Clustering

Classification automatique

Mean – Shift + DBSCAN

Classification :

Définition informelle :Trouver dans un ensemble d'objets des groupes homogènes (classes) et bien distincts les uns des autres.

Classement :

À partir d'exemples d'objets répartis en groupes, construire un algorithme qui détermine le groupe adapté pour un nouvel objet.

Clustering

Collision Français et Anglais :

Français	Anglais
Classification	*Clustering*
Classement	*Classification*

Opposition entre

- supervisé (classement) : groupes fixés, exemples d'objets de chaque groupe
- non supervisé (classification) : on ne connaît pas de groupe

- Regroupement (Clustering): construire une collection d'objets :
  - Similaires au sein d'un même groupe ou homogènes.
  - Dissimilaires quand ils appartiennent à des groupes différents ou bien distincts.
- Le Clustering est de la classification non supervisée : pas de classes prédéfinies.

Quelques applications :

Analyse exploratoire de données (typologie) :

- marketing : typologie des clients
- bioinformatique : regroupement de gènes
- image : segmentation de zones homogènes

Simplification de données :

- recherche d'information : regroupement de pages web (par ex. Clusty)
- données très volumineuses : chaque groupe est remplacé par un représentant

Qu'est ce qu'un bon regroupement ?

- Une bonne méthode de regroupement permet de garantir
  - Une grande similarité intra-groupe
  - Une faible similarité inter-groupe
- La qualité d'un regroupement dépend donc de la mesure de similarité utilisée par la méthode et de son implémentation.

Mesurer la qualité d'un clustering

- Métrique pour la similarité: La similarité est exprimée par le biais d'une mesure de distance
- Une autre fonction est utilisée pour la mesure de la qualité
- Les définitions de distance sont très différentes que les variables soient des intervalles (continues), catégories, booléennes ou ordinales
- En pratique, on utilise souvent une pondération des variables.

Types des variables

- Intervalles:
- Binaires:
- catégories, ordinales, ratio:
- Différents types:

Intervalle (discrètes)

- Standardiser les données
  - Calculer l'écart absolu moyen:

$$s_f = \frac{1}{n}(|x_{1f} - m_f| + |x_{2f} - m_f| + ... + |x_{nf} - m_f|)$$

où

$$m_f = \frac{1}{n}(x_{1f} + x_{2f} + ... + x_{nf})$$

  - Calculer la mesure standardisée (*z-score*)

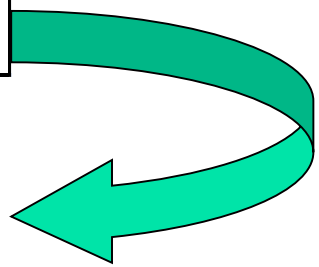
$$z_{if} = \frac{x_{if} - m_f}{s_f}$$

Exemple

	Age	Salaire
Personne1	50	11000
Personne2	70	11100
Personne3	60	11122
Personne4	60	11074

$$M_{Age} = 60 \qquad S_{Age} = 5$$

$$M_{salaire} = 11074 \qquad S_{salaire} = 148$$



	Age	Salaire
Personne1	-2	-0,5
Personne2	2	0,175
Personne3	0	0,324
Personne4	0	2

Exemple: distance de Manhattan

	Age	Salaire
Personne1	50	11000
Personne2	70	11100
Personne3	60	11122
Personne4	60	11074

→ d(p1,p2)=120

d(p1,p3)=132

Conclusion: p1 ressemble plus à p2 qu'à p3 ☹

	Age	Salaire
Personne1	-2	-0,5
Personne2	2	0,175
Personne3	0	0,324
Personne4	0	0

→ d(p1,p2)=4,675

d(p1,p3)=2,324

Conclusion: p1 ressemble plus à p3 qu'à p2 ☺

Variables binaires

- Variable symétrique: Ex. le sexe d'une personne, i.e coder masculin par 1 et féminin par 0 c'est pareil que le codage inverse.

- Variable asymétrique: Ex. Test HIV. Le test peut être positif ou négatif (0 ou 1) mais il y a une valeur qui sera plus présente que l'autre. Généralement, on code par 1 la modalité la moins fréquente
  - 2 personnes ayant la valeur 1 pour le test sont *plus similaires* que 2 personnes ayant 0 pour le test.

Variables Nominales

- Une généralisation des variables binaires, ex: rouge, vert et bleu
- Méthode 1: Matching simple
  - *m*: # d'appariements, *p*: # total de variables

$$d(i, j) = \frac{p - m}{p}$$

- Méthode 2: utiliser un grand nombre de variables binaires
  - Créer une variable binaire pour chaque modalité (ex: variable rouge qui prend les valeurs vrai ou faux)

Variables Ordinales

On qualifie d'ordinale une variable qualitative pour laquelle la valeur mesurée sur chaque individu (parfois qualifiée de catégorie ou de modalité) est numérique.

Une variable ordinale(discrète ou continue) peut être traitée comme les variables intervalles. L'ordre est important(ex: classement).

On peut alors classer les individus par valeurs croissantes ou décroissantes. La moyenne sur plusieurs individus n'a pas toujours de sens mathématiquement, et peu d'intérêt en pratique. On doit s'intéresser à la médiane.

Exemples :

- L'appréciation : très mauvais … excellent (moyenne ???)
- La taille et la température peuvent également être considérées comme des variables quantitatives continues. Ainsi, le calcul de la moyenne devient pertinent.

En Présence de Variables de différents Types

Pour chaque type de variables utiliser une mesure adéquate.

Problèmes: les clusters obtenus peuvent être différents.

On utilise une formule pondérée pour faire la combinaison.

$$d(i,j) = \frac{\sum_{f=1}^{p} \delta_{ij}^{(f)} d_{ij}^{(f)}}{\sum_{f=1}^{p} \delta_{ij}^{(f)}}$$

Approches de Clustering

- Algorithmes de Partitionnement: Construire plusieurs partitions puis les évaluer selon certains critères.
- Algorithmes hiérarchiques: Créer une décomposition hiérarchique des objets selon certains critères.
- Algorithmes basés sur la densité: basés sur des notions de connectivité et de densité.
- Algorithmes de grille: basés sur un structure à multi-niveaux de granularité .
- Algorithmes à modèles: Un modèle est supposé pour chaque cluster ensuite vérifier chaque modèle sur chaque groupe pour choisir le meilleur.

Algorithmes à partionnement

- Construire une partition à ***k*** clusters d'une base ***D*** de ***n*** objets
- Les *k clusters doivent* optimiser le critère choisi
  - Global optimal: Considérer toutes les k-partitions
  - Heuristic methods: Algorithmes *k-means* et *k-medoids*
  - *k-means* (MacQueen'67): Chaque cluster est représenté par son centre
  - *k-medoids* or PAM (Partition around medoids) (Kaufman & Rousseeuw'87): Chaque cluster est représenté par un de ses objets

La méthode des k-moyennes (*K-Means)*

- L'algorithme *k-means* est en 4 étapes :

1. Choisir k objets formant ainsi k clusters

2. (Ré)affecter chaque objet O au cluster C_i de centre M_i tel que dist(O,M_i) est minimal

3. Recalculer M_i de chaque cluster (le barycentre)

4. Aller à l'étape 2 si on vient de faire une affectation

K-Means :Exemple

- A={1,2,3,6,7,8,13,15,17}. Créer 3 clusters à partir de A

- On prend 3 objets au hasard. Supposons que c'est 1, 2 et 3. Ca donne C_1={1}, M_1=1, C_2={2}, M_2=2, C_3={3} et M_3=3

- Chaque objet O est affecté au cluster au milieu duquel, O est le plus proche. 6 est affecté à C_3 car dist(M_3,6)<dist(M_2,6) et dist(M_3,6)<dist(M_1,6)

  On a C_1={1}, M_1=1,

  C_2={2}, M_2=2

  C_3={3, 6,7,8,13,15,17}, M3=69/7=9.86

K-Means :Exemple (suite)

- dist(3,M2)<dist(3,M3)➔3 passe dans C_2. Tous les autres objets ne bougent pas. C_1={1}, M_1=1, C_2={2,3}, M_2=2.5,C_3={6,7,8,13,15,17} et M_3= 66/6=11

- dist(6,M_2)<dist(6,M_3)➔6 passe dans C2. Tous les autres objets ne bougent pas. C_1={1}, M_1=1, C_2={2,3,6}, M_2=11/3=3.67, C3={7,8,13,15,17}, M_3= 12

- dist(2,M1)<dist(2,M2)➔2 passe en C1. dist(7,M2)<dist(7,M3)➔ 7 passe en C2. Les autres ne bougent pas. C1={1,2}, M1=1.5, C2={3,6,7}, M2=5.34, C3= {8,13,15,17}, M3=13.25

- dist(3,M1)<dist(3,M2)➔3 passe en 1. dist(8,M2)<dist(8,M3)➔8 passe en 2 C1={1,2,3}, M1=2, C2={6,7,8}, M2=7, C3={13,15,17}, M3=15

Plus rien ne bouge $\Rightarrow$ Arrêt

Commentaires sur la méthode des *K-Means*

- Force
  - *Relativement efficace*: *O*(*tkn*), où *n* est # objets, *k* est # clusters, et *t est* # itérations. Normalement, *k*, *t* << *n*.
  - Tend à réduire

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} \left| p - m_i \right|^2$$

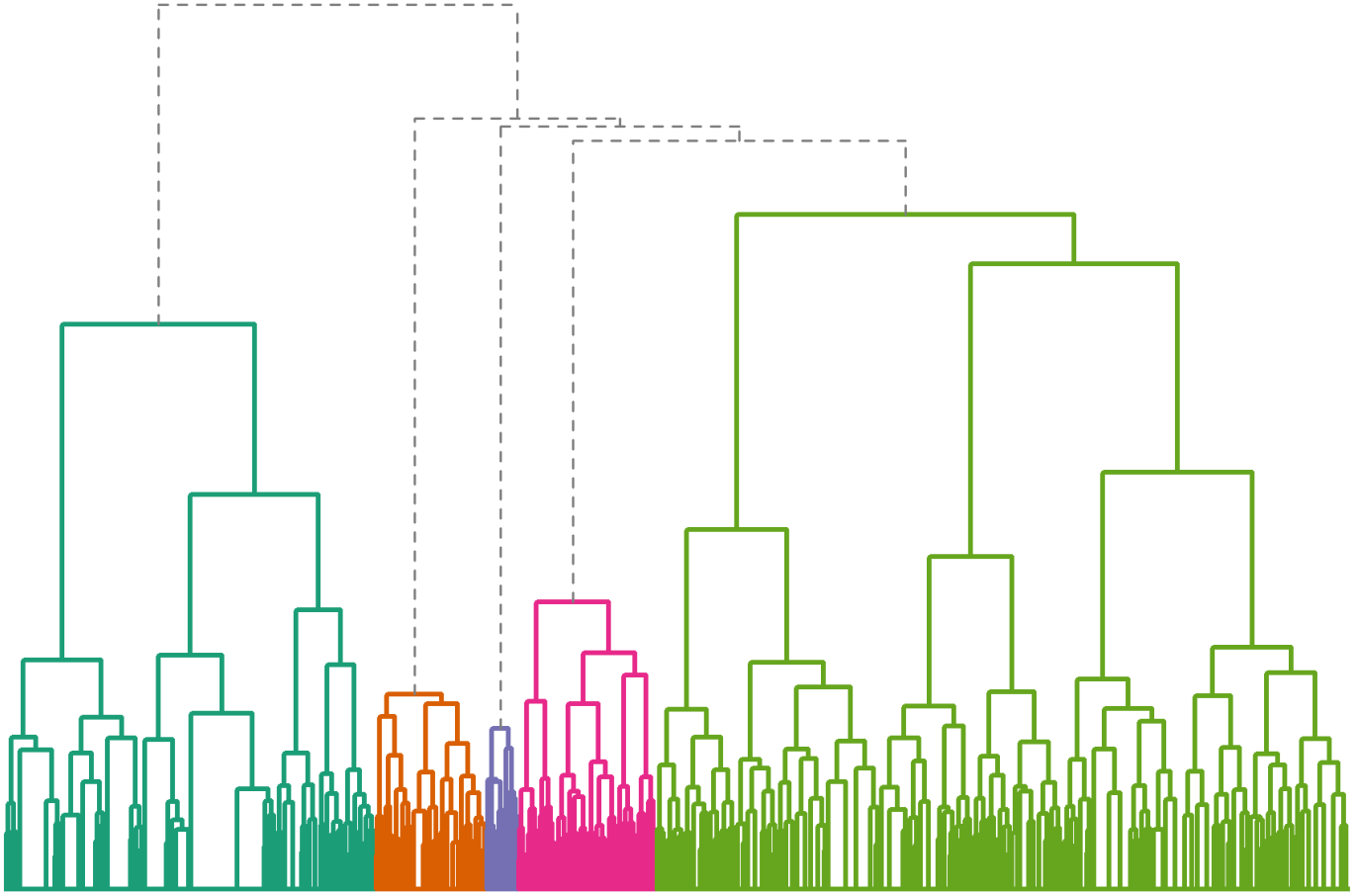
- Faiblesses
  - N'est pas applicable en présence d'attributs qui ne sont pas du type intervalle (moyenne=?)
  - On doit spécifier k (nombre de clusters)
  - Les clusters sont construits par rapports à des objets inexistants (les milieux)
  - Ne peut pas découvrir les groupes *non-convexes*

Variante de K-means

- K-Medians est un autre algorithme de clustering lié à K-Means, sauf qu'au lieu de recalculer les points centraux du groupe en utilisant la moyenne nous utilisons le vecteur médian du groupe.

- Cette méthode est moins sensible aux valeurs aberrantes (en raison de l'utilisation de la médiane), mais elle est beaucoup plus lente pour les ensembles de données plus importants, car un tri est requis à chaque itération pour calculer le vecteur médian.

