hw3

Ying Xiang

3/19/2021

```
#1
library(e1071)
```

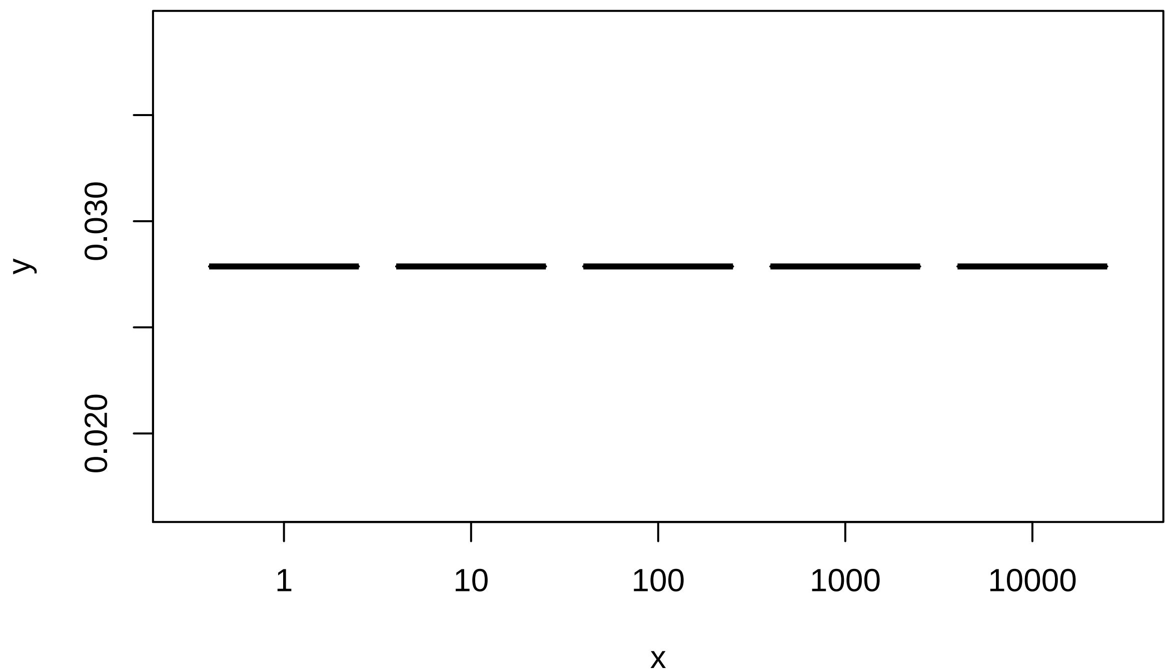
```
## Warning: package 'e1071' was built under R version 4.0.4
```

```
data5=read.table("train_5.txt",sep=",")
data6=read.table("train_6.txt",sep=",")
y5=rep(-1,nrow(data5))
y6=rep(1,nrow(data6))
wholedata=data.frame(label=as.factor(c(y5,y6)),rbind(data5,data6))
```

```
groupsize=nrow(wholedata)*0.2

index=sample(1:nrow(wholedata),size=nrow(wholedata))
grouprows1=index[1:groupsize]
grouprows2=index[(1+groupsize):(2*groupsize)]
grouprows3=index[(1+2*groupsize):(3*groupsize)]
grouprows4=index[(1+3*groupsize):(4*groupsize)]
grouprows5=index[(1+4*groupsize):(5*groupsize)]
```

