred=predSVMV)))
```

```
##    svmPred
## y    1  2
##   1 13  3
##   2  4 12
```

Support vector는 231개이고 총 32개의 데이터 중 7개를 틀린 것을 확인 할 수 있다

hyperprameter tunning

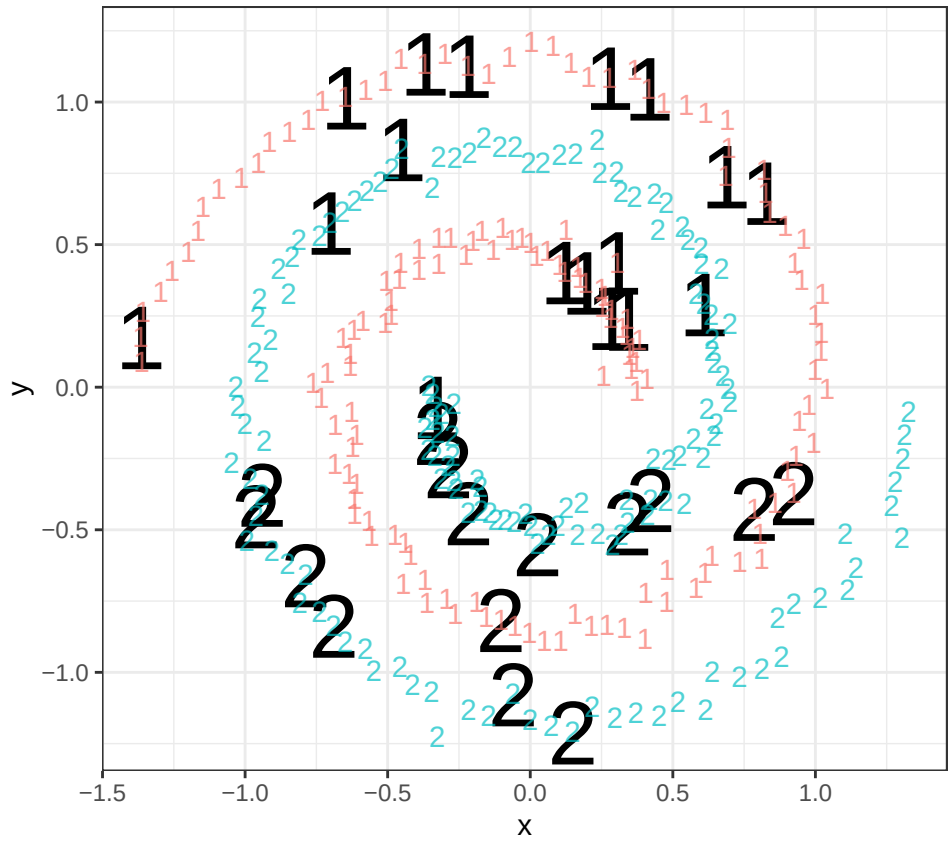
gamma와 cost를 tune 함수를 이용하여 가장 best의 모형을 찾는 것

```
tune.result<-tune(svm, class~x+y,data=sTrain,kernel='linear',type='C-classification',
        range=list(gamma=seq(0,1,0.1),cost=2^(1:9)))

tune.result
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  gamma cost
##      0    2
##
## - best performance: 0.3732194
```

그 결과gamma는 0 cosst는 2 일 때가 가장 좋다는 결론이 나옴