Classification Ascendante Hiérarchique



Classification Ascendante Hiérarchique

- La CAH est une méthode de classification qui permet de mettre en évidence un **Regroupement** « naturel » d'un ensemble d'individus décrits par des caractéristiques (les variables).
- Elle propose une série de partitions emboîtées représentées sous forme d'arbres appelés **dendogrammes**.
- L'algorithme procède par **agrégations successives**, partant de la partition la plus fragmentaire, un individu est égal à une classe, jusqu'à la partition triviale, le regroupement de tous les individus dans une et une seule classe.

Classification Ascendante Hiérarchique

Principe :

- Créer à chaque étape une partition obtenue en agrégeant 2 à 2 les éléments les plus proches.

  Eléments : individus ou groupe d'individus.

- L'algorithme fournit une hiérarchie de partitions : arbre contenant l'historique de la classification et permettant de retrouver **n-1 partitions.**
- Nécessité de se munir d'une **métrique** (distance euclidienne, chi2…)
- Nécessité de fixer une règle pour agréger un individu et un groupe d'individus (ou bien 2 groupes d'individus) ⇨ **Le critère d'agrégation**

Algorithme :

1ère phase : Initialisation de l'algorithme.

- Les classes initiales = n singletons individus.
- Calcul de la matrice des distances des individus 2 à 2

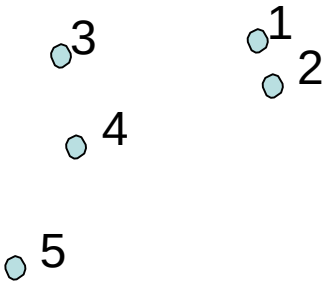
2ème phase : Itération des étapes suivantes.

- Regrouper les 2 éléments (individus ou groupes) les plus proches au sens d'un critère choisi.
- Mise à jour du tableau des distances en remplaçant les deux éléments regroupés par le nouveau et en recalculant sa distance avec les autres classes.

Fin de l'itération : agrégation de tous les individus en une seule classe.

Exemple :

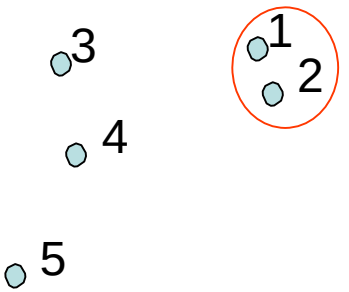
Etape 1 : **n individus / n classes**

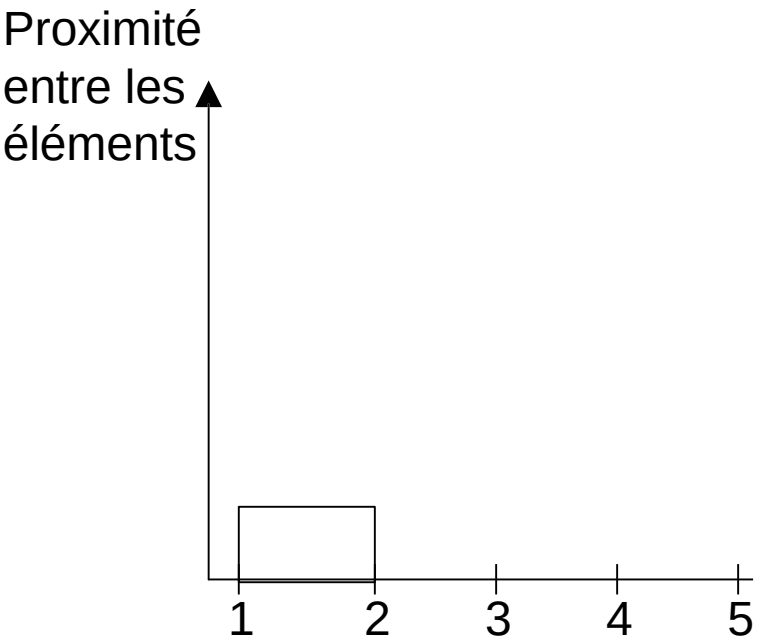


On construit la matrice de distance entre les n éléments et on regroupe les 2 éléments les plus proches.

Exemple :

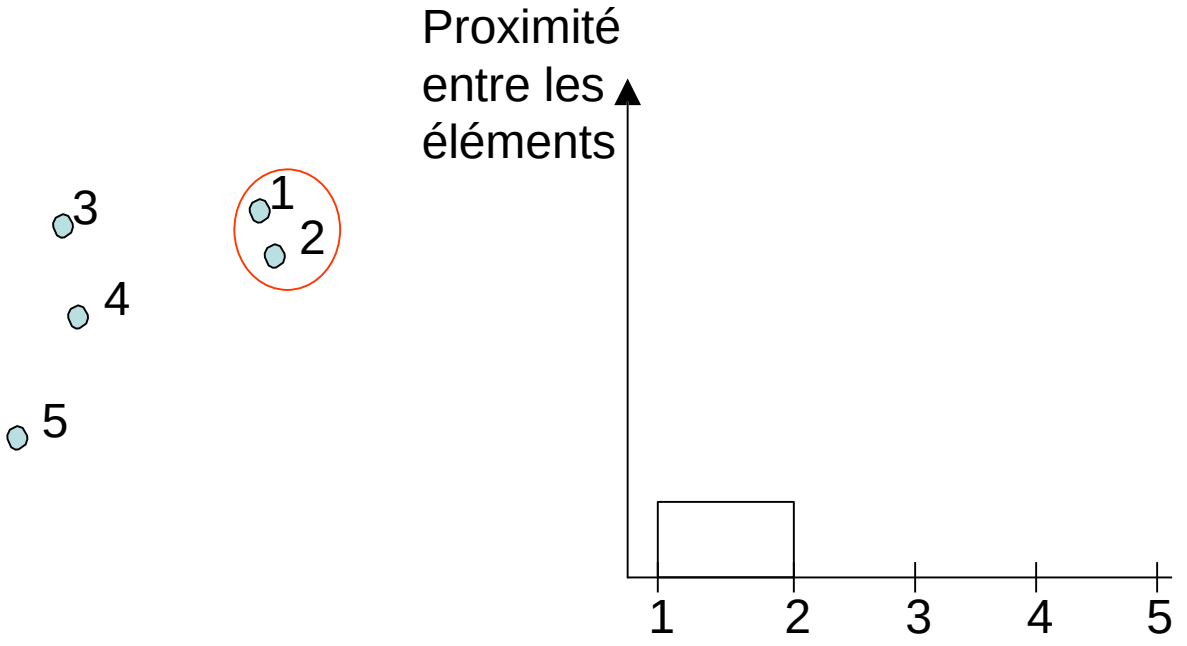
Etape 2 : **n - 1 classes**





Exemple :

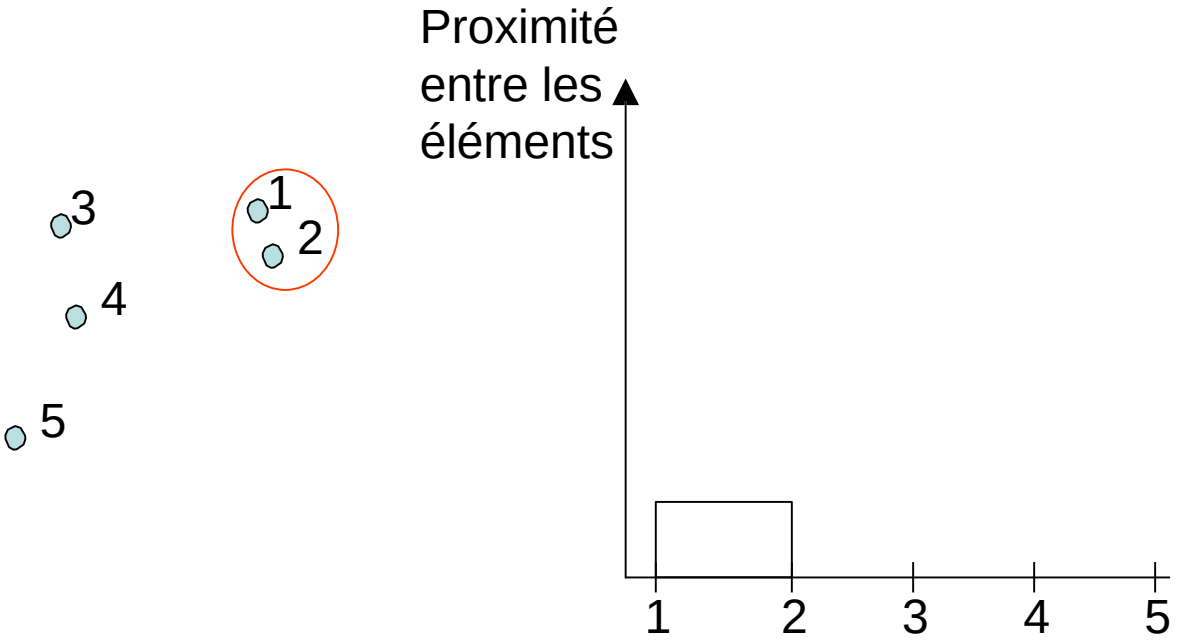
Etape 2 : **n - 1 classes**



Comment mesurer la distance entre une classe et un élément individuel ?

Exemple :

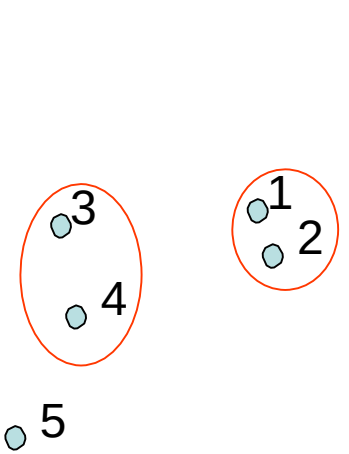
Etape 2 : **n - 1 classes**

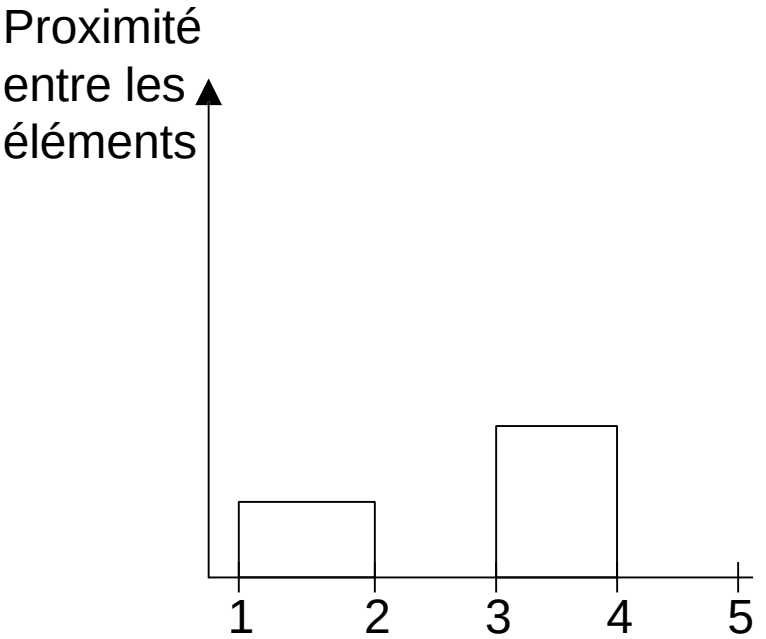


On utilise un critère d'agrégation pour mesurer la distance entre les éléments individuels et les classes déjà formées.

Exemple :

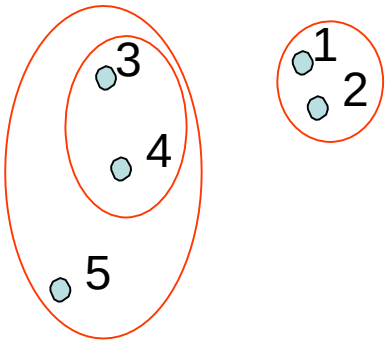
Etape 3 : **n - 2 classes**

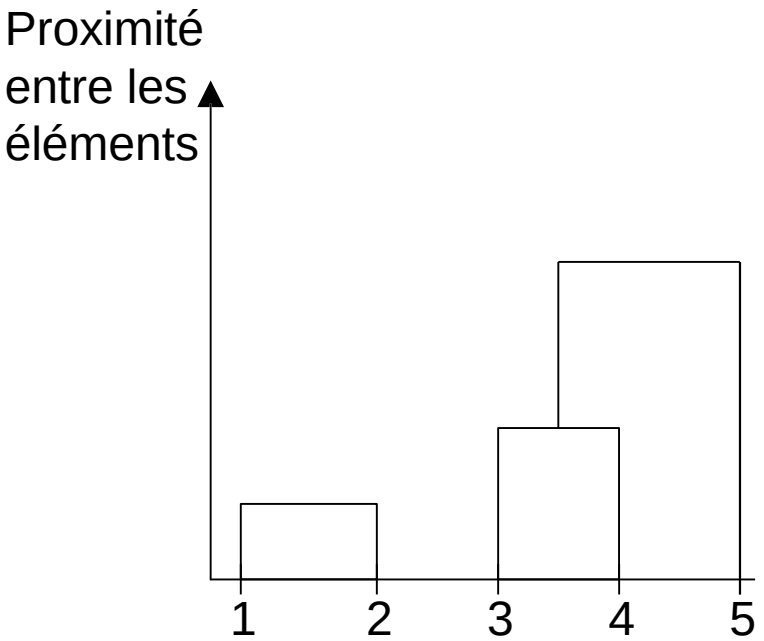




Exemple :

