Introduction to Support Vector Machines

Antoine Dargier

2022-10-23

Exercice I. Introduction to C-SVM

For illustrating kernel methods in general and for Support Vector Machines in particular, we consider a very simple classification problem. Let assume that the data is describe in a 1D space divided into two classes ($+1$ and -1) as follows:

$$\mathcal{S} = \{(\mathbf{x}_1 = 1, y_1 = 1), (\mathbf{x}_2 = 2, y_2 = 1), (\mathbf{x}_3 = 4, y_3 = -1), (\mathbf{x}_4 = 5, y_4 = -1), (\mathbf{x}_5 = 6, y_5 = 1)\}$$

The following script is used for visualizing the data.

```
x = c(1, 2, 4, 5, 6)
class = c(1, 1, 2, 2, 1)

plot(x, rep(0, 5), pch = c(21, 22)[class],
     bg = c("red", "green3")[class],
     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))
```

Of course, linear boundary can't discriminate the two classes and we propose to train a nonlinear SVM classifier combined with a second order polynomial kernel defined as:

$$k(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x_1}^\top \mathbf{x}_2 + 1)^2.$$

First, we import all the libraries needed :

```
library(kernlab)
library(pROC)
library(caret)
library(plotly)
```

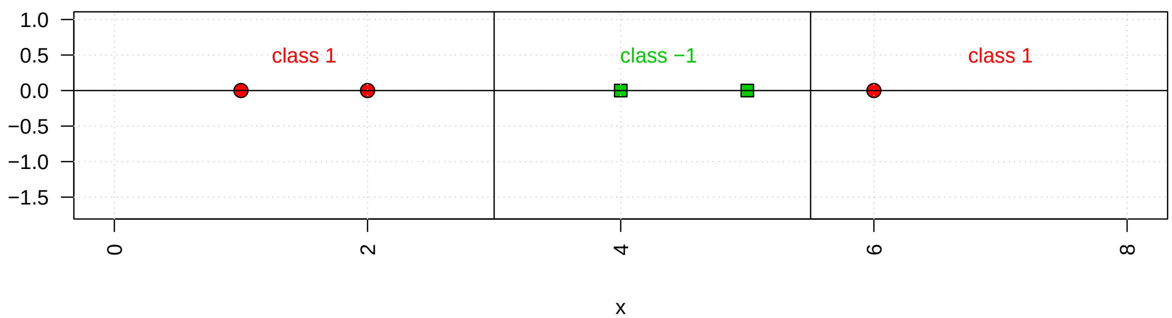


Figure 1: Data Visualisation

Question 1. Write the dual formulation associated with the SVM optimization problem.

The dual formulation of the problem can be written as :

$$\hat{f} = \arg\min_{f \in H_k} (C \sum_{i=1}^{n} \phi_{hinge}(y_i * f(x_i)) + \frac{1}{2} * \|f\|_{H_k}^{2})$$

From the representer theorem, the solution admits a solution of the form:

$$\hat{f} = \sum_{i=1}^{n} \alpha_i k(x_i, .)$$

So, we need to solve :

$$\mu^* = \arg \max_{0 <= \mu <= C; \mu^T y = 0} (\mu^T \mathbb{1} - \frac{1}{2} \mu^T diag(y) K diag(y) \mu)$$
$$\alpha^* = diag(y) \mu^*$$

Question 2. Specify the arguments of the `kernlab:::ipop` to solve this optimization problem.

The quadratic programming solver ipop solves the following problem : $\min(c' * x + 1/2 * x' * H * x$ subject to: $b <= A * x <= b + r; l <= x <= u$

So, by correspondence between the two equations, we can find that $c = 100$
$H = diag(y) * K * diag(y)$
$A = y^T$
$b = 0$
$l = 0$
$u = 0$
$r = 0$

Question 3. With $C = 100$, show that this quadratic optimization yields:

$$\hat{\mu}_1 = 0, \hat{\mu}_2 = 2.5, \hat{\mu}_3 = 0, \hat{\mu}_4 = 7.333 \text{ and } \hat{\mu}_5 = 4.833$$

```r
C = 100
x = matrix(c(1,2,4,5, 6), ncol = 1)
y = c(1,1,-1,-1,1)

rbf = polydot(degree = 2, scale = 1, offset = 1)
K = kernelMatrix(rbf, x)

c = rep(-1,NROW(x))
H = diag(y)%*%K%*%diag(y)

A = t(y)
b = 0
r = 0
l = rep(0, NROW(x))
u = rep(C, NROW(x))

ipop(c, H, A, b, l, u, r)
```

```
## An object of class "ipop"
## Slot "primal":
## [1] 1.277054e-08 2.500000e+00 8.576981e-08 7.333333e+00 4.833333e+00
##
## Slot "dual":
## [1] -9
##
## Slot "how":
## [1] "converged"
```

So, we find the rigth optimizated parameters.

Question 4. From the representer theorem, we know that the solution take the form:

$$f(\mathbf{x}) = \sum_{i=1}^{n} \mu_i y_i k(\mathbf{x}, \mathbf{x}_i) + b^*$$

Deduce that the optimal solution is quadratic of the form:

$$f(\mathbf{x}) = w_2 \mathbf{x}^2 + w_1 \mathbf{x} + w_0$$

where w_0, w_1, w_2 to determine.

Indication: For determining w_0, you can use the fact that $y_i f(x_i) = 1$ for any support vectors x_i.

We know that $f(\mathbf{x}) = \sum_{i=1}^{n} \mu_i y_i k(\mathbf{x}, \mathbf{x}_i) + b^*$.

So :

$$f(\mathbf{x}) = \mu_2 y_2 k(x, x_2) + \mu_4 y_4 k(x, x_4) + \mu_5 y_5 k(x, x_5) + b^*$$
$$\mu_2 = 2,5; y_2 = 1; k(x, x_2) = 4x^2 + 4x + 1$$
$$\mu_4 = 7,3; y_4 = -1; k(x, x_4) = 25x^2 + 10x + 1$$
$$\mu_5 = 4,8; y_5 = 1; k(x, x_5) = 36x^2 + 12x + 1$$

So, when developing, we get : $f(\mathbf{x}) = 0,667x^2 - 5,33x + \tilde{w_0}$

We have : $y_2 f(x_2) = 1 \Rightarrow 0,667 * (2)^2 - 5,33 * 2 + \tilde{w_0} = 1 \Rightarrow \tilde{w_0} = 9$

Question 5. Add the optimal decision function to Figure 1.

```
f <- function(x){
return(0.667*x^2-5.333*x+9)
}
```

```
plot(x, rep(0, 5), pch = c(21, 22)[class],
     bg = c("red", "green3")[class],
     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))

ind = seq(0,8, l =100)
points(ind, f(ind), type = "l", col="blue")
```

Exercice II : Support Vector Machines and cross validation

In this exercise, we study the «Banana »dataset available on Eduano.

Question 1. Import and Visualize this data set.

```
load("C:\\Users\\antoi\\OneDrive\\Bureau\\CS\\3A\\SDI\\ML\\Cours 3\\Banane.Rdata")

plot(Apprentissage[, 2], Apprentissage[, 1], col = Apprentissage[, 3]+3,
     main = "Banana Data", xlab = "x2", ylab = "x1")
```

Question 2. Train a nonlinear SVM combined with gaussian kernel[1] with $\sigma = 5$ and the regularization parameter $C = 5$.

