Module 3 - Méthodes d'apprentissage avec R - Séance 6

DUBii 2019

Frédéric Guyon

2019-02-26

Support Vecteur Machine

Deux classes a priori connues

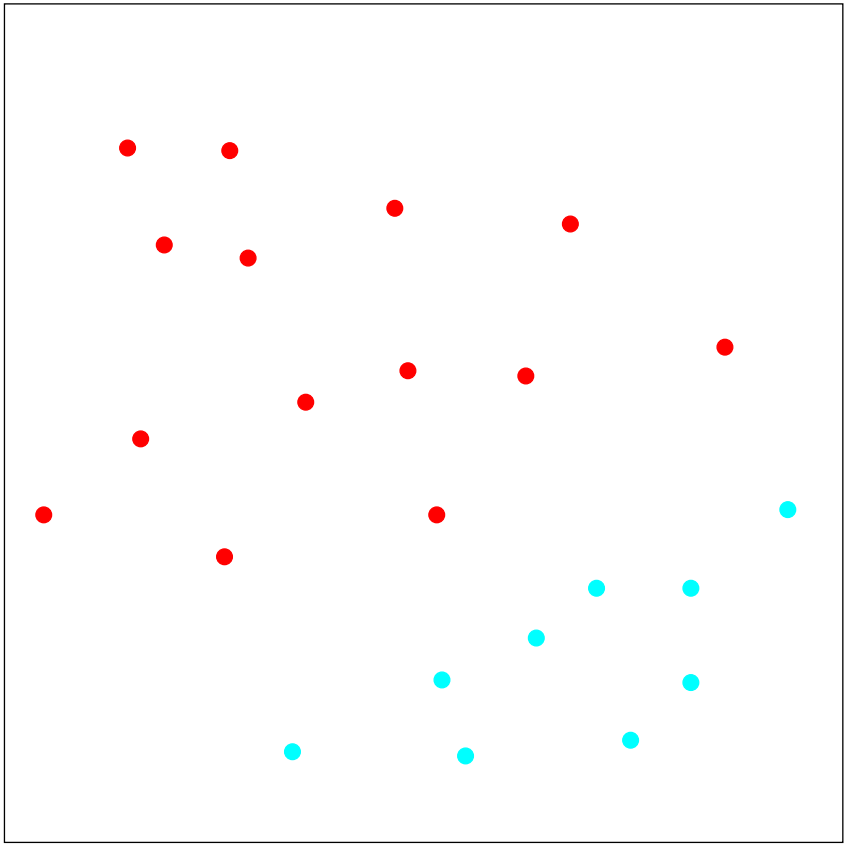


Figure 1: 2 groupes

Séparation linéaire

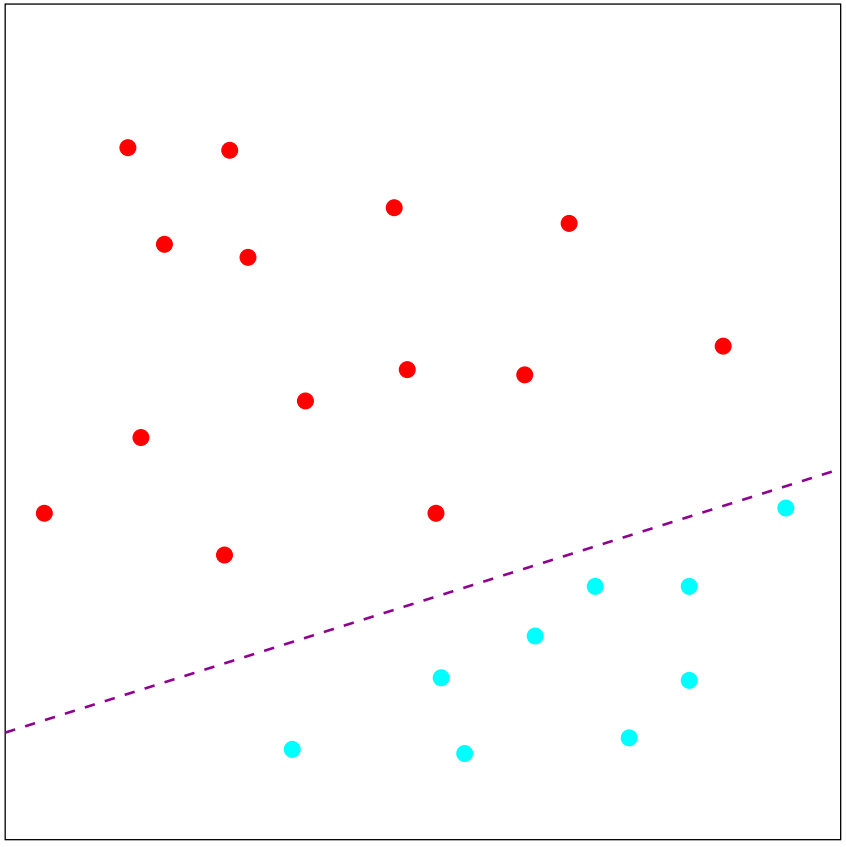


Figure 2: 2 groupes

Modèle d'apprentissage

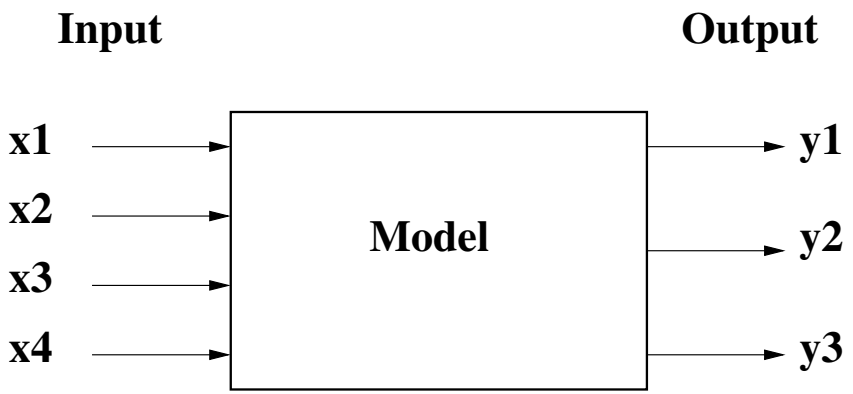


Figure 3: black-box model: SVM ou NN

Modèle d'apprentissage

- les vecteurs d'apprentissage x_i: représentation de nos objets à classer
- classe ou valeur attendue pour chaque objet: y_i
- modèle = paramètres à identifier
  - poids des réseaux de neurones
  - coefficients des hyperplan séparateurs pour les SVM
  - obtenus par minimisation d'une fonction d'écart (loss function)
- fonction de décision: classe=F(x,par)

Historique rapide

- **1957** : Perceptron (Rosenblatt)
- **1980** : Artificial Neural Network (K. Fukushima)
- **1989** : Algorithme de Back-propagation du gradient appliqué à NN à plusieurs couches (ZIP codes)
- Problème : apprentissage difficile et long (vanishing gradient)
- **1990-2000** : Problèmes de convergence, lenteur ont favorisés l'émergence des SVM
- **2007** : apparition du terme Deep Learning (Hinton)
- Actuellement, méthodes les plus performantes sur benchmarks d'évaluation : TIMIT (Reconnaissance de la parole), MNIST (images de chiffres manuscrits)