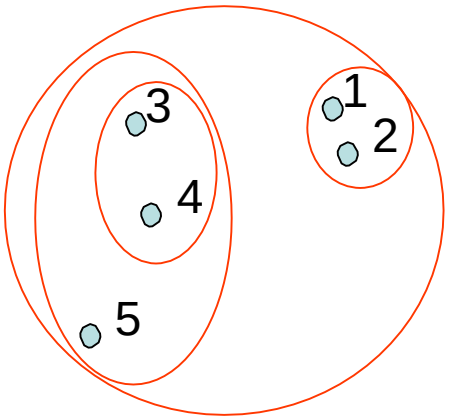
Etape 4 : **n - 3 classes**

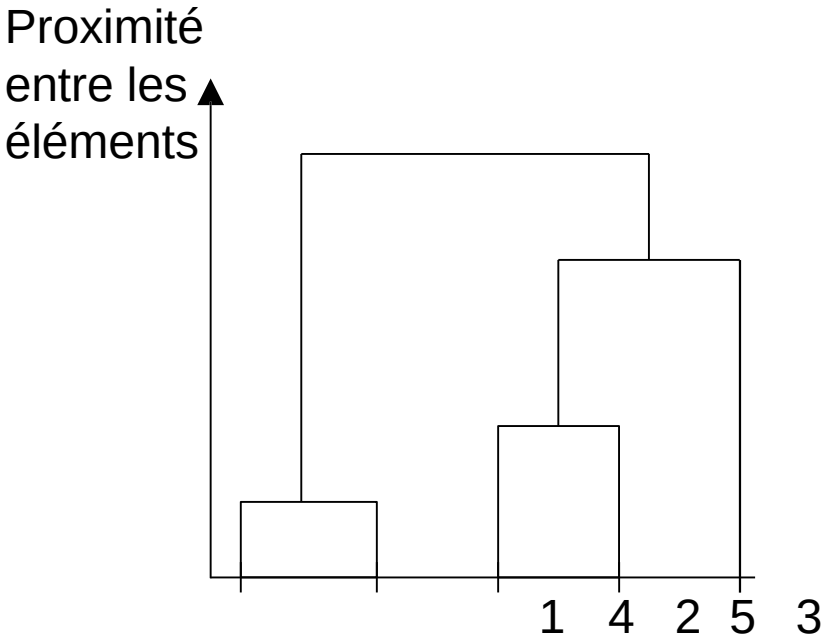




Exemple :

Etape 5 : **n - 4 = 1 classe**

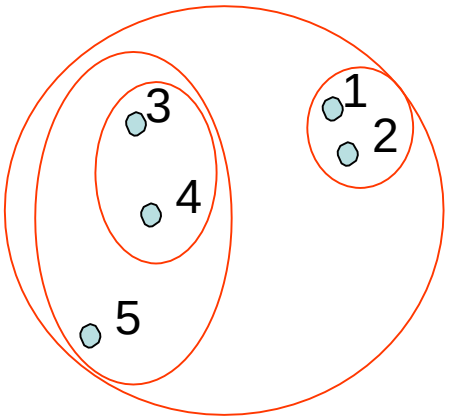


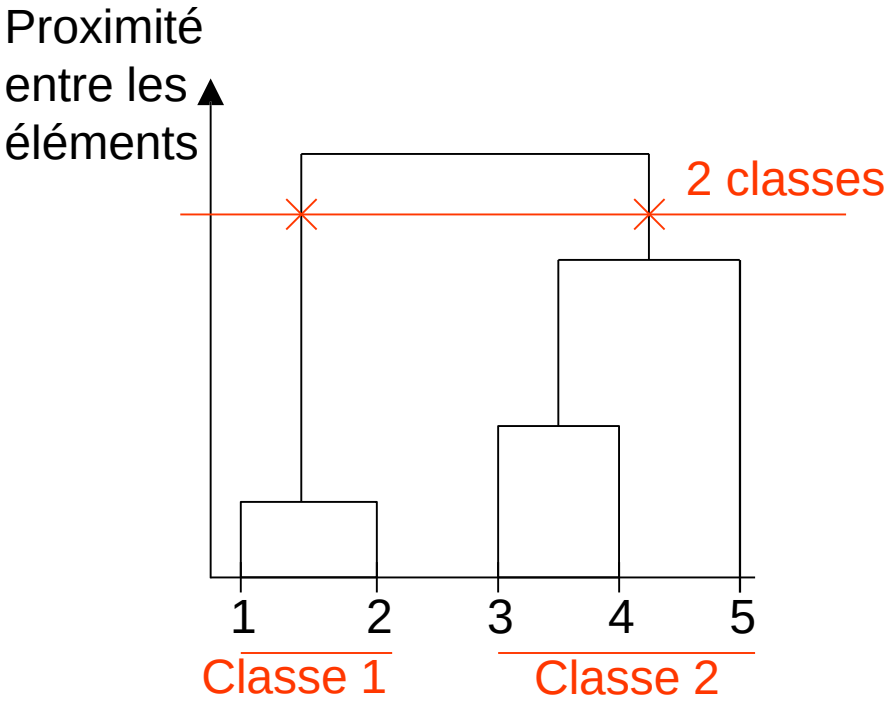


L'historique de la classification est représentée sur l'arbre hiérarchique (dendrogrogramme)

Exemple :

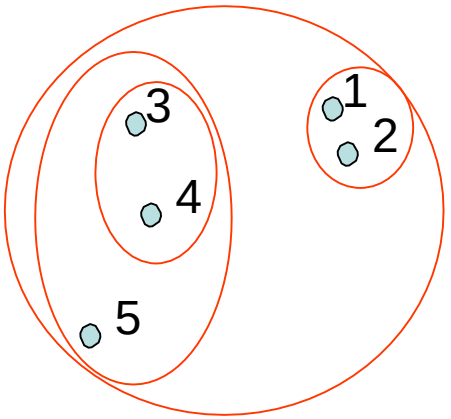
Etape 5 : **n - 4 = 1 classe**

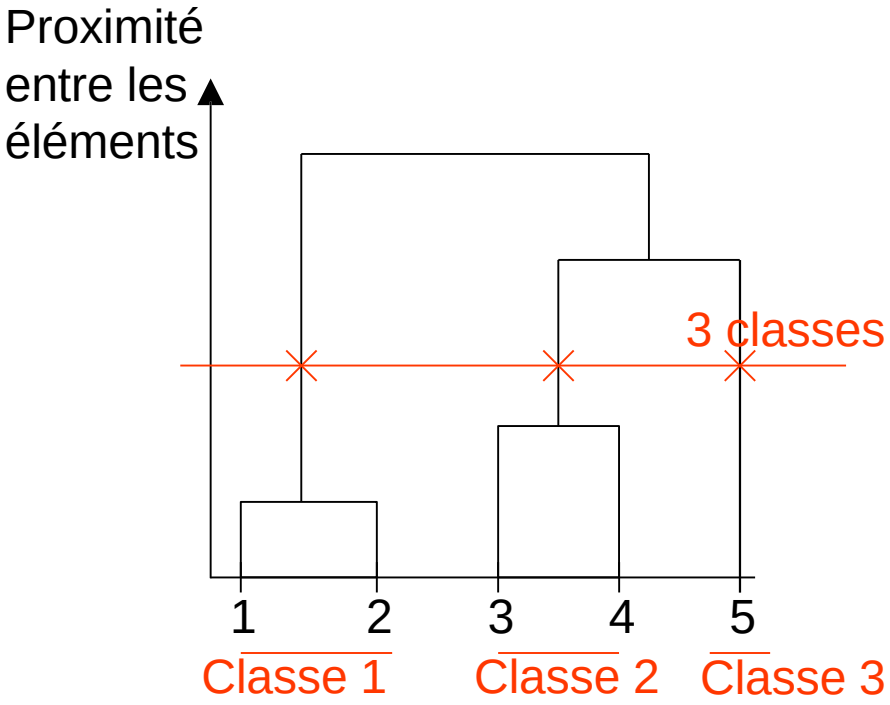




Exemple :

Etape 5 : **n – 4 = 1 classe**





Le choix du nombre de classe est déterminé a posteriori

Réflexions pré-algorithme

- Nécessité de définir une distance entre les individus
- Définir un critère de regroupement des individus à minimiser aussi appelé stratégie ou critère d'agrégation.
- Stratégie pour définir la meilleure typologie finale.

Indice de dissimilarité

- Le choix de la mesure de distance entre individus dépend des données étudiées et des objectifs.
- Exemples :
  - Distance Euclidienne : le type de distance le plus couramment utilisé. Il s'agit d'une distance géométrique dans un espace multidimensionnel.
  - Distance Euclidienne au carré : Permet de "sur-pondérer" les objets atypiques (éloignés), en élevant la distance euclidienne au carré.

Critère d’agrégation

On regroupe les éléments en minimisant l’indice d’agrégation. Plusieurs méthodes ou critères peuvent être utilisés:

- Critères des centres de gravité.
- Distance minimale.
- Distance maximale.
- Critère de ward (la méthode la plus connu).

Critère d'agrégation

la **méthode de Ward** est un algorithme permettant de regrouper deux classes d'une partition pour obtenir une partition plus agrégée.

si $G = \{e_i : i = \{1:n\}\}$ est un groupe d'individus, de centre de gravité *g*, partitionné en k classes d'effectifs $n_1, n_2, .., n_k$ qu'on appellera $G_1, G_2, .., G_k$ qui ont pour centres de gravité $g_1, g_2, .., g_k$ alors :

L'inertie totale du nuage est égale à : $I_t = \frac{1}{n}\sum_{i=1}^{n} d(e_i, g)^2$

L'inertie interclasse est égale à : $I_e = \frac{1}{n}\sum_{i=1}^{k} n_i \times d(g_i, g)^2$

L'inertie intraclasse est égale à : $I_a = \frac{1}{n}\sum_{i=1}^{k}\sum_{j=1}^{n_i} d(e_j, g_i)^2$

Ward (explication 1):

Considérons l'évolution de l'inertie intra-classe au fur et à mesure de la classification:

A l'initialisation, toutes les classes sont composées d'une unique observation. Chaque classe est donc parfaitement homogène, et l'inertie intra-classe est nulle.

à la dernière étape de l'algorithme, toutes les observations sont regroupées pour former une unique classe. L'inertie inter-classe est alors nulle, et l'inertie intra-classe est maximum.

Ward (explication 2):

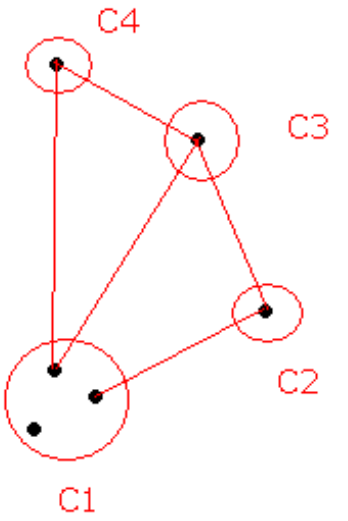
Il n'est par ailleurs pas difficile de constater qu'à chaque étape de la classification, l'inertie intra-classe augmente alors que l'inertie inter-classe diminue, car chaque étape fusion fait perdre de l'homogénéité aux deux classes fusionnées.

L'objectif étant d'aboutir à une partition en K classes d'inertie intra-classe minimum, la stratégie va consister à regrouper à chaque étape les deux classes dont la fusion entraine le plus faible gain d'inertie intra-classe (ou de manière équivalente la plus faible perte d'inertie inter-classe).

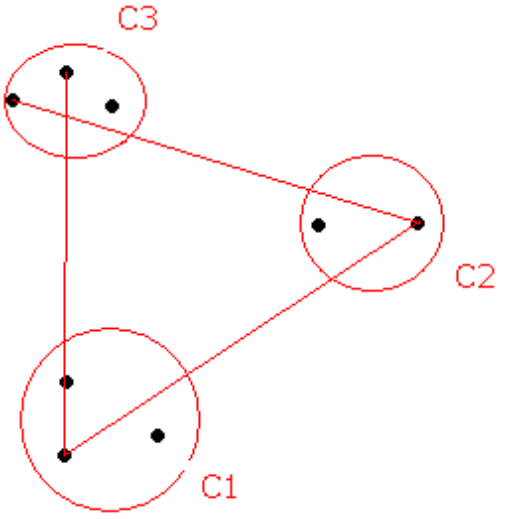
Critère d'agrégation

autres stratégies :

- stratégie du saut minimum ou single linkage : On regroupe les 2 éléments présentant la plus **petite** distance entre éléments des deux classes.



- stratégie du saut maximum ou du diamètre ou complete linkage : On regroupe les 2 éléments présentant la plus **grande** distance entre éléments des deux classes.



Choix de la partition finale

Par construction, l'algorithme CAH fournit une classification des données en K classes, pour tout K entre 1 et n. Ainsi, il n'est pas nécessaire de préciser a priori le nombre de classes que l'on souhaite. Ce choix peut être réalisé a posteriori, en considérant le dendrogramme. Les hauteurs des branches étant proportionnelles à la distance entre classes, on peut choisir une classification en "coupant" l'arborescence lorsque les branches sont jugées trop grandes. En effet, une grande branche indique que l'on regroupe des classes qui ne sont pas homogènes.

- l'utilisateur doit repérer des sauts extrêmement importants.
- Si ces sauts concernent les k derniers nœuds de l'arbre, alors un découpage en (k+1) classes sera pertinent.

CAH: avantages & inconvénients

+ Le principal avantage de la CAH par rapport aux autres méthodes de classification réside dans cette représentation sous forme d'arbre qui met en évidence une information supplémentaire : l'augmentation de la dispersion dans un groupe produit par une agrégation. L'utilisateur peut dès lors avoir une idée du nombre adéquat de classes en choisissant la partition correspondant au saut le plus élevé dans l'augmentation de la dispersion au sein des classes.
- Le principal inconvénient de la CAH est qu'elle nécessite le calcul des distances entre individus pris deux à deux. Ce qui est très rapidement prohibitif dès que la taille du fichier excède le millier d'individus.
- Résultats différents en fonction de la paramétrisation (Distances différentes, Choix d'agrégation différents).
- Les regroupements sont définitifs, ce qui ne permet pas d'optimisation postérieure au *clustering.*

Comparaison CAH / K-means

- Le principal avantage de la CAH vis-à-vis des K-means réside dans sa stabilité. Contrairement aux K-means, la CAH ne nécessite aucune initialisation, en conséquence lancer deux fois l'algorithme CAH sur le même jeu de données donnera deux fois le même résultat.

