Use Monte Carlo simulation to generate observations from two classes $Y = 1$ and $Y = -1$, and, two classifiers X_1 and X_2. Use function `svm` from library `e1071` to fit support vector methods. It uses a parameter `kernel` to fit different models. If `kernel="linear"`, `svm` fits a support vector classifier. If `kernel="polynomial"`, or `kernel="radial"`, `svm` fits a support vector machine (`kernel="polynomial"` requires argument `d` to choose the degree of the polynomial, `kernel="radial"` requires argument `gamma` to choose the value of parameter γ).
It uses a parameter `cost` to adjuste the size of the margin. When `cost` is small, margin is wide (with many support vectors on the margin). When `cost` is large, margin is narrow (with few support vectors on the margin). Note that the response must be defined as a categorical variable.

a) Generate training and test sets, each of 20 observations, from two nonlinearly separable classes. Plot the data to identify possible boundaries.

b) Use `cost=10` to fit a *support vector classifier*. Identify the number of misclassifications and number of support vectors from each class. Repeat with `cost=0.10` and compare the margin sizes.

c) Use function `tune()` to perform ten-fold cross validation on `cost` values.

d) Use best cross validation model to predict class in the test set. What is the test error rate?

e) Repeat this exercise with observations from two linearly separable classes.

f) Generate training and test sets, each of 200 observations, from two nonlinearly separable classes. Plot the data to identify possible boundaries.

g) Use `gamma=1` and `cost=1` to fit a *support vector machine* with radial kernel. Identify then number of misclassifications, number of support vectors from each class. Repeat with `cost=1e5` and compare the margin sizes.

h) Use function `tune()` to perform ten-fold cross validation on `cost` and `gamma` values.

i) Use the best cross validation model to predict class in the test set. What is the test error rate?

j) Generate a data set with 200 observations, from three nonlinearly separable classes. Plot the data to identify possible boundaries.

k) Use `gamma=1` and `cost=1` to fit a *support vector machine* with radial kernel.

```
# svm3.r

library(e1071)     # svm()  tune()

# 200-dataset with nonlinear class boundary
#============================================================
set.seed(1)
n1= rnorm(200)
n2= rnorm(200)
k = 2
aux=c(rep(k,100),rep(-k,50),rep(0,50))
x1 = n1+aux
x2 = n2+aux
dat1=data.frame(y,x1,x2)

yy=c(rep(1,150),rep(2,50))
y = factor(yy)
dat=data.frame(y,x1,x2)
plot(x2~x1,dat,col=(3-yy),pch=15+yy,cex=0.75)
grid()

# training set
train=sample(200,100)

plot(dat$x1[train]~dat$x2[train],col=yy[train],pch=15+yy[train],cex=0.7)
grid()

# radial kernel
#============================================================
svm1=svm(y~., data=dat[train,],kernel="radial",gamma=1,cost=1)
summary(svm1)
plot(svm1, dat[train,]);grid()
# 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

# large cost, irregular boundary
svm2=svm(y~., data=dat[train,], kernel="radial",gamma=1,cost=1e5)
plot(svm2,dat[train,]);grid()
```

```
# cross validation on cost, gamma, values
#=============================================================
set.seed(1)
costs=c(0.1,1,10,100,1000)
gammas=c(0.5,1,2,3,4)
tune.out=tune(svm,y~.,data=dat[train,],kernel="radial",ranges=list(cost=costs,gamma=gammas))
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 0.11595018
# 2  1e+00   0.5  0.13 0.08232726
# 3  1e+01   0.5  0.15 0.07071068
# 4  1e+02   0.5  0.17 0.08232726
# 5  1e+03   0.5  0.21 0.09944289
# 6  1e-01   1.0  0.25 0.13540064
# 7  1e+00   1.0  0.13 0.08232726
# 8  1e+01   1.0  0.16 0.06992059
# 9  1e+02   1.0  0.20 0.09428090
# 10 1e+03   1.0  0.20 0.08164966
# 11 1e-01   2.0  0.25 0.12692955
# 12 1e+00   2.0  0.12 0.09189366   smallest error
# 13 1e+01   2.0  0.17 0.09486833
# 14 1e+02   2.0  0.19 0.09944289
# 15 1e+03   2.0  0.20 0.09428090
# 16 1e-01   3.0  0.27 0.11595018
# 17 1e+00   3.0  0.13 0.09486833
# 18 1e+01   3.0  0.18 0.10327956
# 19 1e+02   3.0  0.21 0.08755950
# 20 1e+03   3.0  0.22 0.10327956
# 21 1e-01   4.0  0.27 0.11595018
# 22 1e+00   4.0  0.15 0.10801234
# 23 1e+01   4.0  0.18 0.11352924
# 24 1e+02   4.0  0.21 0.08755950
# 25 1e+03   4.0  0.24 0.10749677

# test error rate
bestmod=tune.out$best.model
ypred=predict(bestmod,newvalue=dat[-train,])
table(pred=ypred,true=dat[-train,"y"])
#    true
#pred  1  2
#   1 56 18
#   2 21  5
```

```
prop.table(table(pred=ypred,true=dat[-train,"y"]))
#      true
#pred     1    2
#   1 0.56 0.18
#   2 0.21 0.05

# test error rate is 0.21 + 0.18 = 0.39
```

