JieTang_Lab8.R

tjj

2020-08-25

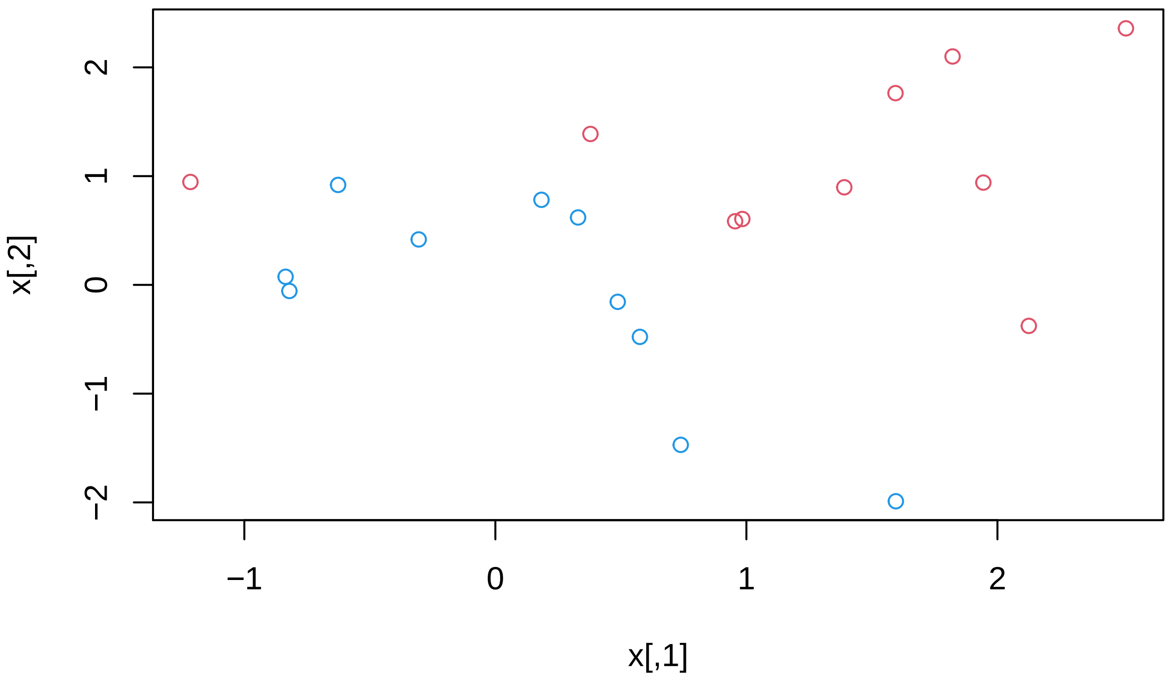
```
#Name:Jie Tang
#Course:Machine learning using R 374815
#Quarter:Summer
#Instructor name : Michael Chang

# Chapter 9 Lab: Support Vector Machines

# Support Vector Classifier
#Quiz part
#Q1
RNGkind(sample.kind = "Rounding")
```

```
## Warning in RNGkind(sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

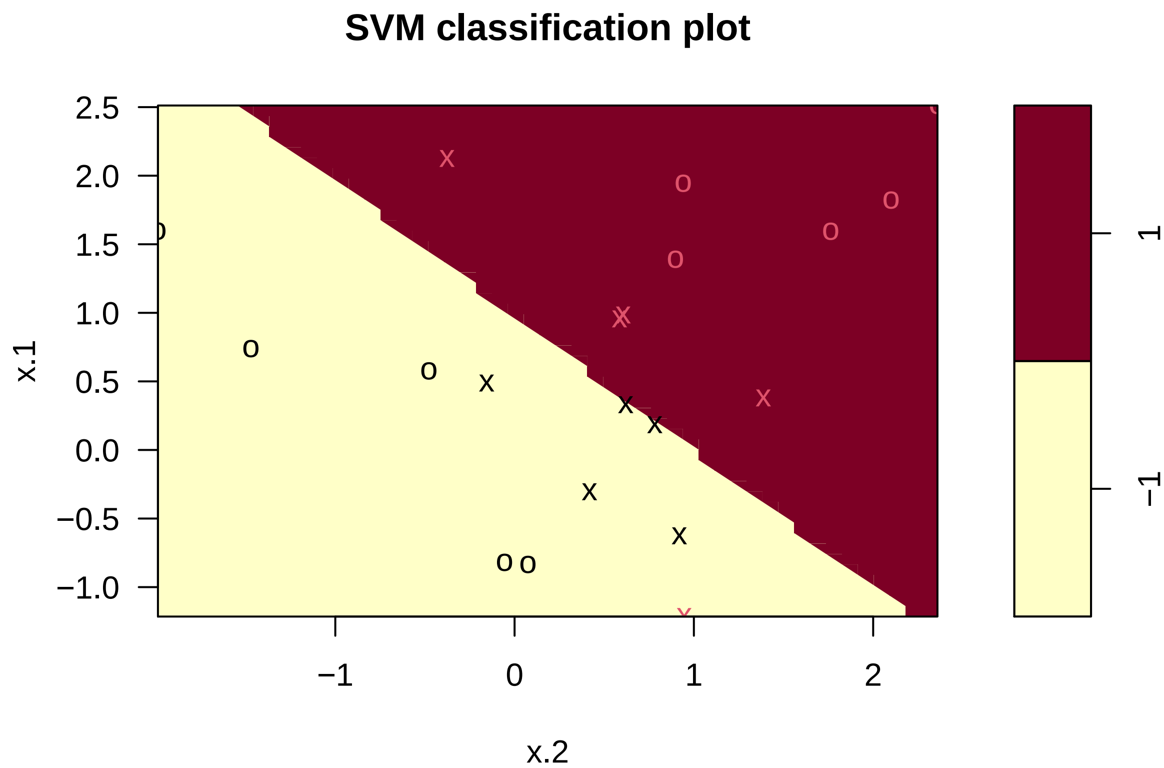
```
set.seed(1)
RNGkind(sample.kind = "Rounding")
```

