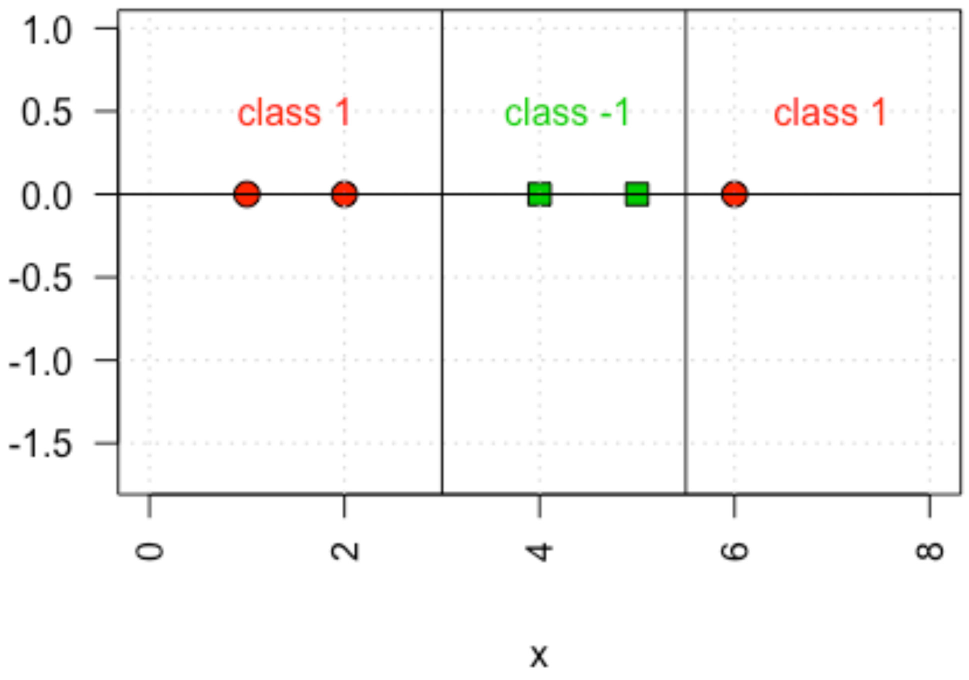
TP1 Machine Learning

MEHDAOUI Abderrahim & LAOUEDJ Abdelmadjid

10/7/2019

Exercice 1

```r
library(kernlab)
x = matrix(c(1, 2, 4, 5, 6))
y = c(1, 1, 2, 2, 1)
plot(x, rep(0, 5), pch = c(21, 22)[y], bg = c("red", "green3")[y], cex = 1.5,
ylim = c(-1.7, 1), xlim = c(0, 8), ylab = "",
xlab = "x", las = 2)
grid()
text(matrix(c(1.5, 4.3, 7, 0.5, 0.5, 0.5), 3, 2), c("class 1", "class -1",
"class 1"),
col = c("red", "green3", "red"))
abline(h=0) ; abline(v=c(3, 5.5))
```



##

Question 1 : La formule duale associée au problème est :

$\max_{\alpha} \sum_{k=1}^{5} \alpha_k + \frac{1}{2}\sum_{i,j}(x_i x_j + 1)\alpha_i \alpha_j y_i y_j$ avec $0 \leq \alpha \leq C$ et $\sum \alpha_k y_k = 0$

Question 2 :

```
library(kernlab)
poly = polydot(degree = 2, scale = 1, offset = 1)
K = kernelMatrix(poly, matrix(x))
C = 100
y=c(1,1,-1,-1,1)
U = matrix(rep(C,nrow(matrix(x))))
l = matrix(rep(0,nrow(matrix(x))))
A = t(matrix(y))
b = 0
r = 0
d = matrix(rep(-1,nrow(matrix(x))))

H=matrix(y)%*%t(matrix(y))*K
fit.svm = ipop(d,H,A,b,l,U,r)
alpha = fit.svm@primal
```