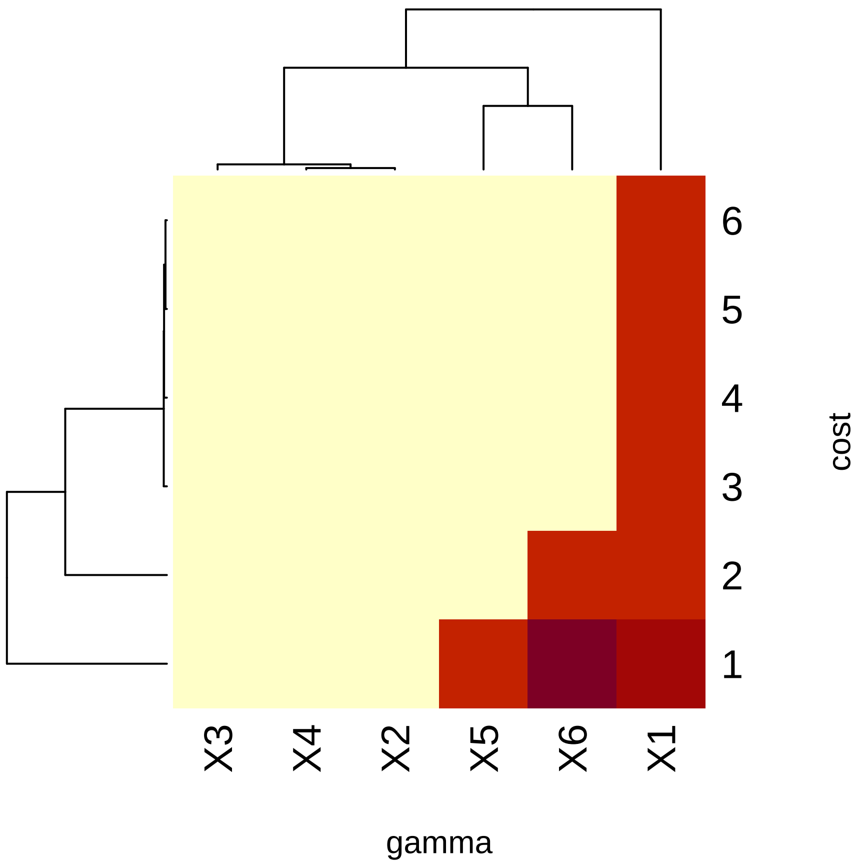
```
#1.1.a
options(warn = -1)
misclass_error_linear=rep(NA,5)
counter=1
cost=c("1","10","100","1000","10000")
for (i in 10^(0:4)){
  options(warn = -1)
  linearsvm1=svm(label~.,data=wholedata[-grouprows1,],kernel="linear",cost=i)
  linearsvm2=svm(label~.,data=wholedata[-grouprows2,],kernel="linear",cost=i)
  linearsvm3=svm(label~.,data=wholedata[-grouprows3,],kernel="linear",cost=i)
  linearsvm4=svm(label~.,data=wholedata[-grouprows4,],kernel="linear",cost=i)
  linearsvm5=svm(label~.,data=wholedata[-grouprows5,],kernel="linear",cost=i)
  misclass_error1=mean(wholedata[grouprows1,]$label!=predict(linearsvm1,wholedata[grouprows1,]))
  misclass_error2=mean(wholedata[grouprows2,]$label!=predict(linearsvm2,wholedata[grouprows2,]))
  misclass_error3=mean(wholedata[grouprows3,]$label!=predict(linearsvm3,wholedata[grouprows3,]))
  misclass_error4=mean(wholedata[grouprows4,]$label!=predict(linearsvm4,wholedata[grouprows4,]))
  misclass_error5=mean(wholedata[grouprows5,]$label!=predict(linearsvm5,wholedata[grouprows5,]))



  misclass_error_linear[counter]=mean(c(misclass_error1,misclass_error2,misclass_error3,misclass_error4
  counter=counter+1
}
```

```
plot(factor(cost),misclass_error_linear,type="l")
```




```
#1.1.b
misclass_error_radial=data.frame("6"=rep(NA,6),"5"=rep(NA,6),"4"=rep(NA,6),"3"=rep(NA,6),"2"=rep(NA,6),
c=10^(0:5)
g=10^(-6:-1)
for (i in (1:6)){
  for (j in (1:6)){

    rbfsvm1=svm(label~.,data=wholedata[-grouprows1,],kernel="radial",cost=c[i],gamma=g[j])
    rbfsvm2=svm(label~.,data=wholedata[-grouprows2,],kernel="radial",cost=c[i],gamma=g[j])
    rbfsvm3=svm(label~.,data=wholedata[-grouprows3,],kernel="radial",cost=c[i],gamma=g[j])
    rbfsvm4=svm(label~.,data=wholedata[-grouprows4,],kernel="radial",cost=c[i],gamma=g[j])
    rbfsvm5=svm(label~.,data=wholedata[-grouprows5,],kernel="radial",cost=c[i],gamma=g[j])
    misclass_error1=mean(wholedata[grouprows1,]$label!=predict(rbfsvm1,wholedata[grouprows1,]))
    misclass_error2=mean(wholedata[grouprows2,]$label!=predict(rbfsvm2,wholedata[grouprows2,]))
    misclass_error3=mean(wholedata[grouprows3,]$label!=predict(rbfsvm3,wholedata[grouprows3,]))
    misclass_error4=mean(wholedata[grouprows4,]$label!=predict(rbfsvm4,wholedata[grouprows4,]))
    misclass_error5=mean(wholedata[grouprows5,]$label!=predict(rbfsvm5,wholedata[grouprows5,]))

    misclass_error_radial[i,j]=mean(c(misclass_error1,misclass_error2,misclass_error3,misclass_error4,m
  }
}
```

```r
heatmap(as.matrix(misclass_error_radial),scale="none",xlab="gamma",ylab="cost")
```




```r
misclass_error_radial
```

```
##           X6         X5         X4         X3         X2        X1
## 1 0.45573770 0.37540984 0.02950820 0.01639344 0.02704918 0.3819672
## 2 0.37540984 0.02786885 0.01885246 0.01147541 0.02540984 0.3803279
## 3 0.02786885 0.01885246 0.02131148 0.01393443 0.02540984 0.3803279
## 4 0.01885246 0.02295082 0.02622951 0.01393443 0.02540984 0.3803279
## 5 0.02295082 0.02704918 0.02622951 0.01393443 0.02540984 0.3803279
## 6 0.02786885 0.02704918 0.02622951 0.01393443 0.02540984 0.3803279
```

```r
min(misclass_error_radial)
```

```
## [1] 0.01147541
```

```r
#1.2
#misclassification rate for linear case(cost=100)
misclass_error_linear
```