```
## Warning in RNGkind(sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
x=matrix(rnorm(20*2), ncol=2)
y=c(rep(-1,10), rep(1,10))
x[y==1,]=x[y==1,] + 1
plot(x, col=(3-y))
```




```
dat=data.frame(x=x, y=as.factor(y))
library(e1071)
svmfit=svm(y~., data=dat, kernel="linear", cost=1,scale=FALSE)
plot(svmfit, dat)
```




```r
svmfit$index
```

```
##  [1]  1  2  5  7 10 13 14 15 16 17
```

```r
summary(svmfit)#get the number by summary()
```

```
##
## Call:
## svm(formula = y ~ ., data = dat, kernel = "linear", cost = 1, scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  10
##
##  ( 5 5 )
##
##
## Number of Classes:  2
##
## Levels:
##  -1 1
```

```r
#Q2
#find highest error rate
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: 0.05
##
## - Detailed performance results:
##    cost error dispersion
## 1 1e-03  0.65  0.3374743
## 2 1e-02  0.65  0.3374743
## 3 1e-01  0.05  0.1581139
## 4 1e+00  0.10  0.2108185
## 5 5e+00  0.15  0.2415229
## 6 1e+01  0.15  0.2415229
## 7 1e+02  0.15  0.2415229
```

```r
#Q3
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
##
## 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  6 5
##      1   2 7
```

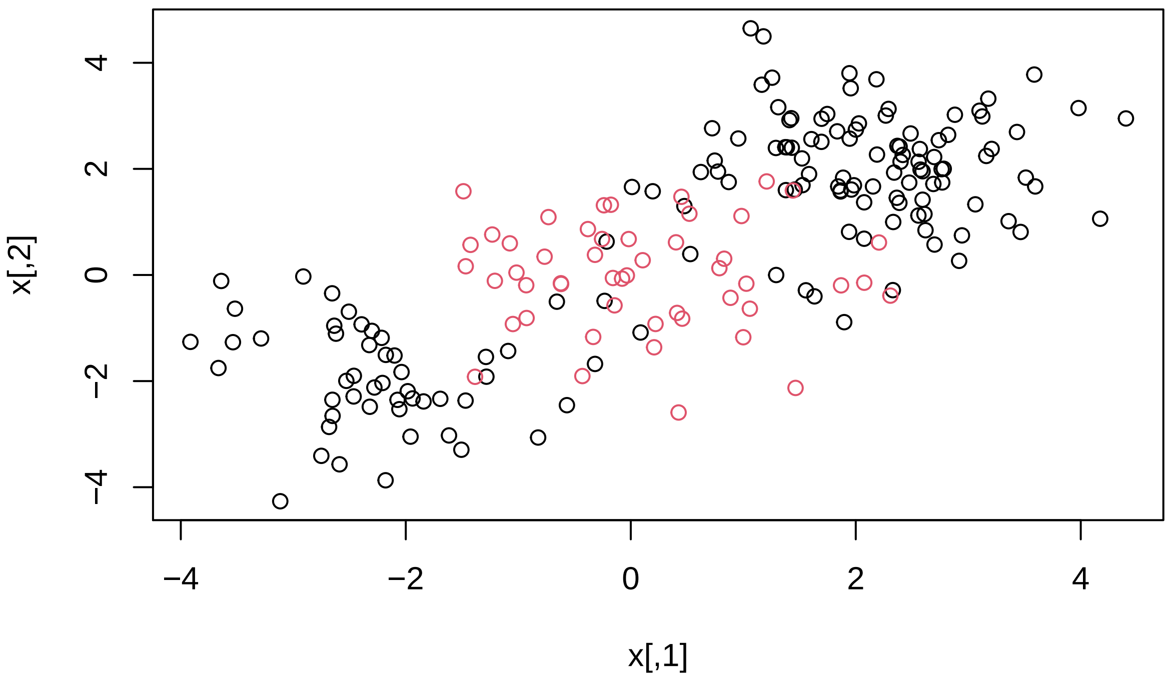
```
#Q4
RNGkind(sample.kind = "Rounding")
```

```
## Warning in RNGkind(sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

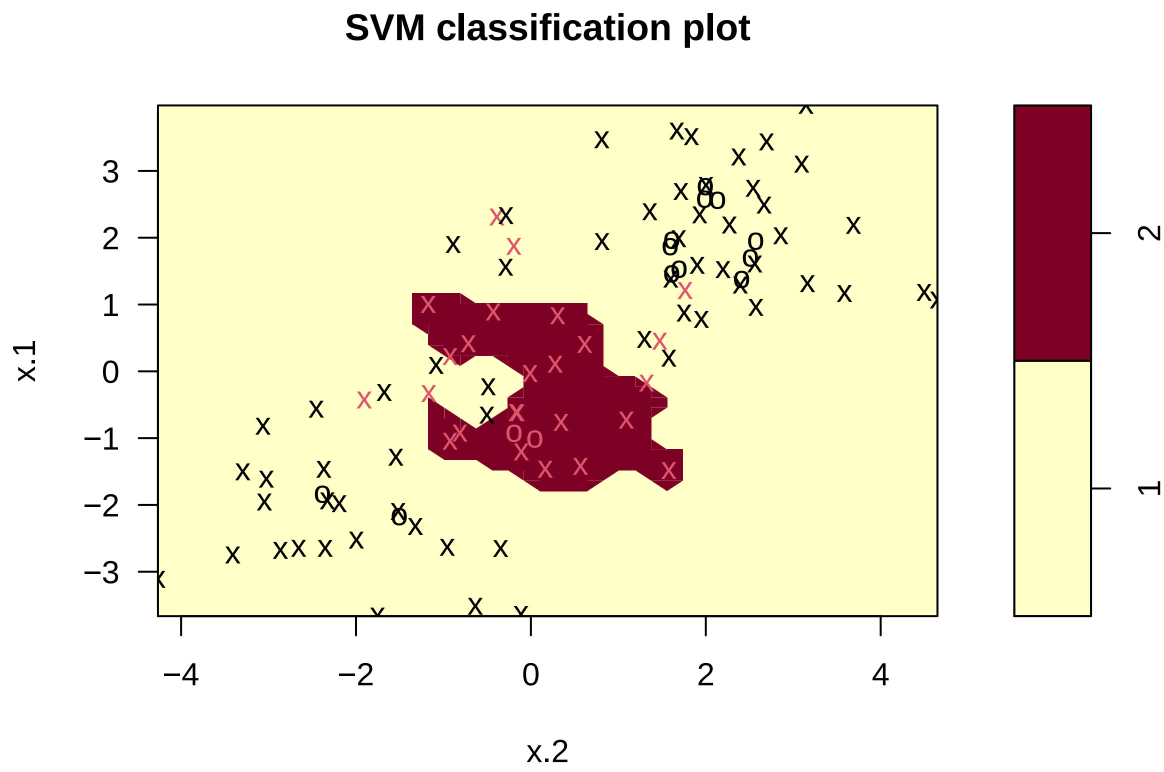
```
set.seed(1)
RNGkind(sample.kind = "Rounding")
```

```
## Warning in RNGkind(sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
x=matrix(rnorm(200*2), ncol=2)
x[1:100,]=x[1:100,]+2
x[101:150,]=x[101:150,]-2
y=c(rep(1,150),rep(2,50))
dat=data.frame(x=x,y=as.factor(y))
plot(x, col=y)
```



