Data Mining

Hierarchical Clustering

An Example

- Study: How different species of Dinoflagellates (Algae) relates to each other by studying their sterol composition
  - identify the relationships via sterol composition similarity amongst dinoflagellates
  - investigate the correspondences between the dinoflagellates sharing a similar sterol compositions and their evolutionary histories.
- Data:
  - Sterol composition of 102 dinoflagellates
- Analysis method
  - Hierarchical Clustering based on Sterol composition data
  - Clustering validation using multiple clustering schemes and clustering criteria

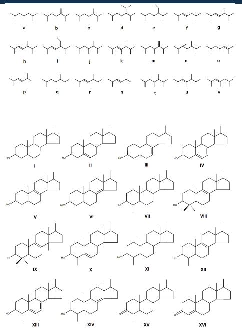
An Example

- Data:
  - 58 named sterols and steroidal ketones
  - 102 dinoflagellate species
- Analysis method
  - Hierarchical Clustering based on Sterol composition data
  - Clustering validation using multiple clustering schemes and clustering criteria

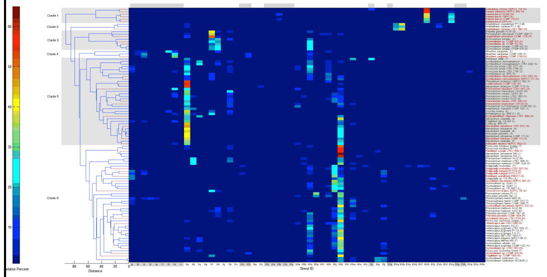
An Example

Structures of sterols in the study

5α-Cholestan-3β-ol (cholestanol)	I a
24-Methylcholesta-24(28)-en-3β-ol (24-methylenecholestanol)	I b
24-Methyl-5α-cholestan-3β-ol	I c
24-Ethyl-5α-cholest-24(28)E-en-3β-ol	I d

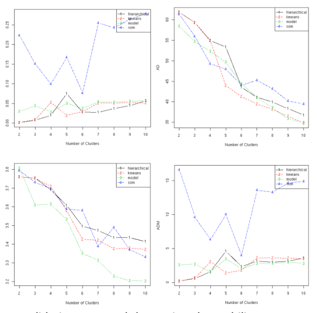


Hierarchical Clustering Result



Dendrogram of dinoflagellate relationships based on sterol compositions accompanied by heat map showing sterol distributions

Clustering Validation



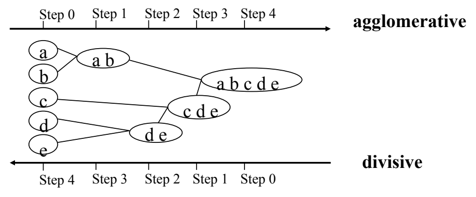
Cluster validation on sterol data using the stability measures

An Example

- Conclusion of the Study:
  - Our results indicated that several, but not all, dinoflagellate genera share similar sterol compositions
  - sterol composition of dinoflagellates has been determined, to a certain extent, by the evolutionary diversification of this lineage.

Agglomerative vs. Divisive Clustering

- Use distance matrix as clustering criteria. This method does not require the number of clusters ***k*** as an input, but needs a termination condition



Single-link vs. Complete-link

- Difference: the way to characterize the similarity between a pair of clusters
  - **single link**: ***minimum*** of the distances between all pairs of patterns drawn from the two clusters
  - **complete link**: ***maximum*** of the distances between all pairs of patterns drawn from the two clusters
  - **average link**: average of the distances between all pairs of patterns drawn from the two clusters
    - UPGMA (Unweighted Pair Group Method with Arithmetic Mean)
- All use agglomerative clustering control structure