You can used the `kernlab:::ksvm()` function.

```
rbf = rbfdot(sigma = 5)
C = 5
model <- ksvm(Y~., data = Apprentissage, kernel = rbf, C=C, type = "C-svc")
error <- error(model)
nSV <- nSV(model)
plot(model, data=Apprentissage)
```

[1] We recall that within the `kernlab` library, gaussian kernel is defined as:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\Big(- \sigma \|\mathbf{x}_i - \mathbf{x}_j\|^2 \Big) \tag{1}$$

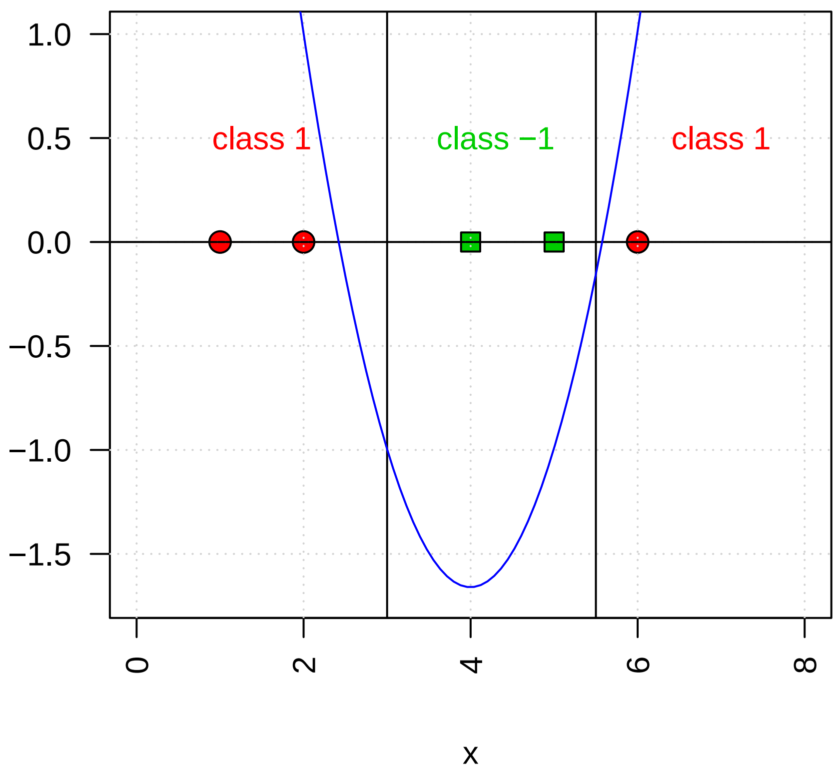


Figure 2: Data Visualisation with decision function

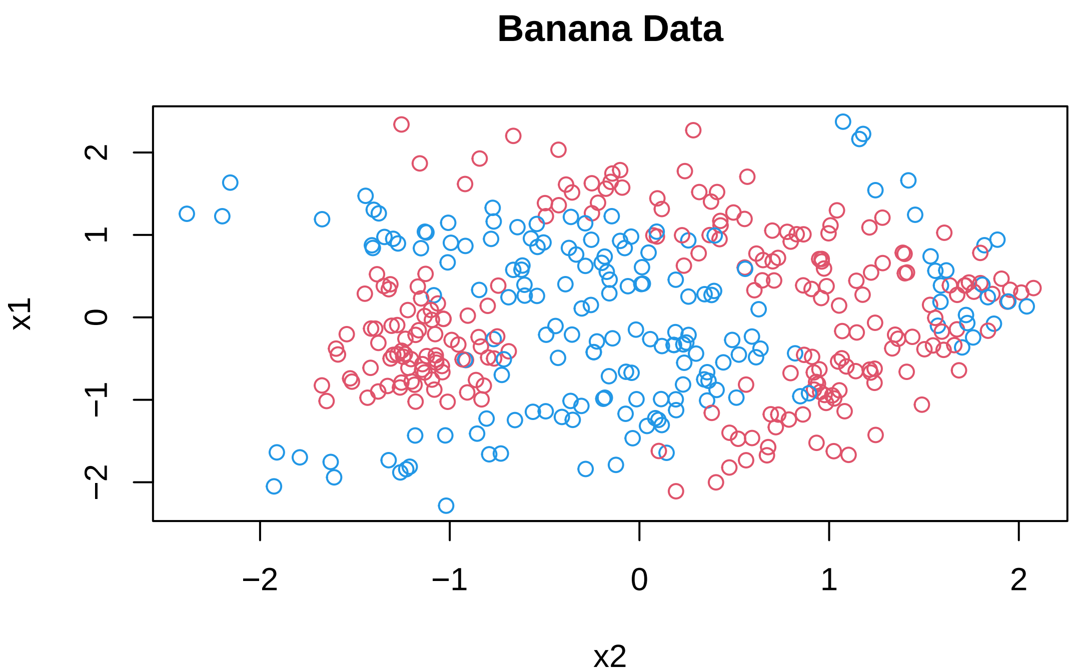


Figure 3: Data Visualisation

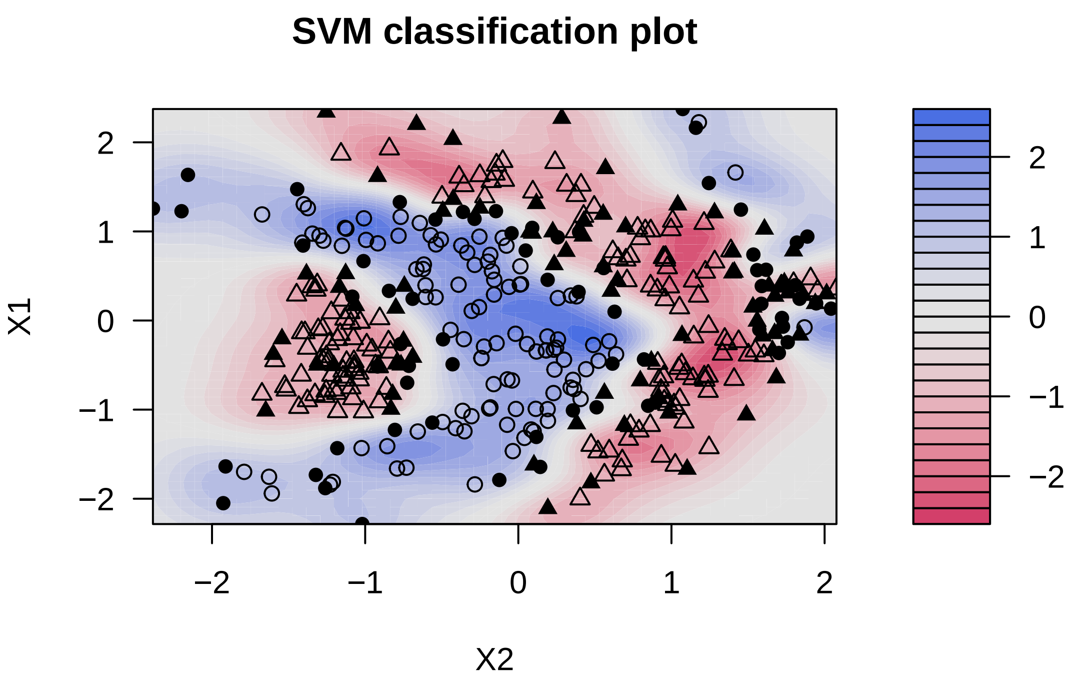


Figure 4: Trained nonlinear SVM with gaussian kernel, C = 5, sigma = 5

We obtain a first trained model, with a training error equal to 0.0775 and 140 support vectors used.