```
ggplot() + geom_text(data=sTest,aes(x=x,y=y,label=predSVMV),size=12) +
    geom_text(data=s,aes(x=x,y=y,label=class,color=class),alpha=0.7) +
    coord_fixed() + theme_bw() + theme(legend.position='none')
```

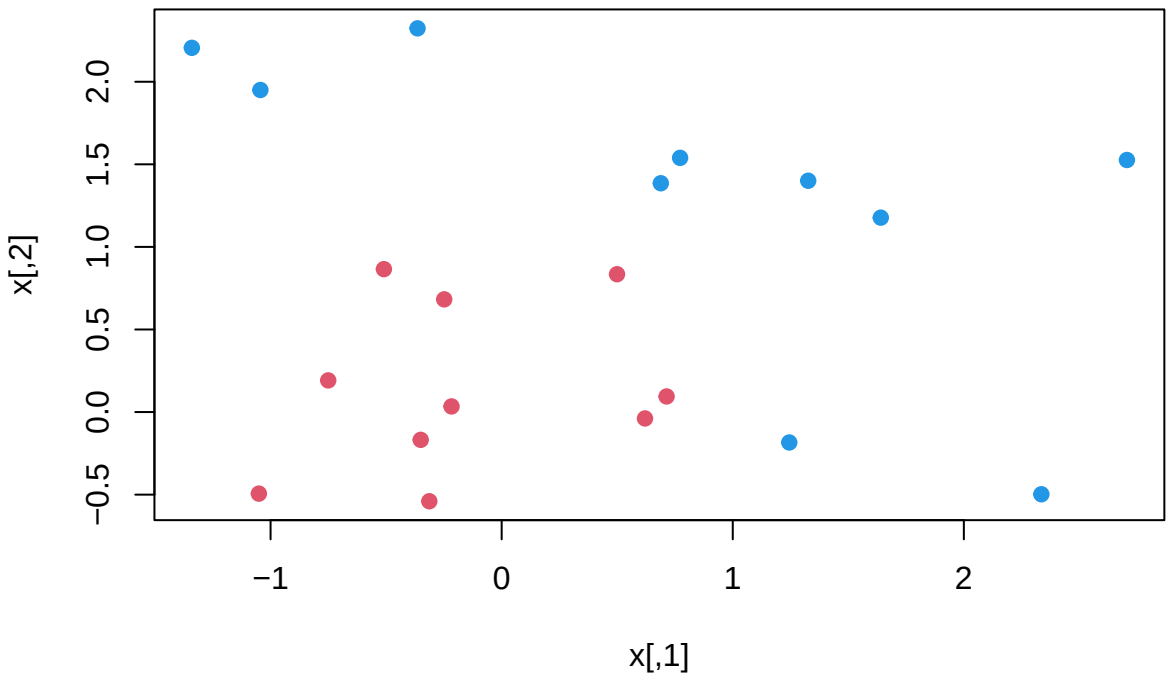


큰 글씨 = 예측값 => 1:4개 2:3개를 틀렸다는 결과처럼 그래프를 보면 총 7개가 잘 못 분류되 어 있다

새로운 데이터로 해보는 svm

new data

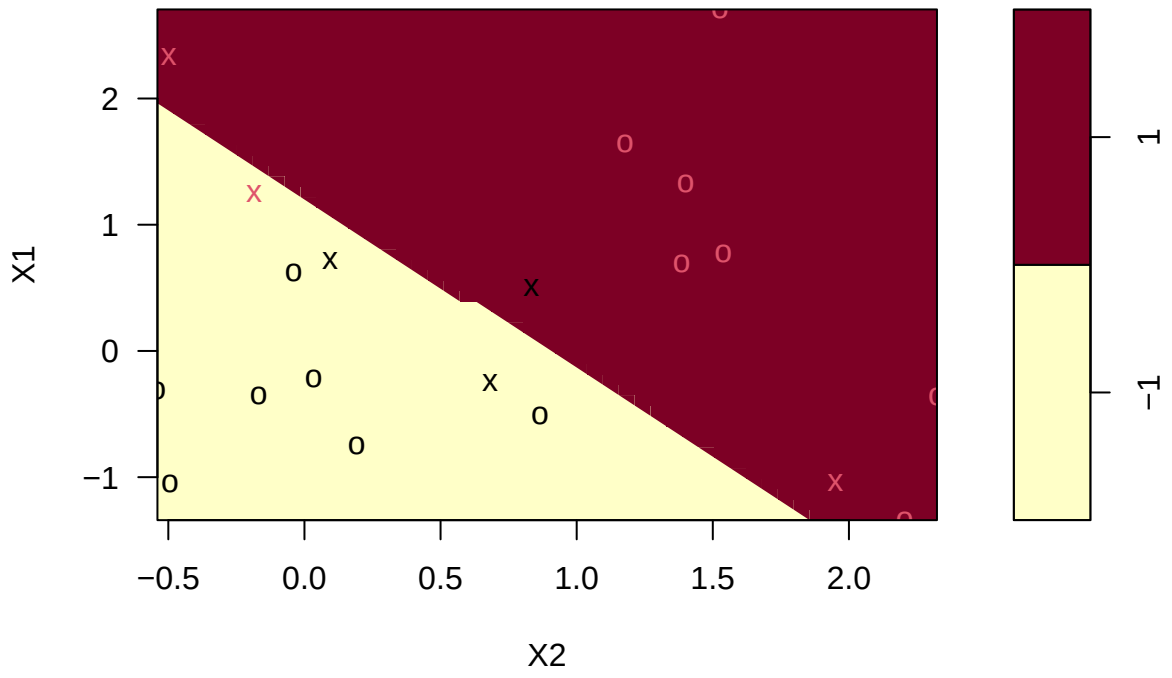
```
set.seed(10111)
x = matrix(rnorm(40), 20, 2)
y = rep(c(-1, 1), c(10, 10))
x[y == 1,] = x[y == 1,] + 1
plot(x, col = y + 3, pch = 19)
```