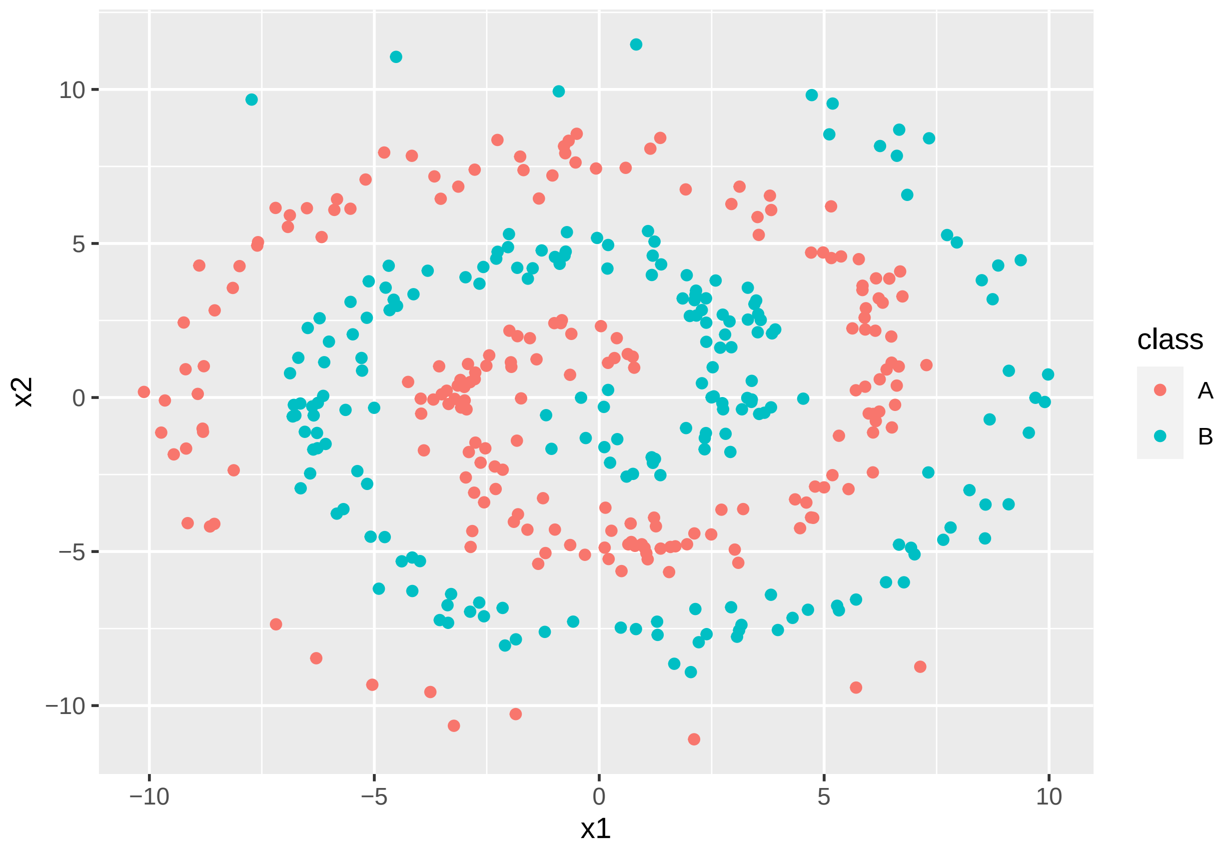
```
## [1] 0.02786885 0.02786885 0.02786885 0.02786885 0.02786885
```

```r
#misclassification rate for non-linear case(cost=100,gamma=10^-3)
min(misclass_error_radial)
```

```
## [1] 0.01147541
```

```r
#should use non-linear SVM.
```

```r
#2.5
data2=read.csv("HW3Problem2.csv")
library(ggplot2)
ggplot(data2,aes(x=x1,y=x2,color=class))+geom_point()
```




```r
data_2=data2
data_2$class=as.factor(c(rep(1,201),rep(-1,201)))
```

