Classification automatique

Chapitre 5 : Classification spectrale et méthodes basées sur la densité

Allou Samé
allou.same@ifsttar.fr

Plan

1. Classification spectrale
   - Graphe de similarité et matrice laplacienne
   - Algorithmes de classification spectrale
   - Liens avec les critères de coupure de graphe
   - Classification spectrale sous R

2. Classification basée sur la densité
   - Algorithme DBSCAN
   - Application sous R

Introduction

Objectifs

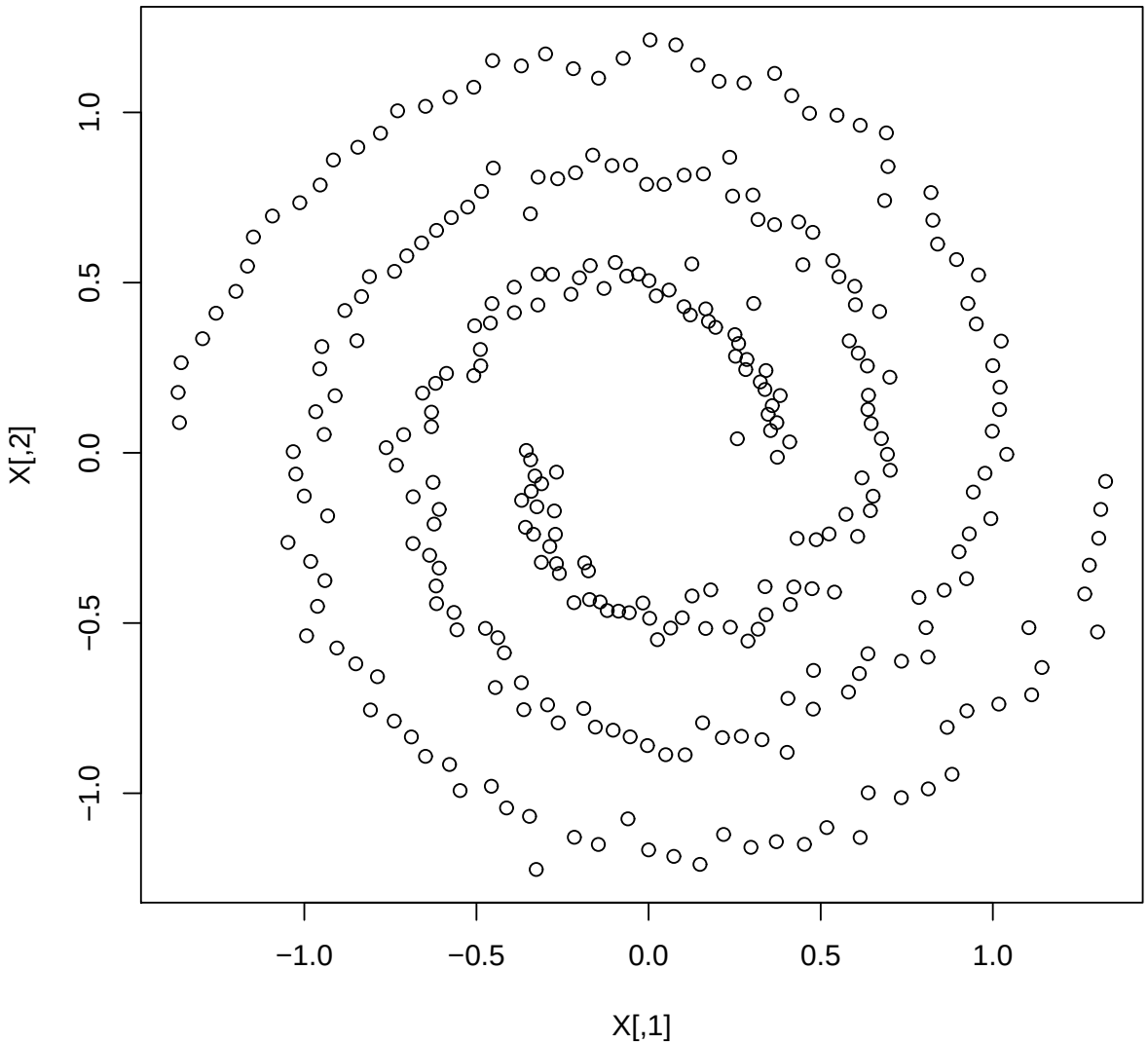
Trouver une partition d'un ensemble de données $E = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ en K classes en se basant sur l'analyse spectrale d'un graphe de similarité

Avantages

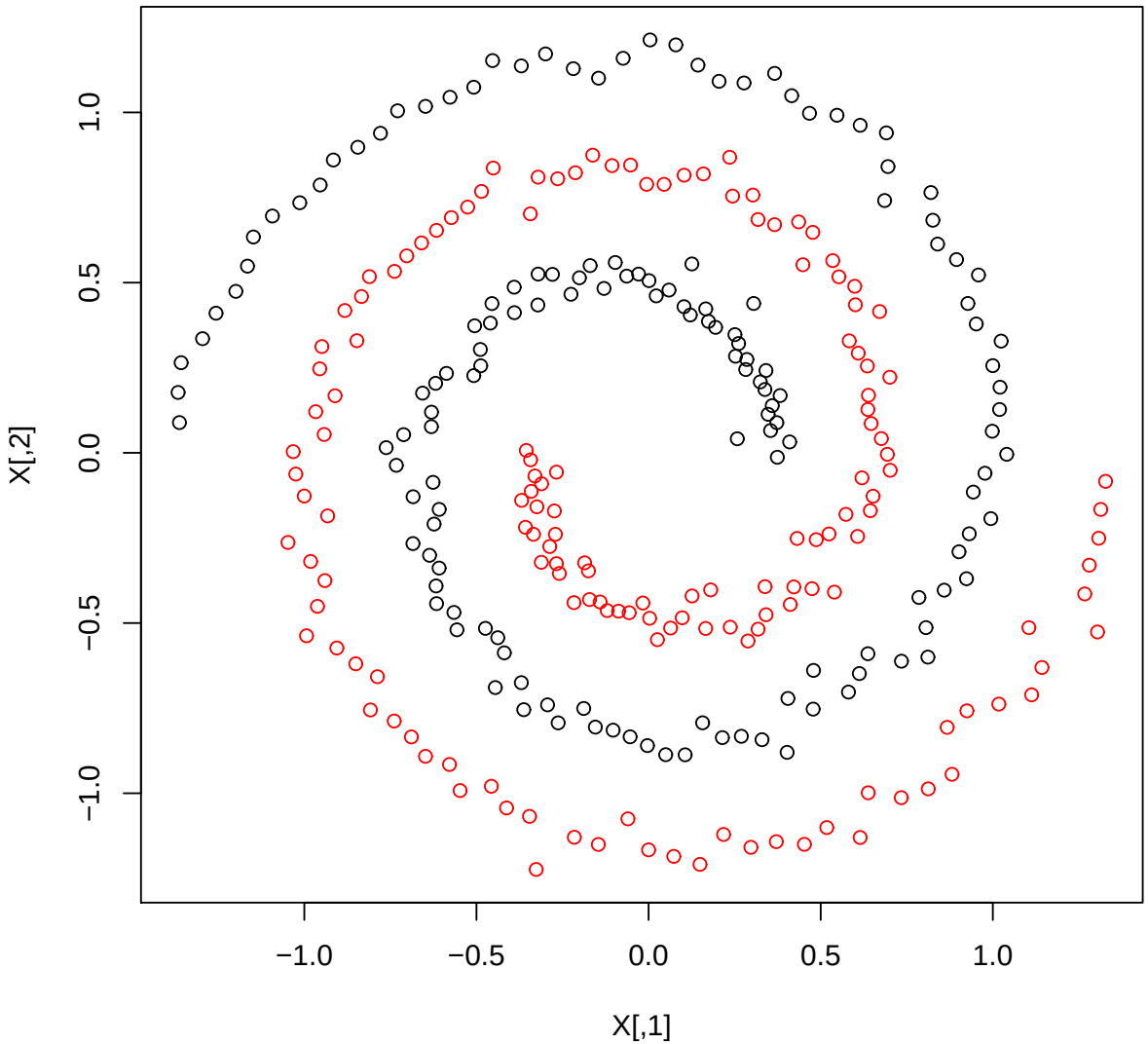
Le clustering spectral peut présenter quelques avantages par rapport aux méthodes étudiées dans les chapitres précédents :

