Clustering

(Groupement)

C’est quoi ?

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

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

Structures de données

- Matrice de données

$$\begin{bmatrix} x_{11} & \cdots & x_{1f} & \cdots & x_{1p} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{i1} & \cdots & x_{if} & \cdots & x_{ip} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{n1} & \cdots & x_{nf} & \cdots & x_{np} \end{bmatrix}$$

- Matrice de similarité

$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & \vdots & \vdots & & \\ d(n,1) & d(n,2) & \cdots & \cdots & 0 \end{bmatrix}$$

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*)

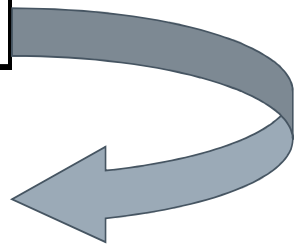
$$z_{if} = \frac{x_{if} - m_f}{s_f}$$

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

Similarité entre objets

- Les distances expriment une similarité
- Ex: la *distance de Minkowski* :

$$d(i,j) = \sqrt[q]{(|x_{i_1} - x_{j_1}|^q + |x_{i_2} - x_{j_2}|^q + ... + |x_{i_p} - x_{j_p}|^q)}$$

où $i = (x_{i1}, x_{i2}, \ldots, x_{ip})$ et $j = (x_{j1}, x_{j2}, \ldots, x_{jp})$ sont deux objets p-dimensionnels et q un entier positif

- Si $q = 1$, d est la distance de Manhattan

$$d(i,j) = |x_{i_1} - x_{j_1}| + |x_{i_2} - x_{j_2}| + ... + |x_{i_p} - x_{j_p}|$$

Similarité entre objets(I)

- *Si q = 2, d est la distance Euclidienne* :

$$d(i,j)=\sqrt{(|x_{i_1}-x_{j_1}|^2+|x_{i_2}-x_{j_2}|^2+...+|x_{i_p}-x_{j_p}|^2)}$$

  - Propriétés
    - $d(i,j) \geq 0$
    - $d(i,i) = 0$
    - $d(i,j) = d(j,i)$
    - $d(i,j) \leq d(i,k) + d(k,j)$

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

- Une table de contingence pour données binaires

		Objet *j*		
		1	0	*sum*
Objet *i*	1	a	b	$a+b$
	0	c	d	$c+d$
	sum	$a+c$	$b+d$	p

a= nombre de positions où i a 1 et j a 1

- Exemple o_i=(1,1,0,1,0) et o_j=(1,0,0,0,1)

a=1, b=2, c=1, d=1

Mesures de distances

- Coefficient d'appariement (matching) simple (invariant pour variables symétriques):

$$d(i, j) = \frac{b + c}{a + b + c + d}$$

Exemple o_i=(1,1,0,1,0) et o_j=(1,0,0,0,1)

$d(o_i, o_j)$=3/5

- Coefficient de Jaccard

$$d(i, j) = \frac{b + c}{a + b + c}$$

$d(o_i, o_j)$=3/4

Variables binaires (I)

- 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 binaires(II)

- Exemple

Nom	Sexe	Fièvre	Toux	Test-1	Test-2	Test-3	Test-4
Jack	M	Y	N	P	N	N	N
Mary	F	Y	N	P	N	P	N
Jim	M	Y	P	N	N	N	N

- Sexe est un attribut symétrique
- Les autres attributs sont asymétriques
- Y et P ≡ 1, N ≡ 0, la distance n'est mesurée que sur les asymétriques

$$d(jack, mary) = \frac{0 + 1}{2 + 0 + 1} = 0.33$$

$$d(jack, jim) = \frac{1 + 1}{1 + 1 + 1} = 0.67$$

$$d(jim, mary) = \frac{1 + 2}{1 + 1 + 2} = 0.75$$

Les plus similaires sont Jack et Mary⇒atteints du même mal

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

- Une variable ordinale peut être discrète ou continue
- L'ordre peut être important, ex: classement
- Peuvent être traitées comme les variables intervalles
  - remplacer x_{if} par son rang $r_{if} \in \{1, \ldots, M_f\}$
  - Remplacer le rang de chaque variable par une valeur dans [0, 1] en remplaçant la variable f dans l'objet I par

$$z_{if} = \frac{r_{if} - 1}{M_f - 1}$$

  - Utiliser une distance pour calculer la similarité

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

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

  - *f* est binaire ou nominale:

    $d_{ij}^{(f)} = 0$ si $x_{if} = x_{jf}$, sinon $d_{ij}^{(f)} = 1$
  - *f* est de type intervalle: utiliser une distance normalisée
  - *f* est ordinale
    - calculer les rangs r_{if} et
    - Ensuite traiter z_{if} comme une variable de type intervalle

$$z_{if} = \frac{r_{if} - 1}{M_f - 1}$$

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

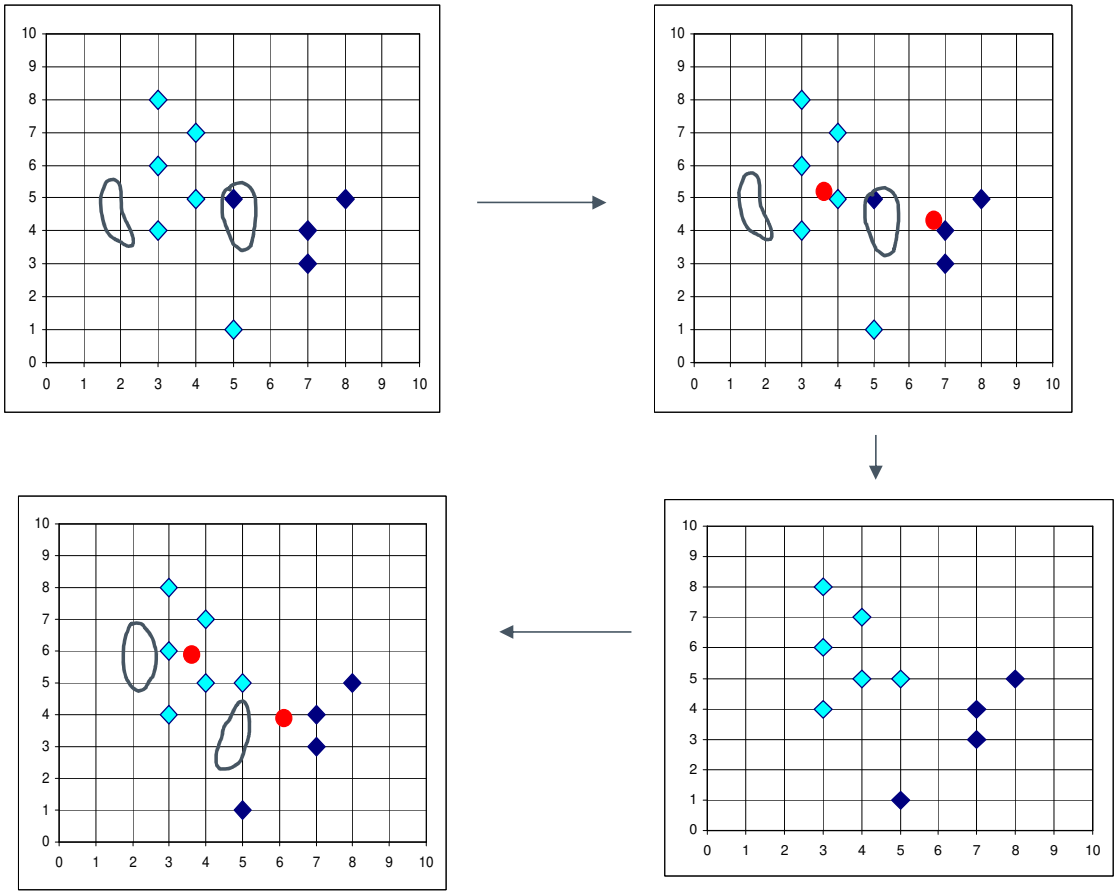
- dist(2,M1)<dist(2,M2)➔2 passe en C1. dist(7,M2)<dist(7,M3)➔ 7 passe en C2. Les autres ne bougent pas. C1={1,2}, M1=1.5, C2={3,6,7}, M2=5.34, C3={8,13,15,17}, M3=13.25

