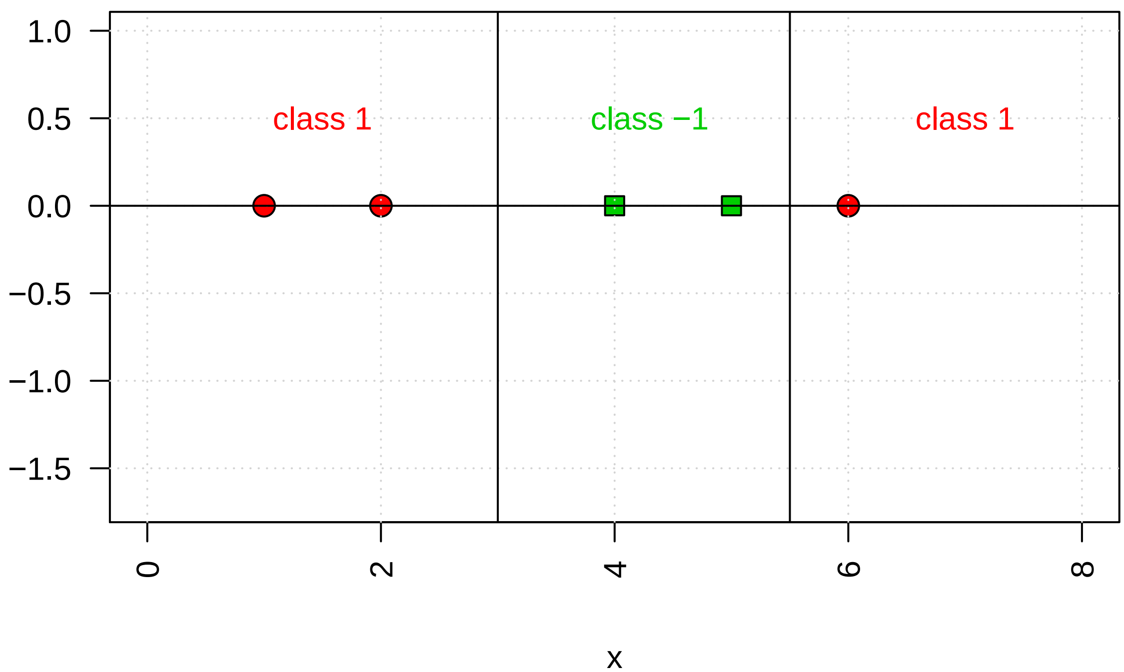
TP ML

Adnan Zeddoun, OMA

10/10/2019

Exercice 1. Introduction aux SVM

```r
x = c(1, 2, 4, 5, 6)
y = c(1, 1, 2, 2, 1)
z = c(1, 1, -1, -1, 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 formulation duale du problème d'optimisation associé aux SVM est (en dimension 1) :

$$\underset{\alpha}{max}\ L(\alpha) = \sum_{k=1}^{5} \alpha_k - \frac{1}{2} \sum_{i,j} (x_i x_j + 1)^2 \alpha_i \alpha_j y_i y_j$$

Sous les contraintes

$$0 \le \alpha_k \le C, \sum_{k=1}^{5} \alpha_k y_k = 0$$

Question 2

```r
library(kernlab)
z = c(1, 1, -1, -1, 1)
ker = polydot(degree = 2, scale = 1, offset = 1) #noyau polynomial de degré 2

c = matrix(rep(-1,5))
H = kernelPol(ker,x,,z)
A = t(z)
b = 0
C = 100
l = matrix(0,5,1)
u = matrix(C,5,1)
r = 0

res = ipop(c,H,A,b,l,u,r)
# @c Vecteur devant le "monôme d'ordre 1" de la fonction à maximiser
# @H Matrice à noyau intervenant dans le terme quadratique de la fonction à maximiser
# @A,b,r contraintes pour que la somme définie dans les contraintes soit nulle
# @l Contrainte imposant que chaque composante du vecteur inconnu soit positive ou nulle
# @u Contrainte imposant que chaque composante du vecteur inconnu soit négative ou nulle
```

Question 3

Le résultat donné est alors