- sa simplicité d'implémentation : essentiellement basée sur des décompositions algébriques (calcul de valeurs et de vecteurs propres)
- sa capacité à extraire des classes non linéairement séparables (exemple de classes en forme de spirale) ; ce qui peut présenter un intérêt dans certaines applications comme la vision par ordinateur

Exemple de classes à séparation non linéaire



Exemple de classes à séparation non linéaire



Graphe de similarité et partitionnement de données

Données : ensemble des observations à classifier $E = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$

Graphe non orienté : on calcule à partir de l'ensemble E une matrice de similarité $W = (w_{ij})_{1 \leq i,j \leq n}$ assimilée à un graphe

- Les sommets (nœuds) du graphe sont les n observations à classifier
- Il existe une arête entre deux nœuds ssi $w_{ij} > 0$
- Le graphe étant non orienté, on a $w_{ij} = w_{ji}$

Matrice des degrés : matrice diagonale $D \in \mathbb{R}^{n \times n}$ dont les éléments diagonaux sont les degrés des sommets : $d_i = \sum_{j=1}^{n} w_{ij}$

Cardinal et volume : soit $A \subset E$; on définit le volume et le cardinal de A par $|A| = cardinal(A)$ et $vol(A) = \sum_{\mathbf{x}_i \in A} d_i$

Composantes connexes : d'après la théorie des graphes, les composantes connexes du graphe W définissent une partition de E

Exemple de similarités

Similarité basée sur un seuil ε ($\varepsilon-$neighborhood graph)

Deux points $\mathbf{x}$ et $\mathbf{y}$ sont connectés si $d(\mathbf{x}, \mathbf{y}) \leq \varepsilon$.

Similarités basées sur les plus proches voisins ($k-$nearest neighborhood graph)

- $\mathbf{x}$ et $\mathbf{y}$ sont connectés si $\mathbf{x}$ fait partie des m plus proches voisins de $\mathbf{y}$ ou si $\mathbf{y}$ fait partie des m plus proches voisins de $\mathbf{x}$.
- $\mathbf{x}$ et $\mathbf{y}$ sont connectés si $\mathbf{x}$ fait partie des m plus proches voisins de $\mathbf{y}$ et si $\mathbf{y}$ fait partie des m plus proches voisins de $\mathbf{x}$ (plus proches voisins mutuels).

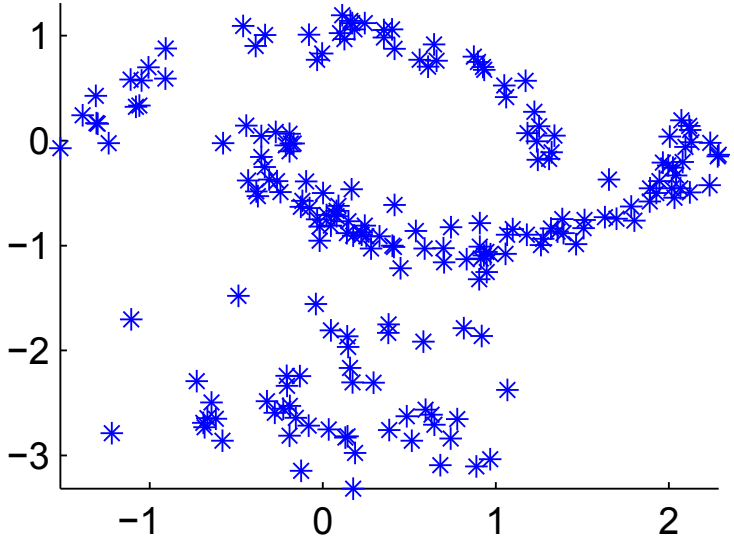
Similarité gaussienne (graphe complet)

Graphe complet dont les poids sont donnés par $w_{ij} = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$.

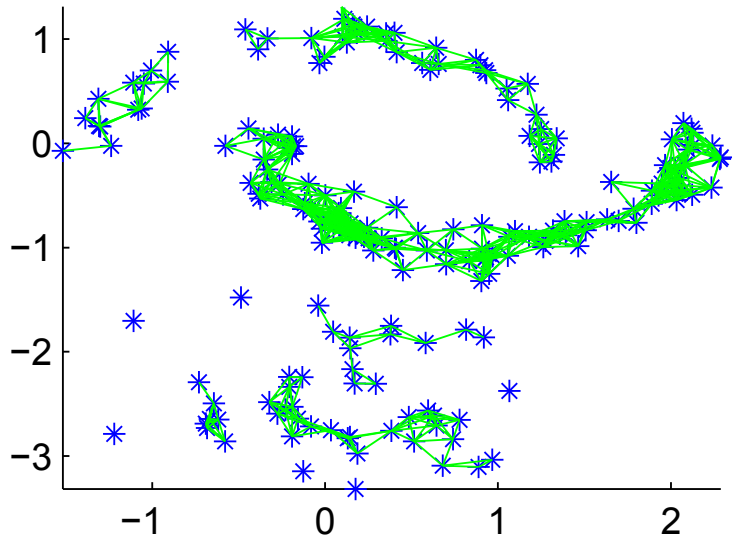
Le paramètre σ contrôle la « largeur » des voisinages, tout comme le paramètre ε ci-dessus

Illustrations des graphes de similarité

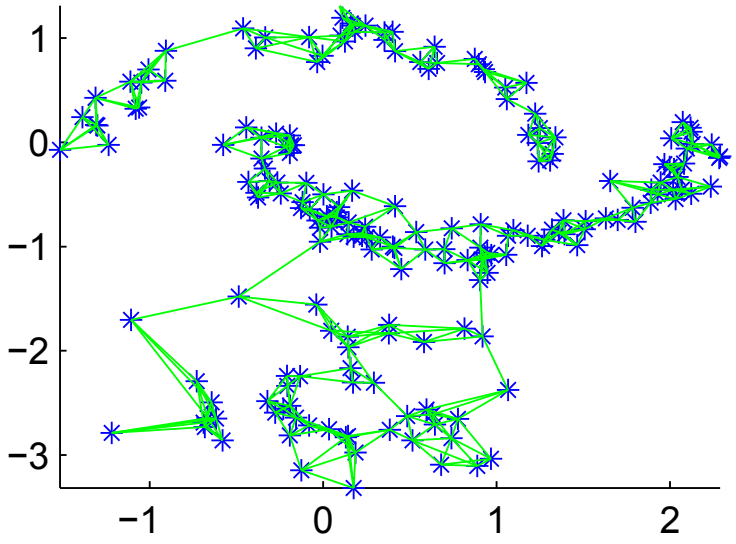
Données (3 classes)