```
dat = data.frame(x, y = as.factor(y))
#y  vector
svmfit = svm(y ~ ., data = dat, kernel = "linear", cost = 10, scale = FALSE)
print(svmfit)
```

```
##
## Call:
## svm(formula = y ~ ., data = dat, kernel = "linear", cost = 10, scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  10
##
## Number of Support Vectors:  6
```

```
plot(svmfit, dat)#x = support vector
```

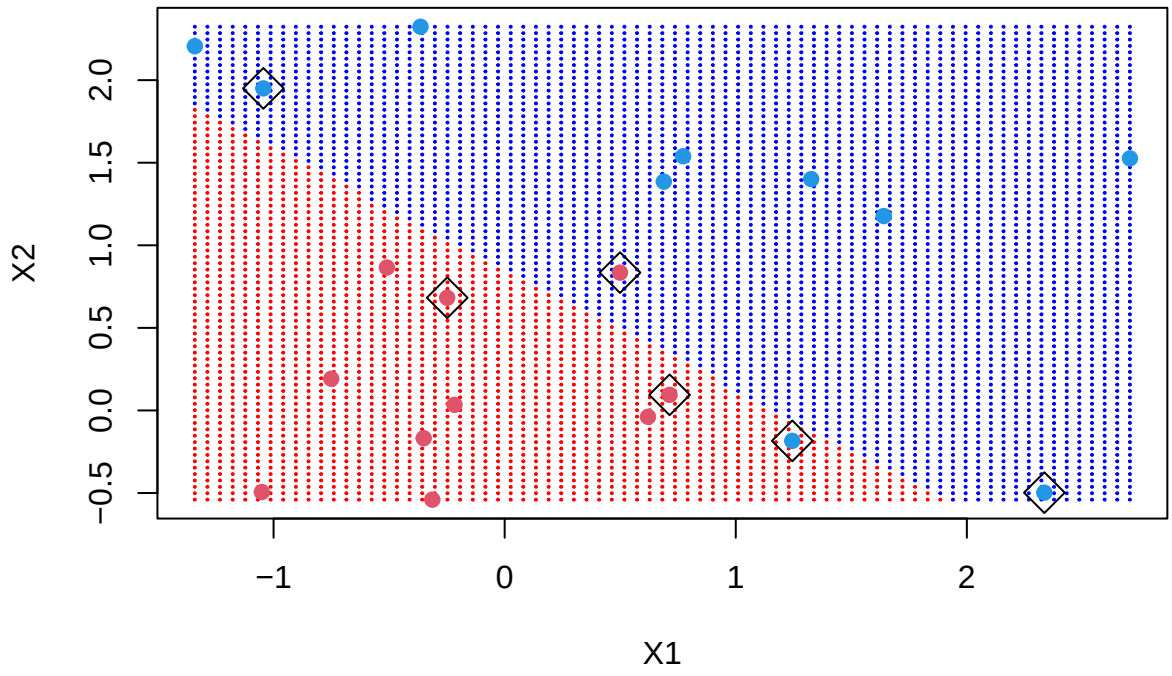


총 6개의 서포트 백터로 구성되어 있다

```
make.grid = function(x, n = 75) {
  grange = apply(x, 2, range)
  x1 = seq(from = grange[1,1], to = grange[2,1], length = n) #grange =        seq
  x2 = seq(from = grange[1,2], to = grange[2,2], length = n)
  expand.grid(X1 = x1, X2 = x2)
}#

