Support Vector Machine

Julien JACQUES

07/01/2020

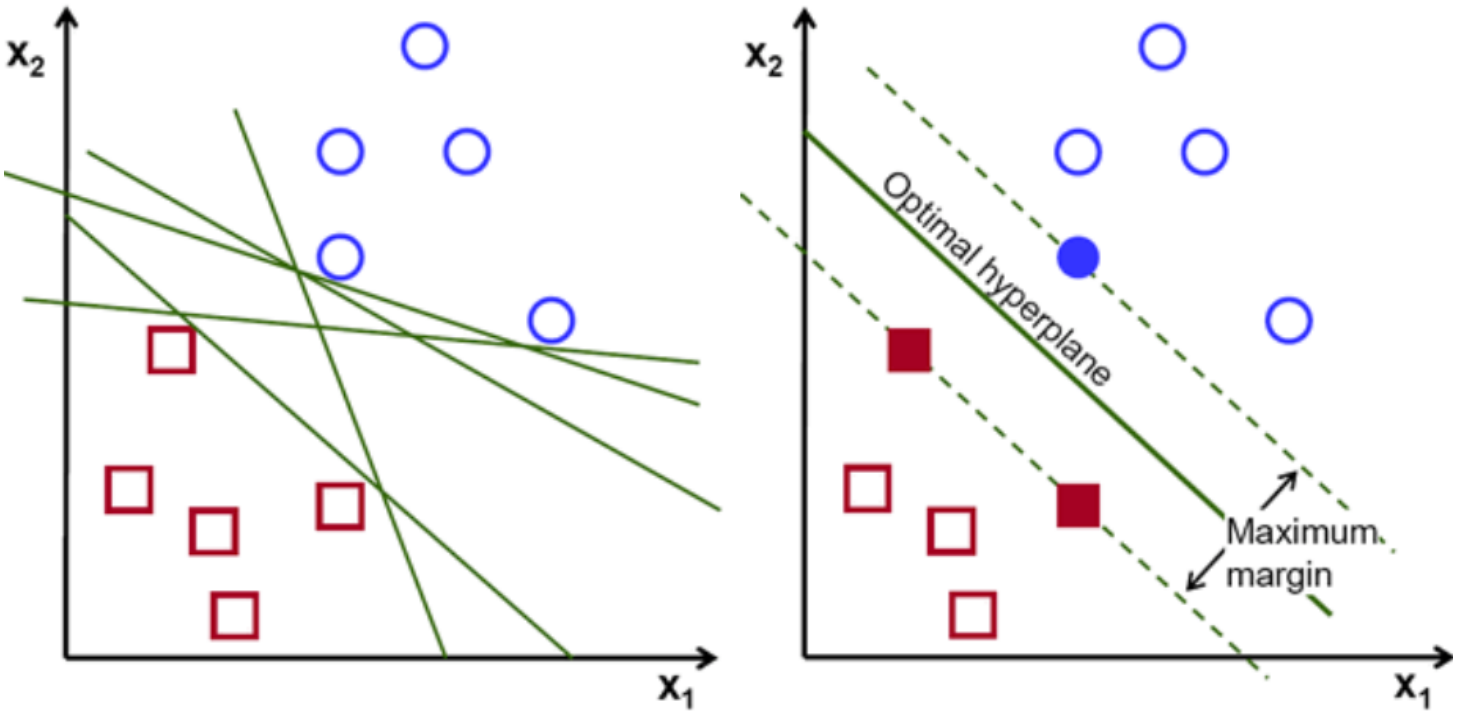
Support Vector Machine (SVM)

Notations :

- $x \in \mathbb{R}^p$ la matrice de données, dont :
  - chaque ligne x_i est un individu décrit par p variables
- $z = (z_1, \ldots, z_n)$ les étiquettes des individus
- on supposera qu'il n'y a pour l'instant que 2 classes : $z_i \in \{-1, 1\}$

Support Vector Machine (SVM)

Dans le cas où les classes sont **linéairement séparables**, l'objectif est de trouver l'hyperplan $y(x) = w^t x + b$ qui sépare au mieux les deux classes (**marge maximale**):