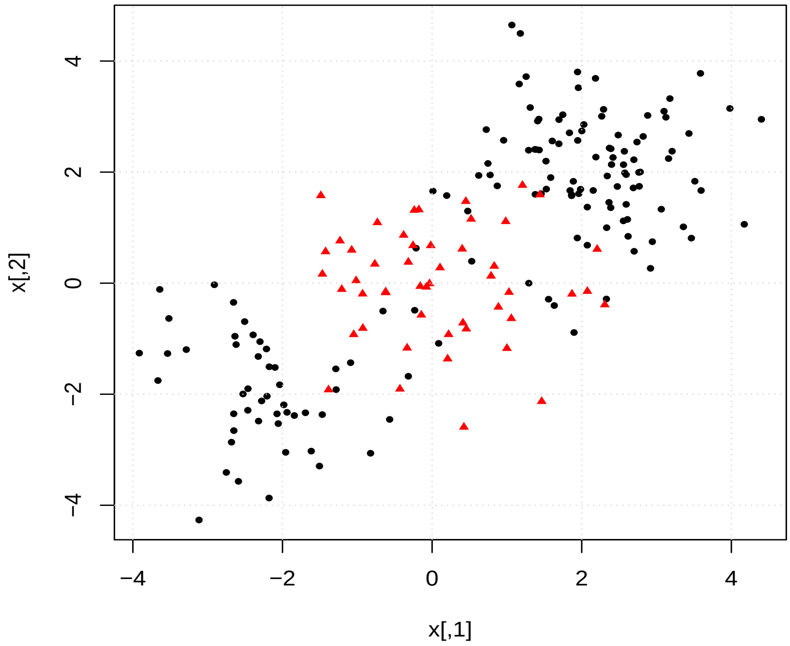

Figure 1: Dataset ($n = 200$) with nonlinear class boundary, for svm()

