Chapter VII: Cluster analysis

Knowledge Discovery in Databases

Luciano Melodia M.A.
Evolutionary Data Management, Friedrich-Alexander University Erlangen-Nürnberg

Summer semester 2021

Chapter VII: Cluster analysis

Cluster analysis: basic concepts.

Partitioning methods.

Hierarchical methods.

Density-based methods.

Grid-based methods.

Evaluation of clustering.

Summary.

What is cluster analysis?

Cluster: A collection of data objects within a larger set that are.

- Similar (or related) to one another within the same group and,
- dissimilar (or unrelated) to the objects outside the group.

Cluster analysis (or clustering, data segmentation, . . .).

- Define similarities among data based on the characteristics found in the data (input from user!).
- Group similar data objects into clusters.

Unsupervised learning:

- No predefined classes.
- I.e., learning by observation (vs. learning by examples: supervised).

Typical applications:

- As a stand-alone tool to get insight into data distribution.
- As a preprocessing step for other algorithms.

Clustering for data understanding and applications

Biology:

Taxonomy of living things: kingdom, phylum, class, order, family, genus, and species.

Information retrieval:

Document clustering.

Land use:

Identification of areas of similar land use in an earth-observation database.

Marketing:

Help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs.

City planning:

Identifying groups of houses according to their house type, value, and geographical location.

Earthquake studies:

Observed earthquake epicenters should be clustered along continent faults.

Climate:

Understanding earth climate, find patterns of atmosphere and ocean.

Quality: what is good clustering?

A good clustering method will produce high-quality clusters.

- **High intra-class similarity:**
  - Cohesive within clusters.
- **Low inter-class similarity:**
  - Distinctive between clusters.

The quality of a clustering method depends on:

- the **similarity measure** used by the method,
- its implementation, and
- its ability to discover some or all of the hidden patterns.

Measure the quality of clustering

Dissimilarity/similarity metric:

Similarity is expressed in terms of a distance function, typically a metric: $d(x, y)$.

The definitions of distance functions are usually rather different for interval-scaled, Boolean, categorical, ordinal, ratio, and vector variables (see chapter 2).

Weights should be associated with different variables
based on applications and data semantics.

Quality of clustering:

There is usually a separate **"quality" function** that measures the "goodness" of a cluster.

It is hard to define "similar enough" or "good enough."

The answer is typically highly subjective.

Considerations for cluster analysis

Partitioning criteria:

Single level vs. hierarchical partitioning.
Often, multi-level hierarchical partitioning is desirable.

Separation of clusters:

Exclusive (e.g., one customer belongs to only one region) vs.
Non-exclusive (e.g., one document may belong to more than one class).

Similarity measure:

Distance-based (e.g., Euclidian, road network, vector) vs.
Connectivity-based (e.g., density or contiguity).

Clustering space:

Full space (often when low-dimensional) vs.
Subspaces (often in high-dimensional clustering).

Requirements and challenges

Scalability:

Clustering all the data instead of only on samples.

Ability to deal with different types of attributes:

Numerical, binary, categorical, ordinal, linked, and mixture of these.

Constraint-based clustering:

User may give inputs on constraints.
Use domain knowledge to determine input parameters.

Interpretability and usability.

Others:

Discovery of clusters with arbitrary shape.
Ability to deal with noisy data.
Incremental clustering and insensitivity to input order.
High dimensionality.

Major clustering approaches

Partitioning approach:

Construct various partitions and then evaluate them by some criterion.
E.g., minimizing the sum of square errors.
Typical methods: k-means, k-medoids, CLARA, CLARANS.

Hierarchical approach:

Create a hierarchical decomposition of the set of data (or objects) using some criterion.
Typical methods: AGNES, DIANA, BIRCH, CHAMELEON.

Density-based approach:

Based on connectivity and density functions.
Typical methods: DBSCAN, OPTICS, DENCLUE.

Grid-based approach:

Based on a multiple-level granularity structure.
Typical methods: STING, WaveCluster, CLIQUE.