- Choix du nombre de classes : ???
  - Il arrive que ce nombre soit directement fixé par la nature.
  - Toutefois, dans la plupart des cas, ce nombre est inconnu.

Comparaison CAH / K-means

Choix du nombre de classes :

- L'algorithme CAH appliqué avec la distance de Ward, il est possible de tracer la courbe de l'inertie intra-classe en fonction de K. On cherche alors à identifier les étapes où l'on observe une rupture dans cette courbe, synonyme d'une forte dégradation de l'inertie intra-classe. Cette dégradation résulte de la forte hétérogénéité des deux classes réunies lors de l'étape considérée, il est alors naturel de considérer un nombre de classes supérieur à celui pour lequel la rupture a lieu. Cette stratégie, parfois dénommée "critère du coude", donne des résultats satisfaisants.
- Lorsqu'appliquée à l'algorithme K-means, l'identification des ruptures peut s'avérer plus difficile. Une méthode pragmatique pour le choix du nombre de classes est de choisir une partition dont il sera possible d'interpréter les classes.

Utilisation jointe des K-means et de la CAH :

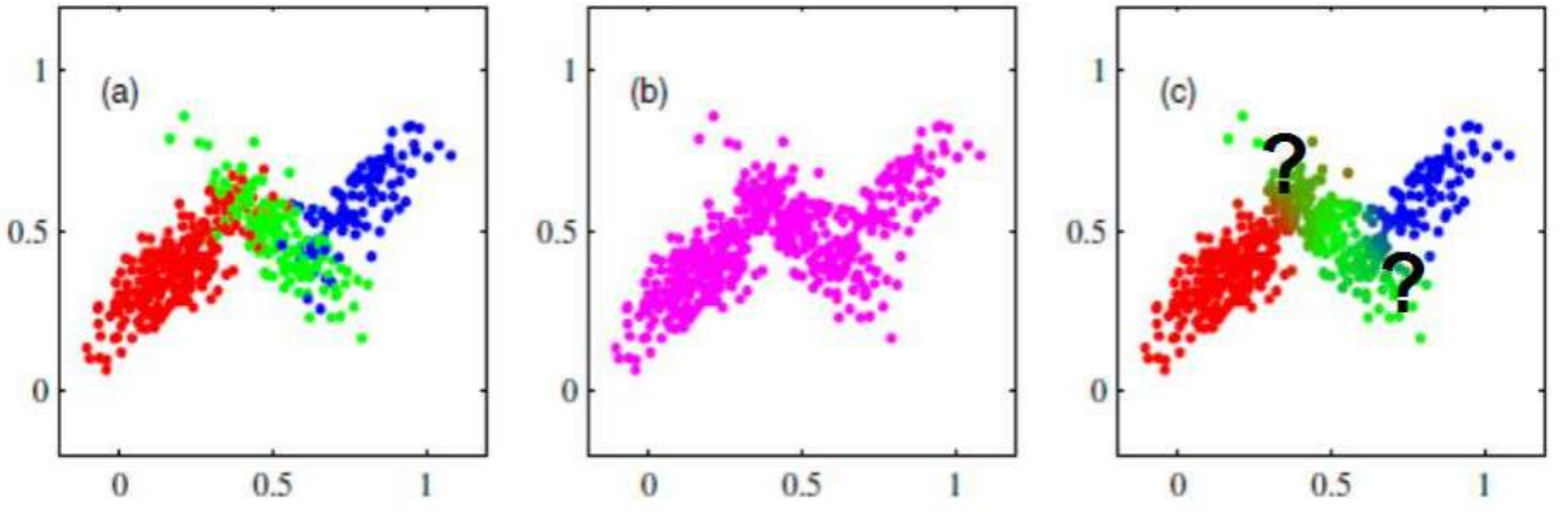
Il existe des cas où il peut être utile d'utiliser les deux algorithmes conjointement.

- **Le premier cas** est celui des grands jeux de données. Les premières étapes du CAH sont les plus couteuses, puisqu'elles nécessitent un grand nombre de calculs de distance. On peut alors réduire le nombre d'individus initial en utilisant les K-means, par exemple pour passer de 1.000.000 d'individus à 10.000 classes, puis réaliser la CAH sur les classes obtenues.

Utilisation jointe des K-means et de la CAH :

- **Le deuxième cas** consiste à utiliser les K-means après la CAH : on constate qu'avec la CAH le classement de deux individus dans une même classe n'est plus remis en cause lors des étapes suivantes. Il peut alors être utile de réaliser la CAH, puis de permettre quelques réallocations des individus en faisant tourner l'algorithme des K-means en partant de la partition obtenue par CAH.

Mélange de gaussiennes



GMM: Gaussian Mixture Model

Mélange de gaussiennes

Motivation :

- estimation de densité : apprendre la loi de probabilité ayant généré les données
- pour générer de nouvelles données réalistes.
- pour distinguer les «vrais» données des «fausses» données (ex: *spam filtering)*
- partitionnement de données / *clustering*

Mélange de gaussiennes

- Un mélange de gaussiennes **suppose** que les données ont été générées comme suit :

  pour $n = 1 \ldots N$

  - choisit un entier $k \in \{1,\ldots,K\}$ selons probabilités $\pi_1, \ldots, \pi_K$
  - génère $\mathbf{x}_n$ d'une loi de la loi de probabilité $\mathcal{N}(\mathbf{x}_n|\mu_k, \mathbf{\Sigma}_k)$

- les entrées sont des échantillons d'une de *K différentes* lois gaussiennes, ayant chacune des moyennes et covariances différentes.

Mélange de gaussiennes

Probabilité à priori :

- On va noter **z** la variable aléatoire correspondant à l'identité de la gaussienne qui a généré une entrée **x**
  - format *one-hot : z_k=1 si* ***x a été générée par la k gaussienne***
- La probabilité de choisir la k^e gaussienne est donc :

$$p(z_k = 1) = \pi_k$$

- on peut aussi écrire (si Z est un vecteur) : $p(\mathbf{z}) = \prod_{k=1}^{K} \pi_k^{z_k}$

Mélange de gaussiennes

Fonction de densité conditionnelle de l'entrée :

- Sachant $\mathbf{z}$ la probabilité (fonction de densité) de $\mathbf{x}$ est

$$p(\mathbf{x}|z_k = 1) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

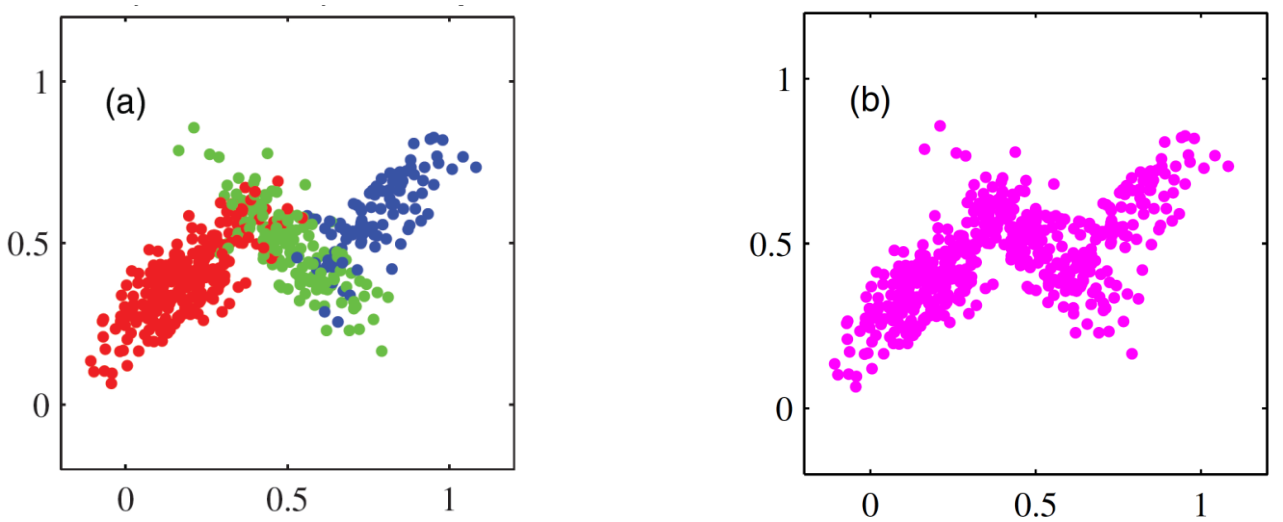
$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\}$$

qu'on peut aussi écrire

$$p(\mathbf{x}|\mathbf{z}) = \prod_{k=1}^{K} \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_k}$$

Mélange de gaussiennes

Dans un mélange de gaussienne, l'appartenance aux K gaussiennes («classes») n'est pas connue :

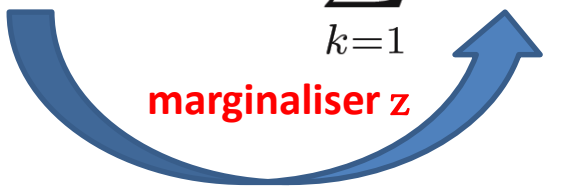


c'est à l'algorithme d'apprentissage de le découvrir en faisant de l'entrainement !!!

Mélange de gaussiennes

Puisqu'on ne connait pas l'appartenance aux gaussiennes z, on va s'intéresser à la probabilité marginale :

$$p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{z})p(\mathbf{x}|\mathbf{z}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$



- C'est ce qu'on appelle un mélange de distributions : c'est la somme des distributions des K populations pondérées par la taille de ces populations π_k (la proportion d'individus appartenant à ces populations).
- C'est de cette façon qu'on va mesurer la performance de notre modèle.

Mélange de gaussiennes

$$p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{z})p(\mathbf{x}|\mathbf{z}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- Les variables mises en jeu dans un modèle de mélange sont donc les X et Z, représentées sous forme d'un couple Y = (X,Z), que l'on appelle données complètes. Cependant, seules les X sont observées et sont appelées données incomplètes. On souhaite donc reconstruire les variables d'appartenance Zk.
- Par exemple sur de nouvelles données (si je veux savoir si c'est un bon modèle) je prends un ensemble de test (respect de validation) et calculer P(x) de chacun de mes exemples, si cette valeur est élevé c'est un bon modèle, sinon (c-a-d, il ne généralise pas bien), il juste capable de reproduire des données d'apprentissage.

Mélange de gaussiennes

Partitionnement de données, clustering :

- À partir d'un mélange de gaussiennes entraîné, on pourrait inférer à quelle gaussienne appartiennent les entrées.
  - on pourrait alors automatiquement catégoriser nos données en fonction des probabilités d'appartenance à chacune des gaussiennes

- Pour classer les individus, on utilise la règle du Maximum A Posteriori (MAP) : on classe l'échantillon t dans le cluster k correspondant à la probabilité maximale.

Mélange de gaussiennes

Affectation :

- À l'aide de la règle de Bayes, on obtient la **probabilité d'appartenance à la *k*^{eme} *gaussienne suivante :***

$$\gamma(z_k) \equiv p(z_k = 1|\mathbf{x}) = \frac{p(z_k = 1)p(\mathbf{x}|z_k = 1)}{\sum_{j=1}^{K} p(z_j = 1)p(\mathbf{x}|z_j = 1)}$$

$$= \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

Mélange de gaussiennes

Apprentissage :

- On va entraîner un mélange de gaussiennes par maximum de vraisemblance

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

  - on va maximiser la (log-)vraisemblance marginale des données d'entraînement

Mélange de gaussiennes

Apprentissage :

- On va entraîner un mélange de gaussiennes par maximum de vraisemblance

$$\underbrace{\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})}_{\text{probabilité de tous les } \mathbf{x}_n} = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

  - on va maximiser la (log-)vraisemblance marginale des données d'entraînement

Mélange de gaussiennes

Apprentissage :

- On va entraîner un mélange de gaussiennes par maximum de vraisemblance

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

- Cas $\boldsymbol{\mu}_k$: la solution doit satisfaire

$$0 = -\sum_{n=1}^{N} \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \boldsymbol{\Sigma}_k(\mathbf{x}_n - \boldsymbol{\mu}_k)$$

Mélange de gaussiennes

Apprentissage :

- On va entraîner un mélange de gaussiennes par maximum de vraisemblance

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

- Cas $\boldsymbol{\mu}_k$: la solution doit satisfaire

$$0 = -\sum_{n=1}^{N} \qquad \gamma(z_{nk}) \qquad\qquad \boldsymbol{\Sigma}_k(\mathbf{x}_n - \boldsymbol{\mu}_k)$$

Mélange de gaussiennes

Apprentissage :

- On va entraîner un mélange de gaussiennes par maximum de vraisemblance

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

- Cas $\boldsymbol{\mu}_k$: si on suppose que les $\gamma(z_{nk})$ sont fixes

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n \ , \qquad \text{où} \ \ N_k = \sum_{n=1}^{N} \gamma(z_{nk})$$

Mélange de gaussiennes

Apprentissage :

- On va entraîner un mélange de gaussiennes par maximum de vraisemblance

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

- Cas π_k : on utilise un multiplicateur de Lagrange

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) + \lambda \left(\sum_{k=1}^{K} \pi_k - 1 \right)$$

Mélange de gaussiennes

Apprentissage :

- On va entraîner un mélange de gaussiennes par maximum de vraisemblance

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

- Cas π_k : la solution doit satisfaire

$$0 = \sum_{n=1}^{N} \frac{\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} + \lambda \quad , \forall k \quad \text{et} \quad \sum_{k=1}^{K} \pi_k = 1$$

Mélange de gaussiennes

Apprentissage :

- On va entraîner un mélange de gaussiennes par maximum de vraisemblance

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

- Cas π_k : si on suppose que les $\gamma(z_{nk})$ sont fixes

$$\pi_k = \frac{N_k}{N}$$

Mélange de gaussiennes

Apprentissage :

- On va entraîner un mélange de gaussiennes par maximum de vraisemblance

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

- Cas $\boldsymbol{\Sigma}_k$: si on suppose que les $\gamma(z_{nk})$ sont fixes

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^{\mathrm{T}}$$

Mélange de gaussiennes

Apprentissage :

- Les solutions pour $\boldsymbol{\mu}_k, \pi_k, \boldsymbol{\Sigma}_k$ supposent que les $\gamma(z_{nk})$ sont fixes
  - par contre, changer $\boldsymbol{\mu}_k, \pi_k$ et $\boldsymbol{\Sigma}_k$ va changer $\gamma(z_{nk})$
  - la supposition que les $\gamma(z_{nk})$ ne changeront pas est donc fausse

- Solution : on alterne entre calculer $\gamma(z_{nk})$ et $\boldsymbol{\mu}_k, \pi_k, \boldsymbol{\Sigma}_k$
  1. Étape **Estimation** : calcul de tous les $\gamma(z_{nk})$
  2. Étape **Maximisation** : calcul des $\boldsymbol{\mu}_k, \pi_k$ et $\boldsymbol{\Sigma}_k$
- On appelle cela l'**algorithme EM**

Mélange de gaussiennes

Apprentissage : Dempster, Laird and Rubin (1977)

- Pseudocode

1. Initialize the means $\boldsymbol{\mu}_k$, covariances $\boldsymbol{\Sigma}_k$ and mixing coefficients π_k, and evaluate the initial value of the log likelihood.