Question 3 :

```
print(alpha)
```

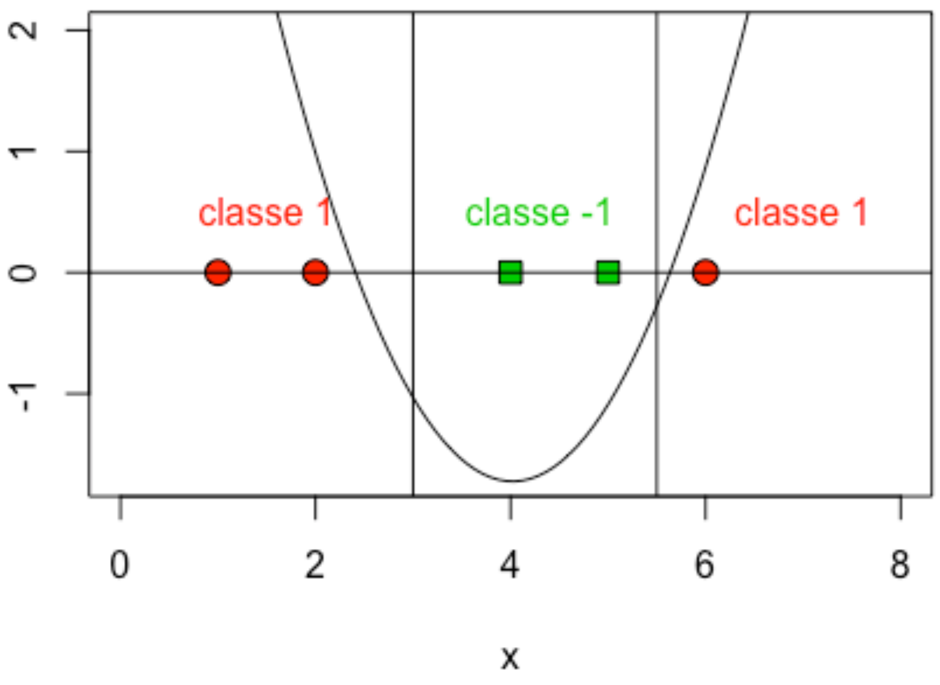
```
## [1] 1.277054e-08 2.500000e+00 8.576981e-08 7.333333e+00 4.833333e+00
```

Question 4 :

On pose alpha1=alpha2=0 et en réinjectant le reste dans la fonction primale on trouve $w_2 = 0.663, w_1 = -5.333$ et $w_0 = 9$

Question 5 :

```
x1 = c(1, 2, 4, 5, 6)
x2 = rep(0, 5)
y = c(1, 1, 2, 2, 1)
plot (x1, x2, pch = c(21, 22)[y], bg = c("red", "green3")[y], cex=1.5,
ylim=c(-1.7, 2), xlim=c(0, 8), ylab="", xlab="x")
text(1.5, 0.5, "classe 1", col="red")
text(4.3, 0.5, "classe -1", col="green3")
text(7, 0.5, "classe 1", col="red")
abline(h=0)
abline(v=3)
abline(v=5.5)
x = seq(0, 8, l=100)
f = 0.663*x^2-5.333*x+9
points(x, f, type = "l")
```



