# Fouille de données - Classification non supervisée

G. Forestier

Fouille de données

C. Wemmert, S. Lebre

## Introduction

- But de la classification automatique : obtenir une représentation simplifiée (structuration) des données initiales
- Organisation d'un ensemble d'objets en un ensemble de regroupements homogènes et/ou naturels
- Deux approches:
  1. classification/clustering/regroupement : découverte en extension de ces ensembles $\rightarrow$ notion de classes ou clusters
  2. formation de concepts : découverte en intention de ces ensembles $\rightarrow$ notion de concepts

## Clustering

Différentes méthodes :

- clustering par partitionnement
- clustering par modélisation
- clustering hiérarchique

Et bien d'autres :

- clustering basé sur la densité
- clustering par "grille"

## Clustering

Différentes méthodes :

- clustering par partitionnement
- clustering par modélisation
- clustering hiérarchique

Et bien d'autres :

- clustering basé sur la densité
- clustering par "grille"

## Partitionnement

Clustering par partitionnement :

- Placer les différents objets dans des clusters (groupes)
- **Hard clustering** : chaque objet est dans une et une seule classe
- **Soft clustering** : un objet peut être dans plusieurs classes
- **Fuzzy clustering** : un objet appartient à toutes les classes mais à des degrés différents

## Partitionnement

- Matrice de classification de $n$ éléments en $K$ classes

$$c_D = \begin{pmatrix} c_{1,1} & \cdots & c_{1,K} \\ & \cdots & \\ c_{n,1} & \cdots & c_{n,K} \end{pmatrix}$$

- Partitionnement dur (hard) :

$$\nexists k, c_{i,k} = 1, c_{i,j} = 1 \quad \text{pour} \quad j \neq k$$

- Partitionnement souple (soft) :

$$\exists k_1, k_t \ldots \text{t.q.}, c_{i,k_i}, c_{i,j} = 1 \quad \text{pour} \quad j \neq k_i$$

## Partitionnement

Partitionnement flou :

- Le nombre d'objets d'une classe est inférieur au nombre total d'objets : $\forall k = 1 \dots K, \forall x_i \in D, c_{i,k} \in [0,1]$
  avec $\forall k = 1 \dots K, 0 < \sum_{i=1}^{n} c_{i,k} < n$
- Un objet appartient à différents degrés, à toutes les classes :
  $\forall x_i \in D, \sum_{k=1}^{K} c_{i,k} = 1$

## Partitionnement

Clustering par partitionnement :

- Trouver l'organisation en classes homogènes telles que deux objets d'une même classe sont plus similaires que deux objets de classes différentes
- Trouver l'organisation en classes homogènes telles que les classes soient les plus différentes possibles
- Comment évaluer ces (dis)similarités ?

## Dissimilarité et distance

- On appelle dissimilarité toute application définie par $d : M^2 \rightarrow R^+$ telle que :

$$\forall (x, y) \in M^2, d(x, y) = 0 \leftrightarrow x = y$$

- Elle est symétrique si :

$$\forall (x, y) \in M^2, d(x, y) = d(y, x)$$

- On appelle distance, tout indice de dissimilarité symétrique qui possède en plus l'inégalité triangulaire :

$$\forall (x, y, z) \in M^3, d(x, z) \leq d(x, y) + d(y, z)$$

## Distance

Exemple :

- La donnée est un vecteur de $R^p$

$$d_q = \sqrt[q]{|x_{i,1} - x_{j,1}|^q + |x_{i,2} - x_{j,2}|^q + \ldots + |x_{i,p} - x_{j,p}|^q}$$

## Partitionnement – mesure point/point

- La donnée est un vecteur de $[V, F]^p$
- Définition d'un tableau de contingence avec $p$ attributs binaires et deux objets $i_1$ et $i_2$ :

| i1 \ i2 | Vrai | Faux | Somme |
|---|---|---|---|
| Vrai | N1 | N2 | N1+N2 |
| Faux | N3 | N4 | N3+N4 |
| Somme | N1+N3 | N2+N4 | p |

- $N1$ = nbre de fois où un attribut est vrai pour les deux objets

## Partitionnement – mesure point/point

- La donnée est un vecteur de $[V, F]^p$
- Mesure symétrique :

$$d_{(i,j)} = \frac{N2 + N3}{N1 + N2 + N3 + N4}$$

- Mesure asymétrique :

$$d_{(i,j)} = \frac{N2 + N3}{N1 + N2 + N3}$$

# Partitionnement – mesure point/point

Exemple :

| Région | Allongée | Texturée | Grande | Isolée | Fréquente |
|---|---|---|---|---|---|
| Ville | NON | OUI | OUI | OUI | NON |
| Quartier | NON | OUI | NON | NON | OUI |
| Lac | NON | NON | OUI | OUI | NON |
| Champs | OUI | NON | NON | NON | OUI |

# Partitionnement – mesure point/point

Exemple : ville / quartier

| Région | Allongée | Texturée | Grande | Isolée | Fréquente |
|---|---|---|---|---|---|
| Ville | NON | OUI | OUI | OUI | NON |
| Quartier | NON | OUI | NON | NON | OUI |

| | OUI | NON |
|---|---|---|
| OUI | 1 | 2 |
| NON | 1 | 1 |

- mesure symétrique : $3/5 \rightarrow S(\text{ville, quartier}) = \mathbf{0.4}$
- mesure asymétrique : $3/4$

# Partitionnement – mesure point/point

Pour les autres couples :

| Région | Allongée | Texturée | Grande | Isolée | Fréquente |
|---|---|---|---|---|---|
| Ville | NON | OUI | OUI | OUI | NON |
| Quartier | NON | OUI | NON | NON | OUI |
| Lac | NON | NON | OUI | OUI | NON |
| Champs | OUI | NON | NON | NON | OUI |

- ville/lac : 1/5 et 1/3 S(ville, lac) = **0.8**
- ville/champs : 1 et 1 S(ville, champs) = **0**
- quartier/champs : 2/5 et 2/3
- quartier/lac : 4/5 et 1
- lac/champs : 4/5 et 1 S(lac, champs) = **0.2**

## Partitionnement – mesure point/point

La donnée est une variable nominale :

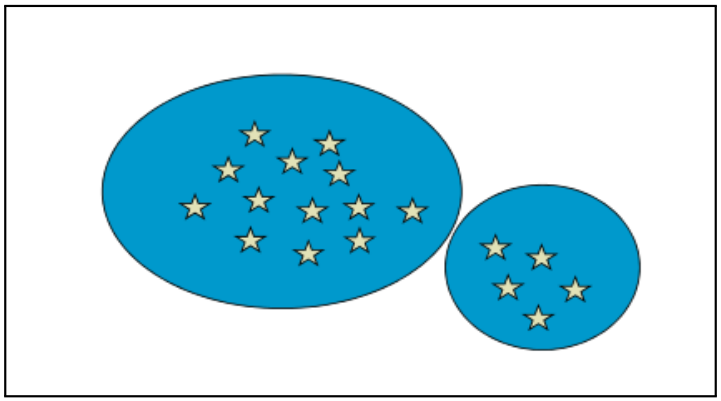
- prend des valeurs finies dans un ensemble pouvant être fini
- sans ordre naturel : couleur $\in$ {rouge, bleu, vert}
- avec : niveau $\in$ {passable, moyen, bon}
- où passable $<$ moyen $<$ bon

La donnée est complexe :

- histogramme
- structures hétérogènes
- images, videos, etc.

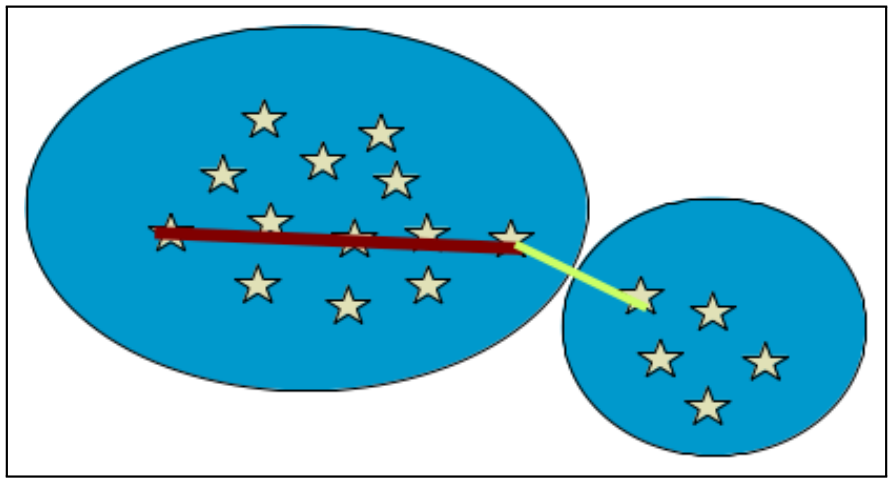
## Partitionnement

- **Objectif :** trouver l'organisation en classes homogènes telles que deux objets d'une même classe se ressemblent plus que deux objets de classes différentes



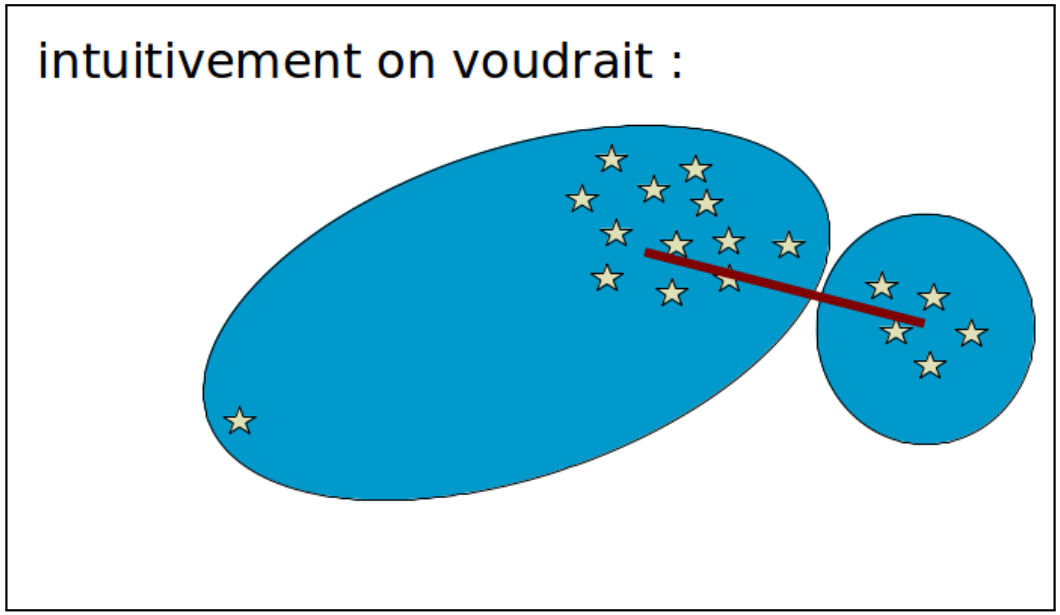
## Partitionnement

- d1 et d2 de même classe, d3 d'une classe différente alors dist(d1,d2) $<$ dist(d1,d3) et dist(d1,d2) $<$ dist(d2,d3)



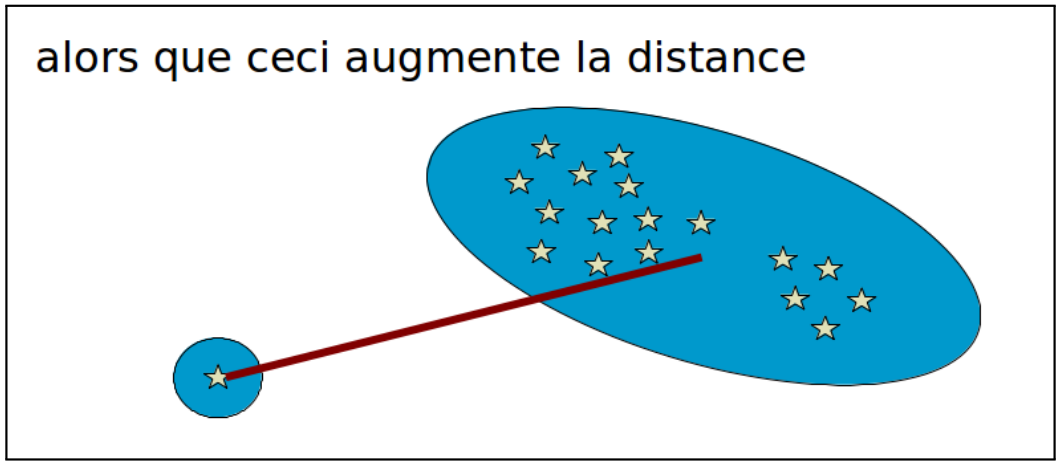
## Partitionnement : qualité

- **Objectif :** trouver l'organisation en classes homogènes telles que les distances entre les classes soient maximales