Major clustering approaches (II)

Model-based approach:

A model is hypothesized for each of the clusters and tries to find the best fit of that model to each other.

Typical methods: EM, SOM, COBWEB.

Frequent-pattern-based approach:

Based on the analysis of frequent patterns.

Typical methods: p-Cluster.

User-guided or constraint-based approach:

Clustering by considering user-specified or application-specific constraints.

Typical methods: COD (obstacles), constrained clustering.

Link-based clustering:

Objects are often linked together in various ways.

Massive links can be used to cluster objects: SimRank, LinkClus.

Chapter VII: Cluster analysis

Cluster analysis: basic concepts.

Partitioning methods.

Hierarchical methods.

Density-based methods.

Grid-based methods.

Evaluation of clustering.

Summary.

Partitioning algorithms: basic concept

Partitioning method:

Partition a database D of n objects $o_j, j \in \{1, \ldots, n\}$ into a set of k-clusters C_i, $1 \leq i \leq k$ such that the sum of squared distances to c_i is minimized (where c_i is the **centroid** or **medoid** of cluster C_i):

$$\min \sum_{i=1}^{k} \sum_{o \in C_i} d(o, c_i)^2. \tag{1}$$

Given k, find a partition of k clusters that optimizes the chosen partitioning criterion.

Globally optimal: exhaustively enumerate all partitions.

Heuristic methods: k-means and k-medoids algorithms.

k-means (MacQueen’67, Lloyd’57/’82):

Each cluster is represented by the center of the cluster.

k-medoids or PAM (Partition around medoids) (Kaufman & Rousseeuw’87):

Each cluster is represented by one of the objects in the cluster.

The *k*-means clustering method

Given k, the k-means algorithm is implemented in four steps:

1. Partition the database into k non-empty subsets.

   E.g. the first $\frac{n}{k}$ objects, then the next $\frac{n}{k}$ objects, . . .

2. Compute the centroids of the **clusters** of the current partitioning.

   The centroid is the center, i.e. mean point, of the cluster.
   For each attribute (or dimension), calculate the average value.

3. Assign each object to the cluster with the nearest centroid.

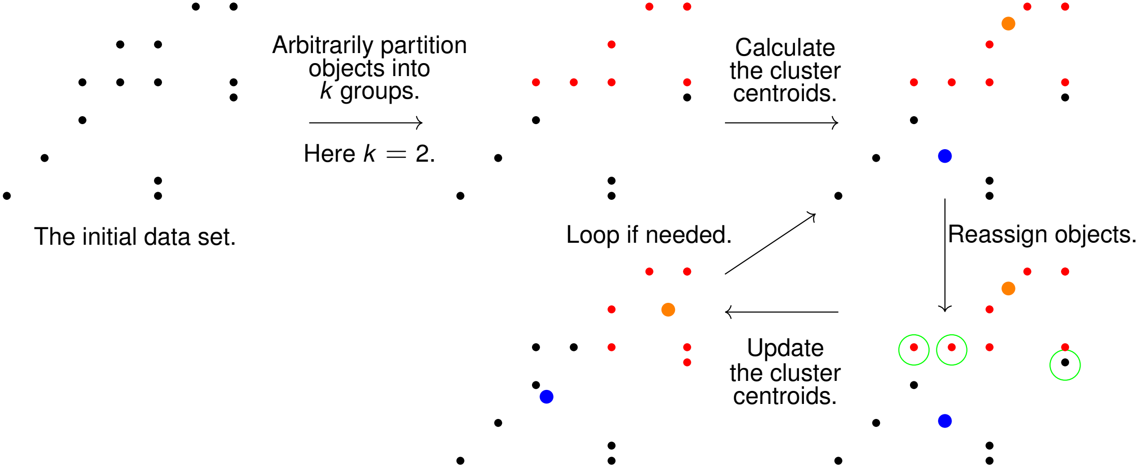
   That is, for each object calculate distance to each of the k centroids and pick the one with the smallest distance.