```r
alpha = res@primal
print(alpha)
```

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

Question 4

On trouve que en réinjectant les α_k non nuls dans la fonction primale à minimiser, on a

$$f(x) = w_2 x^2 - w_1 x + w_0$$

avec

$$w_2 = 0.667, w_1 = 5.33, w_0 = 9$$

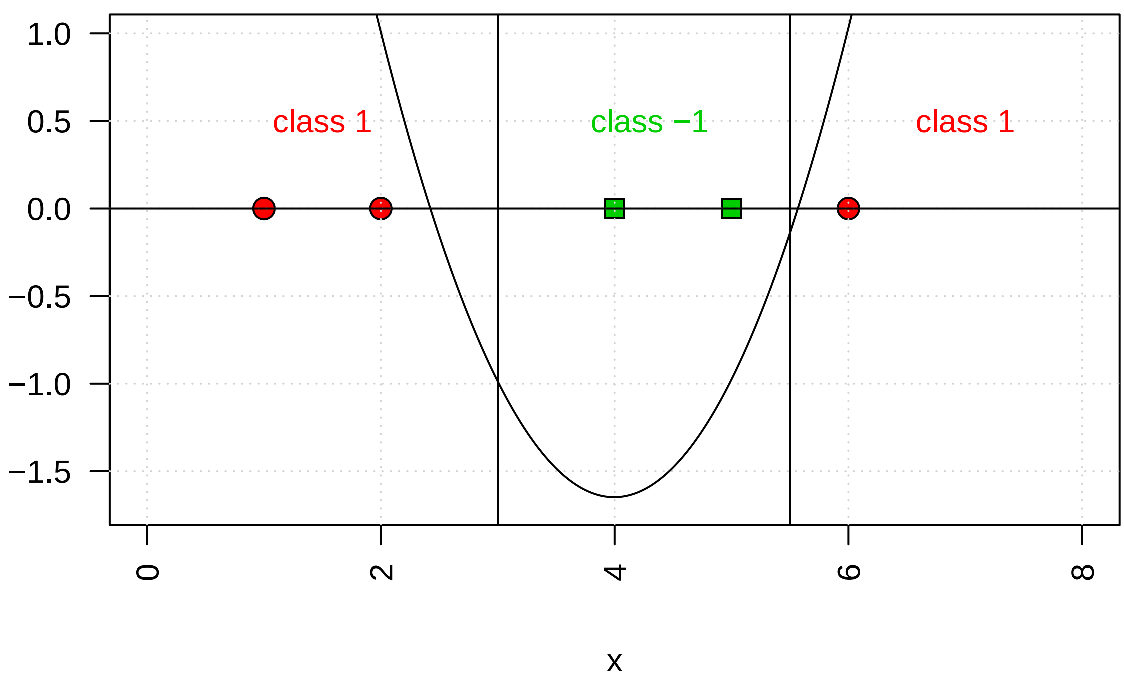
Question 5

```r
abscisse = seq(from = 0, to = 8, by =0.01)

f = function(x){
  n = length(x)
  func = c(rep(0,n))
```

```r
  for (i in 1:n){
    func[i] = 0.667*x[i]^2 - 5.33*x[i] + 9
  }
  return(func)
}