## Partitionnement : qualité

- **Objectif :** trouver l'organisation en classes homogènes telles que les distances entre les classes soient maximales



## Partitionnement : approche métrique

Critère possible :

- minimiser l'inertie **intra**-classe $I_{intra}$ (interne aux classes) :

$$I_{intra}(C_k) = \sum_{x_i \in C_k} p_i \times dist(x_i, g_k)^2$$

$$I_{intra} = \sum_{k=1 \ldots K} I_{intra}(C_k)$$

## Partitionnement : approche métrique

Critère possible :

- maximiser l'inertie **inter**-classe $I_{inter}$ (entre les classes) :

$$I_{inter} = \sum_{k=1...K} P_k \times dist(g, g_k)^2$$

## Partitionnement : approche métrique

Inertie totale du système $I_{totale}$

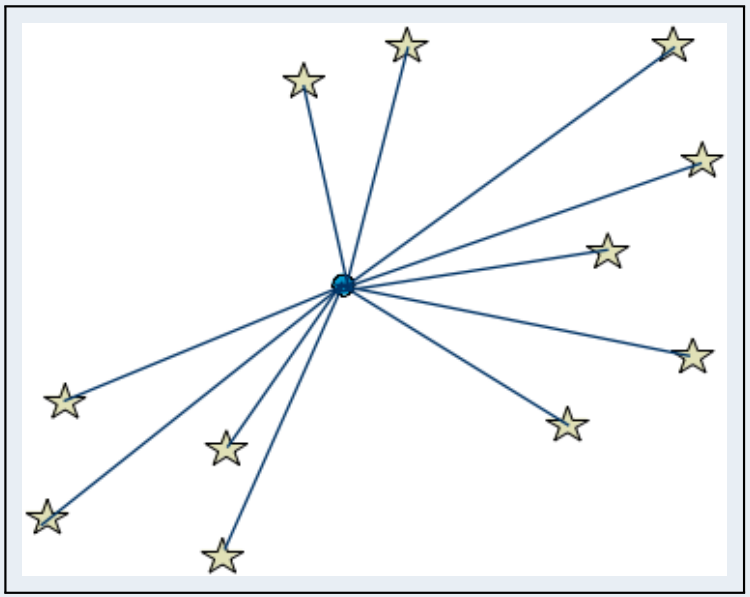
- $I_{totale} = I_{intra} + I_{inter}$

$$I_{totale} = \sum_{i=1\ldots n} p_i \times dist(x_i, g)^2$$

$g$ centre de gravité

## Partitionnement : approche métrique

- Exemple pour K = 3
- Inertie totale

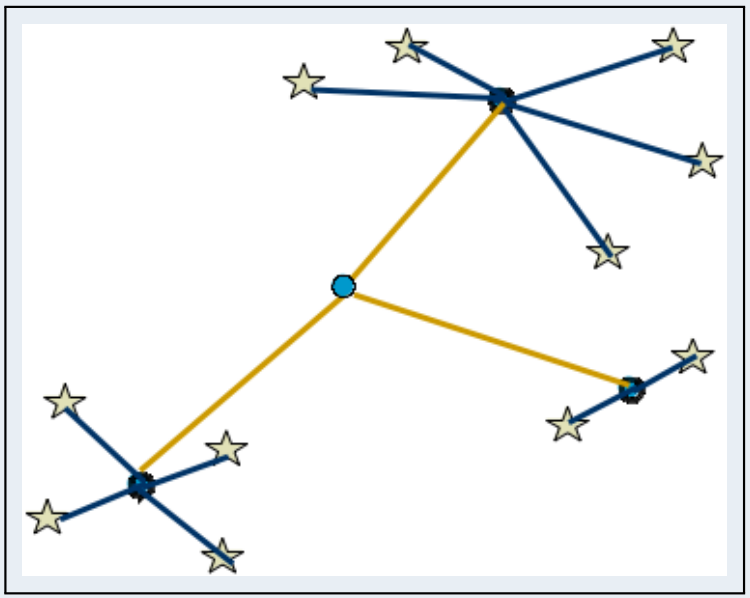


étape 1 :
calcul de $g$

étape 2 :
calcul et somme
des distances

## Partitionnement : approche métrique

- Inertie intra-classe et inter-classe :



étape 1:
calcul des $g_k$

étape 2 :
calcul de l'inertie inter-classe et intra-classe

## Partitionnement : approche métrique

- Le plus souvent, le critère de qualité retenu est la minimisation de l'inertie intra-classe (et donc la maximisation de l'inertie inter-classe)
- Le principe des algorithmes sera donc de trouver des partitions telles que cette inertie soit minimale

## Partitionnement : construction

- Comment construire ces classes ?
- Nombre de partitions de k classes à partir de $n$ données définissables sur D :

$$S(n,k) = \frac{1}{k!} \sum_{i=0\ldots k} (-1)^{i-1} C_i^k i^n$$

- tend vers $S(n,k) \approx \frac{k^n}{k!}$ pour $n$ grand
- $n = 8$ et $k = 4 \rightarrow 1701$
- $n = 12$ et $k = 3 \rightarrow 86526$

## Partitionnement : construction

Critère :

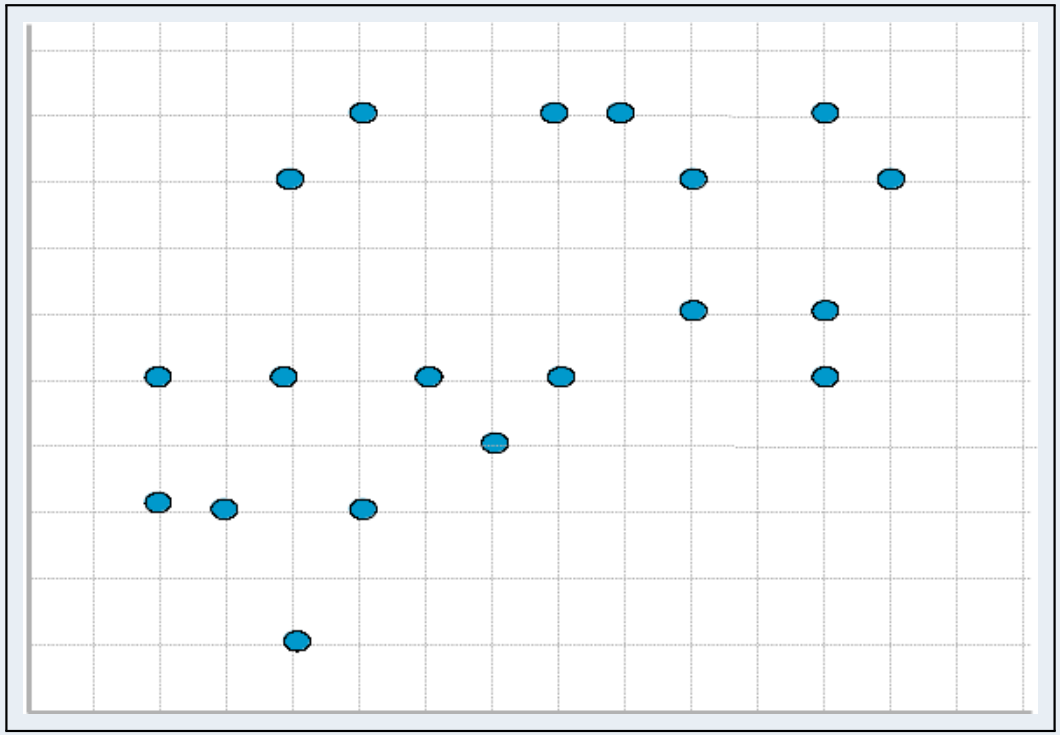
- minimiser l'inertie intra-classe (maximiser l'inertie inter-classes)

Processus des KMeans :

1. choix aléatoire du nombre $K$ de classes et de $K$ noyaux initiaux
2. (ré-)affectation des objets aux noyaux
3. calcul des nouveaux noyaux
4. le processus s'arrête lorsque deux partitions successives sont identiques (ou presque)