4. If any object has changed its cluster, go back to step 2. Otherwise stop.

Variant:

Start with arbitrarily chosen k objects as initial centroids in step 1.
Continue with step 3.

An example of *k*-means clustering



The initial data set.

Arbitrarily partition objects into *k* groups.

Here $k = 2$.

Calculate the cluster centroids.

Reassign objects.

Update the cluster centroids.

Loop if needed.

Comments on the *k*-means method

Strength:

Efficient: $\mathcal{O}(tkn)$, where n is $\#$ objects, k is $\#$ of clusters, and t is the $\#$ of iterations. Normally, $k, t \ll n$.

Comparing: PAM: $\mathcal{O}(k(n-k)^2)$, CLARA: $\mathcal{O}(ks^2 + k(n-k))$.

Comment: Often terminates at a local optimum.

Weakness:

Applicable only to objects in a continuous n-dimensional space.

- Using the *k*-modes method for categorical data.
- In comparison, *k*-medoids can be applied to a wide range of data.

Need to specify k, the number of clusters, in advance.

- There are ways to automatically determine the best k (see Hastie et al., 2009).

Sensitive to noisy data and outliers.

Not suitable to discover clusters with non-convex shapes.

Variations of the *k*-means method

Most of the variants of the *k*-means differ in:

- Selection of the initial *k* subsets (or centroids).
- Dissimilarity calculations.
- Strategies to calculate cluster centroids.

Handling categorical data: *k*-modes:

- Replacing centroids with modes.
  - See Chapter 2: mode = value that occurs most frequently in the data.
- Using new dissimilarity measures to deal with categorical objects.
- Using a frequency-based method to update modes of clusters.
- A mixture of categorical and numerical data: k-prototype method.

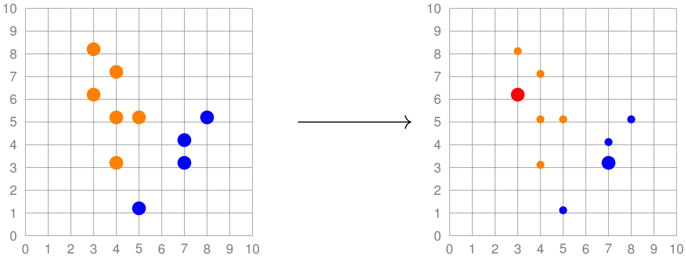
What is the problem of the *k*-means method?

The *k*-means algorithm is sensitive to outliers!

Since an object with an extremely large value may substantially distort the distribution of the data.

***k*-medoids:**

Instead of taking the mean value of the objects in a cluster as a reference point, medoids can be used, which is the most centrally located object in a cluster.



The *k*-medoids clustering method

***k*-medoids clustering:**

Find representative objects (medoids) in clusters.

PAM (Partitioning Around Medoids, Kaufmann & Rousseeuw, 1987):

Starts from an initial set of *k* medoids and iteratively replaces one of the medoids by one of the non-medoids, if it improves the total distance of the resulting clustering.

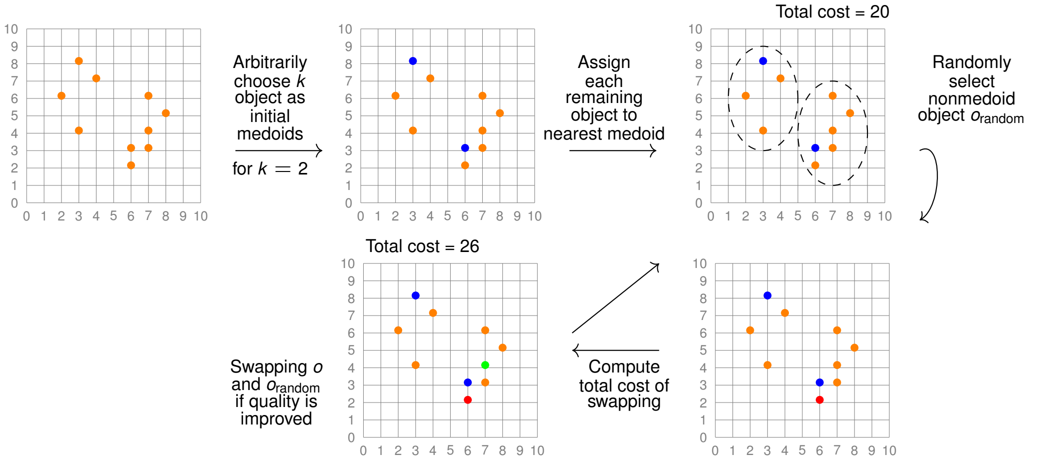
PAM works effectively for small data sets, but does not scale well for large data sets (due to the computational complexity).