Agglomerative clustering

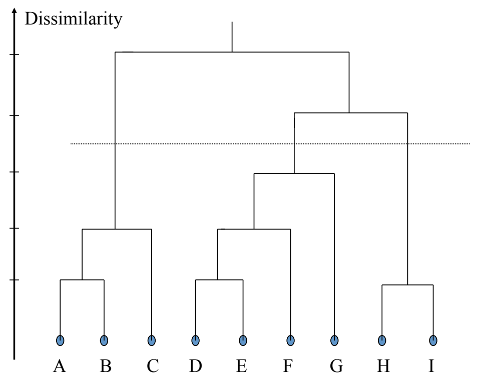
- Step 1: place each pattern in its own cluster

  construct a list of inter-pattern distances for all distinct unordered pairs of patterns, and sort this list in ascending order
- Step through the sorted list of distances, forming for each distinct dissimilarity value d_k, a graph on the patterns where pairs of patterns closer than d_k are connected by a graph edge.

  If all patterns are members of a completely connected graph, stop.

Dendrogram

- **A *Dendrogram* shows how the clusters are merged hierarchically:**
  - Decompose data objects into several levels of nested partitioning (tree of clusters), called a dendrogram.
  - A clustering of the data objects is obtained by cutting the dendrogram at the desired level, then each connected component forms a cluster.



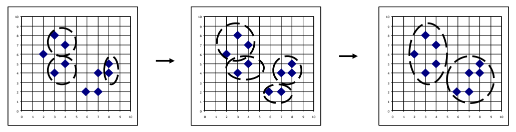
Practice Question

- Cluster the following six objects, using single-link and complete link agglomerative clustering methods:

	Gender	Age	Time	Fever	Cough
Obj1	F	2	2	Y	N
Obj2	M	2	0.5	N	N
Obj3	F	15	3	Y	Y
Obj4	F	18	0.5	Y	N
Obj5	M	58	4	N	Y
Obj6	F	44	14	N	Y

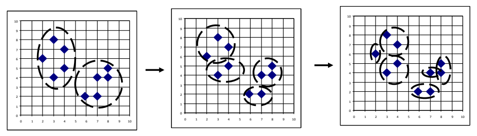
AGNES (Agglomerative Nesting)

- Implemented in statistical analysis packages, e.g., Splus
- Use the Single-Link method and the dissimilarity matrix.
- Merge nodes that have the least dissimilarity
- Go on in a non-descending fashion
- Eventually all nodes belong to the same cluster



DIANA (Divisive Analysis)

- Inverse order of AGNES
- Eventually each node forms a cluster on its own



More on Hierarchical Clustering Methods

- **Major weakness of agglomerative clustering methods**
  - do not scale well: time complexity of at least $O(n^2)$, where n is the number of total objects
  - can never undo what was done previously
- **Integration of hierarchical with distance-based clustering**
  - BIRCH (1996) (Balanced Iterative Reducing and Clustering using Hierarchies): uses CF-tree and incrementally adjusts the quality of sub-clusters
  - CURE (1998): selects well-scattered points from the cluster and then shrinks them towards the center of the cluster by a specified fraction
  - …

- Only works with "metric" attributes
  - *Must have Euclidean coordinates*
- Designed for very large data sets
  - *Time and memory constraints are explicit*
  - *Treats dense regions of data points as sub-clusters*
    - *Not all data points are important for clustering*
  - *Only one scan of data is necessary*

- Incremental, distance-based approach
  - *Does not need the whole data set in advance*
  - *Unique approach: distance based algorithms generally need all the data points to work*
- Does not assume that the probability distributions on attributes is independent

Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH)

- Incrementally construct a CF (Clustering Feature) tree, a hierarchical data structure for multiphase clustering
  - Phase 1: scan DB to build an initial in-memory CF tree (a multi-level compression of the data that tries to preserve the inherent clustering structure of the data)
  - Phase 2: use an arbitrary clustering algorithm to cluster the leaf nodes of the CF-tree
- *Scales linearly*: finds a good clustering with a single scan and improves the quality with a few additional scans
- *Weakness:* handles only numeric data, and sensitive to the order of the data record.