xgrid = make.grid(x)
xgrid[1:10,]
```

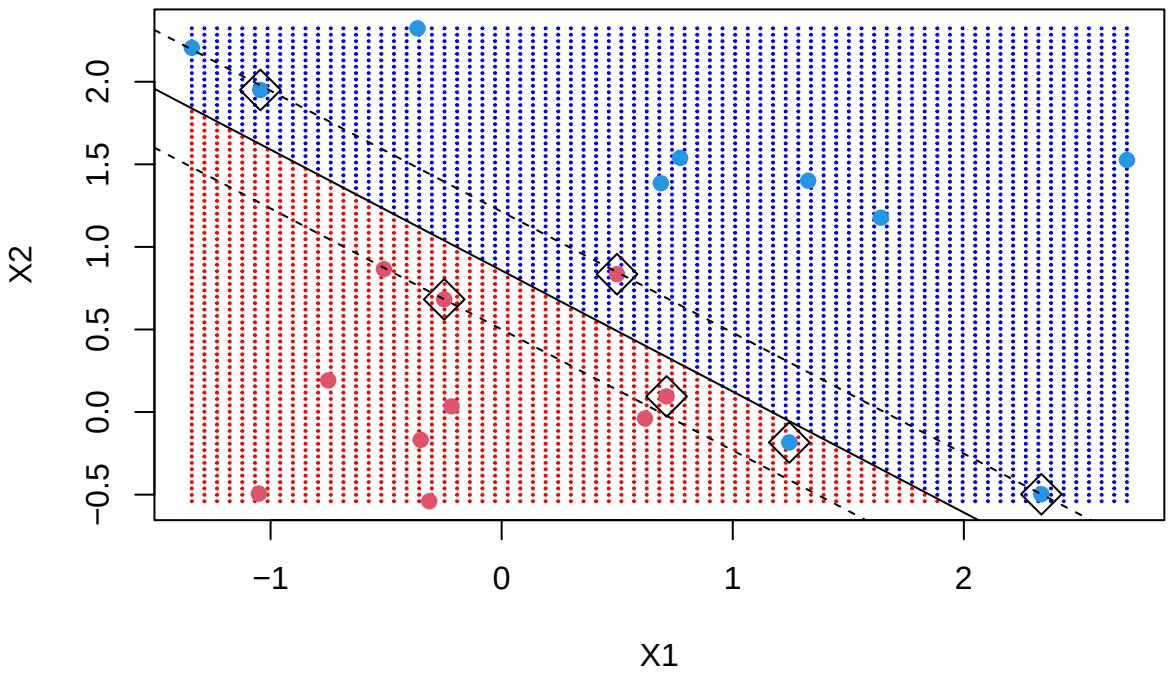
```
##            X1         X2
## 1  -1.3406379 -0.5400074
## 2  -1.2859572 -0.5400074
## 3  -1.2312766 -0.5400074
## 4  -1.1765959 -0.5400074
## 5  -1.1219153 -0.5400074
## 6  -1.0672346 -0.5400074
## 7  -1.0125540 -0.5400074
## 8  -0.9578733 -0.5400074
## 9  -0.9031927 -0.5400074
## 10 -0.8485120 -0.5400074
```

```
ygrid = predict(svmfit, xgrid)
plot(xgrid, col = c("red","blue")[as.numeric(ygrid)], pch = 20, cex = .2)
points(x, col = y + 3, pch = 19)
points(x[svmfit$index,], pch = 5, cex = 2)
```



상자친 서포트 벡터를 표시하여 svm 결과를 표시