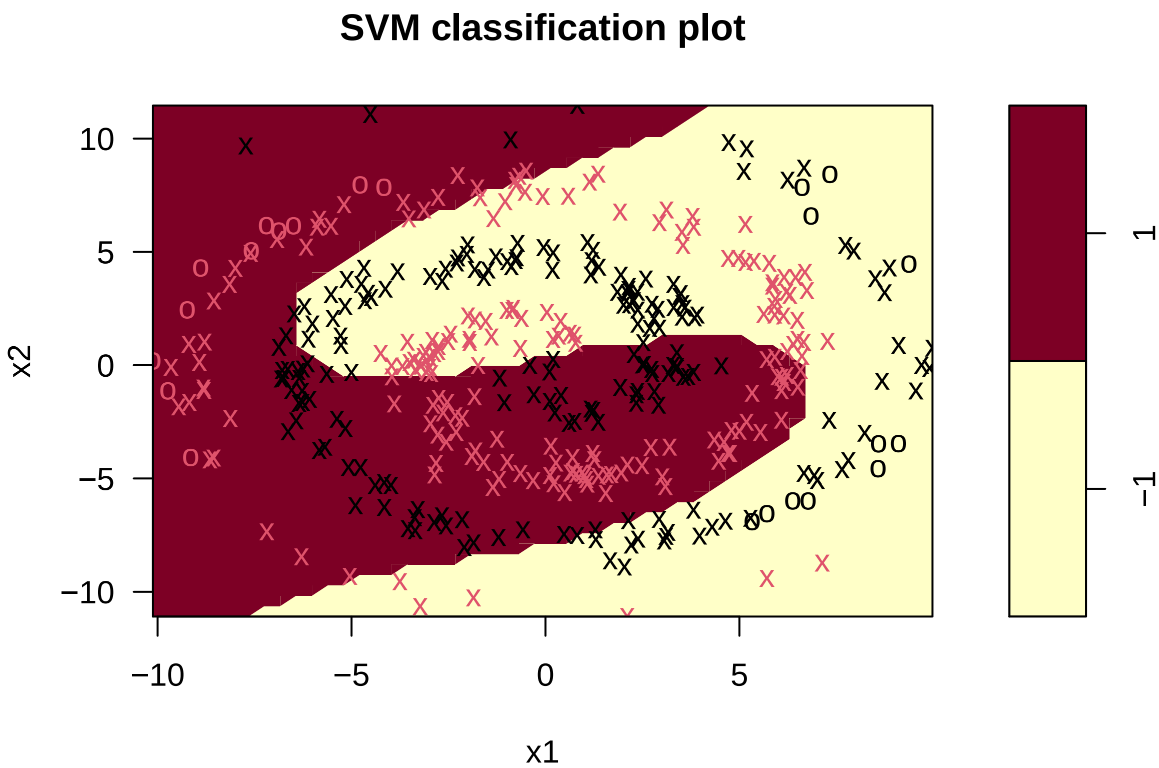
```r
#First, find best cost by using tune.svm()
tuned=tune.svm(class~.,data=data_2,gamma=10^(-4:2),cost=10^(0:3))
summary(tuned)
```