Efficiency improvement on PAM:

CLARA (Kaufmann & Rousseeuw, 1990): PAM on samples.

CLARANS (Ng & Han, 1994): Randomized re-sampling.

PAM: A typical k-medoids algorithm



Arbitrarily choose k object as initial medoids

for $k = 2$

Assign each remaining object to nearest medoid

Total cost = 20

Randomly select nonmedoid object o_{random}

Compute total cost of swapping

Total cost = 26

Swapping o and o_{random} if quality is improved

PAM (Partitioning Around Medoids)

Use real objects to represent the clusters.

Algorithm:

```
1. Arbitrarily choose $k$ objects as the initial mediods.
2. Repeat.
3.    Assign each remaining object to the cluster with the nearest mediod $o_i$.
4.    Randomly select a non-medoid object $o_h$.
5.    Compute the total cost $\mathrm{TC}_{ih}$ of swapping $o_i$ with $o_h$.
6.    If $\mathrm{TC}_{ih} < 0$ then swap $o_i$ with $o_h$ to form the new set of $k$ medoids.
7. Until no change.
```

$$\mathrm{TC}_{ih} = \sum_j C_{jih}$$

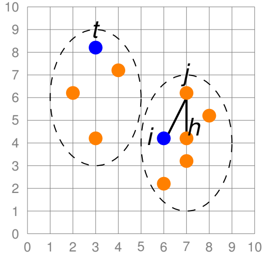
with C_{jih} as the cost for object o_j if o_i is swapped with o_h.
That is, distance to new medoid minus distance to old medoid.

PAM Clustering (II)

Case 1:

o_j currently belongs to medoid o_i. If o_i is replaced with o_h as a medoid, and o_j is closest to o_h, then o_j is reassigned to o_h (same cluster, different distance).

$$C_{jih} = d(o_j, o_h) - d(o_j, o_i). \tag{2}$$

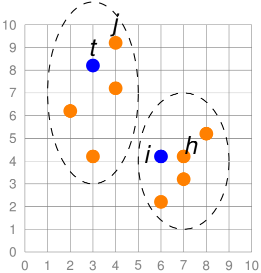


PAM Clustering (III)

Case 2:

o_j currently belongs to medoid o_t, $t \neq j$. If o_i is replaced with o_h as a medoid, and o_j is still closest to o_t, then the assignment does not change.

$$C_{jih} = d(o_j, o_h) - d(o_j, o_i) = 0. \tag{3}$$

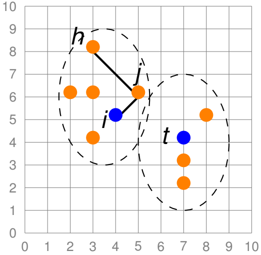


PAM Clustering (IV)

Case 3:

o_j currently belongs to medoid o_i. If o_i is replaced with o_h as a medoid, and o_j is closest to medoid o_t of one of the other clusters, then o_j is reassigned to o_t (new cluster, different distance).

$$C_{jih} = d(o_j, o_t) - d(o_j, o_i). \tag{4}$$

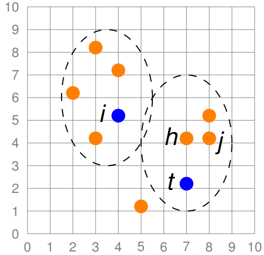


PAM Clustering (V)

Case 4:

o_j currently belongs to medoid o_t, $t \neq j$. If o_i is replaced with o_h as a medoid (from a different cluster!), and o_j is closest to o_h, then o_j is reassigned to o_h (new cluster, different distance).

$$C_{jih} = d(o_j, o_t) - d(o_j, o_i). \tag{5}$$



CLARA (Clustering Large Applications)

(Kaufmann and Rousseeuw, 1990)

Built in statistical-analysis packages, such as S+.