Question 3. For small value of σ, we can reduce the exponential function to its first-order Taylor approximation. In this case, prove that the SVM decision boundary is linear.

We know that

$$f(\mathbf{x}) = \sum_{i=1}^{n} \mu_i y_i k(\mathbf{x}, \mathbf{x}_i) + b^*$$

For small value of σ, with the first-order Taylor approximation, we get :

$$k(\mathbf{x}, \mathbf{x}_i) = \exp\Big(- \sigma \|\mathbf{x} - \mathbf{x}_i\|^2 \Big) \approx 1 - \sigma \|\mathbf{x} - \mathbf{x}_i\|^2 = 1 - \sigma(\|\mathbf{x}\|^2 + \|\mathbf{x}_i\|^2 - 2 < \mathbf{x} \cdot \mathbf{x}_i >)$$

That gives :

$$\begin{aligned} f(\mathbf{x}) &= \sum_{i=1}^{n} \mu_i y_i - \sigma \sum_{i=1}^{n} \mu_i y_i (\|\mathbf{x}\|^2 + \|\mathbf{x}_i\|^2 - 2 < \mathbf{x} \cdot \mathbf{x}_i >) + b^* \\ &= \sum_{i=1}^{n} \mu_i y_i - \sigma \sum_{i=1}^{n} \mu_i y_i \|\mathbf{x}\|^2 - \sigma \sum_{i=1}^{n} \mu_i y_i \|\mathbf{x}_i\|^2 + 2\sigma \sum_{i=1}^{n} \mu_i y_i < \mathbf{x} \cdot \mathbf{x}_i >) + b^* \\ &= \sum_{i=1}^{n} \mu_i y_i - \sigma \|\mathbf{x}\|^2 \sum_{i=1}^{n} \mu_i y_i - \sigma \sum_{i=1}^{n} \mu_i y_i \|\mathbf{x}_i\|^2 + 2\sigma \sum_{i=1}^{n} \mu_i y_i < \mathbf{x} \cdot \mathbf{x}_i >) + b^* \end{aligned}$$

Or, minimizing the loss function in an SVM problem, we get $\mu^T y = 0$, which is $\sum_{i=1}^{n} \mu_i y_i = 0$. So finally, we have :

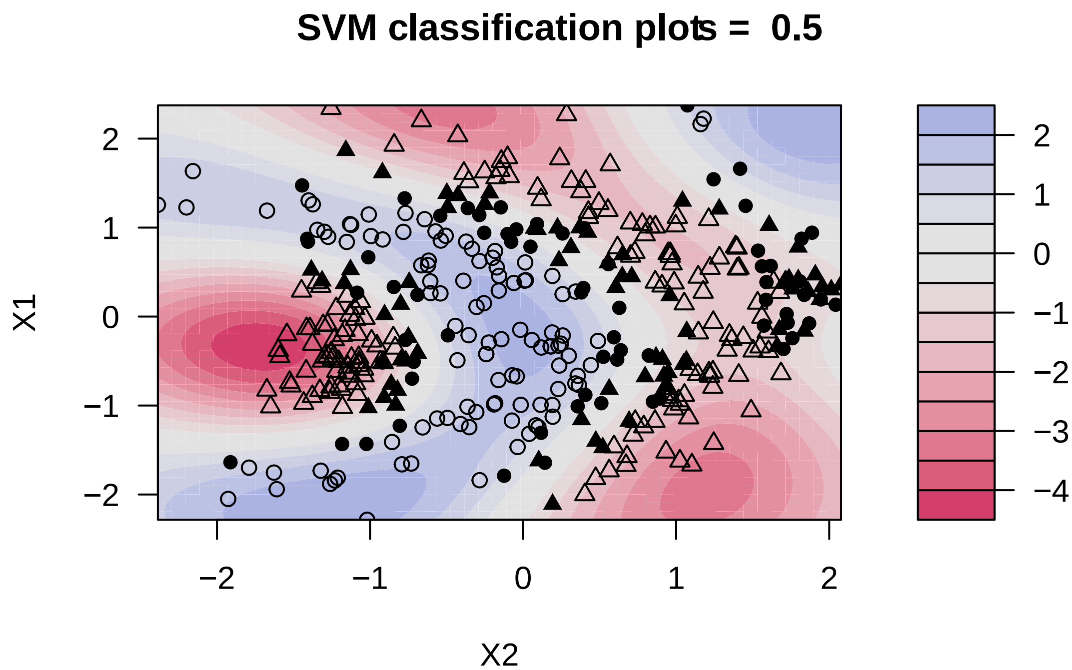
$$f(\mathbf{x}) = -\sigma \sum_{i=1}^{n} \mu_i y_i \|\mathbf{x}_i\|^2 + 2\sigma \sum_{i=1}^{n} \mu_i y_i < \mathbf{x} \cdot \mathbf{x}_i >) + b^*$$

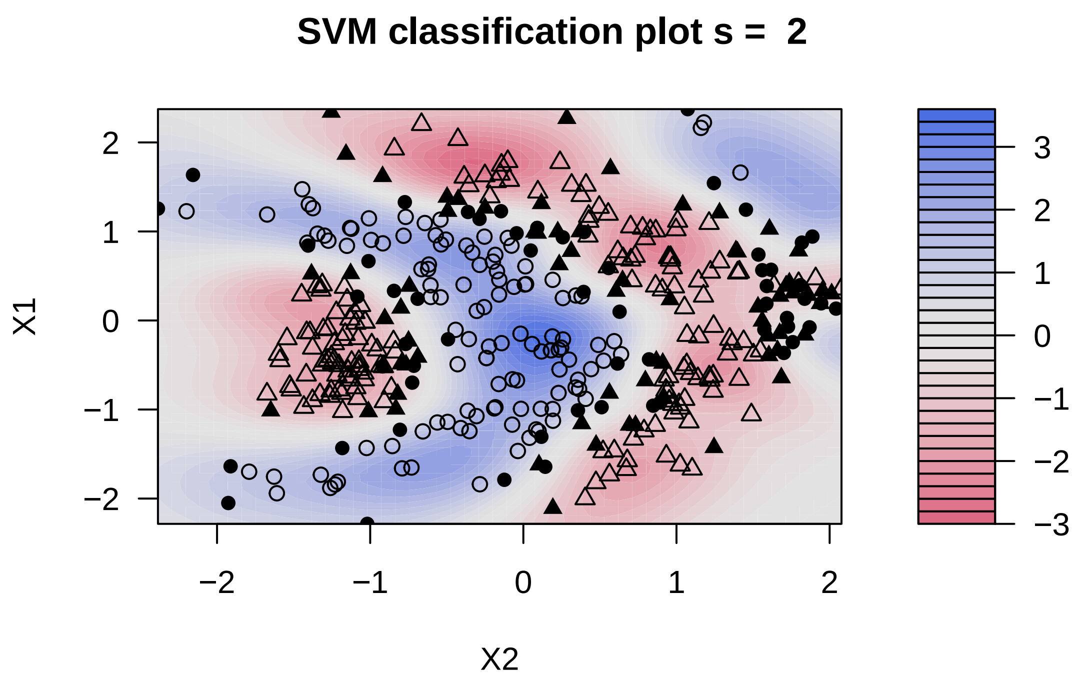
With the linearity of the scalar product, we can conclude that the SVM decision boundary is linear when σ is small.

Question 4. Visualize the SVM model (using ‘plot.ksvm()‘) and discuss the impact of C and σ on the boundary and on the number of support vectors.

