Clustering

Clustering is the distribution of a set of instances of examples into non-known groups according to some common relations or affinities. This means that clustering algorithms allow for **classification** of items into groups which are not known at the beginning, **cluster**, according to some relations or affinities between the items themselves. The proper definition is:

> Given a set of instances *I*, a number of cluster *K*, an objective function *cost(I)*, a **clustering algorithm** computes an assignement of a cluster for each instance **f : I -> {1, ..., K}** that minimizes the objective function *cost(I)*.
>
> Given a set of instances *I*, a number of cluster *K*, an objective function *cost(C,I)*, a **clustering algorithm** computes a set *C* of instances with $|C|=K$ that minimizes the objective function [insert-formula-here] where *d(x,c)* is the distance function between *x* and *c*, and $d^2(x,C)$ is the distance from *x* to the nearest point in *C*.

Examples of clustering applications are the market segmentation of customers, social network communities analysis. Many different algorithms have been developed to solve clustering problems.

K-Means

The goal of the k-means algorithm is to minimize the intra-cluster distances, while maximizing the inter-cluster distances. Each cluster is identified by a *centroid*, a center point for the cluster. The algorithm follows an iterative procedure:

1. Choose k initial centers $C = \{c_1, ..., c_k\}$
2. while stopping criterion has not been met:
   - for $i = 1, ..., N$
     - find closest center $c_k \in C$ to each instance p_i
     - assign instance p_i to cluster C_k
   - for $k = 1, ..., K$
     - set c_k to be the center of mass of all points in C_i

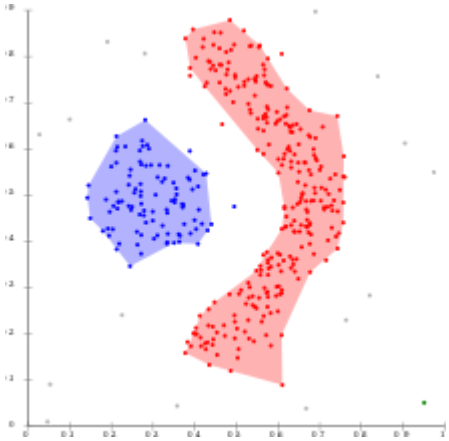
An improved version of the k-means algorithm is **k-means++**, which aims to avoid the sometimes poor clusterings found by the standard *k-means* algorithm by choosing the initial centers (after a random choice for the first one) with a certain probability $d^2(x,C)/cost(C,I)$.

Advantages/Disadvantages

K-means clustering algorithm is very fast to converge, and quite easy to use if there is a vague idea about the dataset to be analysed.

K-means is a parameter-based algorithm: the choice of the initial variables *k* and *epsilon* represent a huge factor in the success and the results of the algorithm. However, its speed allows multiple iterations with different parameters to be run in relatively short time. Moreover, it works best with spherical and well-defined clusters.