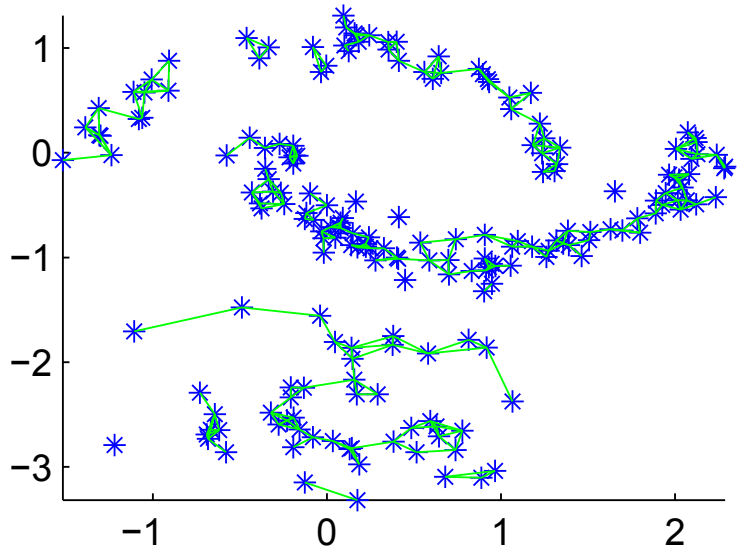
$\varepsilon-$graph ($\varepsilon = 0.3$)



kppv ($k = 5$)



kppv mutuels $k = 5$



Matrice Laplacienne du graphe W

Matrice Laplacienne non normalisée

$$L = D - W$$

Propriétés

(1) $f' L f = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2 \qquad \forall f \in \mathbb{R}^n.$

(2) L est symétrique semi-définie positive.

(3) L possède n valeurs propres réelles $0 \leq \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$.

(4) La plus petite valeur propre de L est 0. Elle est associée au vecteur propre $\mathbb{1}_n = (1, \ldots, 1)^T$.

(5) La multiplicité de la valeur propre $\lambda_1 = 0$ est le nombre de composantes connexes $P_1, \ldots, P_K$ du graphe

Matrices Laplaciennes normalisées

$$\begin{aligned} L_{sym} &= D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}} \\ L_{rw} &= D^{-1} L = I - D^{-1} W \end{aligned}$$

Propriétés

(1) pour tout vecteur $f \in \mathbb{R}^n$,

$$f' L_{sym} f = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} (\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}})^2$$

(2) λ est valeur propre de L_{rw} associée au vecteur propre u ssi
λ est valeur propre de L_{sym} associée au vecteur propre $D^{1/2}u$

(3) λ est valeur propre de L_{rw} associée au vecteur propre u ssi
$L\,u = \lambda\, D\, u$ (λ et u : valeur et vecteur propre généralisés)

Matrices Laplaciennes normalisées

Propriétés

(4) 0 est valeur propre de L_{rw} (resp. L_{sym}) associée au vecteur propre $\mathbb{1}$ (resp. $D^{1/2}\mathbb{1}$)

(6) L_{sym} et L_{rw} sont symétriques semi-définies positives et possèdent n valeurs propres positives ou nulles

(5) La multiplicité de la valeur propre 0 de L_{rw} (resp. L_{sym}) est le nombre de composantes connexes $P_1, \ldots, P_K$ du graphe

Algorithme de classification spectrale non normalisé

1. Construire le graphe de similarité $W \in \mathbb{R}^{n \times n}$.
2. Calculer la matrice Laplacienne L.
3. Calculer les valeurs propres et les vecteurs propres orthonormés de L.
4. Former la matrice $U \in \mathbb{R}^{n \times K}$ dont les colonnes sont les K vecteurs propres associés aux K plus petites valeurs propres de L rangées par ordre croissant.
5. Pour $i = 1, \ldots, n$, construire le vecteur $\mathbf{y}_i \in \mathbb{R}^K$ obtenu par transposition de la i^{e} ligne de U.
6. Partitionner l'ensemble $\{\mathbf{y}_1, \ldots, \mathbf{y}_n\}$ en K classes $C_1, \ldots, C_K$ par l'algorithme des k-means.
7. En déduire la partition $(P_1, \ldots, P_k)$ de E telle que $P_k = \{\mathbf{x}_i \mid \mathbf{y}_i \in C_k\}$.

Algorithme de classification spectrale normalisé (Shi et Malik, 2000)

1. Construire le graphe de similarité $W \in \mathbb{R}^{n \times n}$.
2. Calculer la matrice Laplacienne normalisée L_{rw}.
3. Calculer les valeurs propres et les vecteurs propres orthonormés de L_{rw}.
4. Former la matrice $U \in \mathbb{R}^{n \times K}$ dont les colonnes sont les K vecteurs propres associés aux K plus petites valeurs propres de L_{rw} rangées par ordre croissant.
5. Pour $i = 1, \ldots, n$ construire le vecteur $\mathbf{y}_i \in \mathbb{R}^K$ obtenu par transposition de la i^{e} ligne de U.
6. Partitionner l'ensemble $\{\mathbf{y}_1, \ldots, \mathbf{y}_n\}$ en K classes $C_1, \ldots, C_K$ par l'algorithme des k-means.
7. En déduire la partition $(P_1, \ldots, P_k)$ de E telle que $P_k = \{\mathbf{x}_i \mid \mathbf{y}_i \in C_k\}$.

Algorithme de classification spectrale normalisé (Ng, Jordan et Weiss, 2002)

1. Construire le graphe de similarité $W \in \mathbb{R}^{n \times n}$.
2. Calculer la matrice Laplacienne normalisée L_{sym}.
3. Calculer les valeurs propres et les vecteurs propres orthonormés de L_{sym}.
4. Former la matrice $V \in \mathbb{R}^{n \times K}$ dont les colonnes sont les K vecteurs propres associés aux K plus petites valeurs propres de L_{sym}.
5. Calculer la matrice U en normalisant chaque ligne de la matrice V, c'est-à-dire en posant $u_{ij} = \frac{v_{ij}}{\sqrt{\sum_{\ell=1}^{K} v_{i\ell}^2}}$
6. Pour $i = 1, \ldots, n$ construire le vecteur $\mathbf{y}_i \in \mathbb{R}^K$ obtenu par transposition de la i^{e} ligne de U.
7. Partitionner l'ensemble $\{\mathbf{y}_1, \ldots, \mathbf{y}_n\}$ en K classes $C_1, \ldots, C_K$ par l'algorithme des k-means.
8. En déduire la partition $(P_1, \ldots, P_k)$ de Ω telle que $P_k = \{\mathbf{x}_i \mid \mathbf{y}_i \in C_k\}$.

Remarque sur les algorithmes de classification spectrale

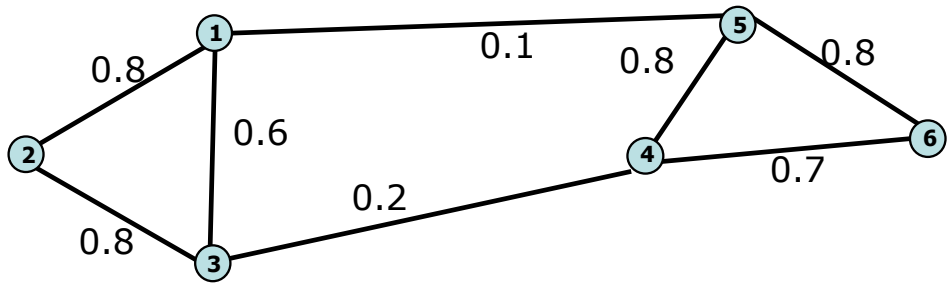
- La méthode utilisée par chaque algorithme consiste à représenter les données initiales $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ dans l'espace spectral (celui des vecteurs propres) qui met davantage en évidence les classes de sorte qu'une simple utilisation de l'algorithme des k-means suffise pour classer les observations dans l'espace propre.
- Les trois algorithmes diffèrent par les graphes Laplaciens utilisés.
- L'algorithme de Ng, Weiss et Jordan nécessite une normalisation supplémentaire des lignes de la matrice U.
- Les algorithmes de classification spectrale permettent de minimiser des critères de coupure de graphe.

