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.

```
# svc3.r

library(e1071)          # svm()  tune()
setwd("C:/Users/USC Guest/Downloads")   # saving folder

# train dataset
#========================================================
set.seed(1)
n1= rnorm(20)
n2= rnorm(20)
yy=c(rep(-1,10), rep(1,10))

par(mfrow=c(1,2))
bound = c(-3,3)
plot(n1~n2, col=(3-yy),pch=19,cex=0.8,xlim=bound,ylim=bound)
grid()

# shifting one category along 45 degree line
k = 1
aux=c(rep(0,10), rep(k,10))
x1 = n1+aux
x2 = n2+aux
plot(x1~x2, col=(3-yy),pch=19,cex=0.75,xlim=bound,ylim=bound)
grid()

# Not linearly separable

# dataframe
y=as.factor(yy)
dat=data.frame(y,x1,x2)

# svc with cost 10
#========================================================
svc1=svm(y~.,data=dat, kernel="linear", cost=10,scale=F)
plot(svc1,dat)

# x1 is vertical axis now
# all obs shown as 0, supp. vectors as X (there are 7)
# red for y = 1 obs, black for y = -1
# There are obs outside margins
plot(svc1,dat,xlim=bound,ylim=c(-2,3))
grid()

names(svc1)
summary(svc1)
# 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

# 7 support vectors (4 from one class, 3 from the other)
# shown as X (color identifies the class)

# svc with cost 0.10
#========================================================
# if cost is smaller, margin becomes wider
svc2=svm(y~., data=dat, kernel="linear", cost=0.1,scale=F)
lim = c(-2.5,3)
plot(svc2, dat,xlim=lim,ylim=lim)
grid()

summary(svc2)

# 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

# 16 support vectors (8 in each class)

# cross validation on cost
#============================================================
# default is 10-fold

set.seed(1)
# cost values to cross validate
aux = c(0.001, 0.01, 0.1, 1,5,10,100)
tune.out=tune(svm,y~.,data=dat,kernel="linear",ranges=list(cost=aux))

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  0.4216370
# 2 1e-02  0.70  0.4216370
```

```
# 3 1e-01  0.10  0.2108185   best
# 4 1e+00  0.15  0.2415229
# 5 5e+00  0.15  0.2415229
# 6 1e+01  0.15  0.2415229
# 7 1e+02  0.15  0.2415229

# cost 0.10 is best model
# fit best model
bestmod=tune.out$best.model

# test set
#============================================================
w1= rnorm(20)
w2= rnorm(20)
k = 1
ytest=sample(c(-k,k), 20, rep=T)
table(ytest)
ytest
# -1  1
# 11  9

# shift
x1test = w1+ytest
x2test = w2+ytest
d1 = data.frame(x1test,x2test,ytest)

plot(x2test,x1test,col=(3-ytest),pch=19,cex=0.75,xlim=bound,ylim=bound)