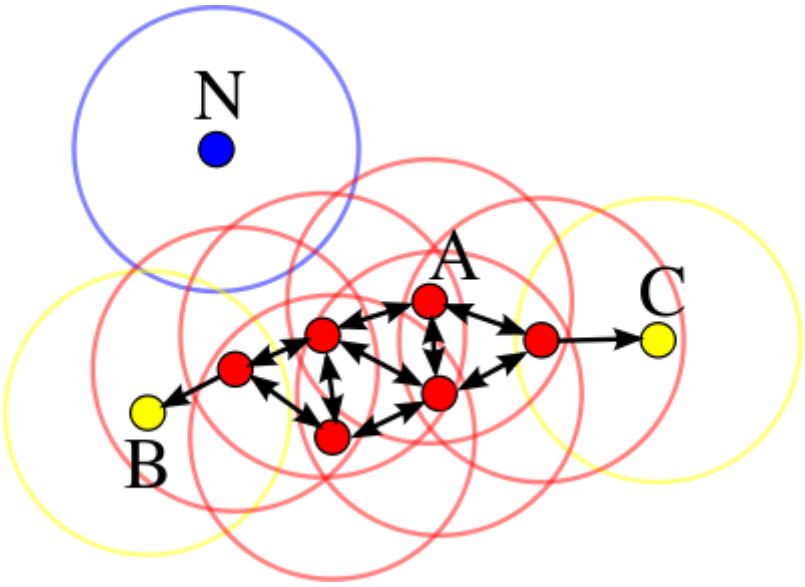
Density based methods



There are some dataset (such as the one in the figure above) for which k-means does not work properly, because it is very difficult to find centroids or there are concentric clusters or the cluster may not be linearly separable.

It is possible to define cluster in function of data points proximity, called **density-reachability**: having defined an *epsilon* maximum radius of neighborhood from a point and a *min_pts* number of points, every point belongs to one of these categories:

- a point *p* is a **core point** if there are more than **min_pts** in its epsilon-neighborhood - the points in this neighborhood are **directly density-reachable**
- a point *p* is **density-reachable** is there is a chain of points p_i where each point is directly density-reachable (all points in the chain are core points) - also known sometimes as **border points**
- a point *p* which is neither density-reachable nor a core point is a **noise point**.



The iterative process to build a cluster *C* becomes:

1. select an arbitrary point *p*
2. retrieve all points density-reachable from *p*
3. if *p* is a **core point**, a new cluster is formed; if it's a **border point**, the algorithm moves to another point from the database.
4. iterate until all points have been processed.

Advantages/Disadvantages

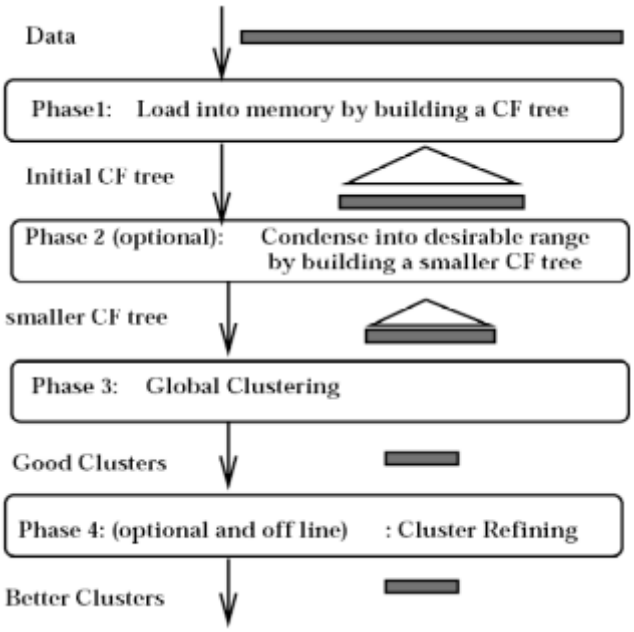
DBSCAN algorithm does not require an apriori definition of the number of clusters to fine (k in the k-means). Moreover, can find pretty easily arbitrarily shaped clusters, even ones that sorround but are not connected to other clusters (avoid the single-link effect, connection between clusters with a thin line of points, thanks to the *min_pts* parameter). DBSCAN is very robust to outliers and noise.

DBSCAN requires a definition of a *distance measure*, generally *euclidean distance*. The **curse of dimensionality**, which means having high-dimensional data, can make this metric pretty useless, just like with any other algorithm based on euclidean distance. DBSCAN is not appropriate when there is a huge difference in density between clusters, due to the *epsilon* threshold.

BIRCH - Balanced Iterative Reducing and Clustering using Hierarchies

BIRCH (balanced iterative reducing and clustering using hierarchies) is an unsupervised data mining algorithm used to perform hierarchical clustering over particularly large data-sets.