Clustering Feature Vector

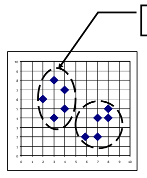
Clustering Feature: *CF = (N, $\overrightarrow{LS}$, SS)*

N: **Number of data points**

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

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

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



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

*Given a cluster of instances $\{\vec{X_i}\}$, we define the **centroid**, the **radius**, and the **diameter**:*

$$\vec{X}0 = \frac{\sum_{i=1}^{N}\vec{X_i}}{N}$$

$$R = \left(\frac{\sum_{i=1}^{N}(\vec{X_i}-\vec{X}0)^2}{N}\right)^{\frac{1}{2}}$$

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

*We define the **Euclidean** and **Manhattan** distance between any two clusters as:*

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

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

*We define the **average inter-cluster**, the **average intra-cluster**, and the **variance increase** distances as:*

$$D2 = \left(\frac{\sum_{i=1}^{N_1}\sum_{j=N_1+1}^{N_1+N_2}(\vec{X_i}-\vec{X_j})^2}{N_1N_2}\right)^{\frac{1}{2}}$$

$$D3 = \left(\frac{\sum_{i=1}^{N_1+N_2}\sum_{j=1}^{N_1+N_2}(\vec{X_i}-\vec{X_j})^2}{(N_1+N_2)(N_1+N_2-1)}\right)^{\frac{1}{2}}$$

$$D4 = \left(\sum_{k=1}^{N_1+N_2}\left(\vec{X_k}-\frac{\sum_{l=1}^{N_1+N_2}\vec{X_l}}{N_1+N_2}\right)^2 - \sum_{i=1}^{N_1}\left(\vec{X_i}-\frac{\sum_{l=1}^{N_1}\vec{X_l}}{N_1}\right)^2 - \sum_{j=N_1+1}^{N_1+N_2}\left(\vec{X_j}-\frac{\sum_{l=N_1+1}^{N_1+N_2}\vec{X_l}}{N_2}\right)^2\right)^{\frac{1}{2}}$$

The algorithm: CF

*A **Clustering Feature** (CF) summarizes a sub-cluster of data points.*

Given a cluster $\{\vec{X_1},\vec{X_2},\ldots,\vec{X_N}\}$

$\mathbf{CF} = (N, \vec{LS}, SS)$

N is the number of data points

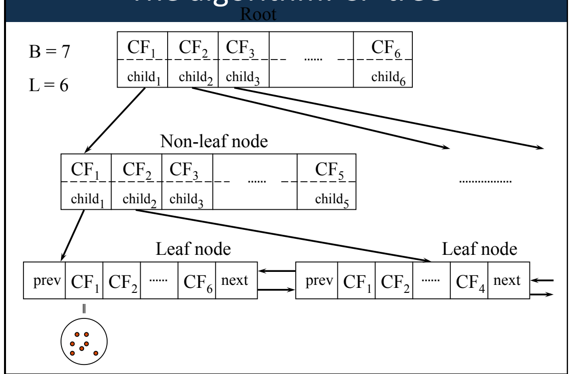
$\vec{LS} = \sum_{i=1}^{N}\vec{X_i}$

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

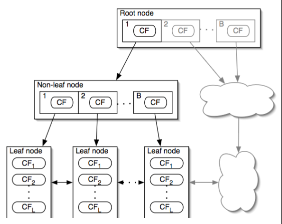
$\mathbf{CF_1} + \mathbf{CF_2} = (N_1+N_2, \vec{LS}_1+\vec{LS}_2, SS_1+SS_2)$

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

The algorithm: CF-tree


Root
B = 7
L = 6
CF$_1$ CF$_2$ CF$_3$ ...... CF$_6$
child$_1$ child$_2$ child$_3$ child$_6$
Non-leaf node
CF$_1$ CF$_2$ CF$_3$ ...... CF$_5$
child$_1$ child$_2$ child$_3$ child$_5$
Leaf node
prev CF$_1$ CF$_2$ ...... CF$_6$ next
Leaf node
prev CF$_1$ CF$_2$ ...... CF$_4$ next

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



- Recurse down from root
  - *Choose the "closest" CF and go to that node*
- Modify the leaf
  - *If the closest CF in the leaf can not absorb, make a new CF entry. If there is no room, split the node*
- Traverse back up
  - *Modifying CFs or splitting nodes*