x = c(1, 2, 4, 5, 6)
y = c(1, 1, 2, 2, 1)
z = c(1, 1, -1, -1, 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))
lines(abscisse,f(abscisse))
```



Exercice 2. Support Vector Machines et validation croisée

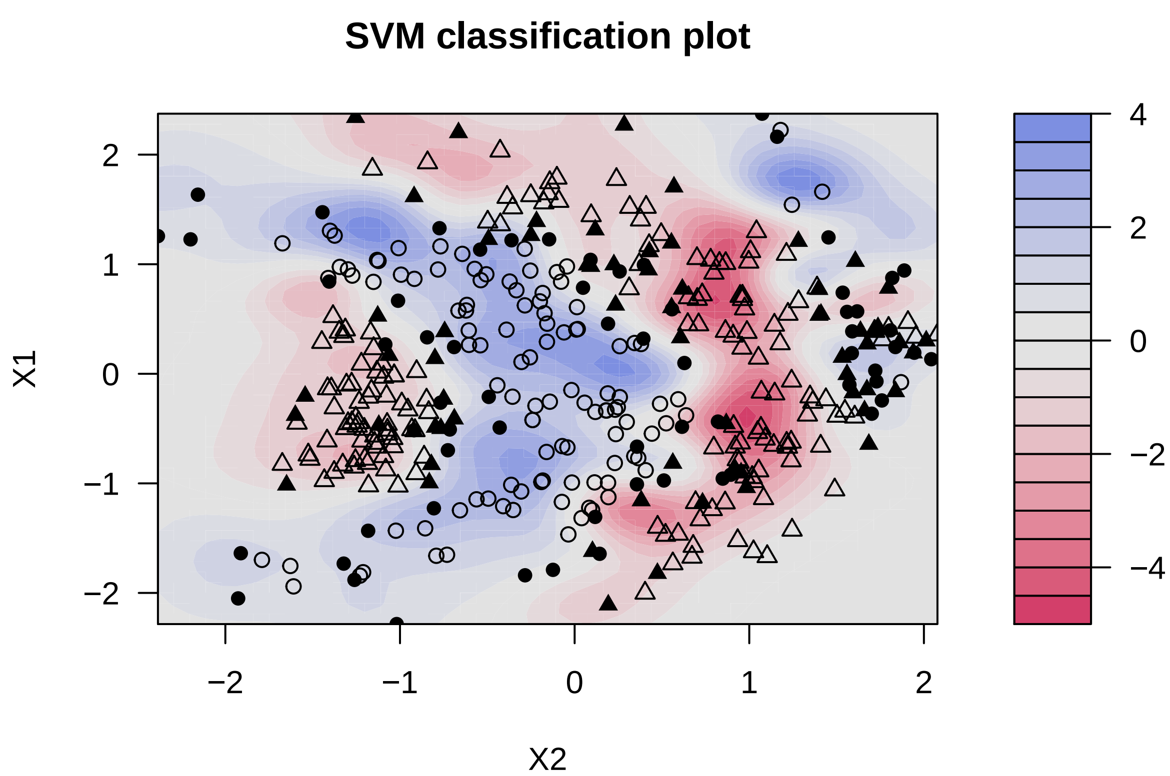
Question 1

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

Question 2

```
a = as.matrix(Apprentissage[,1:2])
b = as.matrix(Apprentissage[,3])