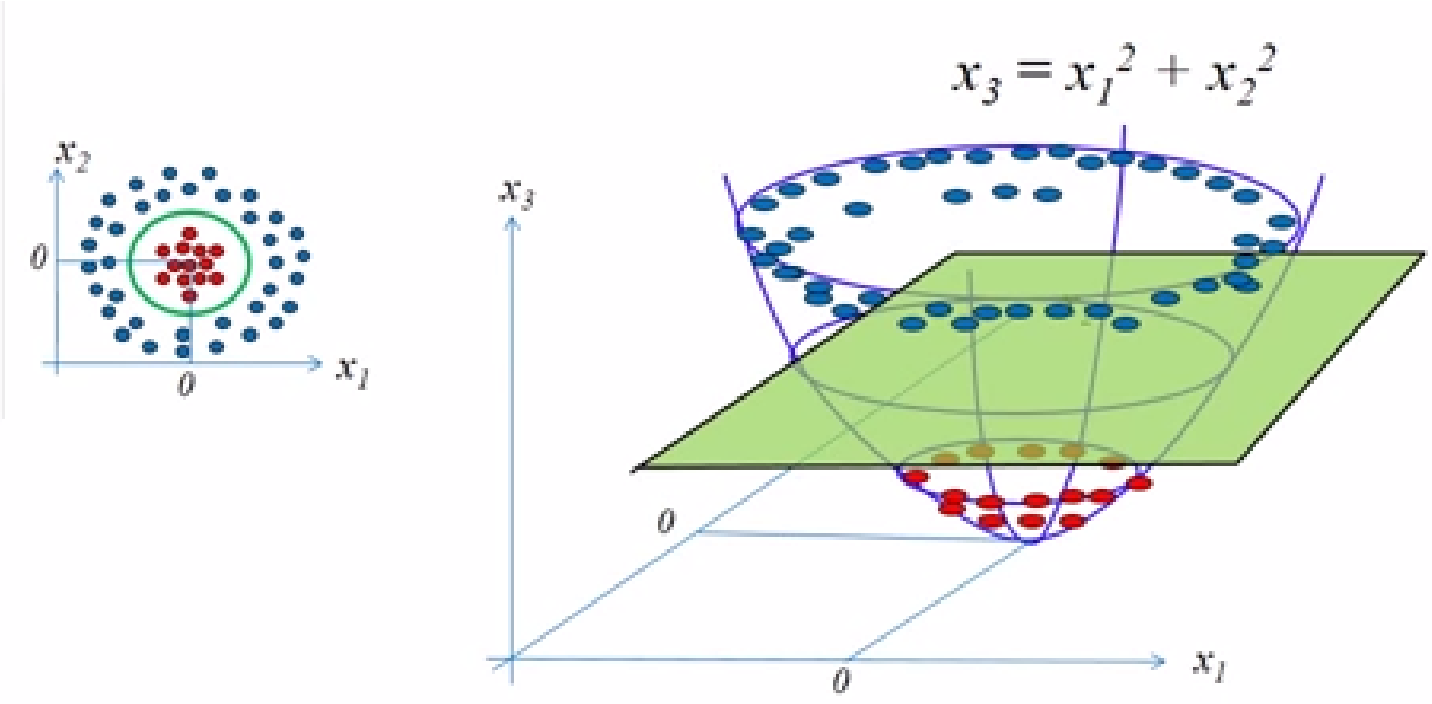


tq $y(x_i) > 0$ si $z_i = 1$ et $y(x_i) < 0$ si $z_i = -1$.

Le problème d'optimisation permettant de trouver (w, b) se résoud à l'aide d'un algorithme d'optimisation numérique.

Support Vector Machine - non linéaire

Bien souvent, les classes ne sont pas linéarement séparables, mais on peut changer de variables $x \rightarrow \phi(x)$ pour que cela le devienne



Le problème d'optimisation consiste alors à trouver les (w, b) de l'hyperplan $y(x) = w^t\phi(x) + b$

Mais la recherche du changement de variable peut-être coûteuse...