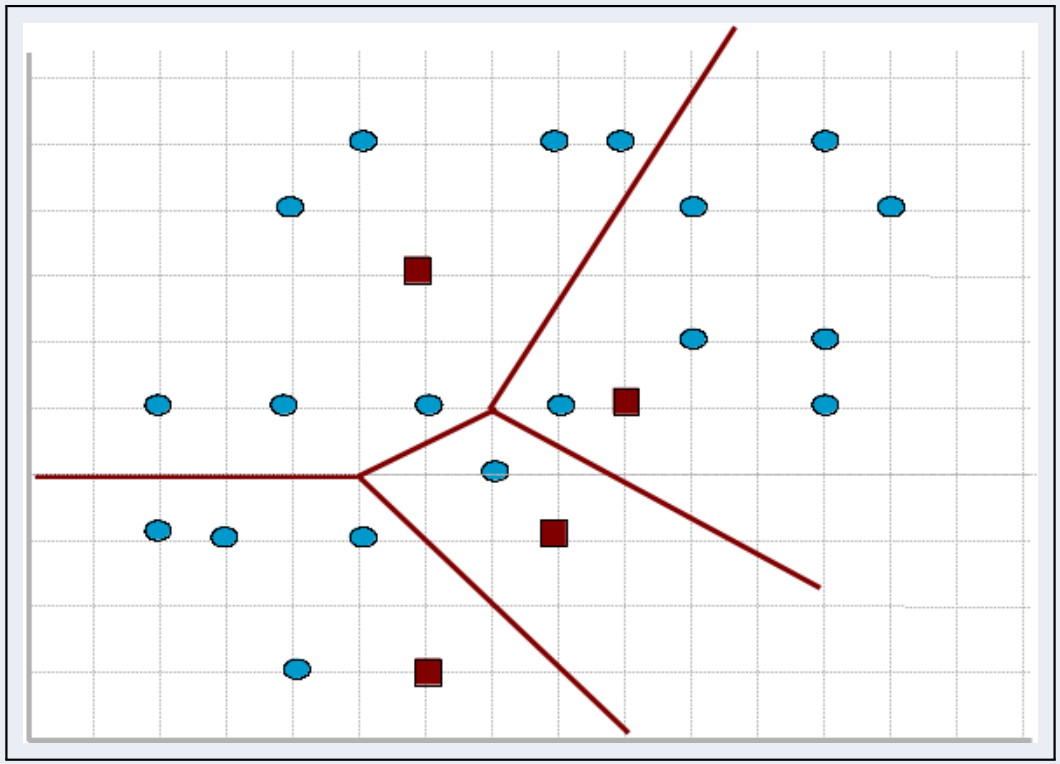
## Kmeans

Avec $K = 4$



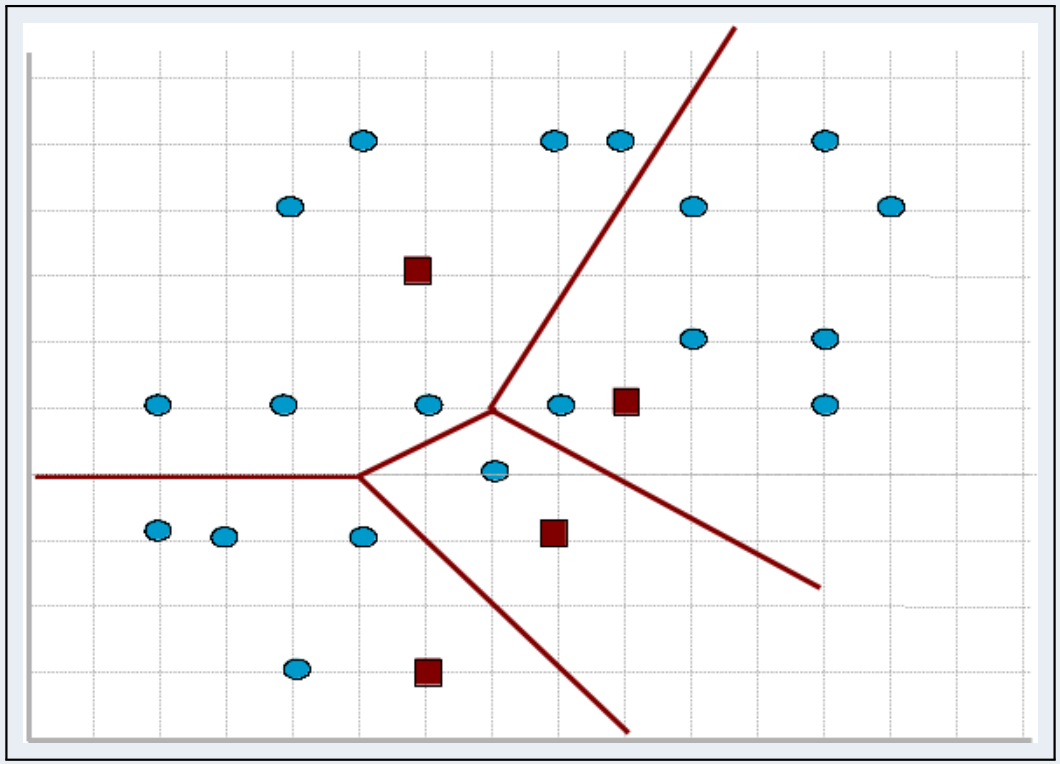
## Kmeans

$K = 4$ : initialisation aléatoire



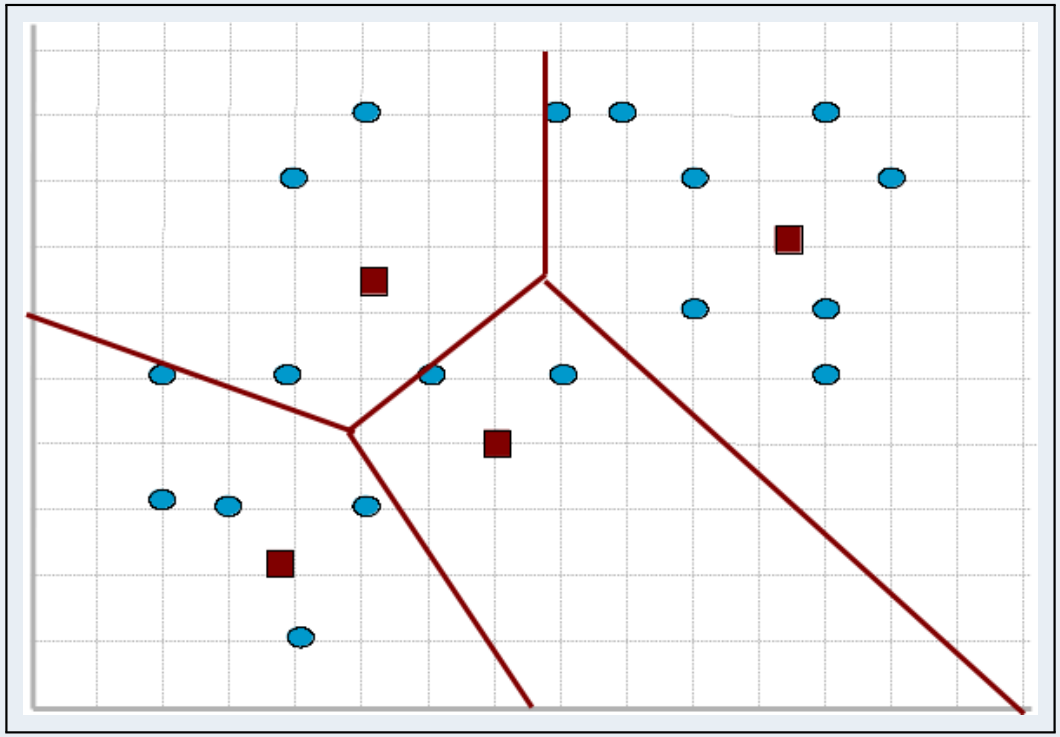
# Kmeans

## Allocation :



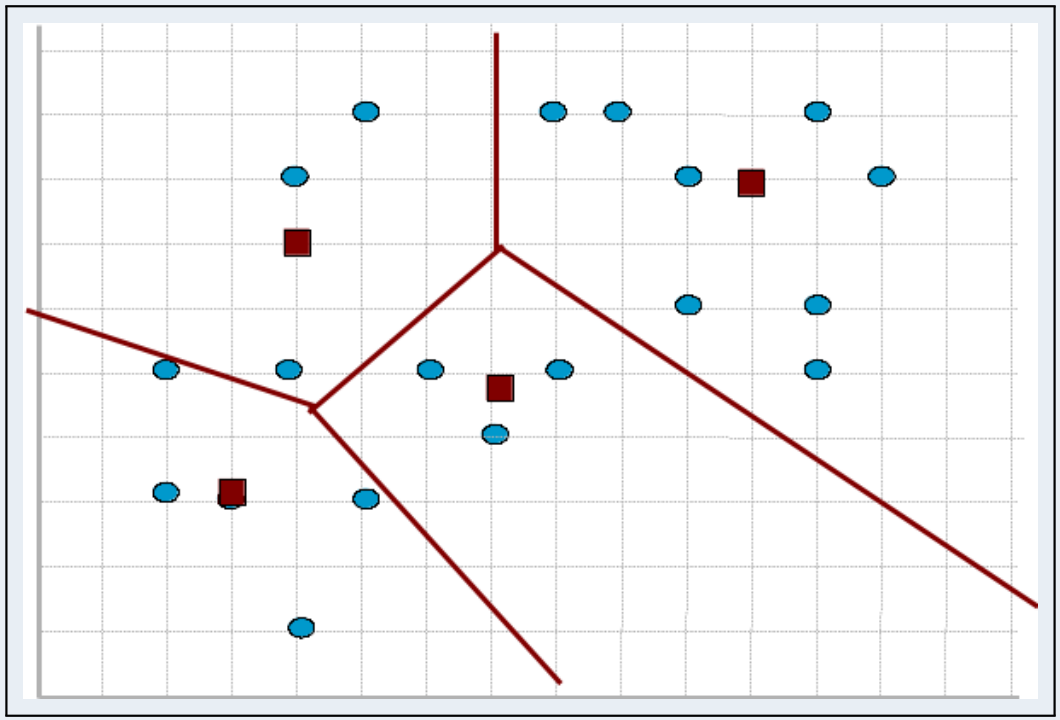
# Kmeans

Déplacement des centres :



## Kmeans

Déplacement des centres :

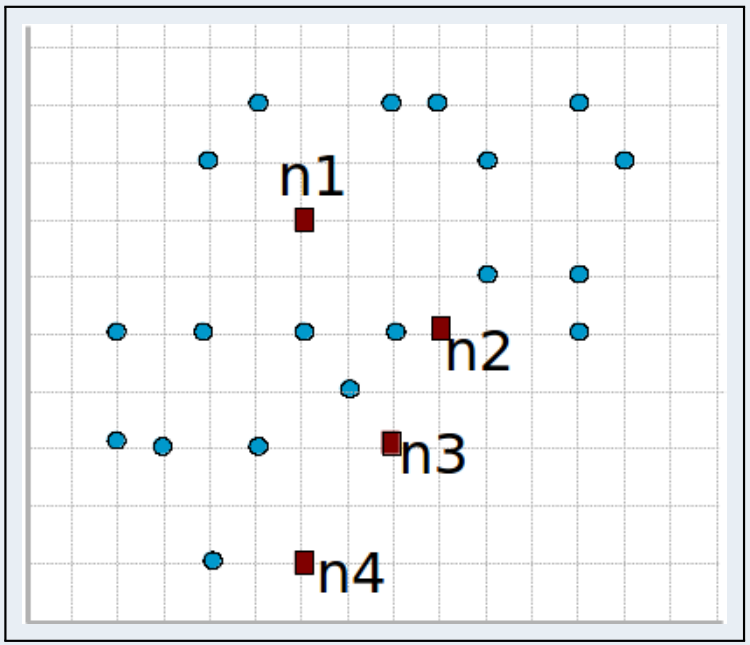


Arrêt.

## Kmeans (propriétés)

Soit à l'instant $t$ :

- $P_K(t) = \{C1, C2, ....Ck\}$ la partition
- $L_K(t) = \{g1, g2, ...., gk\}$ l'ensemble des centres de gravité

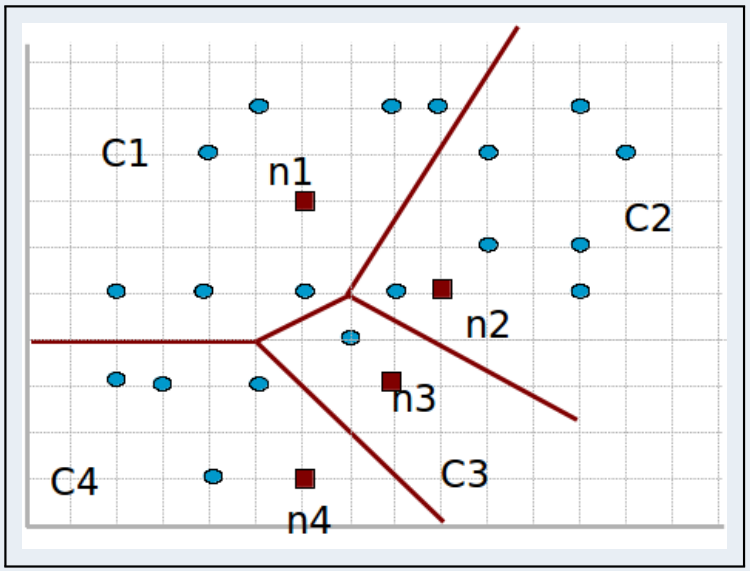



$$P4(0) = D,$$
$$L4(0) = \{n1, n2, n3, n4\}$$

# Kmeans (propriétés)

Soit à l'instant $t$ :

- $P_K(t) = \{C1, C2, ....Ck\}$ la partition
- $L_K(t) = \{g1, g2, ...., gk\}$ l'ensemble des centres de gravité

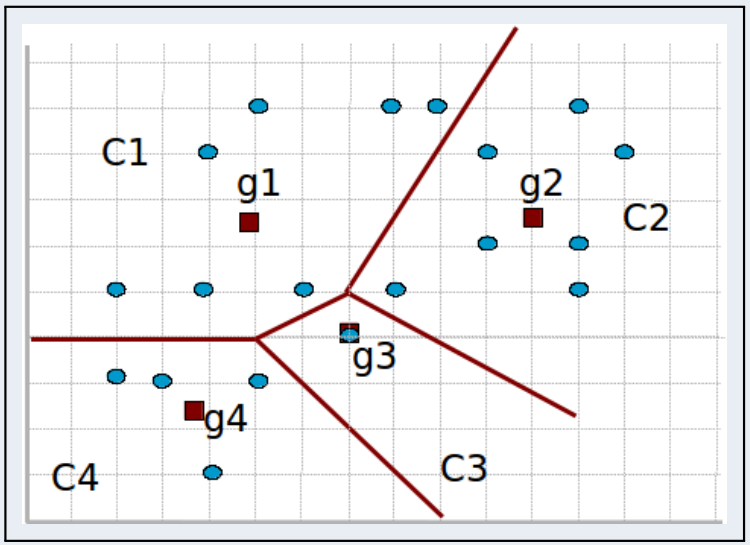



$$P4(1) = \{C1, C2, C3, C4\},$$
$$L4(0) = \{n1, n2, n3, n4\}$$

## Kmeans (propriétés)

Soit à l'instant $t$ :