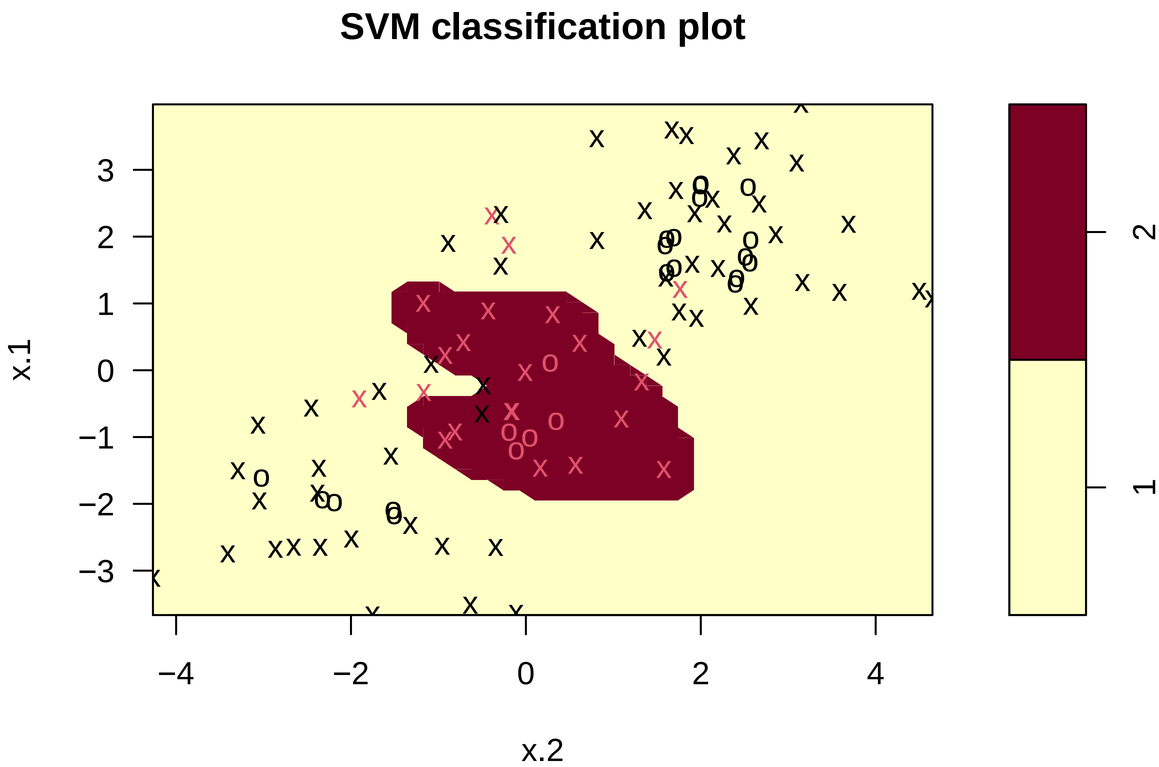

```r
train=sample(200,100)
#gamma = 20
svmfit=svm(y~., data=dat[train,], kernel="radial",  gamma=20, cost=1)
plot(svmfit, dat[train,])
```




```
summary(svmfit)
```

```
##
## Call:
## svm(formula = y ~ ., data = dat[train, ], kernel = "radial", gamma = 20,
##     cost = 1)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  86
##
##  ( 25 61 )
##
##
## Number of Classes:  2
##
## Levels:
##  1 2
```