- 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

Algorithme *K-Means*

- Exemple



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*

La méthode des *K-Medoids (PAM)*

- Trouver des objets représentatifs (medoïdes) dans les clusters (au lieu de la moyenne)
- Principe
  - Commencer avec un ensemble de medoïdes puis itérativement remplacer un par un autre si ça permet de réduire la distance globale
  - Efficace pour des données de petite taille

Algorithme des k-Medoides

Choisir arbitrairement k medoides

Répéter

affecter chaque objet restant au medoide le plus proche

Choisir aléatoirement un non-medoide O_r

Pour chaque medoide O_j

Calculer le coût TC du remplacement de O_j par O_r

Si TC < 0 alors

Remplacer O_j par O_r

Calculer les nouveaux clusters

Finsi

FinPour

Jusqu'à ce ce qu'il n'y ait plus de changement

PAM (Partitioning Around Medoids) (1987)

Choisir arbitrairement ***k*** objets représentatifs

- Pour toute paire (h,j) d'objets t.q h est choisi et j non, calculer le coût TC_{jh} du remplacement de j par h
  - Si $TC_{ih} < 0$, ***j*** est remplacé par ***h***
  - Puis affecter chaque objet non sélectionné au medoïde qui lui est le plus similaire
- Répéter jusqu'à ne plus avoir de changements

La méthode des *K-Medoids*

- TC_{jh} représente le gain en distance globale que l'on va avoir en remplaçant h par j

- Si TC_{jh} est négatif alors on va perdre en distance. Ca veut dire que les clusters seront plus compacts.

- $TC_{jh}=\Sigma_i$ dist(j,h)-dist(j,i)= Σ_i C_{ijh}

La méthode des *K-Medoids:* Exemple

- Soit A={1,3,4,5,8,9}, k=2 et M={1,8} ensemble des medoides

  ➔C1={1,3,4} et C2={5,8,9}

  $E_{\{1,8\}}$=dist(3,1)^2+dist(4,1)^2+dist(5,8)^2+dist(9,8)^2=23

- Comparons 1 et 3➔M={3,8}➔C1={1,3,4,5} et C2={8,9}

  $E_{\{3,8\}}$ =dist(1,3)^2+dist(4,3)^2+dist(5,3)^2+dist(9,8)^2=10

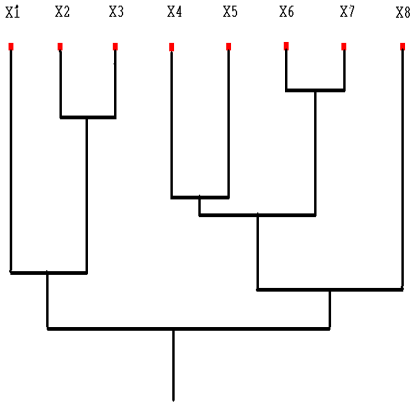
  $E_{\{3,8\}}$ - $E_{\{1,8\}}$= -19 <0 donc le remplacement est fait.

- Comparons 3 et 4➔ M={4,8}➔ C1 et C2 inchangés et $E_{\{4,8\}}$=dist(1,4)^2+dist(3,4)^2+dist(5,4)^2+dist(8,9)^2= 12➔ 3 n'est pas remplacé par 4
- Comparons 3 et 5➔M={5,8}➔ C1 et C2 inchangés et E{5,8}>E{3,8}

Main Techniques (2)

Hierarchical Clustering

- Multilevel clustering: level 1 has n clusters → level n has one cluster, or upside down.
- Agglomerative HC: starts with singleton and merge clusters (bottom-up).
- Divisive HC: starts with one sample and split clusters (top-down).



Dendrogram

Cluster Analysis

- **Définition du problème**
  -) Types de donnees a analyser
  -) Distances et mesures de similarite utilisees
- **Principales méthodes de catégorisation**
  -) Partitioning methods (nuees dynamiques)
    - – *Par recherche du partitionnement optimal des données*
  -) **Hierarchical methods**
    - – *Par division ou agglomération successive des données*
  -) Density-based method
    - – *Par recherche de sous-ensembles denses de données*
  -) Grid-based methods
    - – *Par découpage en cellules de l'espace des données*

AGNES (AGglomerative NESting)

- **Principe de base**
  -) Introduit par Kaufmann et Rousseeuw (1990)
  -) Au depart : un objet = une classe
  -) A chaque etape : regroupement des classes les plus proches
- **Problème posé**
  -) Comment calculer la distance entre deux classes ?
    - *Distance du lien minimum*

      $D_{min}(C_1, C_2) = \min \{ d(O_1,O_2) / O_1 \in C_1 \text{ et } O_2 \in C_2 \}$
    - *Distance du lien maximum*

      $D_{max}(C_1, C_2) = \max \{ d(O_1,O_2) / O_1 \in C_1 \text{ et } O_2 \in C_2 \}$
    - *Distance des centres de gravités*

      $D_{max}(C_1, C_2) = d(M_1,M_2)$
    - *Distance moyenne*

      $D_{max}(C_1, C_2) = 1 / (|C_1|.|C_2|) \quad d(O_1,O_2)$ où $O_1 \in C_1$,$O_2 \in C_2$

DIANA (DIvise ANAlysis)

- **Principe de base**
  -) Introduit par Kaufmann et Rousseeuw (1990)
  -) **Au départ** : une classe composee de tous les objets
  -) **A chaque étape** : division d'une sous-classes en deux nouvelles sous-classes
- **Problème posé**
  -) Selon quels critères effectuer la division ?
    - – *Prendre le découpage qui permet au mieux d'augmenter la qualité de la partition engendrée*
    - – *Condition d'arrêt basée sur un seuil minimal d'augmentation de la qualité de la partition*

Critères de fusion-éclatement

- Exemple: pour les méthodes agglomératives, C1 et C2 sont fusionnés si
  - il existe o1 ∈ C1 et o2∈ C2 tels que dist(o1,o2) ≤ seuil, ou
  - il n'existe pas o1 ∈ C1 et o2∈ C2 tels que dist(o1,o2) ≥ seuil, ou
  - distance entre C1 et C2 ≤ seuil avec

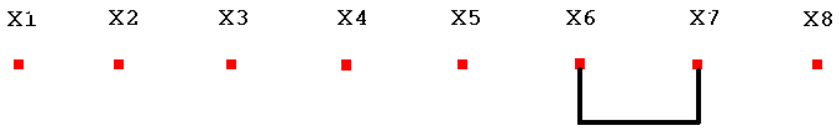
$$dist\ (C_1, C_2) = \frac{1}{n1 * n2} \sum_{o1 \in C1, o2 \in C2} dist(o1, o2)$$

et n1=|C1|.