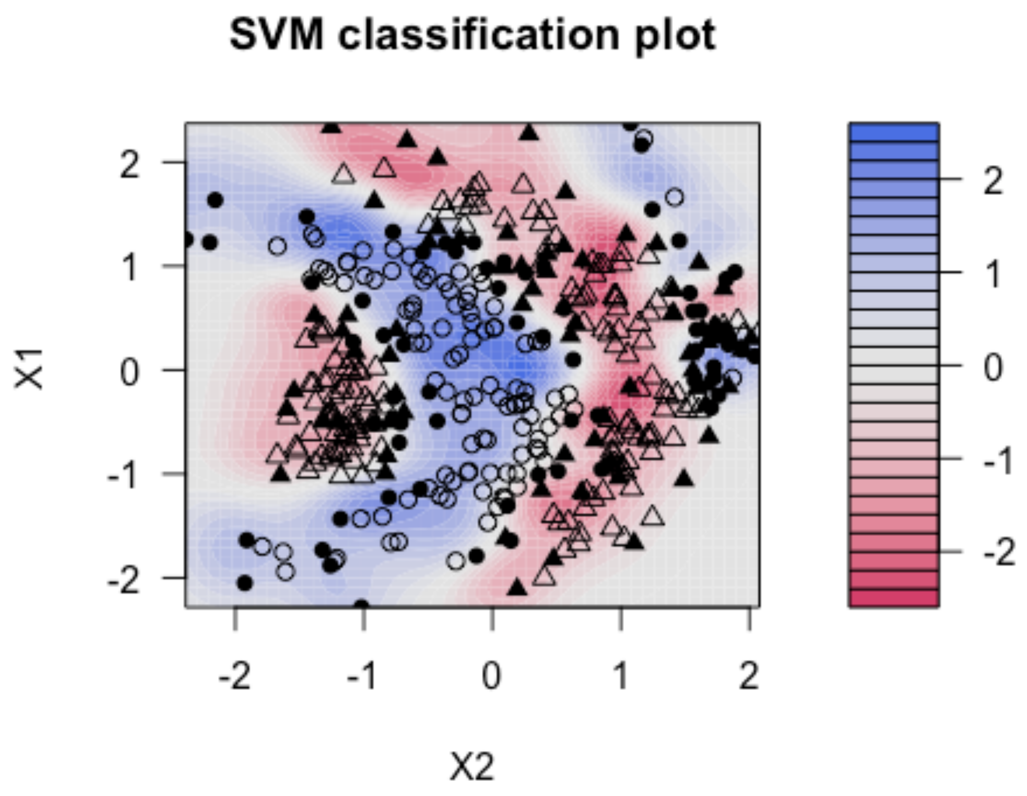
# Exercice 2

Question 1

```r
load("Banane.Rdata")
```

Question 2

```r
donnee = as.matrix(Apprentissage[,1:2])
target = as.matrix(Apprentissage[,3])
classif <-
ksvm(x=donnee,y=target,kernel="rbfdot",kpar=list(sigma=5),C=5,type="C-
svc",cross=3)
plot(classif,data=donnee)
```