Mélange de gaussiennes

Apprentissage :

- Pseudocode

2. **E step**. Evaluate the responsibilities using the current parameter values

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

Mélange de gaussiennes

Apprentissage :

- Pseudocode

3. **M step**. Re-estimate the parameters using the current responsibilities

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n$$

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \left(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}\right) \left(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}\right)^{\mathrm{T}}$$

$$\pi_k^{\text{new}} = \frac{N_k}{N}$$

Mélange de gaussiennes

Apprentissage :

- Pseudocode

4. Evaluate the log likelihood

$$\ln p(\mathbf{X}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

and check for convergence of either the parameters or the log likelihood. If the convergence criterion is not satisfied return to step 2.

Mélange de gaussiennes

Apprentissage :

- Pseudocode

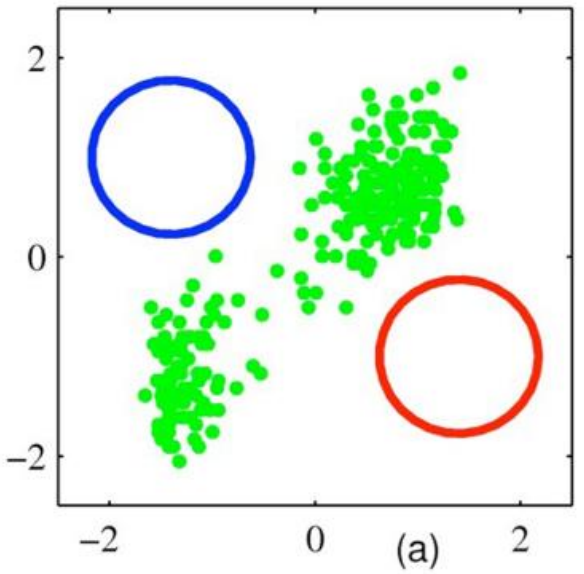
4. Evaluate the log likelihood **on a validation set**

$$\ln p(\mathbf{X}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

and check for convergence of either the parameters or the log likelihood. If the convergence criterion is not satisfied return to step 2.

Mélange de gaussiennes

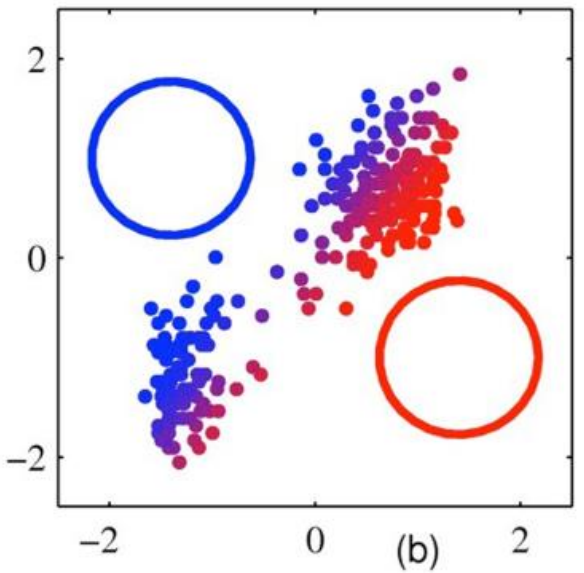
- Exemple d'exécution de l'algorithme EM



Initialisation

Mélange de gaussiennes

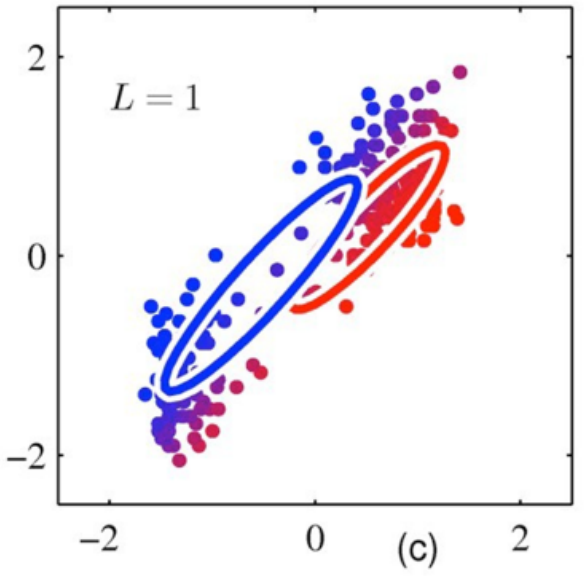
- Exemple d'exécution de l'algorithme EM



Étape E

Mélange de gaussiennes

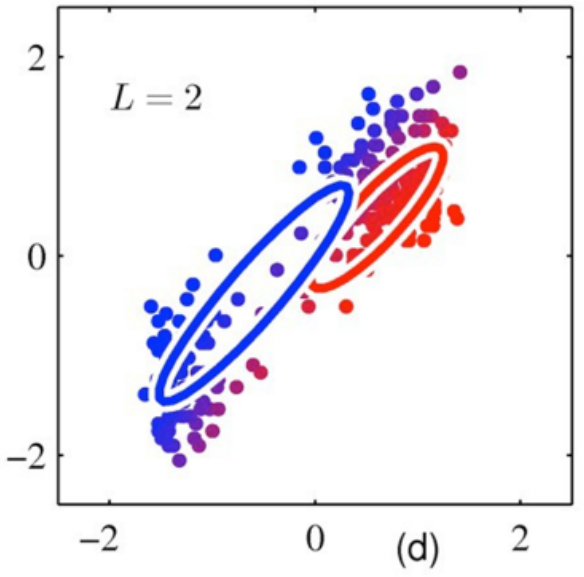
- Exemple d'exécution de l'algorithme EM



Étape M

Mélange de gaussiennes

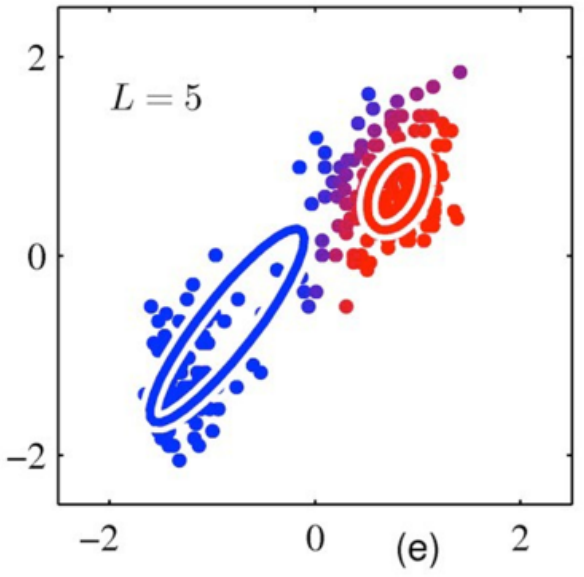
- Exemple d'exécution de l'algorithme EM



Après 2 itérations...

Mélange de gaussiennes

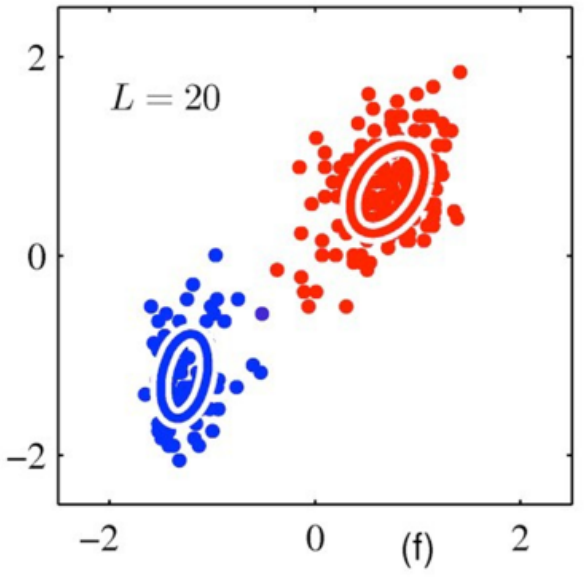
- Exemple d'exécution de l'algorithme EM



Après 5 itérations...

Mélange de gaussiennes

- Exemple d'exécution de l'algorithme EM



Après 20 itérations...

En pratique (1)

- Comme tous les algorithmes itératifs, l'algorithme EM nécessite:

  – l'initialisation des paramètres ou des probabilités a posteriori (dans le cadre particulier des modèles de mélange de distributions). En pratique, le plus simple est souvent de choisir ces probabilités. Si on ne dispose d'aucune information a priori, l'usage est de les choisir au hasard = 1/K.

  – une règle d'arrêt : on s'arrête quand les valeurs des paramètres ou de la quantité à maximiser entre deux itérations successives ne varie "presque" plus.

En pratique (2)

- Bien que cet algorithme soit très performant et souvent simple à mettre en œuvre, son principal problème est sa forte dépendance aux valeurs initiales : pour différentes initialisations, on obtient différentes valeurs des estimations des paramètres, ce qui le rend peu stable. La raison de cette sensibilité est que l'algorithme converge vers des maxima locaux de la vraisemblance.

- Pour s'affranchir de ce problème, une solution consiste :

En pratique (3)

– soit à lancer l'algorithme à partir de plusieurs valeurs initiales et à retenir la meilleure,

– soit à lancer l'algorithme un grand nombre de fois à partir de plusieurs valeurs initiales, à moyenner les valeurs obtenues pour les probabilités a posteriori ou pour les paramètres (suivant le mode d'initialisation choisi), puis à lancer une dernière fois l'algorithme EM en utilisant les valeurs moyennes comme valeurs initiales.

- Notons qu'une autre solution serait d'initialiser l'algorithme en utilisant les méthodes heuristiques présentées dans le chapitre précédent.

En pratique (4)

- On n’a pas de garanties qu’on va retrouver les «vraies» catégories ?

  1. les données de chaque catégorie ne sont peut-être pas gaussiennes
  2. le modèle de mélange entraîné n’est peut-être pas bon.

- Plus les données des différentes catégories seront bien séparées (pas entrelacées), meilleurs seront les résultats

Mélange de gaussiennes

Choix du nombre de populations :

- L'avantage du cadre probabiliste des modèles de mélange par rapport aux méthodes exploratoires présentées précédemment est que l'on dispose de critères théoriques pour choisir ce nombre. Ces critères sont appelés critères pénalisés.
  - le critère BIC (Bayesian Information Criterion).
  - le critère ICL (Integrate Classification Likelihood).