Draws multiple samples of the data set, applies PAM on each sample, and gives the best clustering as the output.

Strength:

Deals with larger data sets than PAM.

Weakness:

Efficiency depends on the sample size.

A good clustering based on samples will not necessarily represent a good clustering of the whole data set if the sample is biased.

CLARANS ("Randomized" CLARA)

A Clustering Algorithm based on Randomized Search (Ng & Han, 1994)

Samples:

Drawn dynamically with some randomness in each step of the search.

Clustering process:

Can be presented as searching a graph where each node is a potential solution, that is, a set of *k* medoids.

If local optimum found,

start with new randomly selected node in search for a new local optimum.

More efficient and scalable than both PAM and CLARA.

Focusing techniques and spatial access structures may further improve its performance. (Ester et al., 1995)

Chapter VII: Cluster analysis

Cluster analysis: basic concepts.

Partitioning methods.

Hierarchical methods.

Density-based methods.

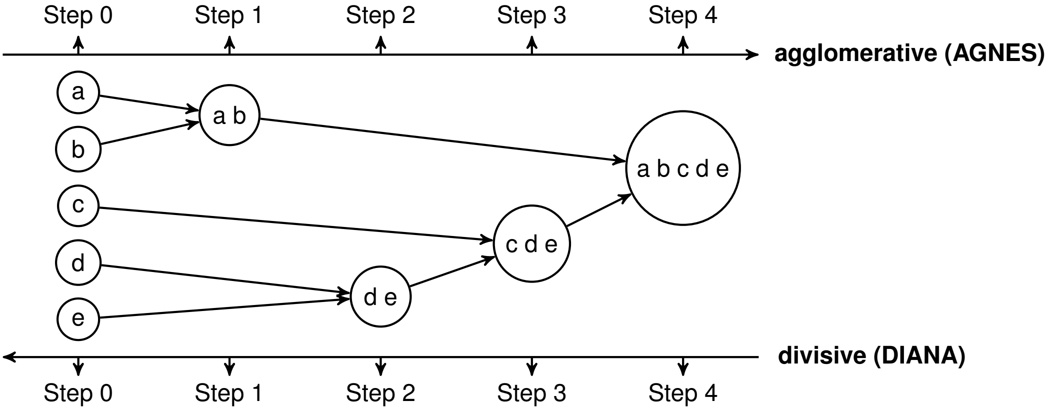
Grid-based methods.

Evaluation of clustering.

Summary.

Hierarchical clustering

Does not require the number of clusters *k* as an input, but needs a termination condition.



AGNES (Agglomerative Nesting)

Introduced in (Kaufmann & Rousseeuw, 1990)

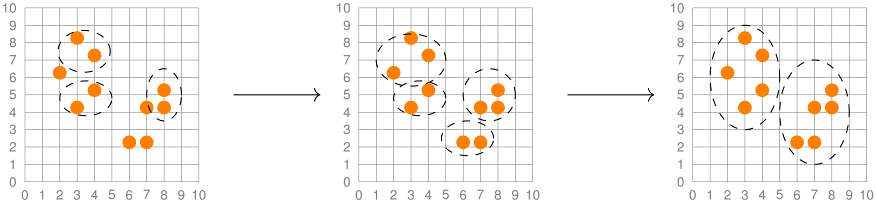
Implemented in statistical packages, e.g., S+.

Use the single-link method. (see below)

Merge nodes that have the least dissimilarity.

Go on in a non-descending fashion.