grid()
text(x2test,x1test,labels=rownames(d1),offset=0.25,pos=1,cex=0.6)


#xtest=matrix(rnorm(20*2), ncol=2)
#ytest=sample(c(-1,1), 20, rep=T)
#xtest[ytest==1,]=xtest[ytest==1,] + k

dtest=data.frame(x1=x1test,x2=x2test,y=ytest)
dtest$y=factor(dtest$y)

plot(x1~x2,dtest,col=(3-ytest),pch=19,cex=0.75,xlim=bound,ylim=bound)
grid()
text(x1~x2,dtest,labels=rownames(dtest),offset=0.25,pos=1,cex=0.6)

# predict response y (class labels)
ypred=predict(bestmod,dtest)

table(prediction=ypred,dtest$y)

# prediction -1  1
#         -1 11  1
#         1   0  8
```

```
# predictions are rows
# one obs misclassified
d4 = data.frame(dtest,ypred)
# in row 4, y=1 was predicted as y=-1

# moving out from optimal cost
#==========================================================

# cost = 0.01
svc4=svm(y~., data=dat, kernel="linear", cost=.01,scale=F)
ypred=predict(svc4,dtest)
table(predict=ypred,dtest$y)

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

# one additional obs is misclassified


# 5-fold cross validation - use tune.control argument
#==========================================================
set.seed(1)
aux2 = tune.control(cross=6)
tune.out=tune(svm,y~.,data=dat,kernel="linear",ranges=list(cost=aux),tunecontrol=aux2)
summary(tune.out)

# Parameter tuning of svm:
# - sampling method: 5-fold cross validation
# - best parameters:
#  cost
#     5
# - best performance: 0.15
# - Detailed performance results:
#    cost error dispersion
# 1 1e-03  0.65  0.2850439
# 2 1e-02  0.65  0.2850439
# 3 1e-01  0.35  0.2850439
# 4 1e+00  0.25  0.1767767
# 5 5e+00  0.15  0.1369306
# 6 1e+01  0.15  0.1369306
# 7 1e+02  0.15  0.1369306

# cost 5 is best
# thus, 5-fold cv is not good

# New dataset barely linearly separable
#==========================================================
k = 1.5
```

```
aux=c(rep(0,10), rep(k,10))
x1 = n1+aux
x2 = n2+aux
plot(x2~x1, col=(3-yy),pch=19,cex=0.75)
grid()
dat=data.frame(y,x1,x2)

# huge cost, very narrow margin
svc5=svm(y~., data=dat, kernel="linear", cost=1e5)
summary(svc5)

# 3 support vectors

plot(svc5, dat)
grid()
plot(svc5,dat,xlim=c(-3,4),ylim=c(-2,4)); grid()
# margin very narrow since obs "0" very close to boundary
# no misclassifications


# small cost, wider margin
svc6=svm(y~., data=dat, kernel="linear", cost=1)
summary(svc6)
plot(svc6, dat,xlim=c(-3,4),ylim=c(-2,4)); grid()

# 1 misclassified
# margin very wide since support vectors far from boundary
```