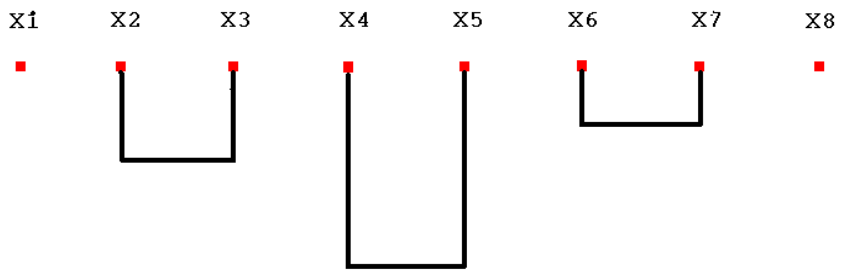
- Ces techniques peuvent être adaptées pour les méthodes divisives

Agglomerative HC Example

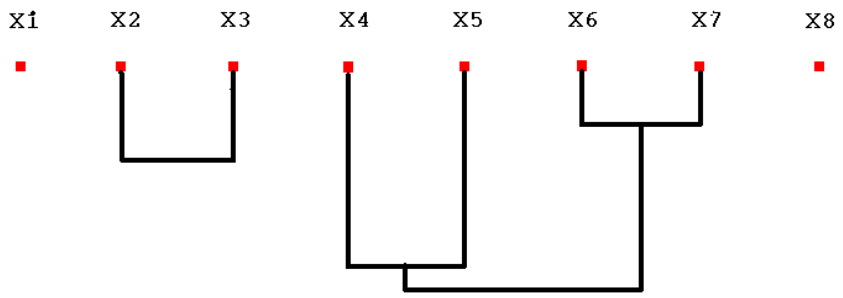
Nearest Neighbor Level 2, k = 7 clusters.



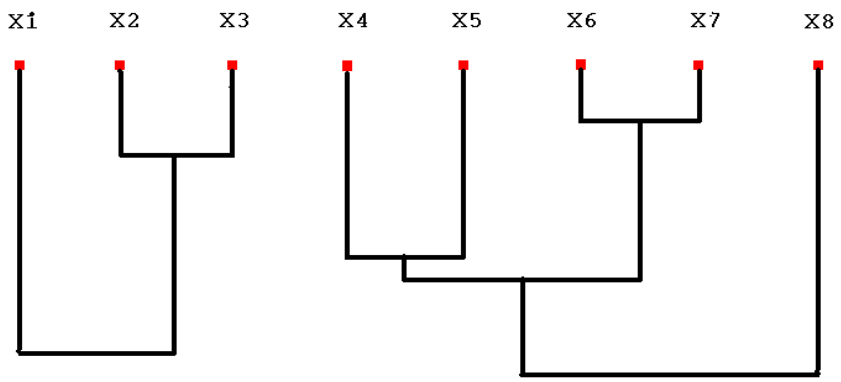
Nearest Neighbor, Level 4, k = 5 clusters.



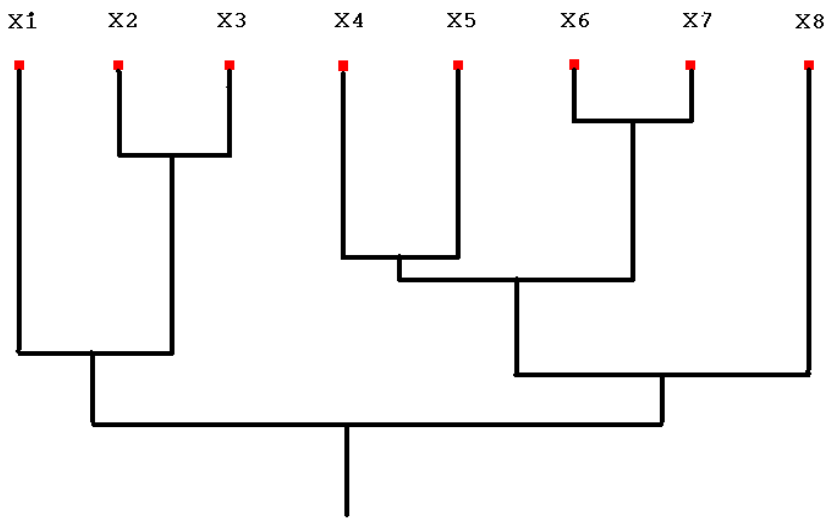
Nearest Neighbor, Level 5, k = 4 clusters.



Nearest Neighbor, Level 7, k = 2 clusters.



Nearest Neighbor, Level 8, k = 1 cluster.



Remarks

	Partitioning Clustering	Hierarchical Clustering
Time Complexity	$O(n)$	$O(n^2 \log n)$
Pros	Easy to use and Relatively efficient	Outputs a dendrogram that is desired in many applications.
Cons	Sensitive to initialization; bad initialization might lead to bad results. Need to store all data in memory.	higher time complexity; Need to store all data in memory.

BIRCH Algorithm

- Designed for very large data sets
  - Time and memory are limited
  - Incremental and dynamic clustering of incoming objects
  - Only one scan of data is necessary
  - Does not need the whole data set in advance
- Two key phases:
  - Scans the database to build an in-memory tree
  - Applies clustering algorithm to cluster the leaf nodes

BIRCH:

Balanced Iterative Reducing and Clustering using Hierarchies

Tian Zhang, Raghu Ramakrishnan, Miron Livny

Presented by Zhao Li

2009, Spring

Similarity Metric(1)

Given a cluster of instances $\{\vec{X}_i\}$*, we define:*

Centroid: $\vec{X0} = \frac{\sum_{i=1}^{N} \vec{X}_i}{N}$

Radius: *average distance from member points to centroid*

$$R = \left(\frac{\sum_{i=1}^{N} (\vec{X}_i - \vec{X0})^2}{N}\right)^{\frac{1}{2}}$$

Diameter: *average pair-wise distance within a cluster*

$$D = \left(\frac{\sum_{i=1}^{N} \sum_{j=1}^{N} (\vec{X}_i - \vec{X}_j)^2}{N(N-1)}\right)^{\frac{1}{2}}$$

Similarity Metric(2)

centroid Euclidean distance:

$$D0 = ((\vec{X}0_1 - \vec{X}0_2)^2)^{\frac{1}{2}}$$

centroid Manhattan distance:

$$D1 = |\vec{X}0_1 - \vec{X}0_2| = \sum_{i=1}^{d} |\vec{X}0_1^{(i)} - \vec{X}0_1^{(i)}|$$

average inter-cluster:

$$D2 = (\frac{\sum_{i=1}^{N_1} \sum_{j=N_1+1}^{N_1+N_2} (\vec{X}_i - \vec{X}_j)^2}{N_1 N_2})^{\frac{1}{2}}$$

average intra-cluster:

$$D3 = (\frac{\sum_{i=1}^{N_1+N_2} \sum_{j=1}^{N_1+N_2} (\vec{X}_i - \vec{X}_j)^2}{(N_1+N_2)(N_1+N_2-1)})^{\frac{1}{2}}$$

variance increase:

$$(\sum_{k=1}^{N_1+N_2} (\vec{X}_k - \frac{\sum_{l=1}^{N_1+N_2} \vec{X}_l}{N_1+N_2})^2$$

$$- \sum_{i=1}^{N_1} (\vec{X}_i - \frac{\sum_{l=1}^{N_1} \vec{X}_l}{N_1})^2 - \sum_{j=N_1+1}^{N_1+N_2} (\vec{X}_j - \frac{\sum_{l=N_1+1}^{N_1+N_2} \vec{X}_l}{N_2})^2)^{\frac{1}{2}}$$