- If we run out of space, increase 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 as small or smaller then the original*

- Phase 1: Load data into memory
  - *Build a CF-tree with the data*
- Phase 2: Condense data
  - *Rebuild the CF-tree with a larger T*
  - *Condensing is optional*

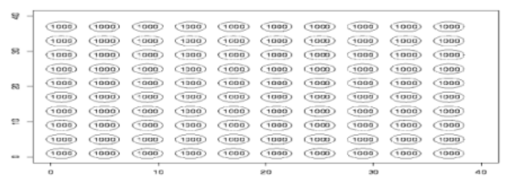
- Phase 3: Global clustering
  - *Use existing clustering algorithm on CF entries*
  - *Helps fix problem where natural clusters span nodes*
- Phase 4: Cluster refining
  - *Do additional passes over the data set and reassign data points to the closest centroid from phase 3*
  - *Refining is optional*

- Why have optional phases?
  - *Phase 2 allows us to resize the data set so Phase 3 runs on an optimally sized data set*
  - *Phase 4 fixes a problem with CF-trees where some data points may be assigned to different leaf entries*
  - *Phase 4 will always converge to a minimum*
  - *Phase 4 allows us to discard outliers*

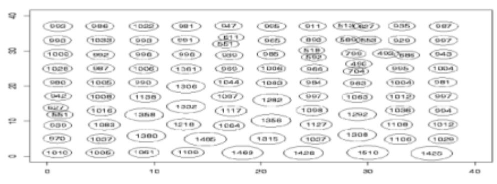
- Input parameters:
  - *Memory (M): 5% of data set*
  - *Disk space (R): 20% of* M
  - *Distance equation:* D2
  - *Quality equation: weighted average diameter (*D*)*
  - *Initial threshold (*T*): 0.0*
  - *Page size (*P*): 1024 bytes*

- Create 3 synthetic data sets for testing
  - *Also create an ordered copy for testing input order*
- KMEANS and CLARANS require entire data set to be in memory
  - *Initial scan is from disk, subsequent scans are in memory*

Intended clustering

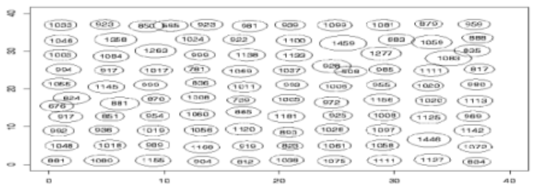


Kmeans Clustering



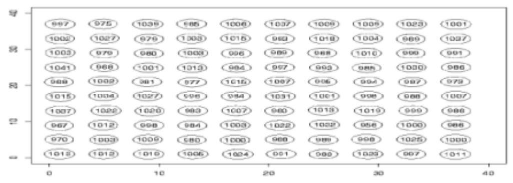
DS	Time	D	# Scan	DS	Time	D	# Scan
1	**43.9**	**2.09**	**289**	1o	33.8	1.97	197
2	13.2	4.43	51	2o	12.7	4.20	29
3	32.9	3.66	187	3o	36.0	4.35	241

CLARANS clustering



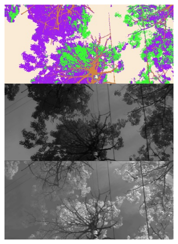
DS	*Time*	*D*	*# Scan*	*DS*	*Time*	*D*	*# Scan*
1	**932**	**2.10**	**3307**	1o	794	2.11	2854
2	758	2.63	2661	2o	816	2.31	2933
3	835	3.39	2959	3o	924	3.28	3369

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

- Pixel classification in images
- From top to bottom:
  - *BIRCH classification*
  - *Visible wavelength band*
  - *Near-infrared band*



- BIRCH works with very large data sets
- BIRCH performs faster then CLARANS or LBG, while getting better compression and nearly as good quality
- Explicitly bounded by computational resources
  - *Runs with specified amount of memory* (P)
- Superior to CLARANS and KMEANS
  - *Quality, speed, stability and scalability*