```
beta = drop(t(svmfit$coefs)%*%x[svmfit$index,])#
beta0 = svmfit$rho

plot(xgrid, col = c("red", "blue")[as.numeric(ygrid)], pch = 20, cex = .2)
points(x, col = y + 3, pch = 19)
points(x[svmfit$index,], pch = 5, cex = 2)
abline(beta0 / beta[2], -beta[1] / beta[2])
abline((beta0 - 1) / beta[2], -beta[1] / beta[2], lty = 2)
abline((beta0 + 1) / beta[2], -beta[1] / beta[2], lty = 2)
```



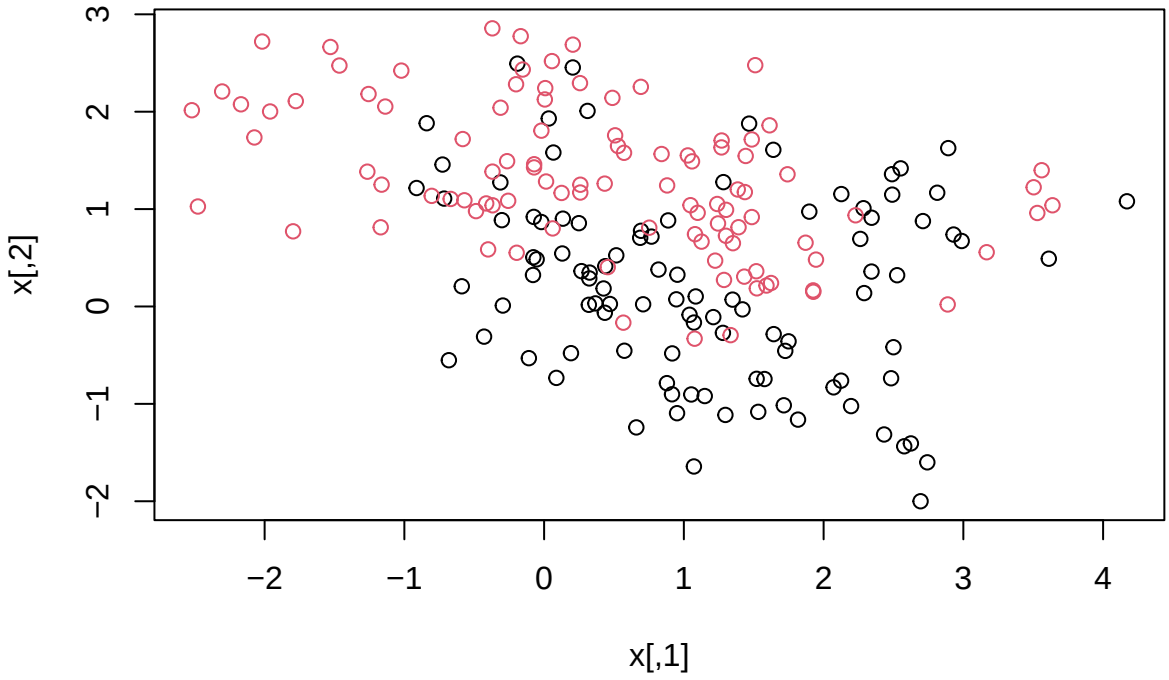
결정경계와 plus,minnus plane을 추가한 그래프

비선형 svm의 다른 예제

```
load(file = "ESL.mixture.rda")
names(ESL.mixture)
```

```
## [1] "x"        "y"        "xnew"     "prob"     "marginal" "px1"      "px2"
## [8] "means"
```

```
rm(x, y)
attach(ESL.mixture)
plot(x, col = y + 1)
```

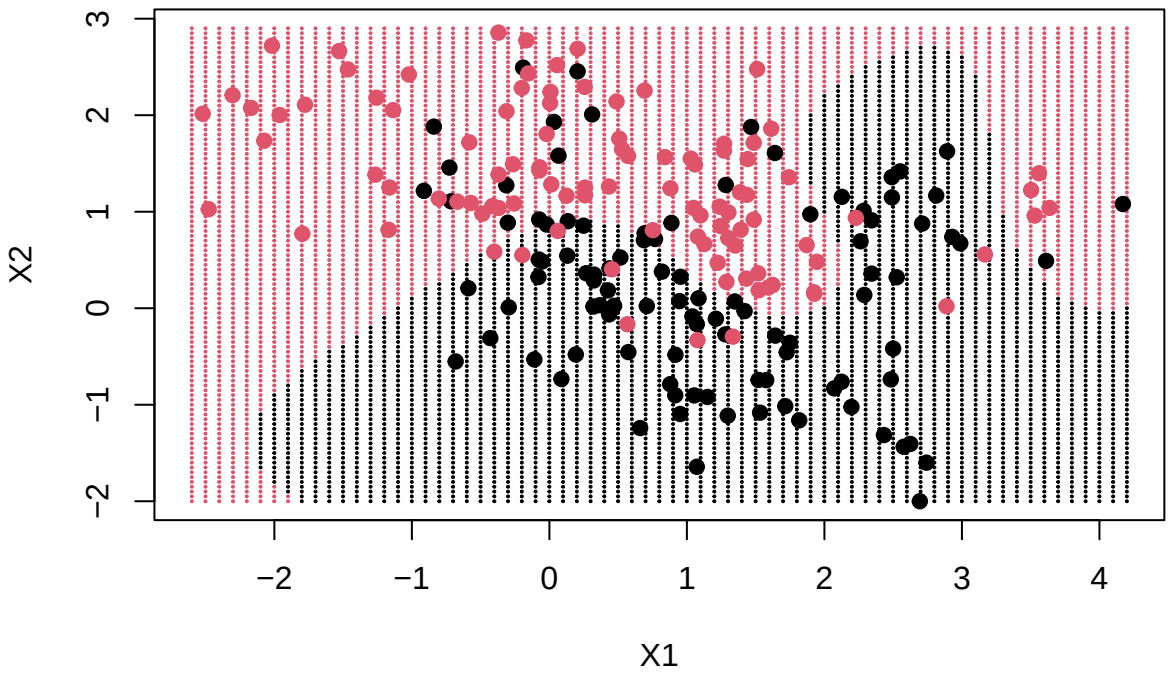