- $P_K(t) = \{C1, C2, ....Ck\}$ la partition
- $L_K(t) = \{g1, g2, ...., gk\}$ l'ensemble des centres de gravité




$$P4(1) = \{C1, C2, C3, C4\},$$
$$L4(1) = \{g1, g2, g3, g4\}$$

## Kmeans (propriétés)

- Soit $Q(P_K(t), L_K(t))$ l'inertie totale interne des classes pour la partition $P_K(t) = \{C1, C2, ..., Ck\}$ et l'ensemble des centres de gravité $L_K(t) = \{g1, g2, ..., gk\}$

$$Q(P_K(t), L_K(t)) = \sum_{k=1}^{K} \sum_{x \in C_k} dist(x, g_k)^2$$

- Montrons qu'à chaque étape, ce critère s'améliore (donc qu'il décroît)

## Kmeans (propriétés)

- Montrons que $Q(P_K(t+1), L_K(t)) \leq Q(P_K(t), L_K(t))$ ?
- La réallocation des objets

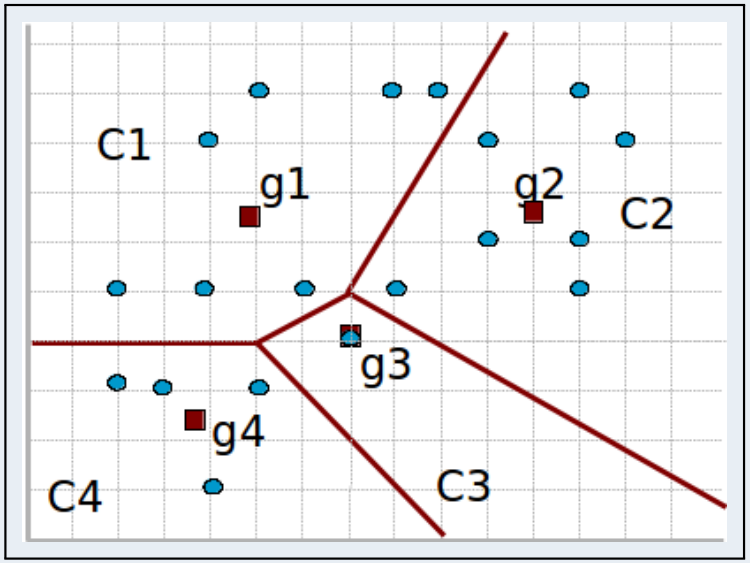
$$P_4(1) = \{C1, C2, C3, C4\}, L_4(1) = \{g1, g2, g3, g4\}$$

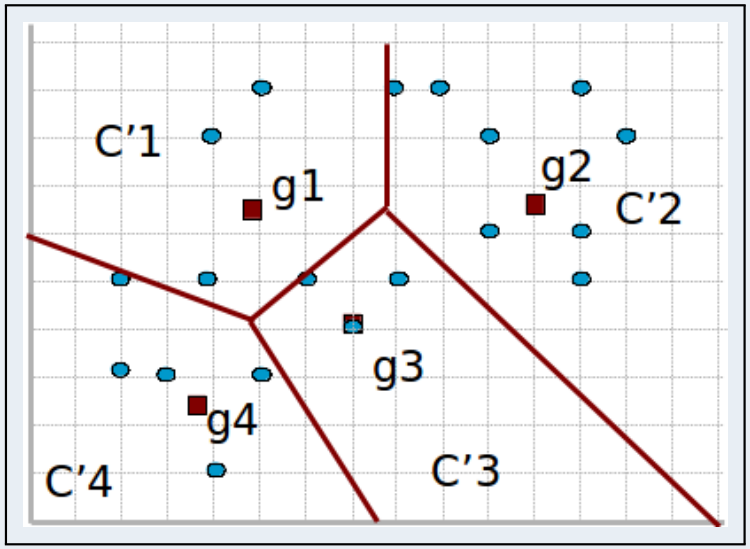
à

$$P_4(2) = \{C'1, C'2, C'3, C'4\}, L_4(1) = \{g1, g2, g3, g4\}$$

diminue-t-elle l'inertie ?

# Kmeans (propriétés)




$P_4(1) = \{C1, C2, C3, C4\}, L_4(1) = \{g1, g2, g3, g4\}$

$P_4(2) = \{C'1, C'2, C'3, C'4\}, L_4(1) = \{g1, g2, g3, g4\}$

## Kmeans (propriétés)

Lors de la construction de $P_K(t+1)$, pour une donnée $x$, de classe $k(x)$

- soit $x$ était déjà associé à cette classe : comme ni le centre de gravité, ni $x$ n'ont bougé, cette $dist(x, g_{k(x)})$ est la même que dans la partition précédente
- soit $x$ était dans la classe $@k(x)$. Or $x$ ne change de classe que si $dist(x, g_{k(x)}) < dist(x, g_{@k(x)})$

$\rightarrow Q(P_K(t+1), L_K(t)) \leq Q(P_K(t), L_K(t))$

## Kmeans (propriétés)

- d'où $\sum_x dist(x, g_{k(x)})^2 \leq \sum_x dist(x, g_{@k(x)})^2$
- comme $Q(P_K(t), L_K(t)) = \sum_{k=1}^{K} \sum_{k \in C_k} dist(x, g_{k(x)})^2$
- peut se réécrire en $Q(P_K(t), L_K(t)) = \sum_x dist(x, g_{k(x)})^2$
- on a bien

$$Q(P_K(t+1), L_K(t)) \leq Q(P_K(t), L_K(t))$$

## Kmeans (propriétés)

- Montrons que $Q(P_K(t+1), L_K(t+1)) \leq Q(P_K(t+1), L_K(t))$
- Le recalcul des centres de gravité de

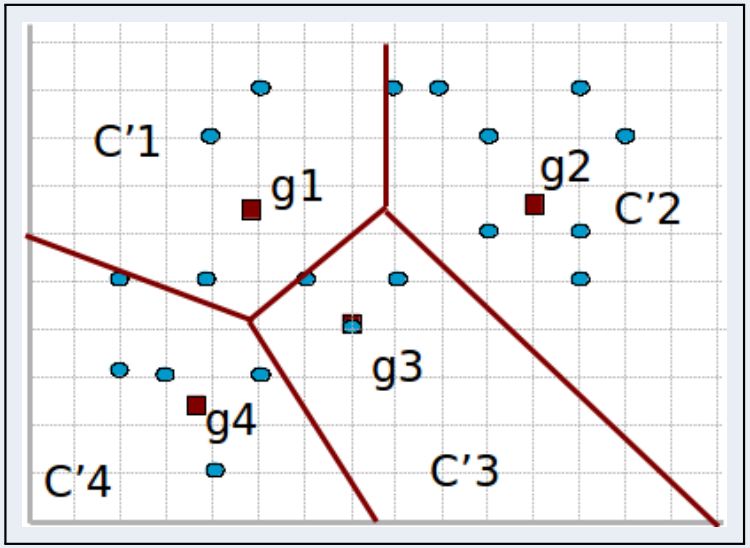
$$P_4(2) = \{C'1, C'2, C'3, C'4\}, L_4(1) = \{g1, g2, g3, g4\}$$

à

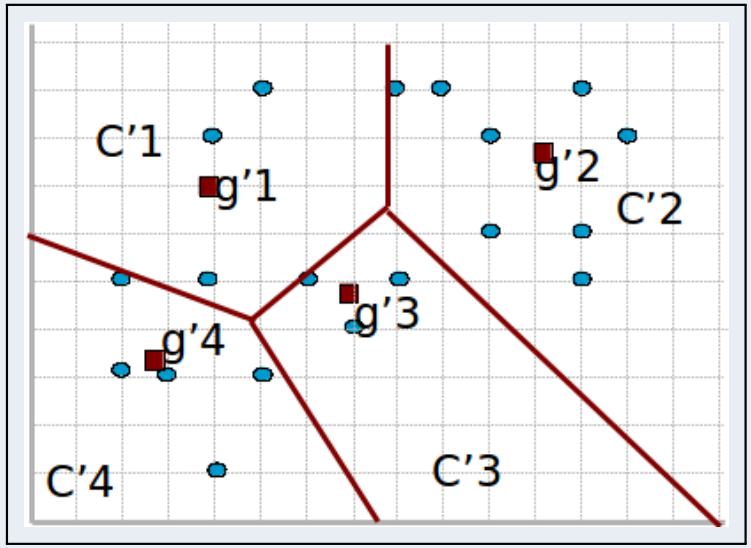
$$P_4(2) = \{C'1, C'2, C'3, C'4\}, L_4(2) = \{g'1, g'2, g'3, g'4\}$$

diminue-t-il l'inertie ?

# Kmeans (propriétés)




$P_4(2) =$
$\{C'1, C'2, C'3, C'4\}, L_4(1) =$
$\{g1, g2, g3, g4\}$




$P_4(2) =$
$\{C'1, C'2, C'3, C'4\}, L_4(2) =$
$\{g'1, g'2, g'3, g'4\}$

## Kmeans (propriétés)

- Par définition, les $g_k$ sont les centres de gravité des classes
- Or la propriété d'optimalité du centre de gravité (théorème de Huygens) entraîne l'inégalité sur l'inertie :
  - Pour toute classe $C_k$, l'inertie de $C_k$ par rapport au centre de gravité est inférieure à l'inertie par rapport à tout autre point
  - En particulier l'inertie $C$ par rapport à $g_k$ est inférieure à l'inertie de $C$ par rapport à @$g_k$

## Kmeans (propriétés)

d'où :

$$\sum_{k=1}^{K} Intertie(C_k, g_k) \leq \sum_{k=1}^{K} Intertie(C_k, @g_k)$$

d'où :

$$\sum_{k=1}^{K} \sum_{x \in C_k} dist(x, g_k)^2 \leq \sum_{k=1}^{K} \sum_{x \in C_k} dist(x, @g_k)^2$$

d'où :

$$Q(P_K(t+1), L_K(t+1)) \leq Q(P_K(t+1), L_K(t))$$

## Kmeans (propriétés)

- Ces deux propriétés entraînent la décroissance de $Q(P_K(t), L_K(t))$
- De plus, le nombre de partitions de D est fini. de fait, le nombre d'ensembles L est lui aussi fini.
- $Q(P_K(t+1), L_K(t))$ est donc une suite
  - décroissante
  - qui ne peut prendre qu'un nombre fini de valeur,
- Elle est donc stationnaire

## Kmeans (propriétés)

- Kmeans est donc un algorithme qui cherche à minimiser l'inertie inter-classe
- Néanmoins, l'algorithme ne fournit pas nécessairement le meilleur résultat mais simplement une suite de couples dont la valeur décroît $\rightarrow$ **optimisation locale**

## Kmeans (propriétés)

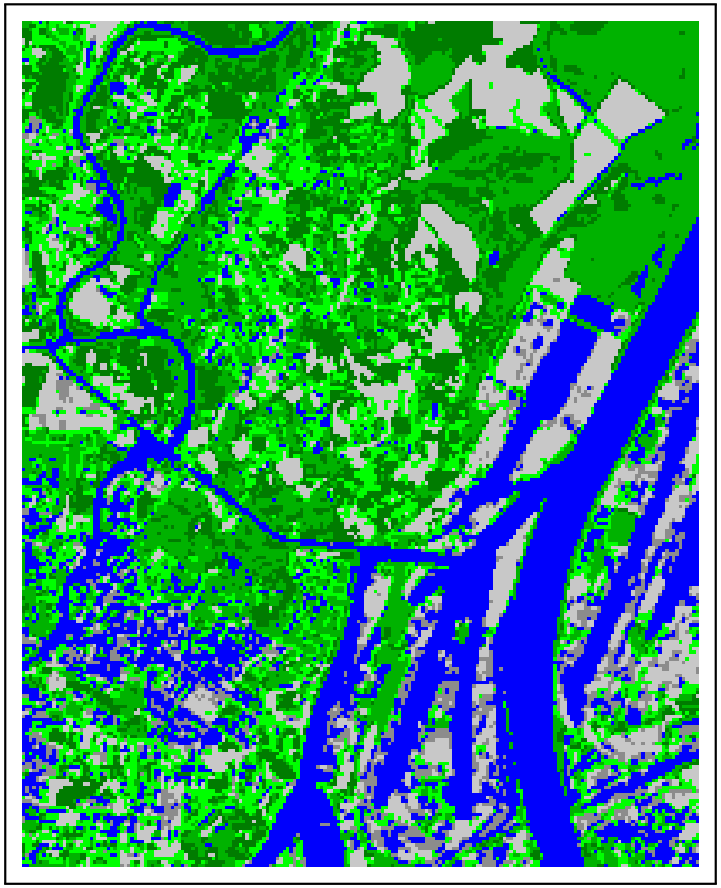
- Plus précisément, c'est un algorithme d'optimisation alternée en deux étapes :
  - recherche de la partition : minimisation de $Q(P, L)$ avec $L$ fixé
  - recherche des centres : minimisation de $Q(P, L)$ avec $P$ fixé
- L'expérience montre que la convergence est en fait assez rapide (moins de 10 tours), même avec de gros volumes de données et une initialisation "malheureuse"