Clustering Feature

- *The Birch algorithm builds a dendrogram called clustering feature tree (CF tree) while scanning the data set.*
- *Each entry in the CF tree represents a cluster of objects and is characterized by a 3-tuple: (N, LS, SS), where N is the number of objects in the cluster and LS, SS are defined in the following.*

$$LS = \sum_{P_i \in N} \vec{P}_i$$

$$SS = \sum_{P_i \in N} \left| \vec{P}_i \right|^2$$

Clustering Feature Vector

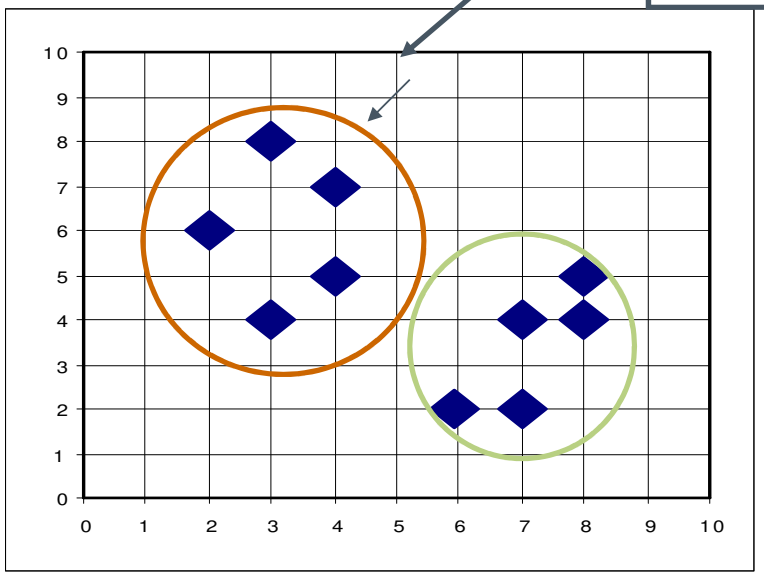
Clustering Feature: ***CF = (N, LS, SS)***

N: **Number of data points**

LS: $\sum_{i=1}^{N} X_i$

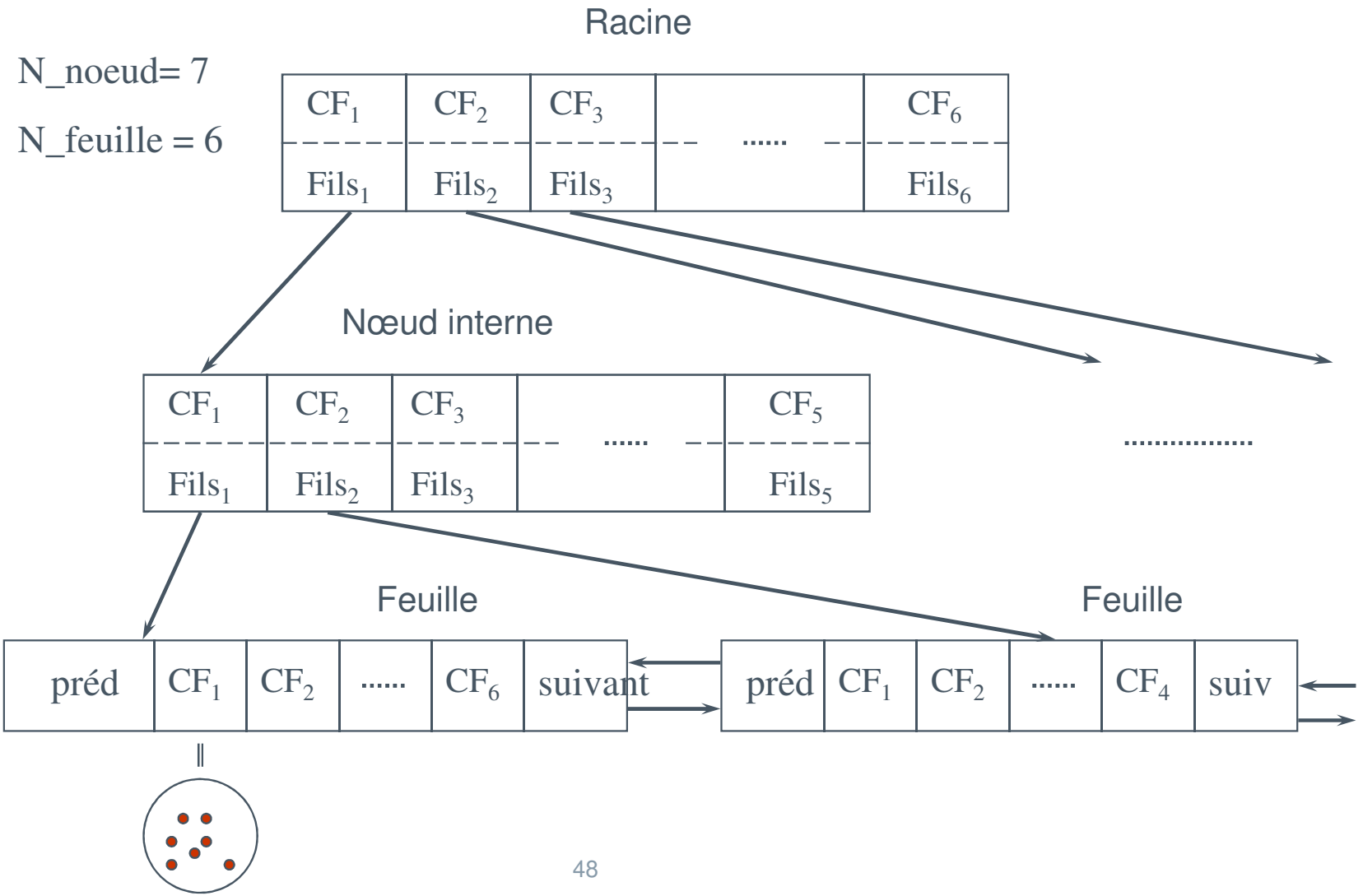
SS: $\sum_{i=1}^{N} X_i^2$

CF = (5, (16,30),(54,190))



(3,4)
(2,6)
(4,5)
(4,7)
(3,8)

CF Tree


Racine
N_noeud= 7
N_feuille = 6
CF1
CF2
CF3
.....
CF6
Fils1
Fils2
Fils3
Fils6
Nœud interne
CF1
CF2
CF3
.....
CF5
Fils1
Fils2
Fils3
Fils5
.....
Feuille
Feuille
préd
CF1
CF2
.....
CF6
suivant
préd
CF1
CF2
.....
CF4
suiv

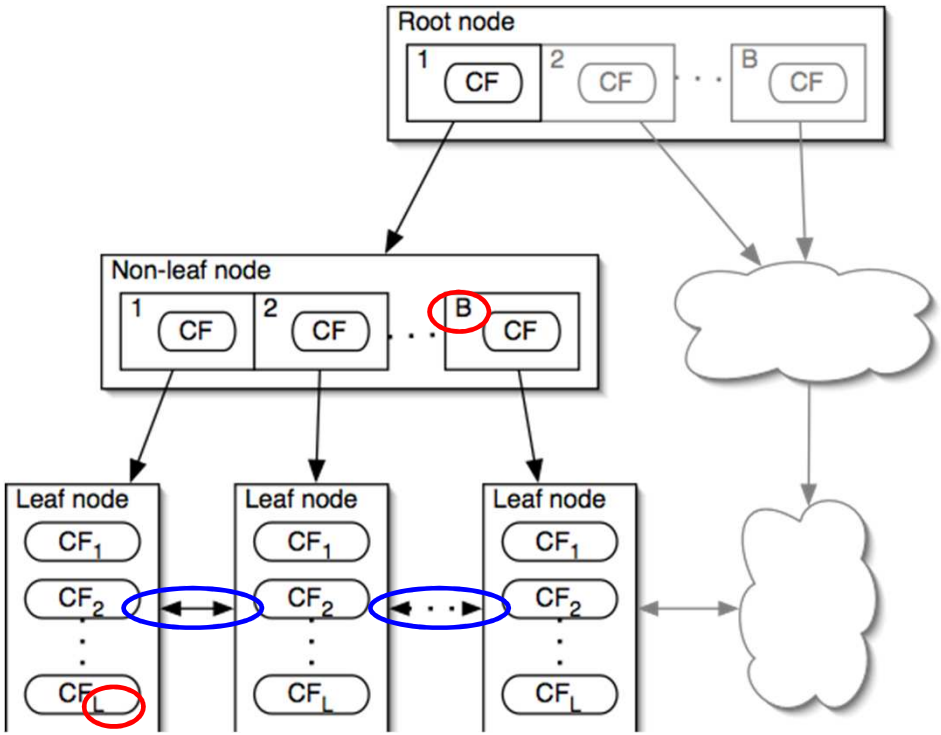
Properties of Clustering Feature

- CF entry is more compact
  - Stores significantly less than all of the data points in the sub-cluster
- A CF entry has sufficient information to calculate D0-D4
- Additivity theorem allows us to merge sub-clusters incrementally & consistently

$$\mathbf{CF_1} + \mathbf{CF_2} = (N_1 + N_2, \vec{LS}_1 + \vec{LS}_2, SS_1 + SS_2)$$