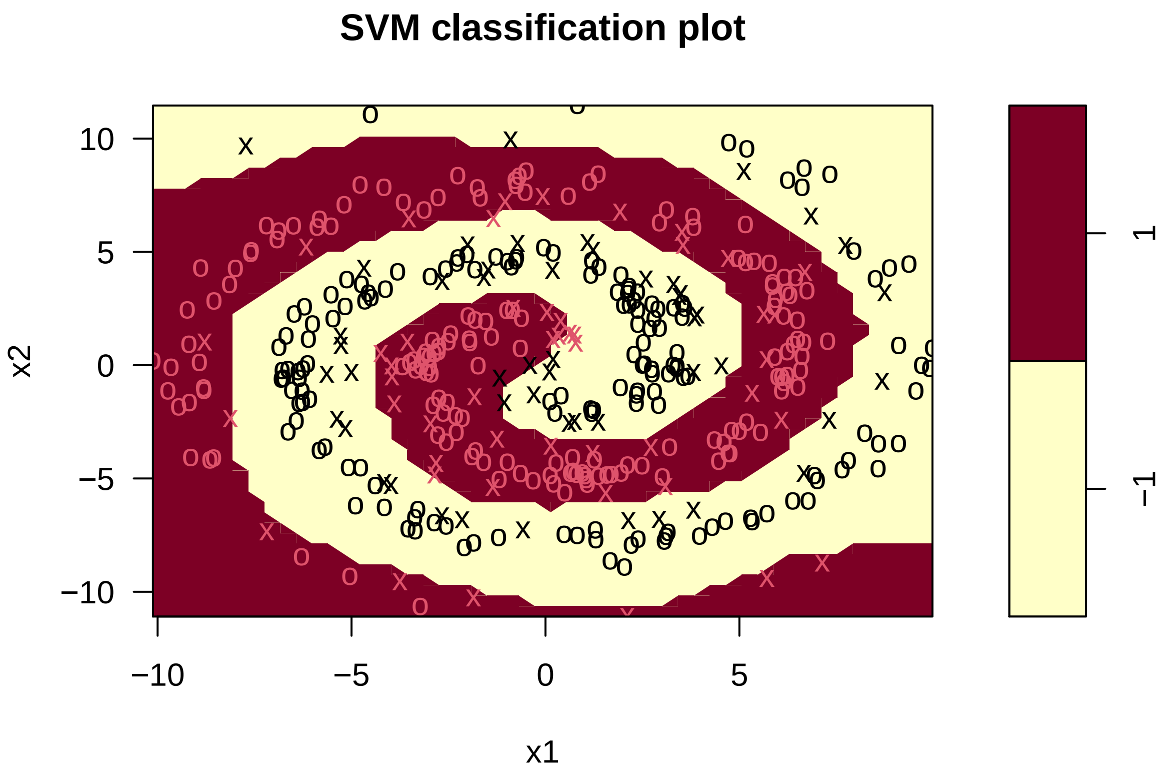
```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  gamma cost
##     10   10
##
## - best performance: 0.01493902
##
## - Detailed performance results:
##    gamma cost      error dispersion
## 1  1e-04    1 0.56719512 0.05989269
## 2  1e-03    1 0.56719512 0.05989269
```

```
## 3  1e-02    1 0.49762195 0.08550871
## 4  1e-01    1 0.55695122 0.09583983
## 5  1e+00    1 0.08475610 0.04139661
## 6  1e+01    1 0.01743902 0.02055802
## 7  1e+02    1 0.04481707 0.03691389
## 8  1e-04   10 0.56719512 0.05989269
## 9  1e-03   10 0.47768293 0.09287759
## 10 1e-02   10 0.48000000 0.07549646
## 11 1e-01   10 0.49250000 0.09979822
## 12 1e+00   10 0.02250000 0.02751262
## 13 1e+01   10 0.01493902 0.02412500
## 14 1e+02   10 0.04231707 0.03344494
## 15 1e-04  100 0.47768293 0.09287759
## 16 1e-03  100 0.45256098 0.05301240
## 17 1e-02  100 0.54713415 0.08774560
## 18 1e-01  100 0.33853659 0.06321419
## 19 1e+00  100 0.02237805 0.02484842
## 20 1e+01  100 0.01493902 0.02412500
## 21 1e+02  100 0.04231707 0.03344494
## 22 1e-04 1000 0.45018293 0.05236022
## 23 1e-03 1000 0.47993902 0.07589304
## 24 1e-02 1000 0.52743902 0.09558379
## 25 1e-01 1000 0.22390244 0.06106213
## 26 1e+00 1000 0.02719512 0.02951035
## 27 1e+01 1000 0.01493902 0.02412500
## 28 1e+02 1000 0.04231707 0.03344494
```

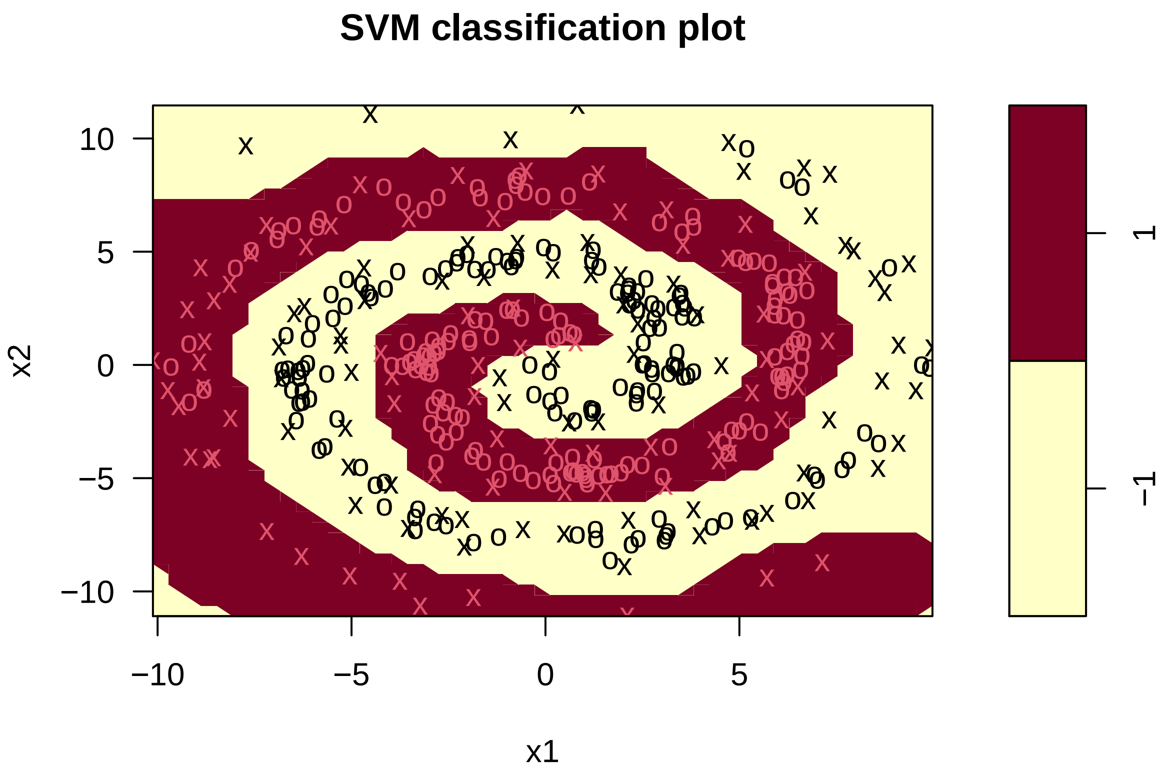
```r
#cost=10
```

```r
for (i in 10^(-1:2)){
  p2svm=svm(class~.,data=data_2,kernel="radial",cost=10,gamma=i)
plot(p2svm,data_2,x2~x1)}
```