## Kmeans - exemple

- Exemple sur une image SPOT

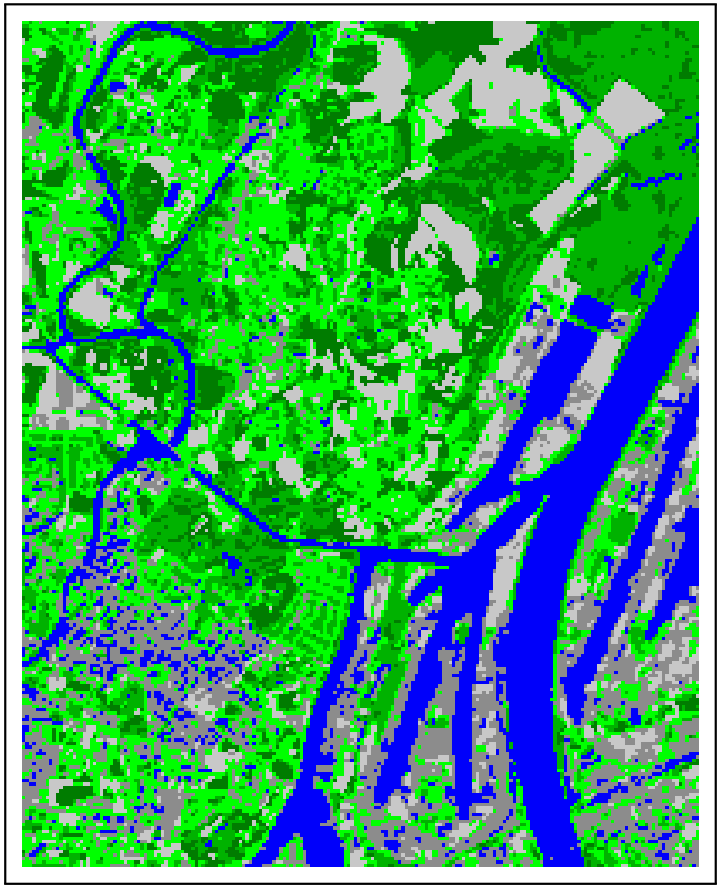
## Kmeans - exemple

- Exemple sur une image SPOT



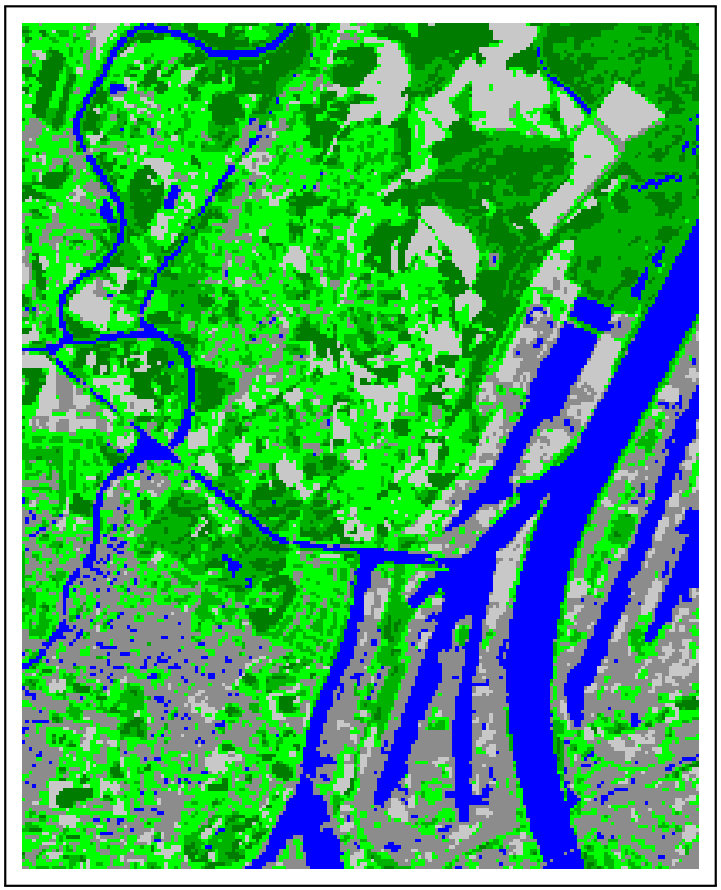
## Kmeans - exemple

- Exemple sur une image SPOT



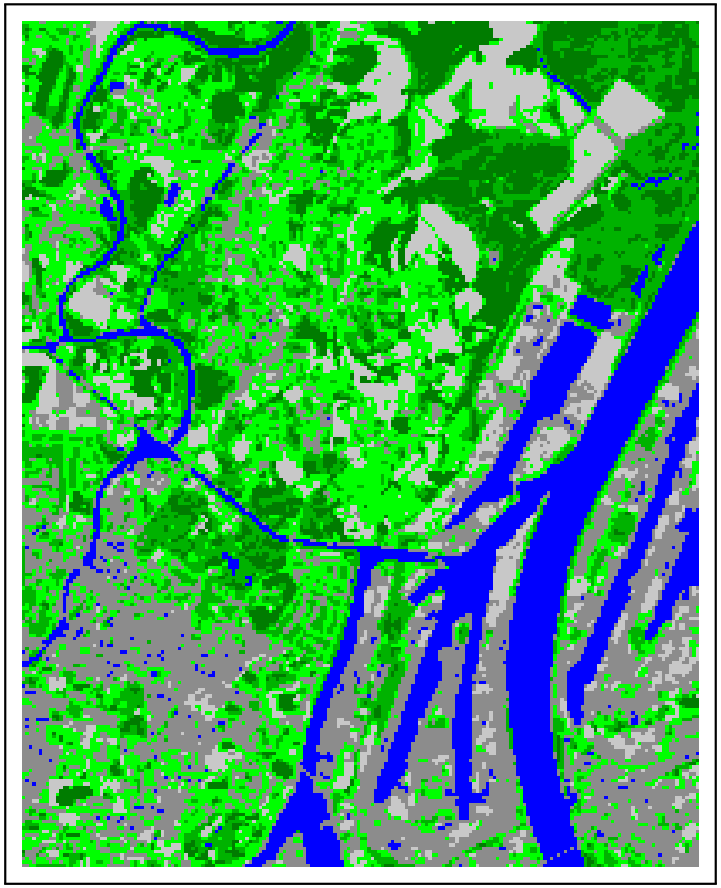
## Kmeans - exemple

- Exemple sur une image SPOT



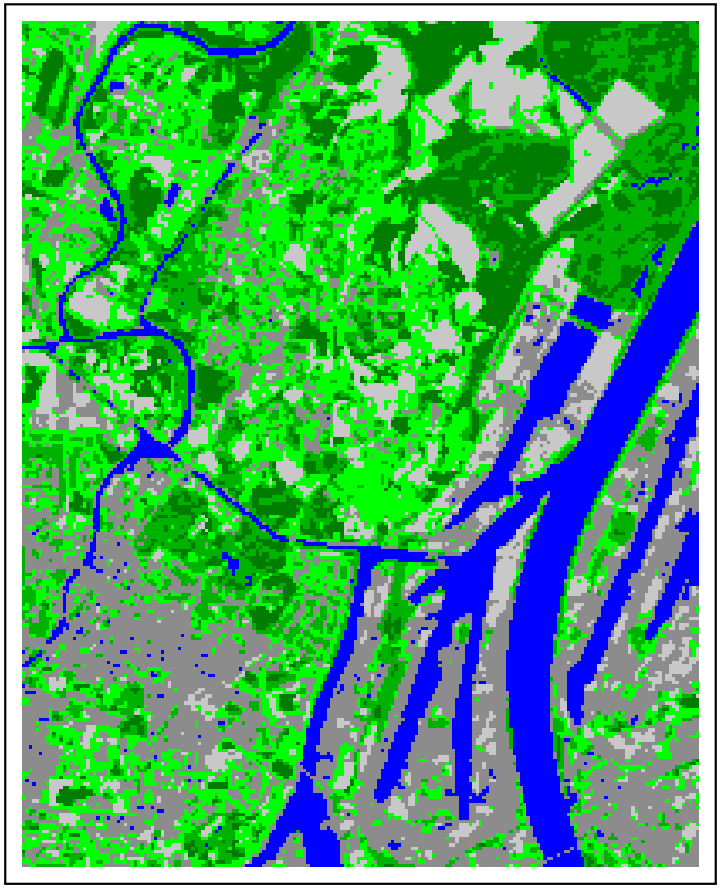
## Kmeans - exemple

- Exemple sur une image SPOT



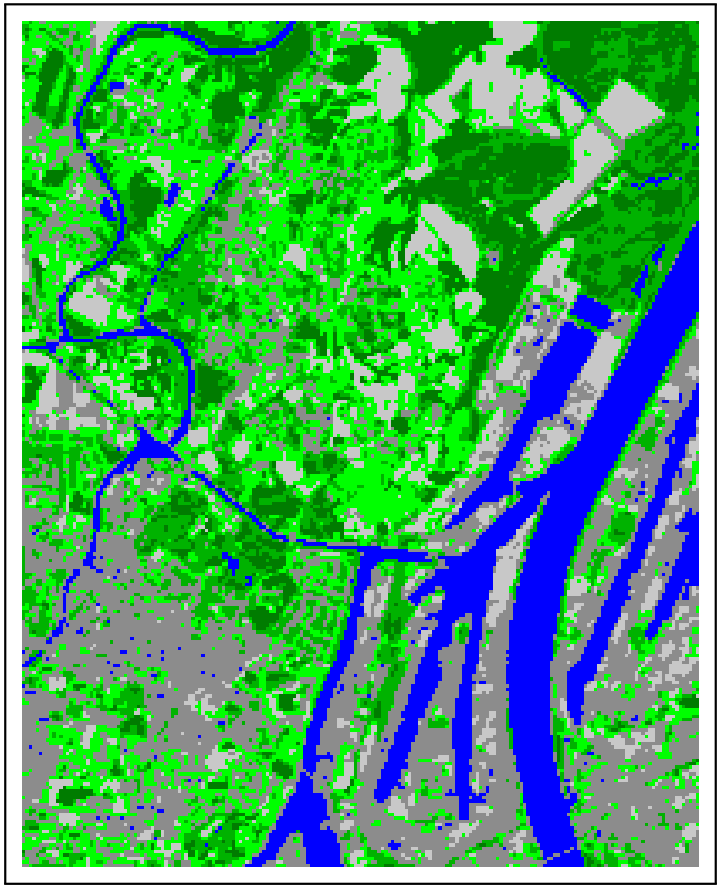
## Kmeans - exemple

- Exemple sur une image SPOT



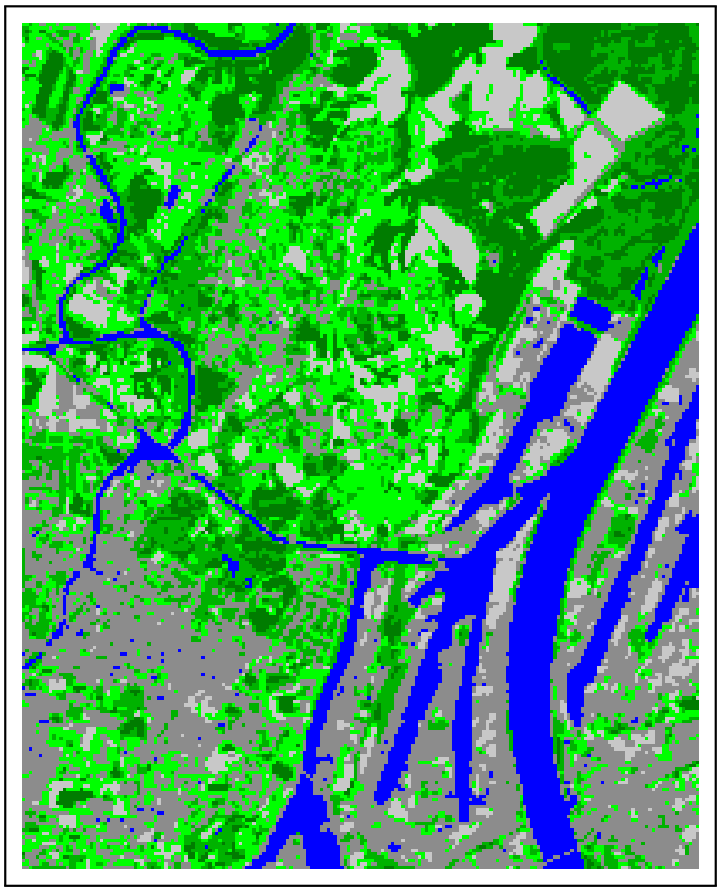
## Kmeans - exemple

- Exemple sur une image SPOT



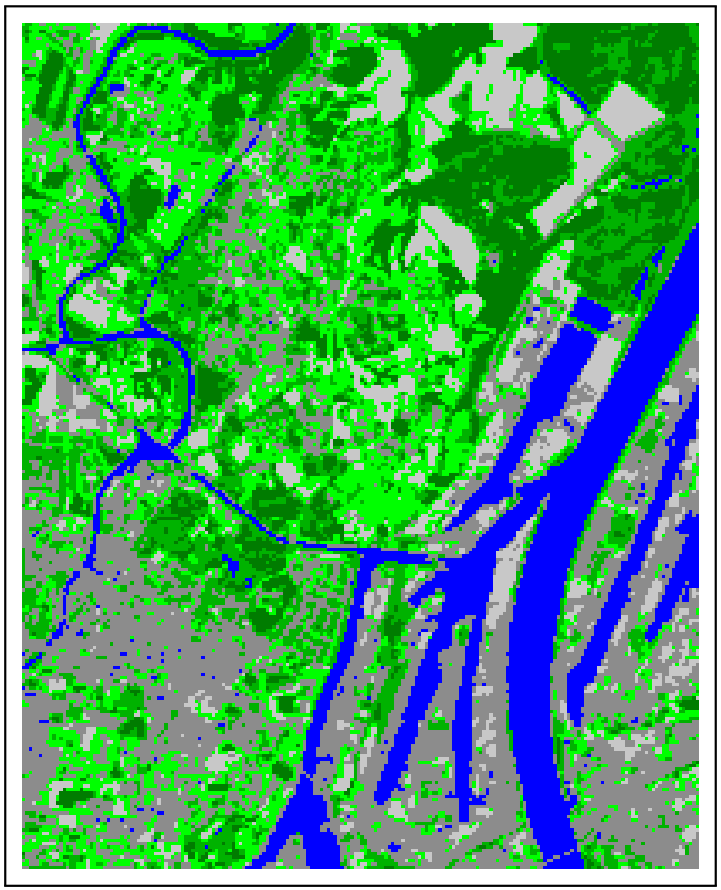
## Kmeans - exemple

- Exemple sur une image SPOT



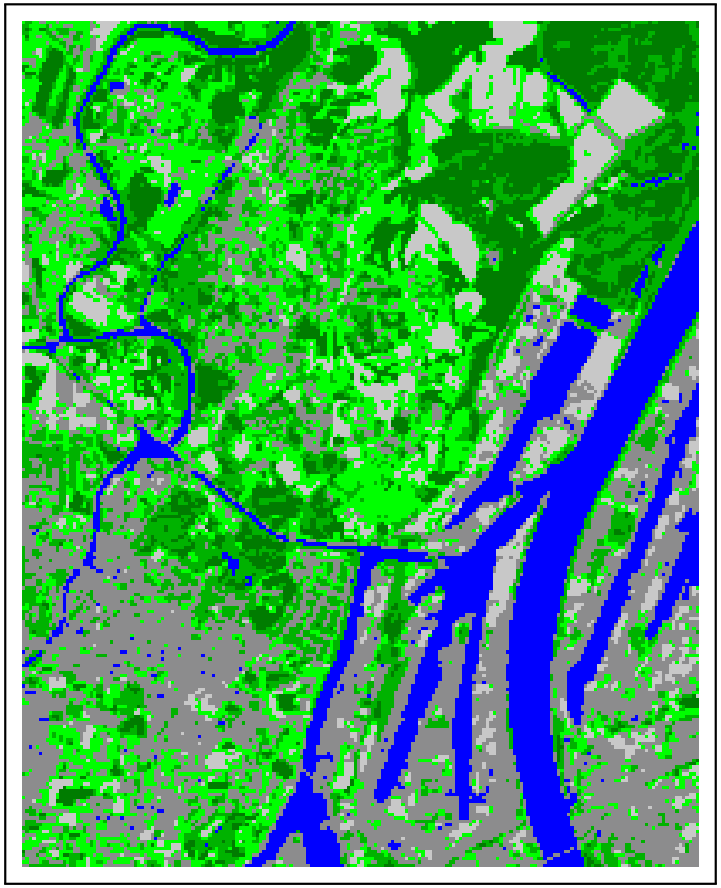
## Kmeans - exemple

- Exemple sur une image SPOT



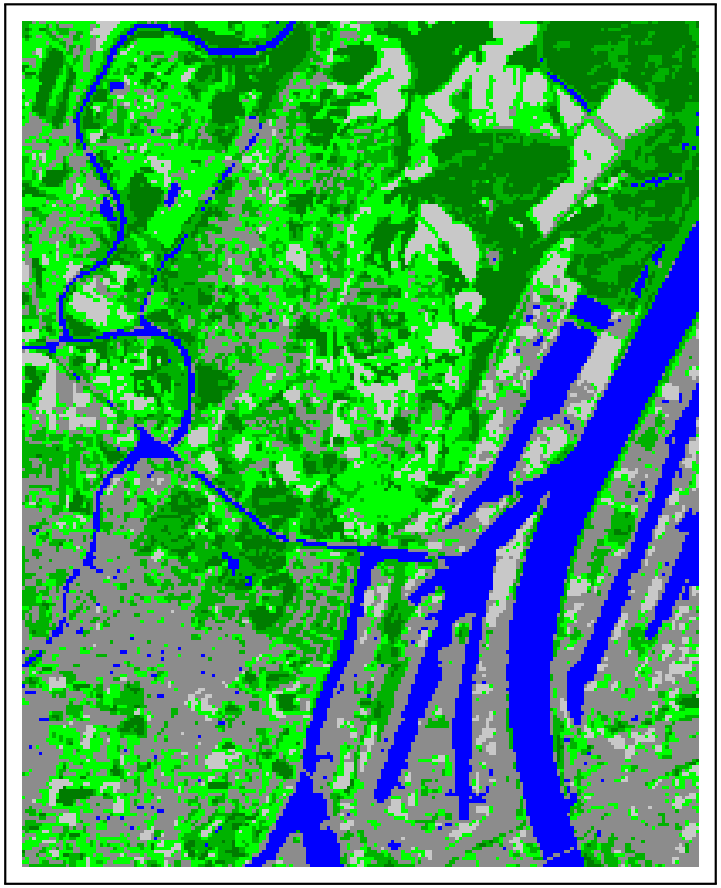
## Kmeans - exemple

- Exemple sur une image SPOT



## Kmeans - exemple

- Exemple sur une image SPOT



## Partitionnement : méthode séquentielle

Méthode séquentielle (MacQueen, 1967)

- Les $K$ noyaux initiaux sont tirés au hasard dans les $D$ points