Nombre de classes

Approche suggérée

Rechercher le saut le plus significatif dans les valeurs propres ; en supposant que les valeurs propres de la matrice Laplacienne sont ordonnées par ordre croissant, cela revient à rechercher K tel que $\lambda_1, \ldots, \lambda_K$ sont relativement petites par rapport à λ_{K+1}.

Exemple illustratif



Matrice de similarité

$$W = \begin{pmatrix} 1 & 0.8 & 0.6 & 0 & 0.1 & 0 \\ 0.8 & 1 & 0.8 & 0 & 0 & 0 \\ 0.6 & 0.8 & 1 & 0.2 & 0 & 0 \\ 0 & 0 & 0.2 & 1 & 0.8 & 0.7 \\ 0.1 & 0 & 0 & 0.8 & 1 & 0.8 \\ 0 & 0 & 0 & 0.7 & 0.8 & 1 \end{pmatrix}$$

Exemple illustratif

Matrice de similarités : W

$$\begin{pmatrix} 1 & 0.8 & 0.6 & 0 & 0.1 & 0 \\ 0.8 & 1 & 0.8 & 0 & 0 & 0 \\ 0.6 & 0.8 & 1 & 0.2 & 0 & 0 \\ 0 & 0 & 0.2 & 1 & 0.8 & 0.7 \\ 0.1 & 0 & 0 & 0.8 & 1 & 0.8 \\ 0 & 0 & 0 & 0.7 & 0.8 & 1 \end{pmatrix}$$

Matrice des degrés : D

$$\begin{pmatrix} 2.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2.6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2.6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2.7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2.7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2.5 \end{pmatrix}$$

Matrice Laplacienne : $L = D - W$

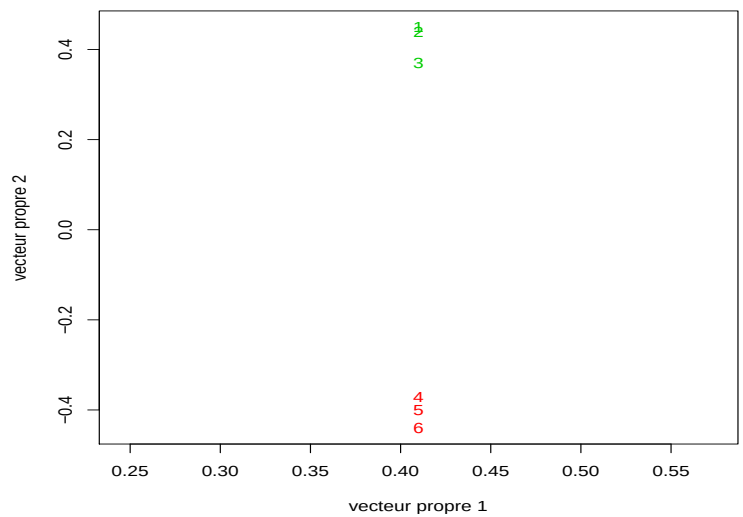
$$\begin{pmatrix} 1.5 & -0.8 & -0.6 & 0 & -0.1 & 0 \\ -0.8 & 1.6 & -0.8 & 0 & 0 & 0 \\ -0.6 & -0.8 & 1.6 & -0.2 & 0 & 0 \\ 0 & 0 & -0.2 & 1.7 & -0.8 & -0.7 \\ -0.1 & 0 & 0 & -0.8 & 1.7 & -0.8 \\ 0 & 0 & 0 & -0.7 & -0.8 & 1.5 \end{pmatrix}$$

Exemple illustratif

Valeurs propres					
0.0	0.2	2.1	2.3	2.5	2.6

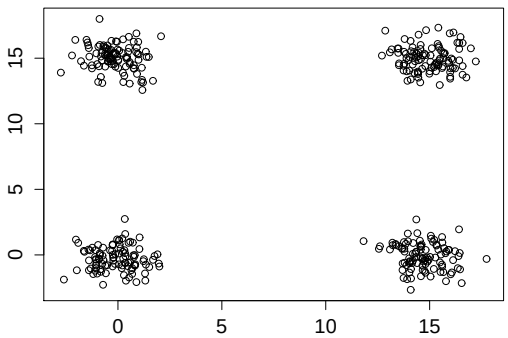
Vecteurs propres					
0.41	0.45	0.64	-0.30	0.37	-0.10
0.41	0.44	-0.01	0.30	-0.70	-0.21
0.41	0.37	-0.63	0.04	0.38	0.36
0.41	-0.37	-0.33	-0.45	0.00	-0.61
0.41	-0.40	0.16	-0.30	-0.35	0.65
0.41	-0.44	0.17	0.71	0.28	-0.08