CF-Tree

- Each non-leaf node has at most B entries
- Each leaf node has at most L CF entries, each of which satisfies threshold T
- Node size is determined by dimensionality of data space and input parameter P (page size)



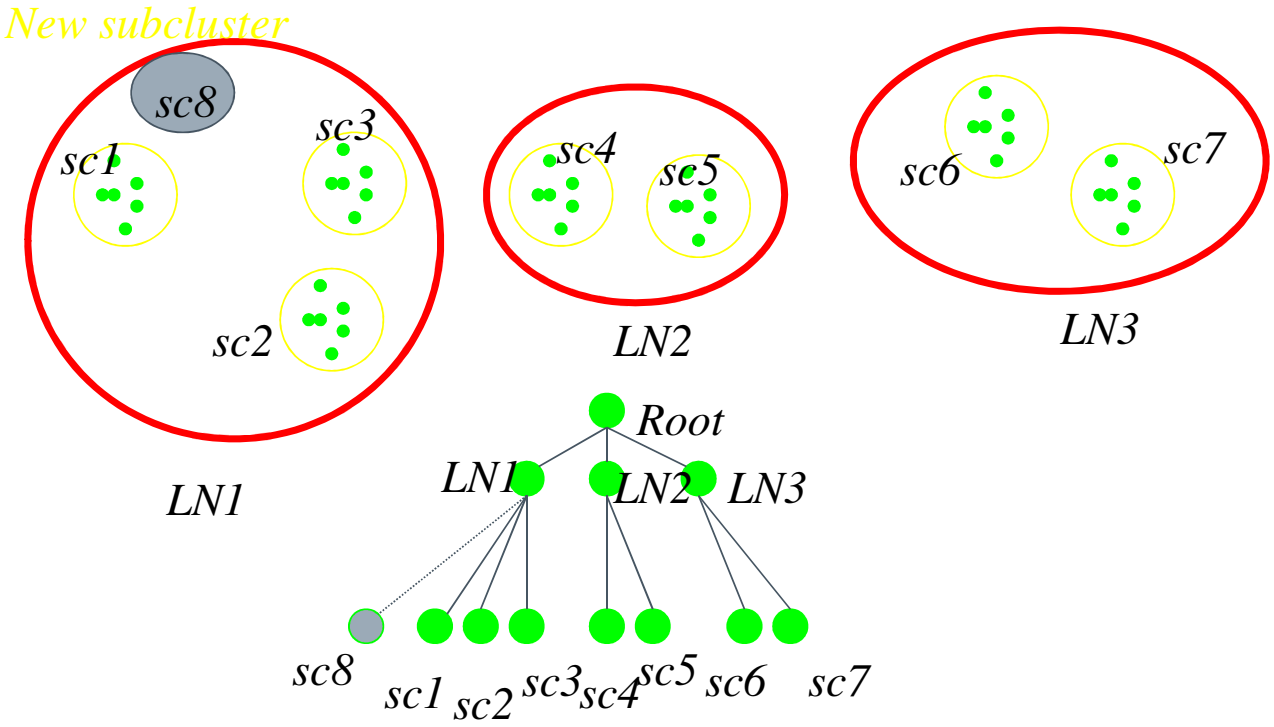
CF-Tree Insertion

- Recurse down from root, find the appropriate leaf
  - Follow the "closest"-CF path, w.r.t. D0 / … / D4
- Modify the leaf
  - If the closest-CF leaf cannot absorb, make a new CF entry. If there is no room for new leaf, split the parent node
- Traverse back
  - Updating CFs on the path or splitting nodes

CF-Tree Rebuilding

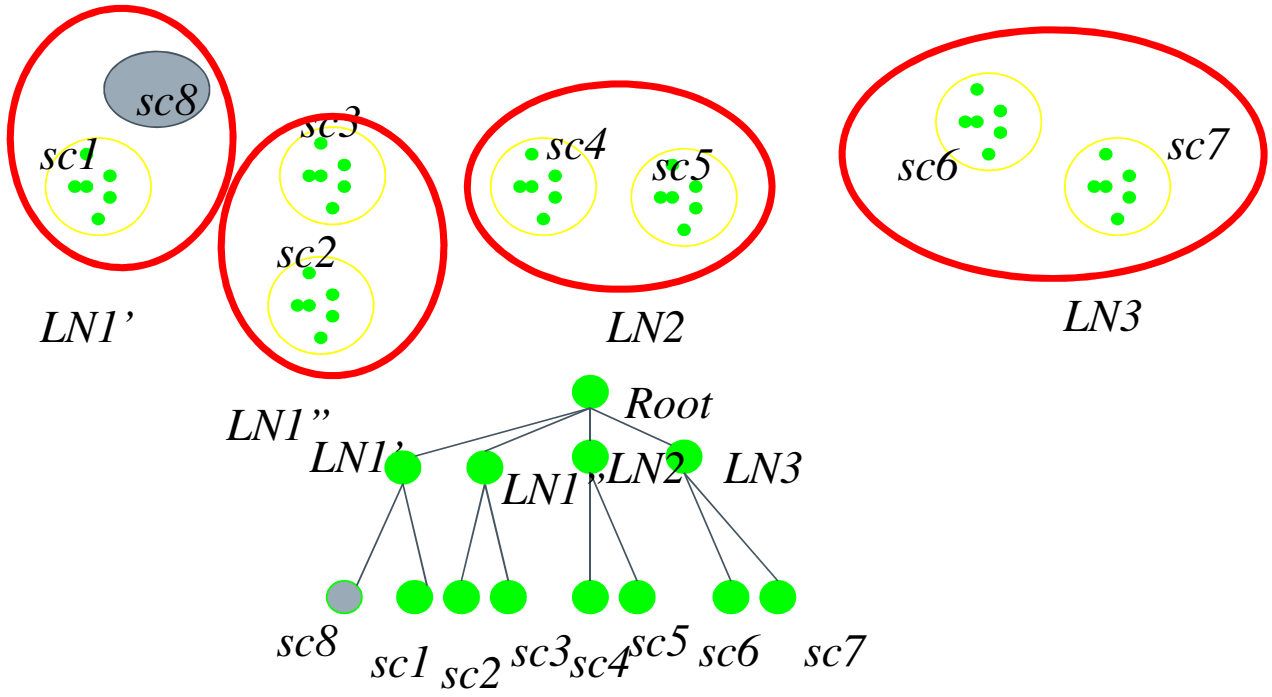
- If we run out of space, increase threshold T
  - By increasing the threshold, CFs absorb more data
- Rebuilding "pushes" CFs over
  - The larger T allows different CFs to group together
- Reducibility theorem
  - Increasing T will result in a CF-tree smaller than the original
  - Rebuilding needs at most h extra pages of memory