Support Vecteur Machine

Séparation linéaire: deux classes

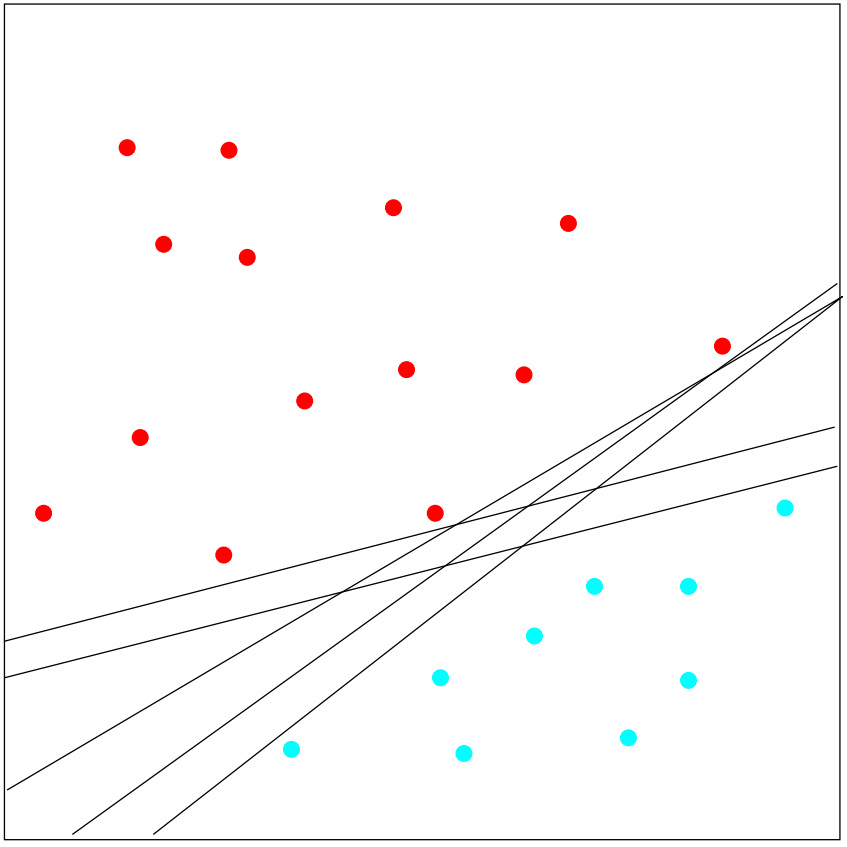
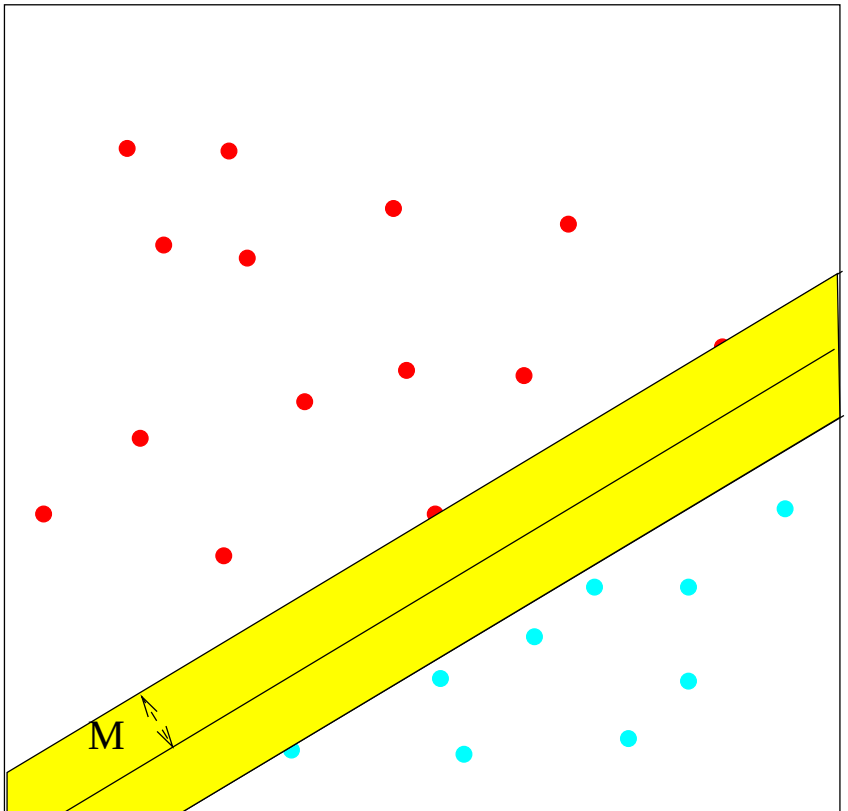


Figure 4: 2 groupes, pas de s<U+00E9>paration unique

Marge entre classes séparables

- Marge: distance au point le plus proche
- Recherche du plan qui maximise cette marge
- Marge large = plus grande stabilité des prédictions
- SVM = Séparateur à Vaste Marge