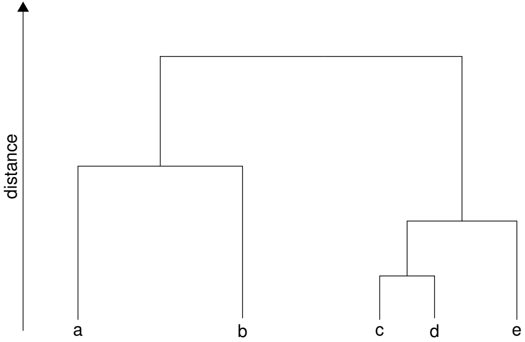
Eventually all nodes belong to the same cluster.



Dendrogram: shows how clusters are merged

Decompose data objects into a 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.



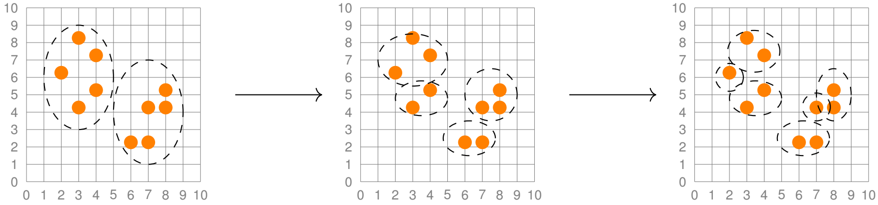
DIANA (Divisive Analysis)

Introduced in (Kaufmann & Rousseeuw, 1990)

Implemented in statistical analysis packages, e.g., S+.

Inverse order of AGNES.

Eventually each node forms a cluster of its own.



Distance between clusters

Minimum distance:

Smallest distance between an object in one cluster and an object in the other, i.e., $\text{dist}_{\text{min}}(C_i, C_j) = \text{min}_{o_{ip} \in C_i, o_{jq} \in C_j} d(o_{ip}, o_{jq})$.

Maximum distance:

Largest distance between an object in one cluster and an object in the other, i.e., $\text{dist}_{\text{max}}(C_i, C_j) = \text{max}_{o_{ip} \in C_i, o_{jq} \in C_j} d(o_{ip}, o_{jq})$.

Average distance:

Average distance between an object in one cluster and an object in the other, i.e., $\text{dist}_{\text{avg}}(C_i, C_j) = \frac{1}{n_i \cdot n_j} \sum_{o_{ip} \in C_i, o_{jq} \in C_j} d(o_{ip}, o_{jq})$.

Mean distance:

Distance between the centroids of two clusters, i.e., $\text{dist}_{\text{mean}}(C_i, C_j) = d(c_i, c_j)$.

Distance between clusters (II)

Nearest-neighbor clustering algorithm:

Uses **minimum distance** to measure distance between clusters.

Single-linkage algorithm:

Terminates if distance between nearest clusters exceeds user-defined threshold.

Minimal spanning-tree algorithm:

View objects (data points) as nodes of a graph.
Edges form a path between nodes in a cluster.
Merging of two clusters corresponds to adding an edge between the nearest pair of nodes.
Because edges linking clusters always go between distinct clusters,
resulting graph will be a tree.
Thus, agglomerative hierarchical clustering that uses minimum distance produces minimal spanning tree.

Distance between clusters (III)

Farthest-neighbor clustering algorithm:

Uses **maximum distance** to measure distance between clusters.

Complete-linkage algorithm:

Terminates if maximum distance between nearest clusters exceeds user-defined threshold.

Good if true clusters are rather compact and approx. equal in size.

Extensions to hierarchical clustering

Major weakness of agglomerative clustering methods:

- Can never undo what was done previously.
- Do not scale well: Time complexity of at least $\mathcal{O}(n^2)$, where n is the number of objects.

Integration of hierarchical and distance-based clustering:

- BIRCH (1996): Uses CF-tree and incrementally adjusts the quality of sub-clusters.
- CHAMELEON (1999): Hierarchical clustering using dynamic modeling.

BIRCH (Balanced Iterative Reducing and Clustering Using Hierarchies)

(Zhang, Ramakrishnan & Livny, SIGMOD'96)

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 numerical data, and sensitive to the order of the data records.