```
dat = data.frame(y = factor(y), x)
fit = svm(factor(y) ~ ., data = dat, scale = FALSE, kernel = "radial", cost = 5)
fit
```

```
##
## Call:
## svm(formula = factor(y) ~ ., data = dat, kernel = "radial", cost = 5,
##     scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  5
##
## Number of Support Vectors:  103
```

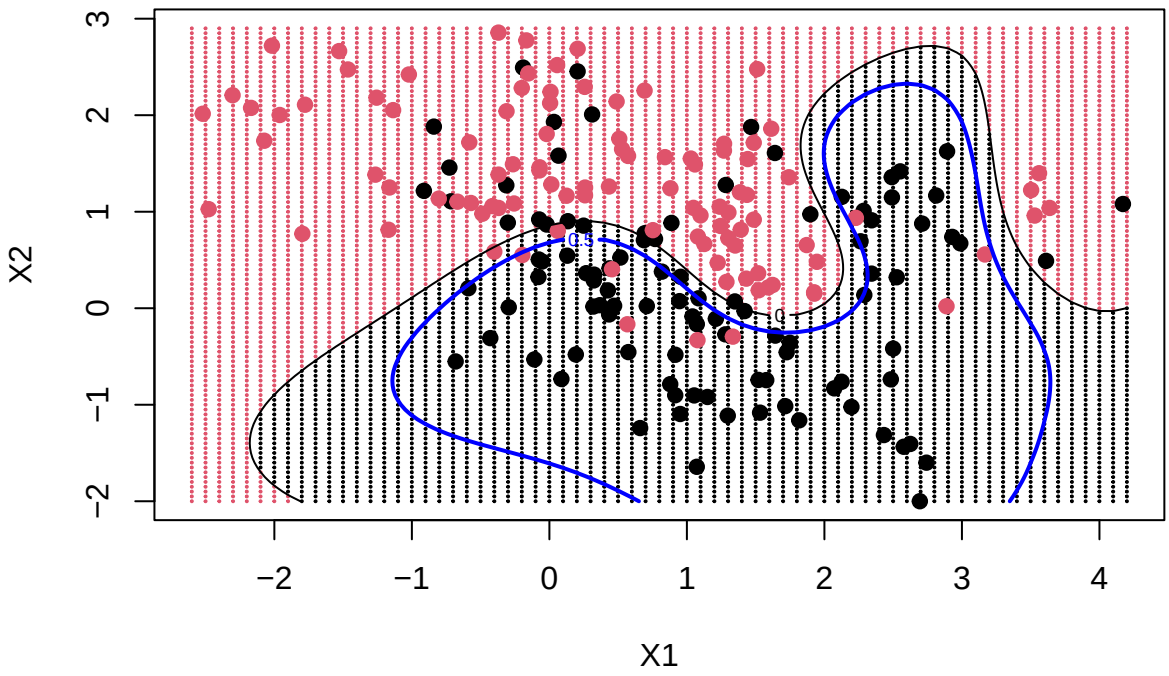
총 103개의 백터와 radial kernel을 활용하여 svm 진행

```
xgrid = expand.grid(X1 = px1, X2 = px2)
ygrid = predict(fit, xgrid)

plot(xgrid, col = as.numeric(ygrid), pch = 20, cex = .2)
points(x, col = y + 1, pch = 19)
```



값과 경계를 표현하면 이런 식으로 그래프가 나옴

```
func = predict(fit, xgrid, decision.values = TRUE)
func = attributes(func)$decision

xgrid = expand.grid(X1 = px1, X2 = px2)
ygrid = predict(fit, xgrid)
plot(xgrid, col = as.numeric(ygrid), pch = 20, cex = .2)
points(x, col = y + 1, pch = 19)

contour(px1, px2, matrix(func, 69, 99), level = 0, add = TRUE)
contour(px1, px2, matrix(func, 69, 99), level = 0.5, add = TRUE, col = "blue", lwd = 2)
```



그 결정경계를 표시하여 등고선으로 그림