- A chaque itération $q$, un individu $x_i$ est choisi au hasard :
  1. détermination du noyau $N_k$ le plus proche de $x_i$
  2. affectation de $x_i$ à la classe $k$
  3. déplacement du noyau $N_k$ par : $N'_k = \frac{x_i + n_k \times N_k}{n_k + 1}$
  4. où $n_k$ est le cardinal de $N_k$.

Nuées dynamiques :

- L'idée est de remplacer les centres de gravité par des représentants adaptés au problème que l'on cherche à résoudre.
- On utilise les principes de Kmeans

## Partitionnement par réallocation : problèmes

Deux problèmes principaux aux méthodes par réallocation

1. **Minimum local** : Kmeans donne des résultats différents suivant le choix des centres initiaux
2. **Choix du nombre de classes** : comment combiner/comparer deux partitions à nombre de classes différents

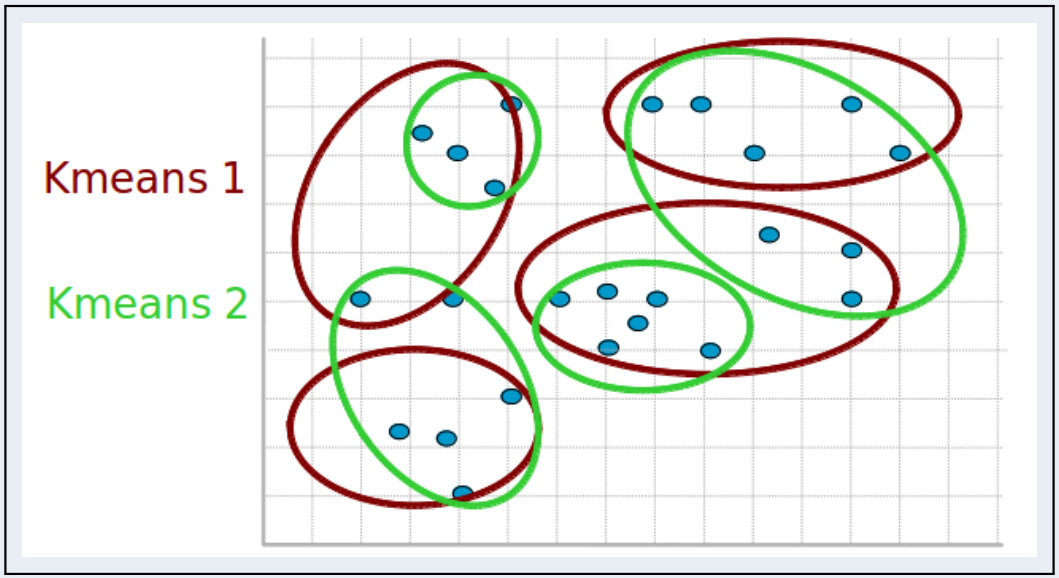
## Partitionnement par réallocation : minimums locaux

**Premier problème** : pour un K donné, comment initialiser les centres ?

- On utilise des informations supplémentaires
- On étudie les données avant : histogrammes, etc.
- On teste des initialisations différentes et on garde le meilleur
- On s'en sert comme résultat intermédiaire