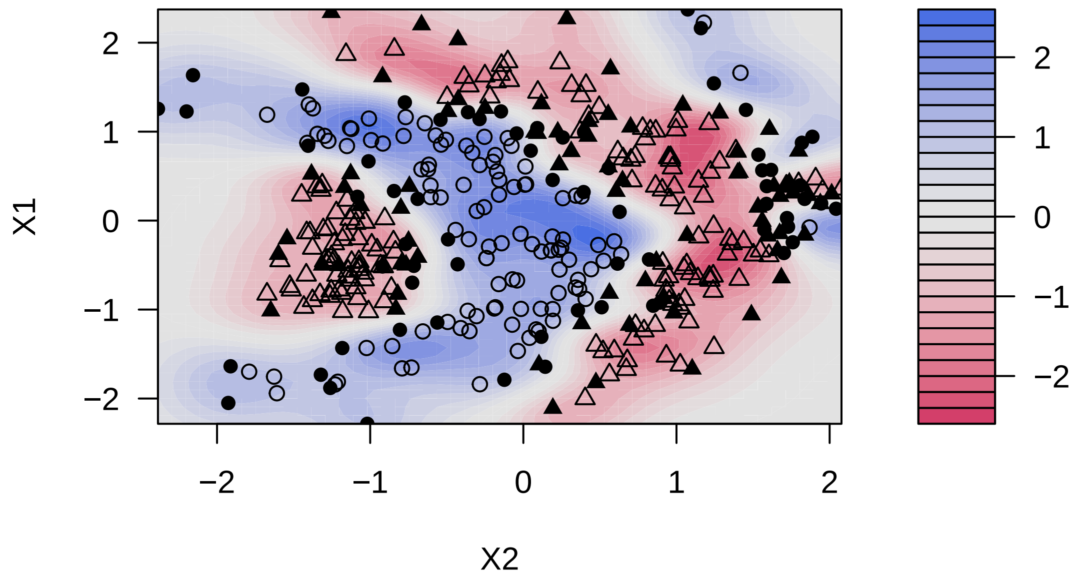
Let fix $C = 5$ and see the impact of σ :

```
sigmas = c(0.5, 2, 5, 10)
C = 5
for (sig in sigmas) {
rbf = rbfdot(sigma = sig)
model <- ksvm(Y~., data = Apprentissage, kernel = rbf, C=C, type = "C-svc")
plot(model, data=Apprentissage, )
title(main = paste("                              s = ", sig))
}
```