kmeans dans l'espace propre

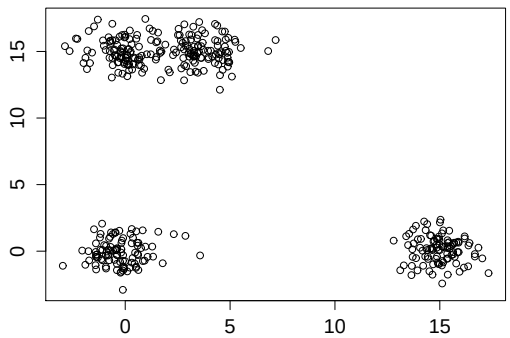


Autres illustrations - Choix du nombre de classes

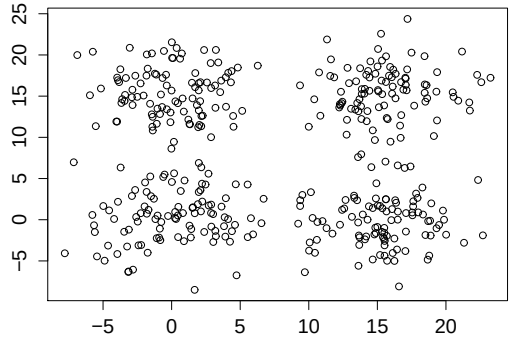
Données



Données

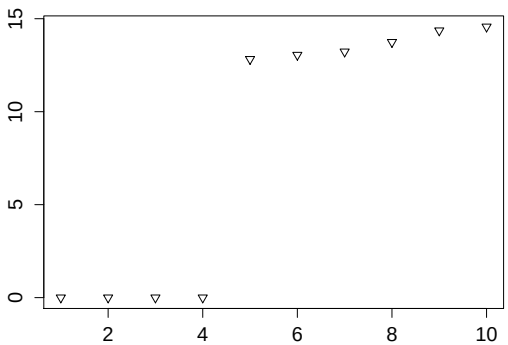


Données

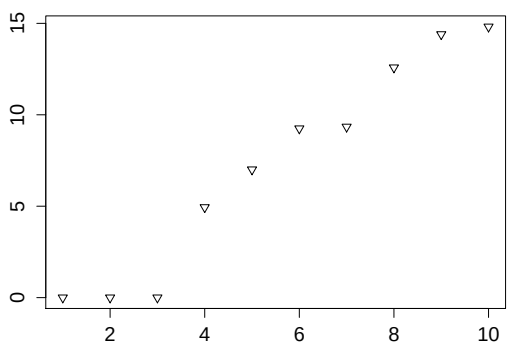


Utilisation de la similarité gaussienne avec $\sigma = 2$

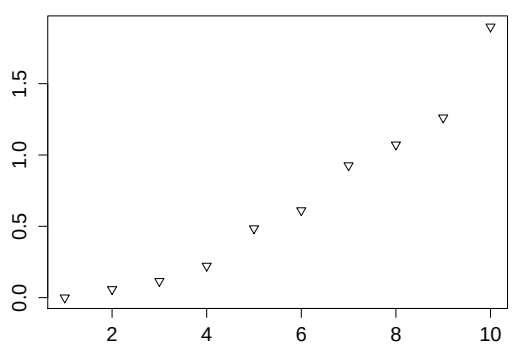
Valeurs propres de L



Valeurs propres de L



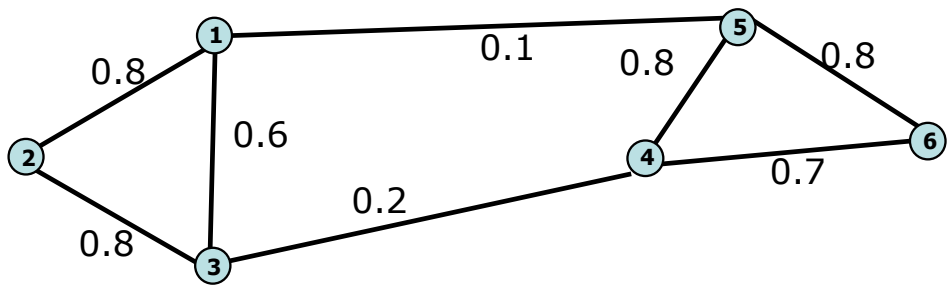
Valeurs propres de L



Critères de coupure de graphe

Idée générale

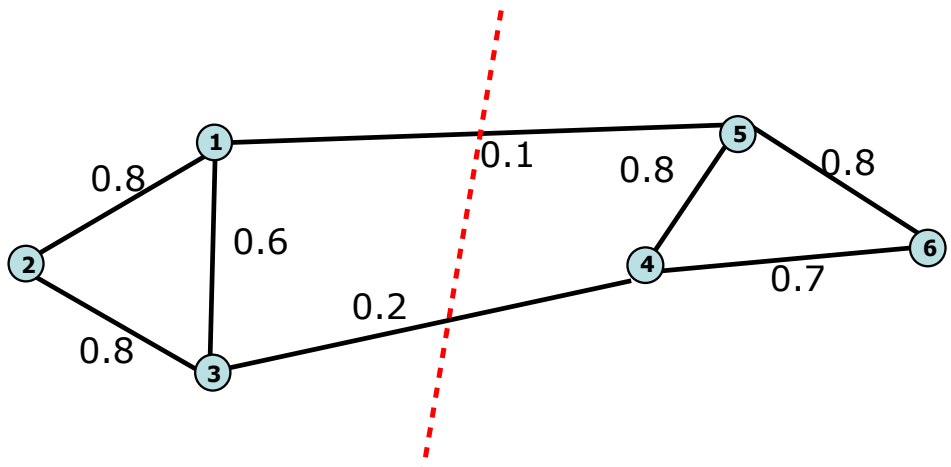
Rechercher une partition telle que les arêtes entre différentes classes aient les poids les plus faibles et que les arêtes au sein d'une même classe aient les poids les plus élevés.



Critères de coupure de graphe

Idée générale

Rechercher une partition telle que les arêtes entre différentes classes aient les poids les plus faibles et que les arêtes au sein d'une même classe aient les poids les plus élevés.



Critères de coupure d'un graphe

Coupure en deux ensembles disjoints

$$\text{cut}(A, B) = \sum_{\mathbf{x}_i \in A} \sum_{\mathbf{x}_j \in B} w_{ij}$$

Coupure associée à une partition en K classes

Recherche de la partition P qui minimise

$$\text{cut}(P) = \sum_{\ell=1}^{K} \text{cut}(P_\ell, \overline{P_\ell}) \quad \text{avec} \quad \overline{P_\ell} = \Omega \backslash P_\ell$$

Remarque

- Ce critère de coupure peut conduire à des partitions déséquilibrées (quelques classes formées de singletons)
- Comment imposer aux classes des tailles minimales ?

Critères de coupure normalisés

Ratiocut

$$\mathrm{RatioCut}(P) = \sum_{\ell=1}^{K} \frac{\mathrm{cut}(P_\ell, \overline{P_\ell})}{|P_\ell|}$$

avec $|P_\ell|$ = nombre d'éléments de la classe P_ℓ

Ncut

$$\mathrm{Ncut}(P) = \sum_{\ell=1}^{K} \frac{\mathrm{cut}(P_\ell, \overline{P_\ell})}{\mathrm{Vol}(P_\ell)}$$

avec $\mathrm{Vol}(P_\ell) = \sum_{\mathbf{x}_i \in P_\ell} d_i$

La minimisation de ces deux critères permet d'obtenir des partitions équilibrées mais elle conduit à des problèmes NP difficiles

Minimisation de RatioCut(P)

Notons $H_P = (h_{ik})$ la matrice $n \times K$ définie par :

$$h_{ik} = \begin{cases} \frac{1}{\sqrt{|P_k|}} & \text{si } \mathbf{x}_i \in P_k \\ 0 & \text{sinon} \end{cases}$$

On peut remarquer que

$$\begin{aligned} \mathrm{RatioCut}(P) &= \mathrm{Trace}(H'_P \, L \, H_P) \\ H'_P \, H_P &= I \end{aligned}$$

Finalement :

$$\min_P \mathrm{RatioCut}(P) \iff \min_P \mathrm{Trace}(H'_P \, L \, H_P)$$

Minimisation de RatioCut(P)

- En approximant l'hypothèse d'appartenance (discrète) aux classes par une appartenance « continue » aux classes, le problème

$$\min_{P} \mathrm{Trace}(H'_P \, L \, H_P)$$

se ramène au problème (relaxé)

$$\min_{F \in \mathbb{R}^{n \times K}} \mathrm{Trace}(F' \, L \, F) \text{ sous la contrainte } F'F = I$$

dont la matrice-solution est constituée (en colones) des vecteurs propres associés aux K plus petites valeurs propres de L.

- Le partitionnement final des données est obtenu en lançant l'algorithme des k-means sur les lignes de la matrice F.
- En conclusion, l'algorithme de classification spectrale (non normalisé) revient à minimiser $\mathrm{RatioCut}(P)$.

Minimisation de NCut(P)

Notons $H_P = (h_{ik})$ la matrice $n \times K$ définie par :

$$h_{ik} = \begin{cases} \frac{1}{\sqrt{\mathrm{Vol}(P_k)}} & \text{si } \mathbf{x}_i \in P_k \\ 0 & \text{sinon} \end{cases}$$

On peut remarquer que

$$\begin{aligned} \mathrm{Ncut}(P) &= \mathrm{Trace}(H'_P \, L \, H_P) \\ (D^{1/2} H_P)' \, (D^{1/2} H_P) &= I \end{aligned}$$

Finalement :

$$\begin{aligned} \min_P \mathrm{Ncut}(P) &\iff \min_P \mathrm{Trace}(H'_P \, L \, H_P) \\ &\iff \min_P \mathrm{Trace}\Big((D^{1/2} H_P)' \, L_{sym} \, (D^{1/2} H_P) \Big) \end{aligned}$$