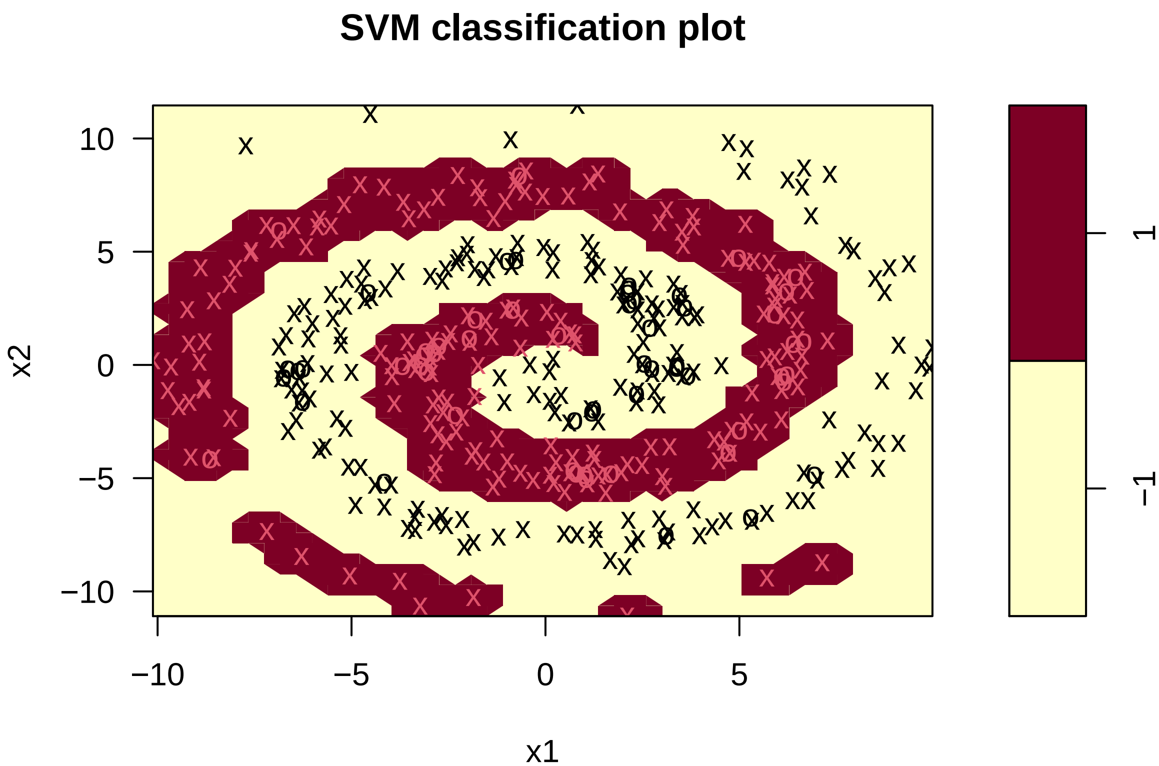

SVM classification plot
x2
10
5
0
−5
−10
−10
−5
0
5
x1
1
−1



SVM classification plot
x2
x1
10
5
0
−5
−10
−10
−5
0
5
1
−1

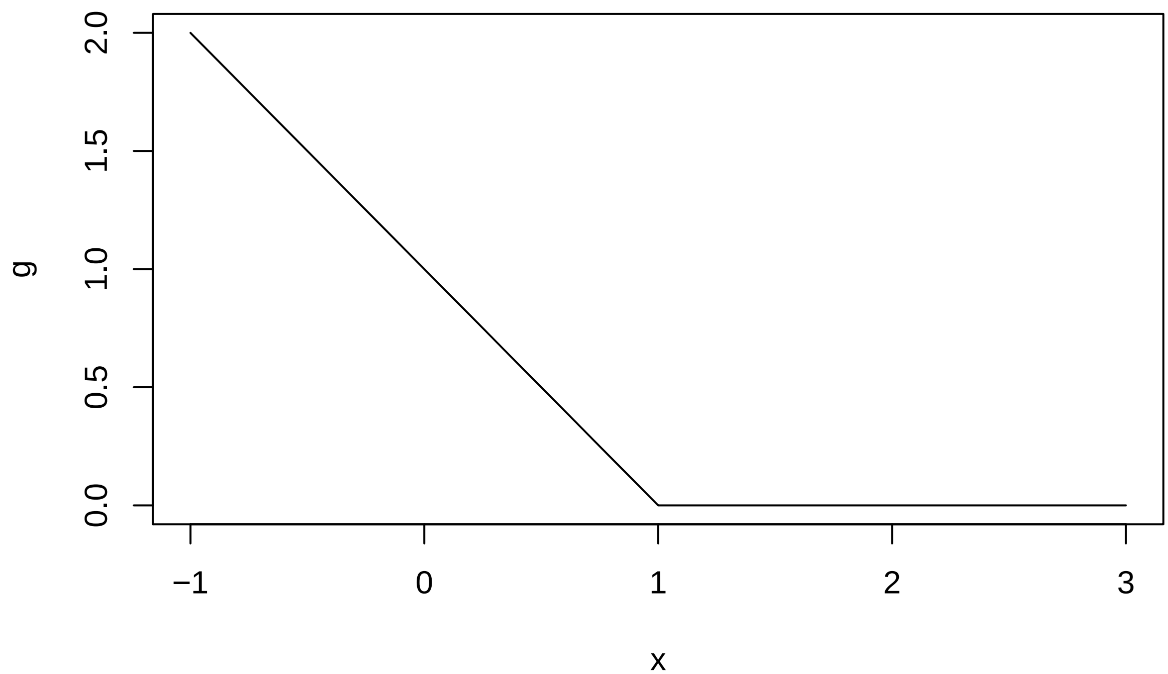


SVM classification plot
x2