fil <- ksvm(x=a,y=b,kernel="rbfdot",kpar=list(sigma=5),C=20,type="C-svc",cross=2)
plot(fil,data=a)
```



Question 3

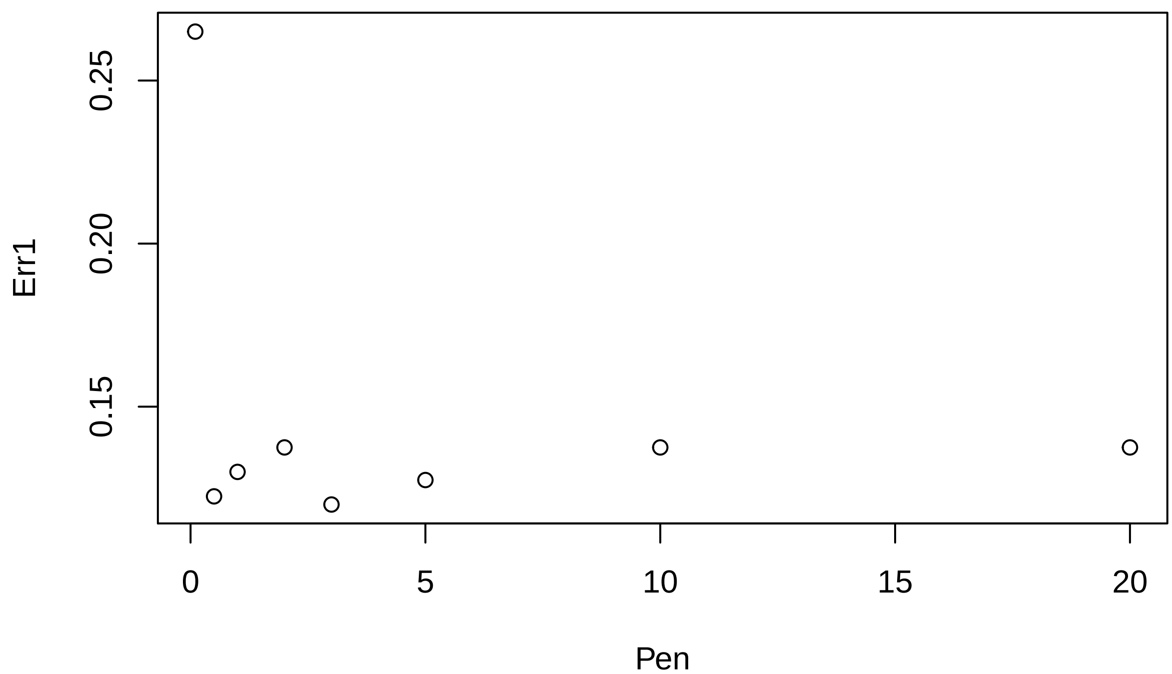
Plus C et σ augmentent et plus la frontière est oscillante permettant une séparation plus fine des points. Trop augmenter ces paramètres peut conduire à une sur-apprentissage du jeu de données.

Question 4

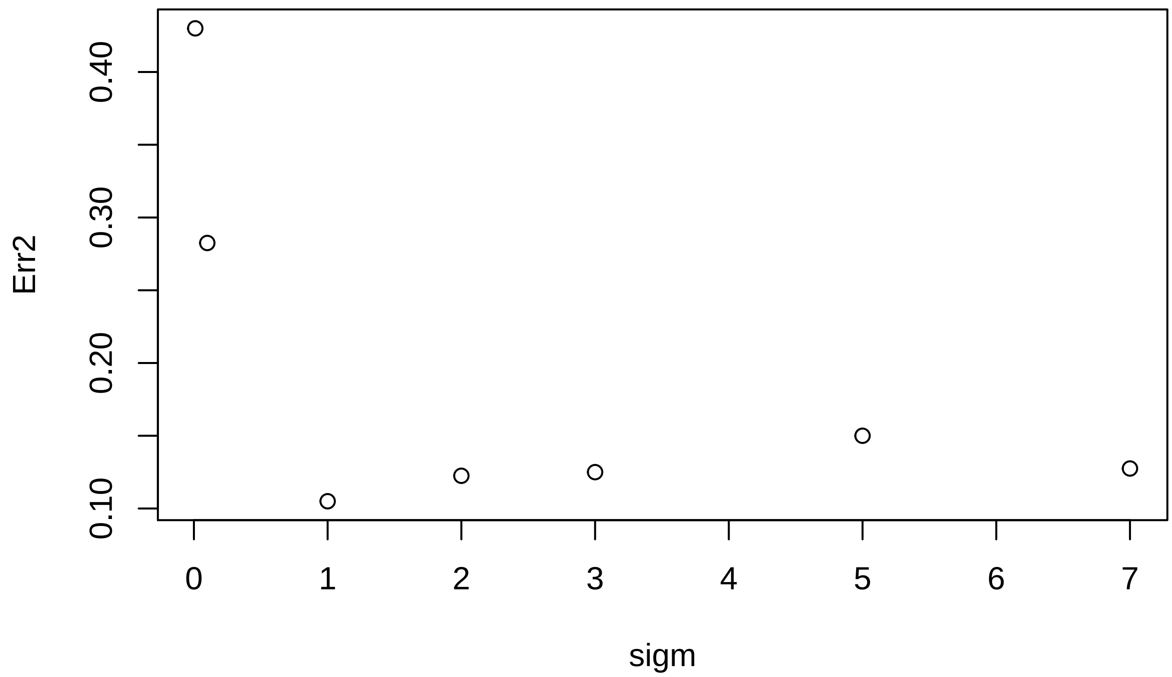
```
Pen = c(0.1,0.5,1,2,3,5,10,20)
sigm = c(0.01,0.1,1,2,3,5,7)
Err1 = matrix(0,length(Pen),1)
Err2 = matrix(0,length(sigm),1)
for (i in 1:length(Pen)){
  s = ksvm(x=a,y=b,kernel="rbfdot",kpar=list(sigma=5),C=Pen[i],type="C-svc",cross=2)
  Err1[i] = s@cross
}
```