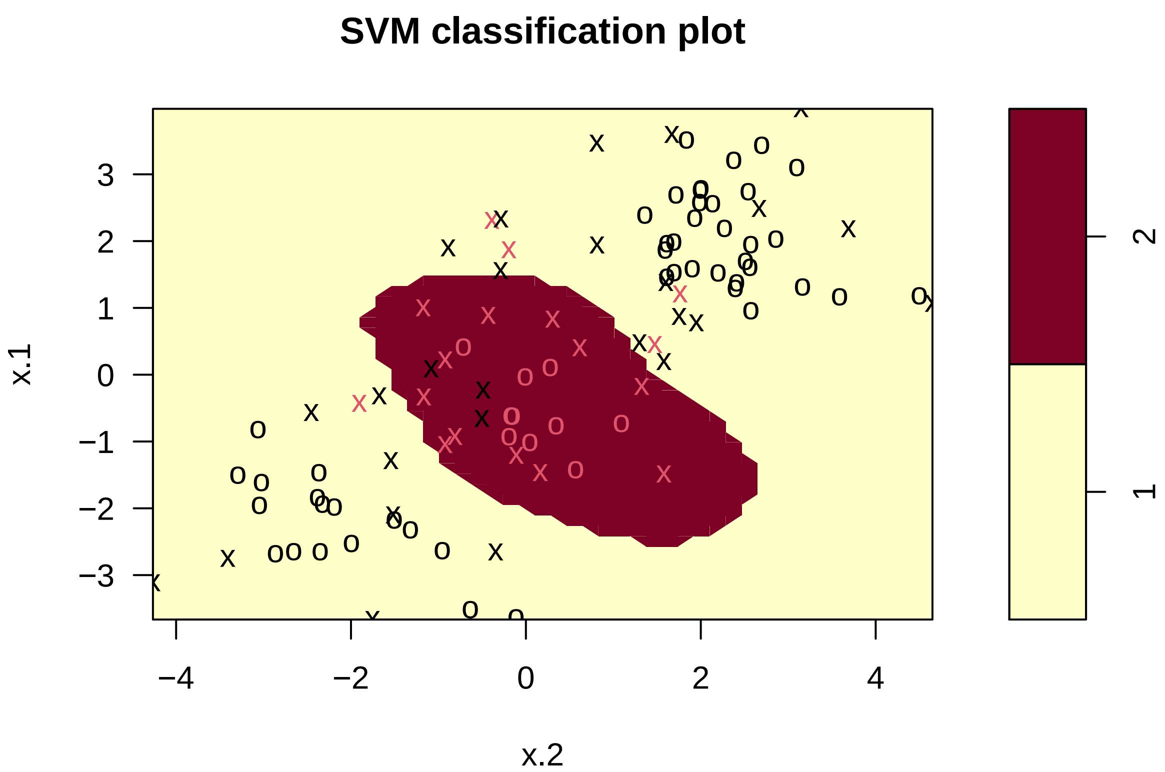
```
#gamma = 10
svmfit=svm(y~., data=dat[train,], kernel="radial",  gamma=10, cost=1)
plot(svmfit, dat[train,])
```




```r
summary(svmfit)
```

```
##
## Call:
## svm(formula = y ~ ., data = dat[train, ], kernel = "radial", gamma = 10,
##     cost = 1)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  76
##
##  ( 22 54 )
##
##
## Number of Classes:  2
##
## Levels:
##  1 2
```

```r
#gamma = 2
svmfit=svm(y~., data=dat[train,], kernel="radial",  gamma=2, cost=1)
plot(svmfit, dat[train,])
```

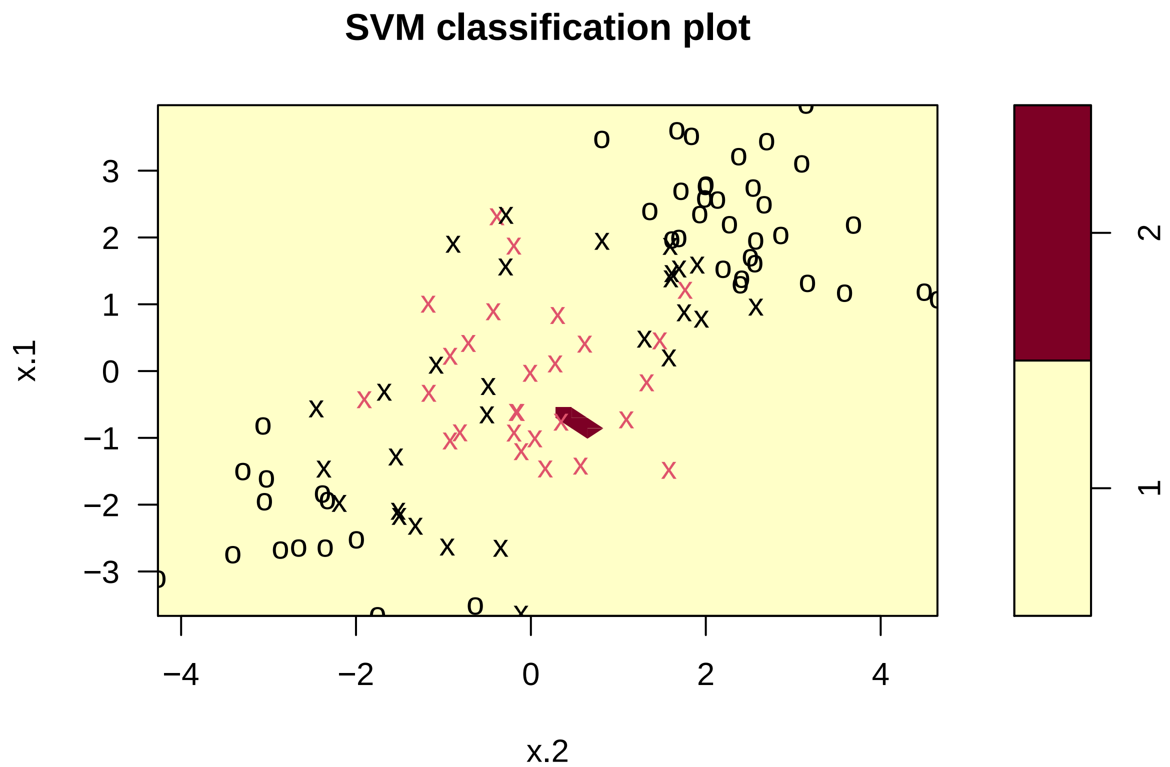



```r
summary(svmfit)
```

```
##
## Call:
## svm(formula = y ~ ., data = dat[train, ], kernel = "radial", gamma = 2,
##     cost = 1)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  43
##
##  ( 17 26 )
##
##
## Number of Classes:  2
##
## Levels:
```

```
##  1 2
```

```r
#gamma = 0.1
svmfit=svm(y~., data=dat[train,], kernel="radial",  gamma=0.1, cost=1)
plot(svmfit, dat[train,])
```




```r
summary(svmfit)
```

```
##
## Call:
## svm(formula = y ~ ., data = dat[train, ], kernel = "radial", gamma = 0.1,
##     cost = 1)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  55
##
##  ( 27 28 )
##
##
## Number of Classes:  2
```