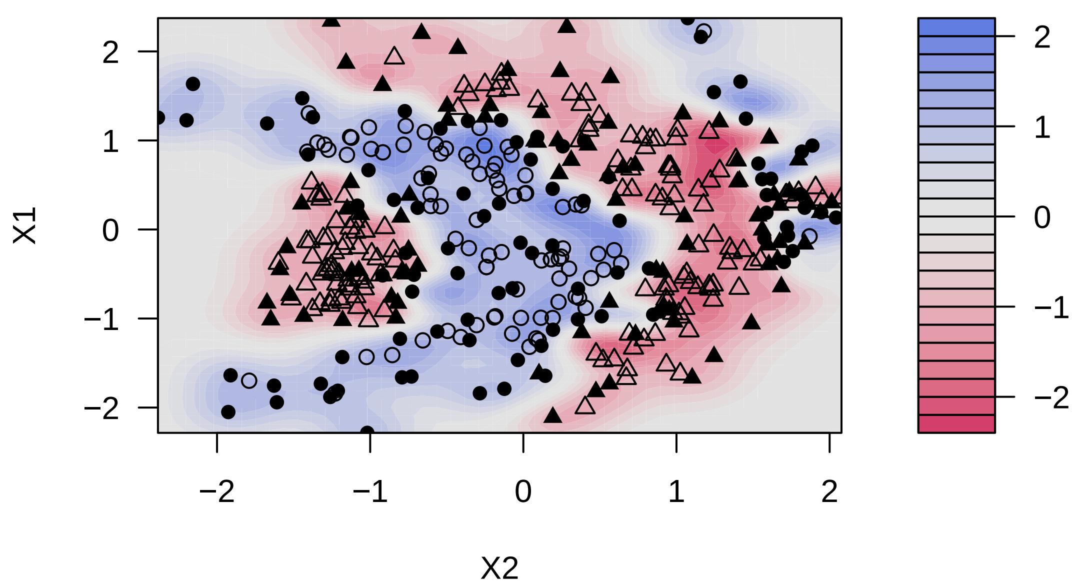

SVM classification plots = 0.5
X1
X2
2
1
0
−1
−2
−3
−4


SVM classification plot s = 2
X1
X2
3
2
1
0
−1
−2
−3

SVM classification plot s = 5

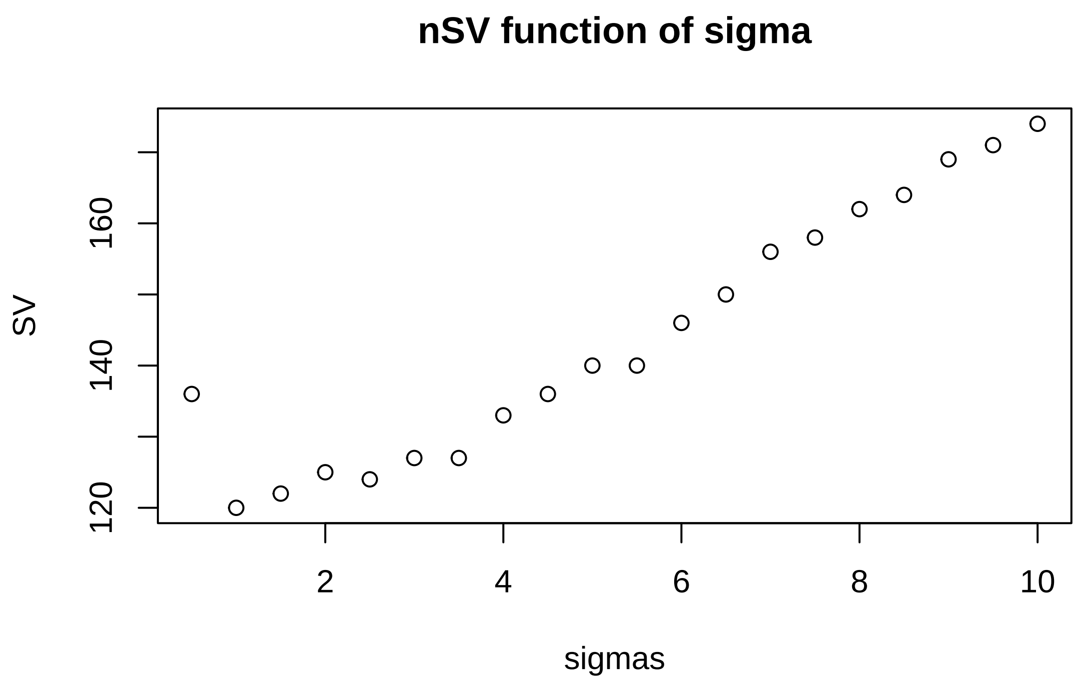


SVM classification plots = 10

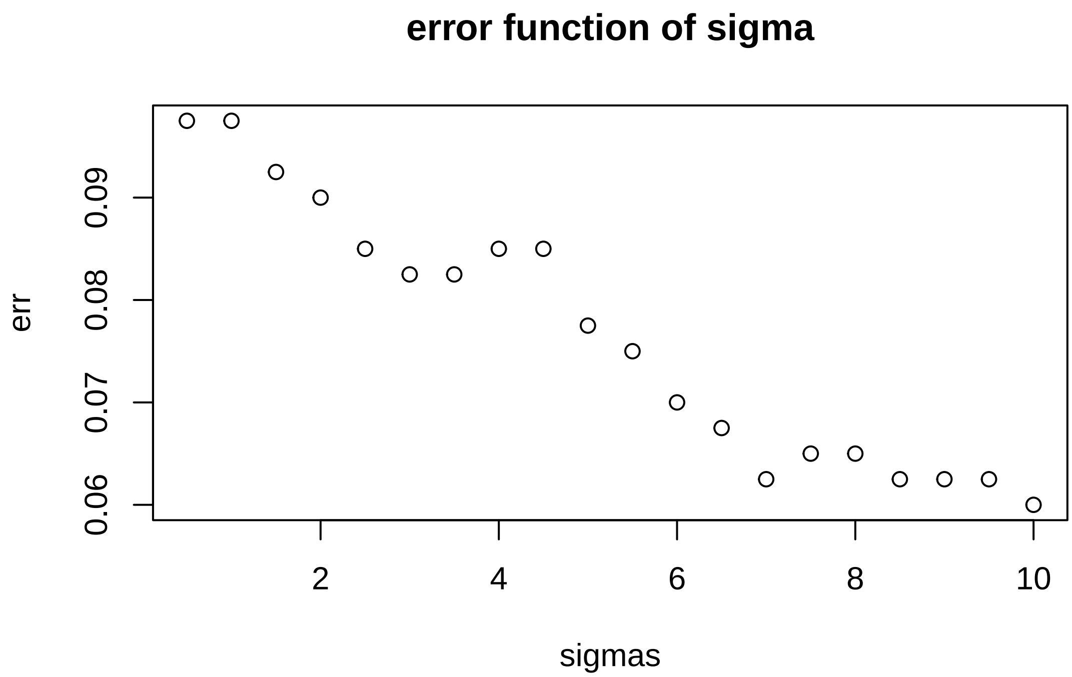



```
sigmas = (1:20)/2
C = 5
SV = c()
err = c()
```

```r
for (sig in sigmas){
rbf = rbfdot(sigma = sig)
model <- ksvm(Y~., data = Apprentissage, kernel = rbf, C=C, type = "C-svc")
SV <- c(SV, nSV(model))
err <- c(err, error(model))
}
plot(sigmas, SV, main = "nSV function of sigma")
```



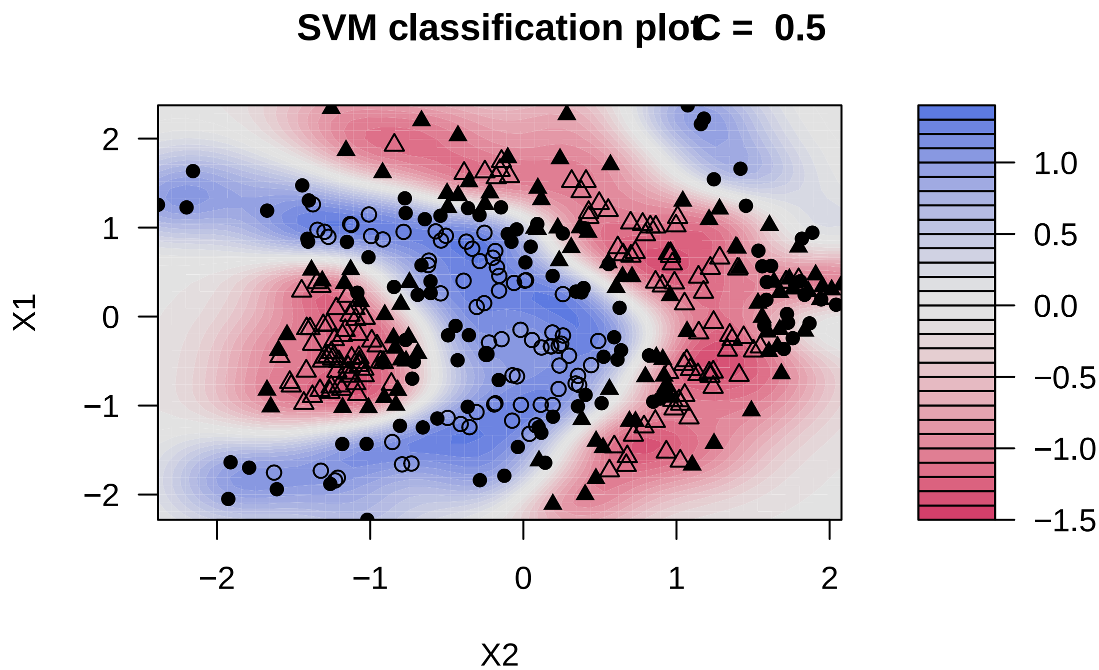

```r
plot(sigmas, err, main = "error function of sigma")
```