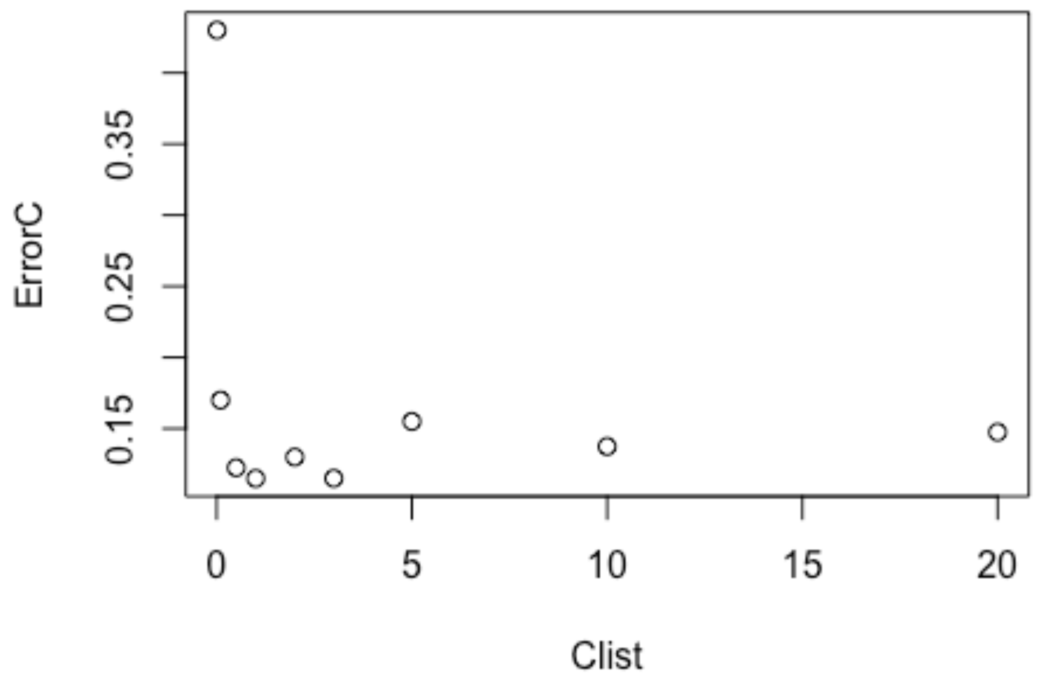


Question 3

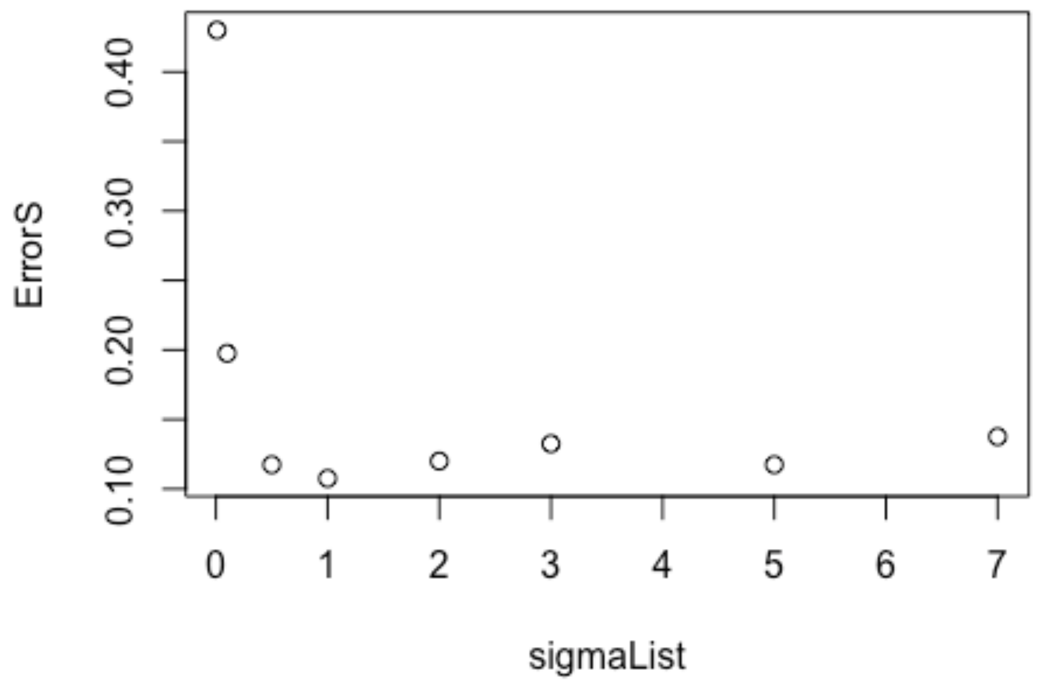
On a de meilleures frontières quand C *et* σ augmentent. Néanmoins trop augmenter ces paramètres peut conduire à une variance trop forte (overfitting)

Question 4

```
Clist = c(0.01, 0.1, 0.5, 1, 2, 3, 5, 10, 20)
sigmaList = c(0.01, 0.1, 0.5, 1, 2, 3, 5, 7)
ErrorC = matrix(0,length(Clist),1)
ErrorS = matrix(0,length(sigmaList),1)
for (i in 1:length(Clist)){
  classif =
ksvm(x=donnee,y=target,kernel="rbfdot",kpar=list(sigma=5),C=Clist[i],type="C-
svc",cross=3)
  ErrorC[i] = classif@cross
}
for (i in 1:length(sigmaList)){
  classif =
ksvm(x=donnee,y=target,kernel="rbfdot",kpar=list(sigma=sigmaList[i]),C=5,type
="C-svc",cross=3)
  ErrorS[i] = classif@cross
}
plot(Clist,ErrorC)
```