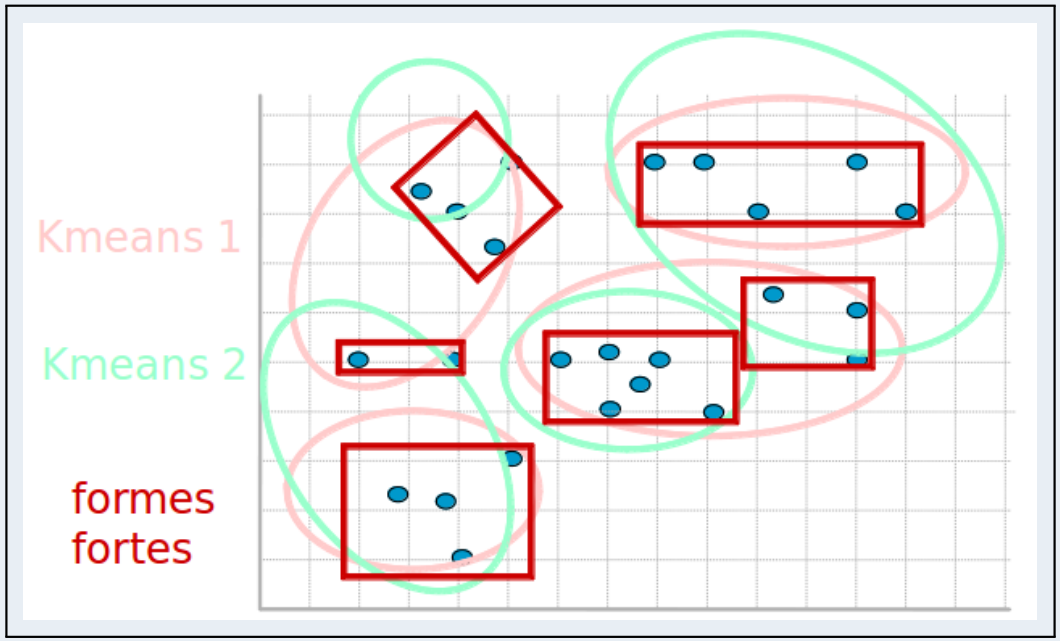
## Formes fortes

- Les formes fortes sont obtenues par superposition (ou intersection des différentes partitions)



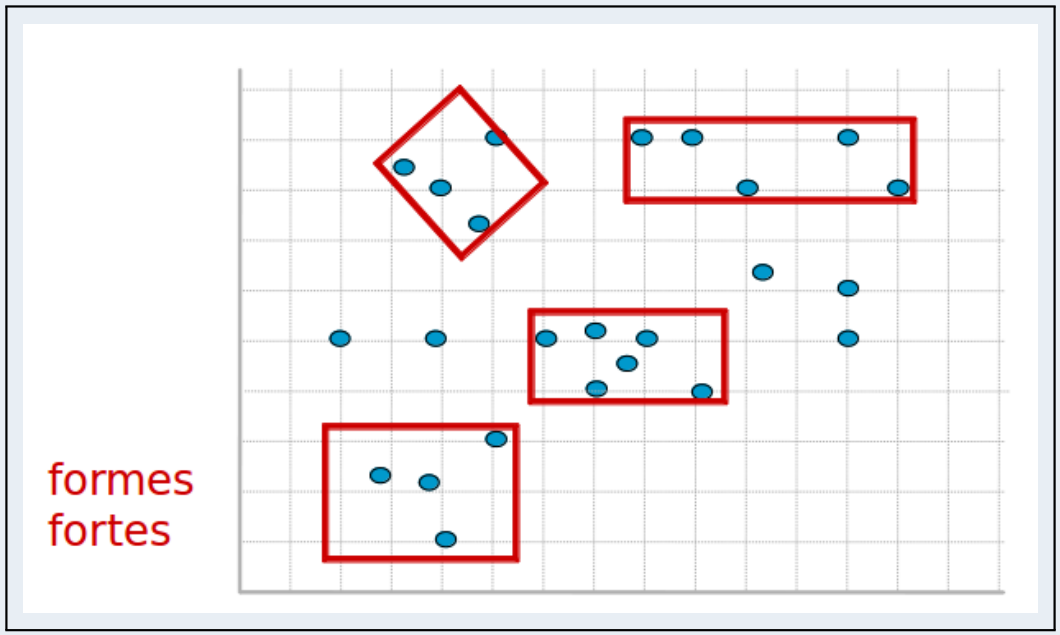
## Formes fortes

- Les formes fortes sont obtenues par superposition (ou intersection des différentes partitions)



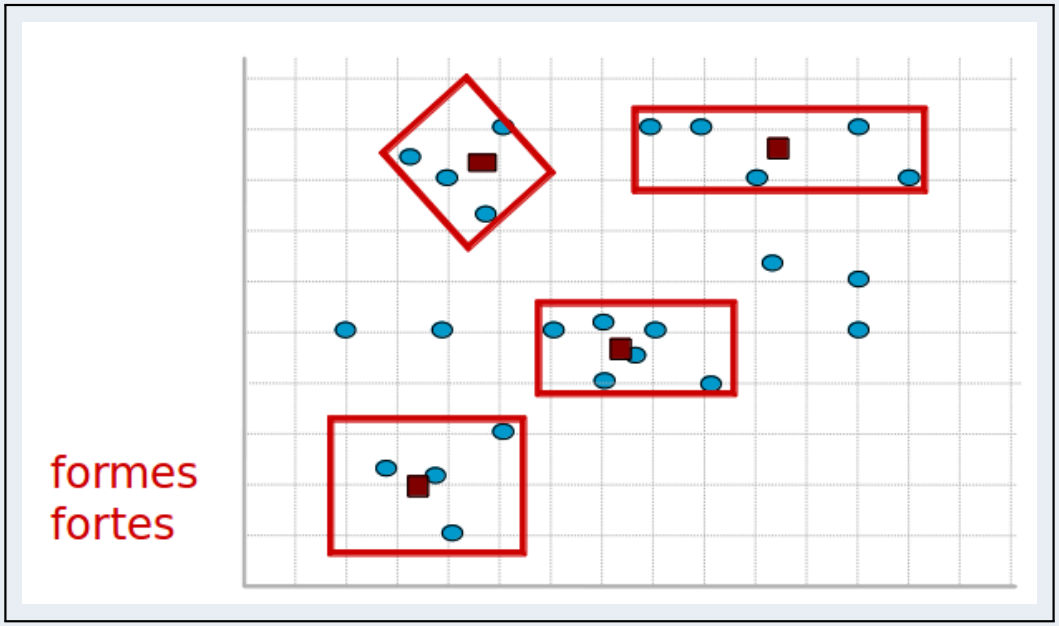
## Formes fortes

- On garde les 4 "meilleures" formes



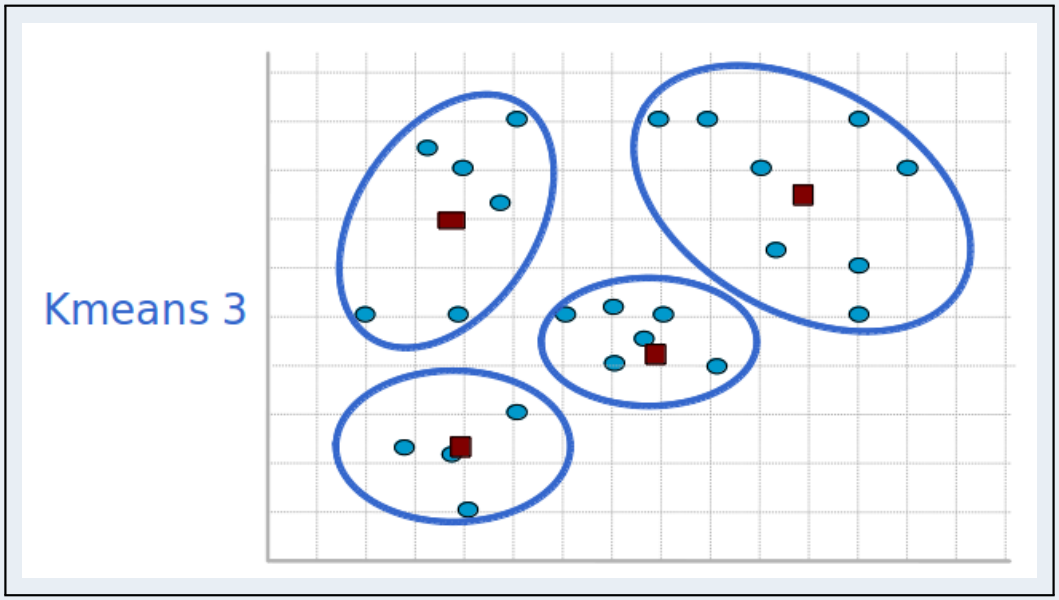
## Formes fortes

- Puis on effectue un Kmeans en prenant les centres de gravité de formes fortes pour centres initiaux



## Formes fortes

- Puis on effectue un Kmeans en prenant les centres de gravité de formes fortes pour centres initiaux



## Etude d'histogramme

**Deuxième problème** : comment choisir K ?

- Premier cas : K est choisi a priori
  - information supplémentaire
  - étude des histogrammes des valeurs
- Deuxième cas : K évolue en cours de classification
  - classification interactive : l'utilisateur crée/supprime des classes
  - utilisation d'un critère statistique

## Nombre de classes ?

- On peut utiliser une mesure qui tient compte de la variance des données dans chaque classe.

$$T = \frac{1}{n} \sum_{k=1}^{K} \sum_{x_i \in C_k} |x_i - g_k|^2$$

## Isodata

Algorithme :

- Répeter
  1. effectuer un groupement par k-moyennes
  2. éliminer les cluster trop petits
  3. scinder les clusters trop dispersés
  4. fusionner les groupes trop proches
- Jusqu'à stabilisation

## Partitionnement flou

Partitionnement flou :

- $\forall k = 1 \dots K, \forall x_i \in D, c_{i,k} \in [0, 1]$
- Le nombre d'objets dans une classe est inférieur au nombre total d'objets :

$$\forall k = 1 \dots K, 0 < \sum_{i=1}^{n} c_{i,k} < n$$

- Un objet appartient à différents degrés, à toutes les classes :

$$\forall x_i \in D, \sum_{k=1}^{K} c_{i,k} = 1$$

## C-Moyennes flou (Fuzzy C-means)

- On va minimiser un critère heuristique :

$$J_{fuzzy} = \sum_{j=1}^{C} \sum_{i=1}^{n} c_{i,j}^{b} ||x_i - \text{centre}_j||^2$$

$$b \geq 1$$

- où $b$ règle le "degré" de flou (en général $b = 2$)

## C-Moyennes flou (Fuzzy C-means)

- On fixe un nombre de clusters K, et une matrice d'appartenance C :
  1. On calcule les centroïdes des classes

$$\text{centre}_k = \frac{\sum_{i=1}^{n}(c_{i,k}^b) * x_i}{\sum_{i=1}^{n}(c_{i,k}^b)}$$

  2. On calcule la nouvelle matrice d'appartenance suivant la position des centres :

$$c_{i,j} = \frac{(\frac{1}{d_{i,k}^2})^{1/(b-1)}}{\sum_{j=1}^{C}(\frac{1}{d_{i,j}^2})^{1/(b-1)}}$$

  3. On recalcule $J_{fuzzy}$ : on reprend à 2 si $J_{fuzzy}$ a augmenté

## ISODATA flou

Si K n'est pas connu, on cherche comme dans Isodata avec les critères :

- Le coefficient de partionnement flou

$$F(M) = \frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} (c_{i,k}^{b})$$

- L'entropie

$$H(M) = \frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} c_{i,k} * log(c_{i,k})$$

## ISODATA flou

Rappel sur l'entropie :

$$H(M) = \sum_{k=1}^{K} p_k * log(p_k) \quad \text{où} \quad p_k = \frac{|C_k|}{|M|}$$

- $H(M)$ est minimale (égale à 0) si les données sont réparties dans une seule classe
- $H(M)$ est maximale (dépend du nombre de classes) si les données sont réparties uniformément dans toutes les classes

## Clustering simple

Différentes méthodes :

- clustering par partitionnement
- clustering par modélisation
- clustering hiérarchique

Et bien d'autres :

- clustering basé sur la densité
- clustering par "grille"

## Clustering simple

Différentes méthodes :