Approche SVM dans le cas général

- Plan séparateur trouvé par minimisation des erreurs de classification
- On n'interdit pas les erreurs de classement, mais on les pénalise
- Paramètre C ou cost = constante de pénalisation pour contrôler les erreurs de classement

Approche SVM dans le cas général

- Souvent problèmes de classification plus faciles dans un espace plus grand (espace de redescription)
- Plus de degrés de libertés pour trouver un modèle
- Séparation linéaire possible dans l'espace de redescription
- Cet espace est décrit par une fonction noyau

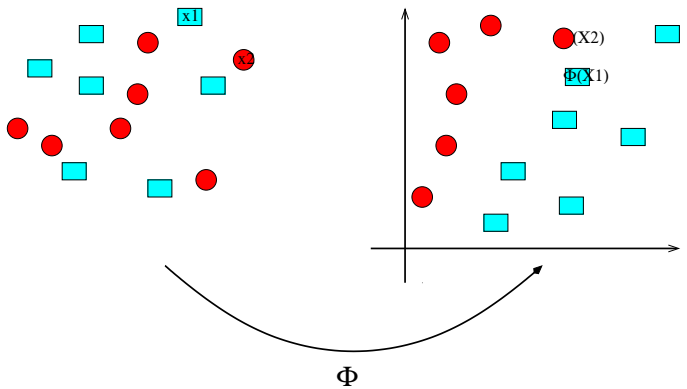
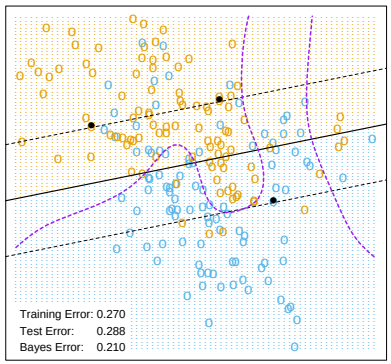
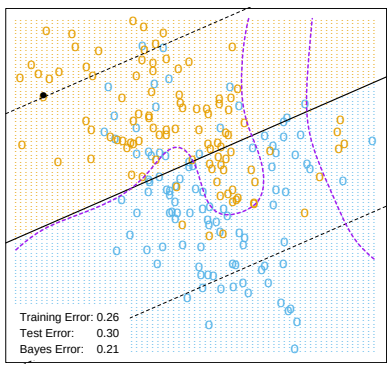


Figure 6: Espace des features

Exemples



$C = 10000$



$C = 0.01$

FIGURE 12.2. *The linear support vector boundary for the mixture data example with two overlapping classes, for two different values of C. The broken lines indicate the margins, where $f(x) = \pm 1$. The support points ($\alpha_i > 0$) are all the points on the wrong side of their margin. The black solid dots are those support points falling exactly on the margin ($\xi_i = 0$, $\alpha_i > 0$). In the upper panel 62% of the observations are support points, while in the lower panel 85% are. The broken purple curve in the background is the Bayes decision boundary.*

Figure 7: S<U+00E9>paration lin<U+00E9>aire

Exemples

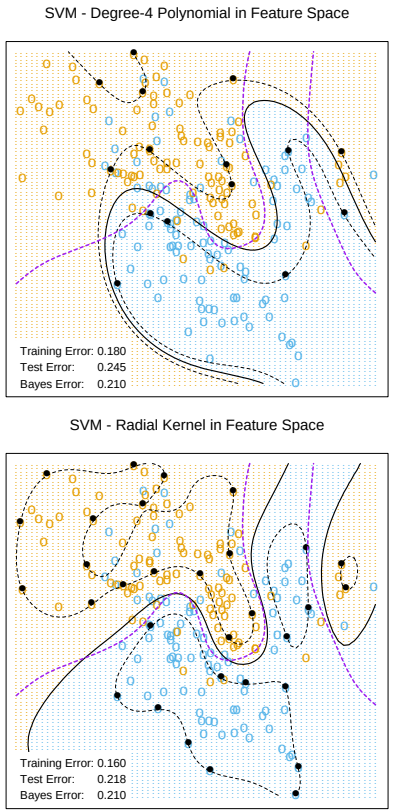


FIGURE 12.3. *Two nonlinear SVMs for the mixture data. The upper plot uses a 4th degree polynomial kernel, the lower a radial basis kernel (with $\gamma = 1$). In each case C was tuned to approximately achieve the best test error performance, and $C = 1$ worked well in both cases. The radial basis kernel performs the best (close to Bayes optimal), as might be expected given the data arise from mixtures of Gaussians. The broken purple curve in the background is the Bayes decision boundary.*