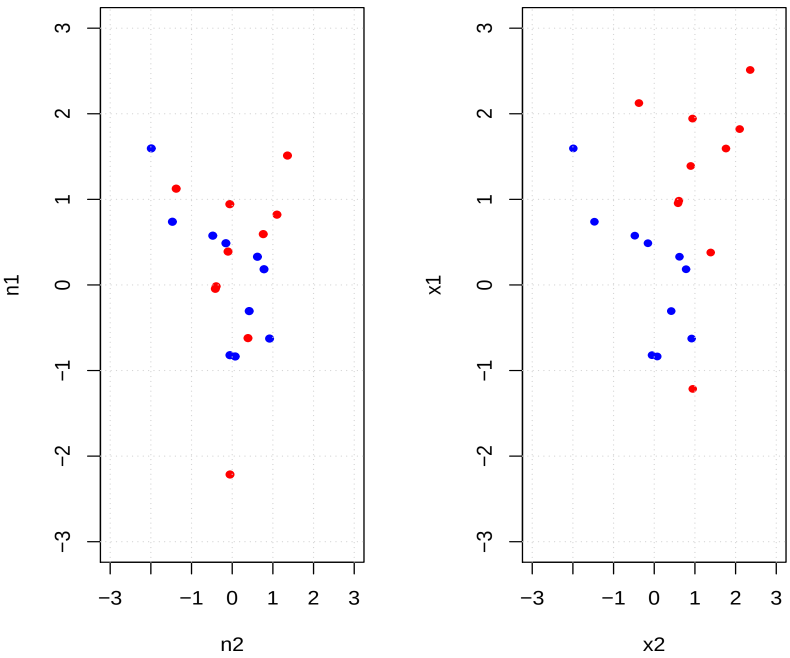

Figure 1: Training set

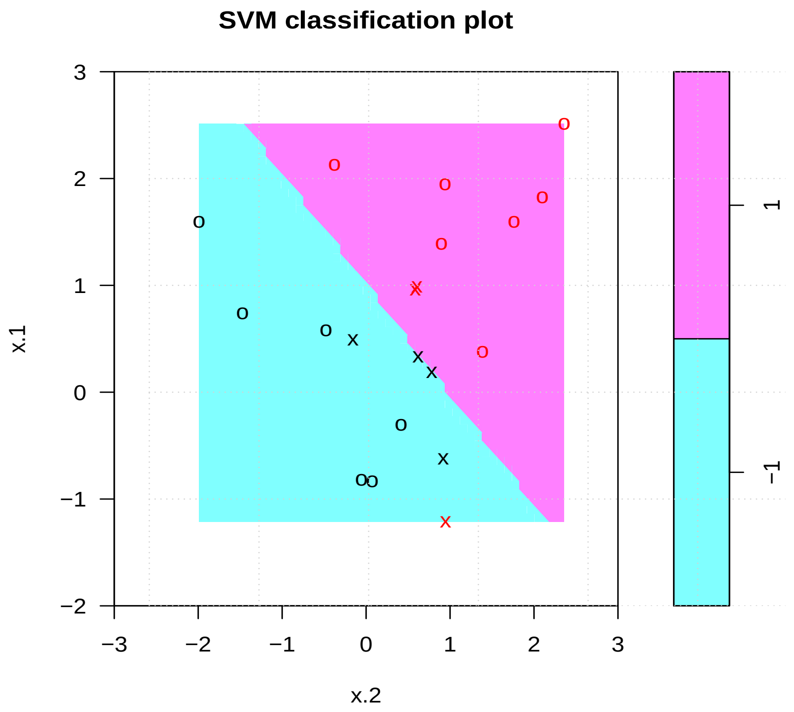

Figure 2: Support vector classifier svc1 on nonlinear separable data (cost 10)

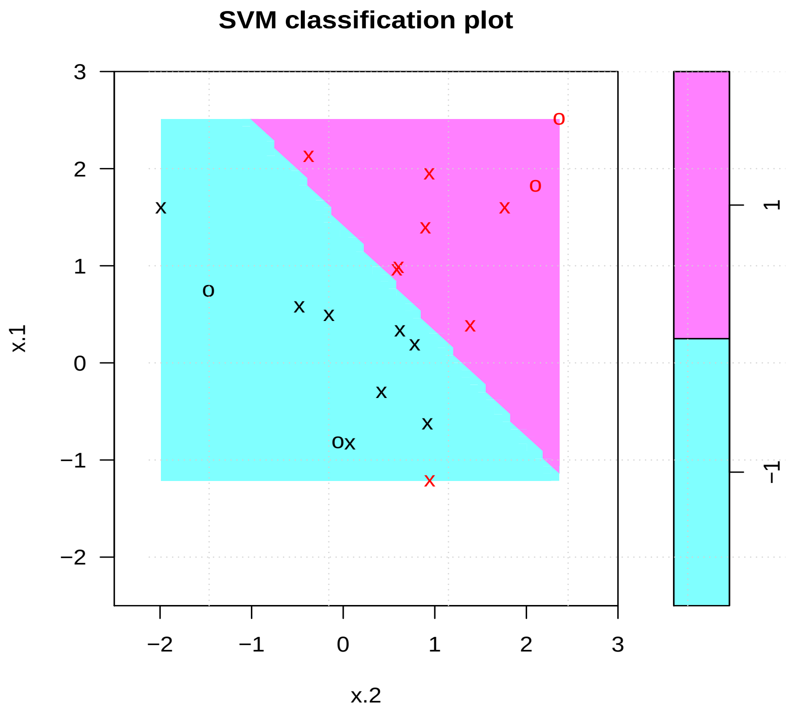

Figure 3: Support vector classifier svc2 on nonlinear separable data (cost 0.10)