Support Vector Machine - kernel trick

Tout l'intérêt des SVM réside dans l'**astuce du noyau** (kernel trick):

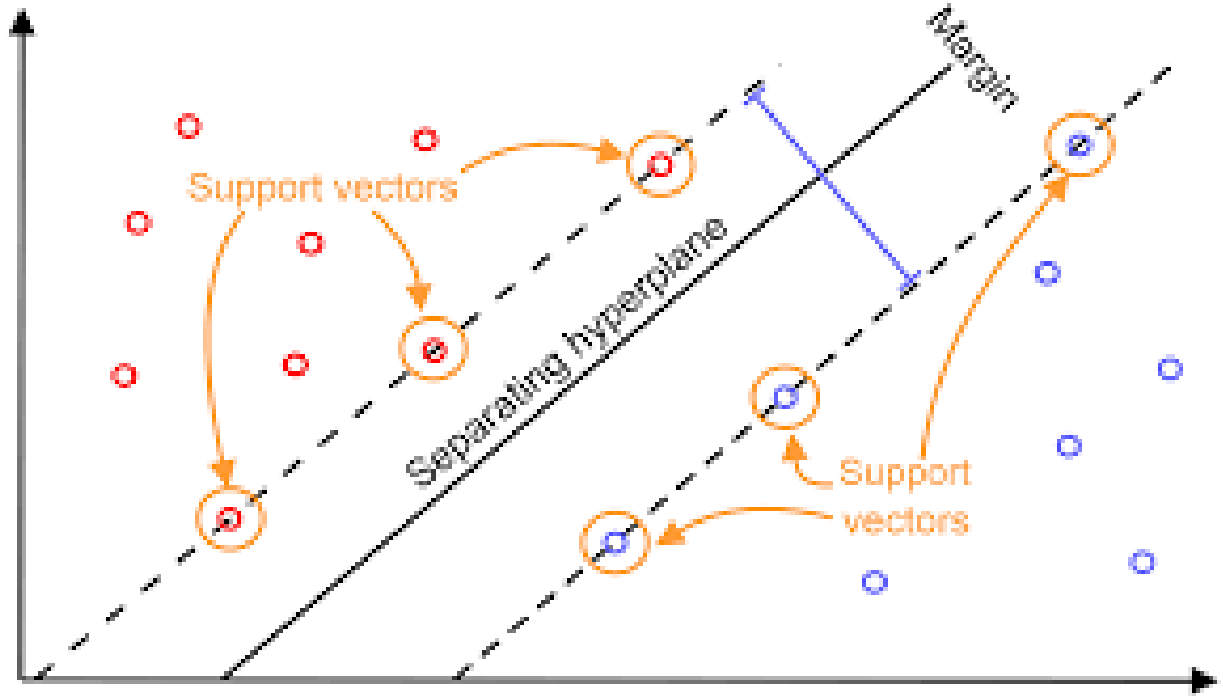
- on peut transformer le problème d'optimisation en un problème dual où seuls interviennents les produits $\phi(x)\phi(x')$ pour tous les (x, x') de l'échantillon d'entrainement
- or, pour un certain nombre de fonction ϕ, il est beaucoup moins coûteux (en terme de nombre d'opération à effectuer) de calculer directement ces produits scalaire plutôt que explicitement les $\phi(x)$
- on appelle **noyau** la fonction $k(x, x') = \phi(x)\phi(x')$

Plutôt que de choisir la transformation ϕ, on choisira parmi un certains nombre de noyaux.

Ce noyau ϵ est un hyper-paramètre de la méthode qu'il faudra optimiser (validation croisée parmi un ensemble de noyau pré-définis).

Support Vector Machine (SVM) - support vectors

Les **supports vectors**, sont les points qui sont à plus proche distance de l'hyperplan séparateur :