Example of BIRCH

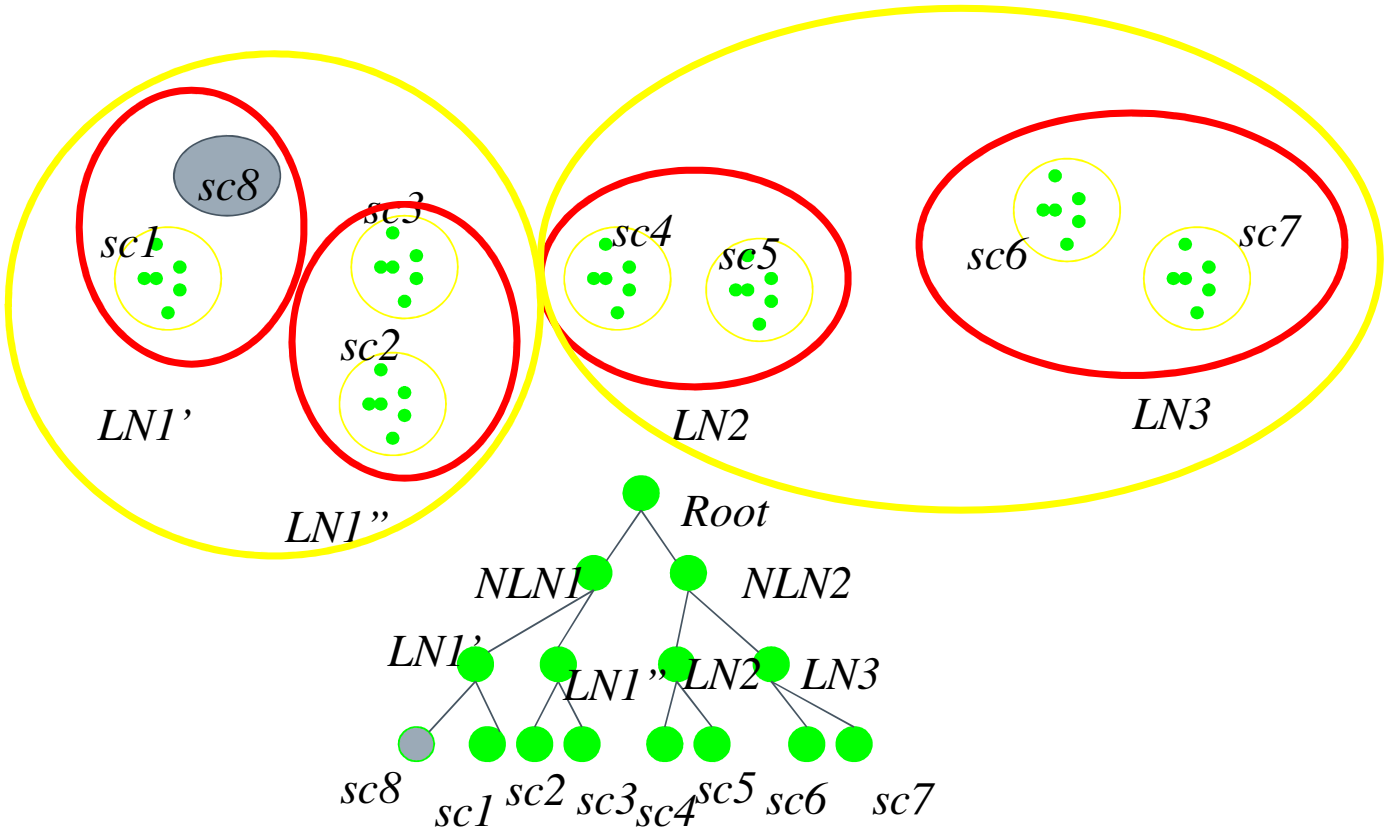


Insertion Operation in BIRCH

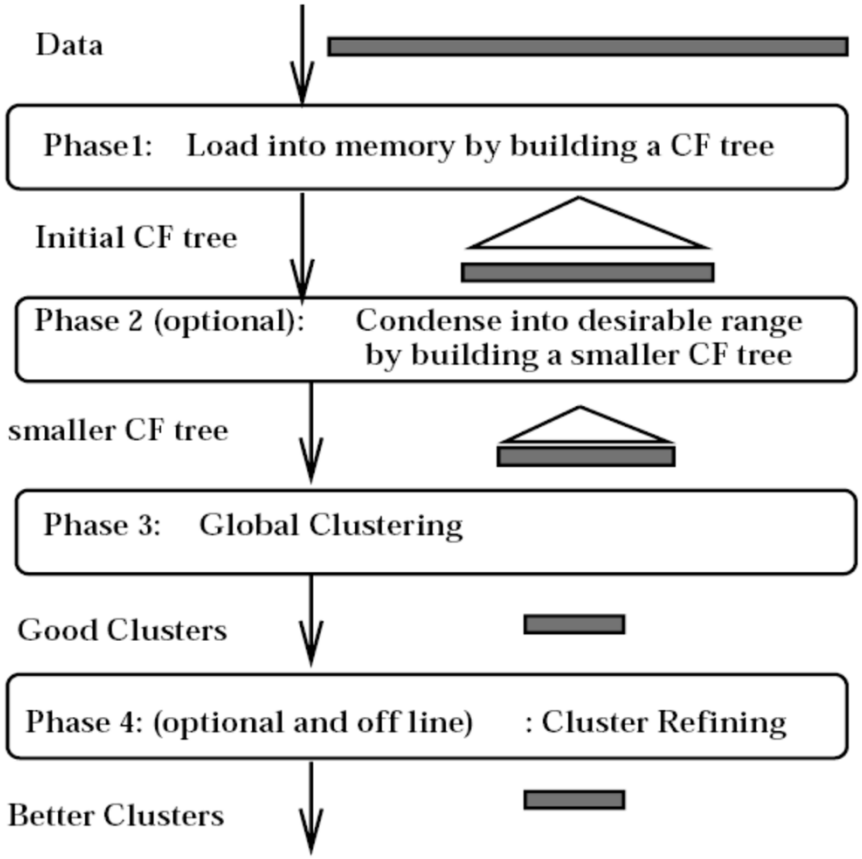
If the branching factor of a leaf node can not exceed 3, then LN1 is split.



If the branching factor of a non-leaf node can not exceed 3, then the root is split and the height of the CF Tree increases by one.



BIRCH Overview



Experimental Results

- Input parameters:
  - Memory (M): 5% of data set
  - Disk space (R): 20% of M
  - Distance equation: D2
  - Quality equation: weighted average diameter (D)
  - Initial threshold (T): 0.0
  - Page size (P): 1024 bytes

Experimental Results

KMEANS clustering

DS	Time	D	# Scan	DS	Time	D	# Scan
1	***43.9***	***2.09***	***289***	*1o*	*33.8*	*1.97*	*197*
2	*13.2*	*4.43*	*51*	*2o*	*12.7*	*4.20*	*29*
3	*32.9*	*3.66*	*187*	*3o*	*36.0*	*4.35*	*241*

BIRCH clustering

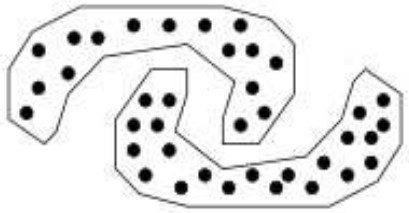
DS	Time	D	# Scan	DS	Time	D	# Scan
1	***11.5***	***1.87***	***2***	*1o*	*13.6*	*1.87*	*2*
2	*10.7*	*1.99*	*2*	*2o*	*12.1*	*1.99*	*2*
3	*11.4*	*3.95*	*2*	*3o*	*12.2*	*3.99*	*2*

Conclusions

- A CF tree is a height-balanced tree that stores the clustering features for a hierarchical clustering.
- Given a limited amount of main memory, BIRCH can minimize the time required for I/O.
- BIRCH is a scalable clustering algorithm with respect to the number of objects, and good quality of clustering of the data.