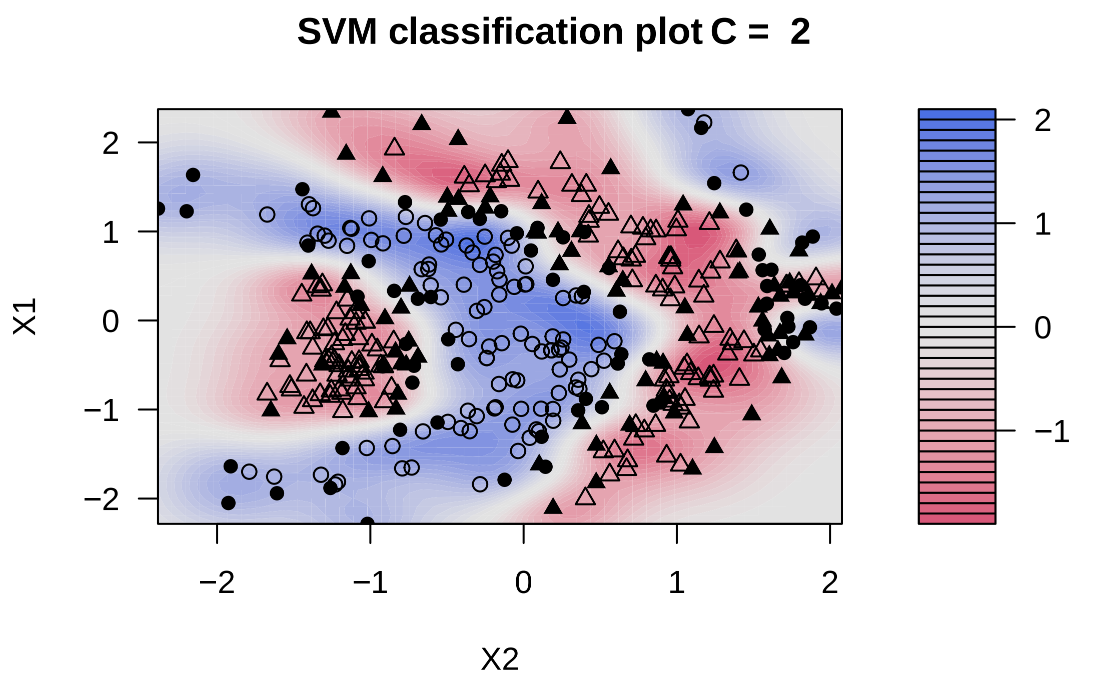


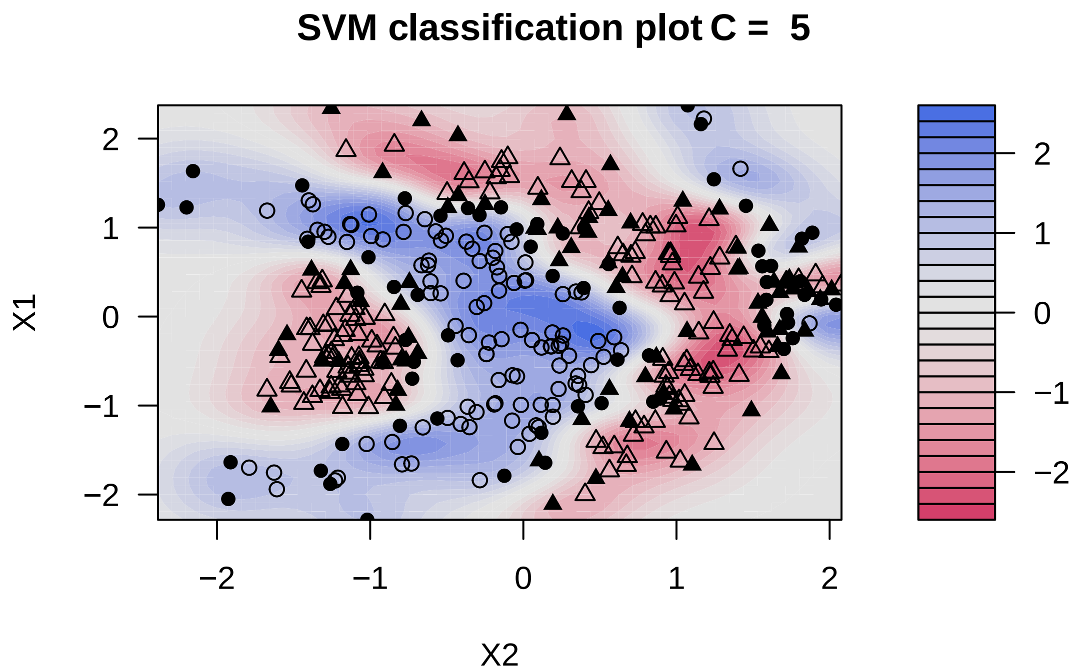
We can see that the number of support vectors increases with σ. The error decreases with σ but it creates a lot of overfitting on the training set. That explains why we need to use cross-validation.

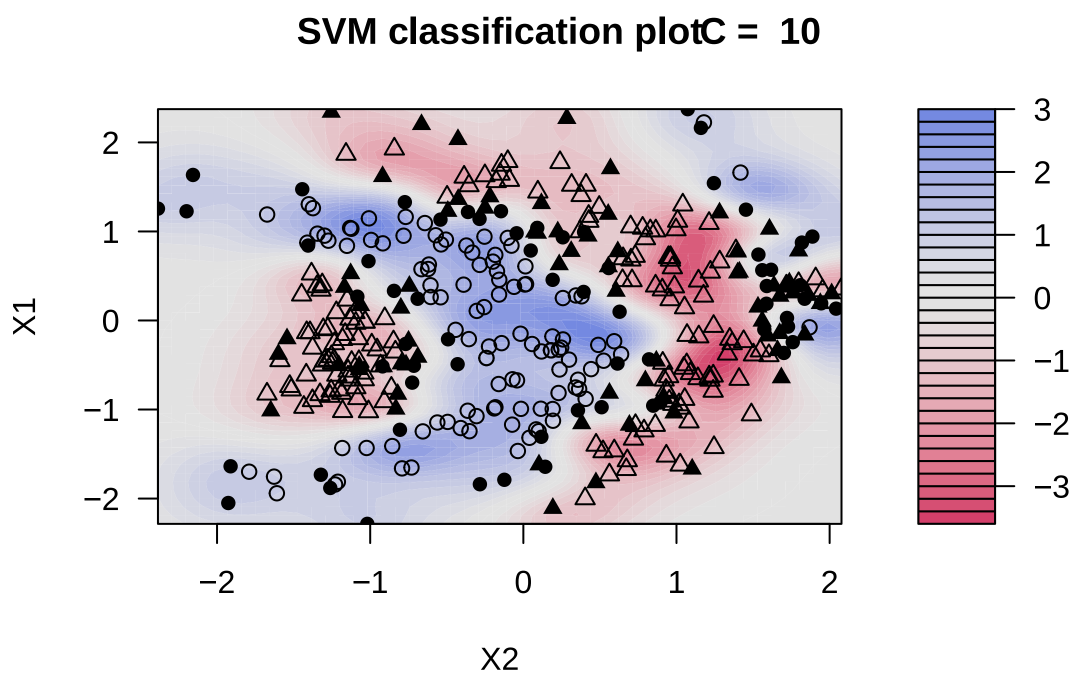
Now, let fix $\sigma = 5$ and see the impact of C :

```
sig = 5
Cs = c(0.5, 2, 5, 10)
for (C in Cs) {
rbf = rbfdot(sigma = sig)
model <- ksvm(Y~., data = Apprentissage, kernel = rbf, C=C, type = "C-svc")
plot(model, data=Apprentissage, )
title(main = paste("                                    C = ", C))
}
```


SVM classification plotC = 0.5
X1
X2
2
1
0
−1
−2
−2
−1
0
1
2
1.0
0.5
0.0
−0.5
−1.0
−1.5