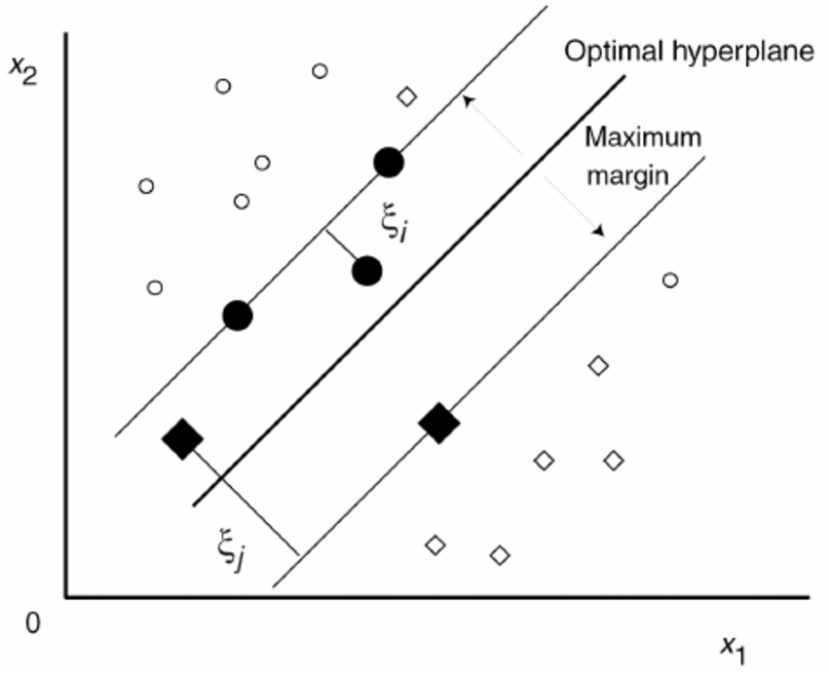


Support Vector Machine (SVM) - prédiction

Pour faire une **prédiction en SVM** pour de nouvelles observations, il suffit de les *comparer* uniquement à ces supports vectors et non à tout l'ensemble d'apprentissage, ce qui est extrément efficace d'un point de vue calculatoire.

Support Vector Machine - cas non séparable

Dans les cas compliqués où les données ne sont difficilement séparables on va permettre à quelques points de pénétrer d'un ϵ dans la marge (dont la largeur est fixée à 1) :



Le problème d'optimisation s'assouplie alors, mais on va néanmoins pénaliser celui-ci en fonction de ces ϵ (comme en régression pénalisée)

Cet ϵ est un hyper-paramètre de la méthode qu'il faudra optimiser (validation croisée).

Support Vector Machine - cas multiclasse

Les SVM sont essentiellement définis pour le cas binaire.

Des extensions existent pour le cas multiclasse, basés pour la pluspart sur l'utilisation de plusieurs SVM binaire pour prédire une classe contre toutes les autres.

Support Vector Machine sous R

```
library(e1071)
```

On commence par transformer les données iris en un problème binaire (et on normalise les données)

```
setosa <- as.factor(iris$Species == "setosa")
iris2 = scale(iris[,-5])
```

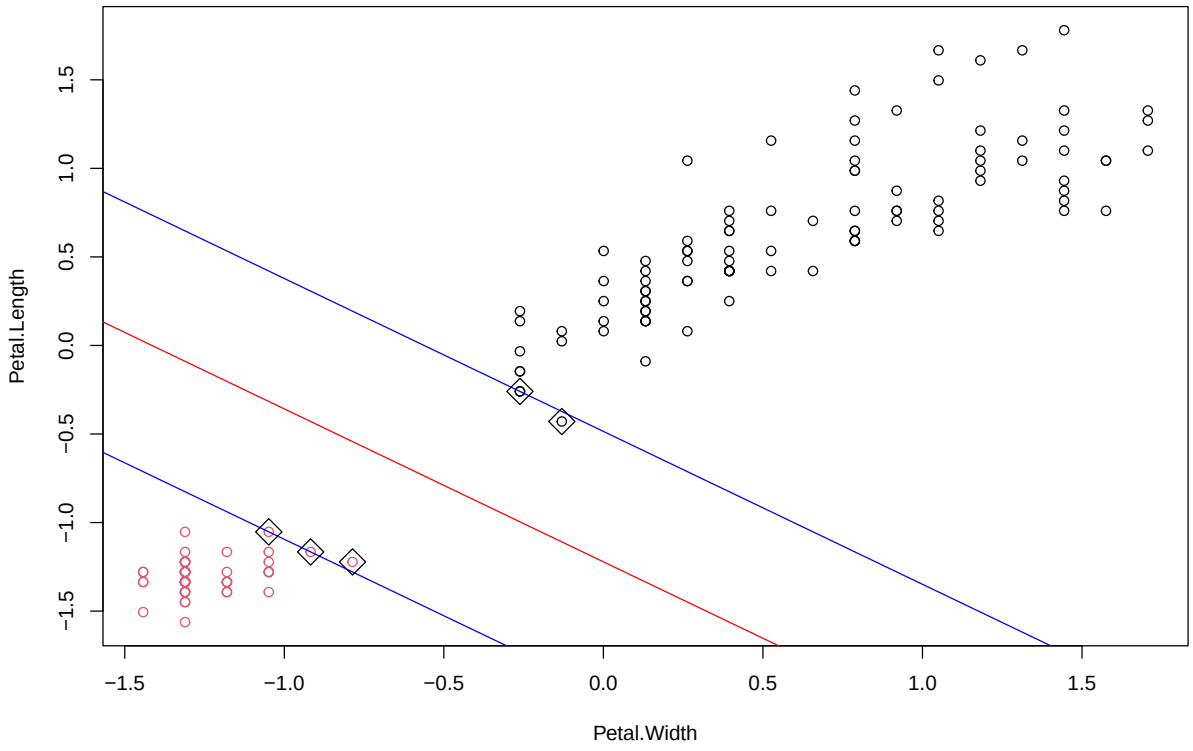
On estime alors un SVM linéaire, et on stocke les coefficients

```
m <- svm(setosa ~ Petal.Width + Petal.Length,
         data = iris2, kernel = "linear")
cf=coef(m)
```