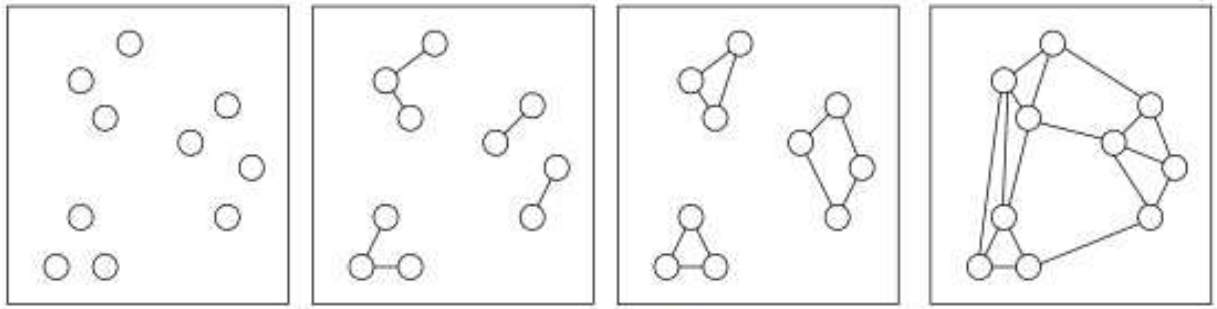
CHAMELON (99)

- **Méthode mixte**
  - Introduite par G. Karpis, E. H. Han et V. Kumar
- **Principe de base**
  - Représentation des données sous-forme la forme d'un graphe
    - *Graphe des k plus proches voisins (k-nearest neighbor graph)*
  - Algorithme en deux phases
    - ***Phase (1) :*** *utilisation d'un algorithme de graphes de partitionnement permettant d'obtenir un nombre important de petites sous-classes*
    - ***Phase (2) :*** *utilisation d'un algorithme de clustering hiérarchique basé sur de nouvelles mesures d'interconnectivité et de proximité entre sous-classes*
- **Points forts**
  - Permet notamment de découvrir des classes non connexes



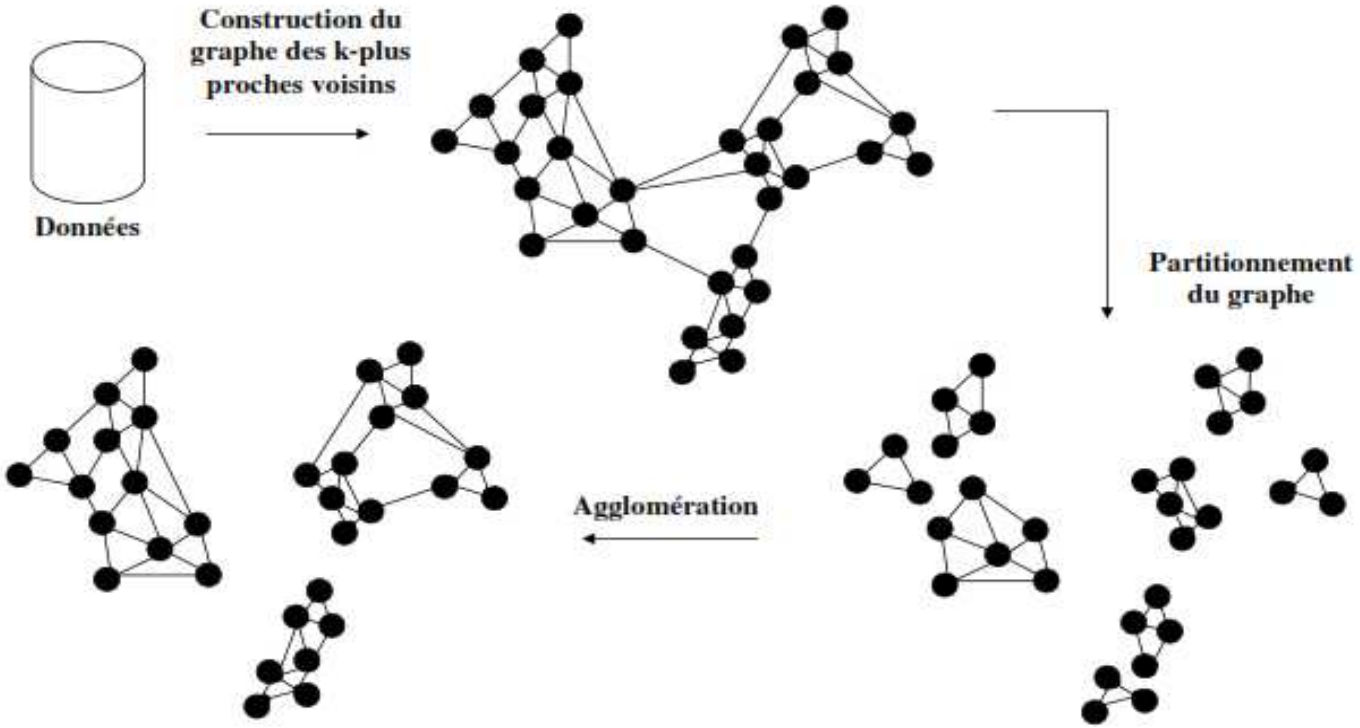
CHAMELON (99)

- **Représentation des données**
  - Graphe des k-plus proches voisins
    - *Tout nœud décrivant un objet donnée est relié à ses k-plus proches voisins*
    - *Tout arc est valué par la similarité entre les objets des nœuds connectés*



Données ***Graphe des 1-plus proches voisins*** ***Graphe des 2-plus proches voisins*** ***Graphe des 3-plus proches voisins***

CHAMELON (99)


Construction du graphe des k-plus proches voisins
Données
Partitionnement du graphe
Agglomération

Clustering de données Catégorielles : ROCK

- ROCK: Robust Clustering using linKs
  - Utilise les liens pour mesurer la similarité/proximité
  - N'est pas basé sur la notion de distance
- Idée :
  - Fonction de similarité et voisins:

    Let T_1 = {1,2,3}, T_2={3,4,5}

$$Sim(T_1, T_2) = \frac{|T_1 \cap T_2|}{|T_1 \cup T_2|}$$

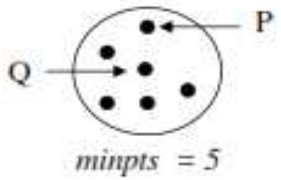
$$Sim(T1, T2) = \frac{|\{3\}|}{|\{1,2,3,4,5\}|} = \frac{1}{5} = 0.2$$

Density-based Clustering Method

- **Principe de base**
  - Le regroupement d'objets au sein d'une même classe est basé sur un critère local évaluant la densité d'objets en un point donné de l'espace d'entrée
- **Caractéristiques de ces méthodes**
  - Découvre des classes de formes quelconques
  - Ne sont pas sensibles à la présence de bruit ou points isolés
  - Nécessite seulement un parcours des données
- **Différentes propositions**
  - **DBSCAN** : *introduite par Ester, et al. (KDD '96)*
  - **OPTICS** : *introduite par Ankerst, et al. (SIGMOD'99)*
  - **DENCLUE** : *introduite par Hinneburg & Keim (KDD'98)*
  - **CLIQUE** : *introduite par Agrawal, et al. (SIGMOD'98)*

DBSCAN (96)