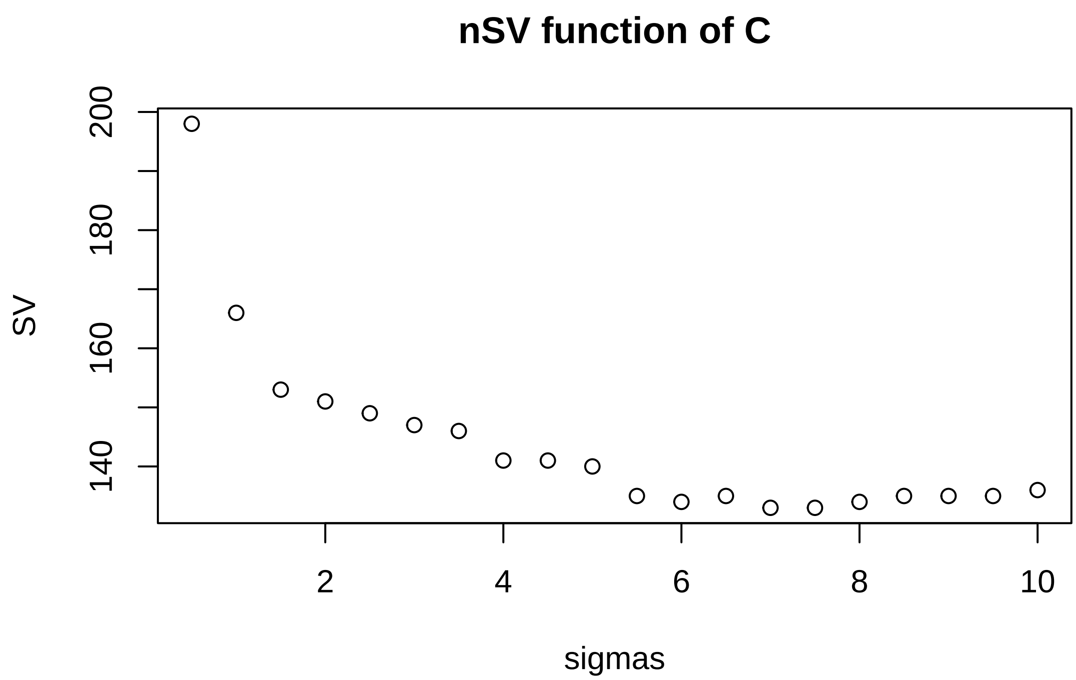

SVM classification plot C = 2
X1
X2
2
1
0
−1
−2
−2
−1
0
1
2
2
1
0
−1



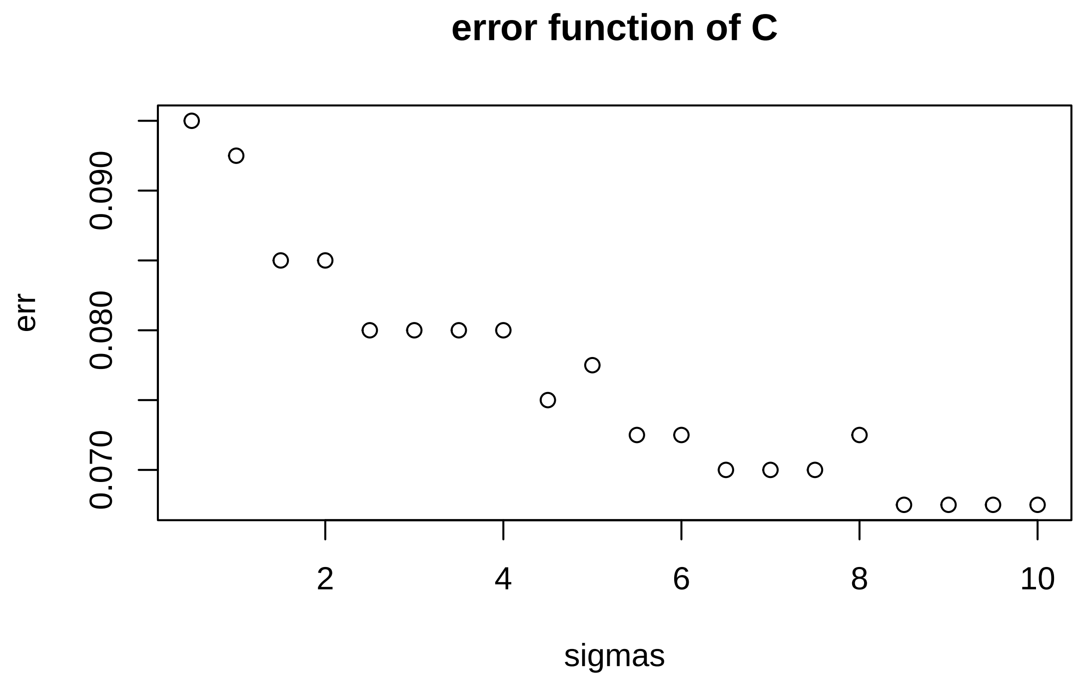



```
Cs = (1:20)/2
sig = 5
SV = c()
err = c()
```

```r
for (C in Cs){
rbf = rbfdot(sigma = sig)
model <- ksvm(Y~., data = Apprentissage, kernel = rbf, C=C, type = "C-svc")
SV <- c(SV, nSV(model))
err <- c(err, error(model))
}
plot(sigmas, SV, main = "nSV function of C")
```




```r
plot(sigmas, err, main = "error function of C")
```



We can see that the number of support vectors increases with C. The error decreases with C but it creates a lot of overfitting on the training set. That explains why we need to use cross-validation.

Question 5. Show the evolution of the cross-validated error rate as function of C and σ. Deduce the optimal values (C^*, σ^*) for C and σ.

We compute the cross validation on the Apprentissage set, for values of σ and C between 0.2 and 10, with a step of 0.2. We choose to divide the Apprentissage set in 7 sets for the cross validation.

```
Cs = (1:100)/10
sigmas = (1:100)/10
c_val <- c()
sig_val <- c()
e <- c()

for (C in Cs){
for (sig in sigmas){
rbf = rbfdot(sigma = sig)
model <- ksvm(Y~., data = Apprentissage, kernel = rbf, C=C, type = "C-svc", cross = 7)
err <- cross(model)
c_val <- c(c_val, C)
sig_val <- c(sig_val, sig)
e <- c(e, err)
}
}
```

```
df = data.frame(c_val, sig_val, e)
ggp <- ggplot(df, aes(c_val, sig_val)) + geom_tile(aes(fill = e))
ggp + scale_fill_gradient(low = "white", high = "black")
```

We will try to find precisely the minimum :

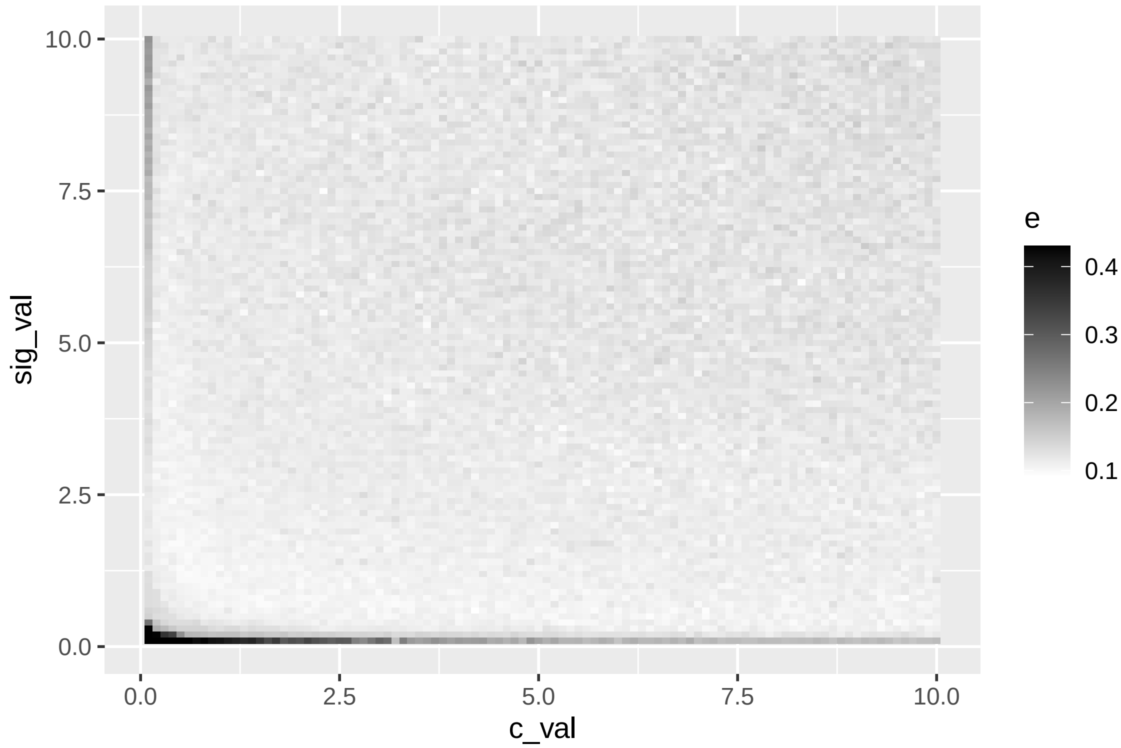


Figure 5: Visualisation de l'erreur en fonction de C et sigma

```
sig_hat <- sig_val[1]
c_hat <- c_val[1]
err_min <- e[1]
for (i in 1:length(e)){
if (e[i]<err_min){
err_min <- e[i]
sig_hat <- sig_val[i]
c_hat <- c_val[i]
}
}
sig_hat
```

```
## [1] 0.6
```

```
c_hat
```

```
## [1] 4.2
```