Clustering feature in BIRCH

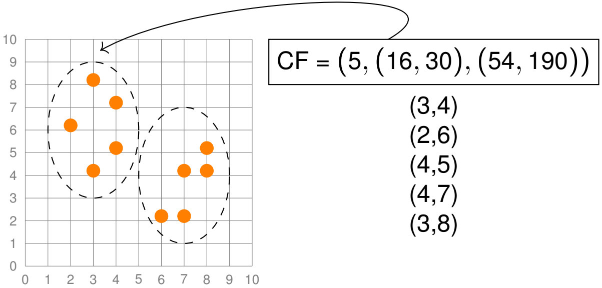
Clustering Feature CF = (n, LS, SS):

3D vector summarizing statistics about clusters.

n: number of data points.

LS: linear sum of *N* points $\sum_{i=1}^{n} x_i$.

SS: sqaure sum of *N* points $\sum_{i=1}^{n} x_i^2$.



$CF = \left(5, \left(16, 30\right), \left(54, 190\right)\right)$

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

Clustering feature in BIRCH (II)

Allows to derive many useful statistics of a cluster:

E.g. centroid: $c_i = \frac{\sum_{i=1}^{n} o_i}{n} = \frac{LS}{n}$.

Additive:

For two disjoint clusters C_1 and C_2 with clustering features $CF_1 = \left(n_1, LS_1, SS_1\right)$ and $CF_2 = \left(n_2, LS_2, SS_2\right)$, the clustering feature of the cluster that is formed by merging C_1 and C_2 is simply: $CF_1 + CF_2 = \left(n_1 + n_2, LS_1 + LS_2, SS_1 + SS_2\right)$.

CF-Tree in BIRCH

Height-balanced tree.

Stores the clustering features for a hierarchical clustering.

Non-leaf nodes store sums of the CFs of their children.

Two parameters:

Branching factor B: max # of children.

Threshold T: maximum diameter of sub-clusters stored at leaf nodes.

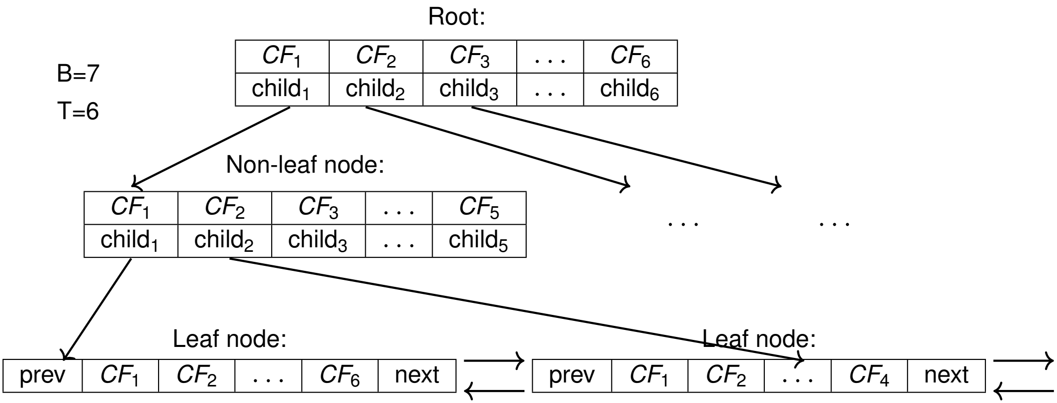
Diameter D: average pairwise distance within a cluster,
reflects the tightness of the cluster around the centroid:

$$D = \sqrt{\frac{\sum_{i=1}^{n} \sum_{j=1}^{n} (o_i - o_j)^2}{n(n-1)}} = \sqrt{\frac{2nSS - 2LS^2}{n(n-1)}}. \tag{6}$$

CF-Tree structure

B=7

T=6



Root:

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

Thank you for your attention.

Any questions about the seventh chapter?

Ask them now, or again, drop me a line:
`luciano.melodia@fau.de`.