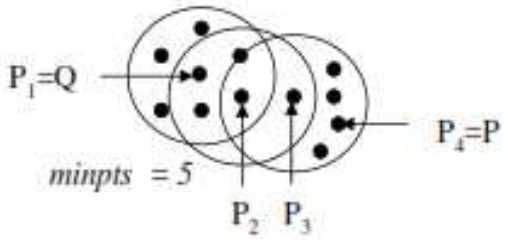
- **Notions de base communes**
  - **Deux paramètres**
    - ε : *rayon maximal de voisinage*
    - ***minpts*** : *nombre minimal de points dans un voisinage*
  - **Voisinage d'un point P par rapport à ε**
    - $N_\varepsilon(P) = \{ P' / d(P, P') < \varepsilon \}$
    - *P est un noyau ssi* $| N_\varepsilon(P) | >$ *minpts*
  - **Point directement accessible** (directly reachable)
    - *Un point P est directement accessible depuis un point Q relativement aux paramètres* ε *et* minpts *si*
      - *P appartient à* $N_\varepsilon(Q)$
      - Q est un noyau



minpts = 5

DBSCAN (96)

- **Suite notions de base**
  - **Point d-accessible** (density reachable)
    - *Un point P est d-accessible depuis un point Q relativement aux paramètres* ε *et* minpts *si il existe une liste de points* $P_1, P_2, ..., P_n$ *telle que*
      - $P_1 = Q$ et $P_n = P$
      - P_{i+1} est directement accessible depuis P_i w.r.t. ε et minpts



  - **Point d-connecté** (density connected)
    - *Un point P est d-connecté à un point Q relativement aux paramètres* ε *et* minpts *si il existe un point O tel que P et Q soient d-accessibles depuis O*

DBSCAN (96)

- **Suite notions de base**
  - **Définition d'une classe**
    - *Une classe est un ensemble maximal de points d-connectés*
    - *Un point est considéré comme étant du bruit s'il n'appartient à aucune classe*
  - **Propriétés fondamentales**
    - *Si P est un point tel que* $| N_{\varepsilon}(P) | >$ *minpts, alors l'ensemble des points d-accessibles depuis P est une classe*
    - *Si C est une classe, alors pour tout point P élément de C, C est égale à l'ensemble des points d-accessibles depuis P*

DBSCAN (96)

- **Algorithme de clustering**
  - **Phases principales de l'algorithme**
    1. Sélectionner aléatoirement un point P
    2. Déterminer l'ensemble E des points d-accessibles depuis P
       - *Efficace en utilisant un index particulier (R*-tree)*
    3. Si P est un noyau, alors E est une classe
    4. Si E est vide, alors sélectionner un autre point P et retourner en (2)
       - *Arrêt quand tous les points ont été sélectionnés*
- **Dernières avancées**
  - Version incrémentale de DBSCAN
    - *Introduite par Ester et al. (VLDB '98)*
  - Détermination des classes pour un rayon maximal variable
    - *Introduite par Ankerst, et al. (SIGMOD'99)*

Grid-Based Clustering Methods

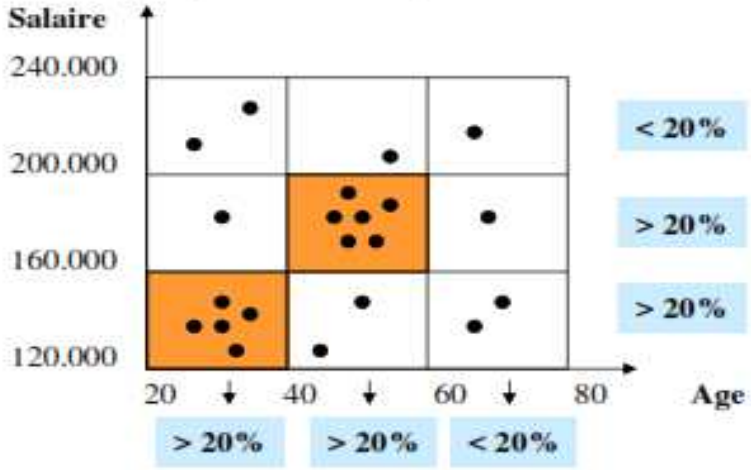
- **Principe de base**
  - Méthodes basées sur une quantification de l'espace d'entrée en un nombre fini de cellules constituant une grille
  - Après initialisation, toutes les opérations de *clustering* seront réalisées sur les cellules, plutôt que sur les données
- **Différentes propositions**
  - **STING** (a STatistical INformation Grid approach)
    - *Introduite par Wang, Yang et Muntz (97)*
  - **WaveCluster** (utilise la notion d'ondelette)
    - *Introduite par Shiekholestami, Chatterjee et Zhang (VLDB'98)*
  - **CLIQUE** (Clustering in QUEst)
    - *Introduite par Agrawal, et al. (SIGMOD '98)*
    - *QUEST : nom du projet de recherche IBM sur l'extraction de connaissance*

CLIQUE (98)

- **Principe de base**
  - CLIQUE est une méthode basée également sur l'**estimation de densités locales de points**
  - **La quantification de l'espace d'entrée** sous la forme d'une grille permet une recherche efficace des zones denses de points
- **Différentes phases**
  - Etant donné un espace d'entrée de M dimensions
  - **Découpage de chaque dimension** de l'espace des données en un nombre fixe d'intervalles de même largeur
  - **Détermination de l'ensemble des cellules denses** dans l'ensemble des sous-espaces de l'espace des données
  - **Détermination des classes** comme ensemble maximal de cellules denses connexes

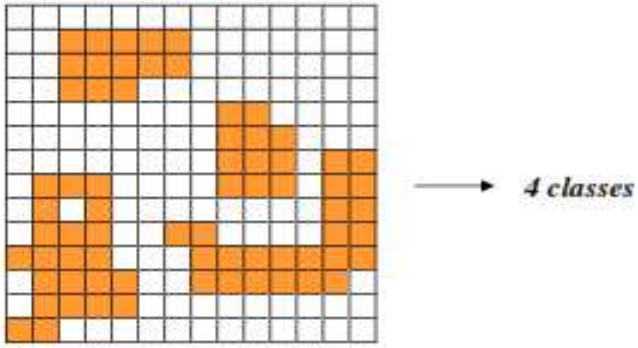
CLIQUE (98)

- **Découpage en cellules et densités**
  - Dans un sous-espace donné, **une cellule est dense** si le pourcentage d'objets qu'elle contient est supérieur à un seuil donné
  - **Propriété fondamentale**
    - *Toute sous-cellule d'une cellule dense est dense*
    - *Permet une recherche ascendante de l'ensemble des cellules denses en commençant par les sous-espaces de dimension 1*



CLIQUE (98)

- **Détermination des classes**
  - Une classe correspond à un ensemble de cellules denses connexes
  - **Graphe considéré**
    - *Un nœud correspond à une cellule dense*
    - *Un arc existe entre deux nœuds si les cellules associées possèdent une face commune*



- **Recherche de descripteurs simples des classes déterminées**
  - Sous forme de règles du type
    - *Age ∈ [20,60] et Salaire ∈ [120KF, 200KF] ⇒ Classe A*

Cluster Analysis : perspectives ?

- **Problèmes**
  - Passage à l'échelle
  - Possibilité de découvrir des classes de formes quelconques
  - Interprétation et utilisation des résultats
    - *Quel sens donner aux classes découvertes ?*
    - *Comment influencer un processus de catégorisation ?*
- **Nouveaux usages en bases de données**
  - Modélisation des distributions de probabilités des données
    - *Pour approximer les calculs des agrégats dans les entrepôts*
      - Somme des ventes réalisées par les vendeurs entre 25 et 35 ans
    - *Pour indexer les données et optimiser les requêtes par similarité*
      - Clients possédant des caractéristiques voisines qu'un client donné