Previous clustering algorithms performed less effectively over very large databases and did not adequately consider the case wherein a data-set was too large to fit in main memory. As a result, there was a lot of overhead maintaining high clustering quality while minimizing the cost of addition IO (input/output) operations. Furthermore, most of BIRCH's predecessors inspect all data points (or all currently existing clusters) equally for each 'clustering decision' and do not perform heuristic weighting based on the distance between these data points.



Algorithm

BIRCH is composed mainly of two key phases:

- scans the database to build an **in-memory** tree, called the **CF-tree (clustering features tree)**
- applies clustering algorithm to cluster the leaf nodes

The **CF-tree** is built during the database scan, and it's made of **CF entries**, each representing a cluster of objects; these clusters are characterized by a 3-tuple **(N, LS, SS)**, where N is the number of multi-dimensional data points in the cluster, while LS (Linear Sum) and SS (Square Sum) are defined as follows:

$$LS = \sum_{P_i \in N} P_i$$

$$SS = \sum_{P_i \in N} |P_i|^2$$

A CF Entry is very compact with respect to storing all data points in the sub-cluster, and has sufficient information to calculate any distance measure (in particular, the ones in the following image). Moreover, the additivity theorem is applicable to CF entries, which means that merging sub-clusters is consistent.

centroid Euclidean distance: $D0 = ((\vec{X0}_1 - \vec{X0}_2)^2)^{\frac{1}{2}}$

centroid Manhattan distance: $D1 = |\vec{X0}_1 - \vec{X0}_2| = \sum_{i=1}^{d} |\vec{X0}_1^{(i)} - \vec{X0}_1^{(i)}|$

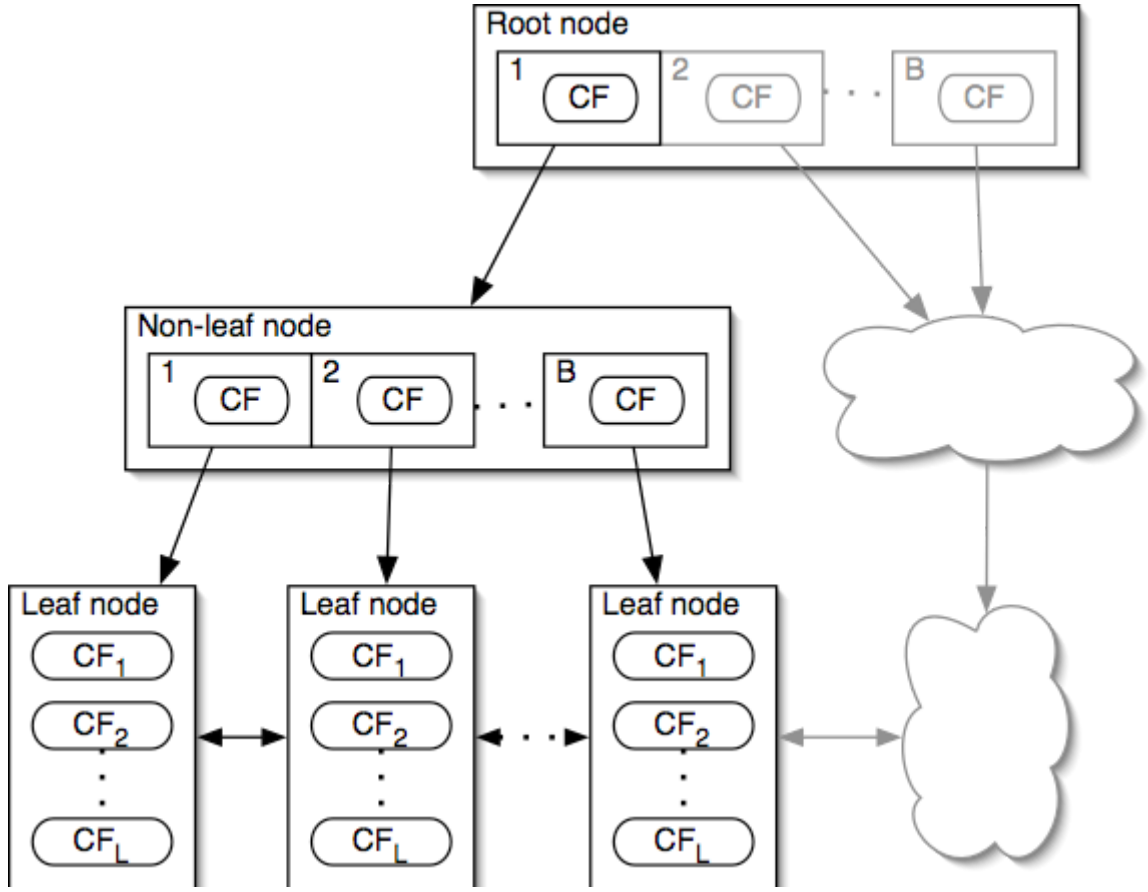
average inter-cluster: $D2 = (\frac{\sum_{i=1}^{N_1} \sum_{j=N_1+1}^{N_1+N_2} (\vec{X}_i - \vec{X}_j)^2}{N_1 N_2})^{\frac{1}{2}}$