x1
10
5
0
−5
−10
−10
−5
0
5
1
−1




```
#bandwidths from 10^-1 to 10^2
```

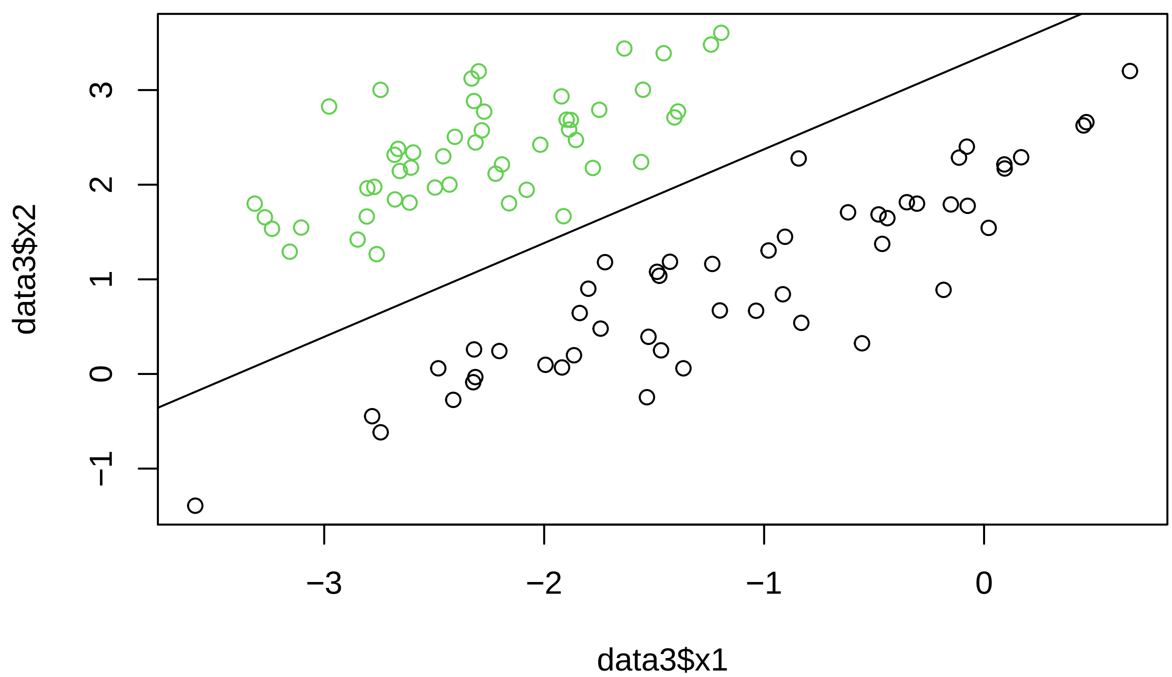
```
#3.i
x=seq(-1,3,0.01)
g=rep(NA,length(x))
for (i in 1:length(x)){
  g[i]=max(0,1-x[i])
}

plot(x,g,type="l")
```