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

```r
#Q5
RNGkind(sample.kind = "Rounding")
```

```
## Warning in RNGkind(sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```r
set.seed(1)
RNGkind(sample.kind = "Rounding")
```

```
## Warning in RNGkind(sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

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

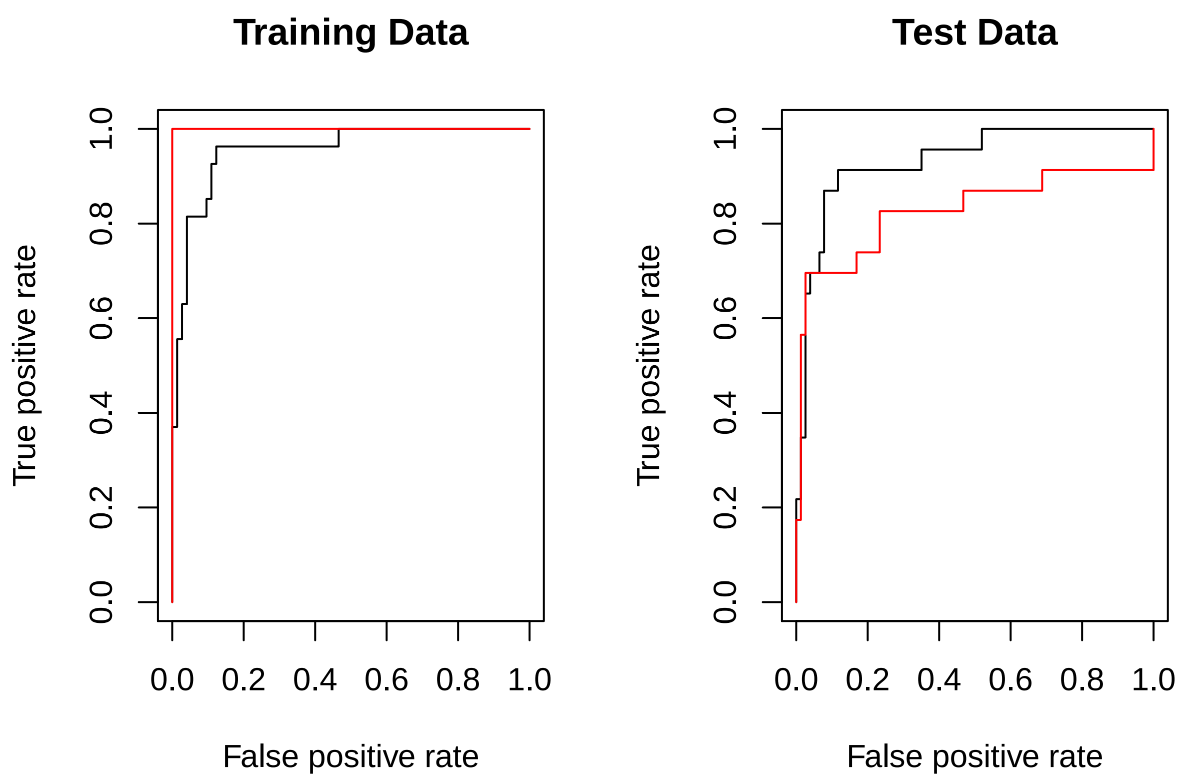
```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##    10   0.1
##
## - best performance: 0.12
##
## - Detailed performance results:
##     cost gamma error dispersion
## 1  1e-01   0.1  0.27 0.11595018
## 2  1e+00   0.1  0.26 0.12649111
## 3  1e+01   0.1  0.12 0.07888106
## 4  1e+02   0.1  0.12 0.07888106
## 5  1e+03   0.1  0.16 0.06992059
## 6  1e-01   0.5  0.27 0.11595018
## 7  1e+00   0.5  0.13 0.08232726
## 8  1e+01   0.5  0.15 0.07071068
## 9  1e+02   0.5  0.17 0.08232726
## 10 1e+03   0.5  0.21 0.09944289
## 11 1e-01   1.0  0.25 0.13540064
## 12 1e+00   1.0  0.13 0.08232726
## 13 1e+01   1.0  0.16 0.06992059
## 14 1e+02   1.0  0.20 0.09428090
## 15 1e+03   1.0  0.20 0.08164966
## 16 1e-01   1.5  0.25 0.13540064
## 17 1e+00   1.5  0.12 0.09189366
## 18 1e+01   1.5  0.17 0.09486833
## 19 1e+02   1.5  0.16 0.09660918
## 20 1e+03   1.5  0.22 0.12292726
```

```
## 21 1e-01   2.0  0.25 0.12692955
## 22 1e+00   2.0  0.12 0.09189366
## 23 1e+01   2.0  0.17 0.09486833
## 24 1e+02   2.0  0.19 0.09944289
## 25 1e+03   2.0  0.20 0.09428090
## 26 1e-01   3.0  0.27 0.11595018
## 27 1e+00   3.0  0.13 0.09486833
## 28 1e+01   3.0  0.18 0.10327956
## 29 1e+02   3.0  0.21 0.08755950
## 30 1e+03   3.0  0.22 0.10327956
## 31 1e-01   4.0  0.27 0.11595018
## 32 1e+00   4.0  0.15 0.10801234
## 33 1e+01   4.0  0.18 0.11352924
## 34 1e+02   4.0  0.21 0.08755950
## 35 1e+03   4.0  0.24 0.10749677
## 36 1e-01   5.0  0.27 0.11595018
## 37 1e+00   5.0  0.16 0.10749677
## 38 1e+01   5.0  0.19 0.09944289
## 39 1e+02   5.0  0.21 0.09944289
## 40 1e+03   5.0  0.24 0.09660918
```

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