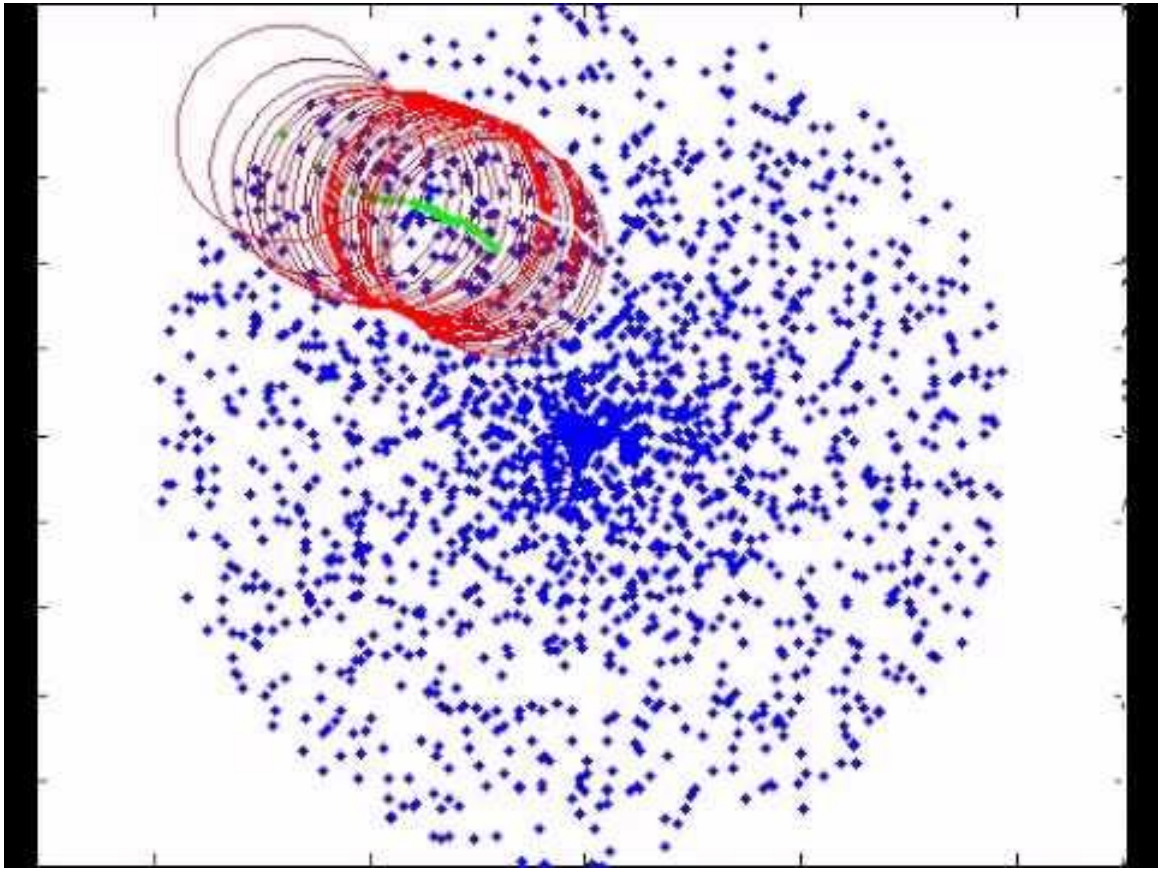
$$\text{Crit}(K) = \log(\text{vrais}) - \text{pen}(K)$$

Mélange de gaussiennes

Choix du nombre de populations :

- En pratique, on effectue la classification des données pour différentes valeurs de nombre de populations K. Pour chacune, on calcule l'ajustement (soit log-vraisemblance).
- Alors plus on considère de populations, mieux on s'ajuste ; Le problème est que plus on a de populations, plus il y a de paramètres à estimer et plus les erreurs d'estimation sont nombreuses.
- Par conséquent, on choisit le K qui maximise le critère pénalisé, c'est-à-dire qui mène au meilleur compromis entre un assez bon ajustement aux données et un nombre raisonnable de paramètres à estimer.

MEAN – SHIFT clustering



Mean-Shift : Algorithme

L'algorithme Mean-Shift est un algorithme itératif qui a pour objectif de faire converger un point vers le maximum local le plus proche :

1. On commence par choisir un point de départ P.
2. On cherche l'ensemble E des points qui sont dans le voisinage de P.
3. On déplace P vers la moyenne de E.
4. On réitère depuis l'étape 2 jusqu'à convergence.

Les déplacements successifs vers la moyenne font converger le point P vers les zones de fortes densités.

Mean-Shift : Algorithme

Le clustering Mean shift est un algorithme basé sur une fenêtre coulissante qui tente de trouver des zones denses de points de données (l'objectif est de localiser les points centraux de chaque classe).

Il fonctionne en mettant à jour les candidats aux points centraux pour qu'ils soient la moyenne des points à l'intérieur de la fenêtre coulissante.

Mean-Shift : Algorithme

Ces fenêtres candidates sont ensuite filtrées au cours d'une étape de post-traitement afin d'éliminer les doublons et de former l'ensemble final des points centraux et leurs groupes correspondants.

Mean-Shift : Algorithme

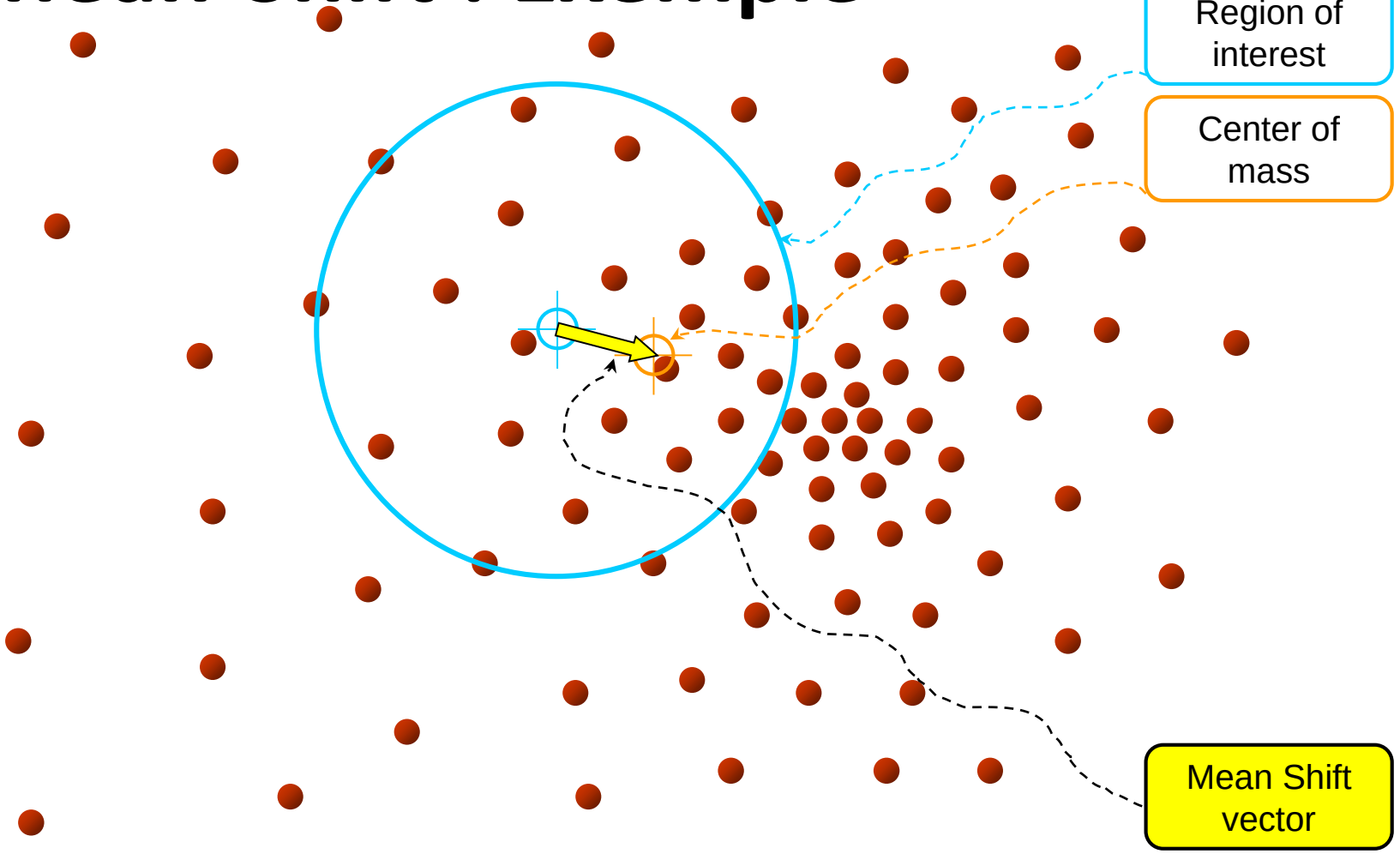
1. On commence par une fenêtre coulissante circulaire centrée en un point C (choisi au hasard) et dont le rayon r est le noyau.

2. L’ algorithme consiste à déplacer ce noyau de façon itérative vers une région à densité plus élevée(la moyenne des points de la fenêtre ;d'où le nom) à chaque étape jusqu'à la convergence.

Mean-Shift : Algorithme

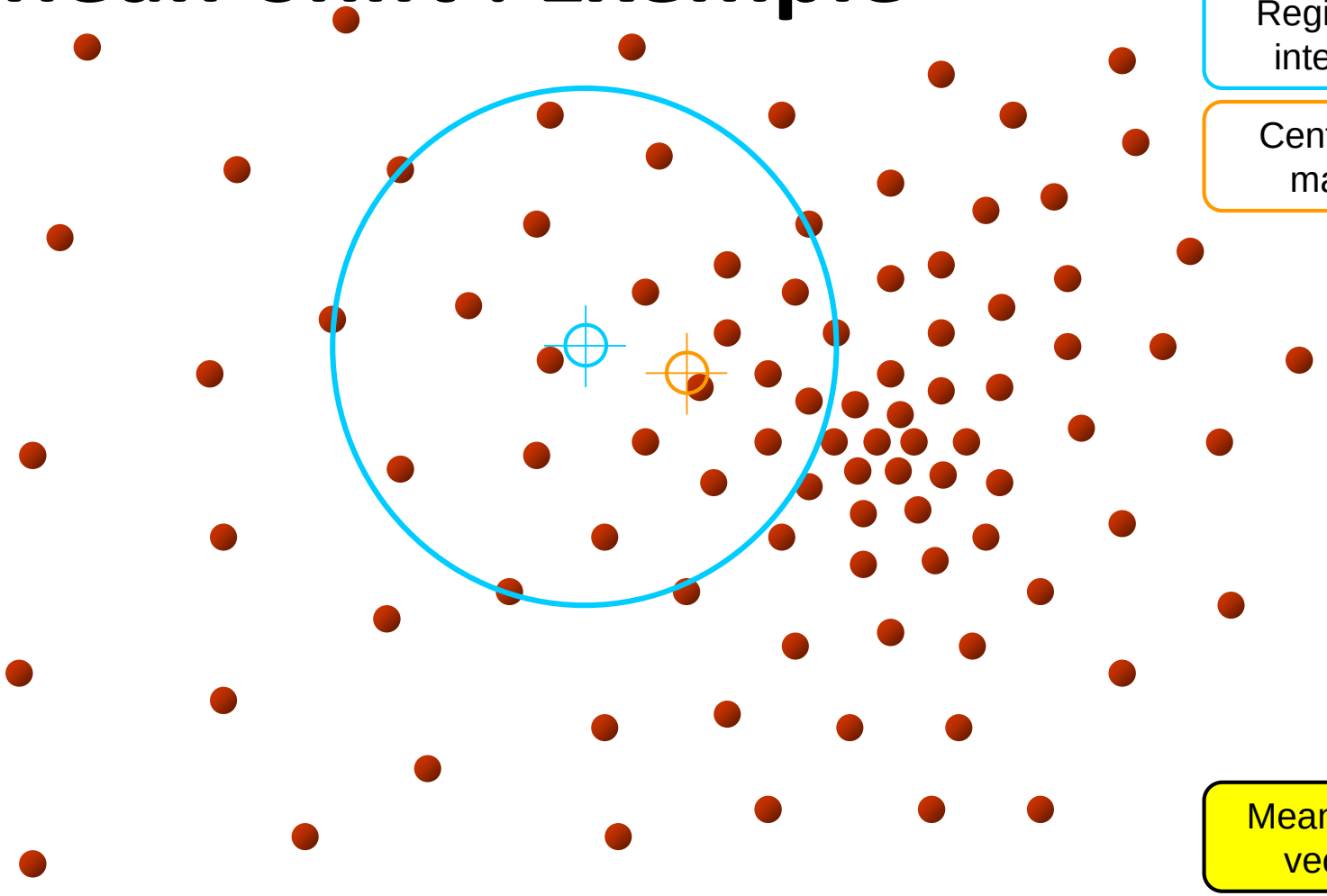
3. On continue à décaler la fenêtre coulissante en fonction de la moyenne jusqu'à ce qu'il n'y ait plus de direction à laquelle un déplacement pourra contenir plus de points.
4. Ce processus se déroule avec de multiples fenêtres coulissantes jusqu'à ce que tous les points se trouvent à l'intérieur d'une fenêtre. Lorsque plusieurs fenêtres coulissantes se chevauchent, la fenêtre contenant le plus grand nombre de points est conservée. Les données sont ensuite regroupées en fonction de la fenêtre coulissante dans laquelle ils se trouvent.