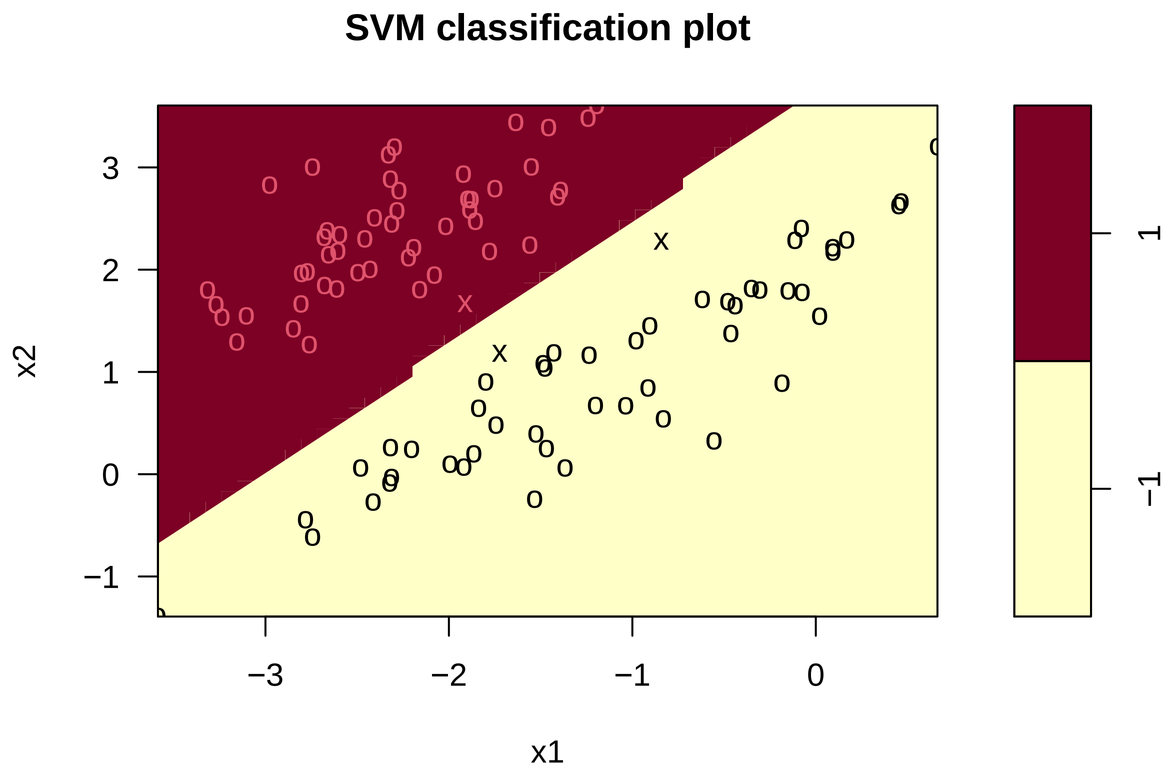

```r
#3.iii
data3=read.csv("svmdata.csv")
lambda=0.25
beta=c(0,0,0)
z=rep(NA,100)
del_Qi=data.frame(Qi_c=rep(NA,100),Qi_w1=rep(NA,100),Qi_w2=rep(NA,100))
for (t in 1:10000){
for (i in 1:100){
  z[i]=data3$y[i]*(beta[1]+beta[2]*data3$x1[i]+beta[3]*data3$x2[i])
  if (z[i]>1) {
    del_Qi[i,]=c(0,lambda*beta[2],lambda*beta[3])
  }
  if (z[i]<1){
    del_Qi[i,]=c(-data3$y[i],lambda*beta[2]-data3$y[i]*data3$x1[i],lambda*beta[3]-data3$y[i]*data3$x2[i
  }
}
del_Q=c(mean(del_Qi$Qi_c),mean(del_Qi$Qi_w1),mean(del_Qi$Qi_w2))
if (((1/t/0.25)*del_Q[1]<10^(-10)) & ((1/t/0.25)*del_Q[2]<10^(-10)) & ((1/t/0.25)*del_Q[3]<10^(-10))){
  print(t)
  break
}
beta=beta-(1/t/0.25)*del_Q
}
```

```
## [1] 1534
```

```r
plot(data3$x1,data3$x2,col=(data3$y+2))
x1=seq(-4,1,0.01)
x2=-beta[2]/beta[3]*x1-beta[1]/beta[3]
lines(x1,x2)
```




```r
data_3=data3
data_3$y=as.factor(data_3$y)
p3svm=svm(y~.,data=data_3,kernel="linear",cost=100000)
plot(p3svm,data_3,x2~x1)
```




```
# The  estimated linear decision boundary from 3.iii is close to the boundary from function svm()

beta
```

```
## [1] -2.7987148 -0.8242207  0.8315462
```