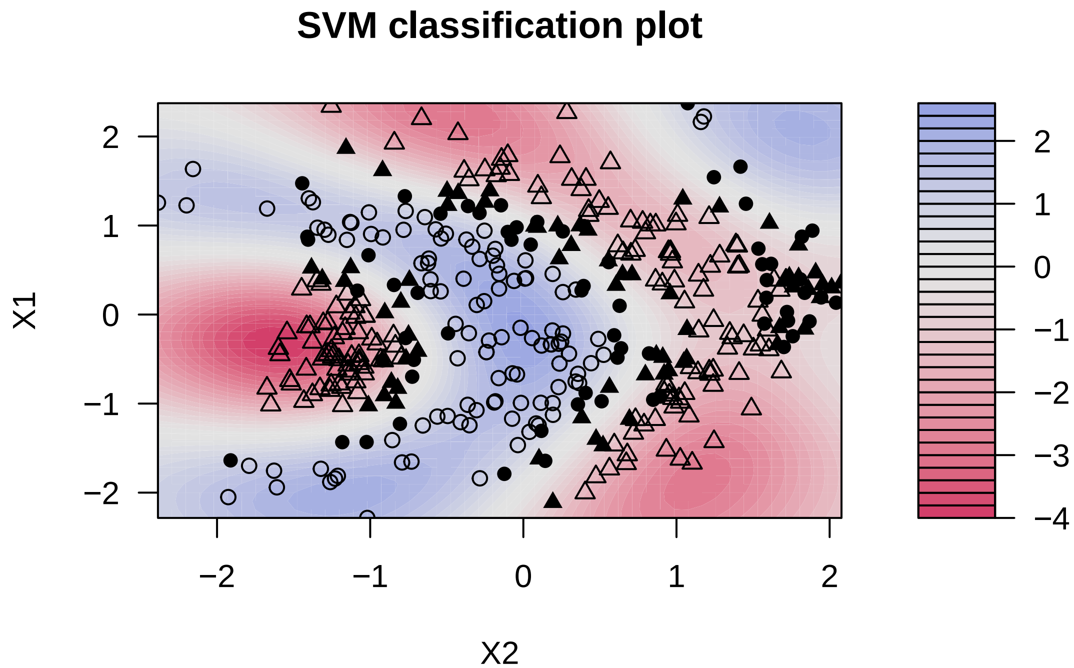
```
err_min
```

```
## [1] 0.0926022
```

Let fix the value of $\sigma = 0.6$ et C = 4.2 to try our model on the Test set .

Question 6. Build the optimal SVM model and evaluate this model on the training set. Report the test error rate.

```r
res.ksvm = ksvm(Y~., data=Apprentissage, kernel="rbfdot", type = "C-svc",
               kpar=list(sigma=sig_hat),C=c_hat,cross=7)

plot(res.ksvm, data = Apprentissage)
```




```r
res.ksvm@error
```

```
## [1] 0.0975
```

```r
res.ksvm@cross
```

```
## [1] 0.1100596
```

```r
res.ksvm@nSV
```

```
## [1] 131
```

```r
ytest_obs = factor(Test[, 3])
ytest_pred = factor(predict(res.ksvm, Test[, -3], type = "response"))

confusionMatrix(ytest_pred, ytest_obs)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   -1    1
```

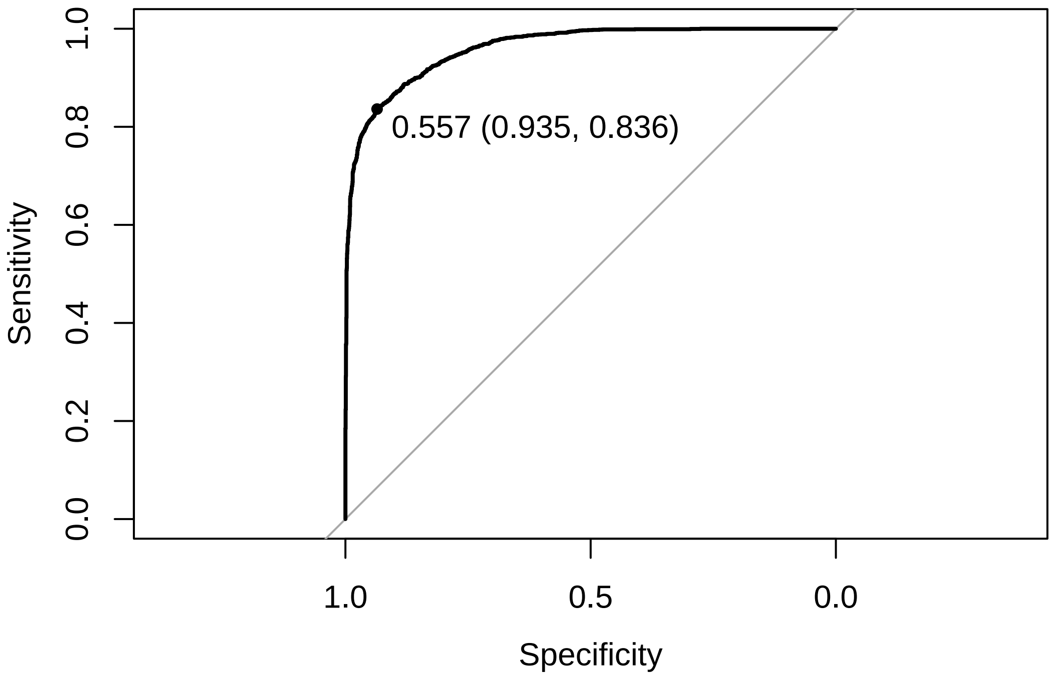
```
##         -1 2531  378
##         1   165 1826
##
##                Accuracy : 0.8892
##                  95% CI : (0.8801, 0.8978)
##     No Information Rate : 0.5502
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7741
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9388
##             Specificity : 0.8285
##          Pos Pred Value : 0.8701
##          Neg Pred Value : 0.9171
##              Prevalence : 0.5502
##          Detection Rate : 0.5165
##    Detection Prevalence : 0.5937
##       Balanced Accuracy : 0.8836
##
##        'Positive' Class : -1
##
```

```
res.ksvm = ksvm(Y~., data=Apprentissage, kernel="rbfdot", type = "C-svc",
                kpar=list(sigma=sig_hat),C=c_hat,cross=7, prob.model = TRUE)

prob_test = predict(res.ksvm, Test[, -3], type = "probabilities")


fit.roc = roc(Test[, 3], prob_test[, 2])
```

```
## Setting levels: control = -1, case = 1
```

```
## Setting direction: controls < cases
```

```
plot(fit.roc, print.thres = "best")

plot(fit.roc, print.thres = "best")
```




```
auc(fit.roc)
```

```
## Area under the curve: 0.9608
```

```
YYY = rep(1, NROW(Test))
YYY[prob_test[, 2]<0.507]=-1

sensitivity = round(1864/(1864+340), 3)
specificity = round(2474/(2474+222), 3)
```

Finally, we find an accuracy of 89% on the Test set, for values of $\sigma = 0.6$ and C $= 4.2$. That seems to be a very good score for our study.