Mean-Shift : Exemple



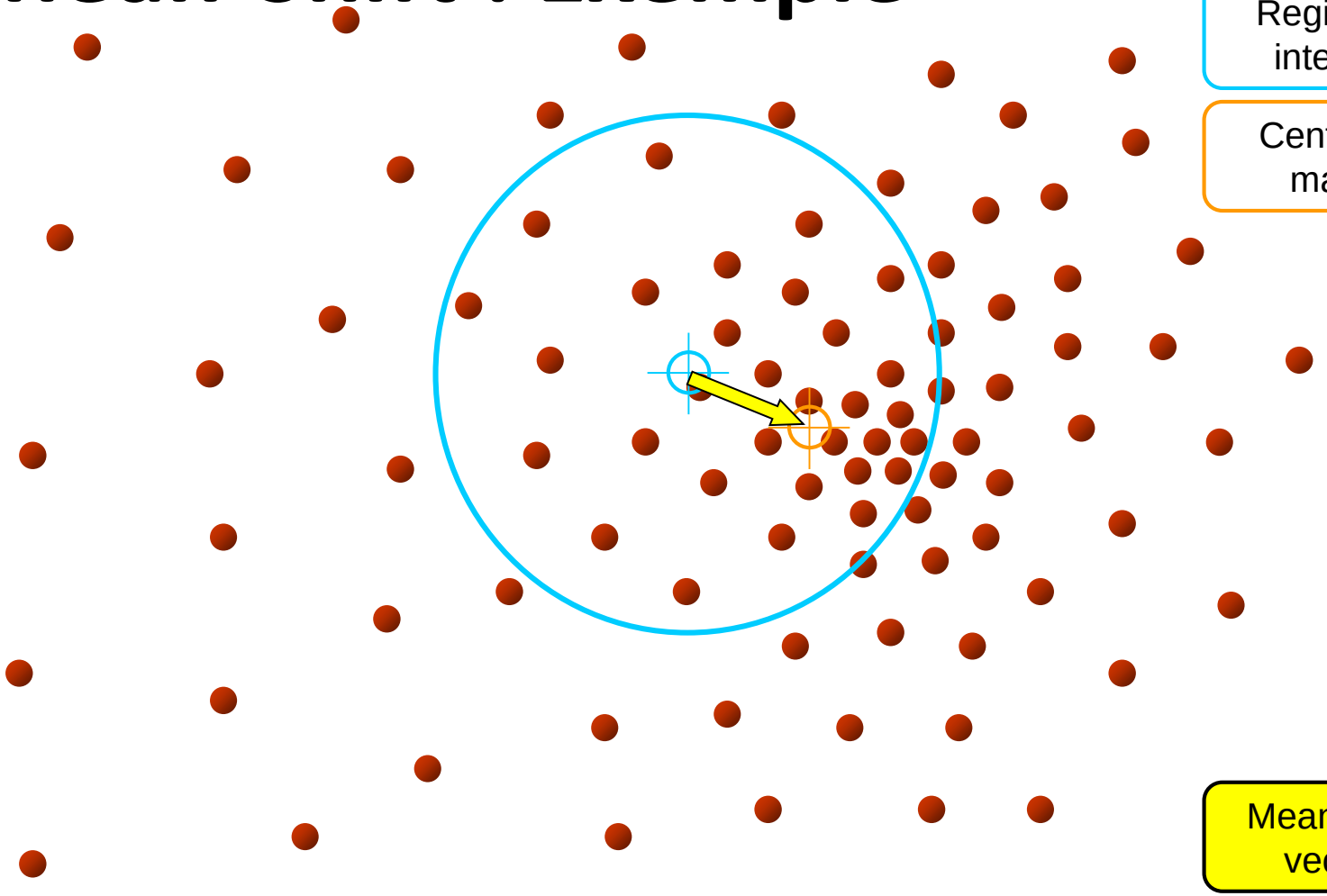
Objective : Find the densest region

Mean-Shift : Exemple



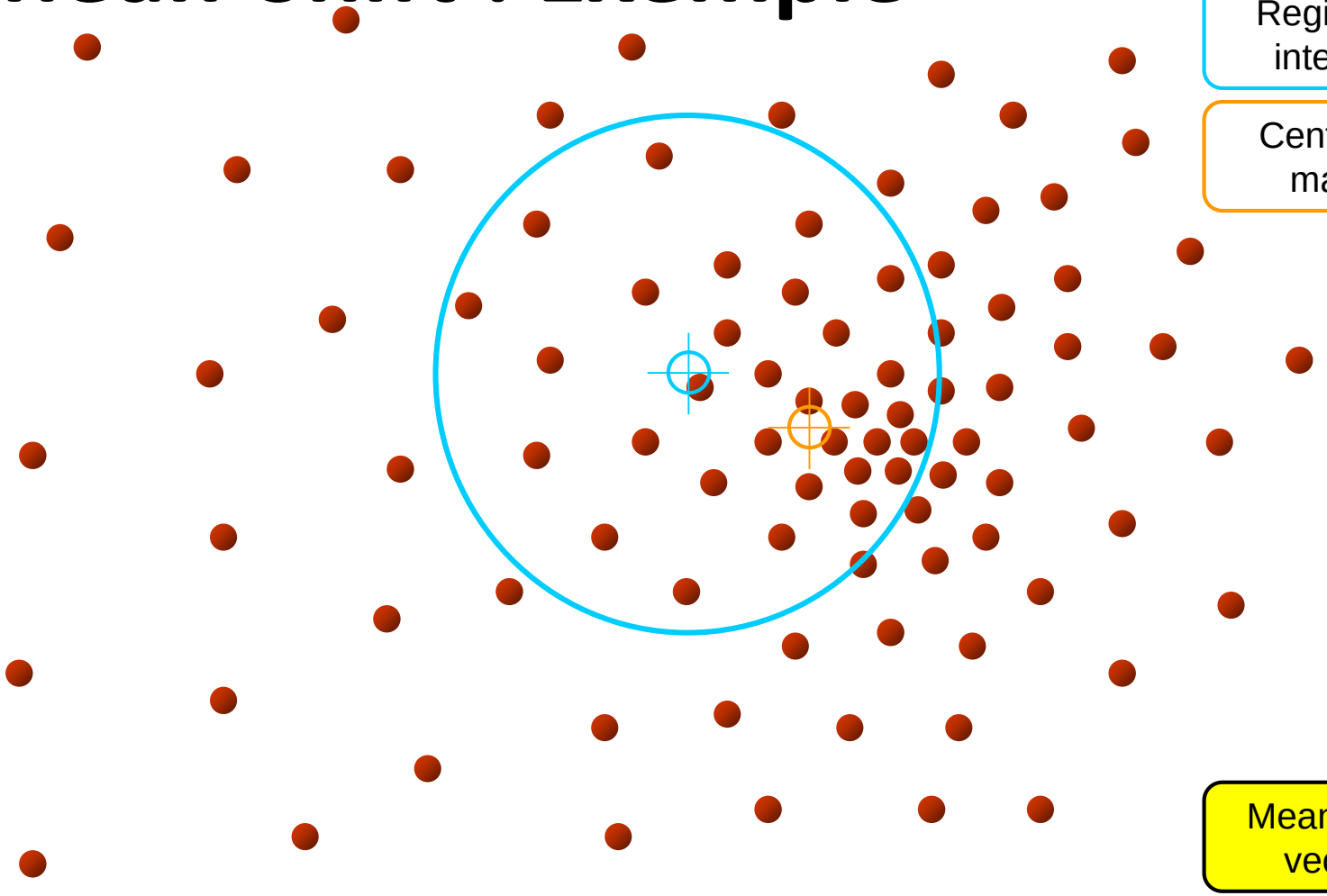
Objective : Find the densest region

Mean-Shift : Exemple



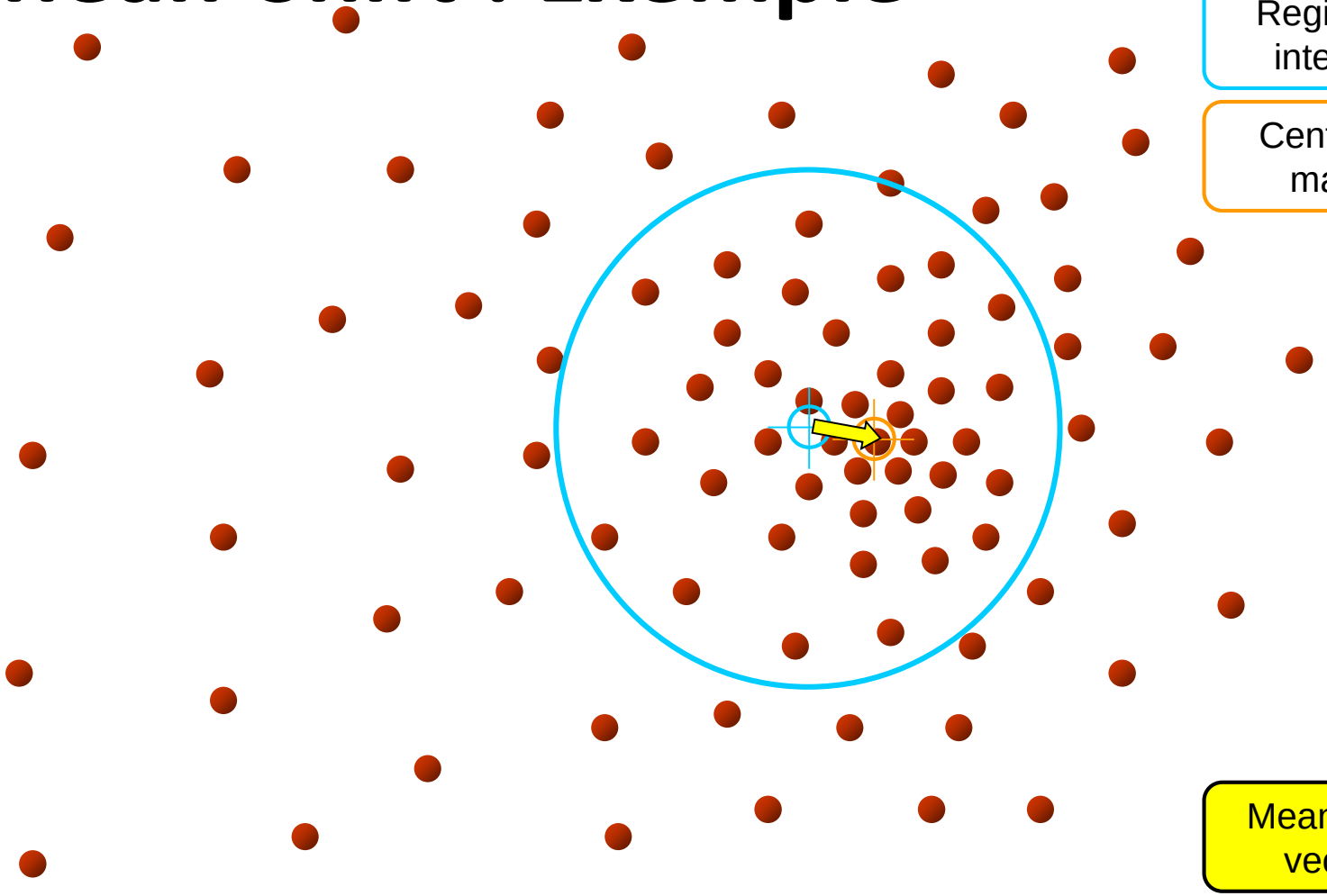
Objective : Find the densest region

Mean-Shift : Exemple



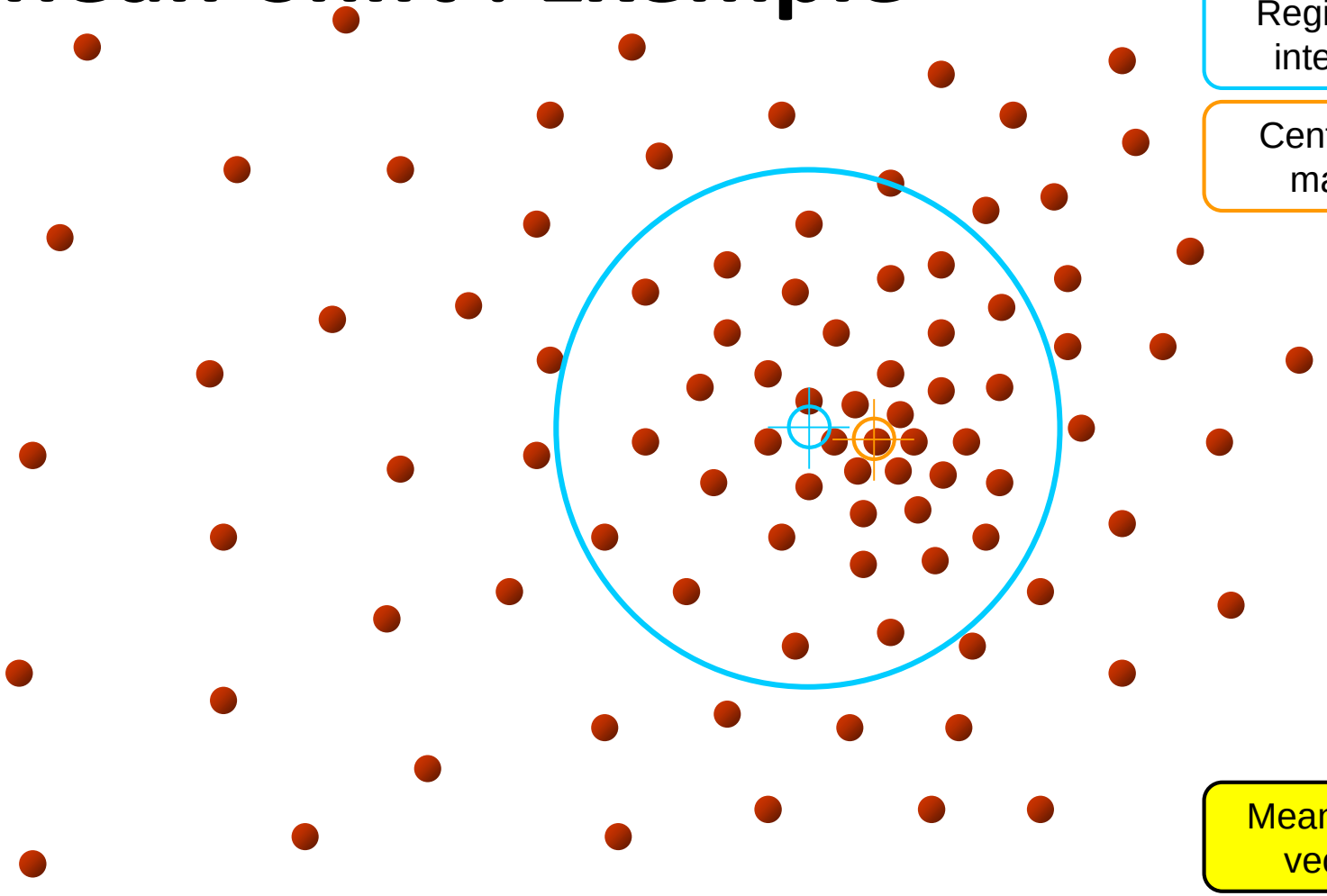
Objective : Find the densest region

Mean-Shift : Exemple



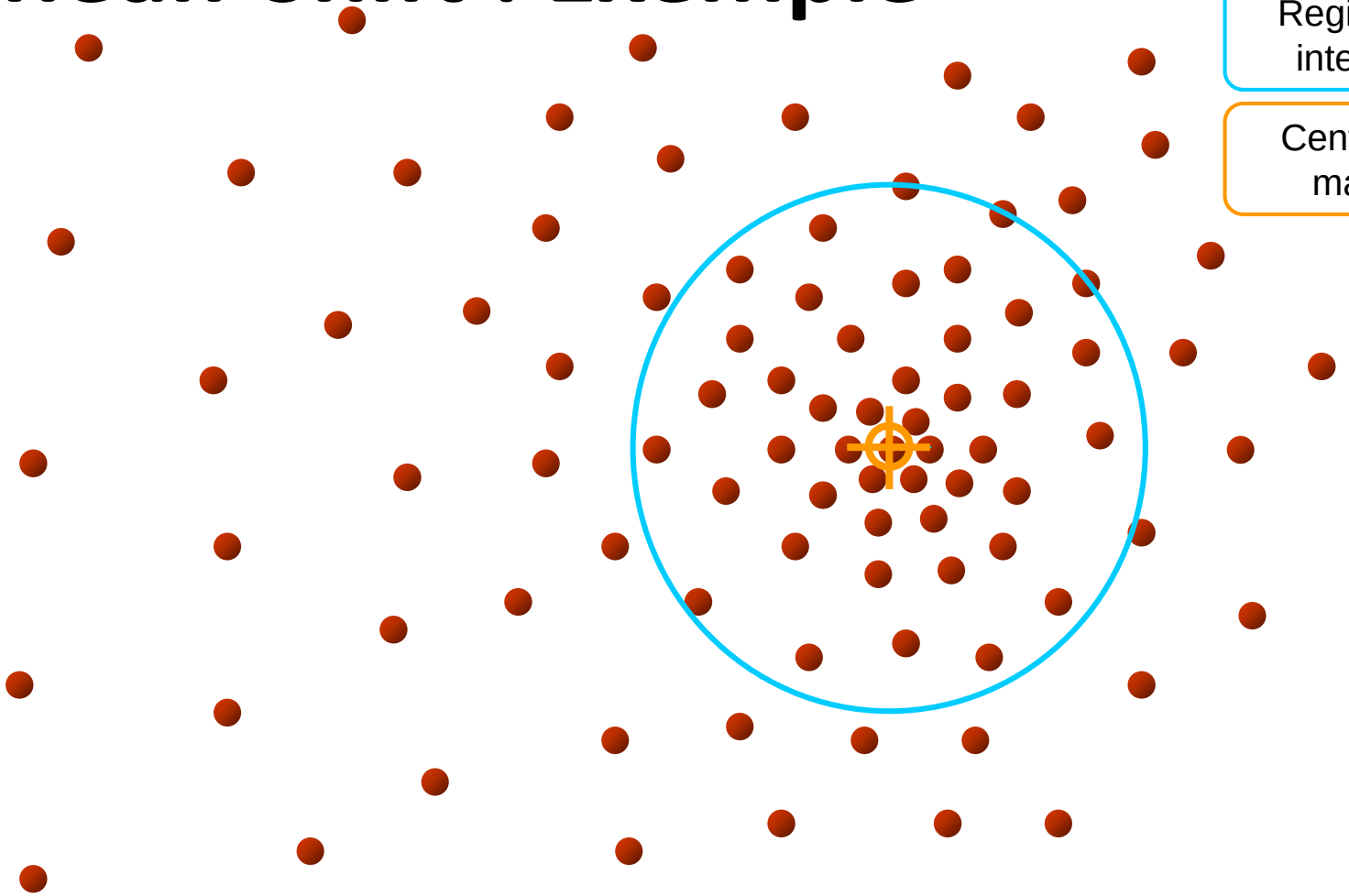
Objective : Find the densest region

Mean-Shift : Exemple



Objective : Find the densest region

Mean-Shift : Exemple



Objective : Find the densest region

Mean-Shift : En pratique

- L'algorithme de mean-shift est une technique de clustering non paramétrique qui n'exige pas la connaissance préalable du nombre de clusters et ne contraint pas la forme des clusters.

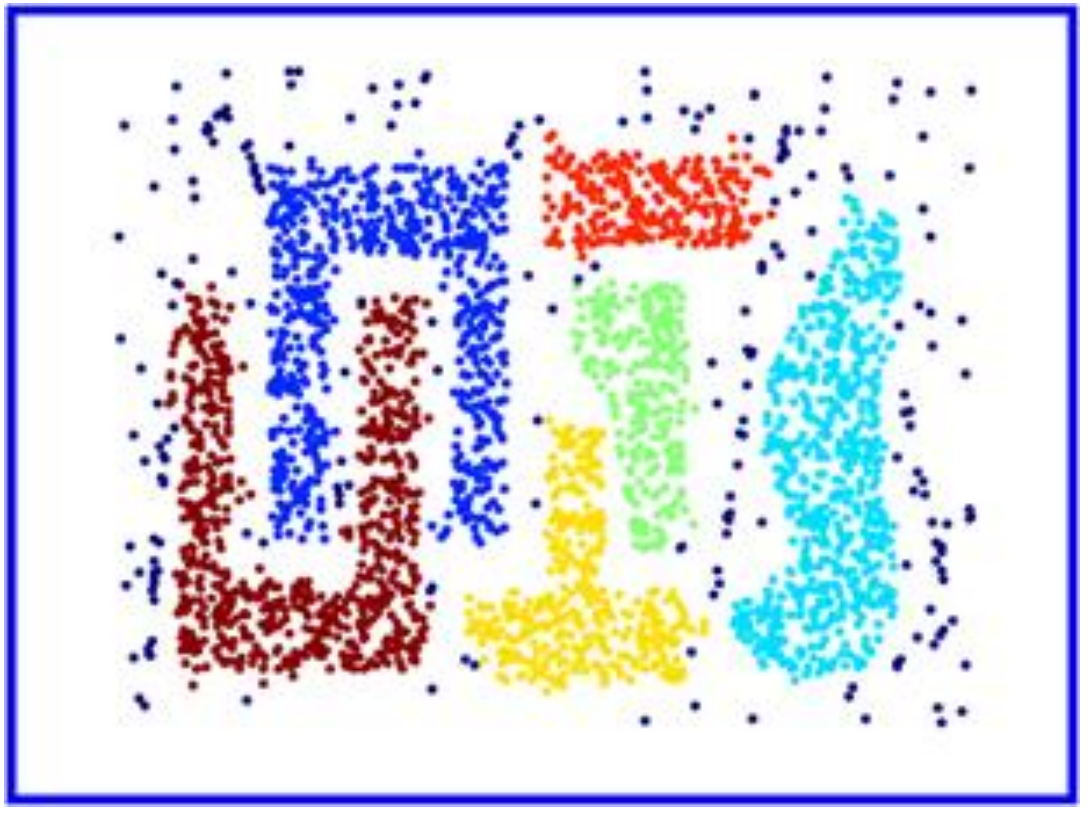
- On peut également utiliser des pondérations particulières de la moyenne pour faire converger le point P vers d'autres zones.

Mean-Shift : En pratique

+ Mean-Shift ne nécessite pas de sélectionner le nombre de clusters(découvre automatiquement).
– Le fait que les centres des clusters convergent vers les points de densité maximale est également plausible (intuitif , guidé naturellement par les données).
– L'inconvénient est que la sélection de la taille de la fenêtre/du rayon "r" peut être non négligeable.

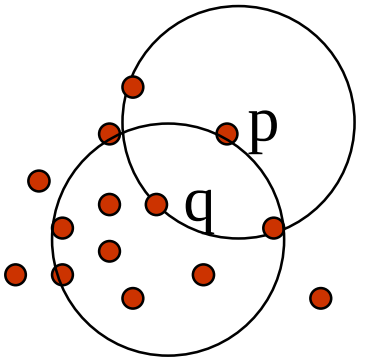
DBSCAN

Density-Based Spatial Clustering of Applications with Noise



DBSCAN : Algorithme

1. DBSCAN commence par un point de départ arbitraire (qui n'a pas été visité). Le voisinage de ce point est extrait à l'aide d'une distance epsilon ε (tous les points qui se trouvent dans la distance ε sont des points de voisinage).

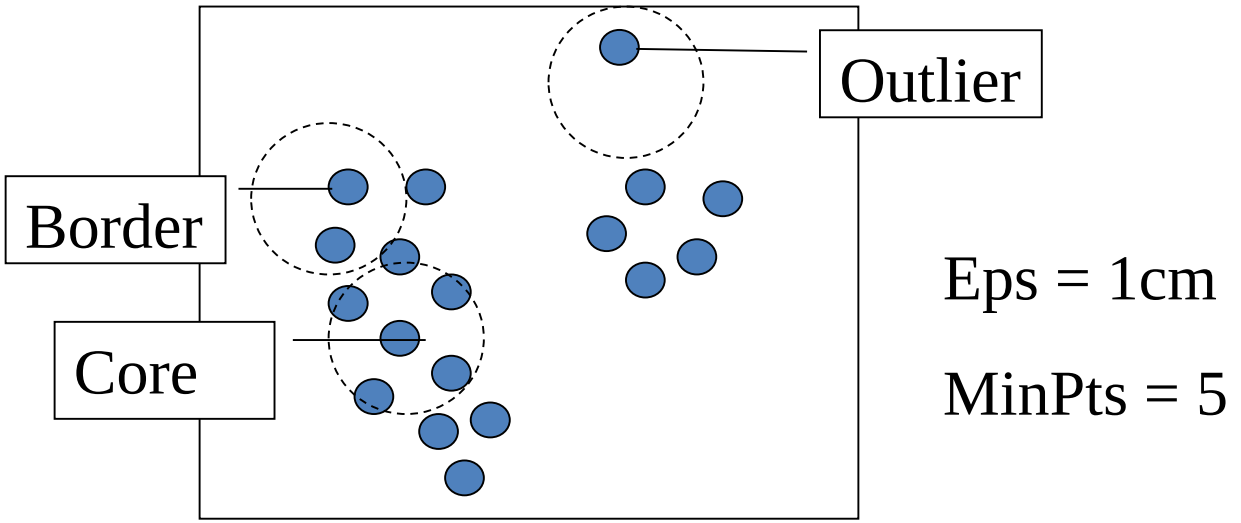


MinPoints = 5

Eps = 1 cm

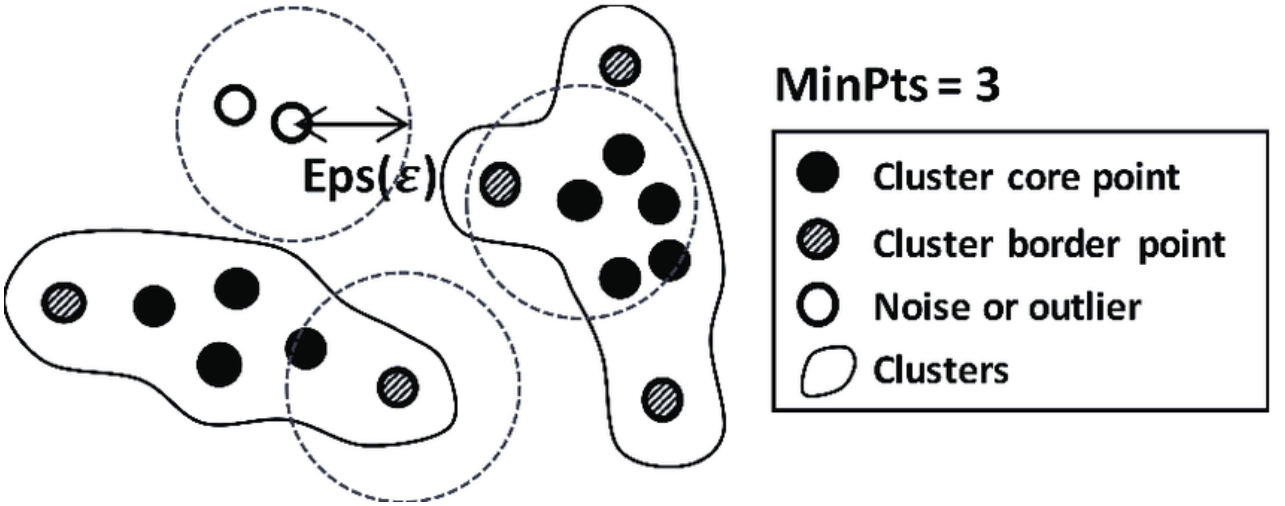
DBSCAN : Algorithme