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

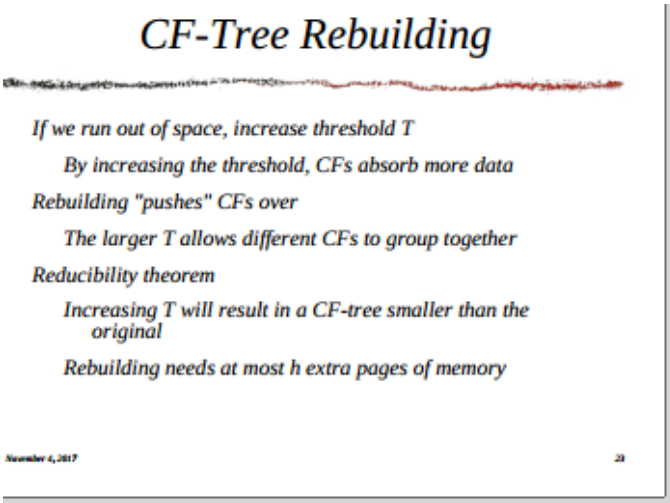
variance increase: $(\sum_{k=1}^{N_1+N_2} (\vec{X}_k - \frac{\sum_{l=1}^{N_1+N_2} \vec{X}_l}{N_1+N_2})^2 - \sum_{i=1}^{N_1} (\vec{X}_i - \frac{\sum_{l=1}^{N_1} \vec{X}_l}{N_1})^2 - \sum_{j=N_1+1}^{N_1+N_2} (\vec{X}_j - \frac{\sum_{l=N_1+1}^{N_1+N_2} \vec{X}_l}{N_2})^2)^{\frac{1}{2}}$