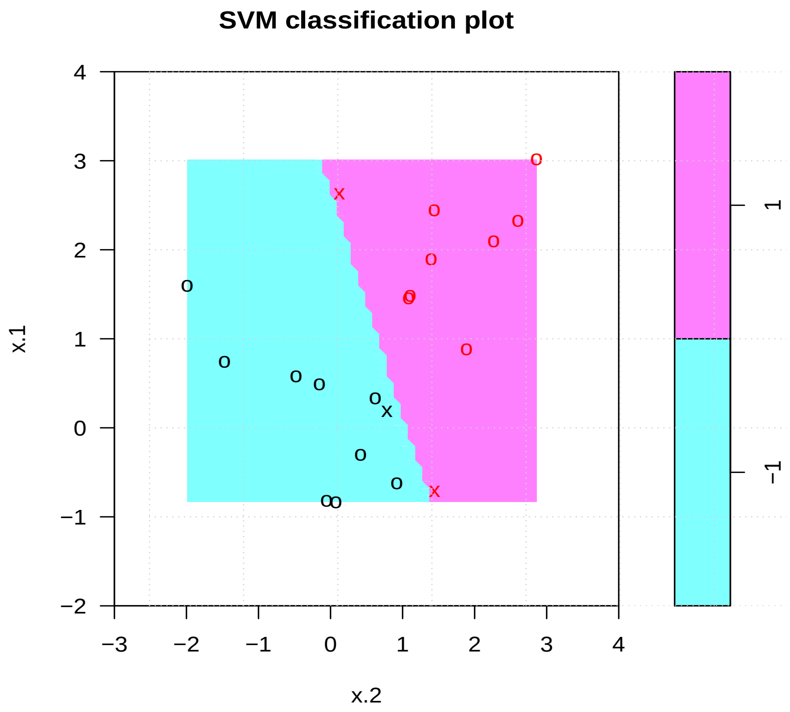

Figure 4: svc5 with on barely linear separable data (cost 1e5, narrow margins, few support vectors)

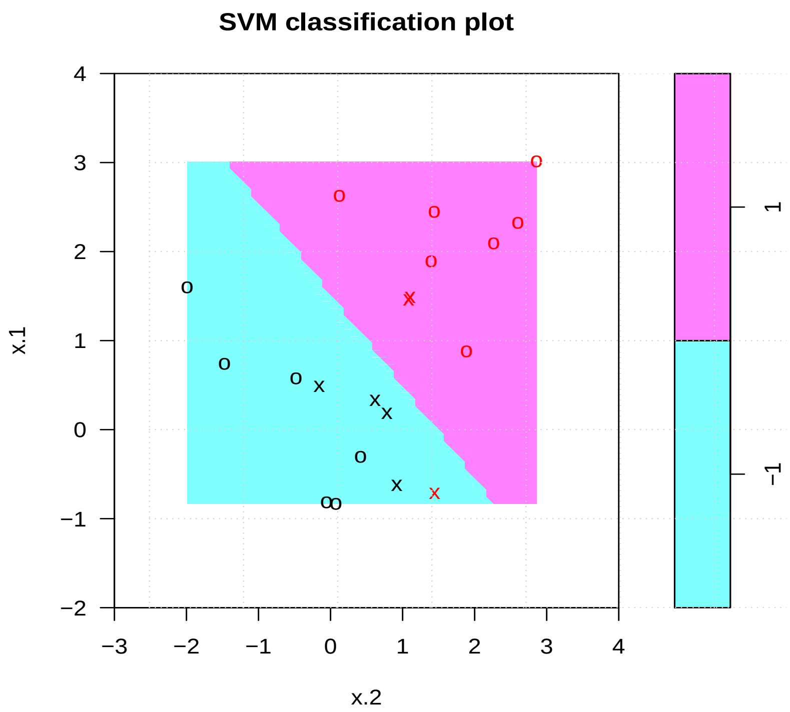

Figure 5: svc6 with on barely linear separable data (cost 1, wide margins, many support vectors)