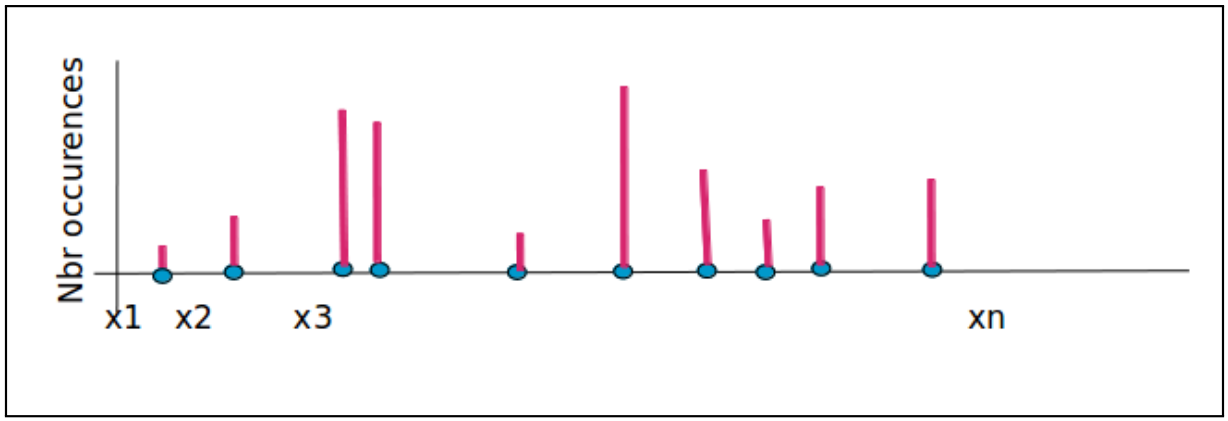
- clustering par partitionnement
- clustering par modélisation
- clustering hiérarchique

Et bien d'autres :

- clustering basé sur la densité
- clustering par "grille"

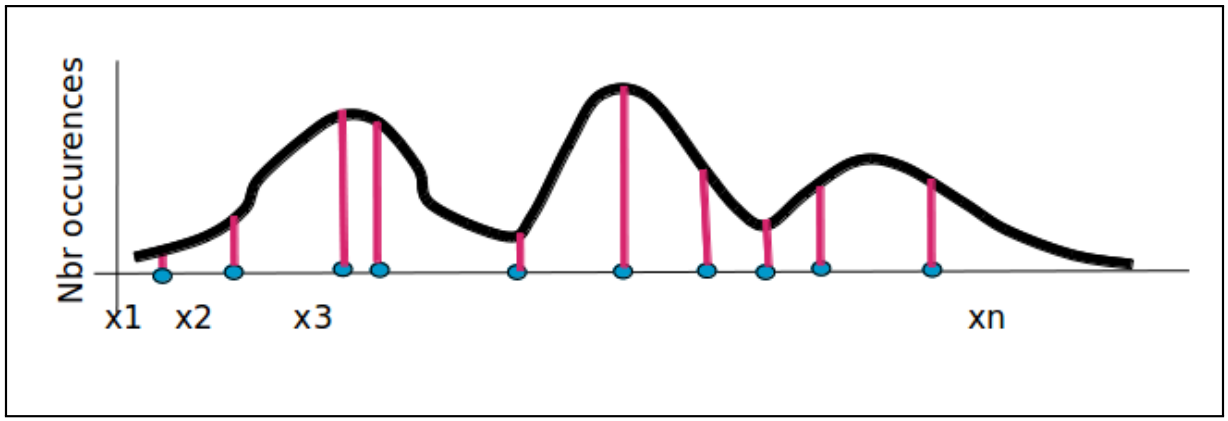
## Mélange de lois

- Le but est de modéliser chaque cluster par une loi de probabilité.
- Soit une "distribution" des valeurs donnée



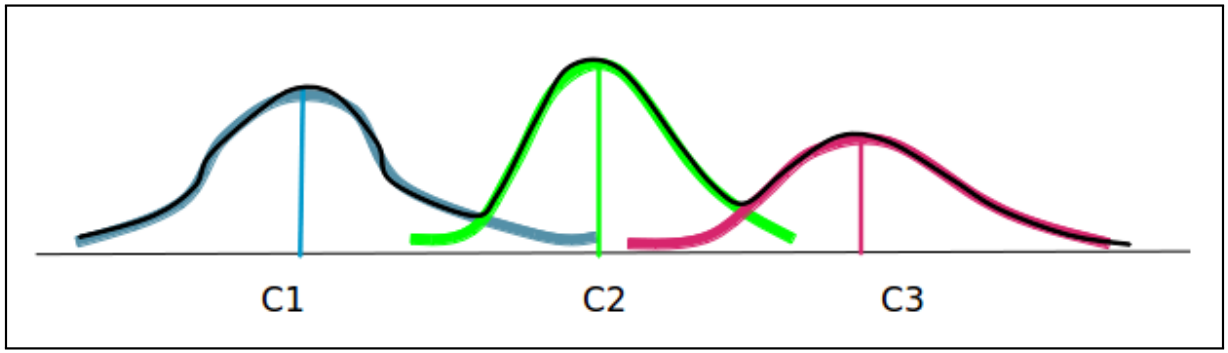
## Mélange de lois

- Le but est de modéliser chaque cluster par une loi de probabilité.
- Soit une "distribution" des valeurs donnée



## Mélange de lois

- On suppose que cette loi est un fait un mélange de lois tel que à chaque cluster correspond une loi de probabilité normale (mélange gaussien) ou non
- Exemple avec trois classes et un seul attribut



## Mélange de lois

Il faut donc retrouver les trois lois

- On suppose que chaque observation est issue d'une seule source C1, C2 ou C3
- Par exemple, si on suppose que les lois sont normales alors, il faut "retrouver " :

$$v = (\alpha_1, \sigma_1, \mu_1, \alpha_2, \sigma_2, \mu_2, \alpha_3, \sigma_3, \mu_3)$$

- avec :

$$p(x|v) = \sum_{k=1}^{3} \alpha_k N(x, \mu_k, \sigma_k) \quad \text{avec} \quad \sum_{k=1}^{K} \alpha_k = 1$$

## Mélange de lois

- Dans le cas général, il faut trouver le "meilleur"

$$v = (\alpha_1, \theta_1, \ldots, \alpha_k, \theta_k, \ldots, \alpha_K, \theta_K)$$

- Estimant :

$$p(x|v) = \sum_{k=1}^{K} \alpha_k * p(x|C_k, \theta_k) \quad \text{avec} \quad \sum_{k=1}^{K} \alpha_k = 1$$

## Mélange de lois

Vraisemblance :

$$L(v|x_1,\ldots,x_n) = \prod_{i=1}^{n} p(x_i|v) = \prod_{i=1}^{n}\prod_{k=1}^{K} \alpha_k * p(x_i|C_k,\theta_k)$$

- La vraisemblance quantifie la probabilité que les observations proviennent effectivement de la loi $p$

## Mélange de lois

- On travaille en fait sur le *ln* pour simplifier :

$$l(v|x_1,\ldots,x_n) = ln(\prod_{i=1}^{n} p(x_i|v)) = \sum_{i=1}^{n} ln(p(x_i|v))$$

$$l(v|x_1,\ldots,x_n) = \sum_{i=1}^{n} ln(\sum_{k=1}^{K} \alpha_k p(x_i|C_k,\theta_k))$$

## Mélange de lois

- Il faut donc trouver la valeur de $\theta = \{\theta_k\}$ qui maximise cette vraisemblance.
- Maximiser $l(X, v)$ est difficile car il n'y a pas de dérivée pour le logarithme d'une somme
- Il faut trouver un moyen de simplifier le calcul du maximum de vraisemblance
- On va utiliser un processus (itératif) de calcul ou d'estimation du maximum

## EM – cas gaussien

- On va donc faire l'hypothèse que chaque objet est issu d'une des lois : mais on ne sait pas laquelle
- Posons $h_i$ une variable donnant l'appartenance de l'objet $o_i$ :
  - $h_k = 1$ si l'objet $o_i$ appartient à la classe $C_k$
  - Un objet se décrit donc par : $o_i = (x_i, h_1, \ldots, h_K)$

Le but est donc de trouver :

- Les paramètres des lois
- Pour chaque objet la loi qui l'a généré

## EM

Deux étapes :

1. Etape E (*expectation*) : avec l'hypothèse $\theta_j$
   - on calcule $h_i$ pour tous les points
   - on calcule l'espérance de la vraisemblance

$$Q = \sum_{i=1}^{n}\sum_{k=1}^{K} ln(p(C_k|x_i,\theta_j)).p(C_k) + \sum_{i=1}^{n}\sum_{k=1}^{K} ln(p(x_i|\theta_j)).p(C_k|x_i,\theta_j)$$

2. Etape M (*maximisation*) : on maximise E
   - on cherche $\theta_{j+1}$ qui maximise cette espérance

## EM – cas gaussien

1. Etape E (*expectation*)

- Soient $h^t, \alpha^t$, etc. les valeurs recherchées à l'étape $t$ alors l'espérance

$$E(h_j^t = 1 | x_i, v^t) = \frac{\alpha_j^t N(x_i, \mu_j^t, \sigma_j^t)}{\sum_{k=1}^{K} E(h_i | x_i, v)^{(t)}}$$

- Posons

$$S_k^{(t+1)} = \sum_{i=1}^{n} E(h_i | x_i, v)^{(t)}$$

## EM – cas gaussien

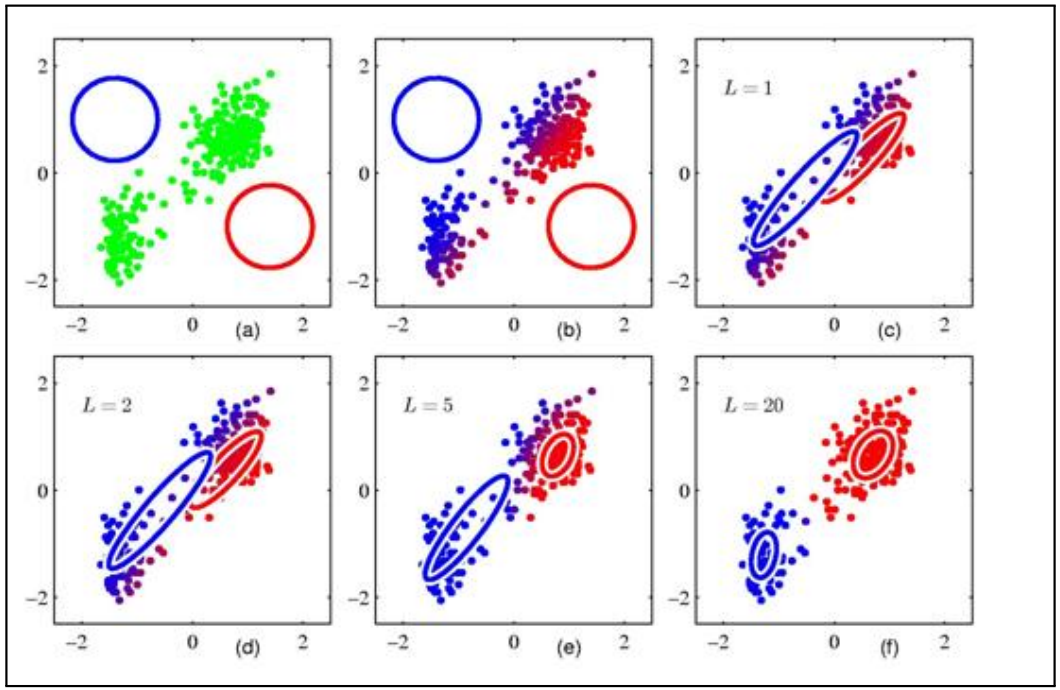
2. Etape M (*maximisation*)

$$\alpha_k^{(t+1)} = \frac{1}{n} \sum_{i=1}^{n} S_k^{(t+1)}$$

$$\mu_k^{(t+1)} = \frac{1}{S_k^{(t+1)}} \sum_{i=1}^{n} x_i E(h_i | x_i, v)^{(t)}$$

$$\sigma_k^{2(t+1)} = \frac{1}{S_k^{(t+1)}} \sum_{i=1}^{n} (x_i - \mu_k^{(t+1)})^2 E(h_i | x_i, v)^{(t)}$$

# Exemple d'itérations de EM:




source : Expectation maximization Algorithm for Gaussian mixtures from C. M. Bishop's book.