```
##     pred
## true  1  2
##    1 71  6
##    2  6 17
```

```r
#Q6
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")
```



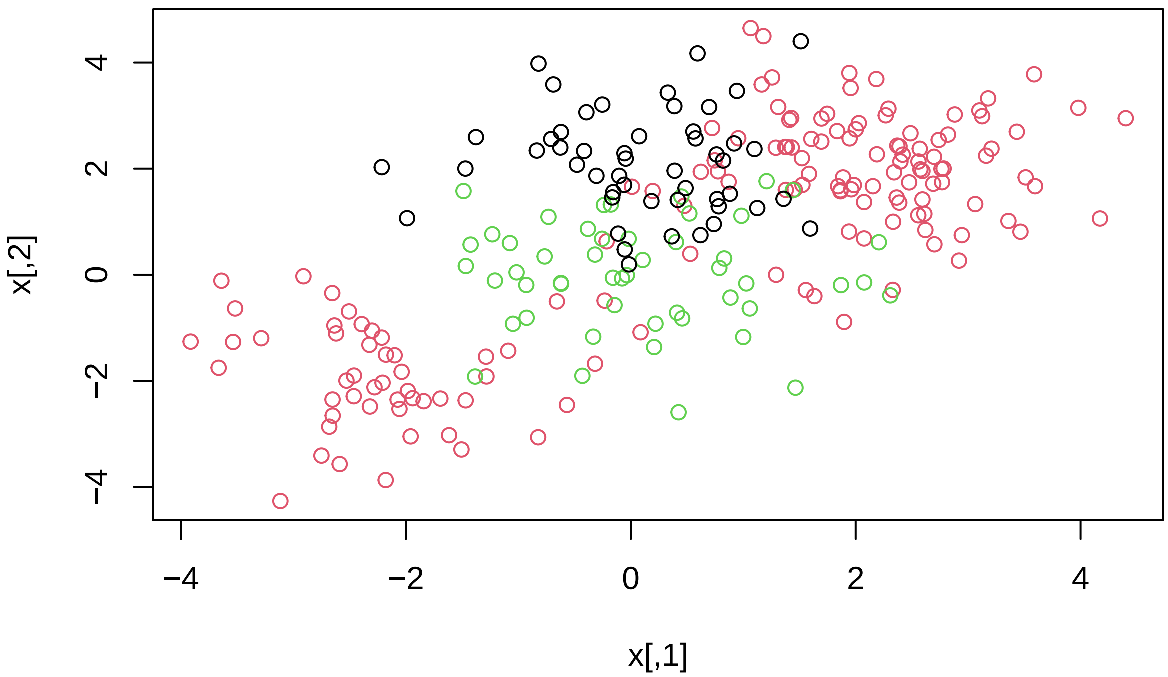

```
#Q7
#no big change shown by graph
RNGkind(sample.kind = "Rounding")
```

```
## Warning in RNGkind(sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
set.seed(1)
RNGkind(sample.kind = "Rounding")
```

```
## Warning in RNGkind(sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

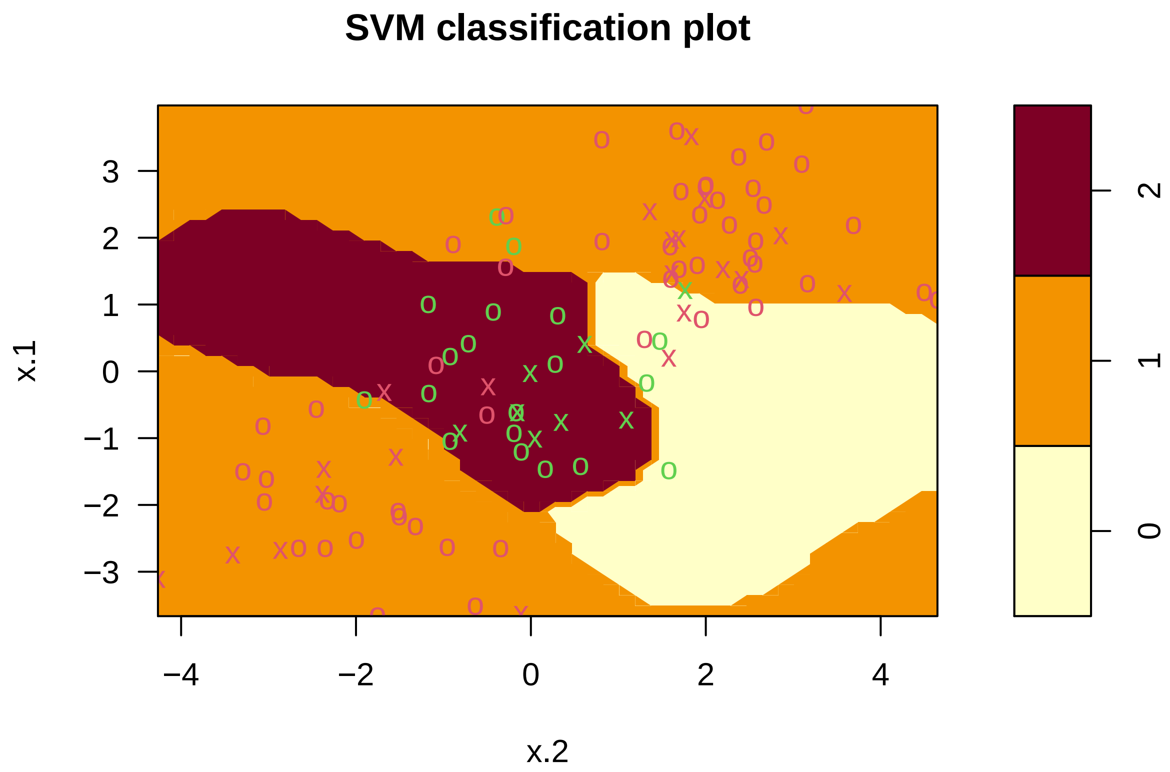
```
x=rbind(x, matrix(rnorm(50*2), ncol=2))
y=c(y, rep(0,50))
x[y==0,2]=x[y==0,2]+2
dat=data.frame(x=x, y=as.factor(y))
par(mfrow=c(1,1))
plot(x,col=(y+1))
```




```r
svmfit=svm(y~., data=dat, kernel="radial", cost=5, gamma=1)
summary(svmfit)
```

```
##
## Call:
## svm(formula = y ~ ., data = dat, kernel = "radial", cost = 5, gamma = 1)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  5
##
## Number of Support Vectors:  105
##
##  ( 40 33 32 )
##
##
## Number of Classes:  3
##
## Levels:
##  0 1 2
```