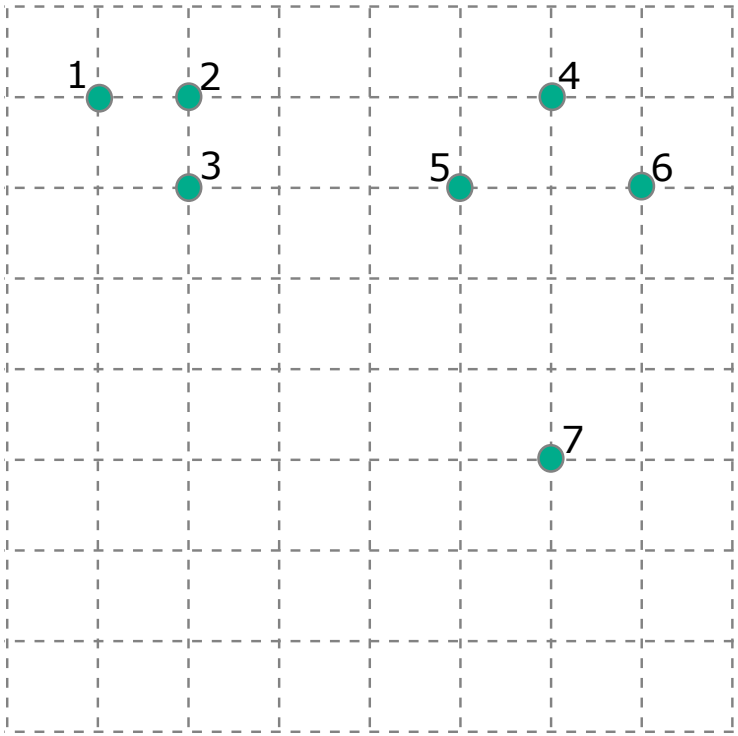
Minimisation de NCut(P)

- En posant $U = D^{1/2}\,F$, le problème (relaxé) s'écrit

$$\min_{U \in \mathbb{R}^{n \times K}} \mathrm{Trace}(U'\,L_{sym}\,U) \text{ sous } U'\,U = I$$

  dont la matrice-solution U est constituée (en colones) des vecteurs propres associés aux K plus petites valeurs propres de L_{sym}.
- La matrice F est obtenue par resubstitution : $F = D^{-1/2}U$.
- D'après les propriétés des matrices Laplaciennes normalisées, la matrice F est formée des vecteurs propres de L_{rw}.
- Le partitionnement final des données est obtenu en lançant l'algorithme des k-means sur les lignes de la matrice F.
- L'algorithme de classification spectrale de Shi et Malik (L_{rw}) conduit donc à la minimisation de $\mathrm{Ncut}(P)$.

Exercice



$$w_{ij} = \begin{cases} 1 & \text{si } d(x_i, x_j) \leq 4 \\ 0 & \text{sinon} \end{cases}$$

$$P = \{\{x_1, x_2, x_3\}; \{x_4, x_5, x_6, x_7\}\}$$

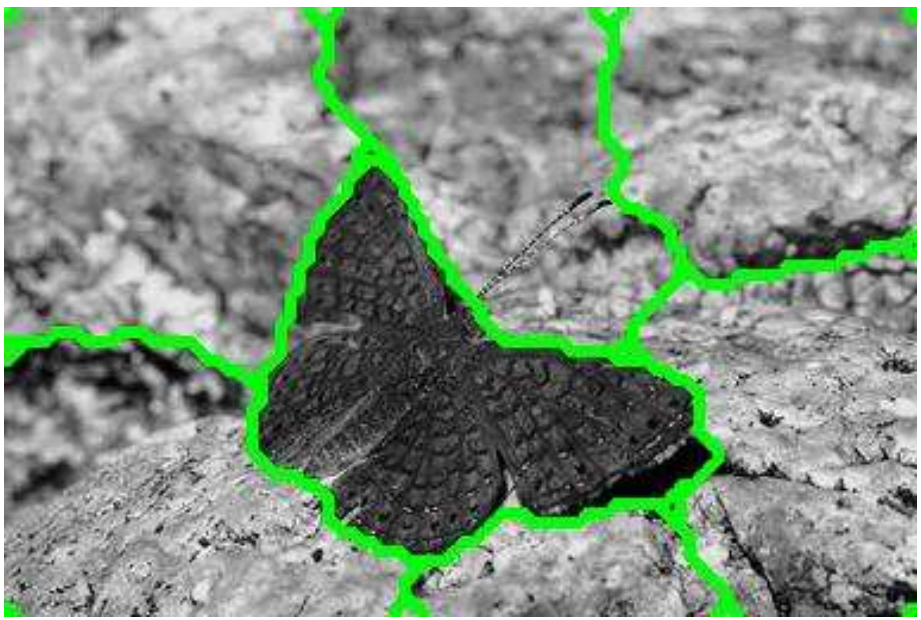
$$Q = \{\{x_1, x_2, x_3, x_4, x_5, x_6\}; \{x_7\}\}$$

1. Comparer les partitions P et Q en utilisant les critères Cut et RatioCut.
2. Sans effectuer de calculs, indiquer laquelle de ces partitions devrait être obtenue par l'algorithme de classification spectrale non normalisé.