2. S'il y a un nombre suffisant de points (selon les MinPoints) dans ce voisinage, le processus de clustering commence et le point actuel devient le premier point du nouveau cluster. Sinon, le point sera étiqueté comme bruit (plus tard ce point pourrait devenir une partie du cluster). Dans les deux cas, ce point est marqué comme "visité".



DBSCAN : Algorithme

3. Pour ce premier point du nouveau cluster, les points de son voisinage ε font également partie du même cluster. Cette procédure consistant à faire en sorte que tous les points du voisinage de ε appartiennent au même cluster est ensuite répétée pour tous les nouveaux points qui viennent d'être ajoutés au groupe de cluster.



DBSCAN : Algorithme

4. Ce processus des étapes 2 et 3 est répété jusqu'à ce que tous les points du cluster soient déterminés, c'est-à-dire que tous les points du voisinage ε du cluster aient été visités et étiquetés.
5. Une fois que nous en avons fini avec le cluster actuel, un nouveau point non visité est récupéré et traité, menant à la découverte d'un autre cluster ou bruit. Ce processus se répète jusqu'à ce que tous les points soient marqués comme étant visités. Étant donné qu'à la fin de cette visite tous les points ont été visités, chaque point a été marqué comme faisant partie d'un cluster ou comme étant du bruit.

DBSCAN : En pratique

Avantages :

- Tout d'abord, il n'a pas besoin d'un nombre déterminé de clusters.
- Résistant au bruit (identifie les valeurs aberrantes comme des bruits).
- Peut manipuler des clusters de **formes** et de **tailles** différentes.

DBSCAN : En pratique

Inconvénients :

- Sensible aux paramètres (MinPoints, epsilon ε).
- Ne peut pas gérer des densités variables.

En effet, le réglage du seuil de distance ε et des MinPoints pour l'identification des points de voisinage varie d'un cluster à l'autre lorsque la densité varie. Cet inconvénient se produit également avec des données à **très haute dimension** puisque, encore une fois, le seuil de distance ε devient difficile à estimer.