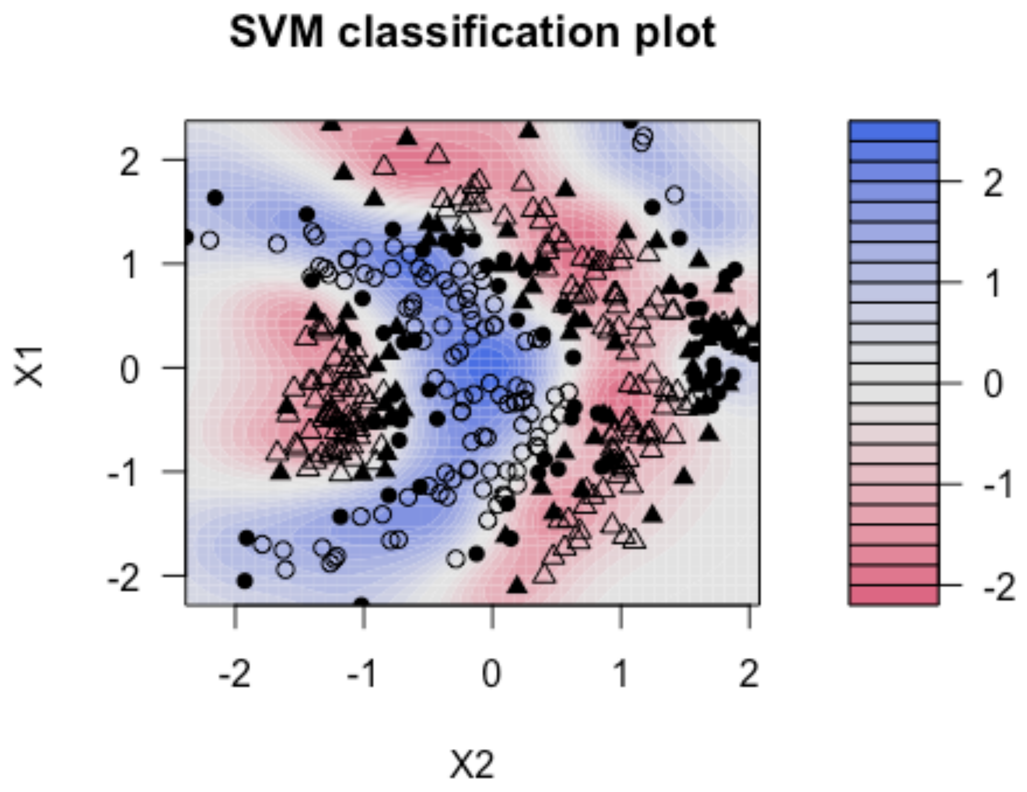
```r
plot(sigmaList,ErrorS)
```



On prend $C = 2$ et $\sigma = 3$

Question 5

```
classif_optimal <-
ksvm(x=donnee,y=target,kernel="rbfdot",kpar=list(sigma=3),C=2,type="C-
svc",cross=2)
donneeTest = as.matrix(Test[,1:2])
targetTest = as.matrix(Test[,3])
pred = predict(classif_optimal,donneeTest)
plot(classif_optimal,data=donnee)
```



```
conf = table(pred, targetTest)
rate = (conf[1,2]+conf[2,1])/(conf[1,2]+conf[2,1]+conf[1,1]+conf[2,2])

print(rate)

## [1] 0.1079592
```

On a donc un taux d’erreur d’environ 11% sur le jeu de données test.