```r
for (i in 1:length(sigm)){
  s = ksvm(x=a,y=b,kernel="rbfdot",kpar=list(sigma=sigm[i]),C=5,type="C-svc",cross=2)
  Err2[i] = s@cross
}

plot(Pen,Err1)
```




```r
plot(sigm,Err2)
```

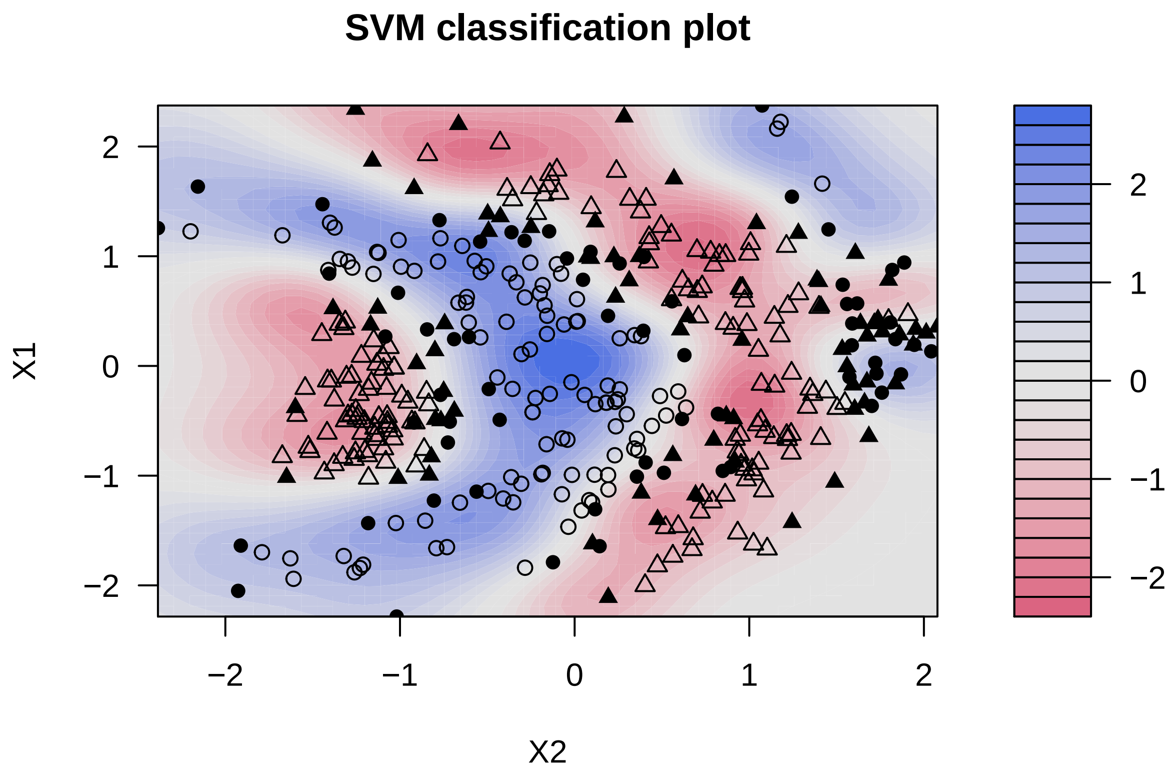


On prend les paramètres minimisant l'erreur par cross-validation. On choisit

$$(C^*, \sigma^*) = (3, 3)$$

Question 5

```
fil2 <- ksvm(x=a,y=b,kernel="rbfdot",kpar=list(sigma=3),C=3,type="C-svc",cross=2)
z = as.matrix(Test[,1:2])
q = as.matrix(Test[,3])
pred = predict(fil2,z)
plot(fil2,data=a)
```




```r
g = table(pred, q)
tauxerr = (g[1,2]+g[2,1])/(g[1,2]+g[2,1]+g[1,1]+g[2,2])
print(tauxerr)
```

```
## [1] 0.1091837
```

Le taux d'erreur est de 11% sur le jeu de données Test.