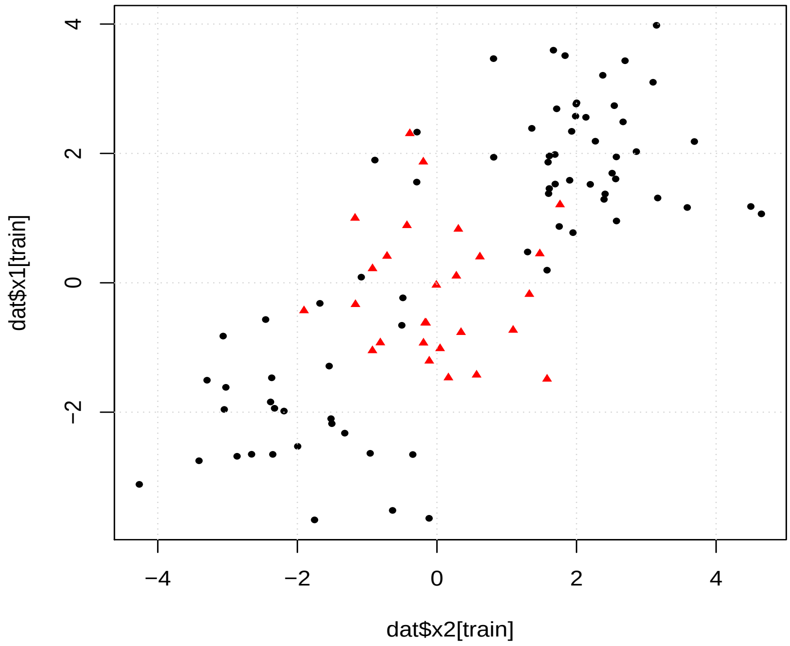

Figure 2: training set ($n = 100$) with nonlinear class boundary, for svm1

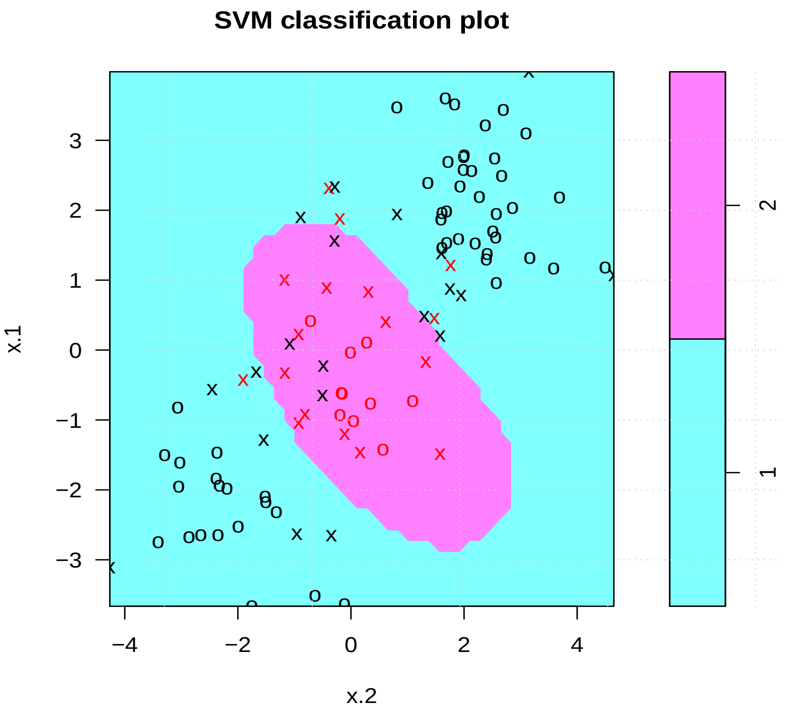

Figure 3: Support vector machine (radial) with cost 1

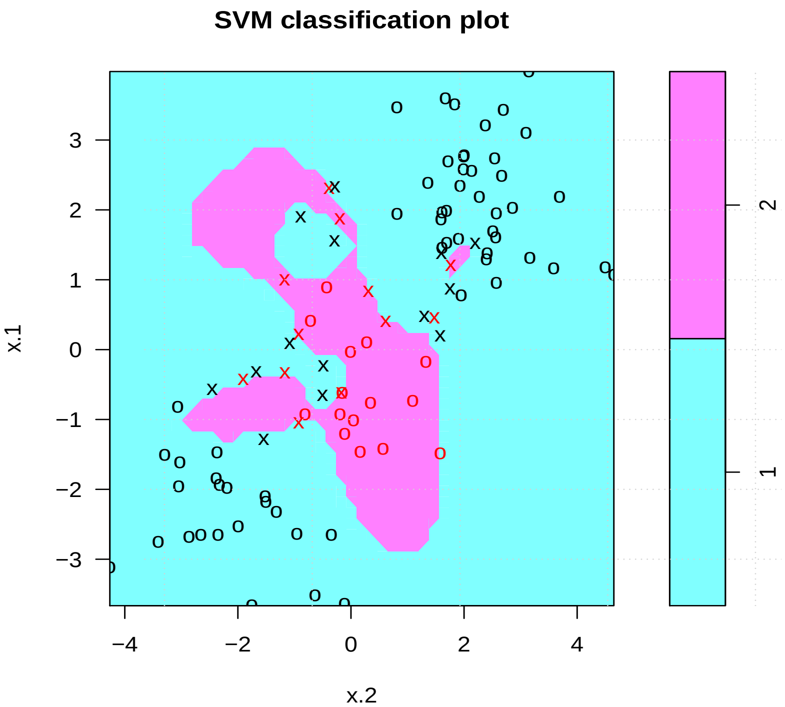

Figure 4: Support vector machine (radial) with cost 1e5