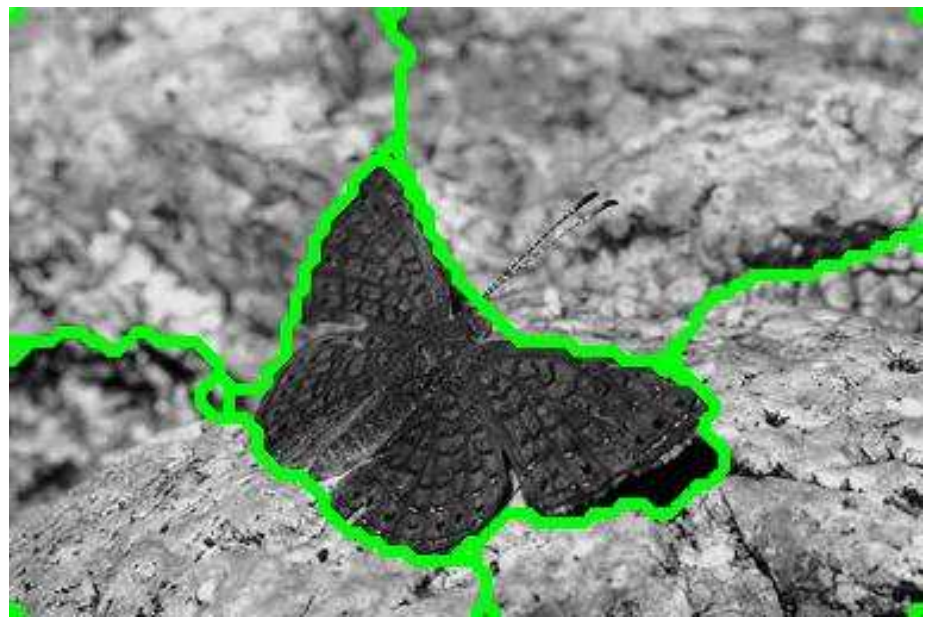
Application à la segmentation d'images

Ref : L. Zelnik-Manor and P. Perona, NIPS 2004

spectral clustering



dérivé de k-means



Application à la segmentation d'images

Ref : L. Zelnik-Manor and P. Perona, NIPS 2004

spectral clustering

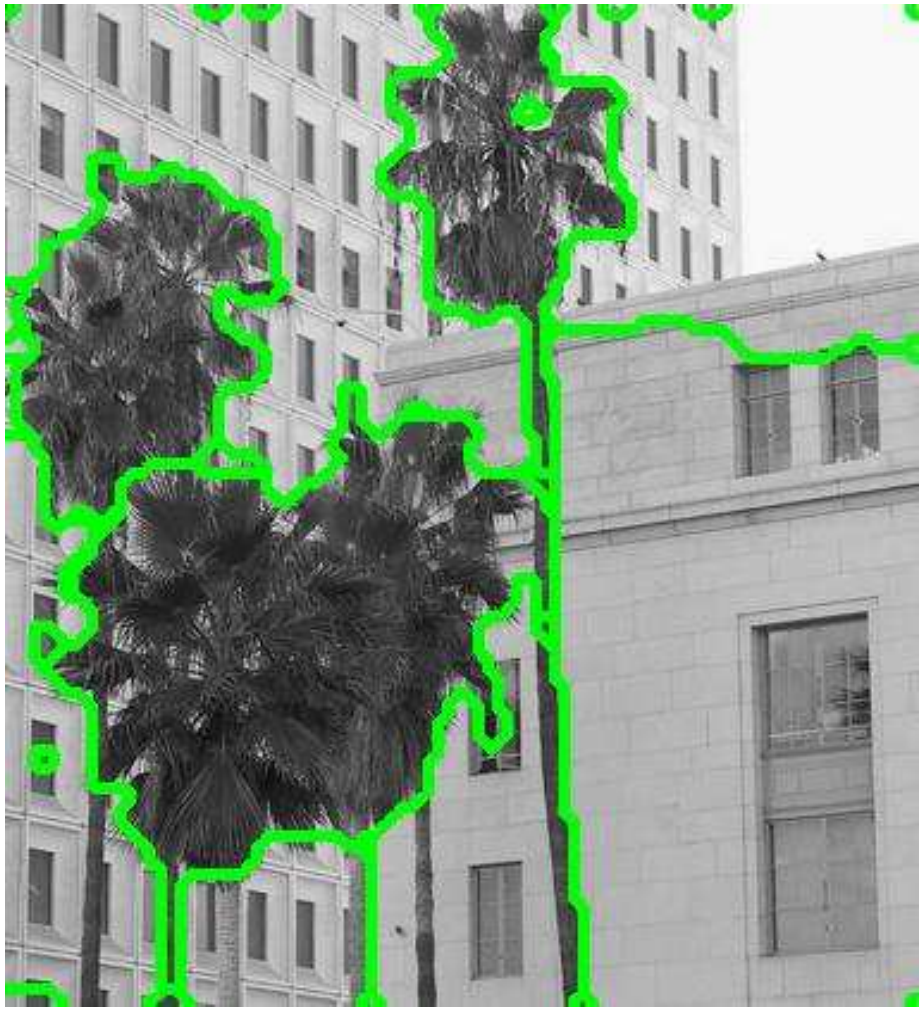
dérivé de k-means

Application à la segmentation d'images

Ref : L. Zelnik-Manor and P. Perona, NIPS 2004

spectral clustering

dérivé de k-means



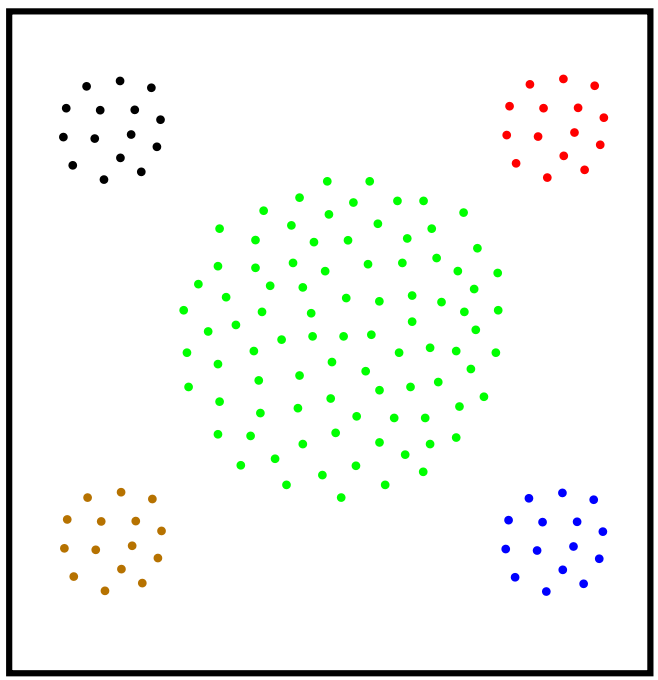
Classification spectrale dans R

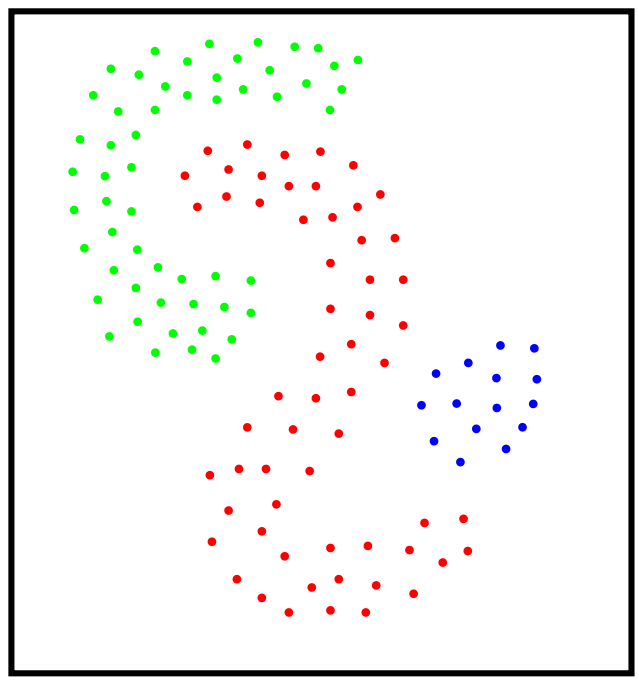
```
library(kernlab)
data(spirals)
sc <- specc(spirals,kernel = "rbfdot",centers=2)
plot(spirals, col=sc)
```

Algorithme DBSCAN

Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

- Méthode de regroupement automatique de données fondée sur la notion de voisinage local ou de « densité locale »
- Identification de classes ayant des formes complexes, que l'on peut rencontrer dans les bases de données spatiales





Notions utilisées dans l'algorithme DBSCAN

- ε-**voisinage** d'un point x : ensemble de points situés à une distance inférieure ou égale à ε de x ; on note cet ensemble $N_\varepsilon(x)$
- **Point intérieur ou cœur** : un point x est qualifié de point intérieur si son $N_\varepsilon(x)$ contient au moins un nombre minimal de point (**MinPts**).
- **Accessibilité par densité** : un point x est dit accessible par densité à partir d'un point y, s'il existe une suite de points $x_1, \ldots, x_n$ telle que $x_1 = x$, $x_n = y$, avec x_{i+1} appartenant à l'ε-voisinage de x_i, ce dernier étant un point intérieur.
- **Connexion par densité** : deux points x et y sont dits connectés par densité, s'il existe un point z tel que x et y sont accessibles par densité à partir de z.

Notions utilisées dans l'algorithme DBSCAN

Classe

Sous-ensemble maximal de points connectés par densité

- Si $\boldsymbol{x}_i$ est accessible par densité depuis un point d'une classe P_k, alors $\boldsymbol{x}_i$ appartient nécessairement à la classe P_k
- Dans une classe toute paire de points est connectée par densité

Point-frontière et bruit

- **Point-frontière** : point non intérieur situé dans le ε-voisinage d'un point intérieur
- **Bruit** : donnée non affectée aux classes