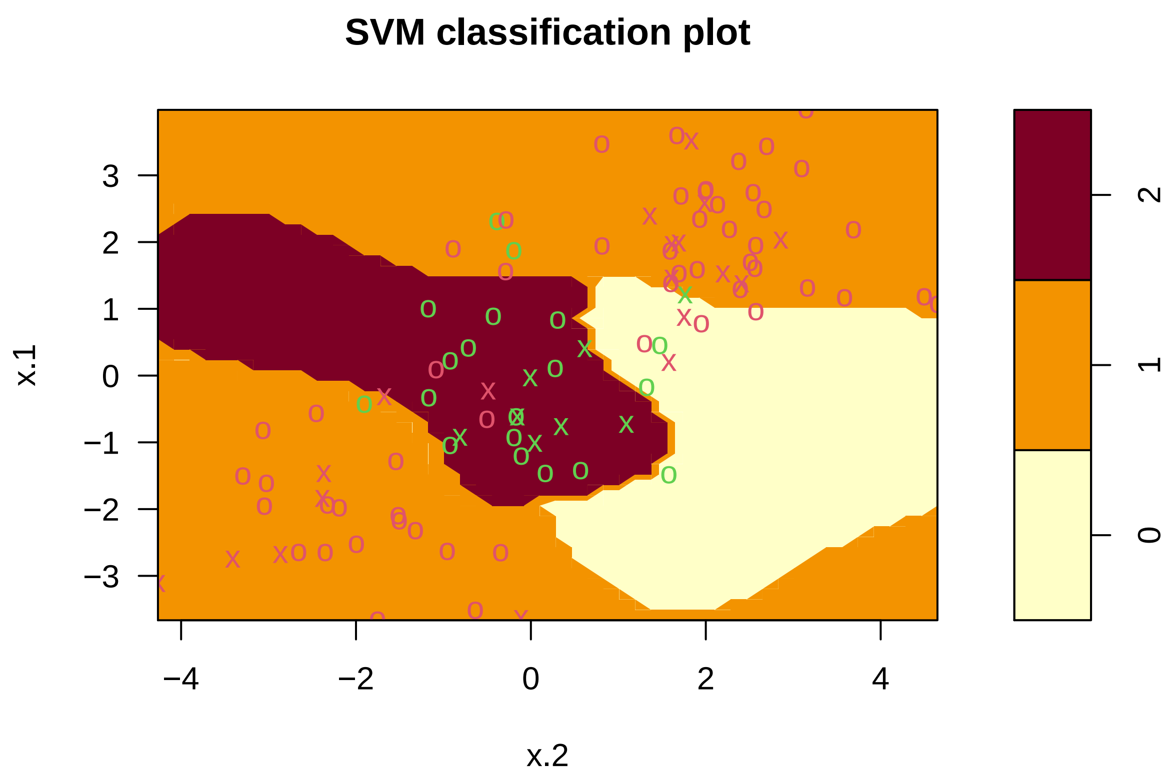
```r
plot(svmfit, dat[train,])
```




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

```
##
## Call:
## svm(formula = y ~ ., data = dat, kernel = "radial", cost = 10, gamma = 1)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  10
##
## Number of Support Vectors:  105
##
##  ( 38 37 30 )
##
##
## Number of Classes:  3
##
## Levels:
##  0 1 2
```

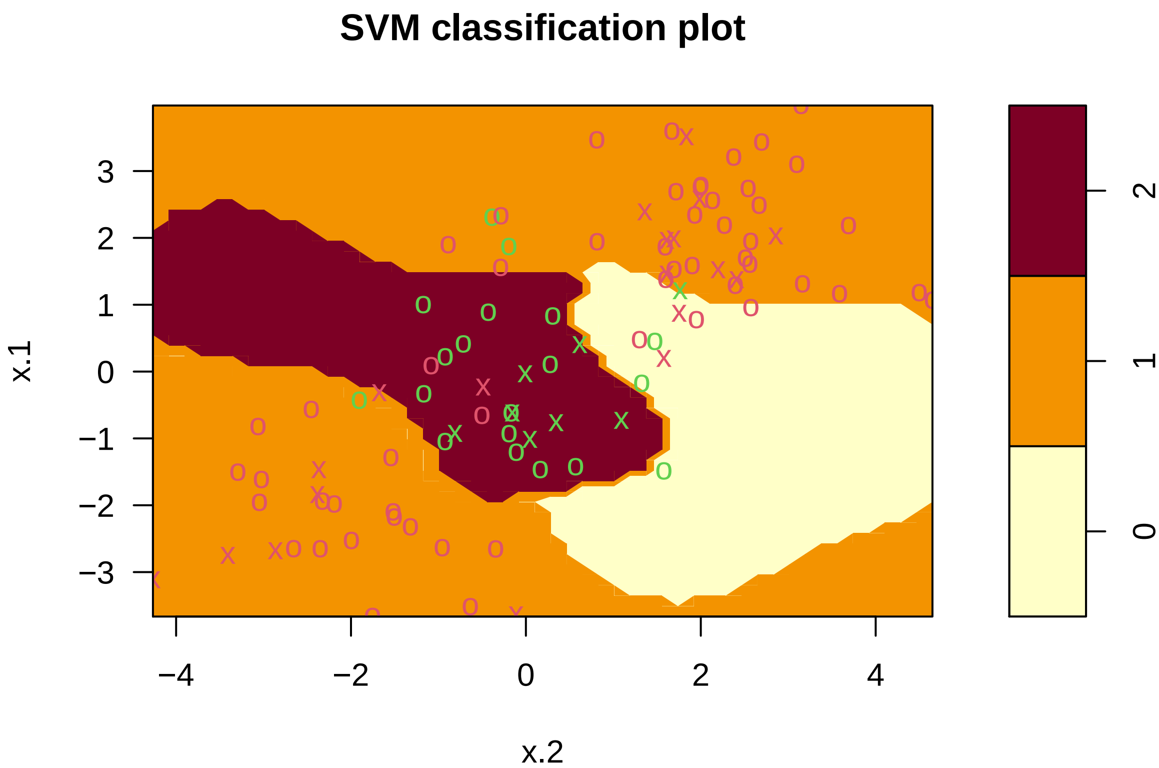
```r
plot(svmfit, dat[train,])
```




```
svmfit=svm(y~., data=dat, kernel="radial", cost=15, gamma=1)
summary(svmfit)
```

```
##
## Call:
## svm(formula = y ~ ., data = dat, kernel = "radial", cost = 15, gamma = 1)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  15
##
## Number of Support Vectors:  102
##
##  ( 38 34 30 )
##
##
## Number of Classes:  3
##
## Levels:
##  0 1 2
```