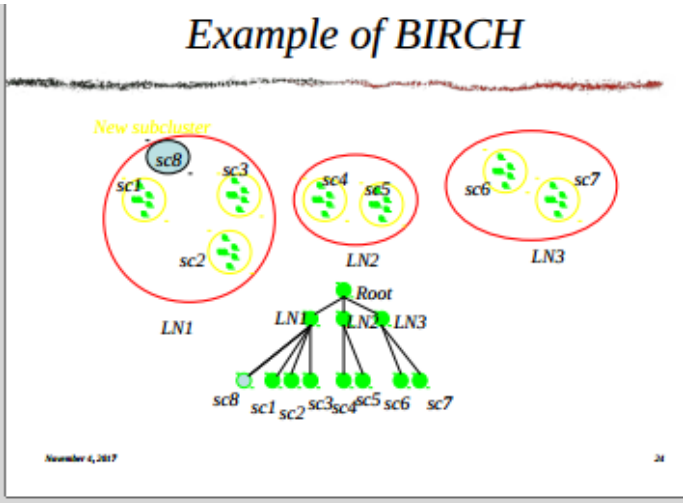
November 4, 2017 18

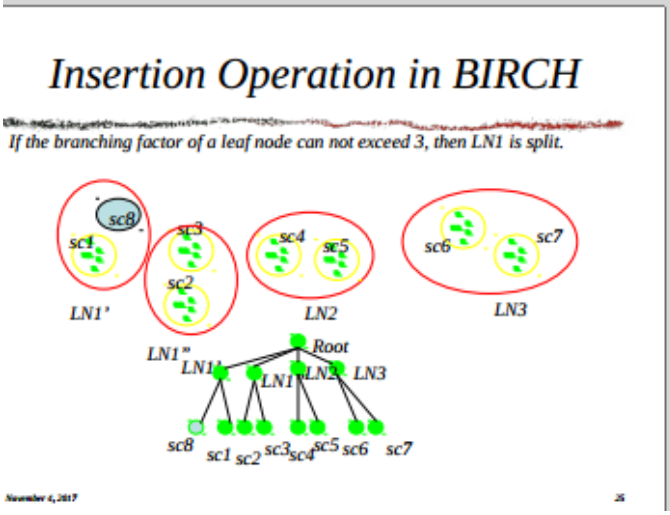
Clustering features are organized in a CF tree, a height-balanced tree with two parameters: branching factor *B* and threshold *T*. Each non-leaf node contains at most *B* entries CF_i. *Each leaf node has at most* L *entries, each of which satisfies threshold* T*.

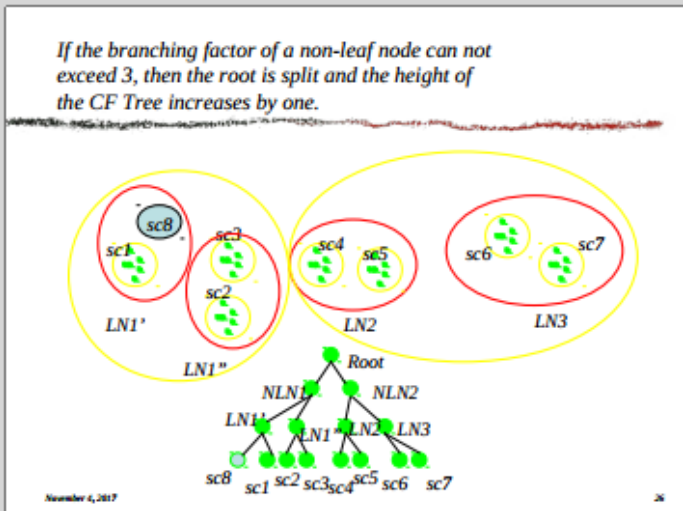


When a new CF entry should be **inserted** in the CF-tree, the tree is recursed down from root, to find the appropriate leaf, moving from leaf to leaf following the closest path with respect to the previously defined distances. Then, the appropriate leaf is modified to absorb the new CF-entry; if the limit L is reached, a new CF-entry is made; if there is no room for a new leaf the parent node is split. Then all CF entries on the path back to the root node are updated.









Advantages / Disadvantages

An advantage of BIRCH is its ability to incrementally and dynamically cluster incoming, multi-dimensional metric data points in an attempt to produce the best quality clustering for a given set of resources (memory and time constraints). In most cases, BIRCH only requires a single scan of the database. BIRCH is local in that each clustering decision is made without scanning all data points and currently existing clusters. It exploits the observation that data space is not usually uniformly occupied and not every data point is equally important. It makes full use of available memory to derive the finest possible sub-clusters while minimizing I/O costs. It is also an incremental method that does not require the whole data set in advance.

Since each node in a CF tree can hold only a limited number of entries due to the size, a CF tree node doesn’t always correspond to what a user may consider a nature cluster. Moreover, if the clusters are not spherical in shape, it doesn’t perform well because it uses the notion of radius or diameter to control the boundary of a cluster.