Illustration des notions

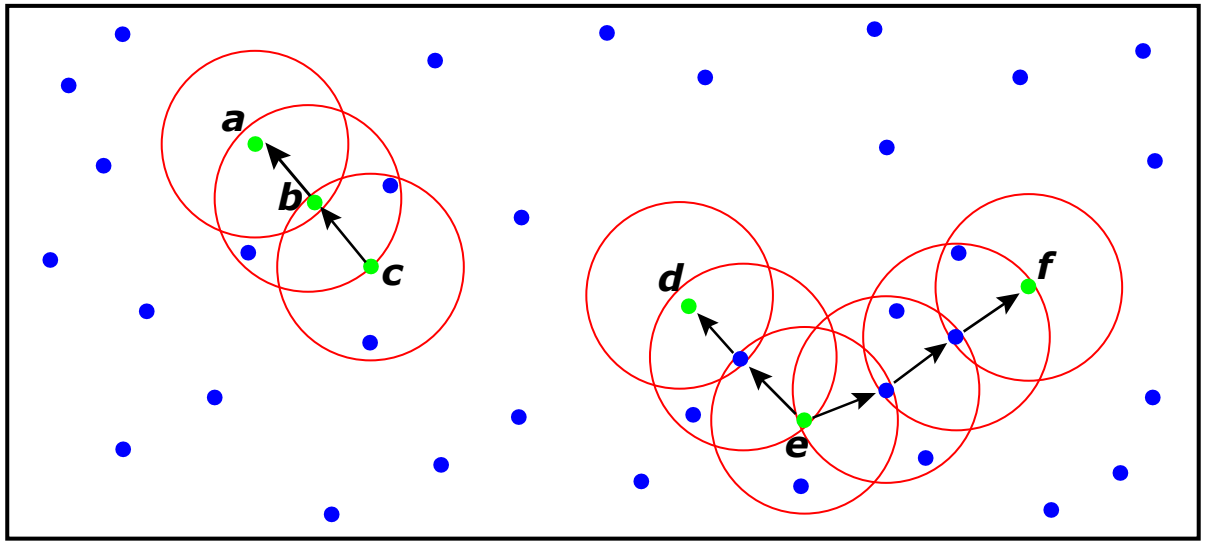


Illustration des notions avec MinPts=3

- b et c sont des points intérieurs dans la mesure où leur ε-voisinage représenté par le cercle contient au moins 3 points, mais le point a n'est pas un point intérieur
- a est accessible par densité à partir du point c mais l'inverse n'est pas vrai
- f et d sont connectés par densité puisqu'ils sont tous les deux accessibles par densité à partir du point e

Algorithme DBSCAN

Entrées : Hyper-paramètres ε et $MinPts$

répéter

- Tirage d'un point x qui n'a pas encore été visité
- Marquer x comme point visité
- Créer le ε-voisinage $\mathcal{V}_\varepsilon(x)$ du point x
- **si** $|\mathcal{V}_\varepsilon(x)| < MinPts$ **alors**
  - Labeliser x comme bruit
- **sinon**
  - Poser $C = \{x\}$
  - AgrandirCluster(C, $\mathcal{V}_\varepsilon(x)$, ε, $MinPts$)
- **fin**

jusqu'à *ce que tous les points aient été visités*;

Sorties : Ensemble des classes C créées et bruit

Algorithme DBSCAN

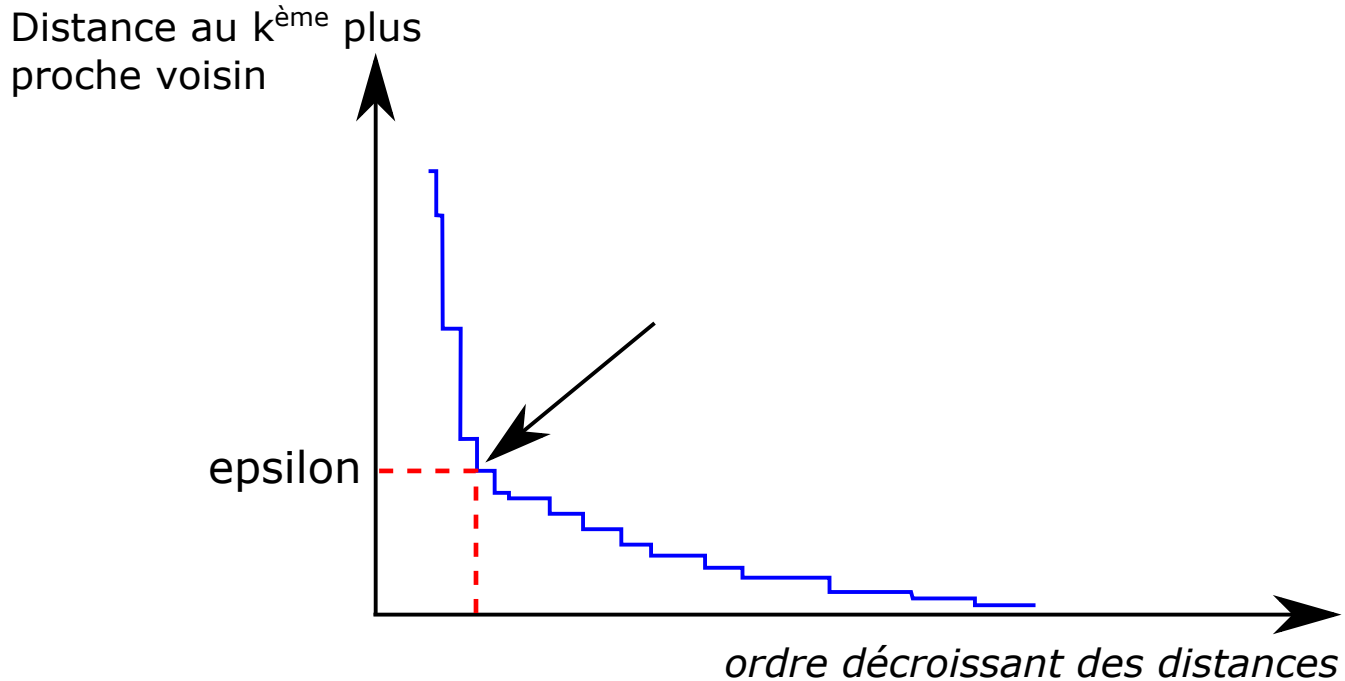
AgrandirCluster

Entrées : C, $\mathcal{V}$, ε, $MinPts$
pour $y \in \mathcal{V}$ **faire**
  si y *n'a pas encore été visité* **alors**
    $\mathcal{V}' = \mathcal{V}_\varepsilon(y)$
    si $|\mathcal{V}'| \geq MinPts$ **alors**
      $\mathcal{V} = \mathcal{V} \bigcup \mathcal{V}'$
  si y *n'appartient à aucune autre classe* **alors**
    Poser $C = C \bigcup \{y\}$
fin
Sorties : C

Remarque : au cours des itérations de l'algorithme, un point initialement considéré comme du bruit peut ensuite être affecté à une classe s'il se trouve dans le voisinage d'un point intérieur.

Algorithme DBSCAN : choix de ε et de *MinPts*

- Choix de MinPts
  - La règle généralement adoptée consiste à prendre MinPts $\geq p + 1$, où p est la dimension des données
- Choix de ε
  - Calculer la distance entre chaque individu $\boldsymbol{x}_i$ et son $k$ème plus proche voisin, avec $k =$ MinPts
  - Tracer la courbe de ces distances réordonnées par ordre décroissant et rechercher le coude de celle-ci



Points forts et points faibles de DBSCAN

Avantages

- Permet l'extraction de classes de formes complexes (non linéairement séparables)
- Ne nécessite pas de connaître à l'avance le nombre de classes
- Rapide à mettre en œuvre (plusieurs versions accélérées sont proposées)

Inconvénients

- Le choix de ε et de MinPts peut être souvent délicat
- N'est pas adapté au cas où les classes ont des densités différentes
- Difficultés à utiliser DBSCAN pour des données de grande dimension (les ε-voisinages ont tendance dans ce cas à ne contenir que très peu de données - « fléau de la dimensionalité »)

DBSCAN sous le logiciel R

```
library(dbscan)
#library(fpc)

# Jeu de donnees
library(factoextra)
data("multishapes")
x = multishapes[,1:2]

# Choix epsilon
D = kNNdist(x,k=3,search="kd")
plot(sort(D,decreasing=TRUE),type="l",xlim=c(0,1000))
abline(h=0.15,col="red",lty=2)

# Lancer algorithme
dbscan.res = dbscan(x,0.15,minPts=3)
dbscan.res
plot(x,col=dbscan.res$cluster)
```