Support Vector Machine sous R

Puis on va représenter l'hyperplan, et les supports vectors

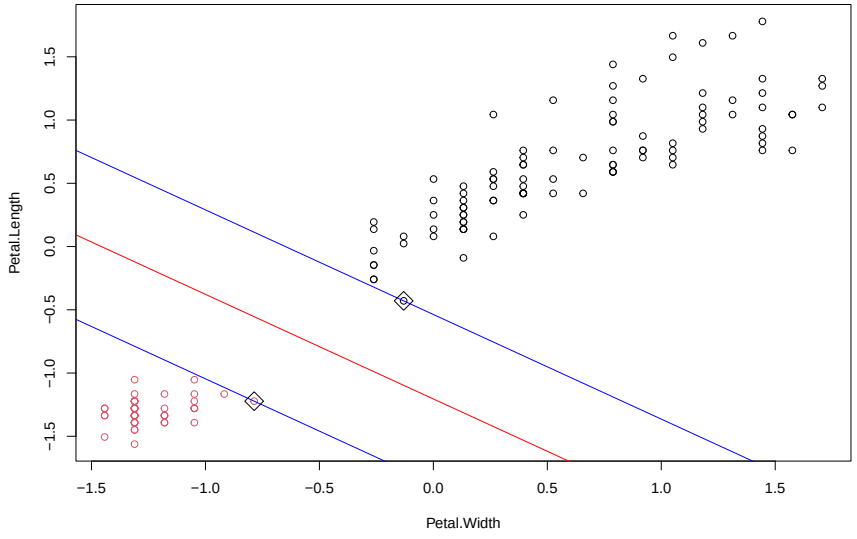
```
plot(Petal.Length~Petal.Width, data=iris2, col=setosa)
abline(-cf[1]/cf[3], -cf[2]/cf[3], col = "red")
abline(-(cf[1] + 1)/cf[3], -cf[2]/cf[3], col = "blue")
abline(-(cf[1] - 1)/cf[3], -cf[2]/cf[3], col = "blue")
points(m$SV, pch = 5, cex = 2)
```



Support Vector Machine sous R

On peut durcir la contrainte en augmentant le coût ϵ de laisser des données pénétrer dans la marge (paramètre cost de la fonction svm)

```
m <- svm(setosa ~ Petal.Width + Petal.Length,
         data = iris2, kernel = "linear",cost=100)
cf=coef(m)
plot(Petal.Length~Petal.Width, data=iris2, col=setosa)
abline(-cf[1]/cf[3], -cf[2]/cf[3], col = "red")
abline(-(cf[1] + 1)/cf[3], -cf[2]/cf[3], col = "blue")
abline(-(cf[1] - 1)/cf[3], -cf[2]/cf[3], col = "blue")
points(m$SV, pch = 5, cex = 2)
```



SVM multi-classes

```
library(kernlab)
ind_app=sample(1:150,100)
svm1=ksvm(Species~., data=iris[ind_app,],kernel="rbfdot",C=10)
pred = predict(svm1,newdata=iris[-ind_app,],
                    type="response")
table(pred, iris[-ind_app,5])
```

```
##
## pred          setosa versicolor virginica
##   setosa          21          0         0
##   versicolor       0         10         5
##   virginica        0          0        14
```

```
cat('taux de bons classements :',mean(pred==iris[-ind_app,5]))
```

```
## taux de bons classements : 0.9
```