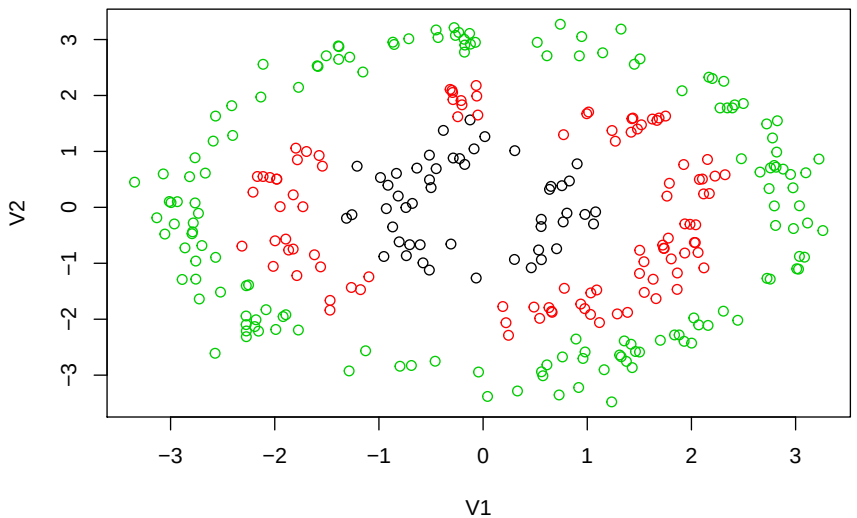
Figure 8: S<U+00E9>paration polyn<U+00F4>miale et gaussienne

Exemple 1: iris avec Support Vector Machine

```
library(e1071)
model=svm(Species~., data=iris)
ypred=predict(model,iris[,1:4])
table(ypred, iris[,5])
```

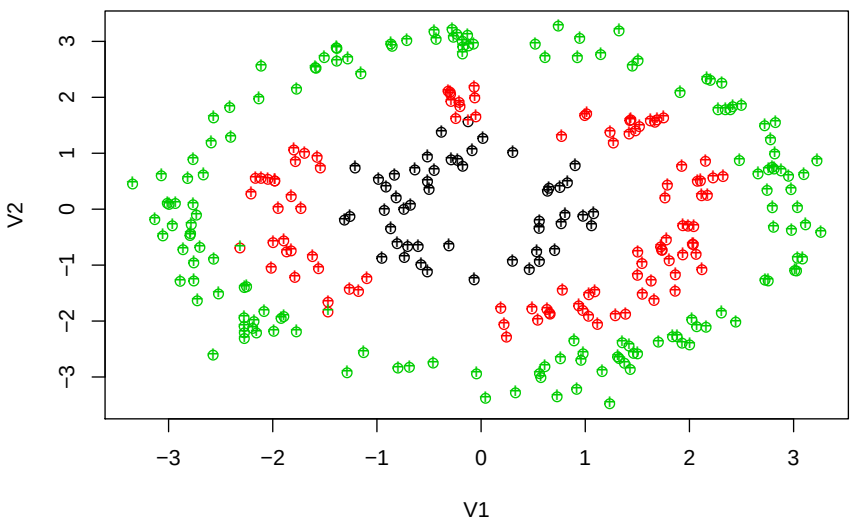
Exemple 2: classification non séparable

```
Data=read.table("data/cercles.dms")
X=as.matrix(Data[,1:2])
y=Data[,3]
plot(X,col=y)
```



Exemple 2: classification non séparable

```
result=svm(X,y, type="C-classification",kernel="radial", co
ypred=predict(result,X)
table(y, ypred)
plot(X, col=y)
points(X,col=ypred, pch="+")
```