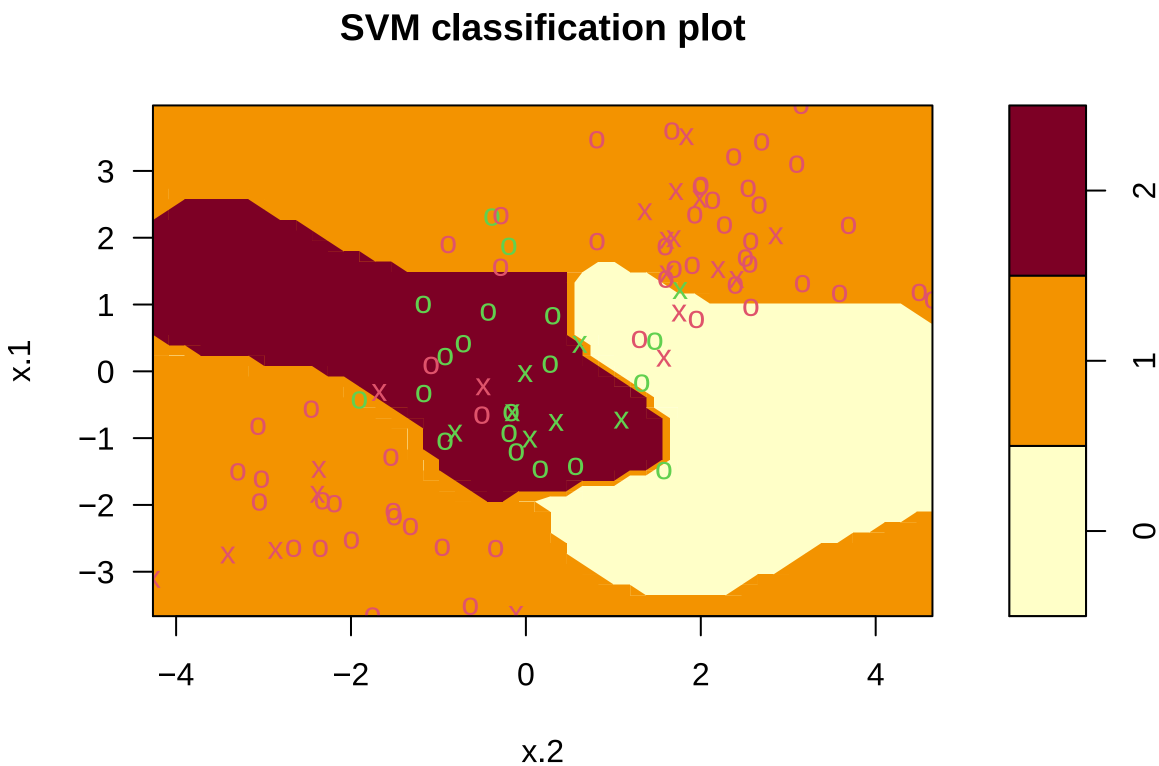
```
plot(svmfit, dat[train,])
```




```r
svmfit=svm(y~., data=dat, kernel="radial", cost=20, gamma=1)
summary(svmfit)
```

```
##
## Call:
## svm(formula = y ~ ., data = dat, kernel = "radial", cost = 20, gamma = 1)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  20
##
## Number of Support Vectors:  99
##
##  ( 39 32 28 )
##
##
## Number of Classes:  3
##
## Levels:
##  0 1 2
```

```r
plot(svmfit, dat[train,])
```




```
#
# #Lab part
# set.seed(1)
# x=matrix(rnorm(20*2), ncol=2)
# y=c(rep(-1,10), rep(1,10))
# x[y==1,]=x[y==1,] + 1
# plot(x, col=(3-y))
# 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$index
# summary(svmfit)
# svmfit=svm(y~., data=dat, kernel="linear", cost=0.1,scale=FALSE)
# plot(svmfit, dat)
# svmfit$index
# 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)
# bestmod=tune.out$best.model
# summary(bestmod)
# 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$y)
# svmfit=svm(y~., data=dat, kernel="linear", cost=.01,scale=FALSE)
# ypred=predict(svmfit,testdat)
# table(predict=ypred, truth=testdat$y)
# 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)
# plot(svmfit, dat)
# svmfit=svm(y~., data=dat, kernel="linear", cost=1)
# summary(svmfit)
# plot(svmfit,dat)
#
# # 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
# y=c(rep(1,150),rep(2,50))
# dat=data.frame(x=x,y=as.factor(y))
# plot(x, col=y)
# train=sample(200,100)
# svmfit=svm(y~., data=dat[train,], kernel="radial",  gamma=1, cost=1)
# plot(svmfit, dat[train,])
# summary(svmfit)
# 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),gamm
# summary(tune.out)
# table(true=dat[-train,"y"], pred=predict(tune.out$best.model,newdata=dat[-train,]))
#
# # ROC Curves
#
# library(ROCR)
# rocplot=function(pred, truth, ...){
#   predob = prediction(pred, truth)
#   perf = performance(predob, "tpr", "fpr")
#   plot(perf,...)}
# svmfit.opt=svm(y~., data=dat[train,], kernel="radial",gamma=2, cost=1,decision.values=T)
# fitted=attributes(predict(svmfit.opt,dat[train,],decision.values=TRUE))$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")
#
```

```
# # 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)
#
# # Application to Gene Expression Data
#
# library(ISLR)
# names(Khan)
# dim(Khan$xtrain)
# dim(Khan$xtest)
# length(Khan$ytrain)
# length(Khan$ytest)
# table(Khan$ytrain)
# table(Khan$ytest)
# dat=data.frame(x=Khan$xtrain, y=as.factor(Khan$ytrain))
# out=svm(y~., data=dat, kernel="linear",cost=10)
# summary(out)
# table(out$fitted, dat$y)
# 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)
```