SVM: les caractéristiques générales

- Un choix de noyau = forme des frontières
- Le paramètre C règle le nombre d'erreurs d'apprentissage
- Plus C est grand:
  - moins les frontières sont régulières (smooth)
  - plus le nombre d'erreurs d'apprentissage est faible (tendance)
  - le nombre d'erreurs de test peut être plus faible
- $C \longrightarrow \infty$: aucun mauvais classement mais sur-apprentissage
- En général l'algorithme renvoie les "support vectors"
- support vecteurs: données (points) utilisées pour la séparation

Réseaux de neurones : le neurone

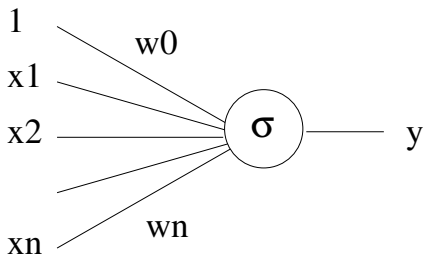


Figure 9: One single neurone

$$y = \sigma\left(w_0 + w_1 x_1 + w_2 x_2 + \dots\right)$$

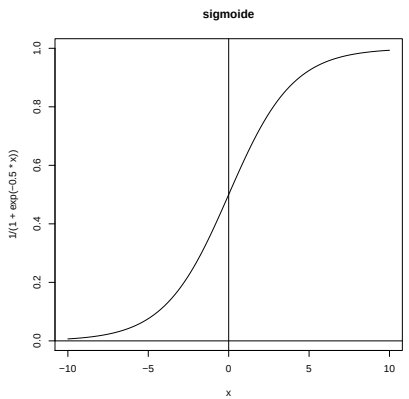


Figure 10: A sigmoid function

Réseaux de neurones : le neurone

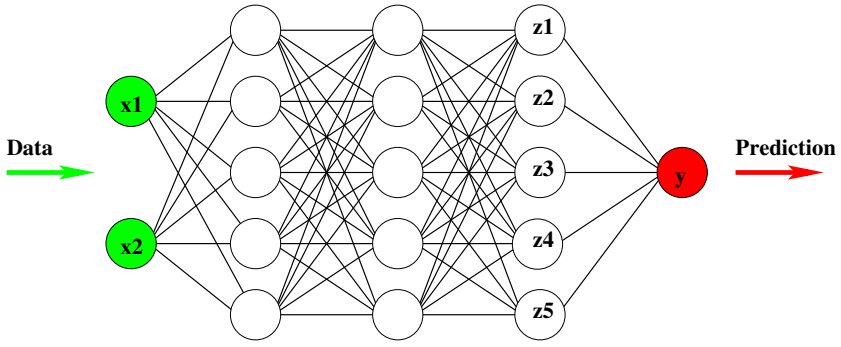


Figure 11: A neural network

- couche d'entrée (input layer)
- couches cachées (hidden layers)
- couche de sortie (output layer)
- package nnet
  - une seule couche cachée
  - une couche de sortie avec fonction d'activation linéaire ou softmax

Fonction nnet: 1 couche cachée

Exemple iris avec réseau de neurones

```
library(nnet)
X=as.matrix(iris[,1:4])
y=as.integer(iris[,5])
# une première façon
Y=class.ind(y)
model=nnet(X,Y, size=2, softmax=TRUE)
ypred=max.col(predict(model,X))
table(y, ypred)
```

Ou bien

```
model=nnet(Species~.,data=iris, size=2)
ypred=max.col(predict(model,X))
table(y, ypred)
```

Plus sérieusement, avec évaluation

```
ind_app=sample(1:nrow(iris),50)
Xapp=iris[ind_app,1:4]
yapp=iris[ind_app,5]
Xtest=iris[-ind_app,1:4]
ytest=iris[-ind_app,5]

Yapp=class.ind(yapp)
model=nnet(Xapp,Yapp, size=2, softmax=TRUE)
```

Evaluation des erreurs

```
# erreur d'apprentissage
ypred=max.col(predict(model,Xapp))
table(yapp, ypred)

# erreur de test
ypred=max.col(predict(model,Xtest))
table(ytest, ypred)
```